



"Resurrecting REXX, Introducing Object Rexx"

ECOOP 2006, Nantes

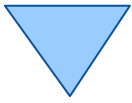
Workshop W10

"Revival of Dynamic Languages"

Monday, 2006-07-03

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)

Wirtschaftsuniversität Wien, Austria (<http://www.wu-wien.ac.at>)



Agenda

- A brief introduction to important REXX concepts
- A very brief introduction to important Object Rexx concepts
- Due to time constraints attempt to introduce concepts with nutshell examples hoping to evoke questions, discussions
- Not all foils are discussed (time restrictions)!

▼ REXX, 1

History

- 1979 by Mike F. Cowlshaw (MFC), IBM Hursley
 - Replacement for the somewhat awkward EXEC II mainframe scripting language
 - REXX should be "human centric" by comparison
 - ANSI Rexx standard in 1996
 - Many interpreters on different platforms
 - Cf. talk "25 Years of Rexx" by MFC at <http://rexxla.org/Symposium/2004/mikec.pdf>
- "Keep the language small"

REXX, 2

Fundamental Concepts (Code 1)

```
say "Hello world!" /* yields: Hello world! */
say 1/3            /* yields: 0.333333333 */

numeric digits 25 /* now use 25 significant digits in arithmetics */
say "1"/3         /* yields: 0.33333333333333333333333333333333 */

"rm -rf *"        /* remove all files recursively */
```

- "Everything is a string"
- Everything outside of quotes gets uppercased
- Arbitrarily precise decimal arithmetic
 - ANSI REXX rules served for defining the rules for other programming languages and is used for implementing decimal arithmetics in hardware
- Unknown statements are passed to invoker

REXX, 3

Stem Variables (Code 2)

```
file.1='max.txt'      /* variable "FILE.1" is assigned a value      */
file.2="pia.txt"     /* variable "FILE.2" is assigned a value      */
file.0=2             /* variable "FILE.0" is assigned a value      */

do i=1 to file.0     /* will loop twice with control variable "I"  */
  say file.i        /* "I" will be substituted with "1" and "2"  */
end
/* yields the following output:
max.txt
pia.txt
*/
```

- Stem: characters up to and including first dot
 - Allows to be used as a sort of an associative array
 - Additional dots delimit identifiers
- Uninitialized variable's value is the uppercased name of the variable itself

REXX, 4

Stem Variables (Code 3)

```
Austria.Tirol=500000      /* population of Tirol      */
austria.tirol.innsbruck=120000 /* population of the city Innsbruck */

a="TIROL.INNSBRUCK"      /* define the tail value    */
say Austria.a           /* displays: 120000        */

a="TIROL"; b='INNSBRUCK'; /* define "index" values   */
say austria.a.b         /* displays: 120000        */

say AusTriA.TiRoL.InnsBruck /* displays: 120000        */
```

- Stem variable uses uninitialized variables
 - Uppercase version of names used for substitution, as everything outside of quotes gets uppercased
- Variable values therefore must use the uppercase name of such variables

▼ Object Rexx, 1

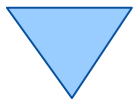
History

- Work started 1988 at IBM Hursley (Simon Nash) as a true object-oriented successor to REXX, the final concept was created and implemented in the US (Rick McGuire)
 - IBM large customers (SHARE) were in favor
 - Became an IBM product in 1997 as part of "OS/2 Warp"
 - Commercial versions for AIX and Windows
 - Experimental versions for Linux and Solaris

▼ Object Rexx, 2

History

- IBM handed source-code over to the non-profit SIG "Rexx Language Association (RexxLA)"
- Spring 2005 first opensource version "3.0"
 - <http://www.oORexx.org>
- Summer 2006 planned version "3.1"
 - Bug fixes
 - New class "CircularQueue"
- Future planned version "4.0"



Object Rexx, 3

Fundamental Concepts

- "Everything is an object"
- Interpreter, backwardly compatible with REXX
- Influenced by Smalltalk
- Among many things
 - Metaclasses
 - Reflection
 - One-off objects
 - ...
- Explicit message operator (~, tilde, "twiddle")
 - Cascading message (~~)

Object Rexx, 4 Class Hierarchy

- Very few classes

Class Name		Comment
Object		Root class, fundamental
	Alarm	Send message asynchronously
	Array	Collection class, no predefinition of size or dimension necessary, ordered
	Class	Metaclass, fundamental
	Directory	Collection class, index is a string, one object per index, no order implied
	List	Collection class, ordered
	Message	Fundamental class
	Method	Fundamental class
	Monitor	Monitors messages sent to objects
	MutableBuffer	Comparable to Java's <code>StringBuffer</code>
	Queue	Collection class, ordered
	Relation	Collection class, index is any object, multiple objects per index possible, no order implied
		Bag Index and associated object are the same object
	Stem	Represents "classic Rexx" stems
	Stream	Stream (e.g. file) input/output
	String	Object's string values are not mutable
	Supplier	Iterator for collection classes
	Table	Collection class, index is any object, one object per index, no order implied
		Set Index and associated object are the same object



Object Rexx, 5

Classic and Object Rexx (Code 4)

```
say reverse("aloha") /* the reverse function returns: aloha */  
say "aloha"~reverse /* the reverse message returns: aloha */
```

- Classic Rexx
 - Syntax still allowed
 - Statements get transformed to the object-oriented version "behind the curtain"



Object Rexx, 6

Defining and Using Classes (Code 5)

```
.Dog ~new("Sweety") ~bark /* create a dog, let it bark */
.BigDog~new("Grobian")~bark /* create a big dog, let it bark */

::class Dog
::method init /* constructor method */
  expose name /* establish direct access to attribute (object variable) */
  use arg name /* retrieve argument, assign it to attribute */
::method name attribute /* define set and get attribute methods */
::method bark
  say self~name:" "Wuff Wuff"

::class BigDog subclass Dog
::method bark
  say self~Name:" "WUFFF! WUFFF!! WUFFF!!!"

/* yields the following output:
  Sweety: Wuff Wuff
  Grobian: WUFFF! WUFFF!! WUFFF!!!
*/
```

Object Rexx, 7

Multiple Inheritance (Code 6)

```
/* Multiple Inheritance */
.RoadVehicle      ~new("Truck")  ~drive
.WaterVehicle    ~new("Boat")   ~swim
.AmphibianVehicle~new("SwimCar")~show_off

::CLASS  Vehicle          /* define the vehicle base class          */
::METHOD name ATTRIBUTE /* let interpreter define a getter and setter method */
::METHOD init            /* define constructor method          */
    self~name=ARG(1)     /* use the setter method to set the attribute's value */

::CLASS  RoadVehicle     MIXINCLASS Vehicle
::METHOD drive          /* define a road vehicle method      */
    SAY self~name": 'I drive now...'" /* use the attribute getter method */

::CLASS  WaterVehicle    MIXINCLASS Vehicle
::METHOD swim           /* define a water vehicle method     */
    SAY self~name": 'I swim now...'" /* use the attribute getter method */

::CLASS  AmphibianVehicle SUBCLASS RoadVehicle INHERIT WaterVehicle
::METHOD show_off      /* demonstrate multiple (implementation) inheritance */
    self ~~drive ~~swim /* using cascading messages (two twiddles) */

/* yields the following output:
Truck: 'I drive now...'
Boat: 'I swim now...'
SwimCar: 'I drive now...'
SwimCar: 'I swim now...'
*/
```

Object Rexx, 8

"Swiss Army Knife": Windows (Code 7)

```
call orexhole.cls          /* get the COM/OLE/ActiveX support          */
/* create an instance (a proxy) of the Internet Explorer      */
myIE = .OLEObject~New("InternetExplorer.Application")

myIE~Width = 1024         /* set the width in pixels          */
myIE~Height = 768        /* set the height in pixels         */
myIE~Visible = .True     /* now show the window              */
myIE~Navigate("http://www.ooRexx.org")

say "sleeping 15 seconds..."
Call SysSleep 15
myIE~quit                 /* now close the Internet Explorer */
```

- Drives Windows and Windows applications via OLE/ActiveX
- Windows objects look like Object Rexx objects to which one can send Object Rexx messages

Object Rexx, 9

"Swiss Army Knife": Java (Code 8)

```
call bsf.cls          /* get the Java support          */
s=.bsf~bsf.import("java.lang.System") /* import the Java System class */
say "java.version:" s~getProperty('java.version')

/* yields the following output (maybe):
   java.version: 1.5.0_06
*/
```

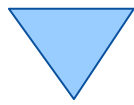
- Drives Java, Java applications and applications having Java programming interfaces
- Java objects look like Object Rexx objects to which one can send Object Rexx messages
- <<http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/>>

Object Rexx, 10

"Swiss Army Knife": OOo (Code 9)

```
call uno.cls /* get the UNO (StarOffice/OpenOffice, OOo) support */
oDesktop=UNO.createDesktop() /* get OOo desktop service object */
xComponentLoader=oDesktop~XDesktop~XComponentLoader
/* open the blank *.sxw - file */
url = "private:factory/swriter" /* define the document URL */
/* the following statement spans two lines */
xWriterComponent=xComponentLoader~loadComponentFromURL(url,"_blank", 0, -
                                                         .UNO~noProps )
xText=xWriterComponent~XTextDocument~getText -- get the OOo text object
xText~setString("Hello World, this is ooRexx speaking!") /* insert text */
```

- Drives OpenOffice.org (Ooo)/StarOffice (SO)
- OOo objects look like Object Rexx objects to which one can send Object Rexx messages
- <<http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/>>
- Multiplatform (runs unchanged on Lin & Win)



Object Rexx, 11

A Few Concluding Remarks

- Interesting, that practically unknown despite an active user base
- Actively developed
- Multiplatform, opensource
- Weakly typed language which turns into a powerful advantage as demonstrated
 - Automating/Scripting Windows (applications)
 - Automating/Scripting Java (applications)
- Questions, remarks, comments, ideas?