Controlling and Transforming vCards, hCards, iCalendars and hCalendars with the Help of ooRexx

Master Thesis
by
Johannes Paul Bielohaubek
h0541119

Thesis adviser: ao. Univ.-Prof. Dr. Rony G. Flatscher
Vienna University of Economics and Business Administration

Version 2010-05-28, 10:29:58 “Quijano”

Table of Contents

Index of Codes

Index of Figures

Index of Tables

1 Introduction

The purpose of this first chapter is to give information about this paper. It describes the problem with which will be dealt in later chapters. Moreover, it gives an overview about the structure of the paper itself.

1.1 Abstract (in English)

This paper deals with the problem how the vCard, iCalendar, hCard and hCalendar standard can be implemented in an ooRexx framework. The vCard and the iCalendar are standards for the representation of business cards and for calendars. hCard and hCalendar are the HTML/xhtml/XML microformat translation of the preliminarily mentioned standards. The resulting collection of classes uses ooRexx for the creation of class libraries in order to save and manipulate the data and Java via BSF4ooRexx in order to parse the XML structured data of the microformats and the GUI.

1.2 Abstrait (en français)

Ce mémoire de maitrise s'occupe d'une implémentation des standards ouverts de vCard, iCalendar, hCard et hCalendar dans le langage de programmation ooRexx. vCard traite la représentation informatique des donnés des cartes de visite. En outre, le standard iCalendar s'occupe de données des calendriers. Les standards restants sont des transpositions sous forme de HTML/xhtml et XML des standards mentionné d'abord. Ils se combinent sous le nom microformats (formats micros). Le programme crée pour ce mémoire de maitrise utilise ooRexx pour sauver et manipuler les données et il bénéficie de la librairie BSF4ooRexx pour joindre ooRexx avec Java qui possède les possibilités d'assimiler les données sous forme de formats micros et faire un environnement graphique.

1.3 About this Thesis

The world of today is quite a complex system of countless information flows. Businesses (and to a certain degree private persons as well) are expanding globally and leave their footprints everywhere. Tasks and chores which were done manually in the past sped up and became computerized.

Two persons who want to exchange information need a common language, a set of words which have the same meaning for the dialogue partners and rules of grammar which define how these words fit together. What is true for two individual human beings is also true for information which are exchanged with the help of computers. The vCard and iCalendar standards are such a “language”, a common standard which was developed in order to ease immensely the exchange of business cards and calendars in a fast and comfortable way. All of these exchanges are done in computer networks. The hugest of these networks is the Internet. First and foremost, the ordinary user thinks of web pages when talking about the Internet. These web pages can contain downloadable files which conform to the vCard and iCalendar standards. Nevertheless, most of the time, the information is embedded in the text elements of the home pages. Regrettably, the computer science is yet not advanced enough to understand texts without any problems. In order to give the machines a hand, “microformats” were developed which enrich the web pages with information about the displayed content. In order to give personal information or scheduling information, the hCard and hCalendar standards were put into existence. These two standards base keenly on the previously mentioned standards.

As a result, a program which can handle these standards would be quite handy. Such a program should be able to read the respective files, process their information and dispense it. What is more, such a program should be quite easy to use. Finally, such a piece of software should use a structure which could generate ideas for comparable programming problems. A programming language which could help this purpose is ooRexx which has the capability to draft and carry such designs without losing itself in the depth of a complicated syntax.

The result of the programming efforts of this thesis is a class library which replicates the iCalendar, hCalendar, vCard and hCard standards with their characteristics and enables the transformation of one calendar/business card standard into its counterpart.

This thesis is structured the following way: First, the thesis deals with the theoretic background of the topic. The aim of this section is not to describe everything all-encompassing in every single detail and aspect. It solely gives an overall idea.

A good deal more, the second part of this paper tries to give the reader an oversight about the programming languages which are used for the realization of the above described program. Besides, the various standards are explained in a way which explains the programmed realization. This section shows especially the author's decisions regarding the hCalendar and hCard standards which are not as comprehensive as the other standards.

What is more, the last sector of the main part is dedicated to the program itself. This sector presents how the concepts are realized and presents a bird's eye of view on the program. Apart from this, this sector goes into some prominent features of the program parts.

The final part is the appendix which harbors the complete source code of the program and further explanations, hints and links. It also gives an overview over supportive resources which helped during the completion of this thesis. This last part also contains the documentation of the created class library.

2 Theoretical Background

The following part of this paper explores the theoretical background and settings of the software and the implemented standards that make up the foundation of the classes representing the properties of the standards which were created in the course of this Master Thesis.

2.1 Software

The following chapters describe the software packages that were used for the completion of the underlying thesis. Further detailed information can be retrieved from the sources which are indicated. All of these software packages have in common that they are freely available. What is more, every piece of software is not only restricted to one operating system.

2.1.1 ooRexx

The classes which were created for this thesis were written in the programming language ooRexx. Thus, a short introduction of this language is necessary in order to get access to the topic.

The abbreviation ooRexx stands for “Open Object Rexx” and it already describes some of the characteristics of this programming language. First, the ooRexx programming language is published under the Common Public License (CPL) v1.0. Second, ooRexx is an object oriented descendant of the classical REXX language. Like classic REXX[1], ooRexx embraces an easily learnable and human centric syntax. The structure of the ooRexx language follows the logic of the (natural) English language. Moreover, ooRexx is an interpreted language i.e. the program code does not need to be compiled given that a program is newly created or changes have been effected. What is more, every variable is considered to be typeless in ooRexx. Finally, the ooRexx language provides a capacious and clear support in order to find programming errors quickly.[W3RX]

The current ooRexx edition which is used for the program created in the course of this thesis is the release 4.0.0. Only this and later versions consist of the new

[1] REXX stands for “restructured extended executor”.

kernel that is needed for the interaction of ooRexx, BSF4ooRexx and Java for the DOM-parsing activities which are described below.

The standard file extension for ooRexx and (classic) Rexx files is `.rex`.

The installation files and instructions can be found on the following homepage [W3RD] and is available for many different operating systems like Linux and Windows. Moreover, an introduction to procedural and object oriented programming with ooRexx is also available on the Internet.[2]

The underlying classes which represent the properties of the standards and the classes for reading the data use ooRexx as its backbone programming language.

2.1.2 BSF4ooRexx

As per February 2010, ooRexx only possesses a SAX[3] parser. However, the requirement to parse XML and HTML/xhtml files following the document object model (DOM) made it necessary to use BSF4ooRexx in order to build a bridge from ooRexx to Java and its DOM-parsing possibilities. Parsing XML and HTML/xhtml files was necessary as the hCard and hCalendar formats are structured this way.

The letters “BSF” stand for Bean Scripting Framework. The original code was created by IBM and is administrated by the Jakarta project of the Apache Software Foundation now. Generally speaking, the Bean Scripting Framework allows to combine a scripting language with Java. This means that methods or Java objects can be accessed through these languages. Besides, the framework allows any Java application to be implemented within the scripting languages. Some of the languages that are already supported are for example JavaScript, Python, Tcl, XSLT Stylesheets or NetRexx.[W3BSF]

ooRexx uses its own BSF engine: BSF4rexx (respectively BSF4ooRexx) The difference between BSF4ooRexx and BSF4rexx lies basically in the required version of Open Object Rexx. The new version of ooRexx (version 4.0) allowed

[2] For further information please refer e.g. to [W3AW].
[3] SAX stands for Simple API for XML.

the BSF4ooRexx engine to overcome shortcomings of the elder versions[4]. This is achieved through a new kernel in ooRexx. Moreover, the BSF4ooRexx engine only operates with this new version. Thus, a change in the name in order to show explicitly the difference was necessary. Some of the new features of the BSF4ooRexx are the creation of Rexx-Proxies for Java interfaces or the implementation of abstract Java methods in ooRexx.[Flat09]

As a convention, there are two different types of file name extensions concerning Rexx-files which require the BSF-support: There is the `.rxj`-suffix which is generally used for BSF-supported Rexx-files. However, if the file in question supports processes in OpenOffice.org the suggested file name extension is named `.rxo` by convention.

Due to the particular needs of the classes for reading and the classes which represent the properties of every individual standard, only the BSF4ooRexx engine can be used. The current version of this framework can be found at [W3BSR]. The required archive is called `BSF4ooRexx_install.zip` and it contains the needed codes including an installation manual. A short introduction to this topic can be found in the Internet.[5]

The class `Reader`[6] and the classes representing the properties[7] use BSF4ooRexx for using the default DOM parser of Java. Moreover, a graphical user interface for using these classes is created.[8]

2.1.3 Java

Java is probably one of the most known programming languages. According to Sun Microsystems, the developer of this programming language, Java is an object oriented, portable i.e. it does not depend on any specific architecture, processor, operating system etc., robust and secure programming language. [W3JAVA]

---

4 Cf. [Flat09] fur further information.
5 For further information refer e.g. to [W3AJ].
6 Cf. chapter “3.2 Reading and Preparation of the Data“ (p. 30 ff.).
7 Cf. chapter “3.3 The vCard/iCalendar Properties in ooRexx“ (p. 34 ff.).
8 Cf. chapter “3.4.2 Graphical User Interface“ (p. 45 ff.) and chapter “3.2 Reading and Preparation of the Data“ (p. 30 ff.).

In comparison to ooRexx, Java is not interpreted. As a result, Java needs to be compiled before executing the code. Moreover, Java is very strict with regard to data types and type declarations. In addition to that, Java is case-sensitive compared to ooRexx.

The advantage of the Java programming language is its widespread distribution, its system independence and its tremendous number of packages. BSF4ooRexx allows to use all of these libraries like they were written for ooRexx using the programming logic of ooRexx.

The regular file name extension for Java programs are `.java` for the source files, `.class` for their compiled counterparts and `.jar` (i.e. Java Archive) for Java files that are embedded in a ZIP-archive

The version of Java which was used for the realization of the underlying programs was Version 6 Update 18 which can be found in the Internet.[W3JD] An introductory tutorial can be found in the Internet[9]. However, Java is already installed on nearly every computer.

Java provides the classes for reading the data and the classes which represent the properties of the standards with a DOM parser[10]. Moreover, it was needed for the GUI[11].

[9] For further information refer e.g. to [W3JTU].
[10] Cf. chapter “3.2 Reading and Preparation of the Data“ (p. 30 ff.).
[11] Cf. chapter “3.4.2 Graphical User Interface“ (p. 45 ff.).

2.2 Implemented and Used Standards

The consecutive part of this paper gives a general overview over the specifications of the implemented standards in this work: vCard, iCalendar, hCard, hCalendar A detailed and all-encompassing description can be found in the respective standard's definitions.

2.2.1 vCard

The aim of the vCard specification is to provide a file format that allows to exchange information concerning persons, businesses or organizations in a system independent and human readable way.

The original vCard standard was defined 1996 by the “versit consortium”. This consortium consisted of representatives of influential companies in the computer industry like e.g. Microsoft, IBM or Apple.[W3VC]

Two years later, in 1998, the vCard standard was published in the RFCs 2425 and 2426.[12] As per February 2010, the IETF is reviewing the vCard standard. [W3NC]

Code 1 shows a simple vCard corresponding to the specifications in RFC 2425 and 2426.

```
begin:vcard
NOTE:Hello/; this is a sample business card.
fn:Johannes Paul Bielohaubek
n:Bielohaubek;Johannes;Paul
version:3.0
EMAIL;TYPE=internet,pref:johannes.paul@wu.ac.at
ORG:Wirtschaftsuniversitaet Wien;Institute for
  Management Information Systems
end:vcard
```

Code 1: Sample vCard.

This simple example illustrates the major points of a vCard. The whole vCard lies embedded between two text lines indicating the begin and the end of a vCard object (`begin:vcard` and `end:vcard`).

Each new line represents a new, single property of the vCard. A property is a specification of a business card like a name or a phone number. The total

[12] RFC stands for Request for Comments, i.e. an official document by the Internet Engineering Task Force, IETF, for Internet systems and standards.

length of each line is limited to 75 characters.[RFC2425] As a result, properties which are longer than this limit must be folded. Folding means that a new line is started. In order to indicate the folding, the new line begins with a white space character (i.e. a space or a horizontal tabulator). This white space character will be removed when the vCard is read and unfolded.[RFC2426] In the example, this is illustrated by a `note`-property.

Each property (or unfolded line) consists of two parts: The first part contains the property name and its parameters. This part is case-insensitive. This means that there is no difference between e.g. `NOTE` or `note` or `noTE`.

The second part consists of the properties' values. These two parts are separated by a colon (`:`). Furthermore, these two halves are broken into smaller pieces by a semi-colon (`;`). In the example above, the name-attribute part optionally starts with a grouping and a point as separator. Then, the property name is added. The next element are parameters which gives details about the property. In the sample, the `email`-property is defined as an Internet and preferred[13] e-mail address. This attribute `TYPE=internet,pref` is separated by a comma (`,`) which indicates a list of values belonging to the same parameter or field-value. Alternatively, this could have been also written `TYPE=internet;TYPE=pref`.[14] Due to the importance of the separating characters (colon, semi-colon, comma), they must be backslash escaped whenever they are used for a text and not as a separator.[RFC2426]

The vCard standard contains of a long list of properties in order to describe the different aspects of a person (or organization). Nevertheless, there is a number of properties which are mandatory for every single vCard in order to make them functional. Obviously, every vCard needs a `begin`- as well as an `end`-property. Additionally, the `version`-property is required in order to distinguish between the different vCard variants. Two other properties that are mandatory are the `fn`- and `n`-property. Both represent a certain way to indicate a name. The `n`-property records a name in a special order whereas the `fn`-property accepts any order. [RFC2426]

---

13 The vCard (and iCalendar) always uses the abbreviation “pref” for preferred.
14 Please note the semi-colon.

Other properties that are also defined in the RFC 2426 are predefined types (`name`, allows to give the vCard a name, `profile`, `source`), identification types (`nickname`, `photo`, `bday`), delivery address types (`adr` and `label`), telecommunications addressing types (`tel`, `email`, `mailer`), geographical types (`tz`, `geo`), organizational types (`title`, `role`, `logo`, `agent`, `org`), explanatory types (`categories`, `note`, `prodid`, `rev`, `sort-string`, `sound`, `uid`, `url`) and security types (`class` and `key`).[RFC2426]

The properties with a special order for their values are the `fn`-property with a sequence of `family-name`, `given-name`, `additional-name`, `honorific-prefix`, honorific-suffix, the `adr`-property with a sequence of the `post-office-box`, the `extended-address`, the `street-address`, the `locality`, the `region`, the `postal-code` and the `country-name`, the `geo`-property with `latitude` and `longitude` and finally the `org`-property with `organization-name` and `organization-unit`. [RFC2426]

One vCard property that could cause problems in the further processing of the data is the `agent`-property. As previously mentioned, every comma, semi-colon and colon must be backslash escaped. In the case of the `agent`-property, a complete other vCard can be encoded as the value of the `agent`-property. As this is seen as a text element, every comma, semi-colon and colon must be escaped. This causes a problem in the differentiation of the escaped signs when working with the vCard within as it is difficult to determine whether or not they must continue staying escaped or not.

In addition to the standard properties, the RFC 2426 also allows the user to define their own properties and/or parameters. Formally, these properties must meet the above mentioned rules concerning structure and separators. Apart from this, the non-standard elements must start with the prefix “`X-`” or “`x-`”. This rule allows to add software specific or bilaterally agreed components to the standard.[RFC2426]

The RFC 4770 adds the `impp`[15]-property to the already existing ones. This property is also considered in the structure of the vCards.[RFC4770] Finally, the common file name extensions for the vCard are `.vcf` or `.vcard`.

[15] IM stands for “instant messaging” and PP for “presence protocol”.

2.2.2 iCalendar

The history of and the conception behind the iCalendar and the vCard are quite the same. Like the vCard, the standard was formulated in 1996 as vCalendar by the versit Consortium. However, in 1998 it was renamed to iCalendar and redefined in the RFCs 2445, 2446 and 2447. Eleven years later, 2009, the RFC 2445 was replaced by the RFC 5545 and the RFC 2446 by RFC 5546.

Here is an example corresponding to the current iCalendar standard:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VTODO
UID:20100206T150300Z-AF23B2@example.com
DTSTAMP:20100206T150300Z
DUE;VALUE=DATE:20100227
SUMMARY:Write thesis
STATUS:NEEDS-ACTION
END:VTODO
BEGIN:VEVENT
UID:20100206T132900Z-AF23B2@example.com
DTSTAMP:20100203T133000Z
DTSTART:20100228T090000Z
DTEND:20100228T180000Z
SUMMARY:Graduation party
END:VEVENT
END:VCALENDAR
```

Code 2: Sample iCalendar.

The structure of an iCalendar is equal to a vCard as the comparison of code 1 and 2 perfectly well illustrates. As a consequence, information about the folding rules, the separators, the case-insensitivity and property build-up can be found in the chapter for the vCard[16].

However, there are some obvious particularities of this standard which need to be addressed or mentioned.

Despite the fact that the standard is called “iCalendar”, it still uses the name of the original specification of 1996[17]. Similar to the vCard standard, each iCalendar must have a `begin`- and an `end`-property as well as the `version`- and `prodid`-property. (The `prodid` is a unique identifier of the software that created the iCalendar.)[RFC5545]

[16] Cf. chapter ”2.2.1 vCard“ (p. 8 ff.).
[17] The calendar is still called `vcalendar` within an iCalendar file.

One special feature of the iCalendar compared to the vCard are the iCalendar components. These are objects within an iCalendar with a special purpose. There is a total of five regular components. Each of them represents the grouping of properties which describe an event, a task etc. As a consequence, each of these components starts with the `begin`-property (and the component name) and closes with the `end`-property.[RFC5545]

The first component is shown in the example in code 2. The `vevent` allows to specify one event. Each `vevent` must have an `uid`-property[18] as well as a `dtstamp`[19]. Besides, there is a huge range of other properties possible. Moreover, a `vevent` could contain another component, the `valarm` which is described later.[RFC5545]

The second component is called `vtodo`. This component gives information about a to-do task. Like the `vevent`, it must have one `uid`-property and a `dtstamp`. Again, a `valarm` could also be placed in a `vtodo` component. [RFC5545]

The following component is the already mentioned `valarm` which contains a grouping of properties that describes what kind of alert will be invoked and when this event is going to occur. It must hold an `action`-property (describes which kind of activity is performed) as well as a `trigger`-property (describes when the alarm will take place).[RFC5545]

The next iCalendar component is the `vjournal` component. This component is dedicated to the description of a singular calendar date. A possible example would be a list of achievements for a particular day. Like for the previous components, the `uid`-property as well as the `dtstamp` are mandatory.[RFC5545]

What is more, an iCalendar could also have a `vfreebusy`-component. The purpose of this component is to specify time periods in which a person, organization is available or not. Again, a `uid` and a `dtstamp` are mandatory. [RFC5545]

[18] The `uid`-property is a unique identifier.
[19] The `dtstamp`-property indicates when the component was created.

The last regular component is the `vtimezone` which is used to specify time zones, the daylight saving time and the normal time. This component must possess a `tzid`-property[20].[RFC5545]

Finally, the common file name extension for the iCalendar are `.ical`, `.ics`, `.ifb` or `.icalendar`.[RFC5545]

2.2.3 hCard

The hCard standard is a microformat. The general goal of a microformat is to provide an open data format that allows a better and more structured blogging and the publishing of web micro content.[W3MI] Focusing on the hCard microformat, the aim of this standard is to transfer the logic of the vCard standard to an HTML/xhtml-document.

The hCard version 1.0 heavily relies on the vCard definition provided by the RFC 2426. The name hCard is deduced from the name of the grounding vCard-standard and HTML as its presentation format. Despite the claims of the authors, it is not exactly a one-to-one representation of the underlying vCard standard.

At the moment of writing, February 2010, a revision of the hCard is in progress. Regrettably, it is not scheduled when this new version of the hCalendar will be published.[W3HC]

2.2.3.1 Structure

The general concept of the hCard is to enrich already existing data of persons or companies embedded in HTML-code with attributes that describe the values. In other words, every element that contains relevant data gets a `class`-attribute which gives information about the type of data. There could be more than just one property that would describe the data that is extracted. In this case, the following property is added to the first property name. The properties are separated by a single white space.[W3HC] The example in code 4 illustrates this concept.

[20] A `tzid`-property is a unique identifier

```
<span>Franky<span>
```

Code 3: Standard HTML-tag.

```
<span class="nickname">Franky</span>
```

Code 4: Enriched HTML-tag.

In this example, we have the name “Franky” embedded in an HTML-span element. In the second step, an attribute of the type “class” is added which has the value “nickname” indicating that the word “Franky” is a nickname as defined in the vCard specification. Another important point is shown in this example. The value of the HTML-attribute must be in lower case.

All of these hCard/vCard-properties that belong to a single hCard/vCard must be embedded within an HTML-element that has the `class`-attribute `vcard` as shown in code 12[21]. A hCard requires at least the root-element `vcard` a formatted name (`fn`-property) and a name (`n`-property). Nevertheless, the `n`-property could be left out given that the implied `n`-optimization[22] is effected. [W3HC]

2.2.3.2 Property Particularities

The hCard standard with its properties was modeled after the vCard standard. However, there are a couple of properties which are supported by the vCard standard but not by the hCard standard. These properties are `name`, `profile`, `source`, `prodid` and `version`. This difference is very important for the translation of a hCard into a vCard as the `version`-property is mandatory for the vCard-standard. As a result, this property must be added in the transformation process of a hCard file into a vCard file. Another difference lies in the `categories`-property which is called `category` in the hCard standard.[W3HC] Other parts of the vCard definition that are not supported by the hCard standard are all of the non-standard properties and non-standard parameters (“`X-` properties” and the “`x-`parameters”).

The consequence of these peculiarities is that a program that transforms an hCard into a vCard must check whether an `n`-property[23] is at hand or not and

[21] Cf. code 12 (p. 18).
[22] This optimization will be explained in the chapter “2.2.3.2 Property Particularities“. Code 12 (p. 18) and 14 (p. 20) show examples using this optimization.
[23] This is due to the implied `n`-optimization which is explained later in this chapter.

adding this property if necessary. Moreover, this program must add a `version`-property to every newly created vCard. The value for this property is always `3.0` as the hCard version 1.0 solely supports this vCard standard. Besides, the hCard definition suggests using the title of the page from which the vCard is extracted as `name`-property. What is more, it also suggests using the `url` of the website as `source`-property. The `prodid` could be provided by the program that translates the hCard.[W3HC]

The hCard format offers a way to create an `n`-property out from the `fn`-property. This procedure is called the implied n-optimization. The rule only applies when the `fn`-property consists of exactly two words separated by a white space. Moreover, no `n`-property must be present. Given that these conditions are fulfilled, the value of the `fn`-property is broken into two parts. Generally, the first word is considered as the given-name and the second word as the family name (e.g. a value assigned to the `fn`-property of "`Johannes Bielohaubek`"[24]). However, this sequence can change. If the first word ends with a comma (`,`) the order is the opposite (e.g. "`Bielohaubek, Johannes`" or "`Bielohaubek, J.`"). Finally, if there is no comma and the second (and last) word ends with a period (`.`) or consists of solely one character, the first word is the Family name (e.g. "`Bielohaubek J.`").[W3HC]

Another particularity deals with a combination of the `fn`- and the `org`-property in one HTML-element. In case that they are exactly the same and that the intention of the provided data is the representation of an organization and not an individual, the `n`-property should have an empty string as value[25].[W3HC]

2.2.3.3 Value Embedding

Concerning the way of storing the processed hCard data, the standard follows two different approaches. First, there is the strategy for values which have a structured order in vCard. These are properties like the `n`-, `adr`-, `geo`- or `org`-property. Each of these elements possesses child elements that contain the relevant data. A quite similar approach is taken for the `type`-attribute. Again, a child element is created containing the relevant data. The short example in

[24] Code 14 (p. 20) shows this case.
[25] Neither the underlying [RFC2425] nor the [RFC2426] give any details if empty values are allowed or not.

code 5 shows the example of an `adr`-property which has in this example child elements carrying the information of the `street-adress`, `postal-code`, `locality` and the `type`-parameter.

```
<div class="adr">
   <span class="type">dom</span>
   <span class="type">work</span>
   <span class="type">postal</span>
   <span class="street-address">Augasse 2-6</span>
   <span class="postal-code">1090</span>
   <span class="locality">Wien</span>
</div>
```

Code 5: hCard Property with Child-Elements.

The corresponding vCard property is shown in code 6. For a better understanding, it must be mentioned, that the sequence of the `adr`-property is `post-office-box`, `extended-address`, `street-address`, `locality`, `region`, `postal-code` and `country-name`. The child elements bear these names.

```
ADR;TYPE=dom,work,postal:;;Augasse 2-6;Wien;;1090
```

Code 6: ADR-Property.

The second strategy concerning the storage location of the hCard data does not need any child elements as it is directly saved in the HTML-elements. The example in code 4 shows this.

The place where the values for a property (or one of its children) are stored depends on certain factors. The most important indicator for this storage place depends on the tag type that is used for the element bearing the hCard. As a rule, if an a-tag[26] (`<a/>`)is used the value of the property can be found in the `href`-attribute. Given the case that in an HTML file an abbreviation-tag is used, the value for extraction is located in the `title`-attribute. If an hCard `Reader` encounters an image-tag the value in question is embedded in the `src`-attribute. Finally, if the data is embedded in an object-tag (`<object/>`) the requested data can be found in the `data`-attribute.[W3HC]

```
<a class=”sample-prop-name” href=”property-value”/>
<abbr class=”sample-prop-name” title=”property-value”/>
<img class=”sample-prop-name” src=”property-value”/>
<object class=”sample-prop-name” data=”property-value”/>
```

Code 7: Tag-related Storage Place of Data.

[26] The a-tag defines an anchor.[W3HTML4.1]

For any other tag type the value is more or less the text content of the element. However, a property could have a child element which has the `class`-attribute `value`. In this case the value of the hCard property indicated in the parent element is extracted from the child element. The place where to find the value within the child element is defined by the already mentioned tag-rules. Here is a simple illustration:

```
<span class="tel">
   <span class="type">Home</span>
   <span class="value">+43</span> (01)
   <span class="value"> 123 45 67</span>
</span>
```

Code 8: Embedded Value.

The telephone number of the example above would be "`+43 123 45 67`". The string "`(01)`" would be ignored.

However, this rule only applies to the first layer of child elements. If one child element has another child element (a grandchild element) with a `value` this element is ignored.[W3VCP]

```
<p class="note">
   <span class="value">
      <em class="value">Watch out! </em>
      <strong>This is an important note!</strong>
   </span>
</p>
```

Code 9: Depth of Value Class Pattern

The value of this `note-`property would be "`Watch out! This is an important note!`" and not only "`Watch out!`"

Due to the structure of the HTML-documents, some of the properties fit well to certain HTML-tag-types. The `url`-property and the `email`-property relate pretty well to the a-tag (`<a>`) which creates a link. In the same manner, photos and logos should be represented in an image element (`<img>`).[W3HC]

```
<a class="email" href="mailto:johannes.paul@wu.ac.at"/>
<a class="url" href="http://www.wu.ac.at/"/>
<img class="logo" src="http://upload.wikimedia.org/wikipedia/de/4/4c/WU_Wien_Logo.png"/>
```

Code 10: Typical Combinations of HTML-Tags and vCard/iCalendar Properties.

Nevertheless, the general rule from the paragraphs above has an exception: This exception takes place, when an element has a child element with a `class`-

attribute `type` and no class-attribute `value`. In this case, the value of the property is the text content of all child elements excluding the `type`-element. [W3HC]

```
<span class="tel">
   <span class="type">Home</span>+43 123 45 67
</span>
```

Code 11: Exception for Unspecified Values.

Despite the above mentioned rules and standards (and their special forming), the hCard standard offers a number of additional exceptions concerning the place where the data of the property (or its parameters) is stored.

The examples in the hCard specifications show examples in which the extraction of the values does not follow the ordinary rules. In this case, two or more hCard properties are defined within one element. The challenge lies in the fact that the properties do not take the same values. The example in code 12 will illustrate this scenario.

```
<span class="vcard">
   <a class="fn org url" href="http://www.wu.ac.at/">WU</a>
</span>
```

Code 12: Combination of Properties in a Single Element.

This example contains three hCard properties: `fn`, `org`, and `url`. Given the standard rules of the hCard scheme, every single property would get the value of the `href`-attribute. However, only the `url` receives the attribute's value whereas the remaining properties receive the text content of the element. Due to the lack of a special rule concerning the distribution of values, the `Reader` class of this thesis looks in the case of multiple values whether or not an `email`- or `url`-attribute is available or not. If there is an `email`- or an `url`-attribute, only these properties get the “standard” value whereas every other attribute receives the text content of the element.

The example in code 12 also contains an exception of the `org`-property. Normally, an `org`-property has two different types of children: The first type bears a `class`-attribute `organization-name` and another with the attribute value `organization-unit`. However, if the `org`-property has no children, the extracted value (in the example “`WU`”) is considered as the `organization-name`.[W3HC]

This example also provides other particularities of the hCard format. Due to the fact that the `fn`-property consists of only one word (“`WU`”) an hCard-vCard-converter must create an empty `n`-property. Moreover, as there is only one word, it must create a `nickname`-property having this value. Given that the value would be “`Wirtschaftsuniversität Wien`”[27], there would be no `nickname`-property as the value consists of more than two words separated by a blank. However, due to the implied `n`-optimization, there would be an `n`-property with “`Wirtschaftsuniversität`” as `given-name` and “`Wien`” as `family-name` which may confuse users without the knowledge of this implied optimization.[W3HC]

Another special treatment could occur in the case of the `geo`-property. Normally, it contains two children: one with the `class`-attribute `latitude`, and another one with `longitude`.[28] However, there are other possibilities to represent this property. Another way would be the use of an `abbr`-element in an HTML-file. In this case, the value for the `latitude` and the `longitude` are saved in the `title`-attribute and separated by a semi-colon (`;`).[W3HCCH] Finally, there is another exception mentioned. Instead of two child elements bearing the `class`-attribute `latitude` and `longitude`, the `geo`-property could also have only one single child element with the `class`-attribute `value`. In this case both values are separated by a semi-colon. Code 13 shows three examples of the encoding of the GEO-property.[W3VCP]

```
<abbr class="geo" title="16.376413;22.084937">home</abbr>

<span class="geo">
   <span class="value">16.376413;22.084937</span>
</span>

<span class="geo">
   <span class="latitude">16.376413</span>
   <span class="longitude">22.084937</span>
</span>
```

Code 13: Different Possibilities of the GEO-Property in hCard (hCalendar).

In the example section of the hCard [W3HCEX], there is an example that shows that it is possible to embed one hCard into another one. This example uses an `org`-property connected with a hCard-root-element. The specification [RFC2426] of the `org`-property does not allow a complete vCard as property

[27] I.e. the German name of Vienna University of Economics and Business Administration.
[28] The rules concerning the storage place of the data follows the logic of the earlier in this chapter mentioned rules. Cf. code 7 (p. 16) and its description.

value. This is the reason why such elements will be ignored by the thesis' `Reader`-class[29] when they appear.

```
<span class="vcard">
   <span class="fn email" href="mailto:h0451119@wu.ac.at">Johannes Bielohaubek</span>
   <span class="org vcard">
      <a class="url fn org" href="http://www.wu.edu">WU</a>
   </span>
</span>
```

Code 14: Unsupported Combination of ORG-Property with a New vCard.

In contrast to the above mentioned `org`-`vcard` combination in the `class`-attribute, the specification of the vCard [RFC2426] allow a complete vCard as value for the `agent`-property.

2.2.3.4 Programming Considerations

Due to the variety of possibilities where to place and hide the values of the properties, the classes which represent the hCard properties and the `Reader`-class follows the steps described in the following paragraphs in order to retrieve the information.

First, the `Reader`-class controls if an element has one or more values. This is required to determine if the case of an exception in the value positioning occurs as described above. If the `class`-attribute consists of an `email` or an `url` property, the exceptional rule for the values is used. If the two exceptions provoking properties are not found, every recognized property receives its ordinary value.[30] This rule ensures, that the badly documented combination of e.g. a `fn` property with an `url` are correctly recognized. It also ensures that the correct value for any other combination is ensured (including the appearance of alleged properties in the `class`-attribute which do not belong to the specification).

In the case of a hCard-property-combination[31] which is not supported by the vCard format, the `Reader`-class neither processes the element node nor its possibly embedded child elements.

---

[29] This class will be described in detail in chapter “3.2 Reading and Preparation of the Data“.

[30] Only the `email` and `url` receives the value according to the tag type, the other properties are attached to the text content. In the example of the combination of the `fn`- and `email`-property in code 14 (p. 20) the `email`-property receives the value “`h0451119@wu.ac.at`” and the `fn`-property the value “`Johannes Bielohaubek`”.

[31] E.g. the embedding of a vCard in an hCard-`org`-property as shown in code 14 (p. 20).

What is more, regrettably, the microformat does not specify the priority of the possible property values given that there is a potential conflict. The classes representing the properties solve this problem in the following way: First, the class looks for the property values in their regular places[32] in compliance to their tag-type. After this, the class looks if the element node has an immediate child node with a `class`-attribute whose value has the string `value`. If this is the case, it designates the value of this child element as property value. Again, the value of the child element is extracted following the general rules[33].

Finally, in order to form valid vCards, the `Reader`-class must examine if the necessary minimum properties are inserted. By default, it must add the `version`-property. Moreover, the `Reader`-class must conduct the implied optimization rules [W3HC] in order to generate these values.[34] This control procedure must be repeated for every single hCard that is retrieved.[35]

2.2.4 hCalendar

The relationship between the hCalendar standard and the iCalendar definitions is nearly the same as between the vCard and the hCard in terms of representation. Like the hCard, the hCalendar is a microformat defined by the participants of the microformat project. However, the hCalendar is not as sophisticated and described in detail as the hCard and some pieces of information are inconsistent or incomplete like the property parameters which are allowed or definitions for `vtimezone`, `vjournal` etc. What is more, not every iCalendar can be transformed into a hCalendar as not every iCalendar component is supported[36].[W3HCL] It must be mentioned that the hCalendar standard is based on the RFC 2445 and not its replacement, the RFC 5545. However, the RFC 5545 is an update of the RFC 2445.[37][RFC5545] As per February 2010, the current version number of the hCalendar is 1.0.[W3HCL] A revision of the hCalendar was also in progress including some changes in the value-class-pattern. A release date for this revised hCalendar standard was not announced.[W3HCL]

---

32 Cf. code 7 (p. 16) and its explanations.
33 Cf. code 7 (p. 16) and its explanations.
34 Especially the implied `n`-optimization is substantial.
35 A homepage could contain more than just one single hCard.
36 The iCalendar components `vtodo`, `valarm`, `vtimezone`, `vjournal`, `vfreebusy` are not supported. Detailed information will be given later in this chapter.
37 The version number of the iCalendar in the [RFC5545] has not changed compared to RFC 2445.

The hCalendar format version 1.0 currently features only the `vevent` component of the iCalendar. The other components (`vtodo`, `valarm`, `vtimezone`, `vjournal`, `vfreebusy`) are not supported yet. This fact is reflected in the root of vCalendar as a hCalendar could have either an attribute of the type `class` with a value `vcalendar` or an `class`-attribute with the value `vevent`.[W3HCL]

The minimum requirements for an hCalendar are a `vevent`-, a `dtstart`- and a `summary`-property. This means that in case of the transformation from this minimum hCard an embracing `vcalendar` must be created with its minimum properties (`prodid` and `version`). Moreover, it must be ensured that every `vevent` component has its own required properties (`dtstamp` and `uid`).

A list of the hCalendar properties can be found on the respective homepage on the Internet.[W3HCLCH] However, most are solely enlisted and not described in detail. What is more, the list of properties does not contain the properties `attach`, `dtstamp`, `created`, `priority`, `sequence`, `tranp`, `recurid`, `comment`, `exdate`, `rstatus`, `related` or `resources`[38]. In the summary of the standard's properties, the property `organizer` is written with “s” (`organiser`).[W3HCLCH] However, it appears that this is a typo. The renaming of the `categories`-property follows its hCard counterpart.[39]

The hCalendar definition also allows some problematic embeddings which cannot be translated to the iCalendar format. In the hCalendar format, the `attendee`-, `contact`- and `organizer`-property could be represented by a hCard. [W3HCL] However, this feature is not supported by the vCalendar. The sole possibility in the vCalendar format is to use an hCard is an URI pointing to a vCard.[RFC5545]

Other problems are caused by the hCalendar's possibility to define the location property as an `adr`-property (taken from the hCard definition.)[W3HCL] Again, a transformation is impossible as the value type of the iCalendar is a text and

[38] On the hCalendar main page, on the contrary, the `attach`-property is allowed. According to the editors of this page, this list is incomplete.[W3HCL] Moreover, the authors use a `dtstamp`-property, a `transp`-property, a `sequence`-property and again an `attach`-property in their examples. In the example section, they even gave an example of a `vtodo`-, a `vjournal`- and a `vfreebusy`-`component`. [W3HCLEX]

[39] Again, it was renamed to `category`.

does not show the structured value of an `adr`.[RFC5545] The same problem occurs, when the location is defined as a `geo`-property.[W3HCL]

Further problems are provoked by the missing definition of the `attendee`-, `contact`- and `organizer`-property. All of these properties contain a long list of descriptive parameters in the iCalendar specifications. However, the hCalendar specifications give no information about these attributes.[40][W3HCLCH]

Despite all problems, Code 15 depicts a simple sample of the hCalendar standard.

```
<div class="vevent">
<span class="summary">Thesis Review</span>
<abbr class="dtstart" title="20100221T080000">Feb. 21, 2010; 8 o'clock</abbr>
<abbr class="dtend" title="20100228T240000">Feb. 28, 2010; midnight</abbr>
</div>
```

Code 15: Sample hCalendar.

Generally, the method of data processing follows the same principles as the hCard. In short, this means that the place where the value of a property is embedded depends on the tag-type which is used and the existence of child elements with the class-attribute `value`.

First, the overall issue of the hCalendar must be mentioned. Despite its name, it only supports the `vevent`. This statement is supported by the hCalendar definitions [W3HCL] and its “cheat sheet” [W3HCLCH]. However, the examples [W3HCLEX] contain parts of other hCalendar components that are not determined yet.[41] Comparing the names of the official authors of the respective pages, the examples are written (with one exception) by other writers than the official standard documentation.[42] The consequence of this controversy is that the iCalendar/hCalendar classes and the `Reader`-class solely supports the `vevent` component and not the `vtodo`, `vjournal`, `vfreebusy`, `vtimezone` or `valarm` components.

[40] In the special case of the `attendee`-property, the RFC 5545 defines eleven different parameters for this iCalendar property. The summary of the hCalendar properties [W3HCLCH] also enlists this property but indicates that this property needs to be documented. As a result, no information about the attributes are available. On the main page of the hCalendar standard is a small, incomplete list of hCalendar properties. This list enlists the `attendee`-property and assigns two attributes/parameters (`partstat`, `role`) to it.

[41] The examples also use properties that are not used in the comparable hCard format like the `version`-property. It could be argued that this property could be left out as the hCard standard as well as the hCalendar standard solely rely on one particular version.

[42] Another issue that must be addressed in this context is the possibility to register anonymously at the microformat homepage and to deliver contribution without any attribution.

Another issue that needs to be addressed is the number of properties that are supported. The hCalendar cheat sheet [W3HCLCH] offers a number of properties that are supposed to be supported by the format. What is more, even the property lists of the hCalendar standard [W3HCLCH][W3HCL] are not congruent as already discussed earlier in this chapter. As a result, the properties that are supported by the `Reader` and the classes representing the iCalendar/hCalendar properties are the same as defined in RFC 5545 for the iCalendar[43].

Regarding the different naming of the `organizer`-property in the hCalendar cheat sheet [W3HCLCH] it is seen as a typo as the property is named corresponding to the RFC in the hCalendar definition [W3HCL]. The “alternative” spelling with “s” (`organiser`) is not supported in this version of the `Reader`-class.

With reference to the missing specifications of the `attendee`-, `contact`- and `organizer`-property, the class `icalendar20_attendee` respects the `parstat` and `role` parameter of the `attendee`-property. The other properties are considered to have no children with property attributes.

Comparable to the hCard standard, the `Reader`-class needs to detect whether the minimum requirements for the iCalendar are met or not. First, the controlling routine must investigate if the already processed data are solely `vevents`. If there are solely `vevents`, they must be embedded within a `vcalendar` that also consists of its minimum properties. Regarding the properties of the `vevent`, the `Reader`-class follows the stricter RFC 5545 and not the RFC 2445 which is used for the hCalendar. Both may have the same properties for the `vevent` in common. RFC 5545 and its predecessor RFC 2445 contain of the same properties. However, the use of the UID- and the `dtstamp`-property are in the RFC 5545 mandatory.

[43] A complite list of the properties which are supported by the ooRexx library can be found in chapter A.3.1.1 “The Classes in Detail“ (p. 67 ff.) in the appendix.

2.2.5 XML, xhtml and HTML

Documents containing microformats[44] use a special document format which allows the author of such documents to give further information about the content of the document.[W3HTML4.1] The hCard and the hCalendar use XML, xhtml and HTML which will be summarized in this section.

The abbreviation XML stand for the words “extensible markup language”. XML is a markup language. The XML Schema and the XML DTD (Document Type Definition) define an individual markup-language. Finally, the XML-standard is a recommendation of the World Wide Web Consortium (W3C).[W3SXML] The regular suffix for XML-files is `.xml`. The XML format is used for a great variety of applications.

```
<?xml version="1.0" encoding="UTF-8"?>
<trunk>
   <limb>
      <leaf color="brown">a &quot;special&quot; leaf</leaf>
      <leaf/>
   </limb>
</trunk>
```

Code 16: Sample XML-Code.

This minimalistic example of code 16 shows a simple XML-document. The tree structure of the XML-files is easily recognizable. Moreover, XML-files consist of an XML-declaration and the root-element. In this example the root-element is called `trunk`. What is more, the sub-child `leaf` has an attribute which describes the tag in greater detail. A tag itself consists of an opening tag, optional attributes, an optional text[45] and a closing tag.[46] Finally, the second `leaf` in code 16 is an empty tag.

The example is considered as well-formed as it follows the rules of the XML-scheme: It has only one root element. Furthermore, every opening tag has an ending tag and the tags are case-sensitive. In addition to this, the values of XML-attributes must be quoted. Besides, the nesting of XML elements must be correct. This means, that every element which is opened inside another element must also be closed in that element. Finally, metacharacters must be escaped.[47] [W3SYN]

[44] In this case, the microformats in question are the hCalendar and the hCard.
[45] This depends on the XML-Scheme or DTD.
[46] However, this could be shortened as shown in the second `leaf` which is an empty tag.
[47] Metacharacters are in this case characters which have a special meaning for the mark-up language.

The introductory example may correspond to the building rules of XML and thus be considered as well-formed. However, it would not be seen as valid. Being valid means that the structure of the XML elements corresponds to a specific set of rules that define how often an element can be used, the attributes of these elements, names of children etc.[W3VAL]

The Hyper Text Markup Language (HTML) is publishing language of the World Wide Web. HTML allows publishing “online documents with headings, text, tables, lists, photos, etc.”, retrieving “online information via hypertext links, at the click of a button”, designing “forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.” and including “spread-sheets, video clips, sound clips, and other applications directly in their documents”[W3IntroHTML]. The roots of HTML lie in the Standard Generalized Markup Language (SGML) which is a “language for the specification of structural markup”[W3IntroHTML]. Due to its origins in SGML, HTML tries to separate the structure from the presentation of the documents. The presentation is more and more outsourced especially to style sheets.[W3IntroHTML] The current version of HTML is 4.01 and is a recommendation of the World Wide Web Consortium (W3C).[W3HTML4.1] Another recommendation is the xhtml standard. “XHTML [sic!] is the reformulation of HTML 4 as an application of XML.”[W3ModXHTML] Its current version is 1.1.[W3XHTML1.1]

2.2.6 SAX and DOM Parser

XML-parsers are needed in order to import and handle data from an XML-file. For the completion of the underlying programming two different XML-processor families were possible: the SAX parser and the DOM parser which differ tremendously in the way they process XML-files.

The first way to process the XML-files in question is the Simple API[48] for XML (SAX). The fundamental function of the parser is to read the data and to raise certain events. This means that a programmer could define what the program

For XML, the greater-than sign “>” is escaped by `>`, the less-than sign “<” by `<`, the ampersand “&” by `&`, the quotations “” by `&quot;` and the apostrophes ' is escaped by `&apos;`.

[48] Acronym for Application Programming Interface.

should do when it encounters e.g. an opening tag or an end tag. As a result of this procedure, SAX parsers are generally faster than their competitors.

There are two possibilities to make use of a SAX parser in an ooRexx oriented programming environment. First, you could use the BSF4ooRexx-environment in order to use a Java SAX parser for processing the XML-files. Second, there is another possibility using a completely in ooRexx programmed SAX parser.[49]

The other way to process XML-data is using a DOM parser. The Document Object Model (DOM) describes the logical structure of documents and thus, allows manipulating and accessing the data.[HEROWO00] The DOM is a recommendation of the World Wide Web Consortium (W3C).[W3CDOM]

The Document-Object-model consists of twelve different node types which represent the data of HTML- or XML-documents. The root of a DOM-tree is the so-called document node. Other important node types are the element node, the text node and the attribute node.[50][RUS02]

In contrast to the SAX parser, the DOM parser reads the complete data first and builds up a tree structure representing the parsed document. After this, it is possible to access, modify, delete and add any node from the existing structure.

The `Reader`-class and the classes representing the properties of the standards use a DOM parser and not a SAX parser. As already described above[51], one XML/xhtml-element can contain several hCard or hCalendar properties. Moreover, there can be additional properties embedded in the child elements.

At the moment, there is no DOM-parser available for ooRexx. However, since the publication of ooRexx version 4.0 in combination with the BSF4ooRexx-engine, ooRexx can use the DOM parsers written for Java.

---

[49] This parser is written by W. David Ashley with contributions of Ruurd Idenburg. This ooRexx program can be found at [W3PARS].

[50] Each of them represents its XML counterpart of the same name.

[51] Cf. chapter “2.2.3.3 Value Embedding“ (p. 15).

The repercussion of this decision was the need to use the BSF4ooRexx engine for the parser and alter the file name extension according to the convention from `.rex` to `.rxj`.

Apart from this, the `Reader`-class uses a non-validating parser in order to parse documents without XML Schema or XML DTDs. However, the files that are parsed must be well-formed in order to be processed by the `Reader`-class.

3 Programming Implementation

The following part of this paper focuses on the created classes with their respective methods and attributes in order to explain their use and the reasoning behind them.

3.1 Opening Remarks

The overall goal of the programming efforts was the creation of ooRexx classes and methods which allow to read and save the data of the earlier described hCalendars, hCards, iCalendars and vCards[52]. Moreover, these classes and methods should be able to create an output either in the IETF-format[53] or the microformat.[54] Finally, the complete code was written in ooRexx[55], an easy to learn and easy to understand programming language.

The overall problem of the coding process was to find methods which allow parsing the text formats of the vCard and iCalendar as well as the XML- and xhtml/HTML-structure of the hCard and hCalendar. Besides, a proper structure for the representation of the properties of the single standards with their values and attributes needed to be created.

The result of the programming activities of this thesis provides several classes for handling the standards. First, there is a class with methods that are dedicated for reading files containing data in a format defined in the hCard, vCard, hCalendar or iCalendar. This class is called `Reader` and can be found in the `reader.rxj`[56]. This class also identifies every single property and creates objects representing them. These objects are instances of class libraries which have the capability to capture every single property of the standards with its parameters and values. Furthermore, these objects are able to return strings which correspond to the formal requirements of either the microformat or IETF-standard. These classes can be found in the files `vcard_classes.rxj`[57] and `icalendar_classes.rxj`[58]. The microformats also use these classes for the

[52] Cf. chapter “2.2 Implemented and Used Standards“.
[53] In this case, the formats in question are the vCard or the iCalendar.
[54] The microformats in question are the hCard or the hCalendar.
[55] Cf. chapter “2.1.1 ooRexx“.
[56] Cf. chapter “reader.rxj“ (p. 143 ff.) for the source code.
[57] Cf. chapter “vcard_classes.rxj” (p. 86 ff.) for the source code.
[58] Cf. chapter “icalendar_classes.rxj” (p. 119 ff.) for the source code.

representation of their properties. Finally, in order to benefit from the classes written for this thesis without additional programming, a simple graphical user interface was added.

The complete code which is written for this thesis is published under the terms of the Apache License Version 2.0. The complete license can be found in the Internet [W3AL] and in the file `ApacheLicense2.0.txt` which is added to the class libraries. Amongst other things, this license allows to use, modify and redistribute the work published under the terms of this license.[W3AL]

3.2 Reading and Preparation of the Data

In order to process the data which is stored in a vCard- iCalendar- hCard- or hCalendar-file, the files bearing the information need to be read. For this thesis, a class called `Reader` which is stored in the file `reader.rxj`[59] was created in order to fulfill certain tasks. In short, these tasks are: First, reading the respective files and unfolding the lines in the case of vCard and iCalendar.[60] Second, identifying the properties of the read file and creating objects[61] representing the information of these properties. Finally, storing these newly created objects in an attribute of an instance and making them available for further processes.

The class `Reader` has three different methods for dealing with the above described tasks. Each of these methods is dedicated to a special purpose[62]. The most important method is `import_fromFile(...)` which allows to process hCards, vCards, hCalendars and iCalendars stored in a file. The second method, `unfolding()` is designed for vCards and iCalendars which are already read but not unfolded. The last method, `type_recognition()` is able to process unfolded[63] vCards and iCalendars.

The first method `import_fromFile(filename_input, input_type)` is the most important method of the `Reader`-class. This method is designed to trigger all

[59] Cf. chapter “reader.rxj” (p. 143 ff.) for the source code.
[60] Further information about the folding of lines can be found in chapter “2.2.1 vCard“ in this paper, in the [RFC2426] in chapter “2.6 Line Delimiting and Folding” or in [RFC5545] chapter “3.1. Content Lines”.
[61] Further information about these objects can be found in chapter “3.3 The vCard/iCalendar Properties in ooRexx“ (p. 34 ff.) and “The Classes in Detail“ (p. 67 ff.).
[62] The methods are described in detail later in this chapter.
[63] Further information about the folding of lines can be found in chapter “2.2.1 vCard“ in this paper, in the [RFC2426] in chapter “2.6 Line Delimiting and Folding” or in [RFC5545] chapter “3.1. Content Lines”.

other methods for processing the files in their correct sequence. These methods are needed for the final outcome of the `Reader`-class: an instance attribute `content` which carries all the information of the read file. This ooRexx queue `content` is also returned when invoking the `method import_fromFile(...)`. The method needs two arguments. The first one is the name of the file which should be processed. The second determines the used standard of the read file. Only `vcard30`, `icalendar20`[64], `hcard`, or `hcalendar` are allowed values for this argument. The values of these arguments are stored in the attributes `filename_input` and `input_type` of an instance of the class `Reader` by the `import_fromFile()`-method.

The instance attribute `content` is an ooRexx queue – an ordered collection class[65] – and in the end, it contains objects[66] representing the properties of the standards. An ordered collection class was needed due to the fact, that the position of a `begin`- and `end`-property of the iCalendar and vCard is important. The choice of a queue was triggered by the special needs of files in the hCalendar standard. As already described in a prior chapter[67], an hCalendar could have either a `vcalendar` or a `vevent` as root-element. In the case of a `vevent` root-element, an additional `begin`- and `end`-property must be added in order to correspond to the iCalendar standard[68]. These properties are added by the function `ical_control()` which is automatically invoked when a hCalendar is processed.

The second method, `unfolding()`, allows to process data which is read and unfolded. Before this method can be called, the attribute `input_type`[69] of an instance of the class must be specified. Moreover, the unfolded lines of a vCard or iCalendar must be stored in the instance attribute `fileLines`. The method `unfolding()` has not return value. The final result – an array containing objects which represent the properties of the processed standard – can be retrieved from the attribute `content` of the instance of the class.

---

64 The number at the end of the value indicates the version number eg. iCalendar Version 2.0.

65 Further information can be found in the ooRexx documentation “Open Object Rexx™ Reference Version 4.0.0 Edition” [ASETAL09] which is installed with the programming language, chapter “5.3.12. The Queue Class”, page 255 or in the internet under [W3TRE] slide 42 – 55.

66 These objects are created while running the program.

67 Cf. chapter “2.2.4 hCalendar“ (p. 21 ff.).

68 In the iCalendar standard, the `vevent` is only a component of the iCalendar and not a complete iCalendar itself.

69 The allowed values are `vcard30` or `icalendar20`.

The last method, `type_recognition()`, is designed for vCard and iCalendar lines which are already unfolded. In order to recognize which standard is used, the attribute `input_type` of an instance of the class `Reader` must receive either `icalendar20` or `vcard30` as value. The unfolded lines are stored in the instance attribute `unfoldedLines`. The result of the method `type_recognition()` can be retrieved from the instance attribute `content`. The example in code 17 illustrates this case.

```
a = .reader~new --creates a new instance of the reader-class
a~input_type = "VCARD30" --defines the input type

                --defines values for the array, values are
                --unfolded lines from a vCard
a~unfoldedLines = .array~of("BEGIN:VCARD",-
                            "FN:Johannes Paul Bielohaubek",-
                            "N:Bielohaubek;Johannes;Paul",-
                            "VERSION:3.0",-
                            "END:VCARD")

a~type_recognition /*calls the method which creates objects for
             the unfolded lines saved in the array "unfoldedLines"
             and saves them in the querry "content".*/

say ""
do item over a~content /*accesses attribute in order to retrieve
                         the final result*/
   say item~makeString --return a string in the vCard format
   say item~class --retrieves information about used class
end  --goes over all values saved in the query "content"

::requires reader.rxj

/* output:

BEGIN:VCARD
The VCARD30_BEGIN class
FN:Johannes Paul Bielohaubek
The VCARD30_FN class
N:Bielohaubek;Johannes;Paul
The VCARD30_N class
VERSION:3.0
The VCARD30_VERSION class
END:VCARD
The VCARD30_END class
*/
```

Code 17: Sample Use of Reader-Class.

The example in code 17 illustrates the use of the `Reader`-class in the case of a vCard. The scenario is that a user has already unfolded lines[70] and wants to use the means of the class `Reader`.[71] The `::requires`-directive indicates the file in which the class `Reader` can be found. After this, an instance of this class is created and the attributes `input_type` and `unfoldedLines` receive values according to their data type.[72] Then, the method `type_recognition()` is called.

[70] Further information about the folding of lines can be found in chapter “2.2.1 vCard“ in this paper, in the [RFC2426] in chapter “2.6 Line Delimiting and Folding” or in [RFC5545] chapter “3.1. Content Lines”.

[71] This scenario implies that no file needs to be read. Moreover, the properties are one single string and not folded.

[72] This is in the case of the `input_type` a string and in the case of the `unfoldedLines` an array.

This method identifies the `properties` saved in the array `unfoldedLines` and creates objects representing them in the queue `content`. Finally, in order to proof the result, a loop is goes over this queue in order to control the output..[73]

The sequence of the methods needed for an hCalendar or hCard are different. In the beginning, it also starts with the method `import_fromFile(...)`. However, this method calls the `xmlReading()`-method[74] this time[75]. Depending on the microformat standard, the method `xmlReading()` either calls `vcard_followNode()` and `vcard_control()` or the methods `ical_followNode()` and `ical_control()`. The methods `vcard_followNode()` and `ical_followNode()` walk down the document tree[76] created by the previous method and identify the `properties` of the respective standards. Like the `ical_followNode()`-method, it also creates objects for them and stores them in the `content` attribute. Finally, the `vcard_control()`- and `ical_followNode()`-method check if the `content`-queue consists of the minimum `properties` of each standard and add further `properties` given that an implied optimization[77] took place.

When the methods `type_recognition()`, `vcard_followNode()` or `ical_followNode()` encounter a property which is not defined in the standards or does not meet the formal requirements of an extension property[78], a remark will be displayed in the command-line interface as shown in code 18.

```
.VCARD30_TESTPROPERTY not defined!
```

Code 18: Sample Message for Unknown Property

The message in code 18 can be interpreted as follows: The part before the underscore (i.e. "`VCARD30`" in this example) indicates the standard in question which would be vCard version 3.0 in this case. The name after the underscore indicates the name of a property which is not defined for a certain standard. For this example, this would be "`TESTPROPERTY`".

[73] The objects representing the properties have to methods which return a string in the microformat or the IETF-format. These methods are `makeString()` or `makeHtml()`. Cf. chapter "3.3 The vCard/iCalendar Properties in ooRexx" (p. 34 ff.).

[74] This method bases on Rony G. Flatscher's `06_listElementNamesIndentedWithAttributes_DOES_NOT_USE_DTD.rxj` which is available in the bsf4oorexx sample section.

[75] This behavior is caused by the second argument given to the `import_fromFile(...)` which indicates the used standard.

[76] Cf. chapter "2.2.6 SAX and DOM Parser" (p. 26 ff.).

[77] Cf. chapter "2.2.3.2 Property Particularities" (p. 14 ff.).

[78] Cf. chapter "2.2.1 vCard" (p. 8 ff.).

Finally, when the method `type_recognition()` for vCards and iCalendars creates a new instance of the identified (existing) property this method calls the `fromString-method` of the respective class and hands over the unfolded line for further processing. In the case of a microformat the method `vcard_followNode()` or `ical_followNode()` calls the method `fromXml` and hands over the document node consisting of the property data. In addition, the method `vcard_followNode()` could also call the method `specialFromXml` in case that multiple properties are stored in the document node and everyone receives the default value[79] and also a method `agentFromXml` for the `agent-property`.[80]

3.3 The vCard/iCalendar Properties in ooRexx

The following part of this paper is dedicated to the ooRexx classes with their methods and attributes which are designed to save and represent the data of vCards, iCalendars, hCards and hCalendars. In order to avoid any confusion when using the term “attribute” the following convention is in place. The term “attribute” is always used in connection of programming purposes. For the “attributes” of a property the term “parameter” or “property attribute” is used.

3.3.1 Class Structure

Every single class in the files `vcard_classes.rxj` and `icalendar_classes.rxj`[81] represents the set of properties of the relevant standard. As the hCard and the hCalendar standards are transformations of the vCard and iCalendar standard in HTML/xhtml/XML they also use the classes of the above mentioned files.

The naming of the classes follows a very simple convention. Every class name has a simple prefix which indicates the standard to which the class belongs[82]. After this, an underscore (`_`) follows. Finally, the name of the property is added as defined in the relevant standard[83]. The vCard standard has a property which is called `sort-string`. The class representing this property is named

---

79 Cf. chapter “2.2.3.3 Value Embedding“ (p. 15 ff.).
80 A description of this exception can be found in chapter “2.2.1 vCard“ (p. 8 ff.).
81 The source code can be found in the chapters “vcard_classes.rxj“ (p. 86 ff.) and “icalendar_classes.rxj“ (p. 119 ff.).
82 This would be either “vcard30” for vCard Version 3.0 and hCard Version 1.0 or “icalendar20” for iCalendar Version 2.0 or hCalendar Version 1.0.
83 Cf. [RFC2426] chapter “3. vCard Profile Features” and [RFC5545] chapter “3.5 Property” and “3.6 Calendar Components”.

`vcard30_sort_string`[84]. The hCard/hCalendar has one property called `category`. The corresponding class for this property would be either `icalendar20_categories` or `vcard30_categories`.

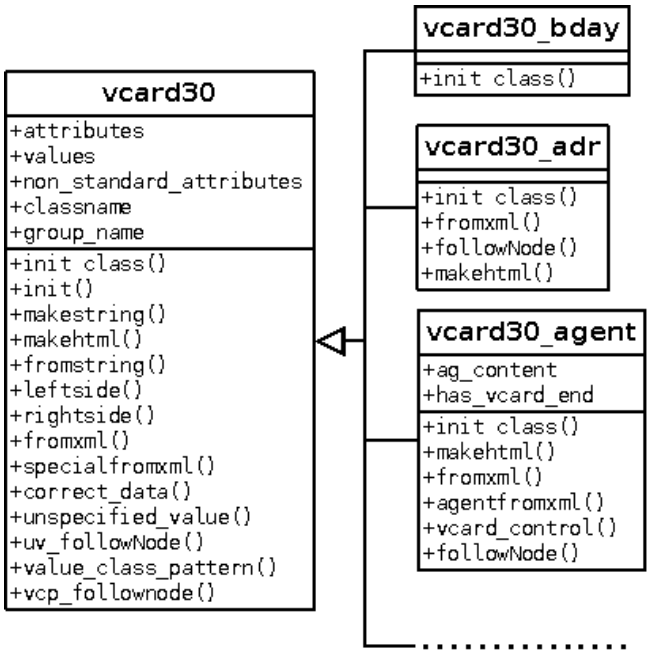

Figure 1: Class Diagram vCard.

Figure 1 depicts a general scheme of the vCard classes. It must be mentioned that this graph is incomplete due to the huge amount of different vCard/hCard properties[85]. As a result, some sample classes are mentioned in order to illustrate the ideas behind[86]. Generally speaking, figure 1 shows the superclass `vcard30` with three of its subclasses. The class `vcard30_bday` stands as a representative for properties which have no child-elements in their HTML-/hCard-representation. The `vcard30_adr`-class demonstrates properties with a structured property value.[87] Finally, the class `vcard30_agent` shows a structure which only occurs for this single vCard/hCard-property. Figure 2 shows the general scheme of the iCalendar classes which is quite similar to the vCard scheme in figure 1. The superclass is called `icalendar20` and two of its subclasses are depicted. The class `icalendar20_summary` stands as a

[84] ooRexx is case insensitive as every character except strings will be translated into uppercase. As a result, there is no difference between `vcard30_sort_string` or `VCARD30_SORT_STRING`.

[85] There are around 30 different properties defined for vCard.

[86] The complete list of properties which have ooRexx classes can be found in the appendix in the chapter “The Classes in Detail“ (p. 67 ff.) table 2 (p. 72) and 3 (p. 76).

[87] Cf. chapter 2.2.3 hCard (p. 13 ff.) for detailed information about the different principles of encoding information in the hCalendar/hCard scheme. Code 5 “hCard Property with Child-Elements.“ (page 16) and code 6 “ADR-Property.” (page 16) show an example of this property.

representative of properties with a structured property value and the class `icalendar20_geo` for properties with no child-elements in their HTML-/hCalendar-representation.[88]

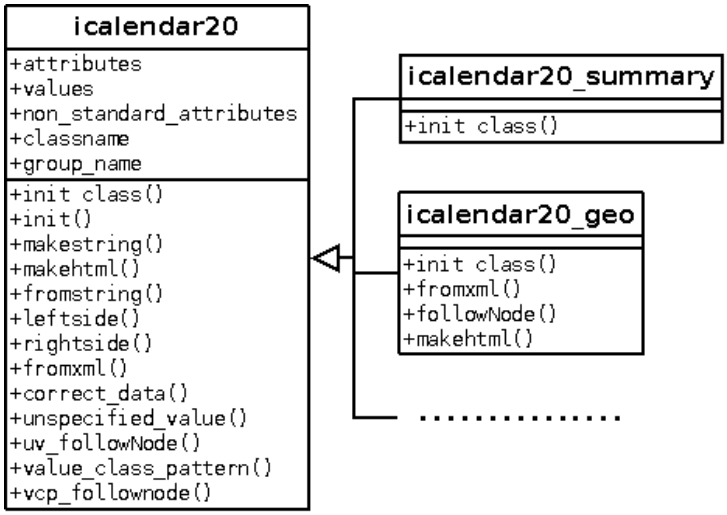

Figure 2: Class Diagram iCalendar.

The attributes of the classes representing the property values and property parameters will be discussed later in detail due to the special concepts behind them. In short, every property parameter or value is represented by an array and is created dynamically.[89]

Generally speaking, the methods can be divided into two main categories. The first category is responsible for the input, the other category for the output. Every class has two methods for creating an output. These methods are `makeString()` and `makeHtml()`. These methods return a string which is structured according to the standard specifications. In the case of the `makeString()`-method, the string corresponds to the iCalendar/vCard format, in the case of the `makeHtml()`-method to the hCalendar/hCard format. Given that the microformat does not specify a property[90] a message is displayed in the command-line interface indicating that the iCalendar/vCard class is not specified in the respective microformat.

[88] The complete list of properties which have ooRexx classes can be found in the appendix in the chapter “The Classes in Detail“ (p. 67 ff.) table 4 (p. 82) and 5 (p. 85).

[89] Cf. chapter “3.3.2 Simple Approach“ (p. 38 ff.) and “3.3.3 Meta-Programming Approach“ (p. 39 ff.). The latter chapter also explains the interaction of the `int()`-, `init class()`-method and the attributes of the classes.

[90] Examples of such properties would be the `version`- or `prodid`-property.

The methods which process the data of the properties analyze the information which they receive and distribute it to the corresponding property parameter or value and stores it in the respective instance attribute. The methods can handle two different types of information. They can handle strings in the iCalendar/vCard format or element nodes[91] containing information in the hCalendar/hCard format.

The most important method for strings in the iCalendar/vCard format is the `fromString()`-method. This method also uses the methods `leftSide()` and `rightSide()` which process the parts of the strings. The `leftSide()`-method deals with the parameter-grouping part of a iCalendar/vCard-property whereas the method `rightSide()` with the values of the property.[92]

The counterpart of the `fromString()`-method is the `fromXml()`-method which deals with elements nodes. In order to retrieve the data from this `element node` according to the hCard/hCalendar specifications this method calls the method `correct_data()`. This method further uses the methods `unspecified_value()` and `uv_followNode()` for unspecified values and `value_class_pattern()` and `vcp_followNode()` for the value-class-pattern exception.[93]

The method `specialFromXml()` is called in the case that multiple properties are stored in an HTML-element[94]. The method in the superclass assigns the text content of the element as property value. In the case of the `email` or `url` property, the `specialFromXml()`-method calls the `fromXml()`-method as these properties are not effected by the exception.

The `fromXml()`-method for hCard/hCalendar properties with child elements[95] is different. This method needs the additional method `followNode()` in order to go over these child elements and retrieve the values from it using the `correct_data()`-method. Moreover, these properties need individual `makeHtml()`-methods for their microformat representation[96].

---

91 `Element nodes` are one type of nodes of the DOM-model.
92 Further information about the structure of the vCard/iCalendar can be found in the respective chapters of this paper. Cf. chapter “2.2.1 vCard“ (p. 8 ff.) and chapter “2.2.2 iCalendar“ (p. 11 ff.).
93 Cf. chapter “2.2.3.3 Value Embedding“ (p. 15 ff.).
94 Cf. chapter “2.2.3.3 Value Embedding“ (p. 15 ff.) and “2.2.3.4 Programming Considerations“ (p. 20 ff.).
95 Cf. code 5 “hCard Property with Child-Elements.“ (page 16). The properties with child elements are `n`, `adr`, `geo` and `org`.
96 The `adr`-property in figure 1 (p. 35) is one of these properties.

Finally, the `agent`-property[97] is an exception in its class structure as it could harbor a complete vCard as property value. This matter of fact made it necessary to program a `makeHtml()`-method which is able to write a complete hCard. Moreover, reading such an hCard property was extensive as the embedded hCard could also have implied property optimizations. As a result, the method `ag_fromXml` is quite similar to the method `vcard_followNode()` of the class `Reader`[98].

Another exception are the `X-`properties[99]. These are properties which are not specified by the [RFC2426] or [RFC4770] but follow the conventions concerning the structure of the property[100]. All property parameters are stored in the attribute `non_standard_attributes`.

3.3.2 Simple Approach

The structure of every property follows a simple basic scheme. Every property can have one or multiple values. Moreover, most of the properties have one or more property parameters. Again, each of these property parameters can have one or multiple parameter values.[101]

The classes[102] which are created for this theses copy this structure as follows. First, every single property is represented by an ooRexx class. These classes have attributes which represent the parameters and the property values. Each property parameter is implemented by an array that collects the values for its respective parameter. The values of a property are also grouped together in an array.

97 Cf. chapter “2.2.1 vCard“ (p. 8 ff.).
98 Cf. chapter “3.2 Reading and Preparation of the Data“ (p. 30 ff.).
99 Cf. chapter “2.2.1 vCard“.
100 The name of such a property must start with an “`X-`”.
101 Cf. chapter “2.2.1 vCard”.
102 These classes can be found in the files `vcard_classes.rxj` and `icalendar_classes.rxj`.

```
::class vcard30_adr public

   --property parameter
   ::attribute type
   --property values
   ::attribute value_data

::method init
/* assigns the data type Array
   to the attributes representing
   the property parameters and values
*/
   self~type = .array~new
   self~value_data = .array~new
```

Code 19: Sample Class Illustrating a Blueprint of the Property Representation.

Code 19[103] illustrates the above mentioned principle of the construction plan of a vCard/hCard/iCalendar/hCalendar-class with an example of an `adr`-property.[104] This property has only one parameter called `type`. Looking at the example in code 19, first, the attributes of the class are determined. This is an attribute `type` which is dedicated for all values of the `type`-parameter and an attribute `value_data` which is dedicated for the values of the property. The method `init` assigns an array[105] to these attributes. Generally speaking, every class which represents a property has an attribute `value_data` in which the property values are stored and for every specified property parameter an instance attribute named after this parameter.

The problem of this solution is the dimension of all vCard/hCard properties and all iCalendar/hCalendar properties. These standards have a large number of properties[106]. What is more, some of these properties have also a large number of property parameters[107]. As a result, every single class needs to specify the parameters of the represented class.

3.3.3 Meta-Programming Approach

Due to the number of different properties with their respective property parameters, a meta-programming approach was reasonable. Meta-

[103] The example is very simplified for illustrative purposes. A complete example of a class would also include the variables `classname` and `group_name` (bearing the name of the class and a possible grouping). Moreover, all of the methods described in chapter “3.3.1 Class Structure“ (p. 34 ff.) would also be available.

[104] A complete description of this property can be found in the [RFC2426] in chapter “3.2.1 ADR Type Definition”. This property has been chosen as an examples as it has been already used in this thesis. Cf. code 5 (p. 16), code 6 (p. 16) and figure 1 (p. 35).

[105] The default type of variables in ooRexx are strings. If no value is assigned to a variable, the variable has its own name in capitals as value.

[106] The vCard standard has 35, the hCard 28, the iCalendar 50 and the hCalendar 28 properties. Cf. Chapter “The Classes in Detail“.

[107] E.g. the iCalendar property `attendee` with eleven different parameters.

programming sets up other program parts or complete programs. A frequent field of use for meta-programming is the reduction or elimination of an error-prone or monotonous programming assignment.[Ort07] In the case of the ooRexx classes, this technique is used in order to ease the programming work. In the end, the classes have the same attributes as described above[108]. However, the number of code lines and the programming effort and the error rate are considerably reduced. Code 20, 21 and 22 illustrates this.

```
::class vcard30_adr public subclass vcard30

/*
attributes and values variables of class
*/
::method init class
   self~attributes = .array~of("type", "value_data")
   forward class (super)
```

Code 20: Sample Property Class After Meta-Programming Transformation.

Code 20 illustrates the transfigured property class.[109] The aim which is achieved by the meta-programming approach is the creation of the attributes representing the property parameters and values dynamically. The class standing for the property has only the class attribute `attributes`[110] which is defined in the superclass[111]. In the `init`-class-method of the sub class, this attribute receives the names of the other attributes[112] of the respective class. After this, the `init`-class-method calls the `init`-class-method of the superclass for the further processing steps. These methods are called class methods as they are associated with the class and not with an instance of the class. In ooRexx, these methods are identified by the keyword `class` after the name of the method. The class method `init` (`init`-class) is the method which is called first when the class itself is created. The corresponding superclass of the `adr`-property in code 20 is stated in code 21.

[108] Cf. chapter “3.3.2 Simple Approach“ (p. 38 ff.).
[109] Cf. code 19 (p. 39).
[110] Cf. code 21 for the `superclass` (p. 41).
[111] Cf. code 20 (p. 40).
[112] These attributes the values and property parameter of a property.

```
::class vcard30 public
::attribute attributes class --attribute used in subclasses

::method init class --init class method is called when classes
                    --are set up

   if self~id="VCARD30" then return --superclass has no
                       --individual attributes which need a getter-
                       --setter-method

   do name over self~attributes --calls every item which is stored
                       --in the array attributes which carries the
                       --names of the property parameters and
                       --values
      self~define(name, makegetter(name)) --creates a getter-
                                          --method
      self~define(name||'=', makesetter(name)) --creates a setter-
                                               --method
   end

   ::method init
      attributes = self~class~attributes --assigns newly created
                                         --values

      do name over attributes
         self~send(name||'=',.array~new)
         --sets initial attributes-value to an array
      end
```

Code 21: Super Class of Property Classes.

The `init`-class-method of the subclass continues with the `init`-class-method of the superclass. In the following a `getter`- and a `setter`-method[113] are created in order to create functional attributes. Normally, the methods which allow retrieving and setting the value of an attribute are created automatically by the interpreter. These attributes are created dynamically in this case. Code 22 shows the corresponding routines which create these methods. These routines return a string which contains the statements for creating the `getter`- respectively `setter`-method. Returning to code 21, the method `define`[114] makes a method out of the string returned by the routines. Finally, when a subclass creates an instance, the inherited `init`-method is called. This method assigns an array to all of the attributes.

[113] Cf. [ASETAL09] (p. 81 f.) for information about the `getter`- and `setter`-method. The getter method looks like this `::method name; expose name; return name`, the setter like this `::method "NAME="; expose name; use arg name`.

[114] Cf. [ASETAL09] (p. 124 f.) for detailed information about the `define`-method.

```
/*
routine creates a getter method for attributes
*/
::routine makegetter
   parse arg name
   code="EXPOSE" name "; RETURN" name
   return code
   --the returned string follows the ooRexx syntax for
   --a getter-method e.g.
   --"EXPOSE name;" allows direct access to attribute
   --"RETURN name" returns value

/*
routine creates a setter method for attributes
*/
::routine makesetter
   parse arg name
   code="EXPOSE" name "; USE ARG" name
   return code
   --the returned string follows the ooRexx syntax for
   --a setter-method e.g.
   --"EXPOSE name;" allows direct access to attribute
   --"USE ARG name" gets argument and assigns it to
   --               the attribute
```

Code 22: Routines Creating Getter- and Setter-Methods.

The routines in code 22 return a string which correspond to the specifications of a `getter`- and a `setter`-method.[115]

The direct comparison of the two different approaches shows the advantage of the meta-programming approach[116] compared to the simple[117] approach[118]. In the case of this thesis, the meta-programming solution allows to reduce the number of code lines of every class representing a property. However, this effect only occurs in this example of meta-programming when there are numerous classes. The vCard/hCard standard as well as the iCalendar/hCalendar standard have such a high number of properties.

[115] Cf. [ASETAL09] (p. 81 f.) for information about the `getter`- and `setter`-method.
[116] Cf. code 20 (p. 40), code 21 (p. 41) and code 22 (p. 42).
[117] The word “simple” is used in this context for contrasting the two approaches and not as evaluation.
[118] Cf. code 19, p. 39.

3.4 Using the Class Libraries

The following chapter is dedicated to the way how to use the `Reader` class and the classes written for the standards.

3.4.1 Command-Line Interface

Along with the files containing the classes for reading data and representing the vCard/hCard and iCalendar/hCalendar comes a very simple program which demonstrates one possibility of using these classes. This program is stored in the file `CMD_GUI.rex`[119] and allows reading a file and creating a new file which contains the parsed data in the desired format.

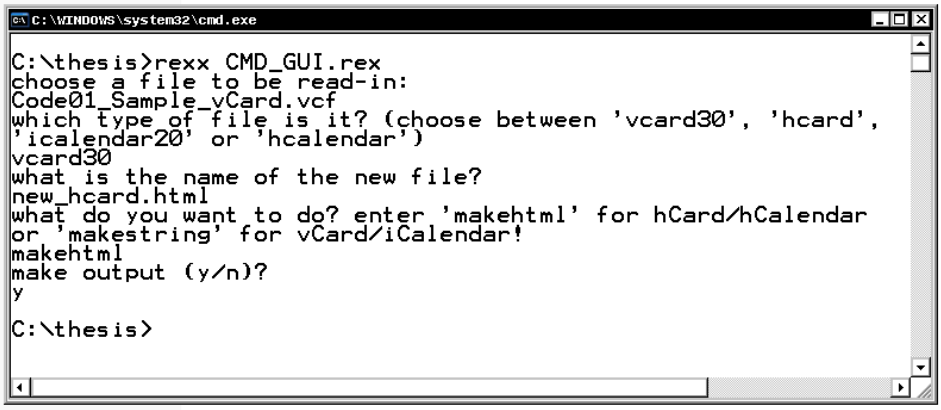

Figure 3: Conversion of vCard into hCard in the CLI.

Figure 3 illustrates the conversion of the vCard from code 1[120] into an hCard.[121] After invoking the program `CMD_GUI.rex` the program requires a couple of information in order to work. First, it needs the name of the file for processing and the standard in which the data is encoded. This information is needed for the `Reader`-class[122]. After this, the name of the new file is needed as well as the format of this file.

[119] Cf. chapter CMD_GUI.rex (p. 160 ff.) for the source code.
[120] Code 1 can be found on page 8 in this paper.
[121] This example is shown in the command-line interpreter `cmd.exe` of Windows XP.
[122] Cf. chapter “3.2 Reading and Preparation of the Data“ (p. 30 ff.).

```
<div>
 <div class="vcard">
  <span class="note">Hello; this is a sample business card.</span>
  <span class="fn">Johannes Paul Bielohaubek</span>
   <span class="n">
   <span class="family-name">Bielohaubek</span>
   <span class="given-name">Johannes</span>
   <span class="additional-name">Paul</span>
  </span>
  <a class="email" href="mailto:johannes.paul@wu.ac.at">
     <span class="type">pref</span>johannes.paul@wu.ac.at</a>
  <span class="org">
   <span class="organization-name">Wirtschaftsuniversitaet Wien</span>
   <span class="organization-unit">Institute for Management Information Systems</span>
  </span>
 </div>
</div>
```

Code 23: New hCard Created From Code 1.

The file which was transformed contained the vCard of code 1[123]. The output of the transformation process is depicted in code 23. In order to allow the user to adopt the appearance of a newly created HTML-file easily, no formatting is done as figure 4 shows.

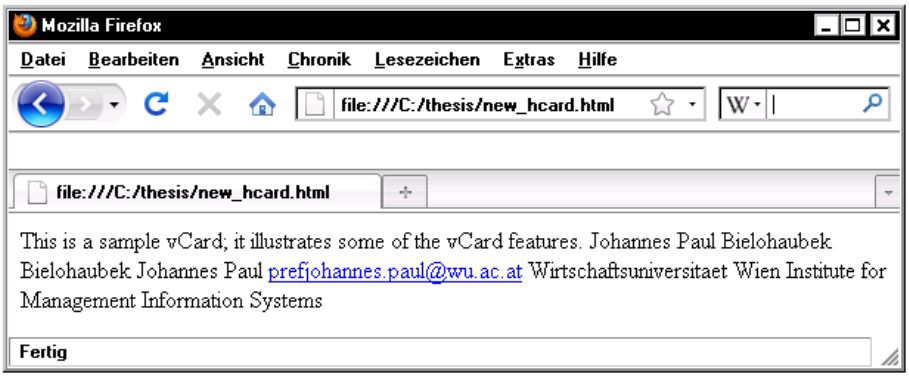

Figure 4: hCard Opened in Browser.

[123] Code 1 can be found on page 8 in this paper.

3.4.2 Graphical User Interface

Another possibility to interactively use the classes designed to carry the information of the already described calendar and business card standards is the program `claro_GUI.rxj`[124]. The `claro_GUI.rxj` enables the creation of a new file in one of the calendar or business card standards.

The advantage of this approach lies in its easy handling. In comparison to controlling the transformation process in the command-line Interface, the parameters for the conversion are now set using a graphical user interface (GUI). Instead of typing in the relevant information, a user can now simply choose the file in question and select the standards.

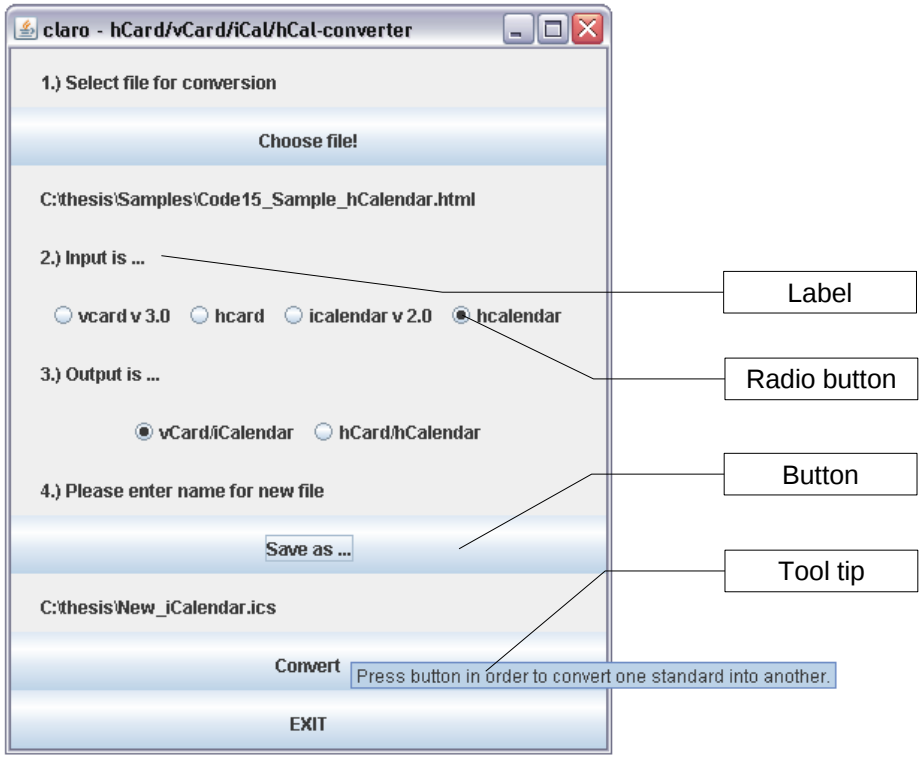

Figure 5: GUI.

Figure 5 illustrates the creation of a file containing the hCalendar example of code 15[125] into a new file in the iCalendar standard. In order to help a user, the buttons and radio buttons have tool tips which give further information about the use of the respective object.

[124] The source code can be found in the chapter “claro_GUI.rxj“ (p. 155 ff.).
[125] Cf. code 15 on page 23.

Java `swing` was used for the creation of the GUI which is available through the functionality of BSF4ooRexx. Alternatively, the `abstract window toolkit` (`awt`) or the `standard widget toolkit` (`swt`) could have been used for the creation of the GUI. The advantage of the `swt` lies in the fact that the created user interfaces imitate the appearance of the relevant operating system. However, it is not a part of the standard Java installation[126].

The window is composed of a frame onto which several labels, buttons and radio buttons are placed in a grid layout. In order to indicate which radio buttons belong together, they are grouped within a button group. What is more, they are also grouped within a panel for displaying. Finally, all of these elements are packed into a window and this window is shown. The visual result can be seen in figure 5 in this chapter.

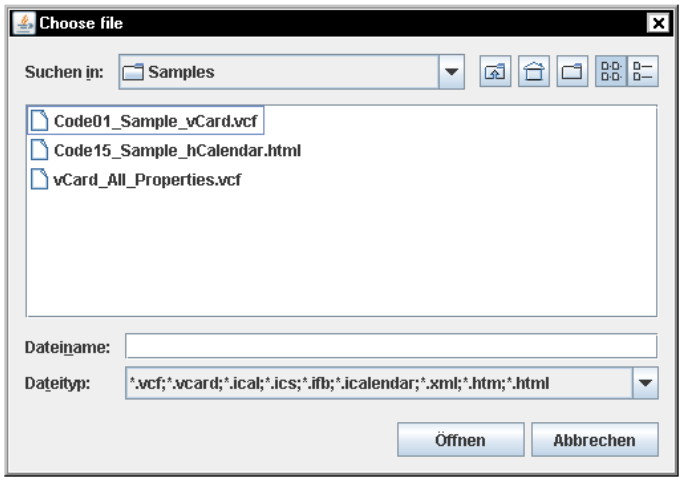

Figure 6: File Chooser.

Figure 6 shows the file chooser which appears when the button “Choose file!” is pressed. In order to facilitate the usage, files of the most common iCalendar, hCalendar, vCard and hCard file formats are filtered out.

Figure 7: CLI While Running claro_GUI.rxj.

[126] `swt` can be found at [W3SWT].

Figure 7 shows the command-line interpreter while running the program in `claro_GUI.rxj`. This window indicates the name and the memory location of the file which will be processed as well as the name and memory location for the new file. Figure 7 also shows the standard message when encountering an unknown property. In this case, this is not an error. Properties are indicated in the microformats in the `class`-attribute of the xhtml-element. When the `Reader`-class encounters such a xhtml-attribute it checks whether or not such a property exists. The start of a hCalendar `vevent` has a xhtml-`class`-attribute with the value `vevent` which is found and indicated in the CLI.

Finally, code 24 shows the result of the transformation of code 15[127]. As already described[128], the `Reader` class automatically adds the required iCalendar properties which are not defined in the hCalendar definitions. In this example, the properties `dtstamp`, `uid`, `version` and `prodid` are added.

```
BEGIN:VCALENDAR
BEGIN:VEVENT
SUMMARY:Thesis Review
DTSTART:20100221T080000
DTEND:20100228T240000
DTSTAMP:20100417T103323Z
UID:h0451119@wu.ac.at-20100417T103323Z-1285
END:VEVENT
VERSION:2.0
PRODID:claroRexx_h0451119@wu.ac.at
END:VCALENDAR
```

Code 24: New iCalendar Created From Code 15.

The graphical user interface benefited from the 2009 edition of BSF4ooRexx. This new version allows inter alia[129] the implementation of Java interfaces. In the case of the GUI, the methods for events are implemented in form of ooRexx methods. Some events are for example clicking on a button, closing a window or pressing a key. For the `claro_GUI.rxj` the Java interfaces `java.awt.event.ActionListener` and `java.awt.event.WindowListener` were implemented. They were needed for the action which takes place when pressing a button or clicking on the x-button for closing the window.

[127] Cf. code 15 (p. 23).
[128] Cf. chapter “3.2 Reading and Preparation of the Data“ (p. 30 ff.) and “2.2.4 hCalendar“ (p. 21 ff.) for further details.
[129] Cf. [Flat09] for the complete list of the new functionality of the 2009 edition of BSF4ooRexx.

3.4.3 Using the Property Classes Directly

The following part will show the use of the classes without the help of the user interfaces which are described above[130].

Code 25 illustrates the creation of vCard/hCard properties in ooRexx. In order to access the classes for the properties, the file `vcard_classes.rxj` is required. For creating a vCard the properties `begin`, `end`, `version`, `n` and `fn` are mandatory.[131] First, an instance of the respective class is made. The naming convention is "`vcard30_`" and the name of the property. After this, the information of the property must be set. Every property has an attribute `value_data` – an array – which is dedicated for the values of a property. Depending on the property, property parameters are possible or not. These parameters are described in the relevant standard.[132] If a parameter is possible, this parameter is represented by an attribute bearing the name of the parameter. In the example of code 25, the properties `tel` and `email` are allowed to have the parameter `type`.

```
/*
This is a short illustration of the creation of objects
representing a vCard/hCard.

The naming convention for the class representing property:
vcard30_propertyname

The number and names of properties (with respective property
parameters) are found in RFC2426
http://www.ietf.org/rfc/rfc2426.txt
*/

/*creation of BEGIN-property representing begin of vCard
property is mandatory for vCard
no property parameters defined */
begin = .vcard30_begin~new
begin~value_data~append("VCARD")

/*creation of VERSION-property representing version of vCard standard
property is mandatory for vCard
no property parameters defined*/
version = .vcard30_version~new
version~value_data~append("3.0")

/*creation of FN-property representing a formatted name
property is mandatory for vCard
no property parameters defined*/
fn = .vcard30_fn~new
fn~value_data~append("Institute for Management Information Systems")

/*creation of N-property representing a name
property is mandatory for vCard
no property parameters defined*/
n = .vcard30_n~new
n~value_data~append("Institute for Management Information Systems")
```

[130] Cf. chapter "3.4.1 Command-Line Interface" (p. 43 ff.) and "3.4.2 Graphical User Interface" (p. 45 ff.).
[131] Cf. chapter "2.2.1 vCard" (p. 8 ff.).
[132] For the vCard the properties with their property parameters are described in [RFC2426] chapter "4. Formal Grammar" (p. 27 ff.).

```
/*creation of ORG-property representing an organisation
no property parameters defined*/
org = .vcard30_org~new
org~value_data~append("Wirtschaftsuniversitaet Wien")
org~value_data~append("Institute for Management Information Systems")

/*creation of URL-property representing an url
no property parameters defined*/
url = .vcard30_url~new
url~value_data~append("http://ec.wu.ac.at/")

/*creation of TEL-property representing a telephone number
TEL-property has one single property parameter specified "TYPE"*/
tel = .vcard30_tel~new
tel~value_data~append("+43 1 313 36 4443")
tel~type~append("WORK")
tel~type~append("PREF") --"prefered"
tel~type~append("VOICE")

/*creation of EMAIL-property representing an e-mail address
EMAIL-property has one single property parameter specified "TYPE"*/
line = "EMAIL;TYPE=PREF:wi-wk@wu.ac.at" --processing of a vCard line
email = .vcard30_email~fromString(line)

/*creation of X-property i.e. non-standard property
classnames or normally generated automatically
however, for X-property individual classnames
starting with "X-" possible thus use of attribute
"classname" in order to specify name of class*/
xprop = .vcard30_x_property~new
xprop~value_data~append("Johannes Bielohaubek")
xprop~classname="X-CREATOR"

/*creation of end-property representing begin of vCard
property is mandatory for vCard
no property parameters defined*/
end = .vcard30_end~new
end~value_data~append("VCARD")

--collection of all objects representing vCard/hCard properties
business_card = .array~of(begin, version, fn, n, org, -
                          url, tel, email, xprop, end)

do property over business_card
   say property~makeHtml --return properties in hCard style
end

::requires 'vcard_classes.rxj'

/*
Further explanation: The VERSION- and X-properties are not specified for the hCard
standard.
*/
```

Code 25: Creation of a hCard.

Code 26 shows the newly created hCard of code 25. The hCard standard does not include the property `version` or any `x-properties`.[133] As a result, a representation of these properties in the hCard style cannot be created.

[133] Cf. chapter “2.2.3 hCard“ (p. 13 ff.).

```
<div class="vcard">

  <span class="fn">Institute for Management Information Systems</span>
  span class="n">
   <span class="family-name">Institute for Management Information Systems</span>
  </span>
  <span class="org">
   <span class="organization-name">Wirtschaftsuniversitaet Wien</span>
   <span class="organization-unit">Institute for Management Information Systems</span>
  </span>
  <a class="url" href=”http://ec.wu.ac.at/”>http://ec.wu.ac.at/</a>
  <span class="tel"><span class="type">work</span><span class="type">pref</span>
   <span class="type">voice</span>+43 1 313 36 4443</span>
  <a class="email" href="mailto:wi-wk@wu.ac.at">
   <span class="type">pref</span>wi-wk@wu.ac.at</a>

</div>
```

Code 26: Newly Created hCard.

There are two ways to assign the data of a property to the attributes of the object representing it. First, the attributes could be addressed directly and the values added to them. This possibility is illustrated in the example of code 25 with the exception of the `email`-property. For this case, the second possibility is used. As already described, every vCard/hCard/iCalendar/hCalendar representing class has a class method `fromString`. The creation of the `email`-property in code 25 uses this method. The method takes an unfolded string in the vCard format as value.

The creation of a non-standard property is a little bit different[134]. In code 25, a property named `X-CREATOR` is created as an instance of the class `vcard30_x_property`. By default, the instance would receive the value `X-PROPERTY` as property name. However, this property can bear any string which starts with `x-` as property name[135]. As a result, the name of the class must be specified in addition with the `classname`-attribute. Specifying the property name is not necessary when using the `fromString`-method.

For the creation of an output in code 25, the objects representing the properties are put into an array as the order of them is important. The `begin`-property must be the first item as well as the `end`-property the last. After this, a loop goes over the items of the array and uses the method `makeHtml()` for creating strings in the hCard format. The result can be seen in code 26. Alternatively, the method `makeString()` for creating strings in the vCard format could have been used.

---

[134] Cf. chapter “3.3.1 Class Structure“ (p. 34 ff.) and chapter “2.2.1 vCard“ (p. 8 ff.).

[135] Cf. chapter “2.2.1 vCard“ (p. 8 ff.) for further information about the `x-property`. The information provided in this chapter also apply to the iCalendar standard. The hCalendar/hCard format does not specify an `x-property`.

This means that the loop in the end of code 25 needs to be changed as shown in code 27.

```
do property over business_card
   say property~makeString --return properties in vCard style
end
```

Code 27: Changed Loop of Code 25 for Creation of a vCard.

The corresponding new output after altering the loop of code 25 with the loop shown in code 27 is depicted in code 28.

```
BEGIN:VCARD
VERSION:3.0
FN:Institute for Management Information Systems
N:Institute for Management Information Systems
ORG:Wirtschaftsuniversitaet Wien;Institute for Management Information Syste
 ms
URL:http://ec.wu.ac.at/
TEL;TYPE=WORK;TYPE=PREF;TYPE=VOICE:+43 1 313 36 4443
EMAIL;TYPE=PREF:wi-wk@wu.ac.at
X-CREATOR:Johannes Bielohaubek
END:VCARD
```

Code 28: New vCard Using Code 25 With The Loop From Code 27.

The functionality of the classes dedicated to the hCalendar/iCalendar standard are identical to the above described way as code 29 shows. In order to access the classes for the properties, the file `icalendar_classes.rxj` is required. For creating a iCalendar the properties `begin`, `end`, `version`, `prodid` are mandatory. Moreover, each `vevent` must have at least a `begin`-, `end`-, `dtstamp`- and `uid`-property. [136] The rest is comparable to the principles illustrated in code 25.

```
/*
This is a short illustration of the creation of objects
representing an iCalendar/hCalendar.

The naming convention for the class representing property:
iCalendar20_propertyname

The number and names of properties (with respective property
parameters) are found in RFC5545 (http://www.ietf.org/rfc/rfc5545.txt)
*/

/* creation of a mandatory begin property indicating the
begin of an iCalendar */
begin_calendar = .icalendar20_begin~new
begin_calendar~value_data~append('VCALENDAR')

/* creation of a mandatory version property */
version = .icalendar20_version~new
version~value_data~append('2.0')

/* creation of a mandatory prodid property (showing another
possibilty to deal with the data */
line = 'PRODID:-//hacksw/handcal//NONSGML v1.0//EN'
prodid = .icalendar20_prodid~fromstring(line)

/* creation of a mandatory begin property indicating the
begin of an vEvent */
begin_event = .icalendar20_begin~new
```

[136] Cf. chapter “2.2.2 iCalendar“ (p. 11 ff.).

```
begin_event~value_data~append('VEVENT')

/* creation of the mandatory dtstamp-property for the
vEvent */
dtstamp = .icalendar20_dtstamp~new
dtstamp~value_data~append('20100508T160202Z')

/* creation of the mandatory uid-property for the
vEvent */
uid = .icalendar20_uid~new
uid~value_data~append('h0451119@wu.ac.at&20100508T160202')

/* creation of a description-property for the vEvent */
description = .icalendar20_description~new
description~value_data~append('A very important meeting.')

/* indicating the start of the vEvent */
dtstart = .icalendar20_dtstart~new
dtstart~value_data~append('20100603T133000Z')

/* indicating the end of the vEvent */
dtend = .icalendar20_dtend~new
dtend~value_data~append('20100603T183000Z')

/* creation of a mandatory end property indicating the
end of an vEvent */
end_event = .icalendar20_end~new
end_event~value_data~append('VEVENT')

/* creation of a mandatory end property indicating the
end of a calendar */
end_calendar = .icalendar20_end~new
end_calendar~value_data~append('VCALENDAR')

/*collection of all objects representing iCalendar/hCalendar
 properties */
my_calendar = .array~of(begin_calendar, version, prodid,-
                        begin_event, dtstamp, uid, -
                        description, dtstart, dtend, -
                        end_event, end_calendar)

do property over my_calendar
   say property~makestring --return properties in iCalendar style
end

::requires 'icalendar_classes.rxj'
```

Code 29: Creation of a iCalendar.

The output of the iCalendar example in code 29 is shown in code 30.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
DTSTAMP:20100508T160202Z
UID:h0451119@wu.ac.at&20100508T160202
DESCRIPTION:A very important meeting.
DTSTART:20100603T133000Z
DTEND:20100603T183000Z
END:VEVENT
END:VCALENDAR
```

Code 30: Newly Created iCalendar.

In order to create an output which follows the rules of the hCalendar, the loop in the end of code 29 needs to be changed as depicted in code 31.

```
do property over my_calendar
   say property~makeHtml --return properties in hCalendar style
end
```

Code 31: Changed Loop of Code 29 for Creation of a hCalendar.

The corresponding new output after altering the loop of code 29 with the loop shown in code 31 is depicted in code 32. In order to understand the output, it must be mentioned that the properties `version`, `prodid`, `dtstamp` and `uid` are not defined in the hCalendar standard[137] and as a result, the method `makeHtml` cannot produce any hCalendar property for them.

```
<div class="vcalendar">


   <div class="vevent">


     <span class="description">A very important meeting.</span>
     <span class="dtstart">20100603T133000Z</span>
     <span class="dtend">20100603T183000Z</span>
   </div>
</div>
```

Code 32: New hCalendar Using Code 29 With The Loop From Code 31.

[137] Cf. chapter “2.2.4 hCalendar“ (p. 21 ff.) for further information.

4 Roundup and Outlook

The goal of this section of the paper is to sum up the content of the thesis. Finally, it tries to present prospects for the future.

4.1 Summary

The last part of this thesis tries to recapitulate the main points of the complete work, its use and the prospects for the future.

The vCard and the iCalendar (formerly vCalendar) formats are internationally known and standardized standards for the representation of persons or organizations and calendars in the case of the iCalendar standard.

The hCard and the hCalendar are so called microformats. The goal of microformats is to enrich web pages with machine processable information about the content of these pages. The hCard is based on the vCard standard whereas the hCalendar corresponds to the iCalendar standard.

The above mentioned standards have a host of different elements in common. These elements are called properties in the format specific terminology. All of these properties share a common structure which solely differs in the characteristics of the descriptive parts of each individual property. The structure of the xhtml/HTML based microformats (hCard, hCalendar) is more complex but can also use the classes which are designed for the vCard and iCalendar standard.

All of the above mentioned standards have many properties. The structure of the ooRexx classes for these properties differs in the number of the attributes which represent the property parameters in the respective vCard, hCard, iCalendar or hCalendar property. Hence, the use of a meta-programming approach reasonable and beneficial. In short, a meta-program is a program which creates a program. In this special case, the meta-programming approach allows defining all of the differing descriptive components of the properties dynamically while the program starts up and initializes the classes. The way this is achieved reduces the possibilities of errors and reduces repetitive

programming procedures. Moreover, this way of solving the problem allows to change the representation of the classes faster if needed.

The programming language which was used for the resolution of the programming tasks was ooRexx, a scripting language with an easy syntax and an excellent support for error detection.

BSF4ooRexx allowed to bridge ooRexx with Java. The `Reader`-class and the classes representing the properties of every single standard use this framework for using the DOM-parsing possibilities provided by Java[138]. What is more, ooRexx is able to use the complete `swing`-infrastructure of Java for the creation of a graphical user interface. Due to changes in the kernel of ooRexx version 4.0 and the adaption of BSF4ooRexx (formerly BSF4rexx) even allowed to integrate itself more into the Java environment.

The classes which are programmed for this thesis can be divided into three main parts. First, there is a class infrastructure representing all properties with their possible attributes. This part of all classes which are programmed for this thesis takes advantage of the above described meta-programming approach.

The second set of classes is responsible for reading the existing files which contain data in the respective data-formats. In order to retrieve the information of the hCard and hCalendar files, the `Reader` uses a non-validating Java-DOM-parser.

The last part is the graphical user interface. The GUI is designed to help the user to process data with the `Reader`-class and the classes designed to represent the properties of the vCard/hCard/iCalendar/hCalendar standards. Apart from this, some of the most prominent `swing` features are demonstrated and how they can be employed with ooRexx.

The most prominent features are the possibility to read vCard and iCalendar files and transform them to their microformat counterparts. The microformat output is a building block which can be embedded into existing web pages. The

[138] This statement refers to the point in time when this thesis was completed which was February 2010.

`Reader`-class also supports the extraction of multiple hCard or iCalendar events from a HTML file.

Due to the fact that the hCalendar does not support every single feature of the iCalendar, the conversion of iCalendar files into the hCalendar format are limited to `vevents`[139]. The conversion from hCalendars into iCalendars do not face such limitations. The transformation of vCard files into hCard files and the other way round are completely supported.

4.2 Prospects

The classes and this thesis replicate the status of the respective vCard, iCalendar, hCard and hCalendar formats at the moment of writing. However, the classes be used also in the future as the standards are widely used and newer versions of the standards will not replace the older version from one day to another.

During the completion of this Master thesis, the standard for the iCalendar has been modified. The used specifications for the vCard standard are also undergoing changes. There is already a draft for an update of the standard. This outline expires on November 10, 2010[140] and it features a couple of new properties[141].

Concerning the specifications of the microformats, the respective homepage also announces changes in the standards. Given the information provided on these web pages, the version 1.0.1 of the hCard and hCalendar version 1.0.1 are in preparation which are announced to address problems and introduce a further developed value-class-pattern.

In the case of the implementation of further iCalendar components for the hCalendar, these components must be implemented as well. The ooRexx class `Reader` in the file `reader.rxj` has a method called `ical_followNode()`. This method is responsible for walking all DOM-nodes. This method is already

[139] `vtodo`, `vjournal`, `vfreebusy`, `vtimezone` and `valarm` are not supported by hCalendar.
[140] This draft can be found at [PeRe10].
[141] Some of these new properties are `dday` for the day of death, `birth` for the place where a person is born or `sex` for the sex of a person. Due to the structure of the class collection representing a vCard, `vcard_classes.rxj`, the necessary adaptation can be achieved through adding the respective classes to the existing vCard classes.

capable of identifying a `vtodo`, `vjournal`, `vfreebusy`, `vtimezone` or `valarm`. However, no further actions are taken as these components are not specified in the version 1.0 of the hCalendar. Also, if the hCalendar allows more iCalendar properties, the `makeHtml`-method of the respective ooRexx class representing this property must be adopted.

Another interesting follow-on-project could be extending the functions of the existing GUI. Such a GUI could allow an easy, guided creation of business cards and calendars and/or the manipulation of already existing data.

5 Bibliography

[ASETAL09] Ashley, D.; Flatscher, R.; Hessling, M.; McGuire R.; Miesfeld M.; Peedin, L.; Tammer, R.; Wolfers, J., Open Object RexxTM Reference Version 4.0.0 Edition, 2009.

[Flat09] Flatscher, R.: The 2009 Edition of BSF4Rexx, 2009, http://wi.wu.ac.at/rgf/rexx/orx20/2009_orx20_BSF4Rexx-20091031-article.pdf, accessed on Feb. 1, 2010.

[HEROWO00] Le Hégaret, P., Robie J., Wood, L.: What is the Document Object Model?, 2000, http://www.w3.org/TR/DOM-Level-2-Core/introduction.html, accessed on Feb. 8, 2010.

[MAETAL08] Marchant, W.; Roth, W.; Singleton, G.; Wey, G. R.: Creating large documents with OpenOffice.org Writer, 2008, https://www.sun.com/offers/docs/creating_large_docs_OpenOffice.pdf, accessed on March 1, 2010.

[Ort07] Ortiz A.: An Introduction to Metaprogramming, 2007, http://m.linuxjournal.com/article/9604, accessed on Feb. 15, 2010.

[Pae09] Paenson, D.: Einführung für Studenten in die Techniken der Textverarbeitung anhand OpenOffice.org, 2009, http://de.openoffice.org/doc/howto_2_0/writer/ooo_fuer_studenten.pdf, accessed on March 1, 2010.

[PeRe10] Perreault, S.; Resnick, P.: vCard Format Specification draft-ietf-vcarddav-vcardrev-11, 2010, http://datatracker.ietf.org/doc/draft-ietf-vcarddav-vcardrev/, accessed on May 11, 2010.

[RFC2425] Dawson F.; Howes T.; Smith M.: A MIME Content-Type for Directory Information, 1998, http://tools.ietf.org/html/rfc2425, accessed on Feb. 2, 2010.

[RFC2426] Dawson, F.; Howes T.: vCard MIME Directory Profile, 1998, http://tools.ietf.org/html/rfc2426, accessed on Feb. 2, 2010.

[RFC4770] Jennings, C., Reschke, J.: vCard Extensions for Instant Messaging (IM), 2007, http://tools.ietf.org/html/rfc4770, accessed on Feb. 15, 2010.

[RFC5545] Desruisseaux B.: Internet Calendaring and Scheduling Core Object Specification (iCalendar), 2009, http://tools.ietf.org/html/rfc5545, accessed on Feb. 3, 2010.

[RUS02] Rusty Harold, E.: Trees, 2002, http://www.cafeconleche.org/books/xmljava/chapters/ch09s04.html, accessed on Feb. 8, 2010.

[W3AJ] Flatscher R.: AutoJava, n. d., http://wi.wu.ac.at/rgf/wu/lehre/autojava/material/foils/, accessed on Feb. 15, 2010.

[W3AL] n. a.: Open Source Initiative OSI - Apache License, Version 2.0: Licensing, 2004, http://www.opensource.org/licenses/apache2.0.php, accessed on Mar. 30, 2010.

[W3AW] Flatscher, R.: AutoWin, n. d., http://wi.wu.ac.at/rgf/wu/lehre/autowin/material/foils/, accessed on Feb. 15, 2010.

[W3AWP] n. a.: AbiWord Portable, n. d., http://sourceforge.net/projects/portableapps/files/AbiWord%20Portable, accessed on March 1, 2010.

[W3BSF] n. a.: Bean Scripting Framework, n. d., http://jakarta.apache.org/bsf/,

accessed on Feb. 1, 2010.
[W3BSR] Flatscher, R: BSF4ooRexx, n. d., http://wi.wu.ac.at:8002/rgf/rexx/bsf4oorexx/current/, accesses on Feb. 15, 2010.
[W3BT] n. a.: Bug Tracker, n. d., http://sourceforge.net/tracker/?atid=684730&group_id=119701&func=browse, accessed on Feb. 18, 2010.
[W3CDOM] n. a.: Document Object Model (DOM) Level 2 Core Specification, 2000, http://www.w3.org/TR/DOM-Level-2-Core/, accessed on April 14, 2010.
[W3EX] n. a.: PDF Import, n. d., http://extensions.services.openoffice.org/en/project/pdfimport, accessed on March 1, 2010.
[W3HC] Çelik, T.; Suda, B.: hCard 1.0, 2009, http://microformats.org/wiki/index.php?title=hcard&oldid=41500, accessed on Feb. 4, 2010.
[W3HCCH] n. a.: hCard cheatsheet, n. d., http://microformats.org/wiki/index.php?title=hcard-cheatsheet&oldid=41554, accessed on Feb. 5, 2010.
[W3HCEX] Çelik T., Suda B.: hCard examples, n. d., http://microformats.org/wiki/index.php?title=hcard-examples&oldid=40748, accessed on Feb. 5, 2010.
[W3HCL] Çelik T., Suda B.: hCalendar 1.0, n. d., http://microformats.org/wiki/index.php?title=hcalendar&oldid=41551, accessed on Feb. 5, 2010.
[W3HCLCH] n. a.: hCalendar cheatsheet, 2008, http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589, accessed on Feb. 5, 2010.
[W3HCLEX] Çelik T., King, R., Prilgrim, M.: hCalendar examples , 2008, http://microformats.org/wiki/index.php?title=hcalendar-examples&oldid=31482, accessed on Feb. 5, 2010.
[W3HTML4.1] Le Hors, A.; Jacobs, I.; Raggett, D.: HTML 4.01 Specification, 1999, http://www.w3.org/TR/1999/REC-html401-19991224/html40.txt, accessed on May 5, 2010.
[W3IntroHTML] Le Hors, A.; Jacobs, I.; Raggett, D.: Introduction to HTML 4, 1999, http://www.w3.org/TR/1999/REC-html401-19991224/intro/intro.html#h-2.2.1, accessed on May 5, 2010.
[W3JAVA] n. a.: The Java Programming Language, n. d., http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html, accessed on Feb. 1, 2010.
[W3JD] n. a.: Downloading Java, n. d., http://www.java.com, accessed on Feb. 15, 2010.
[W3JTU] n. a.: Java Tutorial, n. d., http://java.sun.com/docs/books/tutorial/ , accessed on Feb. 15, 2010.
[W3LF] n. a.: Liberation Fonts, n. d., https://fedorahosted.org/releases/l/i/liberation-fonts/, accessed on March 1, 2010.
[W3LT] n. a.: Language Tool, n. d., http://extensions.services.openoffice.org/en/project/languagetool, accessed on March 1, 2010.
[W3MI] : About Microformats, n. d., http://microformats.org/about, accessed on Feb. 4, 2010.
[W3ModXHTML] Altheim, M.; Boumphrey, F.; Dooley, S.; McCarron, S.;

Schnitzenbaumer, S.; Wugofski, T.: Modularization of XHTML™, Introduction, 2001, http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/introduction.html#s_intro_whatisxhtml, accessed on May 5, 2010.

[W3NC] n. a.: vCard and CardDAV (vcarddav), 2009, http://www.ietf.org/dyn/wg/charter/vcarddav-charter.html, accessed on Feb. 2, 2010.

[W3ODF] n. a.: Sun ODF Plugin for Microsoft Office, n. d., http://www.sun.com/software/star/odf_plugin/, accessed on March 1, 2010.

[W3OOO] n. a.: OpenOffice.org, n. d., http://download.openoffice.org/index.html, accessed on March 1, 2010.

[W3OOP] n. a.: OpenOffice.org Portable, n. d., http://sourceforge.net/projects/portableapps/files/OpenOffice.org%20Portable/, accessed on March 1, 2010.

[W3PARS] Ashley, D.; Idenburg, R.: SAX parser, n. d., http://oorexx.svn.sourceforge.net/viewvc/oorexx/incubator/orxutils/xml/xmlparser.cls?view=markup, accessed on Feb. 15, 2010.

[W3PRX] n. a.: ooRexx Project, n. d., http://sourceforge.net/projects/oorexx/, accessed Feb. 18, 2010.

[W3RD] n. a.: Download ooRexx, n. d., http://www.oorexx.org/download.html, accessed on Feb. 15, 2010.

[W3RF] n. a.: Rexx friendly editors, n. d., http://sourceforge.net/apps/mediawiki/oorexx/index.php?title=REXX_Friendly_Editors, accessed on March 1, 2010.

[W3RX] n. a.: About Open Object Rexx, n. d., http://www.oorexx.org/about.html, accessed on Feb. 1 2010.

[W3SWT] n. a.: SWT: The Standard Widget Toolkit, n. d., http://www.eclipse.org/swt/, accessed on May 10, 2010.

[W3SXML] n. a.: Introduction to XML, n. d., http://www.w3schools.com/xml/xml_whatis.asp, accessed on Feb. 8, 2010.

[W3SYN] n. a.: XML Syntax Rules, n. d., http://www.w3schools.com/xml/xml_syntax.asp, accessed on Feb. 8, 2010.

[W3TE] n. a.: Tails Export, n. d., https://addons.mozilla.org/de/firefox/addon/2240, accessed on March 1, 2010.

[W3TRE] Flatscher, R.: Classification Tree of Object Rexx, 6a, n. d., http://wi.wu.ac.at/rgf/wu/lehre/autojava/material/foils/ooRexx%5f5%2epdf, accessed on Feb. 17, 2010.

[W3VAL] n. a.: XML Validation, n. d., http://www.w3schools.com/xml/xml_dtd.asp, accessed on Feb. 8, 2010.

[W3VC] versit Consortium: vCard The Electronic Business CardVersion 2.1, 1996, http://www.imc.org/pdi/vcard-21.txt, accessed on Feb. 2, 2010.

[W3VCP] Çelik, T.; Ward, B.: Value Class Pattern, n. d., http://microformats.org/wiki/index.php?title=value-class-pattern&oldid=41539, accessed on Feb. 4, 2010.

[W3VIM] n. a.: VIM, n. d., http://www.vim.org/, accessed on March 1, 2010.

[W3VIMP] n. a.: VIM Portable, n. d., http://sourceforge.net/projects/portableapps/files/gVim%20Portable/, accessed on March 1, 2010.

[W3WM] n. a.: WinMerge, n. d., http://winmerge.org/downloads/?lang=en, accessed on March 1, 2010.

[W3WMP] n. a.: WinMerge Portable, n. d.,

http://sourceforge.net/projects/portableapps/files/WinMerge%20Portable/, accessed on March 1, 2010.

[W3XHTML1.1] Altheim, M.; McCarron, S.: XHTML™ 1.1 - Module-based XHTML, 2001, http://www.w3.org/TR/2001/REC-xhtml11-20010531/, accessed on May 5, 2010.

A Appendix

The following section consists of resources which were used for this of this thesis and information of the ooRexx classes.

A.1 Useful Software Resources

The completion of this thesis would have not been possible without the use of a huge variety of different programs. This part of the appendix documents these convenient helpers. Every single piece of software has the advantage that it is freeware or open source.

A.1.1 VIM (Vi IMproved)

VIM is a text editor which highlights the ooRexx syntax. It is available for a huge variety of operating systems including Unix, Windows and Macintosh. The editor can be downloaded from its homepage [W3VIM] in the download section.

For Windows users there is a useful portable version for USB sticks or other changeable medias.[W3VIMP]

A list of other editors which are “Rexx friendly” can be found on [W3RF].

A.1.2 WinMerge

Another open source tool which was during the completion of this thesis was WinMerge. This little open source tool allows to compare two different files and highlights the differences between them. This program was used in order to compare the vCard, iCalendar, hCard and hCalendar files which are generated by the program. For testing purposes, the output of the thesis program was read again and another output file was created (“round trip”). With the help of WinMerge, the resulting files were compared in order to proof whether the program yields a constant output or not. Regrettably, WinMerge is at the moment of writing (Feb. 2010) only available for Windows. The program (and again, a program version which can start under Windows from a USB-stick) can be found at the following links.[W3WM][W3WMP]

A.1.3 OpenOffice.org

This paper has been written completely in the OpenOffice.org swriter. OpenOffice.org is an open source office suite with a huge variety of functionality. This program is available for several operating systems. The swriter is the word processor of this suite. The author used the OpenOffice.org swriter features for creating the table of contents and the bibliography of this paper automatically and comfortably. What is more, the OpenOffice.org swriter supports saving files in the PDF-file-format (and with an add-on the import of PDFs as well.[142]) All in all, the complete range of functions is comparable to the Microsoft Office suite. The OpenOffice.org suite can be found at the following link (the second source directs to a version of OpenOffice.org that can start from a USB-stick under Windows).[W3OOO][W3OOP]

A couple of add-ons proofed to be quite helpful during the completion of this paper. The “LanguageTool”-extension is a language checker for several languages (including English, French and German.)[W3LT]

In order to get further information about using OpenOffice.org for (scientific) writing, the following papers are appealing:

“Creating large documents with OpenOffice.org Writer” by W. Roth, G. R. Singleton, G. Wey and W. T. Marchant.[MAETAL08]

“Einführung für Studenten in die Techniken der Textverarbeitung anhand OpenOffice.org ” by D. Paenson. (This paper is in German.)[Pae09]

A possible alternative for the OpenOffice.org Writer could be AbiWord. Compared to the OpenOffice.org suite, AbiWord is solely a word processing program. However, it offers a huge variety of functions as well. OpenOffice.org can be found at [W3AW] and as a version for Windows which runs from a USB-stick at [W3AWP].

[142] This add-on can be found here: [W3EX].

A.1.4 Sun ODF Plugin for Microsoft Office

The standard format for documents created by the OpenOffice.org suite (and other Open Source programs) correspond to the OASIS OpenDocument Format for Office Applications. The plugin allows MS Office users to open and save documents in the OpenDocument format.[W3ODF]

A.1.5 Liberation Fonts

The complete document uses the liberation fonts. This TrueType font family is also published as Open Source fonts. The special feature of this font family is that they are metric compatible with Microsoft's Arial, Times New Roman and Courier. The fonts can be downloaded in the web.[W3LF]

A.1.6 Tails Export

A very helpful Add-on for the Mozilla Firefox is Tails Export which allows to export embedded microformats. The link can be found in the bibliography. [W3TE]

A.2 Error in ooRexx Version 4.0.0

During the completion of this thesis, a mistake concerning the `changestr`-method was discovered by the author. This mistake appeared when the `changestr`-method – which alters string – was used in order to replace a backslash escaped character with a dummy character. This replacement was necessary for splitting up the data.

When the `changestr`-function replaces a `"00"X` it appends additionally the replacement minus one character at the end of the modified string. In the previous example, it adds the string "`12`" at the end.

```
/*ooRexx Version: Open Object Rexx Version 4.0.0,
Build date: Aug 15 2009*/

/*changestr error*/

a = "00"X || "abc"

b = a~changestr("00"X, "123")

say b -- expected output: 123abc
      --   actual output: 123abc12
/**/
```

Code 33: Error in ooRexx.

The problem has been reported to the Open Object Rexx Project and is fixed. [W3BT][143]

[143] The address of the ooRexx project itself is here [W3PRX].

A.3 The Classes at a Glance

The following part of this paper focuses on the source code of all created classes with their respective methods and attributes.

A.3.1 vCard, hCard, iCalendar and hCalendar

Definition: vCard [RFC2425][RFC2426][RFC4770], hCard [W3HC][W3HCCH], iCalendar [RFC5545], hCalendar [W3HCL][W3HCLCH]

Purpose: representing the properties with their parameters

Naming convention: Every property of the standards is represented by a class, the name of this class consists of a prefix (“icalendar20_” for iCalendar/hCalendar or “vcard30_” for vCard/hCard) and the name of the property as defined in the respective standard definition, a hyphen is replaced by an underscore.
E.g. `vcard30_sort_string` for the property `sort-string`

Important attributes: Every class has an array `value_data` for the values of the property. Moreover, every in the definition defined property parameter is also represented by an array bearing the name of the parameter.
E.g. `partstat` (for the iCalendar property `attendee`)

Important methods:

`makeString()` returns a string corresponding to the vCard/iCalendar definitions

`makeHtml()` returns a string corresponding to the hCard/hCalendar definitions

`fromString() class`: class method, takes a string in the vCare/iCalendar format as value, analyzes it and distributes its content to the respective attribute

`fromXml() class`: class method, takes a DOM-element-node as value, analyzes it and distributes its content to the respective attribute

A.3.1.1 The Classes in Detail

Table 2, 3, 4 and 5 show the classes of the standards with their attributes and methods. Table 1 explains the structure of these tables. First, there is always the name of a property as defined in the respective standard and optional comments. After this, the name of the class, the attributes and the methods are indicated.

`property name` *optional further explanations*		
`classname` (starting either with `icalendar20_` or `vCard30_`	`attributes` *optional further explanations*	`methods`

Table 1: Structure of Tables.

Overview of the Classes for the vCard Standard

Table 2 shows the details of the classes dedicated to represent the properties of the vCard standard version 3.0. This standard is defined in [RFC2425], [RFC2426] and [RFC4770].

Class Name	*Attributes*	*Methods*
superclass		
vcard30	attributes (class) values (class) classname group_name non_standard_attributes	correct_data (class) fromString (class) fromXML (class) init (class) leftSide (class) rightSide (class) unspecified_value (class) uv_followNode (class) value_class_pattern (class) vcp_followNode (class) init makeHTML makeString
adr *address, further details rfc 2426, p. 11, 31*		
vcard30_adr	type value_data	followNode (class) fromXML (class) init (class) makeHTML
agent *further details rfc 2426, p. 19, 33*		
vcard30_agent	ag_content has_vcard_end value value_data	agentfromXML (class) followNode (class) fromXML (class) init (class) vcard_control (class) makeHTML
bday *“birthday”, further details rfc 2426, p. 11, 31*		
vcard30_bday	value value_data	init (class)
begin *further details rfc 2426, p. 5, 30*		
vcard30_begin	value_data	fromXML (class) init (class) makeHTML
categories *further details rfc 2426, p. 20, 33*		
vcard30_categories	language value value_data	init (class) makeHTML

Class Name	Attributes	Methods
class *further details rfc 2426, p. 26, 35*		
vcard30_class	value_data	init (class)
email *further details rfc 2426, p. 15, 32*		
vcard30_email	type value_data	followNode (class) fromXML (class) init (class) specialFromXML (class) makeHTML
end *further details rfc 2426, p. 5, 30*		
vcard30_end	value_data	fromXML (class) init (class) makeHTML
fn *“formatted name”, further details rfc 2426, p. 8, 30*		
vcard30_fn	language value value_data	init (class)
geo *“geographic position”, further details rfc 2426, p. 16, 32*		
vcard30_geo	value_data	fromXML (class) followNode (class) init (class) makeHTML
impp *"instant messaging, presence protocol", further details rfc 4770, p. 1 ff.*		
vcard30_impp	type value_data	init (class) makeHTML
key *“encryption key”, further details rfc 2426, p. 26, 35*		
vcard30_key	encoding type value_data	fromXML (class) init (class) makeHTML
label *further details rfc 2426, p. 13, 31*		
vcard30_label	language type value value_data	init (class) followNode (class) fromXML (class) makeHTML
logo *further details rfc 2426, p. 18, 32*		
vcard30_logo	encoding type value value_data	fromXML (class) init (class) makeHTML

Class Name	Attributes	Methods
mailer *further details rfc 2426, p. 15, 32*		
vcard30_mailer	language value value_data	init (class)
n *“name”, further details rfc 2426, p. 9, 30*		
vcard30_n	language value value_data	followNode (class) fromXML (class) init (class) makeHTML
name *further details rfc 2426, p. 5, 30*		
vcard30_name	value_data	init (class) makeHTML,
nickname *further details rfc 2426, p. 9, 30*		
vcard30_nickname	language value value_data	init (class)
note *further details rfc 2426, p. 21, 33*		
vcard30_note	language value value_data	init (class)
org *"organisation", further details rfc 2426, p. 20, 33*		
vcard30_org	value value_data	followNode (class) fromXML (class) init (class) makeHTML
photo *further details rfc 2426, p. 10, 31*		
vcard30_photo	encoding type value value_data	fromXML (class) init (class) makeHTML
prodid *“product identification number”, further details rfc 2426, p. 21, 34*		
vcard30_prodid	value_data	init (class) makeHTML
profile *further details rfc 2426, p. 5, 30*		
vcard30_profile	value_data	init (class) makeHTML,

Class Name	Attributes	Methods
rev *"last review", further details rfc 2426, p. 22, 34*		
vcard30_rev	value value_data	init (class)
role *“person's role in an organization”, further details rfc 2426, p. 18, 32*		
vcard30_role	language value value_data	init (class)
sort-string *"sorting indicator", further details rfc 2426, p. 22, 34*		
vcard30_sort_string	value value_data	init (class)
sound *further details rfc 2426, p. 23, 34*		
vcard30_sound	encoding type value value_data	fromXML (class) init (class) makeHTML
source *further details rfc 2426, p. 5, 30*		
vcard30_source	context value value_data	init (class) makeHTML
tel *“telephone”, further details rfc 2426, p. 14, 31*		
vcard30_tel	type value_data	followNode (class) fromXML (class) init (class)
title *"person's title", further details rfc 2426, p. 17, 32*		
vcard30_title	language value value_data	init (class)
tz *"time zone", further details rfc 2426, p. 16, 32*		
vcard30_tz	value_data	init (class)
uid *further details rfc 2426, p. 24, 35*		
vcard30_uid	value_data	init (class)
url *further details rfc 2426, p. 25, 35*		
vcard30_url	value_data	init (class) specialFromXML (class) makeHTML

Class Name	*Attributes*	*Methods*
`version` *further details rfc 2426, p. 25, 35*		

`vcard30_version`	`value_data`	`init (class)`
		
		`makeHTML`

`x-property` *further details rfc 2426, p. 29*		

`vcard30_x_property`	`value_data`	`init (class)`
		`leftside (class)`
		
		`makeHTML`

Table 2: Complete List of ooRexx Classes Representing vCard Properties.

Overview of the Classes for the hCard Standard

Table 3 shows the details of the classes dedicated to represent the properties of the hCard standard version 1.0. This standard is defined at [W3HCCH] and [W3HC]. Further information about the purpose of every single property can be found in the [RFC2426].

Class Name	***Attributes***	***Methods***
Superclass		

`vcard30`	`attributes (class)`
	`values (class)`
	
	`classname`
	`group_name`
	`non_standard_attributes`
		`correct_data (class)`
		`fromString (class)`
		`fromXML (class)`
		`init (class)`
		`leftSide (class)`
		`rightSide (class)`
		`unspecified_value (class)`
		`uv_followNode (class)`
		`value_class_pattern (class)`
		`vcp_followNode (class)`
		
		`init`
		`makeHTML`
		`makeString`

`adr` *address, further details rfc 2426, p. 11, 31 and* [W3HCCH]		

`vcard30_adr`	`type`
	`value_data`
		`followNode (class)`
		`fromXML (class)`
		`init (class)`
		
		`makeHTML`

`agent` *further details rfc 2426, p. 19, 33 and* [W3HCCH]		

`vcard30_agent`	`ag_content`
	`has_vcard_end`
	`value_data`
		`agentfromXML (class)`
		`followNode (class)`
		`fromXML (class)`
		`init (class)`
		`vcard_control (class)`
		
		`makeHTML`

`bday` *“birthday”, further details rfc 2426, p. 11, 31 and* [W3HCCH]		
`vcard30_bday`	`value` `value_data`	`init (class)`
`begin` *further details rfc 2426, p. 5, 30 and* [W3HCCH]		

`vcard30_begin`	`value_data`	`fromXML`
		`init (class)`
		
		`makeHTML`

`category` *further details rfc 2426, p. 20, 33 and* [W3HCCH]		

`vcard30_categories`	`value_data`	`init (class)`
		
		`makeHTML`

Class Name	Attributes	Methods
class *further details rfc 2426, p. 26, 35 and* [W3HCCH]		
vcard30_class	value_data	init (class)
email *further details rfc 2426, p. 15, 32 and* [W3HCCH]		
vcard30_email	type value_data	followNode (class) fromXML (class) init (class) specialFromXML (class) makeHTML
end *further details rfc 2426, p. 5, 30 and* [W3HCCH]		
vcard30_end	value_data	fromXML (class) init (class) makeHTML
fn *“formatted name”, further details rfc 2426, p. 8, 30 and* [W3HCCH]		
vcard30_fn	value_data	init (class)
geo *“geographic position”, further details rfc 2426, p. 16, 32 and* [W3HCCH]		
vcard30_geo	value_data	fromXML (class) followNode (class) init (class) makeHTML
key *“encryption key”, further details rfc 2426, p. 26, 35 and* [W3HCCH]		
vcard30_key	value_data	fromXML (class) init (class) makeHTML
label *further details rfc 2426, p. 13, 31 and* [W3HCCH]		
vcard30_label	type value_data	followNode (class) fromXML (class) init (class) makeHTML
logo *further details rfc 2426, p. 18, 32 and* [W3HCCH]		
vcard30_logo	value_data	fromXML (class) init (class) makeHTML
mailer *further details rfc 2426, p. 15, 32 and* [W3HCCH]		
vcard30_mailer	value_data	init (class)

Class Name	Attributes	Methods
n *“name”, further details rfc 2426, p. 9, 30 and* [W3HCCH]		
vcard30_n	value_data	followNode (class) fromXML (class) init (class) makeHTML
nickname *further details rfc 2426, p. 9, 30 and* [W3HCCH]		
vcard30_nickname	value_data	init (class)
note *further details rfc 2426, p. 21, 33 and* [W3HCCH]		
vcard30_note	value_data	init (class)
org *"organisation", further details rfc 2426, p. 20, 33 and* [W3HCCH]		
vcard30_org	value_data	followNode (class) fromXML (class) init (class) makeHTML
photo *further details rfc 2426, p. 10, 31 and* [W3HCCH]		
vcard30_photo	value_data	fromXML (class) init (class) makeHTML
rev *"last review", further details rfc 2426, p. 22, 34 and* [W3HCCH]		
vcard30_rev	value_data	init (class)
role *“person's role in an organization”, further details rfc 2426, p. 18, 32 and* [W3HCCH]		
vcard30_role	value_data	init (class)
sort-string *"sorting indicator", further details rfc 2426, p. 22, 34 and* [W3HCCH]		
vcard30_sort_string	value_data	init (class)
sound *further details rfc 2426, p. 23, 34 and* [W3HCCH]		
vcard30_sound	value_data	fromXML (class) init (class) makeHTML
tel *“telephone”, further details rfc 2426, p. 14, 31 and* [W3HCCH]		
vcard30_tel	value_data	followNode (class) fromXML (class) init (class)
title *"person's title", further details rfc 2426, p. 17, 32 and* [W3HCCH]		
vcard30_title	value_data	init (class)

Class Name	*Attributes*	*Methods*
`tz` *"time zone", further details rfc 2426, p. 16, 32 and* [W3HCCH]		
`vcard30_tz`	`value_data`	`init (class)`
`uid` *further details rfc 2426, p. 24, 35 and* [W3HCCH]		
`vcard30_uid`	`value_data`	`init (class)`
`url` *further details rfc 2426, p. 25, 35 and* [W3HCCH]		

`vcard30_url`	`value_data`	`init (class)`
		`specialFromXML (class)`
		
		`makeHTML`

Table 3: Complete List of ooRexx Classes Representing hCard Properties.

Overview of the Classes for the iCalendar Standard

Table 4 shows the details of the classes dedicated to represent the properties of the iCalendar standard version 2.0. This standard is defined in the [RFC5545].

Class Name	*Attributes*	*Methods*
Superclass		
icalendar20	attributes (class) values (class) classname group_name, non_standard_attributes	correct_data (class) fromString (class) fromXML (class) init (class) leftSide (class) rightSide (class), unspecified_value (class) uv_followNode (class) value_class_pattern (class) vcp_followNode (class) init makeString makeHTML
action *further details rfc 5545, p. 132 – 133*		
icalendar20_action	value_data	init (class) makeHTML
attach *„attachement“, further details rfc 5545, p. 80 – 81*		
icalendar20_attach	encoding fmttype *(“format type” RFC 5545, Section 3.2.8)* value value_data	init (class)
attendee *further details rfc 5545, p. 107 – 109*		
icalendar20_attendee	cutype *(“calendar user type” RFC 5545, Section 3.2.3)* cn *(“common name” RFC 5545, Section 3.2.2)* delegated_to delegated_from dir *(“directory entry reference” RFC 5545, Section 3.2.6)* language member partstat *(“participation status” RFC 5545, Section 3.2.12)* role rsvp *(“rsvp expectation” RFC 5545, Section 3.2.17)* sent_by value_data	init (class) followNode (class) fromXML (class)
begin *further details rfc 5545, p. 50*		
icalendar20_begin	value_data	fromXML (class) init (class) makeHTML

Class Name	Attributes	Methods
calscale *„calendar scale“ further details rfc 5545, p. 76 – 77*		
icalendar20_calscale	value_data	init (class) makeHTML
categories *further details rfc 5545, p. 81 – 82*		
icalendar20_categories	language value_data	init (class)
class *further details rfc 5545, p. 82 – 83*		
icalendar20_class	value_data	init (class)
comment *further details rfc 5545, p. 83 – 84*		
icalendar20_comment	altrep *(“alternate text representation” RFC 5545, Section 3.2.1)* language value_data	init (class)
completed *further details rfc 5545, p. 94 – 95*		
icalendar20_completed	value_data	init (class) makeHTML
contact *further details rfc 5545, p. 109 – 111*		
icalendar20_contact	altrep *(“alternate text representation” RFC 5545, Section 3.2.1)* language value_data	init (class)
created *further details rfc 5545, p. 136*		
icalendar20_created	value_data	init (class)
description *further details rfc 5545, p. 84 – 85*		
icalendar20_description	altrep *(“alternate text representation” RFC 5545, Section 3.2.1)* language value_data	init (class)
dtend *“date-time end”, further details rfc 5545, p. 95 – 96*		
icalendar20_dtend	tzid value value_data	init (class) value_class_pattern (class)

Class Name	Attributes	Methods
dtstamp *“date-time end”, further details rfc 5545, p. 137 – 138*		
icalendar20_dtstamp	value_data	init (class value_class_pattern (class) makeHTML
dtstart *“date-time start”, further details rfc 5545, p. 97 – 98*		
icalendar20_dtstart	tzid value value_data	init (class) value_class_pattern (class)
due *further details rfc 5545, p. 96 – 97*		
icalendar20_due	tzid value value_data	init (class) makeHTML
duration *further details rfc 5545, p. 99*		
icalendar20_duration	value_data	init (class)
end *further details rfc 5545, p. 50*		
icalendar20_end	value_data	fromXML (class) init (class) makeHTML
exdate *„exception date“, further details rfc 5545, p. 118 – 119*		
icalendar20_exdate	tzid value value_data	
exrule *„exception rule“, further details rfc 2445, p. 114 – 115*		
icalendar20_exrule	value_data	init (class) makeHTML
freebusy *further details rfc 5545, p. 100 – 101*		
icalendar20_freebusy	fbtype value_data	init (class) makeHTML
geo *further details rfc 5545, p. 85 – 87*		
icalendar20_geo	value_data	followNode (class) fromXML (class) init (class) makeHTML
last-modified *further details rfc 5545, p. 138*		
icalendar20_last_modified	value_data	init (class)

Class Name	Attributes	Methods
location *further details rfc 5545, p. 87 – 88*		
icalendar20_location	altrep *(“alternate text representation” RFC 5545, Section 3.2.1)* language value_data	init (class)
method *further details rfc 5545, p. 77 – 78*		
icalendar20_method	value_data	init (class) makeHTML
organizer *further details rfc 5545, p. 111 – 112*		
icalendar20_organizer	cn *(“common name” RFC 5545, Section 3.2.2)* dir *(“directory entry reference” RFC 5545, Section 3.2.6)* language sent_by value_data	init (class)
percent-complete *further details rfc 5545, p. 88 – 89*		
icalendar20_percent_complete	value_data	init (class) makeHTML
priority *further details rfc 5545, p. 89 – 90*		
icalendar20_priority	value_data	init (class)
prodid *“product-id”, further details rfc 5545, p. 78 – 79*		
icalendar20_prodid	encoding value type value_data	init (class) makeHTML
rdate *“recurrence date”, further details rfc 5545, p. 120 – 121*		
icalendar20_rdate	tzid value value_data	init (class)
recurrence-id *further details rfc 5545, p. 112 – 114*		
icalendar20_recurrence_id	range tzid value value_data	init (class) makeHTML
related-to *further details rfc 5545, p. 115 – 116*		
icalendar20_related_to	reltype value_data	init (class)

Class Name	Attributes	Methods
repeat *further details rfc 5545, p. 133*		
icalendar20_repeat	value_data	init (class) makeHTML
request-status *further details rfc 5545, p. 141 – 143*		
icalendar20_request_status	language value_data	init (class) makeHTML
resources *further details rfc 5545, p. 91*		
icalendar20_resources	altrep *(“alternate text representation” RFC 5545, Section 3.2.1)*, language value_data	init (class)
rrule *“recurrence rule”, further details rfc 5545, p. 122 – 132*		
icalendar20_rrule	value_data	init (class)
sequence *further details rfc 5545, p. 138 – 139*		
icalendar20_sequence	value_data	init (class)
status *further details rfc 5545, p. 92 – 93*		
icalendar20_status	value_data	init (class)
summary *further details rfc 5545, p. 93 – 94*		
icalendar20_summary	altrep *(“alternate text representation” RFC 5545, Section 3.2.1)* language value_data	init (class)
transp *“transparency”, further details rfc 5545, p. 101 – 102*		
icalendar20_transp	value_data	init (class)
trigger *further details rfc 5545, p. 133 – 135*		
icalendar20_trigger	related value value_data	init (class) makeHTML
tzid *further details rfc 5545, p. 102 – 103*		
icalendar20_tzid	value_data	init (class) makeHTML

Class Name	Attributes	Methods
tzname *"timezone-id", further details rfc 5545, p. 103 – 104*		
icalendar20_tzname	language value_data	init (class) makeHTML
tzoffsetfrom *"timezone, offset from", further details rfc 5545, p. 104 - 105*		
icalendar20_tzoffsetfrom	value_data	init (class) makeHTML
tzoffsetto *"timezone, offset to", further details rfc 5545, p. 105 – 106*		
icalendar20_tzoffsetto	value_data	init (class) makeHTML
tzurl *"tz-url", further details rfc 5545, p. 106*		
icalendar20_tzurl	encoding type value value_data	init (class) makeHTML
uid *further details rfc 5545, p. 117 – 118*		
icalendar20_uid	value_data	init (class) makeHTML
url *further details rfc 5545, p. 116 – 117*		
icalendar20_url	value_data	init (class) makeHTML
version *further details rfc 5545, p. 79 – 80*		
icalendar20_version	value_data	init (class) makeHTML
x-property *further details rfc 5545, p. 9*		
icalendar20_x_property	value_data	init (class) leftside (class) makeHTML

Table 4: Complete List of ooRexx Classes Representing iCalendar Properties.

Overview of the Classes for the hCalendar Standard

Table 5 shows the details of the classes dedicated to represent the properties of the hCalendar standard version 1.0. The definition of the standard can be found at [W3HCL] and [W3HCLCH]. Further information about the purpose of every single property can be found in the [RFC5545].

Classname	***Attributes***	***Methods***
Superclass		
icalendar20	attributes (class) values (class) classname group_name, non_standard_attributes	correct_data (class) fromString (class) fromXML (class) init (class) leftSide (class) rightSide (class), unspecified_value (class) uv_followNode (class) value_class_pattern (class) vcp_followNode (class) init makeString makeHTML
Classname	**Attributes**	**Methods**
attach *further details rfc 5545, p. 80 – 81 and [w3hclch].*		
icalendar20_attach	value_data	init (class),
attendee *further details rfc 5545, p. 107 – 109 and [w3hclch].*		
icalendar20_attendee	partstat (“participation status” RFC 5545, Section 3.2.12) role value_data	followNode (class) fromXML (class) init (class)
begin *further details rfc 5545, p. 50 and [w3hclch].*		
icalendar20_begin	value_data	fromXML (class) init (class) makeHTML
category *further details rfc 5545, p. 81 – 82 and [w3hclch].*		
icalendar20_categories	value_data	init (class)
class *further details rfc 5545, p. 82 – 83 and [w3hclch].*		
icalendar20_class	value_data	init (class)
comment *further details rfc 5545, p. 83 – 84 and [w3hclch].*		
icalendar20_comment	value_data	init (class)

Classname	Attributes	Methods
contact *further details rfc 5545, p. 109 – 111 and [w3hclch].*		
icalendar20_contact	value_data	init (class)
created *further details rfc 5545, p. 136 and [w3hclch].*		
icalendar20_created	value_data	init (class)
description *further details rfc 5545, p. 84 – 85 and [w3hclch].*		
icalendar20_description	value_data	init (class)
dtend *“date-time end” further details rfc 5545, p. 95 – 96 and [w3hclch].*		
icalendar20_dtend	value_data	init (class) value_class_pattern (class),
dtstart *“date-time start”, further details rfc 5545, p. 97 – 98 and [w3hclch].*		
icalendar20_dtstart	value_data	init (class) value_class_pattern (class)
duration *further details rfc 5545, p. 99 and [w3hclch].*		
icalendar20_duration	value_data	init (class)
end *further details rfc 5545, p. 50 and [w3hclch].*		
icalendar20_end	value_data	fromXML (class) init (class) makeHTML
exdate *“exception date/time”, further details rfc 5545, p. 118 – 119 and [w3hclch].*		
icalendar20_exdate	value_data	init (class)
geo *further details rfc 5545, p. 85 – 87 and [w3hclch].*		
icalendar20_geo	value_data	followNode (class) fromXML (class) init (class) makeHTML
last-modified *further details rfc 5545, p. 138 and [w3hclch].*		
icalendar20_last_modified	value_data	init (class)
location *further details rfc 5545, p. 87 – 88 and [w3hclch].*		
icalendar20_location	value_data	init (class)
organizer *further details rfc 5545, p. 111 – 112 and [w3hclch].*		
icalendar20_organizer	value_data	init (class)

Classname	Attributes	Methods
priority *further details rfc 5545, p. 89 – 90 and [w3hclch].*		
icalendar20_priority	value_data	init (class)
rdate *“recurrence date”, further details rfc 5545, p. 120 – 121 and [w3hclch].*		
icalendar20_rdate	value_data	init (class)
related-to *further details rfc 5545, p. 115 – 116 and [w3hclch].*		
icalendar20_related_to	value_data	init (class)
resources *further details rfc 5545, p. 91 and [w3hclch].*		
icalendar20_resources	value_data	init (class)
rrule *“recurrence rule”, further details rfc 5545, p. 122 – 132 and [w3hclch].*		
icalendar20_rrule	value_data	init (class)
sequence *further details rfc 5545, p. 138 – 139 and [w3hclch].*		
icalendar20_sequence	value_data	init (class)
status *further details rfc 5545, p. 92 – 93 and [w3hclch].*		
icalendar20_status	value_data	init (class)
summary *further details rfc 5545, p. 93 – 94 and [w3hclch].*		
icalendar20_summary	value_data	init (class)
transp *“transparency”, further details rfc 5545, p. 101 – 102 and [w3hclch].*		
icalendar20_transp	value_data	init (class)
url *further details rfc 5545, p. 116 – 117 and [w3hclch].*		
icalendar20_url	value_data	init (class) makeHTML

Table 5: Complete List of ooRexx Classes Representing hCalendar Properties.

A.3.1.2 vcard_classes.rxj

```
#!/usr/bin/rexx
/*
   Name:    vcard_classes.rxj
   Purpose: Collection of classes concerning vCard/hCard
   Author:  Johannes Paul Bielohaubek
   Date:    2010-05-04
   Version: 1.0.2
   License:

    ------------------------ Apache Version 2.0 license -------------------------
       Copyright (C) 2010 Johannes Paul Bielohaubek

       Licensed under the Apache License, Version 2.0 (the "License");
       you may not use this file except in compliance with the License.
       You may obtain a copy of the License at

           http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
    -------------------------------------------------------------------------
*/
/*
========================CLASSES=================================
================================================================
CLASS DEFINITION OF VCARD VERSION 3.0
*/

::class vcard30 public
::attribute attributes class --attribute used in subclasses
::attribute values class --attribute used in subclasses
::attribute non_standard_attributes --non standard attributes, x-attributes
                --as defined in RFC 2425, p. 4
::attribute classname
::attribute group_name


/*
----------------------------------------------------------------
init method that creates maker and setter methods for attributes
*/
::method init class

   if self~id="VCARD30" then return

   do name over self~attributes
      self~define(name, makegetter(name))
      self~define(name||'=', makesetter(name))
   end

   do name over self~values
      self~define(name, makegetter(name))
      self~define(name||'=',makesetter(name))
   end

----------------------------------------------------------------

::method init

   attributes = self~class~attributes --assigns newly created values
   values = self~class~values --assigns newly created values

   do name over attributes
      self~send(name||'=',.array~new)
      --sets initial attributes-value to an array
   end

   do name over values
      self~send(name||'=',.array~new) --sets initial values-value to an array
   end

   self~non_standard_attributes = .directory~new

   self~group_name = ''

   self~classname = self~class~id~substr(9)~changestr('_','-')
   --creates classname out of the id of a class

/*
----------------------------------------------------------------
method reads in data and distributes it to according classes
*/
::method fromstring class
   use arg line

   o = self~new
```

```
   right_pos = line~verify(";:",M) --defines position of separator
   if right_pos < 2 then iterate --exception for faulty data
   grouping = line~verify(".",M) --checks if properties are grouped
   if grouping > right_pos then --checks if grouping available
      do
      classname = left(line,right_pos-1)
      group_name = ""
      end
   else
      do
      classname = substr(line,grouping+1, right_pos - grouping - 1)
      if grouping <> 0 then group_name = left(line,grouping-1)
      else group_name = left(line,grouping)
      end

   o~classname = classname
   o~group_name = group_name

   line = line~changestr("\:", "01"X) --replaces backslash escaped ":"

   --replaces non backslash escaped ":" within quotes
   tmp_store = .array~new
   loop while line <> ""
      parse var line part '"' line
      tmp_store~append(part)
   end

   n=2

   if tmp_store~items > 1 then
   do
      do while n <= tmp_store~items
         tmp_store[n] = '"' ||  tmp_store[n]-
               ~changestr(":","02"X) || '"'
         n = n + 2
      end
      n = 2
   end


   do i over tmp_store
      line = line || i
   end


   parse var line left ":" right --divides into a attributes and values part
   left = left~upper~changestr("01"X,"\:")~changestr("02"X,":") --back to orignal values
   right = right~changestr("01"X,"\:")~changestr("02"X,":")

   self~leftside(o, left)
   self~rightside(o, right)
return o


/*
----------------------------------------------------------------------
method processing left side of read-in line
*/
::method leftside class
use arg o left

   tmp1 = left~changestr("\;","01"X)
   loop while tmp1 <> "" --assigns boolean values to attributes
   parse var tmp1 leftside ";" tmp1
      do
      parse var leftside denominator "=" denominations
      if o~class~attributes~hasitem(denominator) then
         do
            loop while denominations <> ""
               parse var denominations tmp2 "," denominations
               o~send(denominator)~append(tmp2)
            end

         end
      if denominator~left(2) = "X-" then
         do
            o~non_standard_attributes[denominator] = denominations
         end
      end
   end

return o


/*
----------------------------------------------------------------------
method processing right side of read-in line
*/
::method rightside class
use arg o right
   tmp2 = right~changestr("\;","01"X)
   loop while tmp2 <> "" --assigns boolean values to attributes
```

```
   parse var tmp2 rightside ";" tmp2
      o~value_data~append(rightside~changestr("01"X,"\;"))
   end
return o

/*
---------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no group names in hCard format allowed

if classname <> 'CLASSNAME' then o~classname = classname~changestr("_","-")

o~value_data~append(self~correct_data(node, .true))

return o

/*
---------------------------------------------------------------------
method processing xml-data in a multi-value case
method used to get text contents even when the tag type would
indicate to take the value from another source
*/
::method specialfromxml class
use arg node classname

o = self~new
o~group_name = ''

atts = node~getAttributes --no group names in hCard format allowed

o~classname = classname~changestr("_","-")

o~value_data~append(extended_strip(node~getTextContent)) --only text
            -- contents allowed

return o


/*
---------------------------------------------------------------------
method retrieves correct property value from the dom-node according
to the hcard-specifications on http://microformats.org/wiki/hcard
*/
::method correct_data class
use arg node us_value --us_value indicates if method should
   --look for unspecified values as defines at
   --http://microformats.org/wiki/hcard

dir = .directory~new --directory that stores attributes from nodes

atts = node~getAttributes

do i=0 to atts~getLength-1 --iterates over all attributes
   att=atts~item(i) --gets attribute node
   dir[att~getName~upper] = att~getValue
end

select --chooses the relevant data according to the tag type
when node~getNodeName = 'abbr' then correct_data = extended_strip(dir[TITLE])
when node~getNodeName = 'a' then correct_data = extended_strip(dir[HREF])
when node~getNodeName = 'img' then correct_data = extended_strip(dir[SRC])
when node~getNodeName= 'object' then correct_data = extended_strip(dir[DATA])
otherwise
   do
   if us_value = .true then correct_data = self~unspecified_value(node)
   else correct_data = extended_strip(node~getTextContent)
   end
end

--value class pattern rule http://microformats.org/wiki/value-class-pattern
--checks if child elements with the attribute 'CLASS="VALUE"' exists
Value_Class_Pattern = self~value_class_pattern(node)
if Value_Class_Pattern~length > 0 then correct_data = Value_Class_Pattern

return correct_data

/*
---------------------------------------------------------------------
method retrieves correct property value given that the value is not
specified
*/
::method unspecified_value class
use arg node

uv=''

if node~hasChildNodes then
   do
      children=node~getChildNodes
```

```
      loop i=0 to children~length-1 --goes over all child nodes
         new_uv = self~uv_follownode(children~item(i))
         uv = uv || new_uv
      end

   end
else uv = node~getTextContent

uv =  extended_strip(uv) --removes characters that are not needed

return uv

/*
-----------------------------------------------------------------------
method walks the document tree recursively looking for text values
outside an element having 'type' as an xml-'class'-attribute
*/
::method uv_followNode class
  use arg node
uv = ''
if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
   do
      atts = node~getAttributes

      if atts~getLength>0 then
         do
            do i=0 to atts~getLength-1
               att=atts~item(i)
               if att~getName="class" then --looks for the attribute type that is used
                        --to indicate hcalendar/hcard values
                  do
                     if att~getValue~wordpos('type') = 0 then
                        do
                           uv = uv || node~getTextContent
                        end
                     else nop
                  end
               else nop
            end
         end
      else nop
   end
else uv = uv || node~getTextContent

if node~hasChildNodes then
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~uv_followNode(children~item(i), uv)
      end
   end
else nop

return uv

/*
-----------------------------------------------------------------------
method retrieves correct property value given that child nodes
with the 'value' attribute exist
*/
::method value_class_pattern class
use arg node

vcp = ''
if node~hasChildNodes then
   do
   children=node~getChildNodes

   loop i=0 to children~length-1 --goes over all child nodes
      value = self~vcp_follownode(children~item(i))
      vcp = vcp || value
   end

   end

return vcp

/*
-----------------------------------------------------------------------
method walks over direct child nodes having the 'class'-attribute
'value'
*/
::method vcp_follownode class
use arg node

value = ''
if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
--cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
   do
      atts = node~getAttributes
```

```
      if atts~getLength>0 then
         do i=0 to atts~getLength-1
            att=atts~item(i)
            if att~getName='class' & att~getvalue = 'value' then
            --looks for the attribute type that is used
                     --to indicate hcalendar/hcard values
               do
                  dir = .directory~new --directory that stores attributes from nodes
                  do i=0 to atts~getLength-1 --iterates over all attributes
                     att=atts~item(i) --gets attribute node
                     dir[att~getName~upper] = att~getValue
                  end

                  select --chooses the relevant data according to the tag type
                  when node~getNodeName = 'abbr' then value = extended_strip(dir[TITLE])
                  when node~getNodeName = 'a' then value = extended_strip(dir[HREF])
                  when node~getNodeName = 'img' then value = extended_strip(dir[SRC])
                  when node~getNodeName= 'object' then value = extended_strip(dir[DATA])
                  otherwise value = extended_strip(node~getTextContent)
                  end
               end
            else nop
         end
      else nop
   end
else nop

return value

/*
---------------------------------------------------------------------
method creates string
*/
::method makestring
expose attributes values

if self~group_name <> "" then
   do
   tmpstr = self~group_name || "." || self~classname
   end
else
   do
   tmpstr=self~classname
   end

--non standard attributes, ordered alphabetically
tmpArr = self~non_standard_attributes~allindexes-
      ~sortwith(.caselesscomparator~new)
do entry over tmpArr
   if self~non_standard_attributes[entry] <> "" then
   do
   tmpstr = tmpstr || ";" || entry || "=" || -
      self~non_standard_attributes[entry]
   end
end


--standard attributes
do attribute over self~class~attributes
if attribute <> 'group_name' then
do
   if self~send(attribute)~items<>0 then
   do
      do prop_attr over self~send(attribute)
         tmpstr=tmpstr ||  ";" || attribute -
         || "=" || prop_attr
      end
   end
end
end

tmpstr=tmpstr||":"

-----creation of output lines, insertion of ";" where needed--------

i = self~send(value_data)~items --counts number of values
j=1

do prop_val over self~send(value_data)
   if j < i then
   do
      tmpstr=tmpstr || prop_val || ";"
      j = j + 1
   end
   else
   do
      tmpstr=tmpstr || prop_val
      j = j + 1
   end
end
```

```
------------folding of lines----------------------------------------

pos = 75
do while pos < tmpstr~length
   tmpstr = tmpstr~insert("0A"X || " ", pos)
   pos = pos + 77
end

return tmpstr



/*
--------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || -
      '">'


--checks if attribute is available and accesses its values
do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || '<span class="' || -
         escape(attribute~lower) || '">' || -
         escape(prop_attr~lower) -
         || '</span>'
      end
   end

end

--adds property values, creation of output lines, insertion of ";"
--where needed

i = self~send(value_data)~items --counts number of values
j=1

do prop_val over self~send(value_data)
   if j < i then
   do
      tmpstr=tmpstr || escape(prop_val) || ";"
      j = j + 1
   end
   else
   do
      tmpstr=tmpstr || escape(prop_val)
      j = j + 1
   end
end

tmpstr = tmpstr ||'</span>'

return tmpstr


/*
====================================================================
CLASS DEFINITION OF ADR
FURTHER DETAILS RFC 2426, P. 11, 31
*/

::class vcard30_adr public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
--------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no groups in hcard
o~classname= classname~changestr("_","-")

if node~hasChildNodes then
do
   children=node~getChildNodes --get NodeList
```

```
   loop i=0 to children~length-1 --0-based indexes! */
      self~followNode(children~item(i), o)
   end
end


do i=1 to 7 by 1
   if o~value_data[i] = .nil then  o~value_data[i] = ""
end

return o

/*
----------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o

   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then --checks if node has attributes
            do
               do i=0 to atts~getLength-1 --goes over all attributes
                  att=atts~item(i)  --defines currently processed attribute
                  if att~getName="class" then
                     select -- distributes the values
                     when att~getValue~upper~wordpos('TYPE') <> 0 -
                       then o~type~append(self~correct_data(node, .false)~upper)
                     when att~getValue~wordpos('post-office-box') <> 0 -
                       then o~value_data[1] = self~correct_data(node, .true)
                     when att~getValue~wordpos('extended-address') <> 0 -
                       then o~value_data[2] = self~correct_data(node, .true)
                     when att~getValue~wordpos('street-address') <> 0 -
                       then o~value_data[3] = self~correct_data(node, .true)
                     when att~getValue~wordpos('locality') <> 0 -
                       then o~value_data[4] = self~correct_data(node, .true)
                     when att~getValue~wordpos('region') <> 0 -
                       then o~value_data[5] = self~correct_data(node, .true)
                     when att~getValue~wordpos('postal-code') <> 0 -
                       then o~value_data[6] = self~correct_data(node, .true)
                     when att~getValue~wordpos('country-name') <> 0 -
                       then o~value_data[7] = self~correct_data(node, .true)
                     otherwise say 'attribute unknown to property'
                     end
               end

            end
      end


if node~hasChildNodes then
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~followNode(children~item(i), o)
      end
   end

return o


/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<div class="' || escape(self~classname~lower) || -
      '">'  || "0A"X

do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || tab || tab || '<span class="' || -
         escape(attribute~lower) || '">' || escape(prop_attr~lower) -
         || '</span>'|| "0A"X
      end
   end

end

--checks if value part has values and gives names
if self~value_data[1] <> "" & self~value_data[1] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="post-office-box">'|| -
   escape(self~value_data[1]) ||'</span>' || "0A"X
```

```
if self~value_data[2] <> "" & self~value_data[2] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="extended-address">'|| -
   escape(self~value_data[2]) ||'</span>' || "0A"X
if self~value_data[3] <> "" & self~value_data[3] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="street-address">'|| -
   escape(self~value_data[3]) ||'</span>' || "0A"X
if self~value_data[4] <> "" & self~value_data[4] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="locality">'|| -
   escape(self~value_data[4]) ||'</span>' || "0A"X
if self~value_data[5] <> "" & self~value_data[5] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="region">'|| -
   escape(self~value_data[5]) ||'</span>' || "0A"X
if self~value_data[6] <> "" & self~value_data[6] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="postal-code">'|| -
   escape(self~value_data[6]) ||'</span>' || "0A"X
if self~value_data[7] <> "" & self~value_data[7] <> .nil then
tmpstr = tmpstr || tab2 ||'<span class="country-name">'|| -
   escape(self~value_data[7]) ||'</span>' || "0A"X



tmpstr = tmpstr || tab ||'</div>'

return tmpstr



/*
======================================================================
CLASS DEFINITION OF PHOTO
FURTHER DETAILS RFC 2426, P. 10, 31
*/

::class vcard30_photo public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "ENCODING", "TYPE")
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no group names allowed in hCard

o~classname = classname~changestr("_","-")

values = self~correct_data(node, .true)

--check if data is inline encoded data or a URI
if values~left(10) = 'data:image' then
   do
   parse var values header '/' type ';' encoding ',' value
   o~type~append(type~upper)
   o~encoding~append('B')
   end
else
   do
   value = values
   o~value~append('URI')
   end



o~value_data~append(value)

return o

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

if self~value~hasitem(uri) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<img class="' || -
         escape(self~classname~lower) || -
         '" src="' || escape(prop_val) || '" alt="" />'
```

```
   end
end

else if self~encoding~hasitem(b) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<img class="' || -
         escape(self~classname~lower) || -
         '" src="data:image/'

      do prop_attr over self~send(type)
         tmpstr = tmpstr || escape(prop_attr~lower)
      end

      tmpstr = tmpstr || ";base64," || escape(prop_val) || '" alt="" />'
   end

end

return tmpstr

/*
===================================================================
CLASS DEFINITION BEGIN
FURTHER DETAILS RFC 2426, P. 5, 30
*/

::class vcard30_begin public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
-------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class

o = self~new
o~group_name = ''
o~classname = "BEGIN"
o~value_data~append("VCARD")

return o


/*
-------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tmpstr = '<div class="vcard">'

return tmpstr


/*
===================================================================
CLASS DEFINITION END
FURTHER DETAILS RFC 2426, P. 5, 30
*/


::class vcard30_end public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
-------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class

o = self~new
o~group_name = ''
o~classname = "END"
o~value_data~append("VCARD")

return o
```

```
/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml

o = self~new
o~group_name = ''
o~classname = "END"
o~value_data~append("VCARD")

return o

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tmpstr = '</div>'

return tmpstr



/*
======================================================================
CLASS DEFINITION NAME
FURTHER DETAILS RFC 2426, P. 5, 30
*/


::class vcard30_name public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
/*
tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
======================================================================
CLASS DEFINITION PROFILE
FURTHER DETAILS RFC 2426, P. 5, 30
*/


::class vcard30_profile public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
/*
tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
======================================================================
CLASS DEFINITION SOURCE
FURTHER DETAILS RFC 2426, P. 5, 30
*/
```

```
::class vcard30_source public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "CONTEXT")
   self~values = .array~of("value_data")
   forward class (super)

/*
-----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
/*
tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
=======================================================================
CLASS DEFINITION FN "FORMATTED NAME"
FURTHER DETAILS RFC 2426, P. 8, 30
*/


::class vcard30_fn public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)


/*
=======================================================================
CLASS DEFINITION N ("NAME")
FURTHER DETAILS RFC 2426, P. 9, 30
*/


::class vcard30_n public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)
/*
-----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no grouping in hCard
o~classname = classname~changestr("_","-")

do i=1 to 5 by 1 --assigns an initial value to the value_data array
   if o~value_data[i] = .nil then  o~value_data[i] = ""
end

if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      self~followNode(children~item(i), o)
   end
end

do i = 1 to 5
o~value_data[i] = o~value_data[i]~substr(2)
end

return o
-------------------------------------
::method followNode class/* walks the document tree recursively */
  use arg node o
```

```
   --say node~getNodeType
   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then
            do
               do i=0 to atts~getLength-1
                  att=atts~item(i)
                  if att~getName="class" then
                     select --attributes values according to their position
                     when att~getValue~wordpos('family-name') <> 0 -
                     then o~value_data[1] = o~value_data[1] || ','|| self~correct_data(node, .true)
                     when att~getValue~wordpos('given-name') <> 0 -
                     then o~value_data[2] = o~value_data[2] || ',' || self~correct_data(node, .true)
                     when att~getValue~wordpos('additional-name') <> 0 -
                     then o~value_data[3] = o~value_data[3] || ',' || self~correct_data(node, .true)
                     when att~getValue~wordpos('honorific-prefix') <> 0 -
                     then o~value_data[4] = o~value_data[4] || ',' || self~correct_data(node, .true)
                     when att~getValue~wordpos('honorific-suffix') <> 0 -
                     then o~value_data[5] = o~value_data[5] || ',' || self~correct_data(node, .true)
                     otherwise say 'attribute unknown to vcard-n used!'
                     end
               end

            end
      end

if node~hasChildNodes then --checks if child nodes are availabe
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~followNode(children~item(i), o)
      end
   end

return o


/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || '">'  -
   || "0A"X

do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || tab2 || '<span class="' || -
         escape(attribute~lower) || '">' -
         || escape(prop_attr~lower) || '</span>'|| "0A"X
      end
   end

end

--checks if value part has values and gives names
if self~value_data[1] <> "" & self~value_data[1] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="family-name">'|| -
   escape(self~value_data[1]) ||'</span>' || "0A"X
if self~value_data[2] <> "" & self~value_data[2] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="given-name">'|| -
   escape(self~value_data[2]) ||'</span>' || "0A"X
if self~value_data[3] <> "" & self~value_data[3] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="additional-name">'|| -
   escape(self~value_data[3]) ||'</span>' || "0A"X
if self~value_data[4] <> "" & self~value_data[4] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="honorific-prefix">'|| -
   escape(self~value_data[4]) ||'</span>' || "0A"X
if self~value_data[5] <> "" & self~value_data[5] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="honorific-suffix">'|| -
   escape(self~value_data[5]) ||'</span>' || "0A"X

tmpstr = tmpstr || tab ||'</span>'

return tmpstr


/*
======================================================================
CLASS DEFINITION NICKNAME
FURTHER DETAILS RFC 2426, P. 9, 30
*/
```

```
::class vcard30_nickname public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)



/*
=====================================================================
CLASS DEFINITION BDAY "BIRTHDAY"
FURTHER DETAILS RFC 2426, P. 11, 31
*/


::class vcard30_bday public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE")
   self~values = .array~of("value_data")
   forward class (super)


/*
=====================================================================
CLASS DEFINITION LABEL (E. G. FOR PRINTING)
FURTHER DETAILS RFC 2426, P. 13, 31
*/


::class vcard30_label public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE", "TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
---------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no groups in hcard
o~classname= classname~changestr("_","-")

if node~hasChildNodes then
do
   children=node~getChildNodes --get NodeList
   loop i=0 to children~length-1 --0-based indexes! */
      self~followNode(children~item(i), o)
   end
end

return o

/*
---------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o

   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then --checks if node has attributes
            do
               do i=0 to atts~getLength-1 --goes over all attributes
                  att=atts~item(i)  --defines currently processed attribute
                  if att~getName="class" then
                     select -- distributes the values
                     when att~getValue~upper~wordpos('TYPE') <> 0 -
then o~type~append(self~correct_data(node, .false)~upper)
                     otherwise say 'attribute unknown to property'
                     end
```

```
               end

            end
         else o~value_data~append(self~correct_data(node, .true))
      end


if node~hasChildNodes then
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~followNode(children~item(i), o)
      end
   end

return o



/*
---------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || '">'  || "0A"X

do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || tab2 || '<span class="' || escape(attribute~lower) || '">' -
         || escape(prop_attr~lower) || '</span>'|| "0A"X
      end
   end

end

--adds property values
do prop_val over self~send(value_data)
   tmpstr = tmpstr || tab || '<pre>' || escape(endofline(prop_val)) || '</pre>'
end



tmpstr = tmpstr || "0A"X ||'</span>'


return tmpstr

/*
=====================================================================
CLASS DEFINITION TEL "TELEPHONE"
FURTHER DETAILS RFC 2426, P. 14, 31
*/


::class vcard30_tel public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
---------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no grouping in hCard defined
o~classname = classname~changestr("_","-")

/*correct_data = ""
if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      if children~item(i)~getNodeType=3 then --checks for text nodes
         correct_data = correct_data || extended_strip(children~item(i)~getTextContent)
   end
end
*/
o~value_data~append(self~correct_data(node, .true))
```

```
--checks if child nodes with values are available
if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      self~followNode(children~item(i), o)
   end
end

return o

/*
----------------------------------------------------------------------
method goes recursively over document tree
*/
::method followNode class
use arg node o

if node~getNodeType=1 then --checks element nodes, "1" indicating
   --element nodes,
   --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
   do
      atts = node~getAttributes
      if atts~getLength>0 then
         do
            do i=0 to atts~getLength-1 --indexes base on 0
               att=atts~item(i)
               if att~getName="class" then
                  select --distributes correct data
                  when att~getValue~upper~wordpos('TYPE') <> 0 -
                  then o~type~append(self~correct_data(node, .false)~upper)
                  otherwise say 'unknown property attribute found'
                  end
            end
         end
   end


if node~hasChildNodes then
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~followNode(children~item(i), o)
      end
   end

return o



/*
======================================================================
CLASS DEFINITION EMAIL
FURTHER DETAILS RFC 2426, P. 15, 32
*/


::class vcard30_email public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = ''
o~classname = classname~changestr("_","-")

--removal of 'mailto:'from read in data
o~value_data~append(self~correct_data(node, .true)~changestr('mailto:', ''))
o~type~append('INTERNET')

if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1 --index base on 0
      self~followNode(children~item(i), o)
   end
end

return o

/*
```

```
-------------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o

   --say node~getNodeType
   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then --controls if attributes available
            do
               do i=0 to atts~getLength-1 --goes over all attributes, index bases on 0
                  att=atts~item(i)
                  if att~getName="class" then
                     select
                     when att~getValue~upper~wordpos('TYPE') <> 0 -
                       then o~type~append(self~correct_data(node, .false)~upper)
                     otherwise say 'unknown property attribute found'
                     end
               end

            end
      end


  if node~hasChildNodes then
  do
     children=node~getChildNodes
     loop i=0 to children~length-1
        self~followNode(children~item(i), o)
     end
   end

return o

/*
-------------------------------------------------------------------------
method processing xml-data in a multi-value case
e-mail class takes ordinary value (href)
*/
::method specialfromxml class
use arg node classname

o = self~fromxml(node, classname)

return o


/*
-------------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<a class="' || escape(self~classname~lower) || '" href="mailto:' || -
   escape(self~value_data[1]) || '">'


do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      if prop_attr~upper <> 'INTERNET' then
      do
      tmpstr = tmpstr || '<span class="' || -
      escape(attribute~lower) || '">' || escape(prop_attr~lower) || '</span>'
      end
   end

end


tmpstr = tmpstr || escape(self~value_data[1]) || '</a>'

return tmpstr



/*
=========================================================================
CLASS DEFINITION MAILER
FURTHER DETAILS RFC 2426, P. 15, 32
*/


::class vcard30_mailer public subclass vcard30

/*
```

```
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)



/*
======================================================================
CLASS DEFINITION TZ "TIME ZONE"
FURTHER DETAILS RFC 2426, P. 16, 32
*/


::class vcard30_tz public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
======================================================================
CLASS DEFINITION GEO "GEOGRAPHIC POSITION
FURTHER DETAILS RFC 2426, P. 16, 32
*/


::class vcard30_geo public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no grouping defined for hCard
o~classname = classname~changestr("_","-")

if node~getNodeName = 'abbr' then
do
   data = self~correct_data(node, .true)
   parse var data latitude ';' longitude
   o~value_data~append(latitude)
   o~value_data~append(longitude)
end

data = self~correct_data(node, .true)
parse var data latitude ';' longitude
o~value_data[1]=latitude
o~value_data[2]=longitude


if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      self~followNode(children~item(i), o)
   end
end


do i=1 to 2 by 1
   if o~value_data[i] = .nil then  o~value_data[i] = ""
end

return o


/*
----------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o
```

```
   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then
            do
               do i=0 to atts~getLength-1 --0 based index
                  att=atts~item(i)
                  if att~getName="class" then
                     select --distributes correct values

when att~getValue~wordpos('latitude') <> 0 then o~value_data[1] = self~correct_data(node, .true)
                     when att~getValue~wordpos('longitude') <> 0 -
                       then o~value_data[2] = self~correct_data(node, .true)
                     otherwise say  'unknown sub-type'
                     end
               end

            end
      end


  if node~hasChildNodes then
  do
     children=node~getChildNodes
     loop i=0 to children~length-1
        self~followNode(children~item(i), o)
     end
   end

return o


/*
---------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || -
      '">'  || "0A"X

do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || tab2 || '<span class="' || -
         escape(attribute~lower) || '">' || -
         escape(prop_attr~lower) -
         || '</span>'|| "0A"X
      end
   end

end

--checks if value part has values and gives names
if self~value_data[1] <> "" & self~value_data[1] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="latitude">'|| -
   escape(self~value_data[1]) ||'</span>' || "0A"X
if self~value_data[2] <> "" & self~value_data[2] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="longitude">'|| -
   escape(self~value_data[2]) ||'</span>' || "0A"X



tmpstr = tmpstr || tab ||'</span>'

return tmpstr

/*
=====================================================================
CLASS DEFINITION TITLE "PERSON'S TITLE"
FURTHER DETAILS RFC 2426, P. 17, 32
*/


::class vcard30_title public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)

/*
=====================================================================
```

```
CLASS DEFINITION ROLE "PERSON'S ROLE"
FURTHER DETAILS RFC 2426, P. 18, 32
*/


::class vcard30_role public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)



/*
==========================================================================
CLASS DEFINITION LOGO
FURTHER DETAILS RFC 2426, P. 18, 32
*/

::class vcard30_logo public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "ENCODING", "TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
--------------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no grouping in hCard, hence empty string

o~classname = classname~changestr("_","-")

values = self~correct_data(node, .true)

--checks if values are inline encoded or URIs
if values~left(10) = 'data:image' then
   do
   parse var values header '/' type ';' encoding ',' value
   o~type~append(type~upper)
   o~encoding~append('B')
   end
else
   do
   value = values
   o~value~append('URI')
   end



o~value_data~append(value)

return o

/*
--------------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

if self~value~hasitem(uri) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<img class="' || -
         escape(self~classname~lower) || -
         '" src="' || escape(prop_val) || '" alt="logo" />'
   end
end

else if self~encoding~hasitem(b) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<img class="' || -
         escape(self~classname~lower) || -
         '" src="data:image/'
```

```
      do prop_attr over self~send(type)
         tmpstr = tmpstr || escape(prop_attr~lower)
      end

      tmpstr = tmpstr || ";base64," || escape(prop_val) || '" alt="logo" />'
   end

end

return tmpstr


/*
======================================================================
CLASS DEFINITION AGENT
FURTHER DETAILS RFC 2426, P. 19, 33
*/

::class vcard30_agent public subclass vcard30

::attribute ag_content class
::attribute has_vcard_end class

/*
attributes and values variables of class
*/
::method init class

   self~ag_content=.queue~new
   self~has_vcard_end = .false
   self~attributes = .array~of("VALUE")
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no group names for hcard defined

o~classname = classname~changestr("_","-")

o~value_data~append(self~correct_data(node, .true))
o~value~append('URI')

return o

/*
----------------------------------------------------------------------
method processing xml-data when the agent property consists
of another inline encoded vCard
*/
::method agentfromxml class
use arg node classname

o=self~new
o~group_name = ''
o~classname = 'AGENT'

ob=.VCARD30_BEGIN~fromxml --adds begin vCard to agent's vCard
self~ag_content~append(ob)
self~has_vcard_end = .true
drop ob

if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      self~followNode(children~item(i))

   end
end

self~vcard_control
 --checks agent's vCard for hCard pecularities -> n-optimization, version etc.

text = ''

i = self~ag_content~items --counts number of values
j=1

do item over self~ag_content
   if j < i then
   do
      text=text || item || "\n"
      j = j + 1
   end
   else
   do
```

```
      text=text || item
      j = j + 1
   end
end
self~ag_content = .queue~new
--text = text || '\nEND:VCARD'

o~value_data~append(backslash_escape(text))
--say '['backslash_escape(text)']'
return o

/*
-----------------------------------------------------------------------
method controls if vcard have at least the required properties
*/
::method vcard_control class
--/*
n = 0 --variable that indicates if a 'n' property is used
begin = 0

--checks if n property is found
do runner over self~ag_content
   if runner~class = .vcard30_n then n = n + 1
   if runner~class = .vcard30_begin then  begin = begin + 1
   else nop
end
--/*
--say 'n' n
--say 'begin' begin

angle_begin = 0
--trace ?i
if n >= begin then nop
else
   do item over self~ag_content
   if item~class = .vcard30_begin then angle_begin = angle_begin + 1
   --say 'angle_begin' angle_begin

   if angle_begin = begin then
   do

      if item~class = .vcard30_fn then
      do
         string = ''
         do value over item~value_data
            string = string value
         end


         if string~words = 2 then
            do

               if string~word(1)~verify(',',M) > 0 then
                  do
                     parse var string family_name given_name ';' rest
                  end
               else
                  do
                     parse var string given_name family_name ';' rest
                  end
               o=.vcard30_N~new
               o~value_data~append(family_name)
               o~value_data~append(given_name)
               o~value_data~append('')
               o~value_data~append('')
               o~value_data~append('')
               o~classname = "N"
               o~group_name = ''
               self~ag_content~append(o)
               drop o

            end
         else
            do
               o=.vcard30_N~new
               o~value_data~append('not indicated')
               o~classname = "N"
               o~group_name = ''
               self~ag_content~append(o)
               drop o
            end


      end
      else nop
   end
   else nop
   end

--*/

if self~has_vcard_end then
   do
```

```
      o=.VCARD30_VERSION~new
      o~value_data~append('3.0')
      o~classname = "VERSION"
      o~group_name = ''
      self~ag_content~append(o)
      drop o
      o=.VCARD30_END~fromxml
      self~ag_content~append(o)
      self~has_vcard_end = .false
   end
else nop
--*/
return


/*
-----------------------------------------------------------------------
method walks the document tree recursively honoring the requirements
for vcards
*/
::method followNode class
  use arg node

deeper = .true --value stores if child elements should be
   --processed as well

if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
--cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
     do
      atts = node~getAttributes

      if atts~getLength>0 then    /* o.k. attributes defined for this element  */
         do
            do i=0 to atts~getLength-1   /* iterate over all attributes, 0-based */
               att=atts~item(i)     /* get attribute node                       */
               if att~getName="class" then --looks for the attribute type that is used
                           --to indicate hcalendar/hcard values
                  do
                     if att~getValue~verify(" ",M) = 0 then --checks if multiple values for html
                                 --attribute "class" are used or not
                        do
                           if att~getValue = "vcard" then
                              do
                                 if self~has_vcard_end = .false then
                                    do
                                       o=.VCARD30_BEGIN~fromxml
                                       self~ag_content~append(o)
                                       self~has_vcard_end = .true
                                    end
                                 else
                                    do
                                       self~vcard_control
                                       o=.VCARD30_BEGIN~fromxml
                                       self~ag_content~append(o)
                                       self~has_vcard_end = .true
                                    end
                              end

                           if att~getValue = "category" then
                              do
                                 o=.VCARD30_CATEGORIES~fromxml(node, 'CATEGORIES')
                                 self~ag_content~append(o)
                              end

                           else
                              do
                                 classname = att~getValue~changestr("-","_")~upper
                                 cls = value("." || 'VCARD30' || "_" ||classname)
                                 --say cls
                                 vtype = value("." || 'VCARD30')
                                 class_check = vtype~send(subclasses) --creates array containing
                                          --all subclasses

                                 if class_check~hasitem(cls) then --checks if subclass is available
                                    do
                                       o=cls~fromxml(node, classname)
                                       self~ag_content~append(o)
                                    end
                                 else nop
                              end
                        end
                     else
                        do
                           --say "multiple classes found!"
                           tmp_var = att~getValue~changestr("-","_")

                           select
                           when tmp_var~wordpos('agent') <> 0 -
                           & tmp_var~wordpos('vcard') <> 0 then --looks for a agent property
                                       --containing an embeded vCard
                              do
                                 deeper = .false --child nodes do not need to processed any further
```

```
                cls = value("." || 'VCARD30' || "_" || 'AGENT')
                --say cls
                o=cls~agentfromxml(node)
                self~ag_content~append(o)
            end

        when tmp_var~wordpos('org') <> 0 -
        & tmp_var~wordpos('vcard') <> 0 then --looks for an org element that
                    --has an embedded vCard; not supported by
                    --the vCard standard
            do
                deeper = .false --child nodes do not need to processed any further
            end

        when tmp_var~wordpos('email') <> 0 -
        | tmp_var~wordpos('url') <> 0 then
            do
                classnames = att~getValue~changestr("-","_")~upper~makeArray(' ')

                do classname over classnames
                --say classname
                    cls = value("." || 'VCARD30' || "_" || classname)
                    --say cls

                    if classname = 'VCARD' then
                    do

                        if self~has_vcard_end = .false then
                        do
                            o=.VCARD30_BEGIN~fromxml
                            self~ag_content~append(o)
                            self~has_vcard_end = .true
                        end
                        else
                        do
                            self~vcard_control
                            o=.VCARD30_BEGIN~fromxml
                            self~ag_content~append(o)
                            self~has_vcard_end = .true
                        end
                    end
                    else nop

                    if classname = 'VEVENT' then deeper = .false --child nodes do
                                            --not need to processed any further

                    vtype = value("." || 'VCARD30')
                    class_check = vtype~send(subclasses) --creates array containing
                    --all subclasses

                    if class_check~hasitem(cls) then --checks if subclass is
                                                     -- available
                        do

                            if classname = 'VEVENT' then deeper = .false
                            --child nodes do not need to processed any further
                                  if classname = 'AGENT' then
                            do
                                o=cls~fromxml(node, classname)
                                self~ag_content~append(o)
                            end

                            if classname = "CATEGORY" then
                            do
                                o=.VCARD30_CATEGORIES~fromxml(node, 'CATEGORIES')
                                self~ag_content~append(o)
                            end

                            else
                            do
                                o=cls~specialfromxml(node, classname)
                                self~ag_content~append(o)
                            end
                        end
                    else nop
                end
            end
        otherwise
            do
                classnames = att~getValue~changestr("-","_")~upper~makeArray(' ')

                do classname over classnames
                --say classname
                    cls = value("." || 'VCARD30' || "_" || classname)
                    --say cls

                    if classname = 'VCARD' then
                    do

                        if self~has_vcard_end = .false then
                        do
                            o=.VCARD30_BEGIN~fromxml
                            self~ag_content~append(o)
```

```
                                        self~has_vcard_end = .true
                                     end
                                     else
                                     do
                                        self~vcard_control
                                        o=.VCARD30_BEGIN~fromxml
                                        self~ag_content~append(o)
                                        self~has_vcard_end = .true
                                     end
                                  end
                                  else nop

                                  if classname = 'VEVENT' then deeper = .false --child nodes do
            --not need to processed any further

                                  vtype = value("." || 'VCARD30')
                                  class_check = vtype~send(subclasses) --creates array containing
                                  --all subclasses

                                  if class_check~hasitem(cls) then --checks if subclass is
                                                               --available
                                     do

                                        if classname = 'VEVENT' then deeper = .false
                                           --child nodes do not need to processed any further
                                                  if classname = 'AGENT' then
                                           do
                                              o=cls~fromxml(node, classname)
                                              self~ag_content~append(o)
                                           end

                                        if classname = "CATEGORY" then
                                           do
                                              o=.VCARD30_CATEGORIES~fromxml(node, 'CATEGORIES')
                                              self~ag_content~append(o)
                                           end

                                        else
                                           do
                                              o=cls~fromxml(node, classname)
                                              self~ag_content~append(o)
                                           end
                                     end
                                  else nop
                               end
                            end
                         end
                      end
                   end
                else nop
             end
          end
       else nop
    end
else nop

if deeper then
   do
      if node~hasChildNodes then
         do
            children=node~getChildNodes /* get NodeList */

            loop i=0 to children~length-1  /* 0-based indexes! */
            self~followNode(children~item(i))  /* recurse */
            end
         end
   end

return

/*
-------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

if self~value~hasitem(uri) then
do
tmpstr = tab || '<a class="' || escape(self~classname~lower) || -
      '" href="'

--adds property values, creation of output lines, insertion of ";"
--where needed

i = self~send(value_data)~items --counts number of values
j=1

do prop_val over self~send(value_data)
   if j < i then
   do
```

```
      tmpstr=tmpstr || escape(prop_val) || ";"
      j = j + 1
   end
   else
   do
      tmpstr=tmpstr || escape(prop_val)
      j = j + 1
   end
end

tmpstr = tmpstr ||'"/>'
end

else
do
string = ""

do item over self~value_data
   string = string || item
end

--parse var
--string = string~changestr("\;",";")~changestr("\,",",")-
--      ~changestr("\:",":")

filelines = string~makearray("\n")

i = 1
linenumber = filelines~items --number of data lines
e = 1
unfoldedlines = .array~new

do while i <= linenumber


   if filelines[i]~substr(1,1) <> " " | filelines[i]~substr(1,1) <> "09"x
               --checks if line starts with a gap
   then
      do
      unfoldedlines[e] = filelines[i] --assigns element
      e = e + 1
      end
   else
      unfoldedlines[e-1] = unfoldedlines[e-1] || filelines[i]~substr(2)
               --adds aditional element
   i = i+1
end


trans_unfoldedlines = .array~new

do item over unfoldedlines
   item=item~changestr('\:',':') ---{SIC}
   parse var item left ":" right
   right = right~changestr("\;",";")~changestr("\,",",")
   left = left~changestr("\;",";")~changestr("\,",",")
   txt = left|| ":" ||right
   trans_unfoldedlines~append(txt)
end

input_type = "vcard30"
ag_content= .array~new

do line over trans_unfoldedlines
   right_pos = line~verify(";:",M) --defines position of separator
   if right_pos < 2 then iterate --exception for faulty data
   grouping = line~verify(".",M) --checks if properties are grouped
   if grouping > right_pos then
      do
      classname = left(line~changestr("-","_"),right_pos-1)
      cls = value("." || input_type || "_" ||classname)
      group_name = ""
      end
   else
      do
      classname = substr(line~changestr("-","_"),-
            grouping+1, right_pos - grouping - 1)
      cls = value("." || input_type || "_" ||classname)
      group_name = left(line,grouping)
      end

   --defines class for further transformation

   if cls=.nil then iterate

   vtype = value("." || input_type)
   class_check = vtype~send(subclasses) --creates array containing
                  --all subclasses
   input_type_length = input_type~length --counts characters
   rootclass = value("." || input_type || "_X_PROPERTY")

   if class_check~hasitem(cls) then --checks if subclass is available
      do
```

```
      o = cls~fromstring(line, classname, group_name)
      ag_content~append(o)
      end
   else
      do
      if left(cls,input_type_length+4) = -
         "." || input_type || "_X_" then
         do
            tmpcls = rootclass~subclass(cls)
            o = tmpcls~fromstring(line, classname, group_name)
            ag_content~append(o)
         end
      else say cls "not defined!"
      end
end

tmpstr = tab || '<div class="vcard agent">'||"0A"X


i = ag_content~items --counts number of values
j=1
--trace ?i
do line over ag_content
   if line~class <> .vcard30_begin then
   do

      if j < i then
      do
         tmpstr = tmpstr|| tab ||line~send(makehtml)||"0A"X
         j = j + 1
      end
      else
      do
         tmpstr = tmpstr|| tab ||line~send(makehtml)
         j = j + 1
      end
   end
   else
      do
         j = j + 1
      end
end

end

return tmpstr

/*
======================================================================
CLASS DEFINITION ORG "ORGANISATION"
FURTHER DETAILS RFC 2426, P. 20, 33
*/

::class vcard30_org public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE")
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

dir = .directory~new

atts = node~getAttributes

do i=0 to atts~getLength-1
   att=atts~item(i)
   dir[att~getName~upper] = att~getValue
end

o = self~new
o~group_name = ''
o~classname = classname~changestr("_","-")

/*
do i=1 to 2 by 1
   --say o~value_data[i]
   if o~value_data[i] = .nil then  o~value_data[i] = ""
   say 'new:'  pp(o~value_data[i])
end
*/

--if o~value_data[2] = .nil then  o~value_data[2] = ""
```

```
o~value_data[1] = self~correct_data(node, .true)

if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      self~followNode(children~item(i), o)
   end
end
--o~value_data[2] = o~value_data[2]~substr(2)
return o

/*
----------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o

   --say node~getNodeType
   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then
            do
               do i=0 to atts~getLength-1
                  att=atts~item(i)
                  if att~getName="class" then
                     select
                     when att~getValue = 'organization-name' -
                       then o~value_data[1] = self~correct_data(node, .true)
                     when att~getValue = 'organization-unit' -
                       then o~value_data~append(self~correct_data(node, .true))
                     --when att~getValue = 'organization-unit' -
                    --then o~value_data[2] = o~value_data[2] ||';'||self~correct_data(node, .true)
                     otherwise say 'attribute unknown to property'
                     end
               end

            end
      end

  if node~hasChildNodes then
  do
     children=node~getChildNodes
     loop i=0 to children~length-1
        self~followNode(children~item(i), o)
     end
   end

return o



/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || -
      '">'  || "0A"X

do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || tab2 || '<span class="' || -
         escape(attribute~lower) || '">' || escape(prop_attr~lower) -
         || '</span>'|| "0A"X
      end
   end

end

if self~value_data[1] <> "" & self~value_data[1] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="organization-name">'|| -
   escape(self~value_data[1]) ||'</span>' || "0A"X

do i = 2 to self~value_data~items
if self~value_data[i] <> "" & self~value_data[i] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="organization-unit">'|| -
   escape(self~value_data[i]) ||'</span>' || "0A"X
end

tmpstr = tmpstr || tab ||'</span>'

return tmpstr
```

```
/*
======================================================================
CLASS DEFINITION CATEGORIES
FURTHER DETAILS RFC 2426, P. 20, 33
*/

::class vcard30_categories public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab


tmpstr = tab || '<span class="category">'


--checks if attribute is available and accesses its values
do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || '<span class="' || -
         escape(attribute~lower) || '">' || -
         escape(prop_attr~lower) -
         || '</span>'
      end
   end

end

--adds property values
do prop_val over self~send(value_data)
   tmpstr = tmpstr || escape(prop_val)
end



tmpstr = tmpstr ||'</span>'

return tmpstr



/*
======================================================================
CLASS DEFINITION NOTE
FURTHER DETAILS RFC 2426, P. 21, 33
*/

::class vcard30_note public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "LANGUAGE")
   self~values = .array~of("value_data")
   forward class (super)


/*
======================================================================
CLASS DEFINITION PRODID "PRODUCT IDENTIFICATION NUMBER"
FURTHER DETAILS RFC 2426, P. 21, 34
*/


::class vcard30_prodid public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
```

```
   forward class (super)

/*
-------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
/*
tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
=====================================================================
CLASS DEFINITION REV "LAST REVIEW"
FURTHER DETAILS RFC 2426, P. 22, 34
*/


::class vcard30_rev public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE")
   self~values = .array~of("value_data")
   forward class (super)


/*
=====================================================================
CLASS DEFINITION SORT-STRING "SORTING INDICATOR"
FURTHER DETAILS RFC 2426, P. 22, 34
*/


::class vcard30_sort_string public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE")
   self~values = .array~of("value_data")
   forward class (super)

/*
=====================================================================
CLASS DEFINITION SOUND
FURTHER DETAILS RFC 2426, P. 23, 34
*/

::class vcard30_sound public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "ENCODING", "TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
-------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no grouping defined for hCards

o~classname = classname~changestr("_","-")

values = self~correct_data(node, .true)

--checks if data is inline encoded or a URI
if values~left(10) = 'data:audio' then
   do
   parse var values header '/' type ';' encoding ',' value
   o~type~append(type~upper)
   o~encoding~append('B')
   end
else
   do
   value = values
```

```
   o~value~append('URI')
   o~type~append('BASIC')
   end



o~value_data~append(value)

return o

/*
--------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

if self~value~hasitem(uri) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<object class="' || -
         escape(self~classname~lower) || -
         '" type="audio/'

      do prop_attr over self~send(type)
         tmpstr = tmpstr || escape(prop_attr~lower)
      end

      tmpstr = tmpstr || '" data="' || escape(prop_val) || '" />'
   end
end

else if self~encoding~hasitem(b) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<object class="' || -
         escape(self~classname~lower) || -
         '" data="data:audio/'

      do prop_attr over self~send(type)
         tmpstr = tmpstr || escape(prop_attr~lower)
      end

      tmpstr = tmpstr || ";base64," || escape(prop_val) || '" />'
   end

end

return tmpstr



/*
====================================================================
CLASS DEFINITION UID
FURTHER DETAILS RFC 2426, P. 24, 35
*/

::class vcard30_uid public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)


/*
====================================================================
CLASS DEFINITION URL
FURTHER DETAILS RFC 2426, P. 25, 35
*/

::class vcard30_url public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
--------------------------------------------------------------------
method processing xml-data in a multi-value case
```

```
url always takes the href attribute, hence normal method
*/
::method specialfromxml class
use arg node classname

o = self~fromxml(node, classname)

return o

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<a class="' || escape(self~classname~lower) || '" href="' -
   || escape(self~value_data[1]) || '">'

tmpstr = tmpstr || escape(self~value_data[1]) || '</a>'


return tmpstr




/*
======================================================================
CLASS DEFINITION "VERSION"
FURTHER DETAILS RFC 2426, P. 25, 35
*/

::class vcard30_version public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml
/*
tab = "09"x

tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
======================================================================
CLASS DEFINITION CLASS
FURTHER DETAILS RFC 2426, P. 26, 35
*/

::class vcard30_class public subclass vcard30

/*
attributes and values variables (e.g. "PUBLIC", "PRIVATE") of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)




/*
======================================================================
CLASS DEFINITION KEY "ENCRYPTION KEY"
FURTHER DETAILS RFC 2426, P. 26, 35
*/

::class vcard30_key public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("TYPE", "ENCODING")
```

```
   self~values = .array~of("value_data")
   forward class (super)

/*
---------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no grouping defined for hcard

o~classname = classname~changestr("_","-")

values = self~correct_data(node, .true)

--checks if values is inline encoded data or a URI
if values~left(16) = 'data:application' then
   do
   parse var values header '/' type ';' encoding ',' value
   o~encoding~append('B')
   end
else
   do
   value = values
   end



o~value_data~append(value)

return o

/*
---------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

if self~encoding~hasitem(b) then
do
   do prop_val over self~send(value_data)

      tmpstr = tab || '<object class="' || -
         escape(self~classname~lower) || -
         '" type="application/octet-stream" data="data:application/octet-stream;base64,'

      tmpstr = tmpstr || escape(prop_val) || '" />'
   end

end
else forward class (super)
-- [sic] konstruktor oberklasse aufrufen!

return tmpstr



/*
=====================================================================
CLASS DEFINITION X-PROPERTY
FURTHER DETAILS RFC 2426, P. 29
*/

::class vcard30_x_property public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
---------------------------------------------------------------------
method processing left side of read-in line
*/
::method leftside class
use arg o left
   tmp1 = left~changestr("\;","01"X)
   loop while tmp1 <> "" --assigns boolean values to attributes
   parse var tmp1 leftside ";" tmp1
      do
      parse var leftside denominator "=" denominations
            o~non_standard_attributes[denominator] = denominations
      end
   end
```

```
return o

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml

tab = "09"x
/*
tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
======================================================================
CLASS DEFINITION IMPP "instant messaging, presence protocol".
FURTHER DETAILS RFC 4770, P. 1 ff.
*/

::class vcard30_impp public subclass vcard30

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("TYPE")
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method creates hcard
*/
::method makehtml
/*
tab = "09"x

tmpstr = tab || '<span class="note">'  -
      || self~class "is not supported"-
      " by the hcard format.</span>"
*/
return ''

/*
=========================ROUTINES=====================================
*/

/*
routine creates a getter method for attributes
*/
::routine makegetter
   parse arg name
   code="EXPOSE" name "; RETURN" name
   return code

----------------------------------------------------------------------

/*
routine creates a setter method for attributes
*/
::routine makesetter
   parse arg name
   code="EXPOSE" name "; USE ARG" name
   return code

----------------------------------------------------------------------

/*
routine replaces forbidden xml-signs; sequence is important!
otherwise already escaped signs would be escaped again; used for
hcard
*/
::routine escape
   parse arg value
   value=value~changestr("&","&amp;")
   value=value~changestr('"',"&quot;")
   value=value~changestr("'","&apos;")
   value=value~changestr("<","&lt;")
   value=value~changestr(">","&gt;")
   value=value~changestr("\;",";")
   value=value~changestr("\,",",")
   value=value~changestr("\:",":")
   return value
----------------------------------------------------------------------

/*
routine replaces a backslash escaped \n with an "0A"X
*/
::routine endofline
   parse arg value
```

```
   value=value~changestr("\n","0A"X)
   return value
-------------------------------------------------------------------

/*
routine removes leading and ending line feeds, carriage returns
and strips the given character set
*/
::routine extended_strip
parse arg string

string = string~strip

line_feed = string~countStr("0A"X)
carriage_return = string~countStr("0D"X)

repetions = line_feed + carriage_return

do repetions

   if string~left(1) = .endofline  -
      | string~left(1) = "0A"X -
      | string~left(1) = "0D"X -
      then
         string = string~substr(2)

   string = string~strip

   length = string~length
   if string~right(1) = .endofline -
      | string~right(1) = "0A"X -
      | string~right(1) = "0D"X -
      then
         string = string~substr(1, length-1)

   string = string~strip
end

string = string~changestr("0A"X, '\n')
string = string~changestr("0D"X, '\n')

return string
-------------------------------------------------------------------

/*
routine backslash-escapes values, used for agentfromxml-method
*/
::routine backslash_escape
   parse arg value
   value=value~changestr(";","\;")
   value=value~changestr(",","\,")
   value=value~changestr(":","\:")
   return value

::requires 'reader.rxj'
```

Code 34: vcard_classes.rxj.

A.3.1.3 icalendar_classes.rxj

```
#!/usr/bin/rexx
/*
   Name:    icalendar_classes.rxj
   Purpose: Collection of classes concerning iCalendar/hCalendar
   Author:  Johannes Paul Bielohaubek
   Date:    2010-05-15
   Version: 1.0.2a
   License:

    ------------------------ Apache Version 2.0 license -------------------------
       Copyright (C) 2010 Johannes Paul Bielohaubek

       Licensed under the Apache License, Version 2.0 (the "License");
       you may not use this file except in compliance with the License.
       You may obtain a copy of the License at

           http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
    -------------------------------------------------------------------------------
*/

/*
=========================CLASSES=====================================
=====================================================================
CLASS DEFINITION OF ICALENDAR (VCALENDAR VERSION 2.0)
```

```
*/

::class icalendar20 public
::attribute attributes class --attribute used in subclasses
::attribute values class --attribute used in subclasses
::attribute non_standard_attributes --non standard attributes, x-attributes
::attribute classname
::attribute group_name

/*
-------------------------------------------------------------------
init method that creates maker and setter methods for attributes
*/
::method init class

   if self~id="ICALENDAR20" then return

   do name over self~attributes
      self~define(name, makegetter(name))
      self~define(name||'=', makesetter(name))
   end

   do name over self~values
      self~define(name, makegetter(name))
      self~define(name||'=',makesetter(name))
   end

-------------------------------------------------------------------

::method init

   attributes = self~class~attributes --assigns newly created values
   values = self~class~values --assigns newly created values

   do name over attributes
      self~send(name||'=',.array~new)
      --sets initial attributes-value to an array
   end

   do name over values
      self~send(name||'=',.array~new) --sets initial values-value to an array
   end

   self~non_standard_attributes = .directory~new

   self~group_name = ''

   self~classname = self~class~id~substr(13)~changestr('_','-')
   --creates classname out of the id of a class

/*
-------------------------------------------------------------------
method reads in data and distributes it to according classes
*/
::method fromstring class
   use arg line

   o = self~new

   right_pos = line~verify(";:",M) --defines position of separator
   if right_pos < 2 then iterate --exception for faulty data
   grouping = line~verify(".",M) --checks if properties are grouped
   if grouping > right_pos then --checks if grouping available
      do
      classname = left(line,right_pos-1)
      group_name = ""
      end
   else
      do
      classname = substr(line,grouping+1, right_pos - grouping - 1)
      if grouping <> 0 then group_name = left(line,grouping-1)
      else group_name = left(line,grouping)
      end

   o~classname = classname
   o~group_name = group_name

   line = line~changestr("\:", "01"X) --replaces backslash escaped ":"

   --replaces non backslash escaped ":" within quotes
   tmp_store = .array~new
   loop while line <> ""
      parse var line part '"' line
      tmp_store~append(part)
   end

   n=2

   if tmp_store~items > 1 then
   do
      do while n <= tmp_store~items
         tmp_store[n] = '"' ||  tmp_store[n]-
               ~changestr(":","02"X) || '"'
```

```
         n = n + 2
      end
      n = 2
   end


   do i over tmp_store
      line = line || i
   end


   parse var line left ":" right --divides into a attributes and values part
   left = left~upper~changestr("01"X,"\:")~changestr("02"X,":") --back to orignal values
   right = right~changestr("01"X,"\:")~changestr("02"X,":")

   self~leftside(o, left)
   self~rightside(o, right)
return o


/*
---------------------------------------------------------------------
method processing left side of read-in line
*/
::method leftside class
use arg o left

   tmp1 = left~changestr("\;","01"X)
   loop while tmp1 <> "" --assigns boolean values to attributes
   parse var tmp1 leftside ";" tmp1
      do
      parse var leftside denominator "=" denominations
      if o~class~attributes~hasitem(denominator) then
         do
            loop while denominations <> ""
               parse var denominations tmp2 "," denominations
               o~send(denominator)~append(tmp2)
            end

         end
      if denominator~left(2) = "X-" then
         do
            o~non_standard_attributes[denominator] = denominations
            --say o~non_standard_attributes[denominator]
         end
      end
   end

return o


/*
---------------------------------------------------------------------
method processing right side of read-in line
*/
::method rightside class
use arg o right
   tmp2 = right~changestr("\;","01"X)
   loop while tmp2 <> "" --assigns boolean values to attributes

   parse var tmp2 rightside ";" tmp2
      o~value_data~append(rightside~changestr("01"X,"\;"))
   end
return o


/*
---------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o = self~new
o~group_name = '' --no group names in hCalendar format allowed

if classname <> 'CLASSNAME' then o~classname = classname~changestr("_","-")

o~value_data~append(self~correct_data(node, .true))

return o

/*
---------------------------------------------------------------------
method retrieves correct property value from the dom-node according
to the hcard-specifications on http://microformats.org/wiki/hcard
*/
::method correct_data class
use arg node us_value --us_value indicates if method should
   --look for unspecified values as defines at
   --http://microformats.org/wiki/hcalendar

dir = .directory~new --directory that stores attributes from nodes
```

```
atts = node~getAttributes

do i=0 to atts~getLength-1 --iterates over all attributes
   att=atts~item(i) --gets attribute node
   dir[att~getName~upper] = att~getValue
end

select --chooses the relevant data according to the tag type
when node~getNodeName = 'abbr' then correct_data = extended_strip(dir[TITLE])
when node~getNodeName = 'a' then correct_data = extended_strip(dir[HREF])
when node~getNodeName = 'img' then correct_data = extended_strip(dir[SRC])
when node~getNodeName= 'object' then correct_data = extended_strip(dir[DATA])
otherwise
   do
   if us_value = .true then correct_data = self~unspecified_value(node)
   else correct_data = extended_strip(node~getTextContent)
   end
end

--value class pattern rule http://microformats.org/wiki/value-class-pattern
--checks if child elements with the attribute 'CLASS="VALUE"' exists
Value_Class_Pattern = self~value_class_pattern(node)
if Value_Class_Pattern~length > 0 then correct_data = Value_Class_Pattern

return correct_data

/*
-------------------------------------------------------------------
method retrieves correct property value given that the value is not
specified
*/
::method unspecified_value class
use arg node

uv=''

if node~hasChildNodes then
   do
      children=node~getChildNodes

      loop i=0 to children~length-1 --goes over all child nodes
         new_uv = self~uv_follownode(children~item(i))
         uv = uv || new_uv
      end

   end
else uv = node~getTextContent

uv =  extended_strip(uv) --removes characters that are not needed

return uv

/*
-------------------------------------------------------------------
method walks the document tree recursively looking for text values
outside an element having 'type' as an xml-'class'-attribute
*/
::method uv_followNode class
  use arg node
uv = ''
if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
   do
      atts = node~getAttributes

      if atts~getLength>0 then
         do
            do i=0 to atts~getLength-1
               att=atts~item(i)
               if att~getName="class" then --looks for the attribute type that is used
                           --to indicate hcalendar/hcard values
                  do
                     if att~getValue~wordpos('type') = 0 then
                        do
                           uv = uv || node~getTextContent
                        end
                     else nop
                  end
               else nop
            end
         end
      else nop
   end
else uv = uv || node~getTextContent

if node~hasChildNodes then
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~uv_followNode(children~item(i), uv)
      end
   end
else nop
```

```
return uv

/*
------------------------------------------------------------------------
method retrieves correct property value given that child nodes
with the 'value' attribute exist
*/
::method value_class_pattern class
use arg node

vcp = ''
if node~hasChildNodes then
do
   children=node~getChildNodes

   loop i=0 to children~length-1 --goes over all child nodes
      value = self~vcp_follownode(children~item(i))
      vcp = vcp || value
   end

end
else nop
return vcp

/*
------------------------------------------------------------------------
method walks over direct child nodes having the 'class'-attribute
'value'
*/
::method vcp_follownode class
use arg node

value = ''
if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
--cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
   do
      atts = node~getAttributes

      if atts~getLength>0 then
         do i=0 to atts~getLength-1
            att=atts~item(i)
            if att~getName='class' & att~getvalue = 'value' then
          --looks for the attribute type that is used
                     --to indicate hcalendar/hcard values
               do
                  dir = .directory~new --directory that stores attributes from nodes
                  do i=0 to atts~getLength-1 --iterates over all attributes
                     att=atts~item(i) --gets attribute node
                     dir[att~getName~upper] = att~getValue
                  end

                  select --chooses the relevant data according to the tag type
                  when node~getNodeName = 'abbr' then value = extended_strip(dir[TITLE])
                  when node~getNodeName = 'a' then value = extended_strip(dir[HREF])
                  when node~getNodeName = 'img' then value = extended_strip(dir[SRC])
                  when node~getNodeName= 'object' then value = extended_strip(dir[DATA])
                  otherwise value = extended_strip(node~getTextContent)
                  end
               end
            else nop
         end
      else nop
   end
else nop

return value

/*
------------------------------------------------------------------------
method creates string
*/
::method makestring
expose attributes values

if self~group_name <> "" then
   do
   tmpstr = self~group_name || "." || self~classname
   end
else
   do
   tmpstr=self~classname
   end

--non standard attributes, ordered alphabetically
tmpArr = self~non_standard_attributes~allindexes-
      ~sortwith(.caselesscomparator~new)
do entry over tmpArr
   if self~non_standard_attributes[entry] <> "" then
   do
   tmpstr = tmpstr || ";" || entry || "=" || -
      self~non_standard_attributes[entry]
```

```
   end
end


--standard attributes
do attribute over self~class~attributes
if attribute <> 'group_name' then
do
   if self~send(attribute)~items<>0 then
   do
      do prop_attr over self~send(attribute)
         tmpstr=tmpstr ||  ";" || attribute -
         || "=" || prop_attr
      end
   end
end
end

tmpstr=tmpstr||":"

-----creation of output lines, insertion of ";" where needed--------

i = self~send(value_data)~items --counts number of values
j=1

do prop_val over self~send(value_data)
   if j < i then
   do
      tmpstr=tmpstr || prop_val || ";"
      j = j + 1
   end
   else
   do
      tmpstr=tmpstr || prop_val
      j = j + 1
   end
end


------------folding of lines----------------------------------------

pos = 75
do while pos < tmpstr~length
   tmpstr = tmpstr~insert("0A"X || " ", pos)
   pos = pos + 77
end

return tmpstr



/*
---------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || -
      '">'


--checks if attribute is available and accesses its values
do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || '<span class="' || -
         escape(attribute~lower) || '">' || -
         escape(prop_attr~lower) -
         || '</span>'
      end
   end

end

--adds property values, creation of output lines, insertion of ";"
--where needed

i = self~send(value_data)~items --counts number of values
j=1

do prop_val over self~send(value_data)
   if j < i then
   do
      tmpstr=tmpstr || escape(prop_val) || ";"
      j = j + 1
   end
   else
   do
      tmpstr=tmpstr || escape(prop_val)
```

```
      j = j + 1
   end
end

tmpstr = tmpstr ||'</span>'

return tmpstr

/*
====================================================================
CLASS DEFINITION OF CALSCALE
FURTHER DETAILS RFC 5545, P. 76 - 77
*/

::class icalendar20_calscale public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
--------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''

/*
====================================================================
CLASS DEFINITION OF METHOD
FURTHER DETAILS RFC 5545, P. 77 - 78
*/

::class icalendar20_method public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
--------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''

/*
====================================================================
CLASS DEFINITION OF PRODID
FURTHER DETAILS RFC 5545, P. 78 - 79
*/

::class icalendar20_prodid public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "ENCODING", "TYPE")
   self~values = .array~of("value_data")

   forward class (super)

/*
--------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
====================================================================
CLASS DEFINITION OF VERSION
FURTHER DETAILS RFC 5545, P. 79 - 80
```

```
*/

::class icalendar20_version public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
-----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
=======================================================================
CLASS DEFINITION OF ATTACH
FURTHER DETAILS RFC 5545, P. 80 - 81
*/

::class icalendar20_attach public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "ENCODING", "FMTTYPE")
   self~values = .array~of("value_data")

   forward class (super)


/*
=======================================================================
CLASS DEFINITION OF CATEGORIES
FURTHER DETAILS RFC 5545, P. 81 - 82
*/

::class icalendar20_categories public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
=======================================================================
CLASS DEFINITION OF CLASS
FURTHER DETAILS RFC 5545, P. 82 - 83
*/

::class icalendar20_class public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~NEW
   self~values = .array~of("value_data")

   forward class (super)

/*
=======================================================================
CLASS DEFINITION OF COMMENT
FURTHER DETAILS RFC 5545, P. 83 - 84
*/

::class icalendar20_comment public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("ALTREP", "LANGUAGE")
   self~values = .array~of("value_data")
```

```
   forward class (super)


/*
======================================================================
CLASS DEFINITION OF DESCRIPTION
FURTHER DETAILS RFC 5545, P. 84 - 85
*/

::class icalendar20_description public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("ALTREP", "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF GEO
FURTHER DETAILS RFC 5545, P. 85 - 87
*/

::class icalendar20_geo public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)


/*
----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no grouping defined for hCalendar
o~classname = classname~changestr("_","-")

if node~getNodeName = 'abbr' then
do
   data = self~correct_data(node, .true)
   parse var data latitude ';' longitude
   o~value_data~append(latitude)
   o~value_data~append(longitude)
end




if node~hasChildNodes then
do
   children=node~getChildNodes
   loop i=0 to children~length-1
      self~followNode(children~item(i), o)
   end
end


do i=1 to 2 by 1
   if o~value_data[i] = .nil then  o~value_data[i] = ""
end

return o


/*
----------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o

 if node~getNodeType=1 then --checks element nodes, "1" indicating
  --element nodes,
  --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
  do
   atts = node~getAttributes
   if atts~getLength>0 then
    do
     do i=0 to atts~getLength-1 --0 based index
```

```
      att=atts~item(i)
      if att~getName="class" then
       select --distributes correct values

when att~getValue~wordpos('latitude') <> 0 then o~value_data[1] = self~correct_data(node, .true)

when att~getValue~wordpos('longitude') <> 0 then o~value_data[2] = self~correct_data(node, .true)
       otherwise say  'unknown sub-type'
       end
      end
    end
  end


  if node~hasChildNodes then
  do
     children=node~getChildNodes
     loop i=0 to children~length-1
        self~followNode(children~item(i), o)
     end
   end

return o


/*
---------------------------------------------------------------------
method createshcalendar
*/
::method makehtml

tab = "09"x
tab2 = tab || tab

tmpstr = tab || '<span class="' || escape(self~classname~lower) || -
      '">'  || "0A"X

do attribute over self~class~attributes
   if self~send(attribute)~items<>0 then
   do prop_attr over self~send(attribute)
      do
      tmpstr = tmpstr || tab2 || '<span class="' || -
         escape(attribute~lower) || '">' || -
         escape(prop_attr~lower) -
         || '</span>'|| "0A"X
      end
   end

end

--checks if value part has values and gives names
if self~value_data[1] <> "" & self~value_data[1] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="latitude">'|| -
   escape(self~value_data[1]) ||'</span>' || "0A"X
if self~value_data[2] <> "" & self~value_data[2] <> .nil then
   tmpstr = tmpstr || tab2 ||'<span class="longitude">'|| -
   escape(self~value_data[2]) ||'</span>' || "0A"X



tmpstr = tmpstr || tab ||'</span>'

return tmpstr



/*
=====================================================================
CLASS DEFINITION OF LOCATION
FURTHER DETAILS RFC 5545, P. 87 - 88
*/

::class icalendar20_location public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("ALTREP", "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
=====================================================================
CLASS DEFINITION OF PERCENT-COMPLETE
FURTHER DETAILS RFC 5545, P. 88 - 89
*/

::class icalendar20_percent_complete public subclass icalendar20

/*
```

```
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF PRIORITY
FURTHER DETAILS RFC 5545, P. 89 - 90
*/

::class icalendar20_priority public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF RESOURCES
FURTHER DETAILS RFC 5545, P. 91
*/

::class icalendar20_resources public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("ALTREP", "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)


/*
======================================================================
CLASS DEFINITION OF STATUS
FURTHER DETAILS RFC 5545, P. 92 - 93
*/

::class icalendar20_status public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF SUMMARY
FURTHER DETAILS RFC 5545, P. 93 - 94
*/

::class icalendar20_summary public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("ALTREP", "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF COMPLETED
```

```
FURTHER DETAILS RFC 5545, P. 94 - 95
*/

::class icalendar20_completed public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF DTEND
FURTHER DETAILS RFC 5545, P. 95 - 96
*/

::class icalendar20_dtend public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "TZID")
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method retrieves correct property value given that child nodes
with the 'value' attribute exist
*/
::method value_class_pattern class
use arg node

vcp = ''
if node~hasChildNodes then
do
   children=node~getChildNodes

   loop i=0 to children~length-1 --goes over all child nodes
      if children~item(i)~getNodeType=1 then
         do
         value = self~vcp_follownode(children~item(i))
         vcp = vcp || 'T' || value
         end
      else nop
   end

   if vcp~length>0 then
   do
      vcp = vcp~substr(2)

      parse var vcp date 'T' time
      date = date~changestr('-','')
      time = time~changestr(':','')

      if time~length < 6 then
         do
            time = time || '00'
         end
      else nop

      vcp = date || 'T' || time
      end
   else nop
end
else nop

return vcp


/*
======================================================================
CLASS DEFINITION OF DUE
FURTHER DETAILS RFC 5545, P. 96 - 97
*/
```

```
::class icalendar20_due public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "TZID")
   self~values = .array~of("value_data")

   forward class (super)

/*
-----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
=======================================================================
CLASS DEFINITION OF DTSTART
FURTHER DETAILS RFC 5545, P. 97 - 98
*/

::class icalendar20_dtstart public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "TZID")
   self~values = .array~of("value_data")

   forward class (super)

/*
-----------------------------------------------------------------------
method retrieves correct property value given that child nodes
with the 'value' attribute exist
*/
::method value_class_pattern class
use arg node

vcp = ''
if node~hasChildNodes then
do
   children=node~getChildNodes

   loop i=0 to children~length-1 --goes over all child nodes
      if children~item(i)~getNodeType=1 then
         do
         value = self~vcp_follownode(children~item(i))
         vcp = vcp || 'T' || value
         end
      else nop
   end

   if vcp~length>0 then
   do
      vcp = vcp~substr(2)

      parse var vcp date 'T' time
      date = date~changestr('-','')
      time = time~changestr(':','')

      if time~length < 6 then
         do
            time = time || '00'
         end
      else nop

      vcp = date || 'T' || time
      end
   else nop
end
else nop

return vcp

/*
=======================================================================
CLASS DEFINITION OF DURATION
FURTHER DETAILS RFC 5545, P. 99
*/

::class icalendar20_duration public subclass icalendar20

/*
attributes and values variables of class
```

```
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF FREEBUSY
FURTHER DETAILS RFC 5545, P. 100 - 101
*/

::class icalendar20_freebusy public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("FBTYPE")
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF TRANSP
FURTHER DETAILS RFC 5545, P. 101 - 102
*/

::class icalendar20_transp public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)


/*
======================================================================
CLASS DEFINITION OF TZID
FURTHER DETAILS RFC 5545, P. 102 - 103
*/

::class icalendar20_tzid public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF TZNAME
FURTHER DETAILS RFC 5545, P. 103 - 104
*/

::class icalendar20_tzname public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class
```

```
   self~attributes = .array~of("LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
-------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
===================================================================
CLASS DEFINITION OF TZOFFSETFROM
FURTHER DETAILS RFC 5545, P. 104 -105
*/

::class icalendar20_tzoffsetfrom public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
-------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'

return ''

/*
===================================================================
CLASS DEFINITION OF TZOFFSETTO
FURTHER DETAILS RFC 5545, P. 105 - 106
*/

::class icalendar20_tzoffsetto public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
-------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
===================================================================
CLASS DEFINITION OF TZURL
FURTHER DETAILS RFC 5545, P. 106
*/

::class icalendar20_tzurl public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "ENCODING", "TYPE")
   self~values = .array~of("value_data")

   forward class (super)

/*
-------------------------------------------------------------------
method creates hcalendar
*/
```

```
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''

/*
property not allowed by now
/*
-----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml

tab = "09"x

tmpstr = tab || '<a class="' || escape(self~classname~lower) || '" href="' -
   || escape(self~value_data[1]) || '">'

tmpstr = tmpstr || escape(self~value_data[1]) || '</a>'


return tmpstr
*/

/*
=======================================================================
CLASS DEFINITION OF ATTENDEE
FURTHER DETAILS RFC 5545, P. 107 - 109
*/

::class icalendar20_attendee public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("CUTYPE", "MEMBER", "ROLE"-
            "PARTSTAT", "RSVP", "DELEGATED_TO",-
            "DELEGATED_FROM", "SENT_BY", "CN", "DIR",-
            "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
-----------------------------------------------------------------------
method processing xml-data
*/
::method fromxml class
use arg node classname

o=self~new
o~group_name = '' --no groups in hcard
o~classname= classname~changestr("_","-")

if node~hasChildNodes then
do
   children=node~getChildNodes --get NodeList
   loop i=0 to children~length-1 --0-based indexes! */
      self~followNode(children~item(i), o)
   end
end

o~value_data~append(self~correct_data(node, .true))

return o

/*
-----------------------------------------------------------------------
method walks the document tree recursively
*/
::method followNode class
  use arg node o

   if node~getNodeType=1 then --checks element nodes, "1" indicating
      --element nodes,
      --cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
      do
         atts = node~getAttributes
         if atts~getLength>0 then --checks if node has attributes
            do
               do i=0 to atts~getLength-1 --goes over all attributes
                  att=atts~item(i)  --defines currently processed attribute
                  if att~getName="class" then
                     select -- distributes the values
                     when att~getValue~upper~wordpos('PARTSTAT') <> 0 -
                      then o~PARTSTAT~append(self~correct_data(node, .TRUE)~upper)
                     when att~getValue~upper~wordpos('ROLE') <> 0 -
                               then o~ROLE~append(self~correct_data(node, .TRUE)~upper)
                     otherwise say 'attribute unknown to property'
                     end
               end
```

```
            end
      end


if node~hasChildNodes then
   do
      children=node~getChildNodes
      loop i=0 to children~length-1
         self~followNode(children~item(i), o)
      end
   end

return o


/*
======================================================================
CLASS DEFINITION OF CONTACT
FURTHER DETAILS RFC 5545, P. 109 - 111
*/

::class icalendar20_contact public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("ALTREP", "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF ORGANIZER
FURTHER DETAILS RFC 5545, P. 111 - 112
*/

::class icalendar20_organizer public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("CN", "DIR", "SENT_BY" -
            "LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF RECURRENCE-ID
FURTHER DETAILS RFC 5545, P. 112 - 114
*/

::class icalendar20_recurrence_id public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "TZID", "RANGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF RELATED-TO
FURTHER DETAILS RFC 5545, P. 115 - 116
*/

::class icalendar20_related_to public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("RELTYPE")
```

```
   self~values = .array~of("value_data")

   forward class (super)


/*
======================================================================
CLASS DEFINITION OF URL
FURTHER DETAILS RFC 5545, P. 116 - 117
*/

::class icalendar20_url public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml

tab = "09"x

tmpstr = tab || '<a class="' || escape(self~classname~lower) || '" href="' -
   || escape(self~value_data[1]) || '">'

tmpstr = tmpstr || escape(self~value_data[1]) || '</a>'


return tmpstr

/*
======================================================================
CLASS DEFINITION OF UID
FURTHER DETAILS RFC 5545, P. 117 - 118
*/

::class icalendar20_uid public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
/*
part programmed for case that property is allowed in hCalendar
tab = "09"x

tmpstr = tab || '<a class="' || escape(self~classname~lower) || '" href="' -
   || escape(self~value_data[1]) || '">'

tmpstr = tmpstr || escape(self~value_data[1]) || '</a>'
*/
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''

return tmpstr

/*
======================================================================
CLASS DEFINITION OF EXDATE
FURTHER DETAILS RFC 5545, P. 118 - 119
*/

::class icalendar20_exdate public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "TZID")
   self~values = .array~of("value_data")

   forward class (super)
```

```
/*
======================================================================
CLASS DEFINITION OF RDATE
FURTHER DETAILS RFC 5545, P. 120 - 121
*/

::class icalendar20_rdate public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "TZID")
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF RRULE
FURTHER DETAILS RFC 5545, P. 122 - 132
*/

::class icalendar20_rrule public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
======================================================================
CLASS DEFINITION OF ACTION
FURTHER DETAILS RFC 5545, P. 132 - 133
*/

::class icalendar20_action public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF REPEAT
FURTHER DETAILS RFC 5545, P. 133
*/

::class icalendar20_repeat public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
```

```
======================================================================
CLASS DEFINITION OF TRIGGER
FURTHER DETAILS RFC 5545, P. 133 - 135
*/

::class icalendar20_trigger public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("VALUE", "RELATED")
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF CREATED
FURTHER DETAILS RFC 5545, P. 136
*/

::class icalendar20_created public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)


/*
======================================================================
CLASS DEFINITION OF DTSTAMP
FURTHER DETAILS RFC 5545, P. 137 - 138
*/

::class icalendar20_dtstamp public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)


/*
----------------------------------------------------------------------
method retrieves correct property value given that child nodes
with the 'value' attribute exist
*/
::method value_class_pattern class
use arg node

vcp = ''
if node~hasChildNodes then
do
   children=node~getChildNodes

   loop i=0 to children~length-1 --goes over all child nodes
      if children~item(i)~getNodeType=1 then
         do
         value = self~vcp_follownode(children~item(i))
         vcp = vcp || 'T' || value
         end
      else nop
   end

   if vcp~length>0 then
   do
      vcp = vcp~substr(2)

      parse var vcp date 'T' time
      date = date~changestr('-','')
      time = time~changestr(':','')
```

```
      if time~length < 6 then
         do
            time = time || '00'
         end
      else nop

      vcp = date || 'T' || time
      end
   else nop
end
else nop

return vcp

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF LAST-MODIFIED
FURTHER DETAILS RFC 5545, P. 138
*/

::class icalendar20_last_modified public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)
/*
======================================================================
CLASS DEFINITION OF SEQUENCE
FURTHER DETAILS RFC 5545, P. 138 - 139
*/

::class icalendar20_sequence public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)


/*
======================================================================
CLASS DEFINITION OF REQUEST-STATUS
FURTHER DETAILS RFC 5545, P. 141 - 143
*/

::class icalendar20_request_status public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~of("LANGUAGE")
   self~values = .array~of("value_data")

   forward class (super)

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''


/*
======================================================================
CLASS DEFINITION OF EXRULE
FURTHER DETAILS RFC 2445, P. 114 - 115
*/
```

```
::class icalendar20_exrule public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)

/*
-----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''

/*
=======================================================================
CLASS DEFINITION OF BEGIN
FURTHER DETAILS RFC 5545, P. 50
*/

::class icalendar20_begin public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)
/*
-----------------------------------------------------------------------
method for xml reading
*/
::method fromxml class
use arg ical_value --e.g. VEVENT

   o= self~new
   o~classname = "BEGIN"
   o~group_name = ''
   o~value_data~append(ical_value)

return o

--/*
/*
-----------------------------------------------------------------------
method creates make hcalendar
*/
::method makehtml

if self~value_data~hasitem("VCALENDAR")
then
   do
   tmpstr = /*'<?xml version="1.0" encoding="UTF-8"?>' || "0A"X ||*/-
    '<div class="vcalendar">'
   end
else
   do class_data over self~value_data
   tmpstr = '<div class="' || class_data~lower || '">'
   end

return tmpstr
--*/

/*
=======================================================================
CLASS DEFINITION OF END
FURTHER DETAILS RFC 5545, P. 50
*/

::class icalendar20_end public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")

   forward class (super)
/*
-----------------------------------------------------------------------
method for xml reading
```

```
*/
::method fromxml class
use arg ical_value

   o= self~new
   o~classname = "END"
   o~group_name = ''
   o~value_data~append(ical_value)

return o
/*
----------------------------------------------------------------------
method creates make hcalendar
*/
::method makehtml

   tmpstr = '</div>'

return tmpstr


/*
======================================================================
CLASS DEFINITION X-PROPERTY
FURTHER DETAILS RFC 5545, P. 9
*/

::class icalendar20_x_property public subclass icalendar20

/*
attributes and values variables of class
*/
::method init class

   self~attributes = .array~new
   self~values = .array~of("value_data")
   forward class (super)

/*
----------------------------------------------------------------------
method processing left side of read-in line
*/
::method leftside class
use arg o left
   tmp1 = left~changestr("\;","01"X)
   loop while tmp1 <> "" --assigns boolean values to attributes
   parse var tmp1 leftside ";" tmp1
      do
      parse var leftside denominator "=" denominations
            o~non_standard_attributes[denominator] = denominations
      end
   end

return o

/*
----------------------------------------------------------------------
method creates hcalendar
*/
::method makehtml
say self~class 'is not allowed yet. '-
'Cf. http://microformats.org/wiki/index.php?title=hcalendar-cheatsheet&oldid=36589'
return ''



/*
=========================ROUTINES=====================================
*/

/*
routine creates a getter method for attributes
*/
::routine makegetter
   parse arg name
   code="EXPOSE" name "; RETURN" name
   return code

----------------------------------------------------------------------

/*
routine creates a setter method for attributes
*/
::routine makesetter
   parse arg name
   code="EXPOSE" name "; USE ARG" name
   return code

----------------------------------------------------------------------

/*
routine replaces forbidden xml-signs; sequence is important!
otherwise already escaped signs would be escaped again; used for
```

```
hcalendar
*/
::routine escape
   parse arg value
   value=value~changestr("&","&amp;")
   value=value~changestr('"',"&quot;")
   value=value~changestr("'","&apos;")
   value=value~changestr("<","&lt;")
   value=value~changestr(">","&gt;")
   value=value~changestr("\;",";")
   value=value~changestr("\,",",")
   value=value~changestr("\:",":")
   return value
-------------------------------------------------------------------

/*
routine removes leading and ending line feeds, carriage returns
and strips the given character set
*/
::routine extended_strip
parse arg string

string = string~strip

line_feed = string~countStr("0A"X)
carriage_return = string~countStr("0D"X)

repetions = line_feed + carriage_return

do repetions

   if string~left(1) = .endofline  -
      | string~left(1) = "0A"X -
      | string~left(1) = "0D"X -
      then
         string = string~substr(2)

   string = string~strip

   length = string~length
   if string~right(1) = .endofline -
      | string~right(1) = "0A"X -
      | string~right(1) = "0D"X -
      then
         string = string~substr(1, length-1)

   string = string~strip
end

string = string~changestr("0A"X, '\n')
string = string~changestr("0D"X, '\n')

return string
-------------------------------------------------------------------

/*
routine backslash-escapes values
*/
::routine backslash_escape
   parse arg value
   value=value~changestr(";","\;")
   value=value~changestr(",","\,")
   value=value~changestr(":","\:")
   return value

::requires 'reader.rxj'
```

Code 35: icalendar_classes.rxj.

A.3.2 Reader

Purpose: preparation of data stored in vCard-, iCalendar- hCard or hCalendar files.

Important attributes: `content`, a queue, stores the end-result of the `import_fromFile()`-method, every item in the queue is an instance of the classes representing the standards.

Important methods:

`import_fromFile(String filename_input, String input_type)` defines the file which should be processed and in which format the data in it is stored. Only `vcard30`, `icalendar20`, `hcard`, or `hcalendar` are allowed values. This method return a queue with objects representing the properties of the standards.

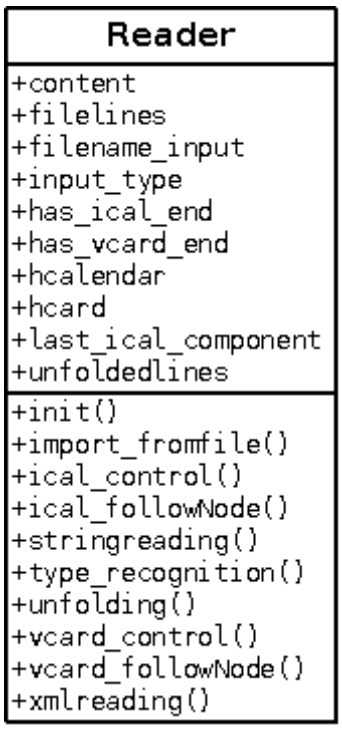

Figure 8: Reader Class.

Figure 8 shows the `Reader`-class with its methods and attributes.

A.3.2.1 reader.rxj

```
#!/usr/bin/rexx
/*
   Name:    reader.rxj
   Purpose: Collection of methods reading in files and distributing them
            to their according vCard/iCalendar classes
   Author:  Johannes Paul Bielohaubek
   Date:    2010-02-28
   Version: 1.0.2
   License:

    ----------------------- Apache Version 2.0 license ------------------------
      Copyright (C) 2010 Johannes Paul Bielohaubek

      Licensed under the Apache License, Version 2.0 (the "License");
      you may not use this file except in compliance with the License.
      You may obtain a copy of the License at
```

```
           http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
    ----------------------------------------------------------------------------
*/


::class reader public
::method input_type attribute
::method filename_input attribute
::method content attribute --array that stores
   --vCard/iCal objects
::method filelines attribute --array that stores
   --folded lines
::method unfoldedlines attribute --array that
   -- stores unfolded lines
::method has_vcard_end attribute --variable that
   --stores the information if an vcard_end
   --for xml parsing is needed or not
::method has_ical_end attribute --variable that
   --stores the information if an icalendar_end
   --for xml parsing is needed or not
::method hcard attribute
::method hcalendar attribute --attributes storing
   --if an hcard or an hcalendar is
   --encountered
::method last_ical_component attribute --stores
   --the last ical component that the
   --xml parser has encountered


::method init
   self~content = .queue~new
   self~filelines = .array~new
   self~unfoldedlines = .array~new
   self~has_vcard_end = .false
   self~has_ical_end = .false
   self~hcard = .false
   self~hcalendar = .false

--   .local~content=self~content--[sic!] kontrollieren!


/*
program part that separates text from xml files
*/
::method import_fromfile
use arg filename_input input_type

self~filename_input = filename_input

select
when input_type~upper = 'HCARD' then self~hcard = .true
when input_type~upper = 'HCALENDAR' then self~hcalendar = .true
otherwise self~input_type = input_type~upper
end


--self~input_type = input_type~upper --ensures that types are always in
            --uppercase


if self~hcard | self~hcalendar -
then
   self~xmlreading
else
   self~stringreading()

return self~content



/*
------------------------------------------------------------------------
method reads in data from file
*/
::method stringreading

filename = .stream~new(self~filename_input) --filename

do item over filename
   self~filelines~append(item)
end

self~unfolding

return
```

```
/*
-------------------------------------------------------------------
method unfolds and distributes data
requires that variable self~filelines (an array) contains data
*/
::method unfolding

i = 1
linenumber = self~filelines~items --number of data lines
e = 1

do while i <= linenumber


   if self~filelines[i]~substr(1,1) <> " " | self~filelines[i]~substr(1,1) <> "09"x
               --checks if line starts with a gap
   then
      do
      self~unfoldedlines[e] = self~filelines[i] --assigns element
      e = e + 1
      end
   else
      self~unfoldedlines[e-1] = self~unfoldedlines[e-1] || self~filelines[i]~substr(2)
               --adds aditional element
   i = i+1
end

   self~type_recognition

return


/*
method recognizes type and distributes data to corresponding
classes for further transformation
method requires variable self~unfoldedlines (an array)
contains data
*/
::method type_recognition

do line over self~unfoldedlines
   right_pos = line~verify(";:",M) --defines position of separator
   if right_pos < 2 then iterate --exception for faulty data
   grouping = line~verify(".",M) --checks if properties are grouped
   if grouping > right_pos then
      do
      classname = left(line~changestr("-","_"),right_pos-1)
      cls = value("." || self~input_type || "_" ||classname)
      group_name = ""
      end
   else
      do
      classname = substr(line~changestr("-","_"),-
            grouping+1, right_pos - grouping - 1)
      cls = value("." || self~input_type || "_" ||classname)
      if grouping <> 0 then group_name = left(line,grouping-1)
      else group_name = left(line,grouping)
      end



   --defines class for further transformation

   if cls=.nil then iterate

   vtype = value("." || self~input_type)
   class_check = vtype~send(subclasses) --creates array containing
                  --all subclasses
   input_type_length = self~input_type~length --counts characters
   rootclass = value("." || self~input_type || "_X_PROPERTY")

   if class_check~hasitem(cls) then --checks if subclass is available
      do
      o = cls~fromstring(line)
      --self~content~append(o)
      self~content~append(o)
      end
   else
      do
      if left(cls,input_type_length+4) = -
         "." || self~input_type || "_X_" then
         do
            tmpcls = rootclass~subclass(cls)
            o = tmpcls~fromstring(line)
            --self~content~append(o)
            self~content~append(o)
         end
      else say cls "not defined!"
      end
end

return
```

```
/*
--------------------------------------------------------------------
method reads in xml-data
programm part based on Rony G. Flatscher's
01_getText.rxj
03_listElementNames.rxj
05_listElementNamesIndentedWithAttributes.rxj
06_listElementNamesIndentedWithAttributes_DOES_NOT_USE_DTD.rxj
available in the bsf4oorexx sample section.
*/
::method xmlreading
/*begin of part taken from
06_listElementNamesIndentedWithAttributes_DOES_NOT_USE_DTD.rxj
*/
/* create an instance of the JAXP DocumentBuilderFactory*/
factory=bsf.loadClass("javax.xml.parsers.DocumentBuilderFactory")~newInstance

/* make sure that parser is set to be namespace aware*/
factory~setNamespaceAware(.true)

/* now create the DocumentBuilder object*/
parser=factory~newDocumentBuilder

/* create a Rexx object for error handling*/
eh=.errorHandler~new

/* create a Java proxy using the Rexx object for error handling*/
javaProxy=BsfCreateRexxProxy(eh, , "org.xml.sax.ErrorHandler")

/* set the parser's error handler*/
parser~setErrorHandler(javaProxy)

/* set the EntityResolver handler to our proxy*/
parser~setEntityResolver(BsfCreateRexxProxy(.EntityResolver~new, , "org.xml.sax.EntityResolver"))


/* parse the file, result is full DOM-tree*/
document=parser~parse(self~filename_input)

/* make important constants available via .local*/
clz=bsf.loadClass("org.w3c.dom.Node")
.local~CDATA_Section_Node=clz~CDATA_SECTION_NODE
.local~TEXT_Node         =clz~TEXT_NODE
.local~ELEMENT_NODE = clz~ELEMENT_NODE

/*end of part taken from
06_listElementNamesIndentedWithAttributes_DOES_NOT_USE_DTD.rxj
*/

if self~hcard then
   do
   self~vcard_followNode(document)
   self~vcard_control
   end
else nop
if self~hcalendar then
   do
   self~ical_followNode(document)
   self~ical_control
   end
else nop
return
/*
--------------------------------------------------------------------
method controls if vcard have at least the required properties
*/
::method vcard_control

n = 0 --variable that indicates if a 'n' property is used
begin = 0

--checks if n property is found
do runner over self~content
   if runner~class = .vcard30_n then n = n + 1
   if runner~class = .vcard30_begin then  begin = begin + 1
   else nop
end

--say 'n' n
--say 'begin' begin

angle_begin = 0

if n >= begin then nop --checks if every new vCard has the n-property or not
else
   do item over self~content
   if item~class = .vcard30_begin then angle_begin = angle_begin + 1
   --counts the vCards within the array
   --say 'angle_begin' angle_begin

   if angle_begin = begin then --looks for the vCard in which the control cercle has to start
   do
```

```
      if item~class = .vcard30_fn then --looks for the fn-property from which
                  --the n-property is extracted
      do
         string = ''
         do value over item~value_data
            string = string value
         end


         if string~words = 2 then --implied n-optimization only valid for
               --strings consisting of two words
            do

               if string~word(1)~verify(',',M) > 0 then
               --checks if the order of the name parts, a ',' indicates that
               --the string starts with the family name.
                  do
                     parse var string family_name given_name
                  end
               else
                  do
                     parse var string part1 part2

                     if part2~length = 1 | part2~verify('.',M) > 0 then
                     --looks for eg. "William Shakespeare" vs. "Shakespeare W."
                     do
                        part1 = family_name
                        part2 = given_name
                     end
                     else
                     do
                        part1 = given_name
                        part2 = family_name
                     end


                  end
               --creation of new property
               o=.vcard30_N~new
               o~value_data~append(family_name)
               o~value_data~append(given_name)
               o~value_data~append('')
               o~value_data~append('')
               o~value_data~append('')
               o~classname = "N"
               o~group_name = ''
               self~content~append(o)
               drop o

            end
         else --occur if implied n-optimization does not follow specified rules
            do
               o=.vcard30_N~new
               o~value_data~append('not indicated')
               o~classname = "N"
               o~group_name = ''
               self~content~append(o)
               drop o
            end



      end
      else nop
   end
   else nop
   end


if self~has_vcard_end then
   do
      o=.VCARD30_VERSION~new --adds version number to data
      o~value_data~append('3.0')
      o~classname = "VERSION"
      o~group_name = ''
      self~content~append(o)
      drop o
      o=.VCARD30_END~fromxml --adds end-property
      self~content~append(o)
      self~has_vcard_end = .false
   end
else nop

return


/*
-----------------------------------------------------------------------
method walks the document tree recursively honoring the requirements
for vcards
*/
::method vcard_followNode
```

```
  use arg node

deeper = .true --value stores if child elements should be
   --processed as well

if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
--cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
     do
      atts = node~getAttributes

      if atts~getLength>0 then    /* o.k. attributes defined for this element  */
         do
            do i=0 to atts~getLength-1   /* iterate over all attributes, 0-based */
               att=atts~item(i)     /* get attribute node                       */
               if att~getName="class" then --looks for the attribute type that is used
                           --to indicate hcalendar/hcard values
                  do
                     if att~getValue~verify(" ",M) = 0 then --checks if multiple values for html
                                 --attribute "class" are used or not
                        do
                           if att~getValue = "vcard" then
                              do
                                 if self~has_vcard_end = .false then --checks if the encountered
                                       --vcard is the first or not
                                    do
                                       o=.VCARD30_BEGIN~fromxml
                                       self~content~append(o)
                                       self~has_vcard_end = .true --indicates that a vcard
                                     --already has been processed
                                    end
                                 else
                                    do
                                       self~vcard_control
                                       o=.VCARD30_BEGIN~fromxml
                                       self~content~append(o)
                                       self~has_vcard_end = .true  --indicates that a vcard
                              -- already has been processed
                                    end
                              end

                           if att~getValue = "category" then --vCard: categories; hCard: category
                              do
                                 o=.VCARD30_CATEGORIES~fromxml(node, 'CATEGORIES')
                                 self~content~append(o)
                              end

                           else
                              do
                                 classname = att~getValue~changestr("-","_")~upper
                                 cls = value("." || 'VCARD30' || "_" ||classname)
                                 --say cls
                                 vtype = value("." || 'VCARD30')
                                 class_check = vtype~send(subclasses) --creates array containing
                                       --all subclasses

                                 if class_check~hasitem(cls) then --checks if subclass is available
                                    do
                                       o=cls~fromxml(node, classname)
                                       self~content~append(o)
                                    end
                                 else say cls "not defined!"
                              end
                        end
                     else
                        do
                           --say "multiple classes found!"
                           tmp_var = att~getValue~changestr("-","_")

                           select
                           when tmp_var~wordpos('agent') <> 0 -
                           & tmp_var~wordpos('vcard') <> 0 then --looks for a agent property
                                    --containing an embeded vCard
                              do
                                 deeper = .false --child nodes do not need to processed any further
                                 cls = value("." || 'VCARD30' || "_" || 'AGENT')
                                 --say cls
                                 o=cls~agentfromxml(node)
                                 self~content~append(o)
                              end

                           when tmp_var~wordpos('org') <> 0 -
                           & tmp_var~wordpos('vcard') <> 0 then --looks for an org element that
                                    --has an embedded vCard; not supported by
                                    --the vCard standard
                              do
                                 deeper = .false --child nodes do not need to processed any further
                              end

                           when tmp_var~wordpos('email') <> 0 -
                           | tmp_var~wordpos('url') <> 0 then --looks if an element is used
                          --that causes an unusual value exception extraction
                              do
```

```
                        classnames = att~getValue~changestr("-","_")~upper~makeArray(' ')

                        do classname over classnames
                        --say classname
                           cls = value("." || 'VCARD30' || "_" || classname)
                           --say cls

                           if classname = 'VCARD' then
                           do
                              if self~has_vcard_end = .false then
                              do
                                 o=.VCARD30_BEGIN~fromxml
                                 self~content~append(o)
                                 self~has_vcard_end = .true
                              end
                              else
                              do
                                 self~vcard_control
                                 o=.VCARD30_BEGIN~fromxml
                                 self~content~append(o)
                                 self~has_vcard_end = .true
                              end
                           end
                           else nop

                           if classname = 'VEVENT' then deeper = .false --child nodes do not
                                           --need to processed any further

                           vtype = value("." || 'VCARD30')
                           class_check = vtype~send(subclasses) --creates array containing
                           --all subclasses

                           if class_check~hasitem(cls) then --checks if subclass is
                                    -- available
                              do

                                 if classname = 'VEVENT' then deeper = .false
                                    --child nodes do not need to processed any further
                                           if classname = 'AGENT' then
                                    do
                                       o=cls~fromxml(node, classname)
                                       self~content~append(o)
                                    end

                                 if classname = "CATEGORY" then
                                          --vCard: categories; hCard: category
                                    do
                                       o=.VCARD30_CATEGORIES~fromxml(node, 'CATEGORIES')
                                       self~content~append(o)
                                    end

                                 else
                                    do
                                       o=cls~specialfromxml(node, classname)
                                       self~content~append(o)
                                    end
                              end
                           else say cls "not defined!"
                        end
                     end
                  otherwise --branch for ordinary multiple value classes in hCard
                     do
                        classnames = att~getValue~changestr("-","_")~upper~makeArray(' ')

                        do classname over classnames
                        --say classname
                           cls = value("." || 'VCARD30' || "_" || classname)
                           --say cls

                           if classname = 'VCARD' then
                           do

                              if self~has_vcard_end = .false then
                              do
                                 o=.VCARD30_BEGIN~fromxml
                                 self~content~append(o)
                                 self~has_vcard_end = .true
                              end
                              else
                              do
                                 self~vcard_control
                                 o=.VCARD30_BEGIN~fromxml
                                 self~content~append(o)
                                 self~has_vcard_end = .true
                              end
                           end
                           else nop

                           if classname = 'VEVENT' then deeper = .false
                                       --child nodes do not need to processed any further

                           vtype = value("." || 'VCARD30')
                           class_check = vtype~send(subclasses) --creates array containing
```

```
                                    --all subclasses

                                    if class_check~hasitem(cls) then
                                    --checks if subclass is available
                                       do

                                          if classname = 'VEVENT' then deeper = .false
                                          --child nodes do not need to processed any further
                                             if classname = 'AGENT' then
                                             do
                                                o=cls~fromxml(node, classname)
                                                self~content~append(o)
                                             end

                                          if classname = "CATEGORY" then
                                    --vCard: categories; hCard: category
                                             do
                                                o=.VCARD30_CATEGORIES~fromxml(node, 'CATEGORIES')
                                                self~content~append(o)
                                             end

                                          else
                                             do
                                                o=cls~fromxml(node, classname)
                                                self~content~append(o)
                                             end
                                       end
                                    else say cls "not defined!"
                                 end
                              end
                           end
                        end
                     end
                  else nop
               end
            end
         else nop
      end
   end
else nop

if deeper then
   do
      if node~hasChildNodes then
         do
            children=node~getChildNodes /* get NodeList */

            loop i=0 to children~length-1  /* 0-based indexes! */
            self~vcard_followNode(children~item(i))  /* recurse */
            end
         end
   end

return

/*
-----------------------------------------------------------------------
method controls if vcard have at least the required properties
*/
::method ical_control

i_begin = .false
i_end = .false
i_version = .false
i_prodid = .false

--checks if minimum properties are found
do runner over self~content
   if runner~class = .icalendar20_begin then
      if runner~value_data~hasitem('VCALENDAR') then i_begin = .true
   else nop

   if runner~class = .icalendar20_version then i_version = .true

   if runner~class = .icalendar20_prodid then i_prodid = .true

end

if i_begin = .false then
do
   o = .icalendar20_begin~new
   o~classname = "BEGIN"
   o~group_name = ''
   o~value_data~append('VCALENDAR')
   self~content~push(o)
   drop o
end

if self~last_ical_component <> 'VCALENDAR' then
do
   o = .icalendar20_dtstamp~new
   o~classname = "DTSTAMP"
   o~group_name = ''
   date = .datetime~new
```

```
   stamp = date~standarddate||'T'||date~hours||date~minutes-
      ||date~seconds||'Z'
   o~value_data~append(stamp)
   self~content~append(o)
   drop o

   o = .icalendar20_uid~new
   o~classname = "UID"
   o~group_name = ''
   string = 'h0451119@wu.ac.at-'||stamp||'-'random(10000)
   o~value_data~append(string)
   self~content~append(o)
   drop o

   o = .icalendar20_end~new
   o~classname = "END"
   o~group_name = ''
   o~value_data~append(self~last_ical_component)
   self~content~append(o)
   drop o
end

if i_version = .false then
do
   o = .icalendar20_version~new
   o~classname = "VERSION"
   o~group_name = ''
   o~value_data~append('2.0')
   self~content~append(o)
   drop o
end

--say prodid i_prodid

if i_prodid = .false then
do
   o = .icalendar20_prodid~new
   o~classname = "PRODID"
   o~group_name = ''
   --datetime=.datetime~new
   prodid = 'claroRexx_h0451119@wu.ac.at'-- || datetime
   o~value_data~append(prodid)
   self~content~append(o)
   drop o
end
else nop

o = .icalendar20_end~new
o~classname = "END"
o~group_name = ''
o~value_data~append('VCALENDAR')
self~content~append(o)

return


/*
-----------------------------------------------------------------------
method walks the document tree recursively honoring the requirements
for icalendar
*/
::method ical_followNode
  use arg node

deeper = .true --value stores if child elements should be
   --processed as well

if node~getNodeType=1 then --checks element nodes, "1" indicating
--element nodes,
--cf. http://java.sun.com/javase/6/docs/api/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE
   do
      atts = node~getAttributes

      if atts~getLength>0 then    /* o.k. attributes defined for this element  */
         do
            do i=0 to atts~getLength-1   /* iterate over all attributes, 0-based */
               att=atts~item(i)     /* get attribute node                       */
               if att~getName="class" then --looks for the attribute type that is used
                           --to indicate hcalendar/hcard values
                  do
                     if att~getValue~verify(" ",M) = 0 then --checks if multiple values for html
                                 --attribute "class" are used or not
                        do
                           if att~getValue = "organiser" then
                              do
                                 o=.icalendar20_organizer~fromxml(node, classname)
                                 self~content~append(o)
                              end

                           if att~getValue = "category" then --vCard: categories; hCard: category
                              do
                                 o=.icalendar20_CATEGORIES~fromxml(node, 'CATEGORIES')
                                 self~content~append(o)
```

```
            end

        if att~getValue = "vcalendar" then
            do
                if self~has_ical_end = .false then
                    do
                        o=.icalendar20_begin~fromxml('VCALENDAR')
                        self~content~append(o)
                        self~has_ical_end = .true
                        self~last_ical_component = 'VCALENDAR'
                    end
                else
                    do
                        o = .icalendar20_version~new
                        o~classname = "VERSION"
                        o~group_name = ''
                        o~value_data~append('2.0')
                        self~content~append(o)
                        drop o

                        o = .icalendar20_prodid~new
                        o~classname = "PRODID"
                        o~group_name = ''
                        --datetime=.datetime~new
                        prodid = 'claroRexx_h0451119@wu.ac.at'-- || datetime
                        o~value_data~append(prodid)
                        self~content~append(o)
                        drop o

                        o=.icalendar20_end~fromxml('VCALENDAR')
                        self~content~append(o)
                        drop o
                        o=.icalendar20_begin~fromxml('VCALENDAR')
                        self~content~append(o)
                        self~last_ical_component = 'VCALENDAR'
                    end
            end

        if att~getValue = "vevent" then
            do
                if self~has_ical_end = .false then
                    do
                        o=.icalendar20_begin~fromxml('VEVENT')
                        self~content~append(o)
                        self~has_ical_end = .true
                        self~last_ical_component = 'VEVENT'
                    end
                else
                    do
                        o = .icalendar20_dtstamp~new
                        o~classname = "DTSTAMP"
                        o~group_name = ''
                        date = .datetime~new
                        stamp = date~standarddate||'T'||date~hours||date~minutes-
                        ||date~seconds||'Z'
                        o~value_data~append(stamp)
                        self~content~append(o)
                        drop o

                        o = .icalendar20_uid~new
                        o~classname = "UID"
                        o~group_name = ''
                        string = 'h0451119@wu.ac.at-'||stamp||'-'random(10000)
                        o~value_data~append(string)
                        self~content~append(o)
                        drop o
                        o=.icalendar20_end~fromxml(self~last_ical_component)
                        self~content~append(o)
                        drop o
                        o=.icalendar20_begin~fromxml('VEVENT')
                        self~content~append(o)
                        self~last_ical_component = 'VEVENT'
                    end
            end

        if att~getValue = "vtodo" then
            do
                deeper = .false --must be removed when component allowed
            /* iCalendar component not yet defined!
                if self~has_ical_end = .false then
                    do
                        o=.icalendar20_begin~fromxml('VTODO')
                        self~content~append(o)
                        self~has_ical_end = .true
                        self~last_ical_component = 'VTODO'
                    end
                else
                    do
                        o=.icalendar20_end~fromxml(self~last_ical_component)
                        self~content~append(o)
                        drop o
                        o=.icalendar20_begin~fromxml('VTODO')
                        self~content~append(o)
```

```
                                    self~last_ical_component = 'VTODO'
                                end
*/
                            end

                        if att~getValue = "vjournal" then
                            do
                                deeper = .false --must be removed when component allowed
                                /* iCalendar component not yet defined!
                                if self~has_ical_end = .false then
                                    do
                                        o=.icalendar20_begin~fromxml('VJOURNAL')
                                        self~content~append(o)
                                        self~has_ical_end = .true
                                        self~last_ical_component = 'VJOURNAL'
                                    end
                                else
                                    do
                                        o=.icalendar20_end~fromxml(self~last_ical_component)
                                        self~content~append(o)
                                        drop o
                                        o=.icalendar20_begin~fromxml('VJOURNAL')
                                        self~content~append(o)
                                        self~last_ical_component = 'VJOURNAL'
                                    end
*/
                            end

                        if att~getValue = "vfreebusy" then
                            do
                                deeper = .false --must be removed when component allowed
/* iCalendar component not yet defined!
                                if self~has_ical_end = .false then
                                    do
                                        o=.icalendar20_begin~fromxml('VFREEBUSY')
                                        self~content~append(o)
                                        self~has_ical_end = .true
                                        self~last_ical_component = 'VFREEBUSY'
                                    end
                                else
                                    do
                                        o=.icalendar20_end~fromxml(self~last_ical_component)
                                        self~content~append(o)
                                        drop o
                                        o=.icalendar20_begin~fromxml('VFREEBUSY')
                                        self~content~append(o)
                                        self~last_ical_component = 'VFREEBUSY'
                                    end
*/
                            end

                        if att~getValue = "vtimezone" then
                            do
                                deeper = .false --must be removed when component allowed
                                    /* iCalendar component not yet defined!
                                if self~has_ical_end = .false then
                                    do
                                        o=.icalendar20_begin~fromxml('VTIMEZONE ')
                                        self~content~append(o)
                                        self~has_ical_end = .true
                                        self~last_ical_component = 'VTIMEZONE '
                                    end
                                else
                                    do
                                        o=.icalendar20_end~fromxml(self~last_ical_component)
                                        self~content~append(o)
                                        drop o
                                        o=.icalendar20_begin~fromxml('VTIMEZONE ')
                                        self~content~append(o)
                                        self~last_ical_component = 'VTIMEZONE '
                                    end
*/
                            end

                        if att~getValue = "valarm" then
                            do
                                deeper = .false --must be removed when component allowed
                                    /* iCalendar component not yet defined!
                                if self~has_ical_end = .false then
                                    do
                                        o=.icalendar20_begin~fromxml('VALARM')
                                        self~content~append(o)
                                        self~has_ical_end = .true
                                        self~last_ical_component = 'VALARM'
                                    end
                                else
                                    do
                                        o=.icalendar20_end~fromxml(self~last_ical_component)
                                        self~content~append(o)
                                        drop o
                                        o=.icalendar20_begin~fromxml('VALARM')
                                        self~content~append(o)
                                        self~last_ical_component = 'VALARM'
                                    end
*/
```

```
                              end


                           else
                              do
                                 classname = att~getValue~changestr("-","_")~upper
                                 cls = value("." || 'ICALENDAR20' || "_" ||classname)
                                 --say cls
                                 vtype = value("." || 'ICALENDAR20')
                                 class_check = vtype~send(subclasses) --creates array containing
                                          --all subclasses

                                 if class_check~hasitem(cls) then --checks if subclass is available
                                    do
                                       o=cls~fromxml(node, classname)
                                       self~content~append(o)
                                    end
                                 else say cls "not defined!"
                              end
                        end
                     else
                        do
                           --say "multiple classes found!"
                           tmp_var = att~getValue~changestr("-","_")

                           classnames = att~getValue~changestr("-","_")~upper~makeArray(' ')

                           do classname over classnames
                              --say classname
                              cls = value("." || 'ICALENDAR20' || "_" || classname)
                              --say cls
                              vtype = value("." || 'ICALENDAR20')
                              class_check = vtype~send(subclasses) --creates array containing
                              --all subclasses

                              if class_check~hasitem(cls) then --checks if subclass is available
                                 do
                                    o=cls~specialfromxml(node, classname)
                                    self~content~append(o)
                                 end
                              else say cls "not defined!"
                           end
                        end
                  end
               else nop
            end
         end
      else nop
   end
else nop

if deeper then
   do
      if node~hasChildNodes then
         do
            children=node~getChildNodes

            loop i=0 to children~length-1
            self~ical_followNode(children~item(i))
            end
         end
   end

return



-----------------------------------

::class ErrorHandler /* will handle "org.xml.sax.ErrorHandler" events*/

::method unknown /* handles "warning", "error" and "fatalError" events */
  use arg methName, argArray  /* fetch arguments*/
  exception=argArray[1] /* retrieve SAXException argument*/
                        /* display error message*/
  .error~say(methName":"  "line="exception~getLineNumber",col="exception~getColumnNumber":" pp(except
ion~getMessage))

-----------------------------------
/*begin of part taken from
06_listElementNamesIndentedWithAttributes_DOES_NOT_USE_DTD.rxj
*/
::class "EntityResolver" /* implements "org.xml.sax.EntityResolver"      */

::method resolveEntity  /* returns an empty DTD InputSource to satisfy the SAX parser  */
  use arg publicID, systemID

      /* return an empty DTD instead; one could also supply a DTD from the local
         file system, a database or another URL instead! */
  charArray=.bsf~new("java.lang.String", "<!-- empty external DTD -->")~toCharArray
  reader=.bsf~new("java.io.CharArrayReader", charArray)  /* create a Reader object  */
  return .bsf~new("org.xml.sax.InputSource", reader)     /* create an InputSource   */
/*end of part taken from
```

```
06_listElementNamesIndentedWithAttributes_DOES_NOT_USE_DTD.rxj
*/


/*
----------------------------------------------------------------------
*/
::requires 'vcard_classes.rxj'
::requires 'icalendar_classes.rxj'
::requires BSF.CLS
```

Code 36: reader.rxj.

A.3.3 Interfaces

Here are the codes for the interfaces which are created for this thesis.

A.3.3.1 claro_GUI.rxj

```
#!/usr/bin/rexx
/*
   Name:    claro_GUI.rxj
   Purpose: GUI for vCard/hCard/iCalendar/hCalendar processing program
   Author:  Johannes Paul Bielohaubek
   Date:    2010-05-04
   Version: 1.0.2
   License:

    ------------------------ Apache Version 2.0 license -------------------------
       Copyright (C) 2010 Johannes Paul Bielohaubek

       Licensed under the Apache License, Version 2.0 (the "License");
       you may not use this file except in compliance with the License.
       You may obtain a copy of the License at

           http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
    -----------------------------------------------------------------------------
*/

/*
============================GUI=====================================
====================================================================
Graphical user interface
program bases on
RGF:
http://wi.wu-wien.ac.at:8002/rgf/rexx/orx20/2009_orx20_BSF4Rexx-20091031-article.pdf, Figure 2
Lee Peedin (available sample section of ooRexx)
fileopendialog.rxj
filesavedialog.rxj
*/

--event hander for window closing
eh=.eventHandler~new

--proxies for buttons
choose = .choose~new
convert = .convert~new
save = .save~new

--proxies for radio buttons
rb_input = .radio_input~new
rb_output = .radio_make~new

--creation of interfaces for listeners

--create a RexxProxy to serve two Java listeners
proxy=BsfCreateRexxProxy(eh, , "java.awt.event.ActionListener", "java.awt.event.WindowListener")

--rexxproxy for buttons
choose_proxy=BsfCreateRexxProxy(choose, , "java.awt.event.ActionListener")
convert_proxy=BsfCreateRexxProxy(convert, , "java.awt.event.ActionListener")
save_proxy=BsfCreateRexxProxy(save, , "java.awt.event.ActionListener")

--rexxproxy for radiobuttons
radio_but_proxy1=BsfCreateRexxProxy(rb_input, , "java.awt.event.ActionListener")
radio_but_proxy2=BsfCreateRexxProxy(rb_output, , "java.awt.event.ActionListener")
```

```
-----------------------------------------------------

--creation of a Java swing window
.bsf~bsf.import('javax.swing.JFrame','JFrame')
tmpWin = .JFrame~new('claro - hCard/vCard/iCal/hCal-converter')
tmpWin~addWindowListener(proxy) -- add our event handler to button

--creation of buttons
.bsf~bsf.import("javax.swing.JButton", "JButton")
fchooserBut = .JButton~new('Choose file!')
fchooserBut~addActionListener(choose_proxy) -- add our event handler to button
fchooserBut~setToolTipText('Press button in order to choose file for processing.')
--tool tip gives further details about object when mouse hovers over it

exitBut = .JButton~new('EXIT')
exitBut~addActionListener(proxy) -- add our event handler to button
exitBut~setToolTipText('Press button in order to leave the program.')
--tool tip gives further details about object when mouse hovers over it

convertBut = .JButton~new('Convert')
convertBut~addActionListener(convert_proxy) -- add our event handler to button
convertBut~setToolTipText('Press button in order to convert one standard into another.')
--tool tip gives further details about object when mouse hovers over it

saveBut = .JButton~new('Save as ...')
saveBut~addActionListener(save_proxy) -- add our event handler to button
saveBut~setToolTipText('Press button in order to choose where to save the new file.'-
                       ' The filename extension for vCard is .vcf, for iCalendar .ics and for the'-
                       'h-standards .html.')
--tool tip gives further details about object when mouse hovers over it

--creation of radiobutton
.bsf~bsf.import("javax.swing.JRadioButton", "JRadioButton")
rb1 = .JRadioButton~new('vcard v 3.0')
rb1~setActionCommand("vcard30")
--rb1~setSelected(.true)
rb1~addActionListener(radio_but_proxy1)
rb1~setToolTipText('Please choose the standard for the read-in file.')
--tool tip gives further details about object when mouse hovers over it

rb2 = .JRadioButton~new('hcard')
rb2~setActionCommand("hcard")
rb2~addActionListener(radio_but_proxy1)
rb2~setToolTipText('Please choose the standard for the read-in file.')
--tool tip gives further details about object when mouse hovers over it

rb3 = .JRadioButton~new('icalendar v 2.0')
rb3~setActionCommand("icalendar20")
rb3~addActionListener(radio_but_proxy1)
rb3~setToolTipText('Please choose the standard for the read-in file.')
--tool tip gives further details about object when mouse hovers over it

rb4 = .JRadioButton~new('hcalendar')
rb4~setActionCommand("hcalendar")
rb4~addActionListener(radio_but_proxy1)
rb4~setToolTipText('Please choose the standard for the read-in file.')
--tool tip gives further details about object when mouse hovers over it

--creation of a buttongroup for the radio buttons
.bsf~bsf.import("javax.swing.ButtonGroup", "ButtonGroup")
bgroup1 = .ButtonGroup~new
bgroup1~~add(rb1) ~~add(rb2) ~~add(rb3) ~~add(rb4)

--creation of a panel for the radio buttons
.bsf~bsf.import("javax.swing.JPanel", 'JPanel')
panel1 = .JPanel~new
panel1~~add(rb1)
panel1~~add(rb2)
panel1~~add(rb3)
panel1~~add(rb4)
panel1~setToolTipText('Please choose the standard for the read-in file.')
--tool tip gives further details about object when mouse hovers over it

--creation of radiobuttons
rb5 = .JRadioButton~new('vCard/iCalendar')
rb5~setActionCommand("makestring")
--rb5~setSelected(.true)
rb5~addActionListener(radio_but_proxy2)
rb5~setToolTipText('Please choose the standard for the output file.')
--tool tip gives further details about object when mouse hovers over it

rb6 = .JRadioButton~new('hCard/hCalendar')
rb6~setActionCommand("makehtml")
rb6~addActionListener(radio_but_proxy2)
rb6~setToolTipText('Please choose the standard for the output file.')
--tool tip gives further details about object when mouse hovers over it

mydir = .directory~new
.local~mydir = mydir



.bsf~bsf.import("javax.swing.ButtonGroup", "ButtonGroup")
bgroup2 = .ButtonGroup~new
```

```
bgroup2~~add(rb5) ~~add(rb6)

.bsf~bsf.import("javax.swing.JPanel", 'JPanel')
panel2 = .JPanel~new
panel2~~add(rb5)
panel2~~add(rb6)
panel2~setToolTipText('Please choose the standard for the output file.')
--tool tip gives further details about object when mouse hovers over it


--labels
.bsf~bsf.import("javax.swing.JLabel", "JLabel")
label1 = .JLabel~new('1.) Select file for conversion')
label2 = .JLabel~new('2.) Input is ...')
inp_file = .JLabel~new
.local~inp_file=inp_file --variable stores the value
        --of the input file

label3 = .JLabel~new('3.) Output is ...')
label4 = .JLabel~new('4.) Please enter name for new file')
op_file = .JLabel~new
.local~op_file=op_file
 --variable stores the name
        --of the output file

mydir = .directory~new
.local~mydir = mydir --directory is made locally available
        --and stores the values of the radio buttons

--creation of a Java swing tool tip
.bsf~bsf.import('javax.swing.JToolTip','JToolTip')
tooltip = .JToolTip~new
tooltip~~setVisible(.true)


--gridlayout
.bsf~bsf.import("java.awt.GridLayout", "Gridlayout")
gridlayout = .Gridlayout~new(12,1)
--/*
--border
--creates a small space around the components
.bsf~bsf.import("javax.swing.BorderFactory", "BorderFactory")
.bsf~bsf.import("javax.swing.border.Border", "Border")

--emptyBorder = .Border~new
emptyBorder = .BorderFactory~createEmptyBorder(5, 20, 0, 20)
--defines the borders in pixel contraclockwise
label1~setBorder(emptyBorder)
label2~setBorder(emptyBorder)
label3~setBorder(emptyBorder)
label4~setBorder(emptyBorder)
fchooserBut~setBorder(emptyBorder)
saveBut~setBorder(emptyBorder)
convertBut~setBorder(emptyBorder)
exitBut~setBorder(emptyBorder)
.inp_file~setBorder(emptyBorder)
.op_file~setBorder(emptyBorder)
panel1~setBorder(emptyBorder)
panel2~setBorder(emptyBorder)
--*/
--prepare window and show it, using cascading messages (two twiddles '~')
tmpWin~setlayout(gridlayout)
tmpwin~~add(label1)
tmpWin~~add(fchooserBut)
tmpWin~~add(.inp_file)
tmpwin~~add(label2)
tmpWin~~add(panel1)
tmpWin~~add(label3)
tmpWin~~add(panel2)
tmpWin~~add(label4)
tmpWin~~add(saveBut)
tmpWin~~add(.op_file)
tmpWin~~add(convertBut)
tmpWin~~add(exitBut)
tmpWin~~pack ~~show ~~setvisible(.true)

eh~waitForExit --waits on a control variable to be changed to .true

/*
-----------------------------------------------------------------------
class stores methods for the occurence if the window needs to be
closed or the "EXIT"-button is pressed
*/
::CLASS "eventHandler"
::method init                   -- set bExitProgram attribute to .false
  expose bExitProgram
  bExitProgram=.false

::method unknown                -- do nothing if other events are raised
  use arg methName, args
  -- uncomment the following line to see the unhandled events
  -- say "unknown: method="pp(methName) "eventObject="pp(args[1]~toString)
```

```
::method windowClosing           -- clear bExitProgram attribute
  expose bExitProgram
  bExitProgram=.true

::method actionPerformed         -- clear bExitProgram attribute
  expose bExitProgram
  bExitProgram=.true

::method waitForExit             -- wait for bExitProgram attribute to gain a specific value
  expose bExitProgram

  guard on when bExitProgram=.true

/*
---------------------------------------------------------------------
class stores methods for the choice of an input file
*/
::class 'choose'
::method actionPerformed

-- Create a file chooser by importing the Java class
    fchooser = .bsf~new('javax.swing.JFileChooser')

/* create an instance of the RexxFileFilter */
description="*.vcf;*.vcard;*.ical;*.ics;*.ifb;*.icalendar;*.xml;*.htm;*.html" /* info for user
feedback */
filter=".vcf .vcard .ical .ics .ifb .icalendar .xml .htm .html" /* blank delimited
list of extensions */
rexxObject=.RexxFileFilter~new(description, filter)/* create the filter */

-- Create a frame to place the file chooser in by importing the Java class
    frame = .bsf~new('javax.swing.JFrame','Frame')
-- Set the current directory by importing the java.io.File class
--(the '.' indicates the current folder)
    fchooser~setCurrentDirectory(.bsf~new("java.io.File", "."))

/* create a RexxProxy, serving as the implementation for the abstract class*/
filter_proxy=BsfCreateRexxProxy(rexxObject, , "javax.swing.filechooser.FileFilter")

-- Give our file chooser a dialog
    fchooser~setDialogTitle('Choose file')

fchooser~setFileFilter(filter_proxy) /* set the filter */
-- Show the dialog and display back to the user their selection
    myfile = fchooser~showOpenDialog(frame)

    if myfile = 0 then
        do
                .inp_file~setText(fchooser~getSelectedFile~getPath)
                say 'Selectec File:' pp(fchooser~getSelectedFile~getPath)
            --file_path = fchooser~getSelectedFile~getPath
           -- file_name = fchooser~getSelectedFile~getName
        end
    else
        do
          say 'Please, choose a file or press "Exit" to leave!'
                .inp_file~setText('Please, choose a file or press "EXIT" to leave!')
        end

::method unknown

/*
---------------------------------------------------------------------
class stores methods to initiate the conversion of the standards
*/
::class 'convert'
::method actionPerformed
filename_input = .inp_file~getText
input_type =  .mydir[ip_type] --specifies vcard30, hcard, icalendar20 etc.
filename_output = .op_file~getText
write_method = .mydir[make] --makehtml or makestring

a = .reader~new
content = a~import_fromfile(filename_input, input_type)

if write_method = 'makehtml' then a = lineout(filename_output,'<div>')
--guarantees that there is always a root element
do line over content
        if line~send(write_method)~length > 0 then
                a = lineout(filename_output,line~send(write_method))
end
if write_method = 'makehtml' then a = lineout(filename_output,'</div>')
--guarantees that there is always a root element

say "File" filename_output "is created. Please press 'Exit' to leave program."
.op_file~setText('New file created. Press "Exit" to leave program.')
::method unknown

/*
---------------------------------------------------------------------
class stores methods for defining the name and the place where the
new file should be stored
*/
::class 'save'
```

```
::method actionPerformed
-- Create a file chooser by importing the Java class
    fchooser = .bsf~new('javax.swing.JFileChooser')
-- Create a frame to place the file chooser in by importing the Java class
    frame = .bsf~new('javax.swing.JFrame','Frame')
-- Set the current directory by importing the java.io.File class
--(the '.' indicates the current folder)
    fchooser~setCurrentDirectory( .bsf~new("java.io.File", ".") )
-- Give our file chooser a dialog
    fchooser~setDialogTitle('Java Swing File SAVE Dialog')
-- Show the SAVE dialog and display back to the user their selection
    myfile = fchooser~showSaveDialog(frame)
-- Application is waiting on results of the dialog
-- Test the results of the SAVE dialog
    if myfile = 0 then
        do
           .op_file~setText(fchooser~getSelectedFile~getPath)
           say 'Selected Saving Place' pp(fchooser~getSelectedFile~getPath)
           --.op_file~setText(fchooser~getSelectedFile~getName)
        end
    else
        do
        say 'Please, indicate an output file or press "EXIT" to leave!'
        .op_file~setText('Please, indicate an output file or press "EXIT" to leave!')
        end

::method unknown

/*
-----------------------------------------------------------------------
class stores methods for the radio buttons dedicated to the
input type
*/
::class 'radio_input'
::method unknown

::method actionPerformed
use arg event
.mydir[ip_type] = event~getActionCommand()

/*
-----------------------------------------------------------------------
class stores methods for the radio buttons dedicated to the
method that is used for the output
*/
::class 'radio_make'
::method unknown

::method actionPerformed
use arg event
.mydir[make] =  event~getActionCommand()

/*
-----------------------------------------------------------------------
class stores methods for the file filter
*/
::class RexxFileFilter
::attribute description /* description for the filter instance */
::attribute filter /* filter */

::method init /* constructor: set bExitProgram attribute value */
        expose description filter /* attributes to hold description and filter */
        use strict arg description, filter /* assign arguments to attributes */


/* implementations of the abstract Java methods */
::method getDescription /* implementation of "getDescription" */
        expose description /* access attribute directly */
        return description /* return value attribute refers to */

::method accept /* implementation of "accept" */
        expose filter /* access attribute directly */
        use arg fileObject /* get java.io.File object to work with */

        if fileObject~isDirectory then /* a directory in hand? */
                return .true /* always include a directory */

        fName=fileObject~getName /* get the filename */
        pos=fName~lastpos('.') /* get last dot in name */

        if pos=0 then /* no extension? */
                return .false /* do not accept file */

        fExt =fname~substr(pos) /* extract file extension without dot */
        return filter~caselessWordPos(fExt) > 0 /* return .true, if found, 0 else */

::requires BSF.CLS
::requires 'reader.rxj'
```

Code 37: claro_GUI.rxj.

A.3.3.2 CMD_GUI.rex

```
#!/usr/bin/rexx
/*
   Name:    CMD_GUI.rex
   Purpose: Program retrieves variables needed for reader.rxj in order to process
            hCards, vCards, iCalendars, hCalendars
   Author:  Johannes Paul Bielohaubek
   Date:    2010-05-04
   Version: 1.0.2
   License:

    ------------------------ Apache Version 2.0 license -------------------------
       Copyright (C) 2010 Johannes Paul Bielohaubek

       Licensed under the Apache License, Version 2.0 (the "License");
       you may not use this file except in compliance with the License.
       You may obtain a copy of the License at

           http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
    -------------------------------------------------------------------------------
*/
/*
external access to vCard-classes
*/
say "choose a file to be read-in:"
parse pull filename_input

say "which type of file is it? (choose between 'vcard30', 'hcard',"
say"'icalendar20' or 'hcalendar')"
parse pull input_type

say "what is the name of the new file?"
parse pull filename_output

say "what do you want to do? enter 'makehtml' for hCard/hCalendar"
say "or 'makestring' for vCard/iCalendar!"
parse pull write_method --makehtml or makestring

a = .reader~new
content = a~import_fromfile(filename_input, input_type)

say "make output (y/n)?"
parse pull decision

if decision = 'y' then
do
if write_method = 'makehtml' then a = lineout(filename_output,'<div>')
--guarantees that there is always a root element
do line over content
        if line~send(write_method)~length > 0 then
                a = lineout(filename_output,line~send(write_method))
end
if write_method = 'makehtml' then a = lineout(filename_output,'</div>')
--guarantees that there is always a root element
end
else nop

::requires 'reader.rxj'
```

Code 38: CMD_GUI.rex.