2001

Basic Java Programming: A Laboratory Approach

Lewis Barnett
University of richmond, lbarnett@richmond.edu

Basic Java Programming

A Laboratory Approach

Joe Kent & Lewis Barnett

University of Richmond

Franklin, Beedle & Associates
8536 SW St. Helens Drive, Suite D
Wilsonville, OR 97070
(503) 682-7668
www.fbeedle.com

Preface

Learning to Program

Programming requires careful analysis of a problem followed by the design of a solution. Only then is the design implemented, using an appropriate programming language. Java is a relatively new programming language; it is object-oriented and is a popular choice for Web applications. It will be the implementation language for these laboratories.

Learning to program requires

- learning the syntax and semantics of a programming language,
- paying attention to details, and
- developing design solutions for given problems.

For beginners the first two components seem to dominate, while the problems to be solved often seem contrived and unrealistic. This is natural. If you were taking a beginning Spanish class, you would have to learn basic vocabulary and use it with correct grammar in spoken and written situations. Initially your conversations would be simple: "Good morning. How are you?" "Can you tell me where to find the train station?" "Do you have a room with a shower available?" "How much is the bill?" The more sophisticated expression of ideas would have to wait until the basics were mastered and vocabulary was expanded.

This text will provide a firm foundation for advanced topics in computer science—moving from solving basic problems with Java to using object-oriented design to implement a sophisticated computer game.

These materials are designed to be a hands-on learning experience. For that reason, each "chapter" is called a "laboratory." Each lab will include some preliminary discussion of the topics covered, allowing the book to be used as both textbook and laboratory manual for those who want to develop programming skills in Java outside of a formal educational environment.

Coverage

This book incorporates materials intended for use in an introductory programming course. The intended audience is first- or second-year college students with no prior experience in programming. In national curricular reports, the topics included in this text comprise the generic CS1 course, the first course for prospective computer science majors.

The text also provides a rapid hands-on introduction to Java. Computer science is more than just programming, so we introduce some important algorithms and problem-solving approaches and preview advanced topics such as event-driven programming. Our goal is to build a firm foundation for continued work in computer science, while providing glimpses of more advanced topics.

Assumptions

No previous programming experience is assumed. We do assume basic computer literacy—familiarity with basic tools and commands to create folders, copy files, and use a Web browser. In formal educational settings, your instructor may provide a preliminary guided experience to introduce you to the local computer laboratory environment. Of course, the process of creating, compiling, and running programming will be explained in the first lab.

We would like these labs to be independent of any computer environment, but that is not totally possible. We assume the environment to be Microsoft's Windows 95/98/NT/2000.

IDEs

It is possible to create Java programs using a simple editor like Notepad and the tools of the free Java Development Kit (now called the SDK) from Sun Microsystems, but that approach is relatively primitive. A number of integrated development environments, or IDEs, are available. These are applications that use a graphical user interface to provide windows, menus, buttons, and so on to aid in program development and testing. We prefer Kawa by Allaire (`http://www.allaire.com/products/kawa/index.cfm`) because it allows for easy creation of very simple programs and can be used with any version of Java. However, we have made every attempt to make this manual IDE-independent. An appendix covers the basics of several common IDEs that support Java™ 2. Only the first laboratory uses the Kawa IDE's steps for compiling and running a program. These steps are easily replaced by those of other IDEs, using the appendix materials.

LabPkg

We want these labs to be as independent as possible of any particular book. For that reason we have developed a Java package, `LabPkg`, that we use for our applications. The package allows your programs to interact with the user via graphical components. The critical class is ViewFrame, which is instantiated as a window with three components:

- a toolbar with a provided "Exit" button that kills the current application
- a scrolled text area for displaying program output
- a "canvas" for displaying images or drawing figures

The canvas component is optional and is not used in most simple applications. The user may add additional buttons to the toolbar for initiating actions. Interactive input is obtained by pop-up dialog windows generated by calls to simple methods.

If you looked at another text, it would have a different package, as every author has a favorite way to do input in Java. Why? Java supports input from the keyboard on a line-by-line basis, but forces the user to process the string of characters to extract the desired numeric values and handle errors. For beginners the details of this task distract from the primary goal of understanding principles of programming that apply generally. Special packages hide the mess. Unfortunately, there is no standard package at this time.

Our package is based on the paradigm of a program as having three components: the model, the view, and the controller. (This is the MVC paradigm developed by Smalltalk programmers more than 20 years ago.) The *model* is simply the data that represents the current state of the program. The *view* is what the user sees of the data. The *controller* is the part that provides interaction of the user with the program. Often the view and controller components are joined in programs that use a graphical user interface. For most of our programs a ViewFrame object provides the view and controller components.

Appendix A provides information about the package and its use, with simple examples.

Getting Files

A diskette accompanies the text and contains all the files required for the laboratories, plus example programs discussed in the pre-lab readings. The file `index.html` provides a guide to the files and documentation and should be opened in a browser.

Using a browser, the files can be "downloaded" by the student from the diskette to a local hard drive or a network drive. It is also possible to use Windows Explorer to copy the files from the diskette. The files are in subfolders of the folder `labfiles`.

The diskette contains information concerning setting up a personal computer to do the hands-on part of the text.

The Authors

We are computer science faculty at the University of Richmond, with more than 38 combined years of teaching experience. We developed early Java versions of these laboratories as special experiences for our students more than five years ago. In the fall of 1999 they were expanded and used in a formal laboratory setting. Experience and feedback from other faculty allowed us to refine those initial materials.

Thanks

We would like to thank faculty colleagues who provided feedback: Anita Hubbard, Gary Greenfield, and Arthur Charlesworth. Special thanks go to Jim Leisy, our editor, who had faith in us. Of course, we cannot forget the hundreds of students at UR who provided candid responses to their experiences with the materials.

For supporting us while we worked on the manuscript, we give special thanks to our wives, Mary Kent and Rebekah Barnett.