NetRexx 4

Marc Remes

remesm@gmail.com

Agenda

- Java 9, its JPMS and NetRexx
- Eclipse Java Compiler
- ADDRESS instruction

JPMS

- Java Platform Module System
  - JSR 376
  - https://openjdk.java.net/projects/jigsaw/spec/
- Modules
  - Container of Java packages
  - Dependencies
  - Exporting
  - More reliable as CLASSPATH, more scalable, better integrity and performance
- Just a jar file

JPMS

- The ‘unnamed’ module
- Module CLI: jdeps, jlink, java –list-modules –describe-modules –add-modules

- Oops, where’s tools.jar

NetRexx on Java 9

- NetRexx 3.x found the java compiler and JVM on CLASSPATH
- Java 9 has moved these classes to modules
- Class loading from modules required

SourceForge

- git clone ssh://git.code.sf.net/p/netrexx/code
- NetRexx is written in NetRexx
- and open source
- -diag and -verbose5 are your friends
- copy/paste also;
- KK’s technical diagrams on the translator logic

Limited updates

- build.xml
  - Ant source and target is 1.7 (vs 1.6)
- org/netrexx/process/RxClassPool.nrx
  - Just a doc update (file=Object)
- org/netrexx/process/RxInterpreter.nrx
  - Just catching JDK12+ inaccessible class modifiers exception
- org/netrexx/process/RxClasser.nrx
  - Bulk of changes

RxClasser.nrx

```
/*   This is the generalized class processor.  The classer object     */
/*   handles everything to do with classes: finding them, testing     */
/*   for them, and so on.                                             */
```

- Constructor
  - check if JPMS is present
    - Object.class.getResource('Object.class')
    - If starts with ‘jrt:’, flag isJrt (‘jrt:/java.base/java/lang/Object.class’)
    - Walks the whole jrt:/ to register packages to modules (method packmodfind)
- Method ImportClasses
  - Run all existing code
  - If isJrt, additionally find packages/class(es) in module(s) (method modfind)

RxClasser.nrx

- Method modfind
  - Register the Path into jrt:/ URI for found class on ClassPool’s property

RxClasser.nrx

- Method loadclass
  - If RxClassInfo’s fileitem is a Path
  - Open InputStream using Path from module
  - Run existing code

RxJrt.nrx

- In examples/new-4.01 directory

```
$ java RxJrt -h
RxJrt : Walks the JPMS jrt:/ file system
 Optional arguments
  [-a | -all]     show all
  [-m | -module]  show module
  [-p | -package] show package (actually a directory..)
```

NetRexx as Module

- As long as Java will support ‘package’ jar files, no need for change
- When/If such support is dropped, we need to package NetRexx as module
- Deliver as Package and as Module?

Eclipse Java Compiler

- Needed to update ecj but https://bugs.eclipse.org/bugs/show_bug.cgi?id=537197 where the compiler fails when .java file is not found.
- Small update on EclipseCompilerImpl.java:

```
[ecj]$ diff org/eclipse/jdt/internal/compiler/tool/EclipseCompilerImpl.java.orig org/eclipse/jdt/internal/compiler/tool/EclipseCompilerImpl.java
146,148c146,149
<     File file = new File(name);
<     if (!file.exists())
<     throw new IllegalArgumentException(this.bind("unit.missing", name)); //$NON-NLS-1$
---
> // MRE NetRexx compiles from memory, don't abort when file does not exist
> //  File file = new File(name);
> //  if (!file.exists())
> //  throw new IllegalArgumentException(this.bind("unit.missing", name)); //$NON-NLS-1$
```

- Tag I20201218-1800

ADDRESS instruction

- Invaluable strength of Rexx interacting with underlying OS’ shell, and other CLI
- Never implemented in NetRexx
- While we’re at it..
- To be identical to Classic Rexx

New option

- -noaddress option to NetRexxC provides old behavior

Source files

- netrexx/lang/RexxAddress.nrx
  - Exec invocation
- org/netrexx/process/RxType.nrx
  - Defines the REXXADDRESS_CLASS type
- org/netrexx/process/NrAddress.nrx
  - ADDRESS instruction parser logic
- org/netrexx/process/NetRexxC.nrx
  - Add -address flag to allow/disallow new instruction
- org/netrexx/process/RxTranslator.nrx
  - Store -address flag
- org/netrexx/process/NetRexxC.properties
  - Add new error messages
- org/netrexx/process/RxClass.nrx
  - Add property for ‘class’-based ADDRESS
- org/netrexx/process/RxParser.nrx
  - Allow ADDRESS invocation, either direct or indirect
- org/netrexx/process/RxExprParser.nrx
  - Add RC special variable

RexxAddress.nrx

- Any CLI driven executable can be addressed
- Addressee is started by ProcessBuilder
- The command to be executed is written into stdin of addressee
- Output is read from stdout, and ‘said’ on console

hello.nrx

```
‘echo “Hello world”’
```

- SYSTEM defaults to COMSPEC, getenv(‘SHELL’) or /bin/sh

DiagAddress.nrx

```
class DiagAddress.ByClass
   address 'cat'

 method ByClass
   'Hello world  -- indirect addressing cat by class'
   address 'bash' 'echo "Hello world  -- direct addressing ''bash''"'
   'Hello world  -- still indirect addressing cat by class'
   var = Rexx 'bash'
   address var
   'echo "Hello world  -- overruled indirect addressing literal by var '||var||'"'
   hello='echo Hello world  -- overruled indirect addressing a var by 'var
   hello
   say 'DiagAddress:' 'OK ! ADDRESS by class'
```

DiagAddressRC.nrx

```
package org.netrexx.diag
options binary strictargs trace1

class DiagAddressRC

method DiagAddressRC
    noRC=RC
    if noRC\=='RC' then signal DiagX('RC not RC')

    'echo "Hello world"'
    if RC\=0 then signal DiagX('RC not 0')

    exitcode=7
    'exit '||exitcode
    retcode=RC
    if retcode\=7 then signal DiagX('RC not 7')
    say 'DiagAddressRC:' 'OK ! RC tests completed'
```

Next?

- Trace ?i
  - Interactive trace..
- IDE integration
  - Eclipse
  - VS Code