

Web Site

We are posting materials relevant to this book at www.modelsoftcorp.com.

Acknowledgements

Over the years we have worked with a number of people who have taught us about their business, given us the opportunity to try our ideas, and helped us learn in the process. We would like to thank these many individuals, starting with our former management at the General Electric R&D Center and continuing to our recent business friends and clients.

Although only two of us have written this second edition, the first edition had three additional authors—Bill Premerlani, Fred Eddy, and Bill Lorensen. We thank them for their past contribution on which this second edition builds. We also thank them for their support and encouragement in our writing of this second edition.

Chris Kelsey had an important role in the second edition that deserves special mention. She is the primary author of Chapter 18 on OO programming languages and also was an active reviewer.

We are grateful to our other reviewers (Mikael Berndtsson, Peter Chang, Bill Premerlani, and John Putnam) for taking the time to read our manuscript and provide thoughtful criticism.

Finally we wish to thank our families and colleagues for being patient with our many distractions and diversions during the writing of this book.

Michael Blaha
Chesterfield, Missouri
blaha@computer.org

James Rumbaugh
Cupertino, California

In addition there are a number of books that present the concepts of the UML. This book is different than most in that it not only explains the concepts, but it also explains their fundamental purpose and shows how to use them to build software. We do not explain every concept and nuance, but we do strive to explain the core of UML—enough to help you learn how to use the UML to build better software.

Changes From the First Edition

It has been fourteen years since we completed the first edition of this book. In the meantime there have been many advances in technology, leading to many changes in this second edition.

- **Notation.** We have replaced the OMT notation with the UML notation, specifically UML 2.0. The UML is now the dominant and standard language for OO modeling.
- **Process.** The second edition adds more content to the software development process. We now distinguish between domain analysis and application analysis. We have added implementation modeling. By intent, we have kept the process simple and lightweight so that it is approachable to students. This book's process is a subset of heavyweight processes, such as IBM Rational's RUP.
- **The three models.** We have carried forward the first edition's focus on "the three models" because we believe such an emphasis is helpful for teaching and learning OO modeling. However, we dropped the functional model, because it was not as useful as we had expected. In its place, we added the interaction model to incorporate use cases and sequence diagrams and to give a more holistic understanding of behavior among several objects.
- **Software engineering.** Part 4 covers several important software engineering topics: iterative development, management of models, and treatment of legacy systems.
- **Programming languages.** Programming languages have changed dramatically over the past decade and a half. Smalltalk has faded, while C and Fortran have diminished in importance. C++ and Java are now the dominant OO programming languages, and we have focused on them accordingly.
- **Databases.** OO models provide a sound basis not only for programming code, but also for relational databases. This book has an entire chapter that shows how to build efficient, correct, and extensible databases from UML models.
- **Case studies.** When the first edition was published, we felt a need to justify OO technology, so we included several case studies. Today, many case studies are available in the literature, so we have eliminated them from this book.

In this second edition, we have attempted to carry forward the first edition's style, emphasis on practical ideas, many examples, and many exercises.

cepts, such as syntax, semantics, recursion, set, procedure, graph, and state; a detailed formal background is not required.

Our emphasis differs from that of some in the object-oriented programming community but is in accord with the information modeling and design methodology communities. We emphasize object-oriented constructs as models of real things, rather than as techniques for programming. We elevate interobject relationships to the same semantic level as classes, rather than hiding them as pointers inside objects. We place somewhat less emphasis on inheritance and methods. We downplay fine details of inheritance mechanisms. We come down strongly in favor of typing, classes, modeling, and advance planning. We also show how to apply object-oriented concepts to state machines.

The book contains four parts. Part 1 presents object-oriented concepts in a high-level, language-independent manner. These concepts are fundamental to the rest of the book, although advanced material can be skipped initially. The UML notation is introduced in Part 1 and used throughout the book. Part 2 describes a step-by-step object-oriented methodology of software development from problem statement through analysis, system design, and class design. All but the final stages of the methodology are language independent. Part 3 describes the implementation of object-oriented designs in object-oriented languages and relational databases. It describes the considerations applicable to different environments, although it is not intended to replace books on object-oriented programming. Part 4 describes software engineering practices needed for successful object-oriented development.

The authors have used object-oriented analysis, design, programming, and database modeling for many years now on a variety of applications. We are enthusiastic about the object-oriented approach and have found it appropriate to almost any kind of application. We have found that the use of object-oriented concepts, together with a graphical notation and a development methodology, can greatly increase the quality, flexibility, and understandability of software. We hope that this book can help get that message across.

The book has a rich variety of exercises that cover a range of application domains and implementation targets. We suggest that you try working some of them as you go along. Ultimately, OO technology is not learned by reading about it, but by trying to practice it. Answers to selected exercises are included at the back of the book.

Comparison With Other Books

There are many books on the market that cover object-oriented technology. This book differs from most in that it teaches how to think about object-oriented modeling, rather than just presenting the mechanics of a programming language or modeling notation.

Many of the available object-oriented books are about programming issues, often from the point of view of a single language. Some of these books do discuss design issues, but they are still mainly about programming. Few books focus on object-oriented analysis or design. We show that object-oriented concepts can and should be applied throughout the entire software life cycle.

notation to be used throughout the entire software development process. The software developer does not need to translate into a new notation at each development stage.

We show how to use object-oriented concepts throughout the entire software life cycle, from analysis through design to implementation. The book is not primarily about object-oriented languages or coding. Instead we stress that coding is the last stage in a process of development that includes stating a problem, understanding its requirements, planning a solution, and implementing a program in a particular language. A good design technique defers implementation details until later stages of design to preserve flexibility. Mistakes in the front of the development process have a large impact on the ultimate product and on the time needed to finish. We describe the implementation of object-oriented designs in object-oriented languages and relational databases.

The book emphasizes that object-oriented technology is more than just a way of programming. Most importantly, it is a way of thinking abstractly about a problem using real-world concepts, rather than computer concepts. We have found this to be a difficult transition for some people. Books that emphasize object-oriented programming often fail to help the programmer learn to think abstractly. We have found that a graphical notation helps the software developer visualize a problem without prematurely resorting to implementation.

We show that object-oriented technology provides a practical, productive way to develop software for most applications, regardless of the final implementation language. We take an informal approach in this book; there are no proofs or formal definitions with Greek letters. We attempt to foster a pragmatic approach to problem solving by drawing upon the intuitive sense that object-oriented technology captures and by providing a notation and methodology for using it systematically on real problems. We provide tips and examples of good and bad design to help the software developer avoid common pitfalls.

Who Should Read This Book?

This book is intended for both software professionals and students. The reader will learn how to apply object-oriented concepts to all stages of the software development life cycle. We do not assume any prior knowledge of object-oriented concepts. We do assume that the reader is familiar with basic computing concepts, but an extensive formal background is not required. Even existing object-oriented programmers will benefit from learning how to design programs systematically; they may be surprised to discover that certain common object-oriented coding practices violate principles of good design.

The database designer will find much of interest here. Although object-oriented programming languages have received the most attention, object-oriented design of databases is also compelling and immediately practical. We include an entire chapter describing how to implement an object-oriented model using relational databases.

This book can be used as a textbook for a graduate or advanced undergraduate course on software engineering or object-oriented technology. It can be used as a supplementary text for courses on databases or programming languages. Prerequisites include exposure to modern programming languages and a knowledge of basic computer science terms and con-

Preface

Welcome to the second edition of *Object-Oriented Modeling and Design*. Much has changed since we finished the first book (1991). Back then object-oriented (OO) technology was considered new. Despite the excitement and enthusiasm, there was concern whether OO was really practical or just a passing fad. Consider all that has changed:

- **OO languages.** C++ is now established and Java has also become popular. The dominant programming languages are now OO.
- **OO databases.** Somewhat surprisingly, OO databases have faded, but relational databases are now including some OO features.
- **OO modeling.** The Unified Modeling Language (UML) standard from the Object Management Group has consolidated the multiple competing notations.
- **OO methodology.** Development methodologies now routinely incorporate OO ideas and concepts.

OO technology has truly become part of the computing mainstream. OO technology is no longer the exception; rather it is the usual practice.

What You Will Find

This book presents an object-oriented approach to software development based on modeling objects from the real world and then using the model to build a language-independent design organized around those objects. Object-oriented modeling and design promote better understanding of requirements, cleaner designs, and more maintainable systems. We describe a set of object-oriented concepts and a language-independent graphical notation that can be used to analyze problem requirements, design a solution to the problem, and then implement the solution in a programming language or database. Our approach allows the same concepts and

Contents	xi
Chapter 22 Managing Models	403
22.1 Overview of Managing Models, 403	
22.2 Kinds of Models, 403	
22.3 Modeling Pitfalls, 404	
22.4 Modeling Sessions, 406	
22.5 Organizing Personnel, 409	
22.6 Learning Techniques, 410	
22.7 Teaching Techniques, 410	
22.8 Tools, 411	
22.9 Estimating Modeling Effort, 413	
22.10 Chapter Summary, 414	
Bibliographic Notes, 414	
References, 415	
Chapter 23 Legacy Systems	416
23.1 Reverse Engineering, 416	
23.2 Building the Class Model, 418	
23.3 Building the Interaction Model, 419	
23.4 Building the State Model, 420	
23.5 Reverse Engineering Tips, 420	
23.6 Wrapping, 421	
23.7 Maintenance, 422	
23.8 Chapter Summary, 422	
Bibliographic Notes, 423	
References, 424	
Appendix A UML Graphical Notation	425
Appendix B Glossary	426
Answers to Selected Exercises	441
Index	469

18.6 Chapter Summary, 342	
Bibliographic Notes, 343	
References, 343	
Exercises, 344	
Chapter 19 Databases	348
19.1 Introduction, 348	
19.2 Abbreviated ATM Model, 352	
19.3 Implementing Structure—Basic, 352	
19.4 Implementing Structure—Advanced, 360	
19.5 Implementing Structure for the ATM Example, 363	
19.6 Implementing Functionality, 366	
19.7 Object-Oriented Databases, 370	
19.8 Practical Tips, 371	
19.9 Chapter Summary, 372	
Bibliographic Notes, 373	
References, 373	
Exercises, 374	
Chapter 20 Programming Style	380
20.1 Object-Oriented Style, 380	
20.2 Reusability, 380	
20.3 Extensibility, 384	
20.4 Robustness, 385	
20.5 Programming-in-the-Large, 387	
20.6 Chapter Summary, 390	
Bibliographic Notes, 391	
References, 391	
Exercises, 391	
Part 4: Software Engineering	393
Chapter 21 Iterative Development	395
21.1 Overview of Iterative Development, 395	
21.2 Iterative Development vs. Waterfall, 395	
21.3 Iterative Development vs. Rapid Prototyping, 396	
21.4 Iteration Scope, 397	
21.5 Performing an Iteration, 398	
21.6 Planning the Next Iteration, 399	
21.7 Modeling and Iterative Development, 399	
21.8 Identifying Risks, 400	
21.9 Chapter Summary, 401	
Bibliographic Notes, 402	
References, 402	

Bibliographic Notes, 264	
References, 264	
Exercises, 264	
Chapter 15 Class Design	270
15.1 Overview of Class Design, 270	
15.2 Bridging the Gap, 271	
15.3 Realizing Use Cases, 272	
15.4 Designing Algorithms, 274	
15.5 Recursing Downward, 279	
15.6 Refactoring, 280	
15.7 Design Optimization, 280	
15.8 Reification of Behavior, 284	
15.9 Adjustment of Inheritance, 284	
15.10 Organizing a Class Design, 288	
15.11 ATM Example, 290	
15.12 Chapter Summary, 290	
Bibliographic Notes, 292	
References, 293	
Exercises, 293	
Chapter 16 Process Summary	298
16.1 System Conception, 299	
16.2 Analysis, 299	
16.3 Design, 300	
Part 3: Implementation	301
Chapter 17 Implementation Modeling	303
17.1 Overview of Implementation, 303	
17.2 Fine-tuning Classes, 303	
17.3 Fine-tuning Generalizations, 305	
17.4 Realizing Associations, 306	
17.5 Testing, 310	
17.6 Chapter Summary, 312	
Bibliographic Notes, 312	
References, 313	
Exercises, 313	
Chapter 18 OO Languages	314
18.1 Introduction, 314	
18.2 Abbreviated ATM Model, 317	
18.3 Implementing Structure, 317	
18.4 Implementing Functionality, 331	
18.5 Practical Tips, 341	

Chapter 11 System Conception	173
11.1 Devising a System Concept, 173	
11.2 Elaborating a Concept, 174	
11.3 Preparing a Problem Statement, 176	
11.4 Chapter Summary, 178	
Exercises, 179	
Chapter 12 Domain Analysis	181
12.1 Overview of Analysis, 181	
12.2 Domain Class Model, 183	
12.3 Domain State Model, 201	
12.4 Domain Interaction Model, 204	
12.5 Iterating the Analysis, 204	
12.6 Chapter Summary, 206	
Bibliographic Notes, 206	
References, 207	
Exercises, 207	
Chapter 13 Application Analysis	216
13.1 Application Interaction Model, 216	
13.2 Application Class Model, 224	
13.3 Application State Model, 227	
13.4 Adding Operations, 233	
13.5 Chapter Summary, 234	
Bibliographic Notes, 236	
References, 236	
Exercises, 236	
Chapter 14 System Design	240
14.1 Overview of System Design, 240	
14.2 Estimating Performance, 241	
14.3 Making a Reuse Plan, 242	
14.4 Breaking a System into Subsystems, 244	
14.5 Identifying Concurrency, 246	
14.6 Allocation of Subsystems, 248	
14.7 Management of Data Storage, 250	
14.8 Handling Global Resources, 252	
14.9 Choosing a Software Control Strategy, 253	
14.10 Handling Boundary Conditions, 255	
14.11 Setting Trade-off Priorities, 255	
14.12 Common Architectural Styles, 256	
14.13 Architecture of the ATM System, 261	
14.14 Chapter Summary, 262	

Contents	vii
Chapter 6 Advanced State Modeling	110
6.1 Nested State Diagrams, 110	
6.2 Nested States, 111	
6.3 Signal Generalization, 114	
6.4 Concurrency, 114	
6.5 A Sample State Model, 118	
6.6 Relation of Class and State Models, 123	
6.7 Practical Tips, 124	
6.8 Chapter Summary, 125	
Bibliographic Notes, 126	
References, 126	
Exercises, 126	
Chapter 7 Interaction Modeling	131
7.1 Use Case Models, 131	
7.2 Sequence Models, 136	
7.3 Activity Models, 140	
7.4 Chapter Summary, 144	
Bibliographic Notes, 144	
References, 145	
Exercises, 145	
Chapter 8 Advanced Interaction Modeling	147
8.1 Use Case Relationships, 147	
8.2 Procedural Sequence Models, 152	
8.3 Special Constructs for Activity Models, 154	
8.4 Chapter Summary, 157	
References, 157	
Exercises, 158	
Chapter 9 Concepts Summary	161
9.1 Class Model, 161	
9.2 State Model, 161	
9.3 Interaction Model, 162	
9.4 Relationship Among the Models, 162	
Part 2: Analysis and Design	165
Chapter 10 Process Overview	167
10.1 Development Stages, 167	
10.2 Development Life Cycle, 170	
10.3 Chapter Summary, 171	
Bibliographic Notes, 172	
Exercises, 172	

Chapter 3	Class Modeling	21
3.1	Object and Class Concepts, 21	
3.2	Link and Association Concepts, 27	
3.3	Generalization and Inheritance, 37	
3.4	A Sample Class Model, 41	
3.5	Navigation of Class Models, 43	
3.6	Practical Tips, 48	
3.7	Chapter Summary, 49	
	Bibliographic Notes, 50	
	References, 51	
	Exercises, 52	
Chapter 4	Advanced Class Modeling	60
4.1	Advanced Object and Class Concepts, 60	
4.2	Association Ends, 63	
4.3	N-ary Associations, 64	
4.4	Aggregation, 66	
4.5	Abstract Classes, 69	
4.6	Multiple Inheritance, 70	
4.7	Metadata, 75	
4.8	Reification, 76	
4.9	Constraints, 77	
4.10	Derived Data, 79	
4.11	Packages, 80	
4.12	Practical Tips, 81	
4.13	Chapter Summary, 82	
	Bibliographic Notes, 83	
	References, 83	
	Exercises, 83	
Chapter 5	State Modeling	90
5.1	Events, 90	
5.2	States, 92	
5.3	Transitions and Conditions, 94	
5.4	State Diagrams, 95	
5.5	State Diagram Behavior, 99	
5.6	Practical Tips, 103	
5.7	Chapter Summary, 103	
	Bibliographic Notes, 105	
	References, 106	
	Exercises, 106	

Contents

Preface

- What You Will Find, xiii
- Who Should Read This Book?, xiv
- Comparison With Other Books, xv
- Changes From the First Edition, xvi
- Web Site, xvii
- Acknowledgements, xvii

Chapter 1 Introduction

1

- 1.1 What Is Object Orientation?, 1
- 1.2 What Is OO Development?, 3
- 1.3 OO Themes, 6
- 1.4 Evidence for Usefulness of OO Development, 8
- 1.5 OO Modeling History, 9
- 1.6 Organization of This Book, 9
- Bibliographic Notes, 10
- References, 11
- Exercises, 11

Part 1: Modeling Concepts

13

Chapter 2 Modeling as a Design Technique

15

- 2.1 Modeling, 15
- 2.2 Abstraction, 16
- 2.3 The Three Models, 16
- 2.4 Chapter Summary, 18
- Bibliographic Notes, 19
- Exercises, 19

S I T Library
Valachil, Mangalore



Accn No: M001197

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Copyright © 2005 by Pearson Education, Inc.

This edition is published by arrangement with Pearson Education, Inc. and Dorling Kindersley Publishing, Inc.

This book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior written consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser and without limiting the rights under copyright reserved above, no part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), without the prior written permission of both the copyright owner and the above-mentioned publisher of this book.

Pearson Prentice Hall™ is a trademark of Pearson Education, Inc.

Pearson® is a registered trademark of Pearson Plc.

Prentice Hall® is a registered trademark of Pearson Education, Inc.

ISBN 978-81-317-1106-4

10 9 8

Srinivas Institute of Technology
Acc. No.: 1197
Call No.:

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives. Circulation of this edition outside of these territories is UNAUTHORIZED.

Published by Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia.

Head Office: 7th Floor, Knowledge Boulevard, A-8(A), Sector-62, Noida 201309, UP, India.

Registered Office: 11, Community Centre, Panchsheel Park, New Delhi 110 017, India.

Printed in India by Chennai Micro Print.

Object-Oriented Modeling and Design with UML™

Second Edition

Michael Blaha
Modelsoft Consulting Corporation

James Rumbaugh
IBM

PEARSON

What Others Have Said About Object-Oriented Modeling and Design with UML, Second Edition

“The first edition of *Object-Oriented Modeling and Design* by James Rumbaugh, Michael Blaha, and their colleagues is already a classic. It has influenced me more than any other book on modeling. I have successfully applied their ideas in large university project courses for over ten years now, and I am glad to see an updated version of this landmark book. It is bound to shape the thinking habits of another generation of software designers and modelers.”

— *Bernd Bruegge, Technical University Munich*

“Blaha & Rumbaugh have done it again. They've updated their classic book for our current times, showing again that by their simple and straightforward explanations, their precise insights, and their key examples and exercises, that the adoption of object-oriented methodology need not be difficult. A must to have, read, and study by any practitioner.”

— *Michael J. Chonoles*

“Our Master and Doctoral programs in information systems are adopting the *Object-Oriented Modeling and Design with UML* (OOMD) methodology. The book, written by two of the leading experts in the field, covers all aspects of OOMD with deep insight, many fine points, and up to date examples. It offers great value to our programs.”

Srinivas ~~POV~~ H. Chang, Lawrence Technological University

Acc. No. 1197

Call No.:

“If you are looking for a book that introduces UML, has a simple and useful object-oriented analysis and design process, and also includes details about important object-oriented concepts, then I strongly recommend that you study this excellent text.”

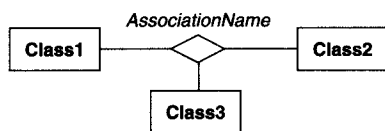
— *Mikael Berndtsson, University of Skövde*

Object-Oriented Modeling and Design with UML™

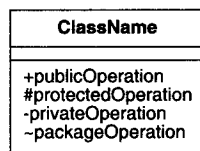
Second Edition

Class Model Notation — Advanced Concepts

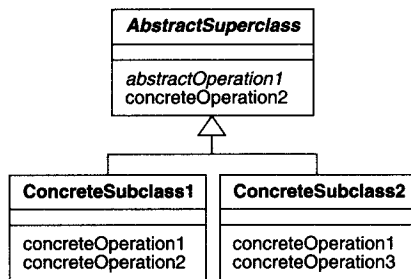
Ternary Association:



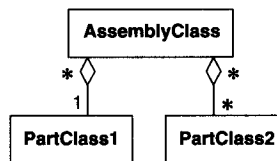
Visibility:



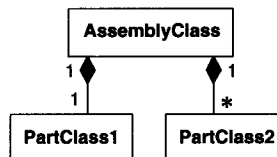
Abstract and Concrete Class:



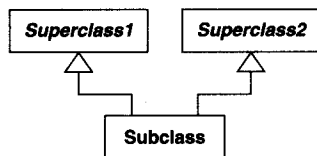
Aggregation:



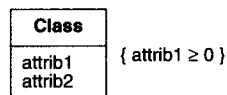
Composition:



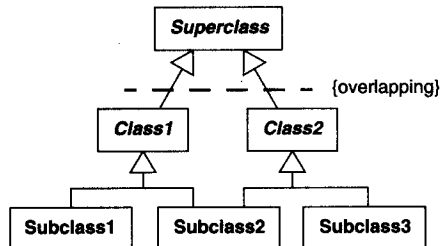
Multiple Inheritance, Disjoint:



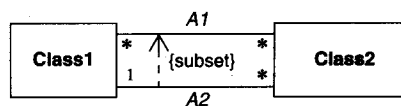
Constraint on Objects:



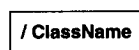
Multiple Inheritance, Overlapping:



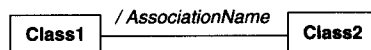
Constraint on Links:



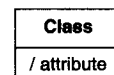
Derived Class:



Derived Association:

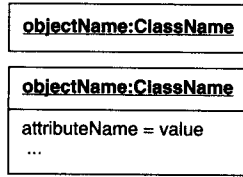


Derived Attribute:

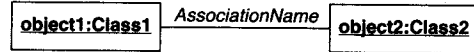


Class Model Notation — Basic Concepts

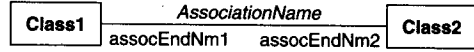
Object:



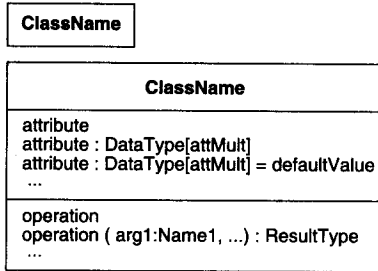
Link:



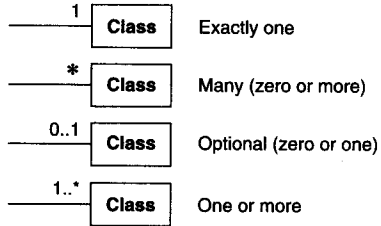
Association:



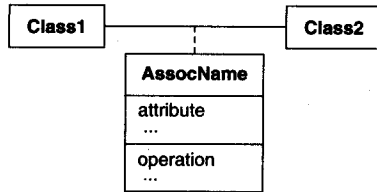
Class:



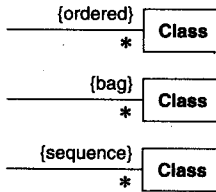
Multiplicity of Associations:



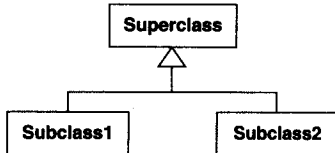
Association Class:



Ordered, Bag, Sequence:



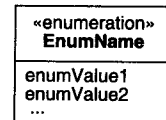
Generalization (Inheritance):



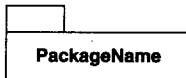
Qualified Association:



Enumeration:



Package:



Comment:

