Java Platform Module System Cheat Sheet

module-info.java file contents

`module module.name` - declares `module.name`

`requires module.name` - this module depends on module `module.name`

`requires transitive module.name` - any modules that depend on this module automatically depend on `module.name`.

`exports pkg.name` - this module exports public members in package `pkg.name`

`exports pkg.name to module.name` - this module allows the target module to access public members in package `pkg.name`

`uses class.name` - this module declares itself as a consumer for service `class.name`

`provides class.name with class.name.impl` - provides an implementation of a service for others to consume

`opens pkg.name` - allows reflective access to the private members of package `pkg.name`

`opens pkg.name to module.name` - opens private members of package `pkg.name` to the given module

Manifest attributes

`Automatic-Module-Name: module.name` - declares stable module name for non-modularized jar
`Add-Exports: <module>/<package>` - exports the package to all unnamed modules
`Add-Opens: <module>/<package>` - opens the package to all unnamed modules

Java command line options

`--module-path or (-p)` is the module path; its value is one or more directories that contain modules.

`--add-reads src.module=target.module` - a command-line form of a `requires` clause in a module declaration.

`--add-exports src.module/pkg.name=target.module` - a command line form of an exports clause.

`--add-opens src.module/pkg.name=target.module` - a command line form of the open clause in a module description.

`--add-modules` - adds the indicated modules to the default set of root modules.

`--list-modules` - displays the names and version strings of the observable modules.

`--patch-module` - adds or overrides classes in a module. Replaces `-Xbootclasspath/p`.

`--illegal-access=permit|warn|deny` - relaxes strong encapsulation of the module system; Java 9 default is `permit`.

Mechanism	Compile Access	Reflection Access
Export	all code → public	all code → public
Qualified Export	specified modules → public	specified modules → public
Open Package	none ⊘	all code → private
Qualified Open Package	none ⊘	specified modules → private
Open Module	none ⊘	all code → private
Default	none ⊘	none ⊘

Module types

`Java SE and JDK modules` - modules provided by JDK: `java.base`, `java.xml`, etc.

`Named application module` - your application modules; contains `module-info.class`; explicitly exports packages; can't read the unnamed module.

`Automatic module` - non-modular jar on the module-path; exports all packages; name derived from the **`Automatic-Module-Name`** `MANIFEST.MF` entry or the filename; can read all modules.

`Unnamed module` - all jars/classes on the classpath; can read all modules.