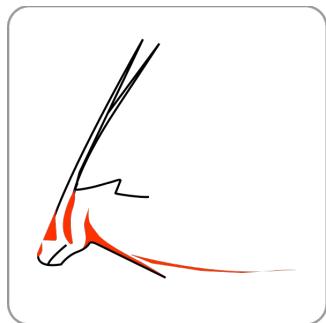


# Introduction to REXX and ooREXX

## From REXX to Open Object REXX (ooREXX)

Rony G. Flatscher



*"REXX was invented to make programming easier than ever before. Therefore the programming language was designed to be 'human centric' and, as a result, it can be learned and understood very quickly!"*

*"ooREXX adds all the object-oriented bells and whistles, yet keeps the language 'human centric'."*

*"This book gives you an incredibly useful, productive and durable 'Swiss Army Knife' (SAK) kind of programming tool. You can quickly create glue code, scripts, macros and even full blown applications for any operating system and deeply understand REXX and ooREXX code easily and clearly."*

*Rony G. Flatscher* works as a professor for Information Systems (German: "Wirtschaftsinformatik") at the *WU Wien*, Austria, a business university with approximately 25,000 students.

He has been experimenting over a decade teaching programming to end-users of information systems using various programming languages. In the course of time, a lecture of two consecutive classes was developed which successfully introduces the students to programming, object-oriented programming and scripting/remote-controlling business applications such as Microsoft Office or (in an operating system independent manner) Apache OpenOffice.

One key element that allows for this to be done in a very short time is the choice of programming language: "*Open Object Rexx (ooRexx)*", a human centric, easy to understand and easy to use programming language that originates in the IBM product "*Object REXX*" which was handed over to the open-source community.

This book introduces the principles of the programming language ooRexx in a very concise manner. It demonstrates all of the introduced concepts immediately with "*nutshell examples*" (very small programs) showing their output, which allows anyone to study and learn the language *by just reading* this introductory book.

---

Author's ooRexx related URL: <http://www.RonyRexx.net>

---

*Howard Fosdick (USA)*, the author of "*Rexx – Programmer's Reference*" says about this book:

*"Excellent work! This is the book we all have been waiting for. I only wish it had been available when I was learning ooRexx!"*

...

*I'd like to conclude by emphasizing that this just is a really fantastic book, Rony, written by the one person who is so well qualified to write it. Congratulations!"*

Rony G. Flatscher

# Introduction to REXX and ooREXX

From REXX to Open Object REXX (ooREXX)

---

First Edition: 1.00, 2013-02-15 (Version 100.20130215)

Copyright © 2013 Rony G. Flatscher  
<http://www.RonyREXX.net>

c/o WU Wien, Augasse 2-6, A-1090 Wien  
All rights reserved.

Printed by: Facultas Verlags- und Buchhandels AG, Austria, Europe

## Acknowledgements

The author thanks the following persons for their feedback, proof reading and help in creating free Rexx-related art (in alphabetical order):

- Gilbert *Barmwater* (U.S.A.): feedback, proof reading
- Daniel A. *Flatscher* (Austria): proof reading
- Howard *Fosdick* (U.S.A.): feedback, proof reading
- René Vincent *Jansen* (The Netherlands): feedback, proof reading
- Les *Koehler* (U.S.A.): feedback, proof reading
- Gerald *Leitner* (Austria): feedback, proof reading
- DI Walter *Pachl* (Austria): feedback, proof reading
- Graham *Wilson* (South Africa): art including icons for BSF4ooRexx
- Jon “Sahananda” *Wolfers* (United Kingdom): feedback, proof reading

## Foreword

**A Brief History of the Rexx programming language.** In 1979 *Mike F. Cowlishaw* (MFC), an English gentleman working for IBM, devised a “human centric” programming language for the IBM mainframes that was easier to understand and to program than the arcane mainframe batch language named *Exec 2*. The design work was carried out at the IBM Research facilities in Hursley under the management of *Dr. Brian Marks*. It was probably the first time in the history of programming language design that IBMers interconnected worldwide via the IBM internal network were able to influence the design by studying the distributed specifications and giving feedback, like *Les Koehler* from IBM USA.

IBM later defined the *REXX* programming language to be the strategic batch/scripting language on all of its operating systems via IBM's *SAA* (System Application Architecture) standard. Another outstanding IBM employee who has probably been the only person to create Rexx interpreters multiple times for multiple operating systems is *Rick McGuire*, who led the development and maintenance of the *IBM SAA REXX* interpreters.

The IBM lab in Vienna created a Rexx compiler for its mainframe *REXX* (*Klaus Hansjakob, Walter Pachl*), which is being sold and maintained by IBM to this very day.

*Mike F. Cowlishaw* documented the *REXX* language in a book named “The Rexx Language” also known as “TRL” and he later became one of the few IBM Fellows<sup>1</sup> due to his continuing innovative and influential work (he is also attributed to be the person who made Java a strategic language and platform within IBM, having ported Java to the IBM OS/2 PC operating system in the 90'ies).

The high impact of the Rexx language can be witnessed by the appearance of numerous non-IBM implementations of the Rexx language, e.g. *Regina* (open source, *Anders Christensen, Mark Hessling*), *Rexx/imc* (open source, *Ian M. Collier*) or *BREXX* (open source, *Vassilis N. Vlachoudis*), but also proprietary and commercial Rexx interpreters like *ARexx* (part of the Amiga operating system), Novell Netware's Rexx (in the 90'ies), Workstation Unix

---

<sup>1</sup> An IBM Fellow is free to research and to work, very much like professors at Universities, who have the freedom to freely determine what they research and what they teach.

## Foreword

Rexx. Some pointers to various Rexx interpreters can be found at <http://www.rexxla.org/rexxlang/mfc/rexxplat.html>.

**ANSI/INCITS REXX Standard.** In 1996 the American National Standards Institute (ANSI) working group X3J18 finalized the “American National Standard for Information Systems - Programming Language REXX”. After ANSI got renamed to “INCITS (InterNational Committee for Information Technology Standards, <http://www.incits.org/>)” the respective standard was named “INCITS 274 :1996 [R2001]”. In 2007 the INCITS 274 REXX standard was extended for another period of ten years, reflecting the importance of the Rexx language in the industry.

The INCITS 274 REXX standards on decimal arithmetic served as the basis for defining decimal formats in IEE 754-2008 and ISO/IEC/IEEE 60559:2011. An ANSI C implementation has been created by *Mike F. Cowlishaw* (<http://www.speleotrove.com/mfc/>), who on behalf of IBM has been a driving force behind standardizing decimal arithmetics in the context of IEEE, Java JSR-13 and who also implemented an open source `decNumber` package in ANSI C.

**A Brief History of the ooRexx Language.** At the end of the 80'ies Dr. Brian Marks oversaw another interesting project, named “Oryx” with the technical lead of *Simon Nash*. The aim of this project was to experiment with a Rexx interpreter that extends the Rexx language with object-oriented features. This work would later lead, under the auspices of *Rick McGuire*, to the IBM product “Object REXX”. It was first distributed with OS/2 Warp 4 in 1997, versions for IBM AIX and MS Windows were created and sold as well by IBM.

In 2004 IBM handed the source code of “Object REXX” over to the non-profit special interest group (SIG) “Rexx Language Association (RexxLA, <http://www.RexxLA.org>)”. RexxLA published the first open source version of IBM’s “Object REXX” as “Open Object Rexx 3.0 (<http://www.ooRexx.org>)” in 2005. The lead architect of this now open-source project has been *Rick McGuire*, who has been working in his own time on the open source version of ooRexx ever since.

## What Is ooRexx?

- A “classic Rexx” interpreter. ooRexx runs any “classic Rexx” program and can be used to write “classic Rexx” programs. There is no need to use any of its new features that extend the Rexx language.

- *An object-oriented Rexx language, hence “ooRexx”:* ooRexx comes with many useful classes (data types) and offers state-of-the art object-oriented features, devised in a “human-centric” way. Among other great things, ooRexx makes it easy to create multithreaded Rexx programs!
- *Fast and powerful:* ooRexx is a fast Rexx interpreter. It is a very powerful interpreter, which can be invoked from C++ or Java (via BSF4ooRexx) to allow Rexx and ooRexx macros/programs to run with C++ and Java applications. For C++ and Java applications it is possible to run multiple ooRexx interpreter instances in the same process space, each of which may execute even multithreaded Rexx code!
- *Great documentation:* IBM not only donated the source code for open sourcing to RexxLA, but also the excellent and professional technical documentation, which has been kept up-to-date. All the ooRexx documentation is available in the form of HTML and PDF-files, which can be nicely printed as books. The documentation is also directly available via the Internet: <http://www.oorexx.org/docs/>.
- *Free and open source:* originally created by IBM and marketed as “Object REXX”, RexxLA received the sources for publishing, maintaining, and enhancing this powerful Rexx interpreter. RexxLA distributes ooRexx with source code for free: <http://www.ooRexx.org>.
- *Multiplatform:* ooRexx is available in 32 and 64 bit versions for the operating systems AIX, Linux, MacOSX, Windows, and can be built for any Unix implementation. Rexx programs written in one operating system environment can execute in any other operating system environment.
- *Extensible:* ooRexx comes with a powerful and easy to use C++ API which is documented in one of the accompanying ooRexx documentation PDFs (cf. `rexxpg.pdf`). This allows you to extend ooRexx with functions and methods implemented in C++, but also to bridge Rexx with other programming infrastructures like Java (cf. “BSF4ooRexx”). In addition it allows C++ applications to create Rexx interpreter instances which execute Rexx programs. This way it is fairly easy/simple to employ ooRexx as a macro language for any C++

## Foreword

applications. The BSF4ooRexx extension package provides the same functionality for Java applications.

The author wishes to acknowledge the following persons important to the Rexx world in the context of RexxLA in alphabetic order: *Gil Barmwater* (vice president), *Mike F. Cowlishaw* (honorary board member), *Chip Davis* (past president), *Mark Hessling* (board member), *René Vincent Jansen* (current president), *Les Koehler* (secretary/treasurer), *Lee Peedin* (past president), *Pam Taylor* (board member), *Jon "Sahananda" Wolfers* (board member).

Of course all the developers of ooRexx (including past) are acknowledged hereby (in alphabetic order): *David Ashley*, *Jean-Louis Faucher*, *Mark Hessling*, *Moritz Hoffmann*, *Rick McGuire*, *Mark Miesfeld*, *Lee Peedin*, *David Ruggles*, *Bruce Skelly*, *Rainer Tammer*, *Jon Wolfers*.

## About this Book

This book introduces the programming language Open Object Rexx, also known as "ooRexx" in two steps:

1. Chapter 1 'The Rexx Language ("Everything Is a String")' introduces the Rexx programming language that was created in 1979 by the IBM employee Mike F. Cowlishaw who later became an IBM Fellow due to his work on Rexx. The most important design philosophy for the language was the principle of "human-orientation", making it easy for programmers to create programs in the Rexx language compared to the arcane IBM mainframe batch language Exec 2 which Rexx successfully replaced. One key success factor of the Rexx language has been its easy English-like syntax that makes it easy to learn, fast to comprehend, easy to apply and inexpensive to maintain. Rexx programs can be read almost like prose. As ooRexx is backwardly compatible to Rexx it can be used to learn Rexx and thereby the fundamentals of programming. The concepts in this chapter apply generally to all existing Rexx interpreters, which sometimes are called "classic Rexx" interpreters (as opposed to ooRexx, which is a leading edge Rexx interpreter that extends classic Rexx nicely into the object-oriented world). ooRexx-only features are highlighted in the text.
2. Chapter 2 'Extensions to the Rexx Language by ooRexx' documents the ooRexx-only keyword instructions (LOOP, RAISE, USE) and enhancements to the Rexx language like short hand assignment operators (e.g. "+=") and the directives ::routine and ::requires that may prove quite helpful to "classic Rexx" programmers.
3. Chapter 3 'The ooRexx Language ("Everything Is an Object")' builds upon the previous chapters and introduces the fundamental concepts of what is known as the "object-oriented (OO) paradigm". This is followed by an overview of the numerous new classes (data types) that come with ooRexx and which could be exploited by Rexx programmers to ease their programming life considerably in most cases. At the end of this chapter the reader should understand the OO-concepts and be able to take advantage of these new, powerful features!
4. Chapter '4 Reaching Out with ooRexx' opens with useful information about the ooRexx runtime system, followed by a categorized overview

## About this Book

of the ooRexx classes (data types, types) that are installed with the interpreter. The ooRexx programmer can directly use these ooRexx classes and take advantage of the features they implement.

5. Chapter 5 'Advanced Topics' introduces the interested reader to defining and implementing Rexx classes (data types), which is very easy and straight-forward. For those programmers who need the ability to create `Rexx` programs in which parts are executed concurrently, there is a concluding section which explains and demonstrates how easy it is to do that with `ooRexx`.

The structure and contents of the book are aimed at people who are interested in learning programming in `Rexx` and afterwards `ooRexx`. Still, it aims to introduce and demonstrate the concepts in a very concise, yet understandable manner. The reader is advised to consult the excellent `ooRexx` reference documentation, which completely documents `ooRexx` and is available as a nicely formatted PDF-book, named `rexxref.pdf` (<http://www.oorexx.org/docs/rexxref/rexxref.pdf>).

The way this book is written should also allow professional programmers to skim the book and learn about the fundamentals of `Rexx` and `ooRexx` by looking out for the definition boxes that are formatted like this:

This is how a definition box is formatted. Definition boxes allow you to quickly get (re-)acquainted with the fundamental concepts that are taught in a chapter. This book will sometimes directly use the definitions of the ANSI/INCITS Rexx standard if possible in this book's context.

In addition, numerous little "nutshell programs" or "code snippets" demonstrate how to apply the introduced concepts. These programs, as short as they may seem, are full programs that can be executed as is by the `ooRexx` interpreter, yielding the output that is sometimes depicted alongside the program as well. "Nutshell programs" are formatted like this:

```
say "Hello world, this is Rexx speaking"
```

The above program will output the string `Hello world, this is Rexx speaking`.

Alternatively, ooRexx for Windows comes with a GUI program (menu entry named "Try Rexx (GUI)") which allows you to enter Rexx code and execute it with the push of a button. ooRexx users on Linux or Mac OSX might want to

install BSF4ooRexx<sup>2</sup> (<https://sourceforge.net/projects/bsf4oorexx/files/GA/>) which comes with a comparable GUI program (menu entry named “GUI RexxTry Program (ooRexxTry.rxj)”).

Finally, ooRexx can be downloaded for free from one of the following locations:

- <http://www.oorexx.org/download.html>
- <https://sourceforge.net/projects/oorexx/files/>

There are editors that support Rexx syntax highlighting, for example the following two free and open source editors:

- “The Hessling Editor (THE)”, which uses Rexx as its macro language, URL: <http://hessling-editor.sourceforge.net/>
- “vim (vi improved)”, a part of many Linux distributions, is generally available for all operating systems, URL: <http://www.vim.org/>

---

<sup>2</sup> This GUI program is also available for Windows, if “BSF4ooRexx” gets installed there. “BSF4ooRexx” is an ooRexx external function package that allows ooRexx programs to interact directly with Java, which gets camouflaged as ooRexx. At the time of writing this external function package is available for Linux, MacOSX and Windows. Cf. 4.1 Exploiting Java on All Platforms, p. 159 below.

## Table of Contents

Acknowledgements .....	ii
Foreword .....	iii
About this Book .....	vii
Table of Contents .....	xi
List of Figures .....	xv
List of Tables .....	xv
List of Codes (Nutshell Examples) .....	xv
Part I .....	21
1 The REXX Language (“Everything Is a String”) .....	23
1.1 “Hello world!” in REXX .....	23
1.2 Fundamental Language Concepts .....	24
1.3 Building Blocks and Definitions .....	26
1.3.1 Characters Allowed in a REXX Program .....	26
1.3.2 Comments .....	27
1.3.3 Literal Strings .....	28
1.3.4 String Values .....	29
1.3.5 Symbols (Names) .....	30
1.3.6 Environment Symbols (ooREXX) .....	31
1.3.7 Operators .....	32
1.3.8 Expressions .....	34
1.3.8.1 Functions .....	34
1.3.8.2 String Concatenation Expressions .....	38
1.3.8.3 Arithmetic Expressions .....	39
1.3.8.4 Boolean Expressions .....	41
1.3.8.4.1 Comparisons .....	41
Comparing String Values .....	41
Comparing Numbers .....	42
1.3.8.4.2 Negator .....	43
1.3.8.4.3 Combining Boolean Values .....	43
1.3.9 Clauses .....	45
1.3.10 Normalizing Clauses (“Behind the Curtain”) .....	46
1.4 Writing REXX Programs .....	49
1.4.1 Assignment Instructions (Variables) .....	49
1.4.1.1 Shorthand Assignment Instructions (ooREXX) .....	50
1.4.1.2 Stem Variables .....	51
1.4.1.3 Variable Names that REXX May Use Without Notice .....	52
1.4.2 Label Instructions (CALL, SIGNAL, EXIT, RETURN, PROCEDURE, EXPOSE Keyword Instructions) .....	53
1.4.3 Message Instructions (ooREXX) .....	59
1.4.4 Keyword Instructions .....	61
1.4.4.1 SAY (Output of Strings) .....	64
1.4.4.2 IF (Choose) .....	64
1.4.4.3 SELECT (Alternatives), LEAVE (ooREXX) .....	65
1.4.4.4 DO (Block), ITERATE, LEAVE .....	66
1.4.4.5 LOOP (ooREXX), ITERATE, LEAVE .....	72

## Table of Contents

1.4.4.6 The Rexx QUEUE (PUSH, QUEUE, PARSE PULL, PULL) ....	72
1.4.4.7 PARSE ..... 1.4.4.7.1 Parsing Using a Blank Delimited Template .....	74
1.4.4.7.2 Parsing Using Literal Strings as Delimiter .....	76
1.4.4.7.3 Parsing Using Positions and Lengths .....	77
1.4.4.7.4 PARSE VALUE .....	80
1.4.4.7.5 PARSE ARG, ARG .....	81
1.4.4.7.6 PARSE PULL, PULL .....	82
1.4.4.7.7 PARSE SOURCE .....	82
1.4.4.7.8 PARSE VERSION .....	83
1.4.4.8 TRACE .....	83
1.4.4.9 INTERPRET .....	84
1.4.5 Command Instructions, ADDRESS .....	85
2 Extensions to the Rexx Language by ooRexx .....	87
2.1 USE ARG .....	88
2.2 Trapping and Raising Conditions (SIGNAL CALL ON OFF, RAISE) .	89
2.3 Directives .....	93
2.3.1 The ::ROUTINE Directive .....	93
2.3.2 The ::REQUIRES Directive .....	98
2.4 The Big Picture .....	101
Part II .....	103
3 The ooRexx Language ("Everything Is an Object") .....	105
3.1 Running ooRexx Programs .....	106
3.1.1 Runtime Environment .....	107
3.1.2 Objects (Values, Instances), Classes (Data Types) .....	109
3.1.2.1 Messages: Interacting with Objects .....	109
3.1.2.2 Classes: Attributes, Methods .....	110
3.1.2.3 Classes (Data Types) Organized as a Class Hierarchy .....	112
3.1.2.4 Unknown Messages .....	114
3.2 The ooRexx Built-in Classes .....	115
3.2.1 Categorizing the ooRexx Classes .....	115
3.2.2 The Fundamental ooRexx Classes .....	115
3.2.2.1 The .Object Class .....	116
3.2.2.2 The .Class Class (the ooRexx Metaclass) .....	117
3.2.2.3 The .Method Class .....	119
3.2.2.4 The .Message Class .....	120
3.2.2.5 The .Routine Class .....	121
3.2.2.6 The .Package Class .....	122
3.2.3 The Classic Rexx Classes .....	123
3.2.3.1 The .String Class .....	123
3.2.3.2 The .Stem Class .....	125
3.2.3.3 The .Stream Class .....	127
3.2.4 The ooRexx Collection Classes .....	128
3.2.4.1 The .OrderedCollection Classes .....	130
3.2.4.1.1 The .Array Class .....	131
3.2.4.1.2 The .List Class .....	133
3.2.4.1.3 The .Queue Class .....	134

## Table of Contents

3.2.4.1.4 The .CircularQueue Class .....	136
3.2.4.2 The .MapCollection Classes .....	136
3.2.4.2.1 The .Directory Class .....	137
3.2.4.2.2 The .Relation Class .....	140
3.2.4.2.3 The .Table Class .....	142
3.2.4.3 The .SetCollection Classes .....	142
3.2.4.3.1 The .Bag Class .....	143
3.2.4.3.2 The .Set Class .....	144
3.2.4.4 Setlike Operations on Collections .....	145
3.2.4.4.1 Example: Setlike Operations with .Bag and .Bag .....	147
3.2.4.4.2 Example: Setlike Operations with .Set and .Bag .....	148
3.2.4.4.3 Example: Setlike Operations with .Bag and .Set .....	149
3.2.5 Miscellaneous ooRexx Classes .....	150
3.2.5.1 The .Alarm Class .....	150
3.2.5.2 The .Comparator Classes .....	151
3.2.5.3 The .DateTime and .TimeSpan Classes .....	152
3.2.5.4 The .File Class .....	153
3.2.5.5 The .Monitor Class .....	154
3.2.5.6 The .MutableBuffer Class .....	155
3.2.5.7 The .RexxContext Class .....	156
4 Reaching Out with ooRexx .....	159
4.1 Exploiting Java on All Platforms .....	159
4.1.1 A Brief Overview of BSF4ooRexx .....	160
4.1.1.1 BSF4ooRexx Menu .....	161
4.1.1.2 ooRexxTry.rxf – A Platform Independent GUI for ooRexx ..	162
4.1.2 The ooRexx Package BSF.CLS .....	162
4.1.3 The ooRexx Package UNO.CLS .....	164
4.1.4 Further Information .....	167
4.2 Windows Platform Only .....	169
4.2.1 A Brief Overview of COM, OLE, ActiveX .....	170
4.2.2 The .OLEObject Class .....	171
4.2.2.1 Rosetta Stone: Visual Basic to/from ooRexx .....	175
4.2.2.2 Some Further Information .....	180
4.2.3 The ooDialog Framework .....	181
4.2.4 Additional ooRexx Windows Classes .....	182
5 Advanced Topics .....	185
5.1 A Few Things that Might Be Helpful to Know .....	185
5.1.1 About ooRexx Directives .....	185
5.1.2 About ooRexx Scopes .....	186
5.1.3 About the .methods Environment Symbol .....	187
5.1.4 About Cascading Messages .....	188
5.1.5 About Required String Values .....	189
5.1.6 About Special ooRexx Methods .....	190
5.1.6.1 Method “init” (Constructor) .....	190
5.1.6.2 Method “unInit” (Destructor) .....	191
5.1.6.3 Method “unknown” .....	193
5.1.6.4 Method “string” .....	193

## Table of Contents

5.1.6.5 Method "makeString" .....	194
5.1.6.6 Method "makeArray" .....	196
5.1.6.7 Comparison Methods .....	196
5.2 Defining ooRexx Classes .....	203
5.2.1 Abstract Data Type (ADT) .....	203
5.2.2 Implementing an ADT with Directives (::CLASS, ::METHOD, ::ATTRIBUTE, ::CONSTANT Directives) .....	205
5.2.2.1 The ::CLASS Directive .....	205
5.2.2.2 The ::ATTRIBUTE and the ::METHOD Directives .....	207
5.2.2.2.1 Method Scope (EXPOSE) .....	212
5.2.2.2.2 "Self" and "Super" in Method Routines .....	212
5.2.2.2.3 FORWARD .....	214
5.2.2.3 The ::CONSTANT Directive .....	214
5.2.3 Examples .....	215
5.2.3.1 Creating a Hierarchy of Classes .....	215
5.2.3.2 Employing Multiple Inheritance .....	217
5.3 Multithreaded Programming .....	219
5.3.1 REPLY, .Alarm, .Object and .Message .....	220
5.3.1.1 REPLY .....	220
5.3.1.2 The .Alarm Class .....	222
5.3.1.3 Method "start" of the .Object Class .....	223
5.3.1.4 Method "start" of the .Message Class .....	223
5.3.2 Synchronizing Rexx Threads (GUARD) .....	224
5.3.2.1 Synchronizing Concurrently Running Method Routines ...	225
5.3.2.2 Waiter .....	228
5.3.2.3 Producer and Consumer .....	230
Index .....	235
Some oo Rexx-Related World-Wide-Web Links .....	ccliv

## List of Figures

Figure 1: ooRexxTry.rxj:a Portable GUI for Experimenting with ooRexx.	162
Figure 2: .bsfDialog's messageBox on Linux, MacOSX and Windows. ....	164
Figure 3: LibreOffice Writer on Linux. ....	166
Figure 4: Apache OpenOffice Writer on MacOSX. ....	167
Figure 5: Apache OpenOffice Writer on Windows. ....	167
Figure 6: Windows Script Host (wsh) Popup. ....	171
Figure 7: ooRexx Homepage ( <a href="http://www.ooRexx.org">http://www.ooRexx.org</a> ). ....	175
Figure 8: ooDialog's Example "New List Controls". ....	182

## List of Tables

Table 1: Rexx Operators. ....	33
Table 2: Overview of the Rexx Built-in Functions (BIFs). ....	37
Table 3: Combining Boolean Values with "&" (AND). ....	43
Table 4: Combining Boolean Values with " " (OR). ....	44
Table 5: Combining Boolean Values with "&&" (XOR). ....	44
Table 8: Some Methods of the ooRexx Root Class "Object". ....	116
Table 9: Some Methods of the ooRexx Meta Class "Class". ....	118
Table 10: Some Methods of the ooRexx Class "Method". ....	119
Table 11: Some Methods of the ooRexx Class "Message". ....	121
Table 12: Some Methods of the ooRexx Class "Routine". ....	121
Table 13: Some Methods of the ooRexx Class "Package". ....	122
Table 14: Some Methods of the ooRexx Class "String". ....	124
Table 15: Some Methods of the ooRexx Class "Stream". ....	127
Table 16: Some Methods of the ooRexx Class "DateTime". ....	152
Table 17: Some Methods of the ooRexx Class "TimeSpan". ....	152
Table 18: Some Methods of the ooRexx Class "File". ....	154
Table 19: Some Methods of the ooRexx Class "MutableBuffer". ....	156
Table 20: Some Methods of the ooRexx Class "RexxContext". ....	157
Table 21: Some Methods of the ooRexx Class "OLEObject". ....	172

## List of Codes (Nutshell Examples)

Code 1.1-1: "Hello world!" ....	23
Code 1.3-1: Nested Block Comments. ....	27
Code 1.3-2: Line Comment. ....	28
Code 1.3-3: Rexx Clauses and Semicolons. ....	45
Code 1.3-4: Rexx Clauses (Using Comma, Dash and Semicolons). ....	46
Code 1.3-5: Rexx Program with a Null Clause. ....	46
Code 1.3-6: Rexx Clause, Spreading Over Five Lines with Whitespace. ....	48
Code 1.3-7: Normalized Clause. ....	48
Code 1.4-1: Stem Variables. ....	51
Code 1.4-2: Stem Defined with a Default Value. ....	51
Code 1.4-3: Stem Array. ....	52
Code 1.4-4: Transferring Control to a Label with SIGNAL. ....	54
Code 1.4-5: Transferring Control to a Label with CALL. ....	54

## List of Codes (Nutshell Examples)

Code 1.4-6: Invoking an Internal Routine as a Function. ....	55
Code 1.4-7: Calling an Internal Routine, that Returns a Value. ....	55
Code 1.4-8: Fetching Arguments Using the ARG() BIF. ....	55
Code 1.4-9: Changing Variables from the Caller in an Internal Routine. ...	56
Code 1.4-10: Procedure Scope - Insulating Caller's Variables. ....	57
Code 1.4-11: Procedure Scope - Exposing One Caller's Variable. ....	57
Code 1.4-12: Procedure Scope - Exposing a Stem from the Caller. ....	58
Code 1.4-13: Using a Stem Array from the RexxUtil Function SysFileTree(). .....	59
Code 1.4-14: Message Instructions. ....	60
Code 1.4-15: Demonstrating the SAY Keyword Instruction. ....	64
Code 1.4-16: Demonstrating the IF Keyword Instruction. ....	64
Code 1.4-17: Demonstrating the IF Keyword Instruction with ELSE. ....	65
Code 1.4-18: Demonstrating the Clause Borders by Semicolons. ....	65
Code 1.4-19: An Alternative for Formatting an IF Keyword Instruction. ....	65
Code 1.4-20: Demonstrating the select Keyword Instruction. ....	66
Code 1.4-21: Demonstrating the do Keyword Instruction. ....	66
Code 1.4-22: Demonstrating the do Keyword Instruction with Repetition. ....	67
Code 1.4-23: Demonstrating the do Keyword Instruction with Repetition. ....	67
Code 1.4-24: Demonstrating Repetition with a Control Variable. ....	68
Code 1.4-25: Demonstrating Repetition with a Control Variable. ....	68
Code 1.4-26: Demonstrating Repetition with a Control Variable. ....	68
Code 1.4-27: Using an Explicit Label Subkeyword. ....	69
Code 1.4-28: Using the do Keyword Instruction with the WHILE Subkeyword. ....	69
Code 1.4-29: Using the do Keyword Instruction with the WHILE Subkeyword. ....	69
Code 1.4-30: Using the do Keyword Instruction with the UNTIL Subkeyword. ....	70
Code 1.4-31: Using the LEAVE and ITERATE Keyword Instructions. ....	70
Code 1.4-32: Using the FOREVER Keyword Instruction. ....	71
Code 1.4-33: Using the DO...over Keyword Instruction. ....	71
Code 1.4-34: Using the LOOP Keyword Instruction. ....	72
Code 1.4-35: Using the QUEUE, PUSH, PULL and PARSE PULL Keyword Instructions. ....	73
Code 1.4-36: Accessing the Output of an External Command via rxqueue. ....	73
Code 1.4-37: Using the PARSE Keyword Instruction. ....	74
Code 1.4-38: Using the PARSE Keyword Instruction, Version 2. ....	75
Code 1.4-39: Using the PARSE Keyword Instruction with Literal Strings. ....	76
Code 1.4-40: Using the PARSE Keyword Instruction with an Expression. ....	77
Code 1.4-41: Using the PARSE Keyword Instruction with Expressions. ....	77
Code 1.4-42: Using the PARSE Keyword Instruction with Positions and Lengths. ....	78
Code 1.4-43: Using the PARSE Keyword Instruction with Absolute Positions. ....	79

## List of Codes (Nutshell Examples)

Code 1.4-44: Using the PARSE Keyword Instruction with Relative Positions. ....	79
Code 1.4-45: Using the PARSE Keyword Instruction with Relative Positions. ....	80
Code 1.4-46: Using the PARSE VALUE Keyword Instruction. ....	80
Code 1.4-47: Using the PARSE VALUE Keyword Instruction with Parentheses. ....	81
Code 1.4-48: Using the PARSE ARG Keyword Instruction with Parentheses. ....	81
Code 1.4-49: Using the PARSE PULL Keyword Instruction. ....	82
Code 1.4-50: Using the PARSE SOURCE Keyword Instruction. ....	83
Code 1.4-51: Using the PARSE VERSION Keyword Instruction. ....	83
Code 1.4-52: Using the TRACE Keyword Instruction. ....	84
Code 1.4-53: Using the INTERPRET Keyword Instruction. ....	85
Code 1.4-54: Demonstrating a Command Instruction. ....	86
Code 1.4-55: Demonstrating Command Instructions to the THE Editor. ....	86
Code 2.1-1: Using the USE Keyword Instruction. ....	88
Code 2.1-2: Using the USE Keyword Instruction with the strict Subkeyword. ....	89
Code 2.2-1: Trapping a SYNTAX Condition. ....	91
Code 2.2-2: Trapping a NOVALUE Condition. ....	92
Code 2.2-3: Using the RAISE Keyword Instruction. ....	92
Code 2.2-4: Using the RAISE Keyword Instruction and Trapping the Condition. ....	93
Code 2.3-1: Demonstrating the Routine Directive. ....	94
Code 2.3-2: Routine Directive and Trapping User Defined Conditions. ....	95
Code 2.3-3: Routine Directive and Trapping User Defined Conditions. ....	96
Code 2.3-4: p1.rex calls p2.rex. ....	97
Code 2.3-5: p2.rex calls p3.rex. ....	97
Code 2.3-6: p3.rex calls p4.rex. ....	97
Code 2.3-7: p4.rex. ....	98
Code 2.3-8: p1.rex requires p2.rex. ....	99
Code 2.3-9: p2.rex requires p3.rex. ....	99
Code 2.3-10: p3.rex requires p4.rex. ....	99
Code 2.3-11: p4.rex. ....	100
Code 3-1: Mixing String Functions and Message instructions. ....	106
Code 3.1-1: The Runtime Environment and Environment Symbols. ....	109
Code 3.1-2: Sending Messages to Objects (Values, Instances). ....	110
Code 3.1-3: Using the Runtime Environment with Environment Symbols. ....	111
Code 3.1-4: Constructor and Destructor Method Routines. ....	112
Code 3.1-5: Resolving Methods Using the Class Hierarchy. ....	113
Code 3.1-6: Unknown Method Routine. ....	114
Code 3.2-1: Demonstrating Using Some Methods of .Object. ....	117
Code 3.2-2: Demonstrating Using Some Methods of .Class. ....	119
Code 3.2-3: Demonstrating Using Some Methods of .Method. ....	120
Code 3.2-4: Demonstrating Using Some Methods of .Message. ....	121

## List of Codes (Nutshell Examples)

Code 3.2-5: Demonstrating Using Some Methods of .Routine. ....	122
Code 3.2-6: Demonstrating Using Some Methods of .Package. ....	123
Code 3.2-7: Demonstrating Using Some Methods of .String. ....	125
Code 3.2-8: Using Stems the Classic Rexx Style. ....	125
Code 3.2-9: Intermixing Classic Rexx Stem Access and Methods of .Stem. ....	126
Code 3.2-10: Demonstrating Using Some Methods of .Stem. ....	126
Code 3.2-11: Demonstrating Using Some Methods of .Stream. ....	128
Code 3.2-12: Single Dimensioned .Array. ....	131
Code 3.2-13: Two-dimensional .Array. ....	132
Code 3.2-14: Sorting a Single Dimensioned .Array. ....	133
Code 3.2-15: Sorting a Single Dimensioned .Array With a Comparator. ....	133
Code 3.2-16: Using a .List to Collect and Process Objects. ....	134
Code 3.2-17: Using a .Queue to Collect and Process Objects. ....	135
Code 3.2-18: Using a .CircularQueue to Collect and Process Objects. ....	136
Code 3.2-19: Using a .Directory to Collect and Process Objects. ....	138
Code 3.2-20: Using the .local Directory to Collect and Process Objects. ....	139
Code 3.2-21: Using a .Relation to Collect and Process Objects. ....	141
Code 3.2-22: Using a .Table to Collect and Process Objects. ....	142
Code 3.2-23: Using a .Bag to Collect and Process Objects. ....	143
Code 3.2-24: Using a .Set to Collect and Process Objects. ....	144
Code 3.2-25: Setlike Operations with a .Bag and a .Bag. ....	147
Code 3.2-26: Setlike Operations with a .Set and a .Bag. ....	148
Code 3.2-27: Setlike Operations with a .Bag and a .Set. ....	149
Code 3.2-28: Demonstrating .Alarm. ....	150
Code 3.2-29: Demonstrating a Custom .Comparator for Sorting an Array. ....	151
Code 3.2-30: Demonstrating .DateTime and .TimeSpan. ....	153
Code 3.2-31: Demonstrating .File. ....	154
Code 3.2-32: Using the .output Monitor. ....	155
Code 3.2-33: Demonstrating .MutableBuffer. ....	156
Code 3.2-34: Using .context (a .RexxContext). ....	157
Code 4.1-1: Using a Java Object as If It Was an ooRexx Object. ....	163
Code 4.1-2: Using a Java Dialog as If It Was From an ooRexx Class. ....	164
Code 4.1-3: Using Java to Interact with Apache OpenOffice/LibreOffice. ....	165
Code 4.2-1: Using a Windows Object as If It Was an ooRexx Object. ....	173
Code 4.2-2: Using an Internet Explorer Object as If It Was an ooRexx Object. ....	174
Code 4.2-3: A Visual Basic Script (VBS) Program. ....	178
Code 4.2-4: An ooRexx Program Matching the Above VBS Program. ....	178
Code 4.2-5: A Visual Basic Application (VBA) Program. ....	179
Code 4.2-6: An ooRexx Program Matching the Above VBA Program. ....	179
Code 5.1-1: .methods Collecting Floating Methods. ....	187
Code 5.1-2: Using Messages. ....	188
Code 5.1-3: Using Cascading Messages. ....	189
Code 5.1-4: Demonstrating the Required String Value. ....	190
Code 5.1-5: Using a Constructor Method Routine. ....	191

## List of Codes (Nutshell Examples)

Code 5.1-6: Using a Destructor Method Routine. ....	192
Code 5.1-7: Using an UNKNOWN Method Routine. ....	193
Code 5.1-8: Using a STRING Method Routine. ....	194
Code 5.1-9: Using a MAKESTRING Method Routine. ....	195
Code 5.1-10: Using a MAKEARRAY Method Routine. ....	196
Code 5.1-11: Implementing a Comparison Method Routine named "="..	198
Code 5.1-12: Implementing a Method Routine named compareTo. ....	199
Code 5.1-13: Inheriting from Orderable and Implementing the Abstract Method compareTo. ....	200
Code 5.1-14: Implementing Comparator Method Routines. ....	202
Code 5.2-1: Implementing the ADT Birthday. ....	205
Code 5.2-2: Implementing the ADT Person. ....	205
Code 5.2-3: Implementing the ADT Birthday. ....	207
Code 5.2-4: Implementing the ADT Person. ....	207
Code 5.2-5: Get and Set Methods the ooRexx Interpreter Creates. ....	208
Code 5.2-6: Example for a Method Routine. ....	209
Code 5.2-7: Implementation of the ADT Birthday. ....	210
Code 5.2-8: Implementation of the ADT Person. ....	211
Code 5.2-9: Method Routine Accesses Attribute NAME Directly. ....	212
Code 5.2-10: self and super in Method Routines. ....	213
Code 5.2-11: Using the FORWARD Keyword Instruction. ....	214
Code 5.2-12: Demonstrating the ::CONSTANT Directive. ....	215
Code 5.2-13: Using a Class Hierarchy. ....	216
Code 5.2-14: Multiple Inheritance (AmphibianVehicle class specializes the RoadVehicle class and inherits from the WaterVehicle). ....	218
Code 5.2-15: Multiple Inheritance (AmphibianVehicle class specializes the WaterVehicle class and inherits from the RoadVehicle). ....	219
Code 5.3-1: Starting Multithreading with the REPLY Keyword Instruction. ....	221
Code 5.3-2: Starting Multithreading with the .Alarm. ....	222
Code 5.3-3: Starting Multithreading with .Object's start Method. ....	223
Code 5.3-4: Starting Multithreading with .Message's start Method. ....	224
Code 5.3-5: Synchronizing Threads with the GUARD Keyword Instruction. ....	227
Code 5.3-6: Waiting for Threads. ....	229
Code 5.3-7: Synchronizing Producer with Consumer. ....	232

## Index

Assignment Instruction .....	49
Control Variable .....	67
DROP .....	49
RC .....	52, 85
RESULT .....	53
self .....	53, 212
SIGL .....	53
Stem Variable .....	51
super .....	53, 212
VB (Visual Basic) .....	175
Rosetta Stone .....	175
VBA (Visual Basic for Applications) .....	175
Rosetta Stone .....	175
VBS (Visual Basic Script) .....	175
Rosetta Stone .....	175

---

## W

WHEN .....	65, 225
GUARD OFF .....	225
GUARD ON .....	225
SELECT .....	65
WHILE .....	69, 72
DO, LOOP .....	69
Windows .....	169
ActiveX .....	170
COM .....	170
MenuObject Class .....	182
OLE .....	170
OLEObject Class .....	171
OLEVariant .....	182
ooDialog .....	181
WindowObject .....	182
WindowsClipboard Class .....	182
WindowsManager Class .....	182
WindowsProgramManager Class .....	182

---

## :

::ATTRIBUTE Directive .....	207
::CLASS Directive .....	186, 205
::CONSTANT Directive .....	186, 214
::METHOD Directive .....	186, 207
::OPTIONS Directive .....	186
::REQUIRES Directive .....	98, 186
::ROUTINE Directive .....	93, 186

---

## .

.context .....	108
----------------	-----

.endofline .....	107
.environment .....	31
Entries for the ooRexx Built-in Classes .....	107
.endofline .....	107
.environment (Global Environment Directory) .....	107
.false .....	107
.local (Local Environment Directory) .....	107
.nil .....	107
.true .....	107
.error .....	107
.false .....	31, 107
.input .....	107
.line .....	108
.local .....	31, 107
.error .....	107
.input .....	107
.output .....	107
.stderr .....	107
.stdin .....	107
.stdout .....	107
.stdque .....	107
.methods .....	108, 187
.nil .....	31, 107
The NIL object .....	107
.output .....	107
.routines .....	108
.rs .....	108
.stderr .....	107
.stdin .....	107
.stdout .....	107
.stdque .....	107
.true .....	31, 107

**~**

~ .....	32, 59
Message Instruction .....	59
Operator .....	32
~~ .....	32, 59
Cascading Message .....	59
Message Instruction .....	59
Operator .....	32

# Some oo|Rexx-Related World-Wide-Web Links

## AUTHOR's ooREXX RELATED HOMEPAGE

Latest information on ooRexx books, projects, updates, seminars, news: <http://www.RonyRexx.net>

## REXX LANGUAGE ASSOCIATION (REXXLA)

Not-for-profit, special interest group (SIG), owner of *BSF4ooREXX*, *NETREXX* and *ooREXX*, organizer of the annual “International REXX Symposium”: <http://www.RexxLA.org>, <http://www.rexxla.org/events/>

## ooREXX

Homepage: <http://www.ooRexx.org>

Download: <http://sourceforge.net/projects/oorexx/files/oorexx/rexxref.pdf>: <http://www.oorexx.org/docs/rexxref/rexxref.pdf>, ooRexx PDF reference documentation

HTML reference : <http://www.oorexx.org/docs/rexxref/book1.htm>

## BSF4OOREXX (Camouflages Java as ooRexx)

Homepage: <http://sourceforge.net/projects/bsf4oorexx/>

Download: <http://sourceforge.net/projects/bsf4oorexx/files/GA/>

## Further information about the REXX family:

Mark Hessling's REXX Function Packages and the Regina REXX Interpreter: <http://www.Rexx.org>

Howard Fosdick's REXX Information: <http://www.RexxInfo.org>

## Some REXX Aware Editors:

*THE (THE HESSLING EDITOR)*, uses REXX and ooRexx as its macro language: <http://hessling-editor.sourceforge.net/>

*VIM (VI IMPROVED*, available for practically all operating systems), includes REXX and ooRexx syntax highlighting: <http://www.vim.org>

## Student's Work Related to REXX and BSF4[oo]REXX including AOO/OOo/LO:

<http://wi.wu.ac.at/rgf/diplomarbeiten/>

*Rony G. Flatscher* works as a professor for Information Systems (German: "Wirtschaftsinformatik") at the *WU Wien*, Austria, a business university with approximately 25,000 students.

He has been experimenting over a decade teaching programming to end-users of information systems using various programming languages. In the course of time, a lecture of two consecutive classes was developed which successfully introduces the students to programming, object-oriented programming and scripting/remote-controlling business applications such as Microsoft Office or (in an operating system independent manner) Apache OpenOffice.

One key element that allows for this to be done in a very short time is the choice of programming language: "*Open Object Rexx (ooRexx)*", a human centric, easy to understand and easy to use programming language that originates in the IBM product "*Object REXX*" which was handed over to the open-source community.

This book introduces the principles of the programming language ooRexx in a very concise manner. It demonstrates all of the introduced concepts immediately with "*nutshell examples*" (very small programs) showing their output, which allows anyone to study and learn the language *by just reading* this introductory book.

---

Author's ooRexx related URL: <http://www.RonyRexx.net>

---

*Howard Fosdick (USA)*, the author of "*Rexx – Programmer's Reference*" says about this book:

*"Excellent work! This is the book we all have been waiting for. I only wish it had been available when I was learning ooRexx!"*

...

*I'd like to conclude by emphasizing that this just is a really fantastic book, Rony, written by the one person who is so well qualified to write it. Congratulations!"*