Object-Oriented Implementation of Numerical Methods

An Introduction with Java and Smalltalk

Object-Oriented Implementation of Numerical Methods

An Introduction with Java and Smalltalk

Didier H. Besset

DB Consulting

MORGAN KAUFMANN PUBLISHERS

AN IMPRINT OF ACADEMIC PRESS

A Harcourt Science and Technology Company

SAN FRANCISCO SAN DIEGO NEW YORK BOSTON

LONDON SYDNEY TOKYO

Acquisitions Editor Diane D. Cerra
Production Editor Howard Severson
Production Assistant Mei Levenson
Editorial Assistant Stacie Pierce
Cover Design Yvo Reizebos
Cover Image ???
Text Design, Composition, and Illustrations Windfall Software, using ZzTeX
Copyeditor Laura Larson
Proofreader Ken DellaPenta
Indexer Bill Meyers
Printer Courier Corporation

ACADEMIC PRESS
A Harcourt Science and Technology Company
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA
http://www.academicpress.com

Academic Press
Harcourt Place, 32 Jamestown Road, London, NW1 7BY, United Kingdom
http://www.academicpress.com

Morgan Kaufmann Publishers
340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205
http://www.mkp.com

Printed in the United States of America

05 04 03 02 01 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Besset, Didier H.
Object-oriented implementation of numerical methods : an introduction with Java and Smalltalk / Didier H. Besset
p. cm.
ISBN 1-55860-679-3
1. Computer software—Development. 2. Object-oriented programming (Computer science). I. Title.
QA76.76.D47 B475 2001
005.1'17—dc21 00-063399

This book is printed on acid-free paper.

Contents

Code Listings

Preface

Si je savais une chose utile à ma nation qui fût ruineuse à une autre,
je ne la proposerais pas à mon prince,
parce que je suis homme avant d'être Français,
parce que je suis nécessairement homme,
et que je ne suis Français que par hasard.[1]

—Charles de Montesquieux

When I first encountered object-oriented programming I immediately became highly enthusiastic about it, mainly because of my mathematical inclination. After all, I learned to use computers as a high-energy physicist. In mathematics, a new, high order concept is always based on previously defined, simpler concepts. Once a property is demonstrated for a given concept, it can be applied to any new concept sharing the same premises as the original one. In object-oriented language, this is called *reuse* and *inheritance*. Thus, numerical algorithms using mathematical concepts that can be mapped directly into objects.

This book is intended for object-oriented programmers who need to implement numerical methods in their applications. The algorithms exposed here are mostly fundamental numerical algorithms with a few advanced ones. The purpose of the book is to show that implementing these algorithms in an object-oriented language is feasible—and quite easily feasible. I expect readers to be able to implement their own favorite numerical algorithm after seeing the examples discussed in this book.

The scope of the book is limited. It is not a Bible about numerical algorithms. Such Bible-like books already exist and are quoted throughout the chapters. Instead, I wanted to illustrate mapping between mathematical concepts and computer objects. I have limited the book to algorithms, which I have implemented and used in real applications over 12 years of object-oriented programming. Thus, the reader can be certain that the algorithms have been tested in the field.

Because the book's intent is to show numerical methods to object-oriented programmers, the code presented here is described in depth. Each algorithm is presented with the same organization. First the necessary equations are introduced with short

1. If I knew some trade useful to my country but that would ruin another, I would not disclose it to my ruler, because I am a man before being French, because I belong to mankind while I am French only by a twist of fate.

explanations. This book is not about mathematics, so explanations are kept to a minimum. Then the general object-oriented architecture of the algorithm is presented. Finally, because this book is intended to be a practical one, the code implementation is exposed. First, I describe how to use it, for readers who are just interested in using the algorithm, and then I discuss and present the code implementation.

The question was which language to use?

I decided to select two languages: Smalltalk and Java. Java is an obvious choice today since the world seems to lean toward Java. In spite of all the current emphasis on Java, I personally prefer Smalltalk for any practical use. But, as the computer industry evolves, and because I am anything but a language fanatic, the reader must be aware that this last statement may no longer be true by the time these lines are read. New computer languages will certainly combine object-oriented features with novel concepts, and I am ready to try any serious new contender.

Having stated my preference for Smalltalk does not mean that I don't like Java. Java has brought forth interesting innovations and certainly paved a golden way for object-oriented programming. I personally have gained a lot of insight in learning Java.

The code is this book has been written in both languages and is presented in separate sections. My hope is that Smalltalk and Java programmers will be exposed to the other language by reading this book. If they keep an open mind and do indeed read both code sections, they certainly will gain something from the comparison as I did myself in writing them.

As far as possible, each algorithm is presented with examples of use. I did not want to build contrived examples and instead have used examples personally encountered in my professional life. Some people may think that some examples are coming from esoteric domains. This is not so. Each example has been selected for its generality. The reader should study each example regardless of the field of application and concentrate on the universal aspects of it.

Acknowledgments

I wish to express my thanks to the many people with whom I had interactions about the object-oriented approach—Smalltalk and Java in particular—on the various electronic forums. One special person is Kent Beck, whose controversial statements raised hell and started spirited discussions. I also thank Kent for showing me tricks about the Refactoring Browser and eXtreme Programming. I also would like to thank Eric Clayberg for pulling me out of a ditch more than once and for making such fantastic Smalltalk tools.

A special mention goes to Donald Knuth for being an inspiration to me and many other programmers with his series of books *The Art of Computer Programming* and for making this wonderful typesetting program TEX. This book was typeset with TEX, LATEX, and ZzTEX.

Furthermore, I would like to give credit to a few people without whom this book would never have been published. First, Joseph Pelrine persuaded me that what I was doing was worth sharing with the rest of the object-oriented community. I express

my most sincere thanks to the reviewers who toiled on the early manuscripts of this book. Without their open-mindedness, it never would have made it to the publisher.

Special thanks go to David N. Smith for triggering interesting thoughts about random number generators and to William Leo for checking the equations. Another special mention goes to An Lam, who did a terrific job at checking the orthodoxy of the Java code.

Finally, my immense gratitude is due to Stephane Ducasse of the University of Bern, who checked the orthodoxy of the Smalltalk code and did an excellent job of rendering the early manuscript into not only readable but entertaining prose.