Institut für Betriebswirtschaftslehre und Wirtschaftsinformatik

Automatisierung von Windows Anwendungen (6)

Object Rexx Umgebungen (.local, .environment),
"Das große Bild",
Unknown, Forward

Prof. Dr. Rony G. Flatscher

Wirtschaftsuniversität Wien ■ Augasse 2-6 ■ A-1090 Wien

Object Rexx Umgebung (Environment), 1

- Object Rexx-Programme
  - Abarbeitungsreihenfolge
    - Syntaxüberprüfung
    - Abarbeitung von Direktiven
    - Start des Object Rexx-Programms
  - Mögliche Fragen
    - Wie werden aufgefundene Routinen, Methoden und Klassen zur Verfügung gestellt?
    - Wie werden öffentliche Routinen und Klassen sichtbar gemacht?
    - Gibt es eine Möglichkeit für Rexx-Programm(teile), Objekte mteinander zu teilen (Kopplung)?

Object Rexx Umgebung (Environment), 2

- Der Interpreter baut vier Verzeichnisobjekte (vom Typ: Directory) auf
  - Das Verzeichnis "Source"
    - Enthält alle einem Programm/Modul zur Verfügung stehenden Routinen, Methoden und Klassen
      - Im Programm/Modul definierte Routinen, Methoden und Klassen
      - Öffentliche Routinen und Klassen eines aufgerufenen (CALL oder ::REQUIRES-Direktive) Programms/Moduls
      - .METHODS, ein Verzeichnis der **"freilaufenden"** Methoden oder unbelegt (Zeichenkette: ".METHODS")
      - Nicht zugänglich gemacht, nur für das Laufzeitsystem verfügbar
    - Für jedes Programm/Modul wird vom Laufzeitsystem aus ein **individuelles** "Source"-Verzeichnis aufgebaut und gepflegt!
      - Die öffentlichen Routinen und Klassen **des zuletzt aufgerufenen Programms/Moduls ersetzen alle** entsprechenden öffentlichen Routinen und Klassen von zuvor aufgerufenen Programmen/Modulen

Object Rexx Umgebung (Environment), 3

- Das lokale Verzeichnis "Local"
  - Über das Umgebungssymbol **.LOCAL** zugänglich
  - Wird für jeden Prozeß eigens angelegt
  - Enthält prozeßbezogene Objekte, z.B.

    .error (Monitorobjekt für Fehlermeldungen),
    .input (Monitorobjekt für Eingaben),
    .output (Monitorobjekt für Ausgaben)
    [Monitorobjekte erlauben den Austausch der überwachten Objekte!]

    .stderr (Streamobjekt , das .error überwacht),
    .stdin (Streamobjekt , das .input überwacht),
    .stdout (Streamobjekt , das .output überwacht):

Object Rexx Umgebung (Environment), 4

- Das Verzeichnis "Environment"
  - Über das Umgebungssymbol **.ENVIRONMENT** zugänglich
    - Wird für jeden Prozeß eigens angelegt
  - Enthält eine Reihe wichtiger Objekte, z.B.
    - ***Sämtliche*** öffentlichen Klassen von Object Rexx
    - Grundlegenden Objekte, z.B.

      .nil das "Nichtobjekt" ("Nil" ist Englisch für: "Nichts"!): zeigt an, dass kein Objekt verfügbar ist

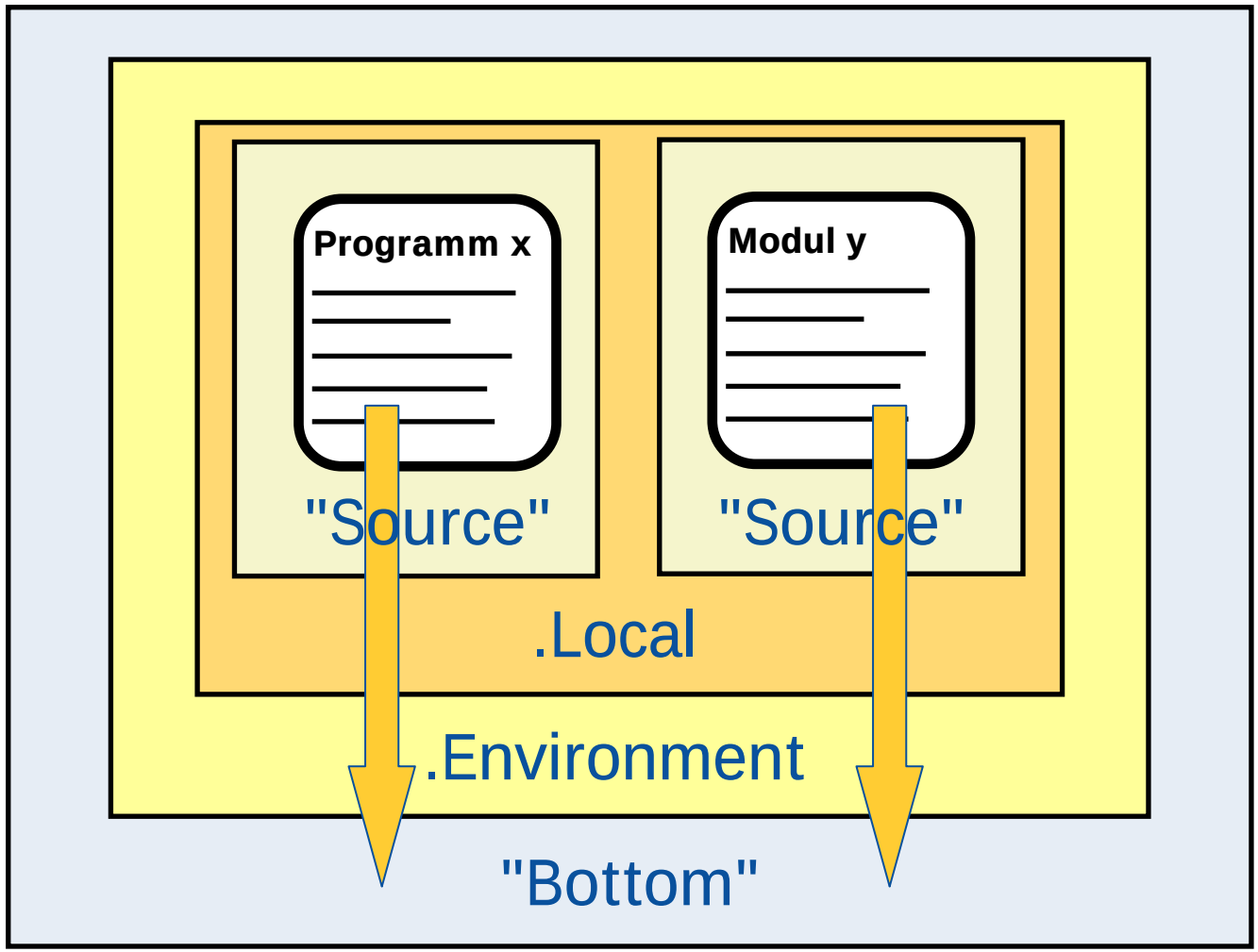
      .true der Boole'sche Wert "1" für wahr

      .false der Boole'sche Wert "0" für falsch
- Das Verzeichnis "Bottom"
  - Weitere, vom Laufzeitsystem benötigte Objekte
  - Nicht zugänglich gemacht, nur für das Laufzeitsystem verfügbar

Umgebungssymbole (Environment Symbols), 1

- Bezeichner beginnen immer mit einem Punkt
  - Systemseitige Umgebungssymbole des Interpreters weisen im Bezeichner keinen weiteren Punkt auf
  - Benutzerdefinierte Umgebungssymbole sollen per Konvention einen weiteren Punkt mitten in ihrem Bezeichner aufweisen
- Suche in der Umgebung
  - **Laufzeitsystem** entfernt den Punkt am Anfang des Bezeichners und **durchsucht alle Umgebungsverzeichnisse** in folgender fixierten Reihenfolge:
    1. Source-Verzeichnis
    2. Local-Verzeichnis (als .local zugänglich)
    3. Environment-Verzeichnis (als .environment zugänglich)
    4. Bottom-Verzeichnis

Umgebungssymbole (Environment Symbols), 2



Umgebungssymbole (Environment Symbols), 3

  - Wenn kein Eintrag mit der Bezeichnung des Umgebungssymbols zur Verfügung steht, dann wird als Ergebnis das in Großbuchstaben übersetzte Umgebungssymbol (eine Zeichenkette) zurückgegeben

- **Warnung!**
  - Keine Klassen definieren, die dieselben Bezeichnungen wie die eingebauten aufweisen ("Source"-Verzeichnis wird *vor* dem .Environment-Verzeichnis aufgesucht!)
  - **.nil**, **.true** und **.false** ***nicht*** umdefinieren!
- Weitere Informationen, z.B.

  `http://wi.wu-wien.ac.at/rgf/rexx/orx07/Local.pdf`

Object Rexx-Umgebung
Beispielsausgabe von .LOCAL

```
/* Ausgabe der Einträge des lokalen Verzeichnisses */
 tmpArr = sort(.local)
 DO entry OVER tmpArr
    SAY LEFT(entry, 25, '.') .local~at(entry)~string
 END
 ::REQUIRES sort_util.cmd /* aus "ORX8.ZIP", zum Sortieren */
```

Ausgabe:

```
ERROR..................... a Monitor
INPUT..................... a Monitor
LOCALSERVER............... a server
NLS.DEFAULT.ALIAS........ iso8859-1
OUTPUT.................... a Monitor
STDERR.................... STDERR
STDIN..................... STDIN
STDOUT.................... STDOUT
STDQUE.................... SESSION
```

Sortiermodul verfügbar unter: <http://wi.wu-wien.ac.at/rgf/rexx/orx08/>

Das "Große Bild"
Initialisierung von Object Rexx

- Interpreter wird gestartet
- Die Object Rexx Umgebungen **.environment** und **.local** werden eingerichtet
- Rexx-Programm, das als Argument angegeben wurde
  - Wird auf Syntaxfehler hin überprüft
  - Direktiven werden befolgt, "Quell-Umgebung" **source** eingerichtet
    - ::REQUIRES lädt das angegebene Programm, führt Syntaxüberprüfung durch und befolgt die Direktiven (entsprechende "Quell-Umgebung" wird eingerichtet) die Ausführung der Programmanweisungen beginnt ab Zeile eins
    - Klassenobjekte für die Object Rexx-Klassen werden gebildet und im Quellverzeichnis gespeichert
  - Ausführung des Programmes beginnt mit den Anweisungen ab Zeile eins

UNKNOWN

- Für den Fall, daß eine Methode nicht gefunden wird
  - UNKNOWN-Methode wird aufgerufen, sofern vorhanden
    - Argumente
      - Name der nicht gefundenen Methode
      - Array-Objekt mit den bereitgestellten Argumenten

```
/* Beispiel für die Definition einer UNKNOWN-Methode */
::METHOD UNKNOWN
    USE ARG meth_name, meth_args
    SAY "unknown method: ["meth_name"]“
    DO i=1 TO meth_args~items
       SAY "  arg #" i": ["i"] value: ["meth_args[i]"]“
    END
```

  - Ansonsten wird die NOMETHOD-Ausnahme aufgeworfen
    - Wenn keine Ausnahmebehandlung mit der SIGNAL ON Anweisung definiert ist, wird vom Laufzeitsystem ein Syntaxfehler erzeugt, der zum Programmabbruch führt

FORWARD-Anweisung

- "Nachrichten-Umleitung"
  - Änderung des Zielobjekts (TO)
  - Weitergabe der Nachricht an Superklassen (CLASS)
  - Änderung der Nachrichtenbezeichnung (MESSAGE)
  - Leitet erhaltene Argumente weiter, außer
    - ARGUMENT oder
    - ARRAY ist angegeben
  - Kehrt nach der Nachrichtenweiterleitung an die Stelle zurück, von der aus ursprünglich die Nachricht abgesendet wurde, außer
    - CONTINUE ist angegeben