PREFACE TO SECOND VERSION

About the book

This book provides a coherent coverage of relevant topics for a comprehensive course in programming using Java. It is ideal either for a one-semester course or a two-semester course at college and university level. Techniques for problem solving on the computer are emphasised, and the Java programming language is used for implementing the solutions.

The book assumes no prior knowledge of programming beyond the basic skills required to use a computer. It is also both platform- and programming-tool independent. The book is backed by a website that offers lecture slides, source code for every programming example in the book, links to Java-related resources and more.

This book is a second version of our previous book, *Java Actually: A first course in programming*, and the topics covered in the first three parts of the second version essentially comprise the first version of the book. The second version, *Java Actually: A comprehensive primer in programming*, includes additional and more advanced topics suitable for a more comprehensive course.

The topics covered in this book are up-to-date with Java technology as of JDK 1.6. The aim is *not* to cover every Java-based technology under the sun, but to teach fundamental programming concepts and thereby build a foundation that the reader can use to move on to the more specialised and advanced technologies that use Java.

Our approach can be called *Objects ASAP*, meaning that objects are introduced as soon as possible. Enough structured programming concepts are covered first to write meaningful examples, before proceeding to object-based programming (OBP) for working with objects. The book emphasises testing of program behaviour using assertions, and includes common sorting and searching algorithms. It covers the way in which programs interact with their environment via the terminal window, text files, and simple GUI (Graphical User Interface) dialog boxes.

Object-oriented programming (OOP) and its application are the central themes in this book. Test-driven program development is emphasised and encouraged. Use of collections (lists, sets, maps) is covered after an introduction to generic types. Recursion, as a powerful programming technique, and exception handling, for developing robust programs, are discussed and demonstrated. File I/O and GUI development are used to show how a program communicates with its surroundings.

Basic UML (Unified Modelling Language) diagrams are used to illustrate Java language constructs, basic program design and programming concepts.

Book audiences

The book should be readily accessible to the following audiences:

- Anyone learning how to program for the first time.
- Students who intend to pursue studies in fields other than Computer Science and only require an introductory course in programming. The first three parts of the book can suffice for this purpose.
- Students who intend to pursue studies in Computer Science and require a sound foundation in object-oriented programming (OOP).
- Programmers with backgrounds in other languages who want to migrate to Java.

Prerequisites

The book assumes basic knowledge in the use of the following:

- Basic computer equipment, i.e. computer with keyboard, mouse and screen.
- A normal graphical user interface with windows, buttons and menus.
- A command-line window, to execute commands in the operating system, for example to start a program.
- The file system, to create, delete and find files.
- A text editor, to write text files, for example *emacs* or *vi* on Unix, or *Notepad* on Windows. An IDE (integrated development environment) can be substituted at a later stage in the course.
- The operating system, to install new programs.
- A web browser, to search the Internet for programming resources.

Book themes

The book emphasises the following themes:

- **OOP and its application.** The book is structured around OOP and shows its application in different contexts. The book requires no previous experience of Java programming, and explains concepts from the ground up. It uses the classes from the Java standard library and includes numerous examples of the development of user-defined classes. A case study of developing a game (*Four in a Row*) is used to illustrate test-driven program development.
- **Concepts before syntax.** The book emphasises concepts and shows how these are implemented by the language features in Java. Java syntax is illustrated through examples of typical usage of the language constructs, in which the elements of the syntax are clearly identified.
- **Fundamental data modelling.** Both *data modelling* and *programming* are necessary in order to solve problems on the computer. Modelling of abstractions and data structures is thoroughly explained and illustrated with diagrams.
- **Development of algorithms.** The book encourages algorithm development and uses pseudocode to show the progression from problem analysis to implementation of the solution.

- **Example-driven exposition.** The book uses appropriate examples to explain and apply concepts. Each program example is complete and shows the output, or a screenshot, from the program, so that one can easily reproduce and compare the results. The reader is made aware of the common pitfalls in programming, and practical usage of Java is emphasised through examples.
- **Use of UML (Unified Modelling Language).** We illustrate important programming concepts using easy-to-understand diagrams based on UML. Appendix H provides a short introduction to the notation. The book does not require any prior exposure to UML, and its use is applied only where it intuitively makes sense.
- **Focus on problem solving techniques.** The book uses a few well-chosen case studies to illustrate programming concepts. This approach ensures that the reader becomes familiar with the problem, and the book can focus on problem-solving techniques. Testing program behaviour using assertions is emphasised.

Book features

The source code for all the program examples in the book is available on the book's website and can be downloaded and experimented with. All examples are complete and can be compiled and run immediately. They have been tested thoroughly on several platforms.

Each chapter contains the following sections:

- **Learning objectives.** The objectives listed at the beginning of each chapter clearly outline the concepts and topics covered by the chapter.
- **Review questions.** Ample review questions test the topics covered in each chapter. Annotated answers to all review questions are provided in Appendix A.
- **Programming exercises.** The programming exercises vary in scale and level of difficulty. They help to develop programming skills.

In addition, the book offers the following:

- **Best practice callouts.** These callouts help to promote and facilitate good programming practices.
- **Appropriate cross-references.** The book provides appropriate cross-references that link related concepts.
- **An extensive index.** A comprehensive index aids in locating definitions, concepts and topics in the book.

Practical use of Java

We have paid special attention to the presentation and practical use of the Java programming language. The book emphasises the following aspects:

- **Platform-independent programming language.** Java encourages platform-independent programming, and so does this book. Specific platform-dependent details are provided only where necessary.
- **Programming-tool-independent exposition.** The book uses Java 6.0 and the standard tools provided by Java Development Kit 1.6 (JDK 1.6). Details about compiling

and running Java programs using the standard command line tools *javac* and *java* are given in Appendix G. If it is desirable, other programming tools or IDEs can be used. However, we feel that it is a good idea to avoid the idiosyncrasies of a full-blown IDE at the early stages when learning to program.

- **Use of the Java standard library.** Methods from classes in the Java standard library, which are used in the programming examples, are fully documented where they are used in the book. In addition, we recommend that the reader should have access to the documentation for the Java standard library, either online or installed locally.
- **Creating dialogue between the program and the user.** In the examples we do not use customised classes for interaction between the program and the user. However, the book offers two classes that can be used for this purpose: a class (`Console`) that can be used to read values from the terminal window, and a class (`GUIDialog`) for creating simple graphical user interfaces for reading values via dialog boxes (see Appendix E). The `Console` class encapsulates the use of the `java.util.Scanner` class, while the `GUIDialog` class uses the `javax.swing.JOptionPane` class. Both classes offer methods for reading integers, floating-point numbers and strings.
- **Emphasising the Java advantage.** The book utilizes what the Java language has to offer for an introductory course in programming. For example, assertions are introduced early in the book and used for verifying the behaviour of programs. The `Scanner` class is used to read input from the terminal window, and the `printf()` method is used to format values written to the terminal window or stored in files. Other Java 2 features in the book include the enhanced `for` loop and enumerated types.

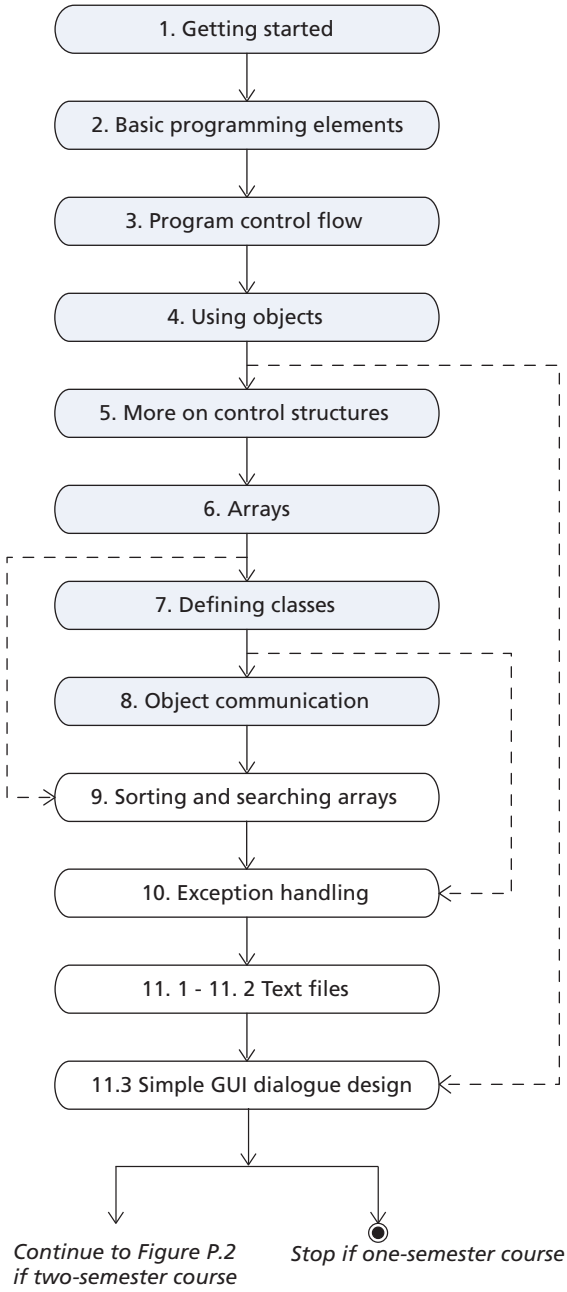
Topics for programming courses

We have organized the material as follows:

- Part 1: Structured programming (Chapters 1–3)
- Part 2: Object-based programming (OBP) (Chapters 4–8)
- Part 3: Useful techniques for building programs (Chapters 9–11)
- Part 4: Object-oriented programming (OOP) (Chapters 12–13)
- Part 5: Applying OOP (Chapters 14–20)
- Part 6: Appendices (A–H)

Readers can choose either a linear or a non-linear route through the book depending on their choice of topics. Some suggestions as to how the book can be used to tailor different types of programming courses (both in size and topic sequence) can be found in Figure P.1 and Figure P.2. Topics up to and including OBP ought to be covered in all courses: these topics are shown in shaded boxes in Figure P.1. The extent of the course can be varied by selecting the remaining topics to be included from the two diagrams. Solid arrows show optimal coverage of the material, and dashed arrows show the earliest point at which additional topics can be introduced to form appropriate course variants.

FIGURE P.1 Core course topics



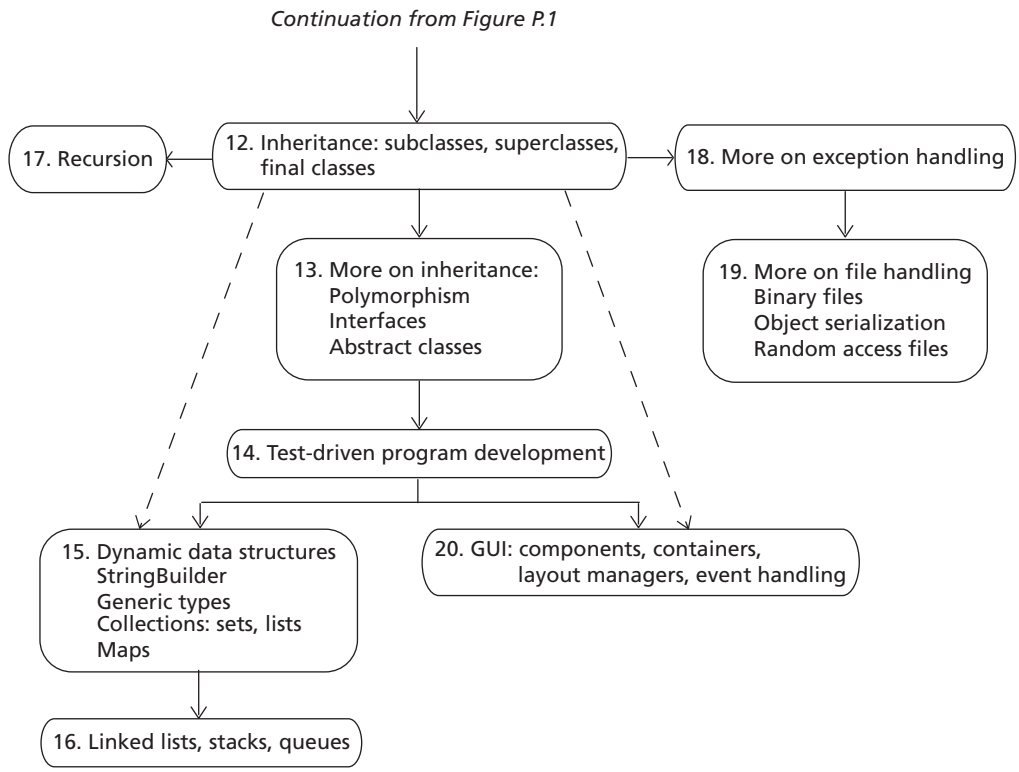
Structured programming (i.e. control structures, together with strings and arrays) and OBP (i.e. objects only and no inheritance) are covered in Part 1 and Part 2 of the book respectively. This organization allows objects to be introduced as early as possible. In Part 3, useful programming techniques include an initial look at simple sorting and searching algorithms for arrays, handling exceptions, reading and writing text files, and creating simple GUI dialog boxes. This approach lays the foundation for more advanced topics in OOP (classes and interfaces using inheritance).

The first three parts of the book, shown in Figure P.1, have been successfully used for a one-semester course. Figure P.2 shows topics from Part 4 and Part 5 of the book that can be included to create a more comprehensive two-semester course.

Inheritance and its consequences for OOP are covered in Part 4. After covering Chapter 12 on inheritance, the remaining chapters of the book can be selected more or less independently. Part 5 includes a number of topics where OOP techniques are applied: a case study in test-driven program development, an introduction to generic types, using dynamic data structures, applying recursive techniques, understanding the exception hierarchy, doing file I/O and developing GUIs for programs.

Appendices in Part 6 provide annotated answers to review questions, useful references (keywords, operators, primitive data types, character codes, formatting code) and succinct introductions to supplementary topics (console I/O and simple GUI dialog design, number representation, JDK tools and UML).

FIGURE P.2 Additional course topics



Conventions used in the source code

Names in Java source code

All class and interface names begin with an uppercase letter. Names of packages, variables and methods begin with a lowercase letter. Constants are always specified with uppercase

letters in their names. In addition, all method names in the text end with `()` to distinguish them from other names.

Code line references in the text

Code lines in examples or code snippets in the text often end with a comment, where a number in parentheses, `()`, is specified after the comment characters `//`:

```
System.out.println("Important business"); // (4)
```

The number is used in the text to refer to the corresponding line in the code. For example, (4) in this text refers to the code line above that has the corresponding number.

The book's website

We have created a website for the book at `http://www.ii.uib.no/~khalid/jac/`.

The book's website offers the following resources:

- Errata.
- Source code for all the examples in the book.
- Links to other useful resources: articles/tutorials on programming, web browsers, Java tools and more.

In addition, the following resources for lecturers can be found here:

- Lecture slides.
- Source code for all the examples in the lecture notes.
- Links providing suggestions for projects and weekly assignments.

Feedback

We appreciate getting feedback. Questions, comments, suggestions and corrections can be sent to: `jac@ii.uib.no`.

The authors

In 1997 the Department of Informatics, University of Bergen (UoB), switched to Java in its introductory course in programming. Mughal and Rasmussen were responsible for developing a new format for this course. In autumn 2004 the Norwegian Quality Reform in Higher Education led to further changes in the curriculum. Mughal was one of the main architects behind the revision of the programming courses at the Department of Informatics, UoB.

The three authors have collaborated on an introductory textbook for programming in Java, written in Norwegian, *Java som første programmeringsspråk / Java as First Programming Language*, third edition, Cappelen Akademisk Forlag, ISBN 82-02-24554-0, Sept. 2006, `http://www.ii.uib.no/~khalid/jfps3u/`. Both versions of our book, *Java Actually*, are heavily based on the Norwegian textbook.

Mughal and Rasmussen are also co-authors of a book on the first exam in certification for the Java technology, *A Programmer's Guide to Java Certification: A Comprehensive Primer*, second edition, Addison-Wesley, ISBN 0201728281, Aug. 2003, `http://www.ii.uib.no/~khalid/pgjc2e/`.

The three authors have been involved in developing web-based variants of the programming courses offered by the Department of Informatics, UoB. These web-based courses are offered every spring (see `http://nettkurs.uib.no/jafu/`). Mughal and Hamre have run a series of seminars on object orientation at the Department of Informatics, UoB. The authors also collaborate on research into the application of object-oriented technology.

Principal author – Khalid Azim Mughal

Khalid A. Mughal is an Associate Professor at the Department of Informatics, UoB. He has developed and given courses for students and the IT industry on programming in Java and Java-related technologies. In 1999, on the basis of the introductory course in programming using Java, he was awarded the Best Lecturer prize at the Faculty of Mathematics and Natural Sciences, UoB. His teaching experience spans programming languages, object-oriented software development, web application development, e-learning, databases and compiler techniques.

His current work involves applying object-oriented technology in the development of learning content management systems and building software security into applications.

He has spent over four years at the Department of Computer Science, Cornell University, both as a Visiting Fellow and as a Visiting Scientist. He is a member of the ACM.

Co-author – Torill Hamre

Torill Hamre is a Research Leader at the Nansen Environmental and Remote Sensing Center in Bergen. The Nansen Center is affiliated to the University of Bergen, where she is an Adjunct Associate Professor at the Department of Informatics.

Her main research is in marine information technologies. She develops object-oriented solutions for marine information systems. She has also given courses in object-oriented software development, both at the University and to the IT industry.

Co-author – Rolf W. Rasmussen

Rolf W. Rasmussen is a System Development Manager at *vizrt* in Bergen, a company that provides solutions for the television broadcasting industry worldwide. He works on control and information systems, video processing, typography and real-time graphics visualization. Over the years he has worked both academically and professionally with numerous programming languages, including Java. He has contributed to the development of GCJ (GNU's implementation of Java), which is a part of the GNU Compiler Collection, where he has worked on the clean room implementation of graphics libraries for Java.

Acknowledgements

First, we would like to thank Gaynor Redvers-Mutton and Matthew Lane at Cengage Learning. Without Gaynor's conviction that we could pull this off, we are not sure whether we would have embarked on this venture. She managed to convince us to write, not one version of the book, but two! Thank you also for catering to our whims and for all the support during the writing process of the second version as well.

We were impressed with the results that FrameMaker guru Steve Rickaby of WordMongers managed to conjure forth when it came to the design of the first version of the book. Using the same book design to write the second version was a breeze. Steve put the finishing touches to create the final product, with which we are very pleased. Again, many thanks, Steve!

We are also very much indebted to the four anonymous technical reviewers of the first version of this book, who gave us encouraging and invaluable feedback on the initial draft of some of the book chapters.

We are fortunate to have Marit Seljeflot Mughal as our personal expert on language washing for our writing. Your efforts have saved us, time after time, not only from language bloopers, but also from errors in our Java code. Thank you for reading countless chapter drafts for both versions and providing invaluable advice on improvements to the manuscript.

We would also like to thank the Department of Informatics, UoB, for providing an environment conducive to writing this book, and testing the course material on which the book is based.

Khalid Mughal would also like to thank the Department of Computer Science, Cornell University, for allowing him to spend his sabbatical (Fall 2007 to Spring 2008) there. Many chapters for the second version were written and revised at that department.

Without family support, this book simply would not have seen the light of day. Many thanks for being patient, both when the dinner got cold and when family plans changed in favour of working on the book. It it not always easy to get our priorities right when the book deadline is approaching at the speed of light.

Bergen/Ithaca, 21 December, 2007.

Khalid A. Mughal
Torill Hamre
Rolf W. Rasmussen