EMPLOYING OBJECT REXX FOR TEACHING MBA STUDENTS THE OO-PARADIGM

Rony G. Flatscher

Department of Management and Information Systems

Vienna University of Economics and Business Administration

"10th International REXX Symposium“, Jacksonville, Florida, U.S.A., May 3rd-5th, 1999

ABSTRACT

At the WU Vienna - with over 20,000 students one of the largest Economics and Business Administration Universities in the world - an experiment was started in the summer semester of 1999 to teach MBA Students the OO-paradigm with the help of Object REXX. It has been very well known that taught material is learned the best, if the concepts are worked out by the people themselves. Therefore it may make sense to teach OO-concepts to MBA students by the means of exercises using an object-oriented programming language. Object REXX was chosen because of its simple syntax which draws from Mike F. Cowlishaw's original work on procedural REXX, while at the same time implementing a very powerful OO-model. Object REXX is available on a variety of operating systems, namely AIX, Linux, OS/2 and Windows95/98/NT.

This article introduces a syllabus for teaching MBA students the OO-paradigm with the help of Object REXX, thereby introducing the key features of the language itself and reports about preliminary conclusions.

.

1 INTRODUCTION

The "Wirtschaftsuniversität Wien" (abbreviated: WU, English translation: "University of Economics and Business Administration"; cf. [W3WU] for an English set of foils giving an additional overview of the University) is located in Vienna (Austria). It was founded as a Business School in 1899 and hosts over 20,000 MBA students in 1999.

Of these students 51% study Business Administration, 36% Commerce, 8% Economics and 5% Business Education (cf. foil 20 at [W3WU]). These studies are divided up into two parts, each lasting formally two years (four semesters)[1)], yielding the MBA degree.

1.1 The Study of "Business Administration"

The Business Administration part I studies last at least four semesters and consist of the following:

- Business Administration (15-16 hours),
- Economics (13 hours),
- Private Law (8-12 hours),
- Mathematics and Statistics (8-12 hours),
- Foreign Language (8 hours),
- Foreign Language II (12 hours) and
- Sociology (8-12 hours).

The part I studies are concluded with a series of comprehensive tests, which entitle the students to enter the second and final part of the studies. The Business Administration part II studies last at least another four semesters and consist of the following:

- General Business Administration (13 hours),
- First special field of Business Administration (12 hours),

1) The teaching system is *not* class-based and the students are usually free to choose the number and mix of courses freely. As many of the students have to work besides their studies the average time of study in order to arrive at the MBA is in effect more than fourteen semesters; the drop-out rate tops 60%!

- Second special field of Business Administration (12 hours),
- Elective (8 hours),
- Economics (10 hours) and
- Public Law (8 hours).

While conducting the part II studies the student has to work on a Master Thesis, usually needing six months up to several years.[2)] Two special fields of Business Administration have to be chosen out of a set of 23 subjects (cf. foil 14 at [W3WU]), of which "Management Information System" (abbreviated: MIS) is the one this article concentrates about.

Formally, for a special field of Business Administration there are a total of 12 hours of teaching available, eight for lectures, two for a pro-seminar and two for a seminar. After attending all classes successfully a written (4 hours of duration) and an oral comprehensive test have to be mastered by the students.

1.2 Business Administration "Management Information Systems"

The special Business Administration "Management Information Systems" consists of the following classes:

- "Electronic Commerce" (lecture, 2 hours),
- "Electronic Financial Services" (lecture, 2 hours),
- Optional workshop "Business Process Modelling" (lecture, 2 hours), knowledge needed anyway for the pro-seminar,
- Seminar (2 hours) with variable contents reflecting the changes in the field.

Choice of a lecture (2 hours) and the optional appropriate workshop (2 hours), knowledge needed anyway for the pro-seminar:

- "Introduction to Solving Problems with Enduser Tools",
- "Introduction to Developing Information Systems with CASE".

[2)] The time-frame for working on the Master Thesis depends very much whether a student has to work for a living or is able to exclusively concentrate on the studies.

Choice of the appropriate pro-seminar (2 hours):

- "Introduction to Solving Problems with Enduser Tools",
- "Introduction to Developing Information Systems with CASE".

Choice of at least two classes (each 2 hours) from the following pool, which are integral parts of the comprehensive tests, but the students are allowed to gather the necessary knowledge via other means as well:

- "Computer Law",
- "Electronic Money, Payment Systems and Security",
- "Information Systems in Finance and Accounting",
- "IT Market and Information Management from the Viepoints of Suppliers and Consumers",
- *"Introduction to Procedural and Object-oriented Programming (Object REXX)"*, held for the first time in the summer-semester of 1999.

1.3 Need for Introducing the Object-oriented Paradigm

The need for a course which introduces students to the object-oriented paradigm in the context of the MIS-studies was seen for various reasons, the most important ones being object-oriented development trends which are relevant to assessing and/or choosing business applications,

- especially in enterprise resource programs (abbreviated: "ERP"), which start to employ (and to introduce) terms like "Business objects", "Business components", "Patterns", "Frameworks" and the like,
- the object-oriented standards which the "Object Management Group" (abbreviated: "OMG", cf. [W3OMG]) has been developing and which have been introduced into the market place by many companies,
- the operating system Windows through which Microsoft has been introducing and exposing the users to object-based and lately, to object-oriented technology.[3)]

[3)] Witness the Microsoft technologies "Object Linking and Embedding" (OLE) respectively the newer

Taking these developments into account it becomes clear that MBA students face quite a few problems with the advent of object-oriented technology in the realm of the business world: they have no working knowledge of the fundamental concepts of the object-oriented paradigm like “classes”, “objects”, “messages”, “methods”, “inheritance of properties” and the like. As a result they have no ability to fully evaluate and assess object-oriented based business applications, nor a working knowledge for analyzing and devising object-oriented models.[4)]

2 MANDATORY OBJECT-ORIENTED CONCEPTS

This section introduces the mandatory object-oriented (OO) concepts which MBA students should learn through this class. Following that subsection a discussion will follow as to why Object REXX was chosen for this particular class and how its object-oriented features allow for teaching the mandaory OO concepts.

2.1 Defining Mandatory Object-Oriented Concepts

Before setting out a syllabus for a new student class, it is important to define a set of mandatory object-oriented concepts, which should be taught in order for teaching the MBA students the conceptual knowledge which allows them to at least assess the latest business software offerings at a conceptual level.[5)] In general the aspect of re-use and mastering of complexity by breaking complex problems up into more manageable sub-problems. In this context subproblems could get solved with an object-oriented, pluggable component architecture approach, benefitting from inheriting tailored and

versions being dubbed “ActiveX”, employing the Microsoft “Component Object Model” (COM). This has been more visible to end users who have been using Microsoft’s Visual Basic for scripting the Microsoft Office parts, like Microsoft Excel or Microsoft Word. Ultimately, with the popularization of the “Windows Scripting Host” (WSH) every Microsoft Windows power user will get exposed to the object-oriented paradigm one way or the other.

4) Interestingly, although these MBA students have to go through mandatory assignments in Extended-Entity-Relationship modelling (EERM), which include classification trees, they feel according to discussions, that they have no - not even conceptual (!) - knowledge whatsoever with respect to object-oriented modelling (OOM). It may also be interesting to note in this context that in OMG’s UML there is no modelling language available for devising conceptual datamodels (which then may get mapped to more concrete class diagrams).

5) In addition it must be clear that most of the MBA students have never been exposed to programming, so it is also necessary to teach the basic structural programming concepts beforehand, including the reuse of code by defining callable procedures and functions.

tested pluggable components. Discussing the problems with collegues who have been teaching MIS classes the following object-oriented concepts were chosen to be of utmost importance and therefore should get taught:

- “Class”: this should be explained with the concept of an “Abstract Data Type”, which defines common properties (attributes and behaviour in form of functions/procedures resp. methods). Classes are usually organized in the form of a rooted “Classification Tree”, which can be employed for classifying objects, as well as determining the generalization and specialization of classes with respect to each other, thereby introducing the important concept of “inheritance”.[6)]

- “Objects”, “Instances”: instantiation of classes and the lifetime of such objects should be understood, especially the “creation” (constructing) and the “destroying” (de-constructing) of objects.

- Sending of “Messages”: the communication in an object-oriented system is of utmost importance as well as the process of de-coupling direct function/procedure-calls into message issuing systems, where conceptually the addressed objects themselves lookup and invoke the “methods” in response to received messages. In this part the ramification of the inheritance concept should be made clear.

- Advanced concepts “metaclass” and “concurrency”: metaclasses allow for managing classes and introspecting them at runtime, which is very important in dynamic systems, which may define and alter classes at runtime or which need to adjust themselves to changes in the “environment” they are running. In addition the concept of “concurrency” which allows for implementing powerful solutions to business problems (like simulation, concurrently serving requests e.g. from the internet etc.).

---

6) Mulitple inheritance was regarded to be important, but not a mandatory object-oriented concept to be fully understood by MBA students, realizing that there are important programming languages like Java, which do not offer that particular feature at all. If possible, the students should be made aware of problems which are easily solved, if multiple inheritance is in place and which need to be solved somehow in other systems not offering this particular feature.

2.2 Object REXX

Object REXX has been developed by IBM following a request of IBM's largest user group in the world, SHARE. It was introduced as part of OS/2 Warp, release 4, in 1997 for the first time. Object REXX for Linux, Windows 95, Windows 98 and Windows/NT were released a year later, and finally IBM created a version for its Unix operating system AIX and made it available in 1999.

Object REXX is fully compatible to IBM's REXX[7)] which was used for creating an ANSI REXX standard in 1997. The author of REXX, IBM fellow Mike F. Cowlishaw, intended to create a "human friendly" programming language [Cowl94], with an easy to learn syntax and a firm set of built in functions, of which an important part makes it very easy to manipulate text. It has since become IBM's "System Application Architecture" (SAA) procedural language of choice and has been therefore ported to all IBM operating systems.

Object REXX builds on the same philosophy as REXX, i.e. easy syntax and a minimal, but extremely useful implementation of functions and classes. Its object-model seems to be derived from Smalltalk's, extended with the ability of multiple-inheritance. Compared to REXX it introduced a new operator "~" (called "twiddle"), which serves as the message operator, an "environment" which can be used to store objects and which allows for coupling Object REXX programs executing within the same process space (concurrently, if desired), "arguments by reference" and the ability to create and require packages containing public "classes" and "routines".

These are some of Object REXX main features:

- Backward compatible with "classic" REXX, yet implementd internally in a full object-oriented manner.[8)]
- Abstract data types can be easily implemented with the "CLASS"- and "METHOD"-directives, including attributes.
- There is a pre-fabricated, extensible classification tree available, supplying e.g. the fundamental classes "Object", "Class", "Method" and "Message", collection

7) There exist numerous commercial as well as open source and free implementations of REXX [W3REXXLA].

8) "Classic" REXX statements are transformed internally to their OO equivalents.

classes (“Array”, “Bag”, “Directory”, “List”, “Queue”, “Relation”, “Set”, “Table”) and miscellaneous, handy other classes (e.g. “Alarm”, “Monitor”, “Stream”, “String”).

- Object REXX allows to employ multiple inheritance.
- Constructor (method named “INIT”) and destructor (method named “UNINIT”) methods can be defined, helping tailoring the creation and destruction of objects.
- Interception of messages for which no implemented methods exist via the definition of an “UNKNOWN” message in the class (or one of its superclasses).
- It is possible to define metaclasses and to create class[9)] methods.
- Object REXX allows for concurrent execution of methods, including standard support for inter-object concurrency as well as *intra*[10)]-object concurrency.

From this brief list of Object REXX features it becomes evident, that it would be possible to use this programming language to demonstrate and excercise object-oriented concepts taught to MBA students. The syntax is very easy to understand and hence to learn, Object REXX programs can be read as if they were pure pseudo code, i.e. without any need to explain syntax rules in detail.[11)] This allows students to re-think concepts taught in theory by turning to some easy to understand code and re-iterating what was introduced. In addition, should a student wish, it would be possible to even learn a true and powerful programming language in the process of attending a class employing Object REXX for its programming examples.

One last remark regarding the missing pre-knowledge of MBA students with respect to programming: exposing them to the object-oriented paradigm, it is *very* important to carefully choose examples for demonstrating the OO-concepts which can be understood easily by them.[12)]

---

9) Following the Smalltalk model a “class method” is merely an instance method of a metaclass, which gets used for creating class objects.

10) Methods executing concurrently for the very *same* instance of a class.

11) There are only two hours per week available for this class for MBA students. Therefore it is extremely important, that it does not become necessary to teach and explain syntax issues in detail, in order for the students to understand the code as is the case with C++ , Java, JavaScript, Perl, TcL or Visual Basic for that matter.

12) The author sometimes wonders about examples demonstrating specific features of programming languages, how complicated they become sometimes, as if the respective authors wished to demonstrate the power of their preferred language by solving most complex problems.

3 A SYLLABUS FOR TEACHING THE OO CONCEPTS

In this section the carefully devised syllabus for introducing MBA students to the object-oriented concepts is introduced. The individual installments, each lasting 90 minutes, follow the pattern:

- introduction of the theory,
- explaining an adequate problem and
- the formulation of a possible solution for the problem employing the just learned concepts, thereby effectively repeating the theory in a little different manner.

This is followed by voluntary assignments reiterating the taught concepts at the end of the class.

As the MBA students cannot be expected to have been exposed to programming concepts, it is necessary to introduce them to structured (procedural) programming concepts, before turning to the core of the course: the teaching of the object-oriented paradigm. It is also expected, that especially in the part where the object-oriented concepts are introduced, the students need some repetitions in order to learn the vocabulary in a form which does not alienate them of the OO concepts denominated by the numerous new “termini technicii”.

3.1 Class 1: Overview, Introduction

The MBA students learn about the history of REXX, Object REXX and a REXX-like version of Java, called NetRexx. The intention of the author to create a “human centric” [Cowl94] is explained and demonstrated with syntax examples, pointing at the pseudo-code like look of REXX programs.

3.2 Class 2: Introducing the Building Stones

The MBA students learn about “literals”, “variables”, “constants”, “comments” and finally about “statements”, defining a “block” of statements, “conditional branch” and “iteration” (“looping”). All of these concepts are demonstrated with simple examples implemented in the form of short REXX programs, which look like pseudo-code.

3.3 Class 3: Creating Procedures and Functions, Introducing Scopes

This class stresses the re-usage of code by moving commonly used regions of code into a procedure block. First the term and the definition of “label” is introduced, the possibility to jump to labels and to return from them. In addition the students learn about the concept of “scope” in this context.

3.4 Class 4: Built-in Functions (BIF) and Utility Functions

The MBA students learn about the full set of built-in functions (BIF) of REXX, which is possible, as the authors of REXX - and the ANSI committee standardizing it - intentionally kept it as small as possible. The BIFs are introduced in a categorized manner. In addition operating system and non-language dependent functions available through the “REXXutil” function package are hinted at, whithout exposing the students to them. They should just learn about extensions to programming language with the means of function packages, introducing them to class libraries and the like in this context.

3.5 Class 5, 6: Exceptions, Routines

This installment concludes the introduction to the fundamental procedural programming concepts by explaining and teaching the concepts of “exceptions” and the ability to intercepti (“catch”) them. This class will also introduce the MBA student to the non-OO scopes introduced by Object REXX, namely the ability to define packages with public routines and the resolution of required packages.

Class 6 merely repeats all the concepts taught so far, introduces examples and their possible solutions, reiterating them in detail. This repetition concludes the introduction of the concepts needed for structured programming.

3.6 Class 7: Introducing “Abstract Data Type”, “Class”, “Object”

Starting out with the term “datatype”, which the MBA students should have learned attending the introduction to MIS lecture in the first part of their studies, “abstract data types” (ADT) are introduced. Using this approach should allow the students to refer to

known concepts and ease the understanding of what happens, if an ADT is being implemented with defining a “Class” representing it. This should also make it clear, that the properties (attributes and methods) of a class replicate those of the ADT. The creation of “instances”, i.e. “objects”, from a class should as a result of choosing this route of explanation be comprehensible by the students.

The interaction among objects is done via sending messages to objects using the message operator “~” (a tilde, also known as “twiddle” in Object REXX). The students are briefly exposed to “cascading messages” which will be re-iterated in later classes. This way they are able to associate both with message sending. This class concludes with discussing the lifetime of an object, from its creation until its destruction, introducing the conceps of constructors (methods named “INIT”) and destructors (methods named “UNINIT”).

3.7 Class 8: Specializing, Inheritance, Multi-threading, Scopes

This class starts out repeating all the newly introduced concepts and terms of class 7. Then it proceeds with the presentation of the “inheritance” concept, which plays an important role if one specializes existing classes, followed by inducting the term “classification tree” (short: “class tree”) and pointing at the fact that practically all object-oriented programming languages (with the notable exception of C++) and systems define such a classification tree. The classification tree is being used to explain the rules applied to resolving methods by the object which receives a message. In this context multi-threading and scopes needed for deterministic execution of concurrent methods are made aware to the students and are discussed briefly.

3.8 Classes 9, 10, 11: Method Resolution, Classification Tree

As Object REXX comes with a minimal, yet very powerful set of pre-fabricated classes, they are introduced and classified. Numerous little examples demonstrate the features made available with them. The “clustering” (classifying of classes) is as follows:

- “Fundamental Classes”: “Object”, “Class”, “Method” and “Message”. These classes allow the runtime system to create classes and methods according to the given definitions in the form of directives, instantiate the classes and manage the communication of messages among the objects.

- “Miscellaneous Classes”: “Alarm”, “Monitor”. These classes allow the runtime system (and programmers who are aware of their behaviour) to maintain various important aspects of the Object REXX interpreter.

- “Classic REXX Classes”: “String”, “Stem”, “Stream”. These classes are available in Object REXX for allowing the interpreter to remain backwardly compatible with “classic” REXX programs, yet be able to internally carry out all statements and functions in an object-oriented manner.

- “Collection Classes” are divided in the types with externally (predefined) indices (“Array”, “List”, “Queue”) and those where programmers are able to choose values for the indices (“Bag”, “Directory”, “Relation”, “Set”, “Table”). As these classes can help solve problems in innovative ways, quite some time is spent explaining these classes in detail.

3.9 Class 12: Class Methods, Metaclasses, One-off Objects

Object REXX - similiar to Smalltalk - allows for creating and altering metaclasses, which are used in the process of creating class objects. The relationship between classes, class objects, as well as instance methods and class methods are explained. The students should understand the concepts by learning how the Object REXX runtime system carries out class- and method-directives at runtime. Every student should be able to understand, that in Object REXX a class possesses instance methods only, and that what is dubbed a “class method” in effect is an instance method of the appropriate metaclass.

The concept “metaclass” and the ability to define specializations of metaclasses are further discussed by first explaining the term “design pattern” (GoF, “Gang of Four”, [GHJV95]), and then by introducing the documented factory patterns “Singleton” and “Manager”, their possible implementations and their ramifications. This advanced installment concludes with presenting the concept of “one-off-objects”, which are instances of the same class, yet differ in some detail, represented mostly by the fact that they possess one or more different implementations of methods.

In order for the students to be able to put the concepts they have learned so far together and to understand how they relate to each other, a “big picture” is presented,

starting out with what happens when a program gets loaded and how it gets started, including the definition and instantiation of classes.

3.10 Class 13: Environments

Interpreted object-oriented languages like Smalltalk or Object REXX store runtime information in a directory, which is called “environment”. This class extensively discusses this concept, and introduces all of the environments made available by Object REXX, giving examples how programs can be coupled via them by applying them to exchange information, i.e. objects.

3.11 Class 14: Object REXX Based Utilities

This class introduces the students to various recurring problems in the context of creating object-oriented programs and describes a set of solutions for them, which are realized in the form of short Object REXX programs themselves. Studying the classes and routines of these utilities [Flat96c, Flat97] by looking up the source, helps to further a deeper understanding of the OO concepts being at ones disposal.

3.12 Class 15: Concurrency Issues

Although the students were exposed briefly to concurrency issues in object-oriented systems (cf. “Class 8” above), this installment repeats what was learned and discusses the concepts in further detail. As creating concurrently executing (multi-threaded) methods is very easy in Object REXX, this class attempts to mediate the power of this concept.

3.13 Class 16: Advanced Items (Security Manager, D/SOM, OLE)

In order to enhance the applicability of a programming environment, the controlled execution of programs retrieved from untrusted sources like the WWW is very important. Some execution environments like Java employ the concept of a security manager which supervises the execution of programs. This concept is introduced and demonstrated to the MBA students with Object REXX’ security manager, which allows to implement security policies tailored to the needs of any business.

Finally, the course concludes with a brief discussion of how an object-oriented programming system like Object REXX can plug into independently developed object-oriented programming systems created either with IBM's "System Object Model" (SOM), respectively OMG's "Common Object Request Broker Architecture" (CORBA) or Microsoft's "Object Linking and Embedding" (OLE). The discussion should lead the MBA students to become aware of the possibilities which arise, if it becomes possible to communicate with independently developed OO systems as well as the risks which may be incurred by them.

4 PRELIMINARY CONCLUSIONS

As of the writing of this article (1999-04-30) the first 11 lectures took place already. At the beginning of each class the content of the previous one is repeated and students are able to place questions and to raise discussions. On the other hand, the author has been actively querying the MBA students to find out, whether they really are able to comprehend all of the taught concepts. These are some of the preliminary conclusions the author has been able to draw so far:

- It seems that the MBA students have understood all of the introduced new concepts, *including* multiple inheritance.
- The students had *no* problems whatsoever to understand the examples presented in the Object REXX syntax. The examples can truly be read as if they were formulated in an easy to understand pseudo-code style.
- It seems possible to teach MBA students the fundamental procedural (structured programming) and object-oriented *concepts* in a lecture of two hours per week only.
- Object REXX seems to be ideally suited for crafting example code and teaching the concepts also with the means of reiteration in the form of applied short programs. This is attributed by the author to mainly three features of Object REXX compared to e.g. with C++, Java, JavaScript, Perl or Visual Basic: extremely simple syntax, powerful OO-model and masterable with respect to the amount (and features) of the built-in classes.
- Preparing the classes is *extremely* time consuming at a conceptual level (introducing which concepts in what sequence), as well as at devising examples and

excercises, which are restricted to stress the newly taught concepts *only*. I.e. taking great care not to conceive difficult to understand problems for the uninitiated.[13)]

5 REFERENCES

[Cowl84] Cowlishaw M.: "The design of the REXX language", IBM Systems Journal, Volume 23, No. 4 (1984).

[Cowl90] Cowlishaw M.: "The REXX Language", Prentice Hall (second edition), 1990.

[Cowl94] Cowlishaw M.: "The Early History of REXX", IEEE Annals of the History of Computing, Vol 16, No. 4, Winter 1994, pp15-24.

[Ende97] Ender T.: "Object-Oriented Programming with REXX", John Wiley & Sons, New York et.al. 1997.

[Flat96a] Flatscher R.G.: "Local Environment and Scopes in Object REXX", in: Proceedings of the "7th International REXX Symposium, May 12-15, Texas/Austin 1996", The REXX Language Association, Raleigh N.C. 1996.

[Flat96b] Flatscher R.G.: "Object Classes, Meta Classes and Method Resolution in Object REXX", in: Proceedings of the "7th International REXX Symposium, May 12-15, Texas/Austin 1996", The REXX Language Association, Raleigh N.C. 1996.

[Flat96c] Flatscher R.G.: "ORX_ANALYZE.CMD - a Program for Analyzing Directives and Signatures of Object REXX Programs", in: Proceedings of the "7th International REXX Symposium, May 12-15, Texas/Austin 1996", The REXX Language Association, Raleigh N.C. 1996.

[Flat97] Flatscher R.G.: "Utility Routines and Utility Classes for Object REXX, Part II", in: Proceedings of the "8th International REXX Symposium, April 22nd-24th, Heidelberg/Germany 1997", The REXX Language Association, Raleigh N.C. 1997.

[13)] The author estimates that for one unit of lecture, he needed ten hours of preparation! Altogether this particular class caused a time investment of approximately 320 hours, equivalent to two (!) Man months.

[GHJV95] Gamm E., Helm R., Johnson R., Vlissides J.: "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley 1995.

[TurWah97] Turton T., Wahli U.: "Object REXX for OS/2 Warp", Prentice-Hall, London 1997.

[VeTrUr96] Veneskey G., Trosky W., Urbaniak J.: "Object REXX by Example", Aviar, Pittsburgh 1996.

[WahHolTur97] Wahli U., Holder I., Turton T.: "Object REXX for Windows 95/NT, With OODialog", Prentice Hall, London 1997.

[W3C] Homepage of the World Wide Web Consortium, URL (1999-04-30): `http://www.w3c.org`

[W3ClaEx] Class: exercises (in German), URL (1999-04-30): `http://wwwi.wu-wien.ac.at/Studium/LVA-Unterlagen/poolv/1999s/poolv_aufgaben.htm`

[W3ClaPgm] Class: sample solutions to [W3ClaEx], commented in English, URL (1999-04-30): `http://wwwi.wu-wien.ac.at/Studium/LVA-Unterlagen/poolv/1999s/pgm`

[W3ClaWin] Class: sample for demonstrating a Euro-calculator implemented in Object REXX with IBM's GUI-development product "OODialog", commented in English, URL (1999-04-30): `http://wwwi.wu-wien.ac.at/Studium/LVA-Unterlagen/poolv/1999s/oodialog/info.htm`

[W3Hobbes] Homepage of the ftp-server "hobbes", containing Object REXX archives, URL (1999-04-25): `http://hobbes.nmsu.edu/pub/os2/dev/oREXX`

[W3ObjREXX] Homepage of Object REXX, URL (1999-04-30): `http://service.software.ibm.com/dl/REXX/oREXX-d`

`http://www2.hursley.ibm.com/oREXX/`

[W3OMG] Homepage of the Object Management Group, URL (1999-04-30): `http://www.omg.org/`

[W3REXX] Homepage of REXX, URL (1999-04-30): `http://www2.hursley.ibm.com/REXX/`

[W3REXXLA] Homepage of the REXX Language Association, URL (1999-04-30): `http://www.REXXLA.org`

[W3WU] English set of Foils introducing the Wirtschaftsuniversität to the English speaking audience, URL (1999-04-30):

`http://www.wu-wien.ac.at/rektorat/Kenndaten99_E/KeyData.html`

Date of Article: 1999-04-30.

Published in: Proceedings of the "10th International REXX Symposium“, Jacksonville, Florida, U.S.A., May 3rd-5th, 1999. The REXX Language Association, Raleigh N.C. 1999.

Presented at: "10th International REXX Symposium“, Jacksonville, Florida, U.S.A., May 3rd-5th, 1999.