# Preface

NetRexx is a new general-purpose programming language inspired by two very different programming languages, Rexx and Java™. It is designed for people, not computers. In this respect it follows Rexx closely, with many of the concepts and most of the syntax taken directly from Rexx or its follow-on, Object Rexx. From Java it derives static typing, binary arithmetic, the object model, and exception handling.

The first implementation of NetRexx produces classes for the Java Virtual Machine, and in so doing demonstrates the value of that concrete interface between language and machine: NetRexx classes and Java classes are entirely equivalent – NetRexx can use any Java class (and vice versa) and inherits the portability and robustness of the Java environment.

When I started designing NetRexx, it was an experiment: there was no guarantee that an 'Object Rexx for the Java environment' was viable. Critical areas of the design – notably the syntax for referring to types, and automatic conversions – took many iterations to get right, followed later by extensive tuning in response to feedback from the early users of the language. The result is a single language that not only provides the scripting capabilities and decimal arithmetic of Rexx, but also seamlessly extends to large application development with fast binary arithmetic.

More than a decade ago, I wrote that I hoped that future programming languages would improve on Rexx, and that Rexx was just a start in the direction of languages designed for people rather than computers. Although disappointingly little research has been carried out in this area in the interim, it was encouraging to discover that the philosophy behind the Rexx design can be applied so completely to a very different class of language, one both object-oriented and statically typed.

I believe that NetRexx is indeed an improvement on Rexx. Although Rexx has been widely used for 'mission critical', and sometimes quite enormous, applications, its greatest strength was always in scripting and customizing applications. NetRexx supports these tasks equally well (indeed, some Rexx programs can be processed as NetRexx programs directly), and yet gains the scalability, power, and robustness of Java.

I had thought that programmers would always need two languages: one PL/I-like or C-like for high performance 'low level' programming, and one Rexx-like for prototyping, rapid development, and scripting. I now see that the future is rosier: designing a single language for these traditionally disparate uses is indeed possible. This truly makes programming easier than it was before.

## NetRexx on the World Wide Web

Many NetRexx resources can be found on the World Wide Web, including on-line documentation, sample programs, and the NetRexx reference implementation. See: `http://www2.hursley.ibm.com/netrexx`

## Acknowledgements

Much of NetRexx is based on earlier work, and I am indebted to the hundreds of people who contributed to the development of Rexx, Object Rexx, and Java.

In recent years I have gained many insights from the deliberations of the members of the X3J18 technical committee, which, under the remarkable chairmanship of Brian Marks, led to the 1996 ANSI Standard for Rexx. Many of the committee's suggestions are incorporated in NetRexx.

Equally important have been the comments and feedback from the pioneering users of NetRexx, and all those people who sent me comments on the language either directly or in the NetRexx mailing list or forum. I would especially like to thank Ian Brackenbury, Barry Feigenbaum, Davis Foulger, Norio Furukawa, Dion Gillard, Martin Lafaix, Max Marsiglietti, and Trevor Turton for their insightful comments and encouragement.

I also thank IBM; my appointment as an IBM Fellow made it possible to make NetRexx a reality in months, rather than years.

Finally, this book has relied on old but trusted technology for its creation: its GML markup was processed using macros originally written by Bob O'Hara, and formatted using SCRIPT/VS, the IBM Document Composition Facility. Geoff Bartlett provided critical advice on character sets and fonts.

*Mike Cowlishaw*