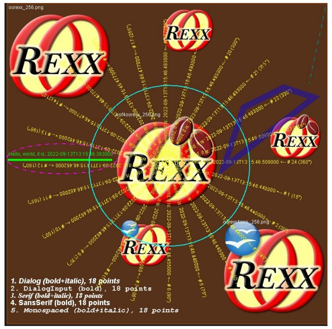
Department of Information Systems
and Operations Management

BSF4ooRexx: Introducing the JDOR Rexx Command Handler for Easy Creation of Bitmaps and Bitmap Manipulations on Windows, Mac and Linux

Make Java's 2D graphics methods available as Rexx commands

International RexxLA Symposium, 2022-09 (https://www.RexxLA.org)

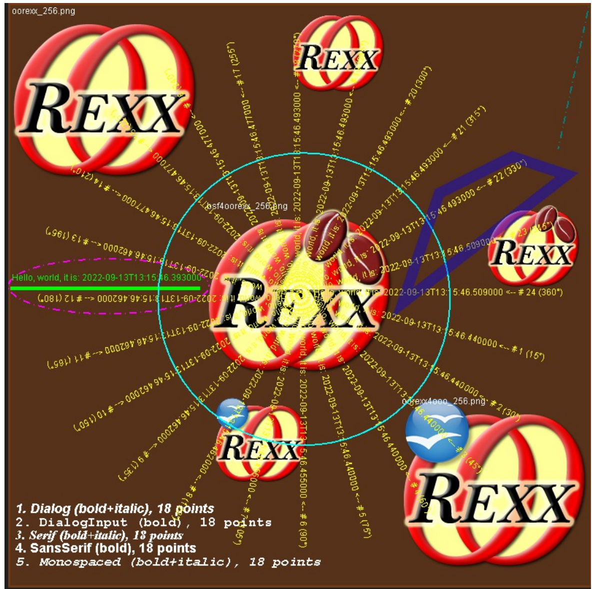

1. Dialog (bold+italic), 18 points
2. DialogInput (bold), 18 points
3. Serif (bold+italic), 18 points
4. SansSerif (bold), 18 points
5. Monospaced (bold+italic), 18 points

Overview

- Java awt 2D graphics
- REXX command handler
- JDOR command handler

Java awt Graphics2D

- Available for all operating systems
- Allows creating and manipulating images/bitmaps
  - E.g. drawing strings, lines, rectangles, ovals,, images, ...
- Quite comprehensive
- Tutorials, e.g. Oracle‘s
  - https://docs.oracle.com/javase/tutorial/2d/TOC.html
- BSF4ooRexx allows one to use ooRexx to exploit it
  - e.g. samples/3-100_create_bitmap.rxj


oorexx_256.png
bsf4oorexx_256.png
Hello, world, it is: 2022-09-13T13:15:46.393000
oorexx4ooo_256.png
1. Dialog (bold+italic), 18 points
2. DialogInput (bold), 18 points
3. Serif (bold+italic), 18 points
4. SansSerif (bold), 18 points
5. Monospaced (bold+italic), 18 points

Commands in REXX, 1

- REXX commands easy to be deployed from REXX programs
- ADDRESS environment command
  - ooRexx defined environment names: SYSTEM ("", COMMAND), PATH, CMD, bash, sh, ...
- Possible to define a default environment with „ADDRESS environment“
  - Default is set to the current operating system‘s shell, hence easy to send operating system commands from Rexx programs
- Exploited early on, e.g. for allowing commands from Rexx to be addressed to editors, specialized software systems
- Quite important and popular on mainframes

Commands in REXX, 2

- Can be very handy for other operating systems
- Easying interfacing with any software system by supplying appropriate Rexx command handlers
- Command handler‘s results (can be any value) are saved in the variable RC (return code) by the Interpreter
- Command handlers may raise an *error condition* or a *failure condition*
  - Neither condition will interrupt the execution of a Rexx program
  - An error condition may be raised because of an error in a command (.RS set to 1)
  - A failure condition is regarded to be serious and may be raised because some important basic feature cannot be used or failed when attempting to use it (.RS set to -1)

Commands in REXX, 3

- The Rexx interpreter will trace a failure condition, such that on the screen one can see the command that caused a failure in the command handler
- Debugging Rexx programs with commands
  - Use "signal on failure" and "signal on error" to intercept these conditions
  - Commands do not have to be quoted, unless they clash with Rexx keywords
  - If SIGNAL ON NOVALUE or SIGNAL ON ANY is set, then one needs to quote commands
  - To debug the evaluation of commands before passing them to the environment use "TRACE I" or "TRACE R"

JDOR Command Handler

- A Rexx command handler to serve commands that exploit Java‘s awt 2D classes for creating and manipulating graphics (bitmaps)
  - JDOR: "Java Drawing for ooRexx"
  - Implemented in Java using BSF4ooRexx850 to allow among other things accessing, creating and dropping Rexx variables in the caller's context
- Enable interfacing with the Java awt graphics 2D subsystem
- Need to devise sensible commands and their arguments
  - Follow the Rexx philosophy to make it as easy and simple as possible for Rexx programmers to take advantage of the Java awt graphics 2D subsystem
- Need to set the Rexx interpreter to load and employ specialized Rexx command handlers

JDOR Command Handler Overview, 1

- A Rexx command handler implemented in Java
- Rexx programmer does not need to use Java
- Intended to make it easy to exploit Java2D
- Allow to optionally display the current state of an image (graphic) on the user interface, including the ability to move, size, hide, close, show the current window with the image in it
- Allow exploiting the Graphics and Graphics2D drawing capabilities of the Java awt package via commands
  - draw strings (text), lines, rectangles, ovals, images, ...
  - set colors, fonts, strokes

JDOR Command Handler Overview, 2

- Allow accessing the current state and JDOR data and the directories/HashMaps of loaded colors, fonts and strokes
  - This makes it possible to define additional colors, fonts, strokes from the Rexx program and save them in the respective HashMaps to be usable for further commands
- Halt execution temporarily to become able to animate drawings
- Make it easy to save and restore graphic configurations and image states at run time
- Make it easy to save and load images
- Allow recording and replaying of commands, effectively creating Rexx macros for Java 2D graphics that can be stored even in plain text file

JDOR Command Handler Overview, 3

- Drawing area is a canvas of width and height pixels (picture elements)
  - Origin (x=0, y=0): left upperhand corner, can be moved with the translate command
  - x increases to the right
  - y increases to the bottom/down
- Graphics and Graphics2D methods include the x and y coordinate
  - Commands separate the x and y coordinate using the command goto x y/pos x y
- Commands invoking the Graphics and Graphics2D draw methods do not include the x and y coordinates but use the previsously set x and y position

JDOR Command Handler Overview, 4

- Search „Javadocs 8 Graphics“ or „Javadocs 8 Graphics2D“ for further documentation on the draw methods that can be invoked by commands
  - As of 2022-09-13

    https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html

    https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics2D.html
  - Tutorial as of 2022-09-13, e.g.

    Oracle: https://docs.oracle.com/javase/tutorial/2d/basic2d/index.html

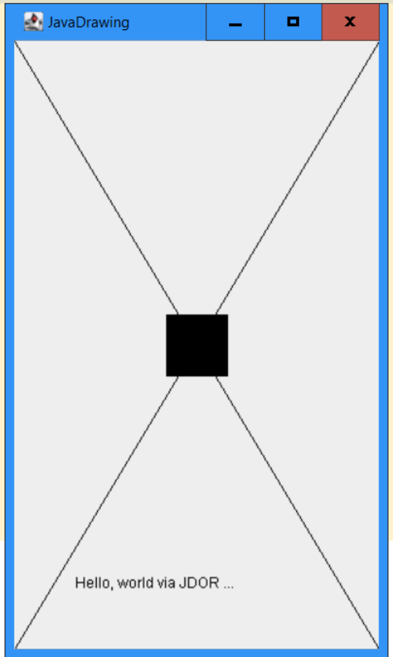
JDOR Command Samples

- Drawing colored lines and strings
- Drawing ovals, rectangles
- Loading and drawing bitmaps
- Musings with translate, scale and rotate
- Get access to current settings and directories/HashMaps

JDOR Sample "sample01.rex"

- Drawing lines, rectangles and strings

```
#!/usr/bin/env rexx
call jdor.cls              -- makes routine addJdorHandler([environmentName]) available
call addJdorHandler        -- load and add the Java Rexx command handler, using default name: JDOR
address jdor               -- set default environment to JDOR

new 300 500                -- create new bitmap
winShow                    -- show frame
   -- draw two lines forming a big X
drawLine 300 500           -- currX=0, currY=0
goto 300 0                 -- currX=300, currY=0
drawLine 0 500
   -- draw a rectangle at the center of the bitmap
goto (300-50)/2 (500-50)/2
fillRect 50 50
drawRect 50 50
   -- draw a test at the lower half of the bitmap
goto 50 450
drawString "Hello, world via JDOR ..."
sleep 5                    -- sleep for five seconds
```



JDOR Utility Rexx Program/Package "jdor.cls"

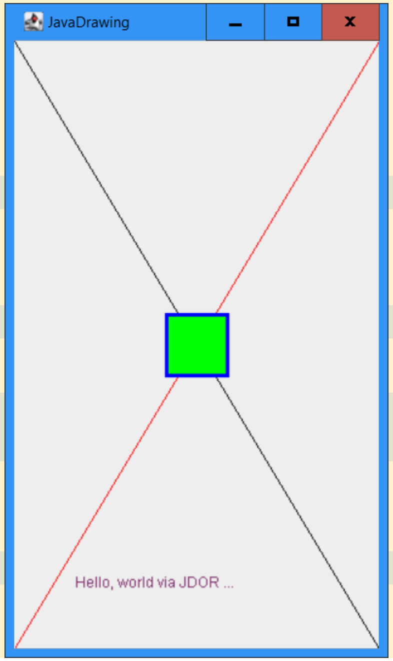
- A simple ooRexx program to ease loading the JDOR command handler

```
#!/usr/bin/env rexx

/** Creates a new command handler that will serve the environment by the name
 *  of the optional environmentName argument.
 *  @param environmentName the address environment name to use, if omitted defaults to "JDOR"
*/
::routine addJdorHandler   public
   use strict arg environmentName="JDOR"

   newHandler=.bsf~new("JavaDrawingHandler")
   call BsfCommandHandler "add", environmentName, newHandler

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

JDOR Sample "sample01a.rex"

- Drawing lines, rectangles, strings; adding colors and strokes

```
#!/usr/bin/env rexx
call jdor.cls            -- makes routine addJdorHandler([environmentName]) available
call addJdorHandler      -- load and add the Java Rexx command handler, using default name: JDOR
address jdor             -- set default environment to JDOR

new 300 500              -- create new bitmap
winShow                  -- show frame
   -- draw two lines forming a big X
drawLine 300 500         -- currX=0, currY=0
goto 300 0               -- currX=300, currY=0
color red
drawLine 0 500
   -- draw a rectangle at the center of the bitmap
goto (300-50)/2 (500-50)/2
color green
fillRect 50 50

stroke str3 3            -- set line stroke to 3 points
color blue
drawRect 50 50
   -- draw a test at the lower half of the bitmap
goto 50 450
color somePurple 127 46 111
drawString "Hello, world via JDOR ..."
sleep 5                  -- sleep for five seconds
```

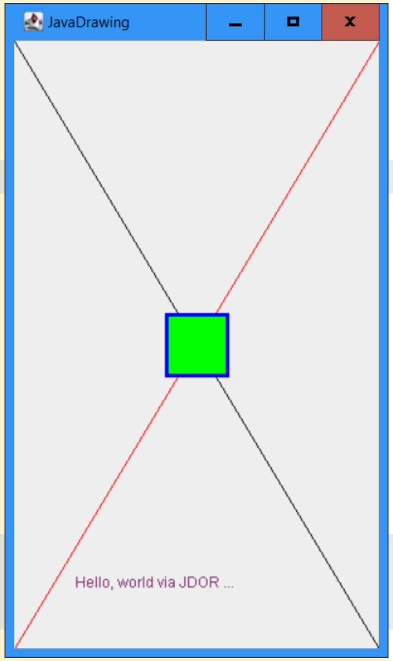


JDOR Sample "sample01b.rex"

- Testing RC (command handler's result) and .RS ...

```
call addJdorHandler     -- load and add the Java Rexx command handler, using default name: JDOR
address jdor            -- set default environment to JDOR
new 300 500             -- create new bitmap
winShow                 -- show frame
   -- draw two lines forming a big X
drawLine 300 500        -- currX=0, currY=0
goto 300 0              -- currX=300, currY=0
say "-->" "rc="pp(rc) ".rs="pp(.rs)
color red
drawLine 0 500
   -- draw a rectangle at the center of the bitmap
goto (300-50)/2 (500-50)/2
color green
fillRect 50 50
stroke str3 3           -- set line stroke to 3 points
color blue
drawRect 50 50
   -- draw a test at the lower half of the bitmap
goto 50 450
color somePurple 127 46 111
say "-->" "rc="pp(rc) ".rs="pp(.rs)
say "    " "rc~toString:" pp(rc~toString)
drawString "Hello, world via JDOR ..."
sleep 5                 -- sleep for five seconds

::requires jdor.cls        -- makes routine addJdorHandler([environmentName]) available
```

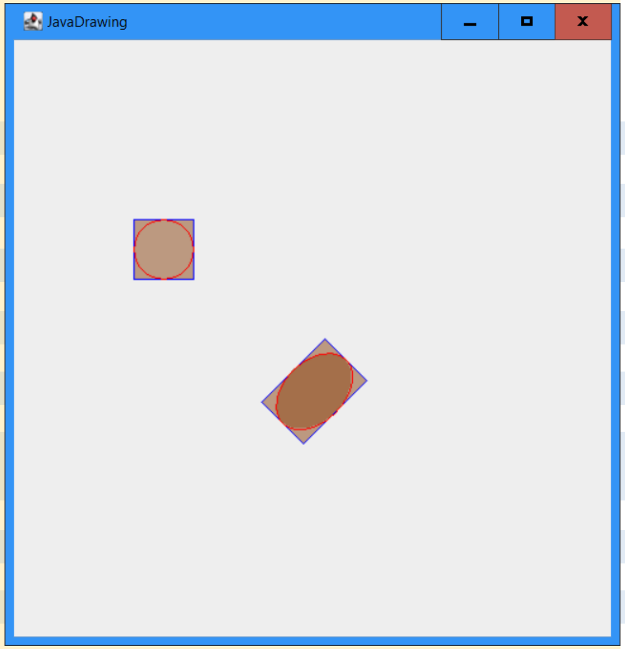
```
--> rc=[pos 300 0] .rs=[0]
--> rc=[java.awt.Color@6b09bb57] .rs=[0]
    rc~toString: [java.awt.Color[r=127,g=46,b=111]]
```



JDOR Sample "sample02.rex"

- Drawing and filling rectangles, ovals; transform and rotate

```
call addJdorHandler      -- load and add the Java Rexx command handler, using default name: JDOR
address jdor             -- set default environment to JDOR
new 500 500              -- create new bitmap
winShow                  -- show frame
   -- draw two lines forming a big X
goto 100  150            -- currX=100, currY=150
color saddlebrown50 139 69 19 127   -- R,G,B,alpha=127 (50 % transparency)
fillRect 50 50           -- first fill
color blue
drawRect 50 50           -- draw border
color red
drawOval 50 50           -- draw border
   -- draw a rectangle at the center of the bitmap
translate 260 250        -- move origin 0,0 to x=0+260, y=0+250
goto 0 0                 -- 0,0 (effectively: x=260, y=250)
rotate 45
color saddlebrown50      -- R,G,B,alpha=127 (50 % transparency)
fillRect 50 75           -- fill 50% transparency
fillOval 50 75           -- fill (add) 50 % transparency
color blue
drawRect 50 75           -- draw border in blue
color red
drawOval 50 75           -- draw border in red
sleep 5                  -- sleep for five seconds

::requires "jdor.cls"    -- makes routine addJdorHandler([environmentName]) available
```

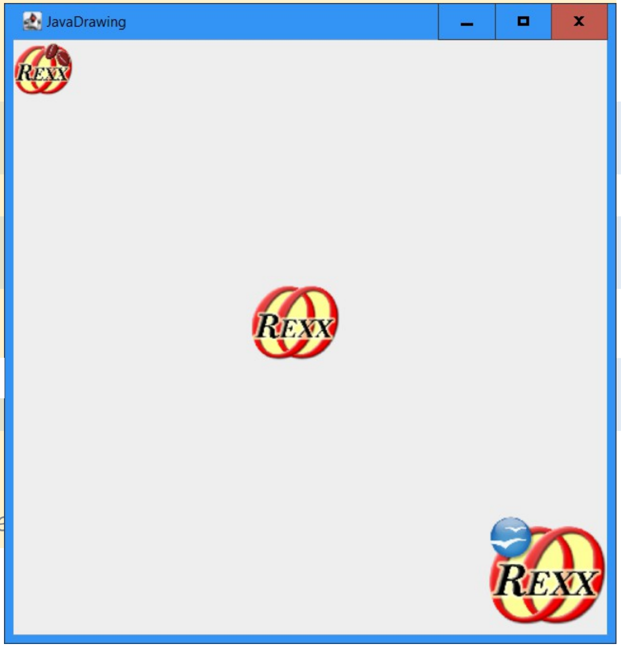


JDOR Sample "sample03.rex"

- Loading and drawing images

```
call addJdorHandler     -- load and add the Java Rexx command handler, using default name: JDOR
address jdor            -- set default environment to JDOR

new 500 500             -- create new bitmap
winShow                 -- show frame
loadImage b4r_logo bsf4oorexx_256.png
drawImage b4r_logo 50 50
goto 200 200
loadImage orx_logo oorexx_256.png
drawImage orx_logo 75 75
goto 400 400
say "rc:" rc            -- rc contains the return value
loadImage ooo_logo oorexx4ooo_256.png
drawImage ooo_logo 100 100
sleep 5                 -- sleep for five seconds

::requires "jdor.cls"   -- makes routine addJdorHandler([environmentName
```

```
rc: pos 400 400
```

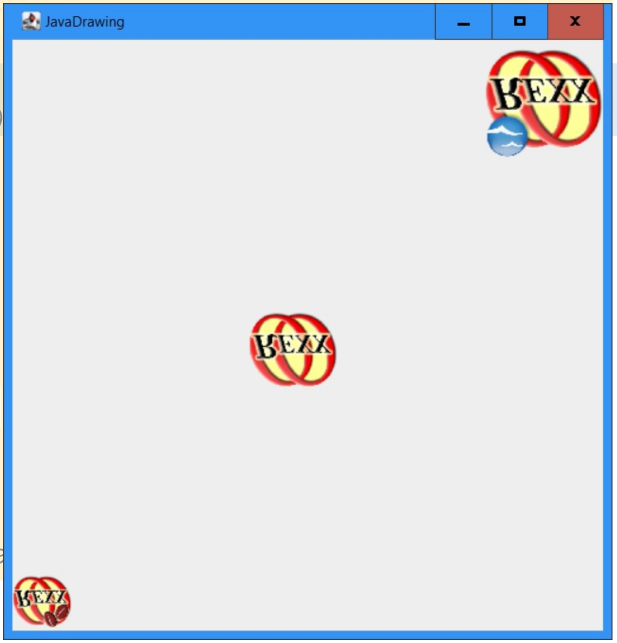


JDOR Sample "sample03b.rex"

- Loading and drawing images, applying translate and scale

```
call addJdorHandler      -- load and add the Java Rexx command handler, using default name: JDOR
address jdor             -- set default environment to JDOR
new 500 500              -- create new bitmap
winShow                  -- show frame
translate 0 500          -- move origin (0,0) to lower left hand corner
"scale     1 -1"         -- flip y-axis (positive y grow up, however ...)
loadImage b4r_logo bsf4oorexx_256.png
drawImage b4r_logo 50 50
goto 200 200
loadImage orx_logo oorexx_256.png
drawImage orx_logo 75 75
goto 400 400
say "rc:" rc             -- rc contains the return value
loadImage ooo_logo oorexx4ooo_256.png
drawImage ooo_logo 100 100
sleep 5                  -- sleep for five seconds

::requires "jdor.cls"    -- makes routine addJdorHandler([environmentName
```

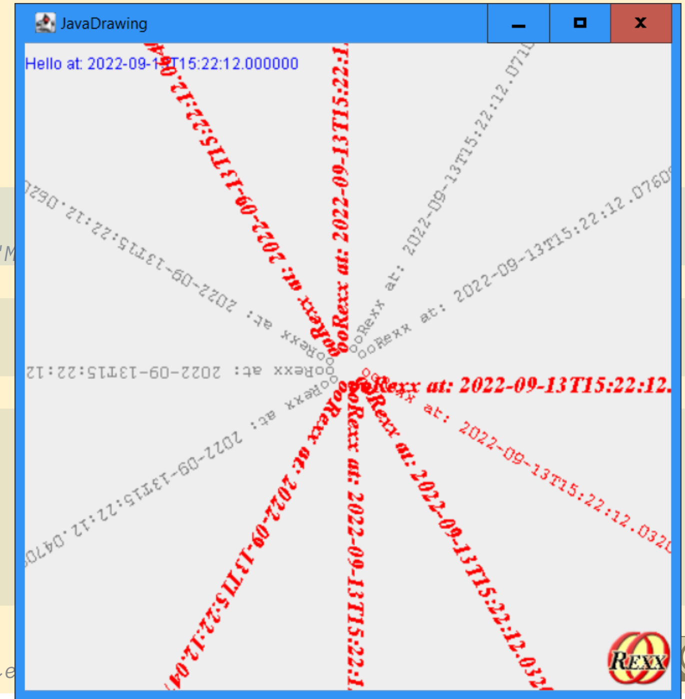
```
rc: pos 400 400
```



JDOR Sample "sample04.rex"

- Rotating while drawing strings with different fonts and colors

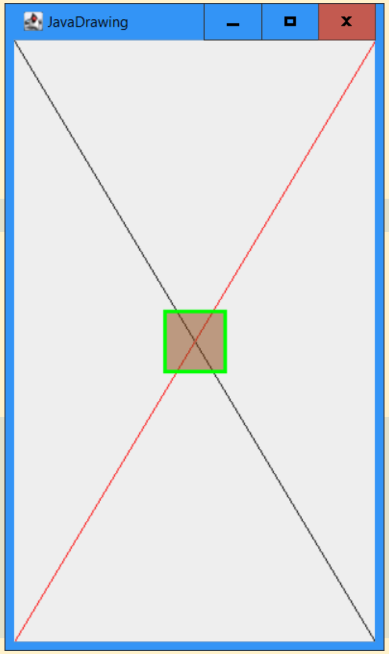
```
call addJdorHandler     -- load and add the Java Rexx command handler, using default name: JDOR
address jdor            -- set default environment to JDOR
winShow                 -- will implicitly create a 500x500 bitmap
goto 450 450
loadImage orx oorexx_256.png
drawImage orx 50 50
color blue
goto 0 20
drawString "Hello at:" .dateTime~new
goto
translate 250 250       -- move 0,0 to center
color red
fontstyle 3             -- bold=1 + italic=2
fontsize 18             -- size=18
font serif1 "Serif"     -- can be: "Dialog", "DialogInput", "Serif", "SansSerif", "M
drawString "ooRexx at:" .dateTime~new
fontstyle 0             -- normal
fontsize 14             -- size=14
font mono "Monospaced"
goto 10 0               -- currX=10, currY=0
angle=30
do i=2 to 360/angle
   rotate angle         -- rotates entire image
   drawString "ooRexx at:" .dateTime~new  -- now draw
   if random(0,1) then do;  color red;    font serif1; end
                  else do; "color gray"; "font mono" ; end
end
sleep 5                 -- sleep for five seconds

::requires "jdor.cls"   -- makes routine addJdorHandler([environmentName]) available
```


JavaDrawing
Hello at: 2022-09-13T15:22:12.000000
ooRexx at: 2022-09-13T15:22:12.

JDOR Sample "sample05.rex"

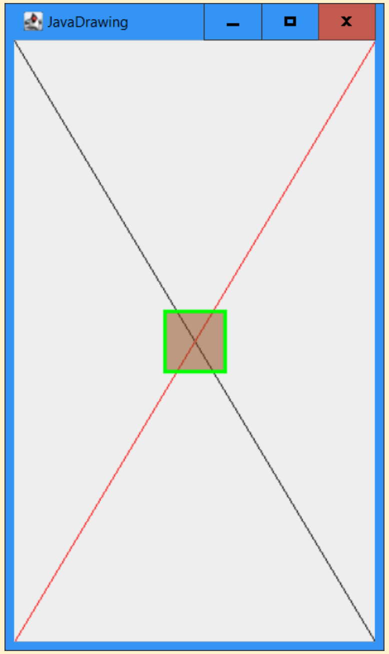
- Fetching current state of the JDOR command handler

```
call addJdorHandler     -- load and add the Java Rexx command handler, using default name: JDOR
address jdor            -- set default environment to JDOR
new 300 500             -- create new bitmap
winShow                 -- show frame
   -- draw two lines forming a big X
drawLine 300 500        -- from currX=0, currY=0 to x2=300 y2=500
pos  300 0              -- alias "goto": currX=300, currY=0
color red
drawLine 0 500          -- from currX=300, currY=0 to x2=0 y2=500
   -- draw a rectangle at the center of the bitmap
goto (300-50)/2 (500-50)/2 -- currX=
say "rc:" pp(rc) ".rs="pp(.rs)
color saddlebrown50 139 69 19 127   -- R,G,B,alpha=127 (50 % transparency)
fillRect 50 50
stroke str3 3           -- set line stroke to 3 points
color green
drawRect 50 50          -- draw frame
-- get current state
say "a) var(""currState""):" pp(var("currState"))  -- does not exist, returns: 0
"getState currState"    -- quote, result should be saved to variable 'currState'
say "b) var(""currState""):" pp(var("currState"))  -- now exists, returns: 1
say "currState:" pp(currState) "currState~items:" pp(currState~items)
do counter c idx over currState~allIndexes~sort
   say "  " right(c,2)":" left(pp(idx),20,".") pp(currState[idx])
end
sleep 5                 -- sleep for five seconds

::requires "jdor.cls"   -- makes routine addJdorHandler([environmentName]) available
```



JDOR Sample "sample05.rex", 2 Output

```
rc: [pos 125 225] .rs=[0]
a) var("currState"): [0]
b) var("currState"): [1]
currState: [a StringTable] currState~items: [26]
     1: [COLORS]............ [java.util.HashMap@4b4523f8]
     2: [CURRANGLE]......... [0.0]
     3: [CURRFONTSIZE]...... [12]
     4: [CURRFONTSTYLE]..... [0]
     5: [CURRIMAGEBUFFER]... [java.awt.image.BufferedImage@eec5a4a]
     6: [CURRIMAGEGC]....... [sun.java2d.SunGraphics2D@2b2948e2]
     7: [CURRIMAGEHEIGHT]... [500]
     8: [CURRIMAGETYPE]..... [2]
     9: [CURRIMAGEWIDTH].... [300]
    10: [CURRSCALEX]........ [0.0]
    11: [CURRSCALEY]........ [0.0]
    12: [CURRSHEARX]........ [0.0]
    13: [CURRSHEARY]........ [0.0]
    14: [CURRTRANSLATEX].... [0]
    15: [CURRTRANSLATEY].... [0]
    16: [CURRX]............. [125]
    17: [CURRY]............. [225]
    18: [FONTS]............. [java.util.HashMap@731a74c]
    19: [GC.BACKGROUND]..... [java.awt.Color@6ddf90b0]
    20: [GC.COLOR].......... [java.awt.Color@67784306]
    21: [GC.FONT]........... [java.awt.Font@57536d79]
    22: [GC.STROKE]......... [java.awt.BasicStroke@185d8b6]
    23: [GCSTACK]........... [java.util.ArrayDeque@3b0143d3]
    24: [IMAGES]............ [java.util.HashMap@1f28c152]
    25: [IMAGESTACK]........ [java.util.ArrayDeque@5a8e6209]
    26: [STROKES]........... [java.util.HashMap@369f73a2]
```



Defining an ooRexx Object in Java and Setting It to a Rexx Context Variable

```
… cut …
    // create a RexxStringTable, fill it with current settings, stacks, maps and return it;
    // if optional nameCtxtVariable supplied, store it as a context variable in addition
case GET_STATE:     // "getstate [ctxtVariableName]" returns a StringTable with current variables, stacks and HashMaps;
    {
        if (arrCommand.length>2)
        {
            throw new IllegalArgumentException("this command needs ...");
        }
        RexxProxy rop = (RexxProxy) newStringTable(slot);   // create and get a StringTable instance
        rop.sendMessage2("SETENTRY", "currX"      , currX); // send a Rexx message to the StringTable object
        rop.sendMessage2("SETENTRY", "currY"      , currY);
        rop.sendMessage2("SETENTRY", "currAngle"  , currAngle);
        rop.sendMessage2("SETENTRY", "currImageBuffer" , bufImage );
        rop.sendMessage2("SETENTRY", "currImageGC"     , bufGC    );
        rop.sendMessage2("SETENTRY", "currImageWidth"  , bufImage.getWidth());
        rop.sendMessage2("SETENTRY", "currImageHeight" , bufImage.getHeight());
        rop.sendMessage2("SETENTRY", "currImageType"   , bufImage.getType());
… cut …
        if (arrCommand.length==2) // save in a Rexx variable in the caller context (name supplied as argument)
        {
            String contextVariableName = arrCommand[1];
            setContextVariable(slot, contextVariableName, rop); // set context variable
        }
        return rop;                 // return StringTable object (RC variable will refer to it)
    }
… cut …
```

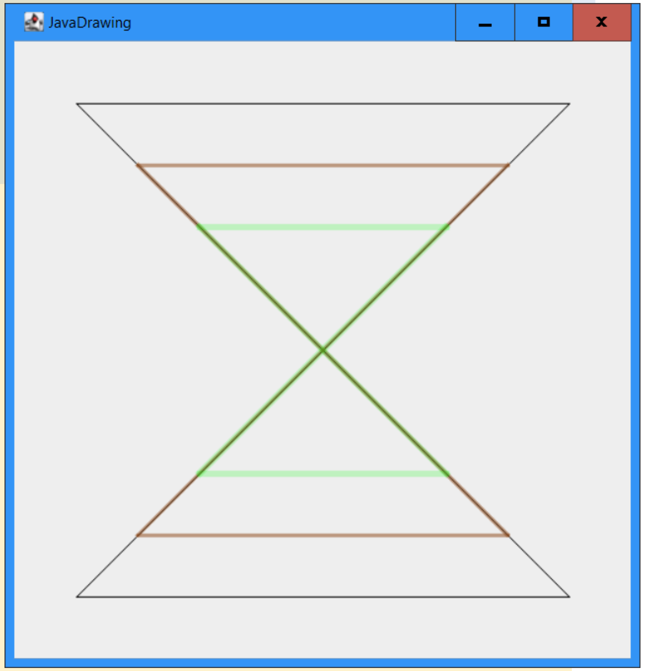
JDOR Redirecting Command Handlers

- If input is redirected, commands are taken from it
- If error is redirected, error messages will be written to it
- If output is redirected, all received commands get written in canonized form to output
  - Allows to record the commands
  - Rexx can save them and use them later as text macros to replay the commands

JDOR Sample "sample10.rex"

- Drawing colored lines

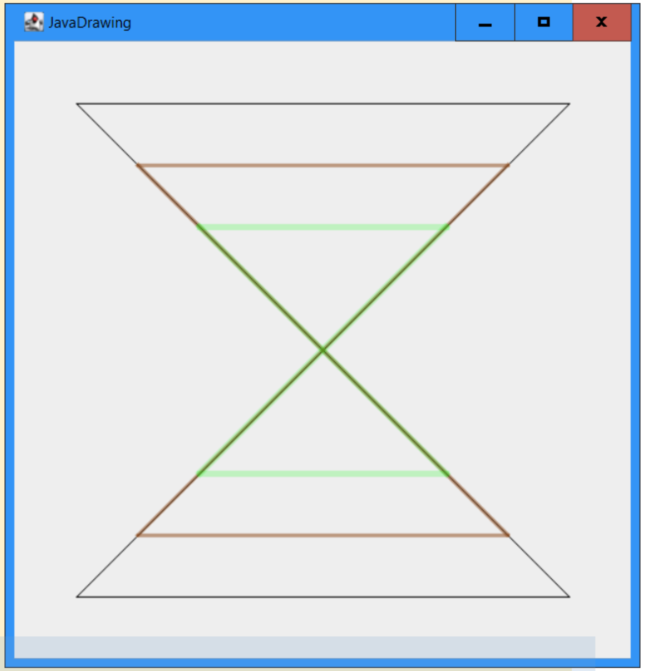
```
call addJdorHandler       -- load and add the Java Rexx command handler, using default name: JDOR
address jdor              -- set default environment to JDOR
stroke st5 5              -- define stroke of 5 pts
stroke st3 3              -- define stroke of 3 pts
stroke st1 1              -- define stroke of 1 pt
color saddlebrown50 139  69 19 127   -- R,G,B,alpha=127 (~50 % transparency)
color green20         0 255  0 50    -- R,G,B,alpha=127 (~20 % transparency)
strokes=st1, st3, st5                 -- Rexx array of stroke nick names
colors =black, saddlebrown50, green20 -- Rexx array of color nick names
width=500
height=500
new width height          -- create new bitmap
winShow                   -- show frame
do i=1 to 3
  m=50+(i-1)*50            -- margin
  w = width-m
  h = height-m
  color  colors[i]
  stroke strokes[i]
  goto m m; drawLine w m
  goto w m; drawLine m h
  goto m h; drawLine w h
  goto w h; drawLine m m
end
sleep 5                   -- sleep for five seconds

::requires "jdor.cls"   -- makes routine addJdorHandler([environmentName]) available
```



JDOR Sample "sample10a.rex"
Redirecting Output to a Stream (File), 1

- Drawing colored lines, redirecting output to a stream (file)

```
call addJdorHandler     -- load and add the Java Rexx command handler, using default name: JDOR
s=.stream~new("samp10a_commands.txt")~~open("replace")
address jdor with output using (s)      -- commands will be saved in canonized form
"stroke st15 15"              -- define and set stroke (15 pixels wide)
"stroke st05  5"              -- define and set stroke (5 pixels wide)
"stroke st01  1"              -- define and set stroke (1 pixel wide)
"color saddlebrown50 139  69 19 127"    -- R,G,B,alpha=127 (~50 % transparency)
"color green20         0 255  0 50"     -- R,G,B,alpha= 50 (~20 % transparency)
strokes=st01, st05, st15                -- define Rexx array of defined strokes
colors =black, saddlebrown50, green20   -- define Rexx array of defined colors
width =500
height=500
"NEW" width height            -- create new bitmap
"winshow"                     -- show frame
do i=1 to 3
  m=50+(i-1)*50               -- increase margin
  w = width-m
  h = height-m
  "color"  colors[i]          -- set defined color
  "STROKE" strokes[i]         -- set defined stroke
  "PoS"   m m; "DRAWLINE" w m -- sets the x and y position
  "GOTO" w m; "drawline" m h  -- GOTO is a synonym for POS
  "pos"   m h; "DRAWLINE" w h -- sets the x and y position
  "GOTO" w h; "drawline" m m  -- GOTO is a synonym for POS
end
s~close
saveImage  "samp10a.png"
sleep 5                 -- sleep for five seconds
::requires "jdor.cls"   -- makes routine addJdorHandler([environmentName]) available
```

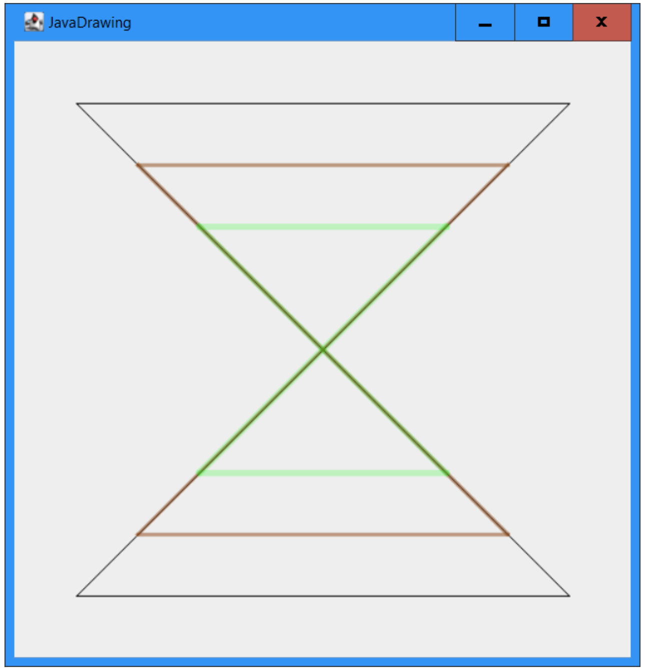


JDOR Sample "sample10a.rex" Redirecting Output to a Stream (File), 2

- Content of the stream (file) "samp10a_commands.txt"

```
stroke st15 15
stroke st05 5
stroke st01 1
color saddlebrown50 139 69 19 127
color green20 0 255 0 50
new 500 500
winShow
color BLACK
stroke ST01
pos 50 50
drawLine 450 50
pos 450 50
drawLine 50 450
pos 50 450
drawLine 450 450
pos 450 450
drawLine 50 50
color SADDLEBROWN50
stroke ST05
pos 100 100
```

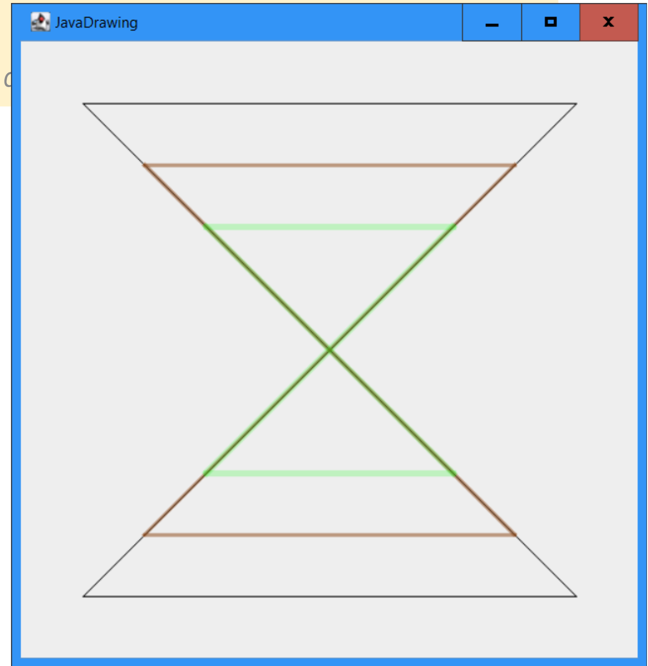
```
drawLine 400 100
pos 400 100
drawLine 100 400
pos 100 400
drawLine 400 400
pos 400 400
drawLine 100 100
color GREEN20
stroke ST15
pos 150 150
drawLine 350 150
pos 350 150
drawLine 150 350
pos 150 350
drawLine 350 350
pos 350 350
drawLine 150 150
saveImage samp10a.png
sleep 5.0
```



JDOR Sample "sample10b.rex" Feeding Commands from Redirected Input

- Drawing colored lines, getting commands from a stream (file)

```
call addJdorHandler     -- load and add the Java Rexx command handler, using default name: JDOR

s=.stream~new("samp10a_commands.txt")~~open("read")
address jdor with input using (s)
"-- a comment"    -- we need any command or comment to get the handler invoked
s~close

::requires "jdor.cls"   -- makes routine addJdorHandler([environmentName])
```



Roundup and Outlook

- BSF4ooRexx850 introduces the ability to implement redirecting command handlers in Java
- One can create command handlers in Java to e.g.
  - Replace/modernize outdated command handlers
  - Create new command handlers for new application areas to
    - Ease interaction and control with complex software systems
    - Make it possible to create textual macros of any complexity
    - Allow any application to create commands that can be executed by simple Rexx programs at any time
    - Allow to load and combine any number of command handlers at runtime
    - ...
- Once BSF4ooRexx850 goes beta the JDOR command handler will be placed in beta as well