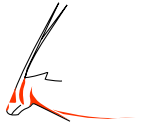
Department of Information Systems
and Operations Management

JDOR - Java2D for ooRexx (and Other Programming Languages)

From the Bean Scripting Framework for ooRexx

Multiplatform Rexx Commands for Powerful 2D graphics

The 2024 International Rexx Symposium
Brisbane, Queensland, Australia
March 3rd – March 6th 2024



Overview

- ooRexx instructions
  - Command instruction
- ooRexx-Java bridge BSF4ooRexx850
  - Java2D
  - JDOR (Java2D for ooRexx)
- Roundup

ooRexx – Instruction Types

- ooRexx is an extension and compatible to Rexx and supports therefore all Rexx instruction types
  - *Assignment instruction* (second token is an equal sign)
  - *Keyword instruction* (first token is a Rexx keyword like DO or CALL)
  - *Command instruction*: any other string
    - Will be sent by default to the operating system for execution
    - However, any other existing Rexx command handler can be addressed using the ADDRESS keyword instruction
- ooRexx adds a fourth instruction type, the *directive instruction*
  - At the end of a program, always led in with two colons (::)
  - Directive instruction get executed in the setup phase, right before running the program with the first statement from the top of the program

ooRexx – Instruction Types (Example)

```
a="Hello world"          /* assignment  */
do i=1 to 3                 /* keyword  */
   say "... #" i":" a       /* keyword  */
end                         /* keyword  */
cmd="echo" a "..."       /* assignment  */
say "cmd:" pp(cmd)          /* keyword  */
cmd      /* command (has return code)  */
say "return code:" pp(RC)  /* keyword  */

::routine pp             /* directive   */
  return "["arg(1)"]"       /* keyword  */
```

Output:

```
... # 1: Hello world
... # 2: Hello world
... # 3: Hello world
cmd: [echo Hello world ...]
Hello world ...
return code: [0]
```

ooRexx – Redirecting Standard Files

- Starting with ooRexx 5.0 redirections of the standard files are supported
  - stdin (0, by default the keyboard), to change input to a file use '< filename'
  - stdout (1, by default the screen), to change to a file '> filename'
  - stderr (2) by default the screen), to change to a file '**2**> filename'
- These standard files get usually created for each process by the operating system
- `PARSE PULL var` … reads text from stdin and assigns it to the variable *var*
- `SAY someText` … writes *someText* to stdout
- `CALL LINEOUT 'stderr:',someText` … writes *someText* to stderr

  ooRexx in addition:
- `.error~say(someText)` w rites *someText* to stderr

Prof. Rony G. Flatscher

ooRexx – Redirecting Standard Files (Example)
Redirecting to Operating System's “sort” command !

```
"echo hello world 1"          -- by default addresses system
say "--> echo's return code:" rc
address system "echo hello world 2"   -- explicit addresss
say "--> echo's return code:" rc
uArr=.array~of("John", "Hans", "Alicia", "Xaver", "Josep")    -- unsorted array
sArr=.array~new  -- array to receive the sorted names
ADDRESS system "sort" WITH INPUT USING (uArr) OUTPUT USING (sArr)
say "--> sort's return code:" rc
say "unsorted names:" uArr~makeString("L",", ")   -- turn into comma delimited string
say "sorted names:  " sArr~makeString("L",", ")   -- turn into comma delimited string
```

Output:

```
hello world 1
--> echo's return code: 0
hello world 2
--> echo's return code: 0
--> sort's return code: 0
unsorted names: John, Hans, Alicia, Xaver, Josep
sorted names:   Alicia, Hans, John, Josep, Xaver
```

BSF4ooRexx850, 1

- Bidirectional, comprehensive ooRexx-Java bridge
  - An external ooRexx function and class package
  - Takes advantage of the ooRexx message paradigm to simplify interaction with Java
    - Use the requires directive for BSF.CLS, an ooRexx package (program)
  - To use Java it is sufficient to study the HTML Java documentation from the Internet, no need to know how to code in Java!
  - Being able to use Java means among other things
    - All Java functionality is immediately available to the Rexx programmer
    - All such programs run unchanged on all platforms where Java and ooRexx are available!

BSF4ooRexx850, 2

- BSF4ooRexx850 introduced the ability to implement Rexx command handlers in Java!
  - An initial proof-of-concept demonstrated this ability by creating a few Rexx commands for Java2D functionality in a Java Rexx command handler named “JDOR” (**J**ava2**D** for **oo****R**exx)
  - In the past months all of the powerful Java2D functionality got implemented and is now available with the JDOR Rexx command language
    - No need to know how to program in Java
    - Rexx commands are plain strings that get forwarded to the Rexx command handler for processing

BSF4ooRexx850, An Example, 1

- The ooRexx program will exploit a Java class named java.awt.Dimension
  - A Java class from the abstract window toolkit (awt)
  - Maintains the fields width and height
  - A Rexx program merely requires BSF.CLS which camouflages Java as ooRexx
    - Java objects understand ooRexx messages! :)
    - For an ooRexx programmer it does not matter whether the object to which he sends messages is an ooRexx object, a Java object or even a Windows OLE object for that matter!
    - Marshalling of the arguments and return values is done automatically

BSF4ooRexx850, An Example, 2

```
   -- create a java.awt.Dimension Java object
d=.bsf~new("java.awt.Dimension",111,222)
say "d:          " d  -- displays object's name
say "d~toString:" d~toString  -- send message to Java
say                -- new line
say "... d~setSize(333,444):"
d~setSize(333,444)             -- send message to Java
say "d~toString:" d~toString  -- send message to Java
say                -- new line
say "... Java fields as if they were Rexx attributes:"
d~width =555       -- camouflaged as Rexx attribute
d~height=666       -- camouflaged as Rexx attribute
say "d~toString:" d~toString  -- send message to Java
say "            d~width:" d~width "d~height:" d~height

::requires "BSF.CLS"    -- get ooRexx-Java bridge
```

Output:

```
d:          java.awt.Dimension@6b27808d
d~toString: java.awt.Dimension[width=111,height=222]

... d~setSize(333,444):
d~toString: java.awt.Dimension[width=333,height=444]

... Java fields as if they were Rexx attributes:
d~toString: java.awt.Dimension[width=555,height=666]
            d~width: 555 d~height: 666
```

JDOR Rexx Command Handler, 1

- The JDOR Rexx command handler is part of BSF4ooRexx850
  - Many samples in `BSF4ooRexx850/samples`
  - JDOR Rexx command language documentation in `BSF4ooRexx850/documentation/jdor/jdor_doc.html`
- The JDOR Rexx command language covers all Java2D functionality
- Using Rexx JDOR commands instead of the Java2D classes makes creating and manipulating (Java) 2D graphics considerably easier
  - No need to know how to program Java
  - Need to understand 2D graphics concepts which allows one to understand the Java HTML documentation and the Rexx JDOR command language
  - There are JDOR commands for displaying and printing graphics, to copy graphics to the clipboard or paste graphics from the clipboard for Java2D

JDOR Rexx Command Handler, 2

- The JDOR Rexx command handler supports redirection of commands
  - Allows one to create macros, i.e. text files containing all the JDOR commands used for creating the Java2D graphics by simply redirecting stdout to a text file
  - Allows one to redirect stdin (input) from text files that contain JDOR Rexx commands line by line
  - Allows one to get extensive error information if redirecting stderr to a text file

JDOR Rexx Command Handler, Example 1, 1

- The JDOR example program
  - Loads and sets the JDOR command handler as the default handler
    - stdout gets redirected, such that all JDOR commands will get sent (in a canonized form) to .output
  - Creates an image of 200 by 200 pixels and shows it in a window
  - After loading the RexxLA logo its dimension gets shown
  - The image gets downscaled (1/7) and drawn at x=16 and y=25
  - Then the BSF4ooRexx logo gets loaded, its dimension shown
  - The image gets downscaled (1/7) and drawn at x=75 and y=145
  - Finally the image gets saved as 20_images.png

JDOR Rexx Command Handler, Example 1, 2

```
jdh=.bsf~new("org.oorexx.handlers.jdor.JavaDrawingHandler")
call BsfCommandHandler "add", "JDOR", jdh -- add handler

address JDOR with output using (.output)  -- set JDOR handler

newImage 200 200          -- create image width=200, height=200
winShow                   -- show window
fn1='rexxla.png'          -- filename
loadImage img1 fn1        -- load image, RC gets its dimension
say "--" fn1": RC=["RC"]"   -- show file name and RC
parse var RC cbsW cbsH    -- parse RC to get width and height
moveTo 16 25              -- set location to x=16 y=25
drawImage img1 (cbsW/7) (cbsH/7)   -- resize image to 1/7
fn2='bsf4oorexx_256.png'    -- filename (used also as nickname)
loadImage img2 fn2        -- load image, RC gets its dimension
say "--" fn2": RC=["RC"]"   -- show file name, RC
parse var RC ptW ptH      -- parse RC to get width and height
moveTo 75 145             -- set location to x=75 y=145
drawImage img2 (ptW/5) (ptH/5)  -- resize image to 1/5
saveImage "20_images.png"   -- save image to file
sleep 3                   -- sleep a bit
parse pull .              -- user needs to press <enter>

::requires "BSF.CLS"      -- get ooRexx-Java bridge
```

Output (canonized JDOR Rexx commands):

```
newImage 200 200
winShow
loadImage IMG1 rexxla.png
-- rexxla.png: RC=[1200 927]
moveTo 16 25
drawImage IMG1 171 132
loadImage IMG2 bsf4oorexx_256.png
-- bsf4oorexx_256.png: RC=[256 256]
moveTo 75 145
drawImage IMG2 51 51
saveImage 20_images.png
sleep 3.0
```

JDOR Rexx Command Handler, Example 2, 1

- A Java2D tutorial for programming games with Java demonstrates 2D affinity transform with a Java program that can be fetched from

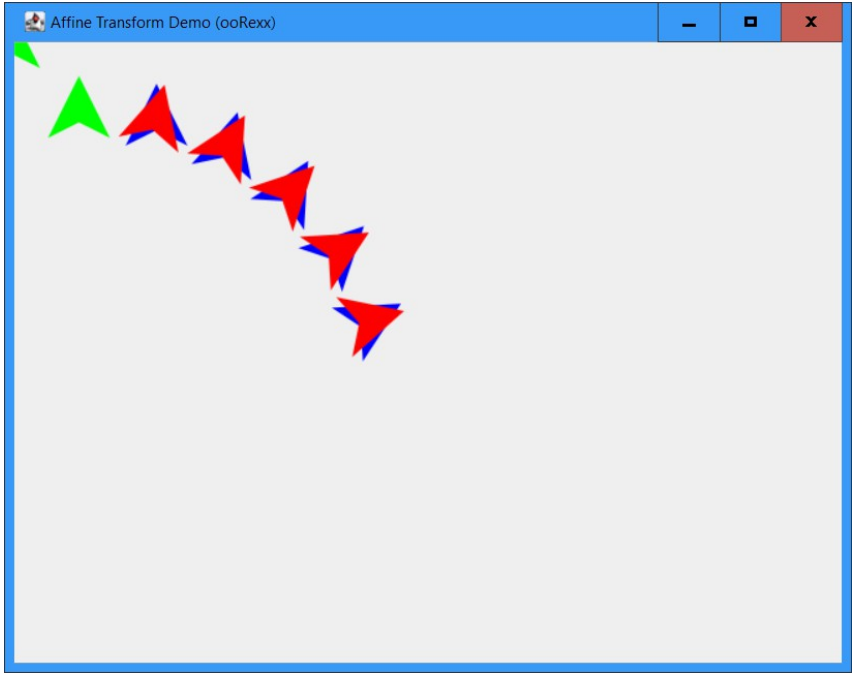
  <https://www3.ntu.edu.sg/home/ehchua/programming/java/J8b_Game_2DGraphics.html#zz-2.2>

- The JDOR Rexx command example is smaller and easier to understand by comparison

JDOR Rexx Command Handler, Example 2, 2

```
jdjdh=.bsf~new("org.oorexx.handlers.jdor.JavaDrawingHandler")
call BsfCommandHandler "add", "JDOR", jdh -- add  handler
address JDOR       -- set JDOR as default environment
newImage 640 480   -- JDOR command to create new image
winShow            -- show image in a window
winTitle "Affine Transform Demo (ooRexx)" -- set title
polygonXs="(-20,0,+20,0)"  -- define four x coordinates
polygonYs="(20,10,20,-20)" -- define four y coordinates
shape myP polygon polygonXs polygonYs 4   -- create polygon
color green        -- set color to green
fillShape myP      -- fill (and show) the polygon shape
translate 50 50    -- move origin (x=x+50, y=y+50)
scale 1.2 1.2      -- increase scale symmetrically by 20%
fillShape myP      -- fill (and show) the polygon shape
do 5               -- repeat five times
   translate 50 5  -- move origin (x=x+50, y=y+5)
   color blue      -- set color to blue
   fillShape myP   -- fill (and show) the polygone shape
   rotate 15       -- rotate by 15°
   color red       -- set color to red
   fillShape myP   -- fill (and show) the polygone shape
end
sleep 3            -- sleep a bit
parse pull .
::requires "BSF.CLS"    -- get ooRexx-Java bridge
```

Output:



Package: java.awt

JDOR Rexx Command Handler Samples

- Numerous samples including animations in
  - `BSF4ooRexx850/samples`
  - File name contains the string "`JDOR`"
- To get a brief explanation for each sample load
  - `BSF4ooRexx850/samples/index.html`
- A short (five minutes) video demonstrating JDOR can be seen at
  - `https://zenodo.org/record/8003114`

Roundup

- BSF4ooRexx850
  - Allows implementing Rexx command handlers in Java (in addition to C++)
  - Includes a fully developed Rexx command language named JDOR (Java2D for ooRexx)
    - All Java2D features are available as much simpler Rexx commands
    - Using redirection one can create macros (sequence of all issued JDOR commands)
      - Can be used to replay all commands to recreate the image
      - Other programming languages can create Rexx JDOR commands by writing them to stdout and piped into a JDOR Rexx command handler
    - All JDOR commands are documented in `BSF4ooRexx850/documentation/jdor/jdor_doc.html`
    - One can use all Javadocs for Java2D to get more information, e.g. searching for "`javadoc java.awt.Shape`" or "`javadoc affineTransform`"
- Powerful and a lot of fun!