



Capstone Courseware, LLC

33 Boylston Street
Jamaica Plain, MA 02130

877-227-2477
capstonecourseware.com

Introduction to Java Programming

Renel Fredricksen
Robert J. Oberg

Instructor's Guide

Revision 6.0



Revision Notes

Revision 6.0 updates the course for Java 6. There are no content changes at all; the labs have simply been re-tested on the version-6 JDK, and the Eclipse overlay has been rebuilt for Eclipse Europa.

Revision 5.0 is a major revision, bringing the course up to date with Java 5. Since this is an introductory course, not all Java-5 features are appropriate for coverage here, but several are quite fundamental and so the course has been retooled to give them full exposure. Other significant changes include:

- Each exercise now has **build** and **run** scripts. The old approach of having students type “javac *.java” and “java MyProgram” was simple enough but it allowed the user’s CLASSPATH variable, if present, to block loading of the class from the working directory. The new scripts assure that this won’t happen.
- Formatted output is covered in the course, and is used instead of NumberFormat code that was in 1.4.1.
- The new for-each loop is covered and is used throughout the code where appropriate. (Since we don’t cover arrays until Chapter 10 it’s useless until then.)
- Enumerated types are now covered along with booleans in Chapter 5.
- The collections section of the final chapter has been reworked to use Java-5 generic collections, and also covers auto-boxing.
- The InputWrapper class has been rewritten for a more natural and traditional use of exception handling.
- The Cards and Bridge examples have been consolidated into one track called Cards.
- There is a new optional exercise in Chapter 6 on finding prime numbers.
- The section on Exceptions in Chapter 12 has been edited and various corrections have been made there. There is a new example of checked vs. unchecked exceptions.
- A final example shows the use of the packaged algorithms from **java.util.Collections**.





Revision 1.4.1 is a point revision primarily to remove the appendix about VS.NET and J++. As of this revision, we offer optional Eclipse support, and so the old IDE sections have outlived their usefulness. A few very minor fixes were incorporated in this revision: typos in code and in the book.

Revision 1.4 is a major revision. The instructional flow has been improved, a number of new examples and labs introduced, and some content added, notably a chapter on advanced features that introduces inheritance, interfaces and collection classes. The Electronic Commerce Game case study, which proved somewhat problematical in classroom use, has been deleted. A new appendix outlines the use of Visual Studio .NET as a Java IDE, which might be useful in training environments where both Java and Microsoft technologies are taught.

Revision 1.1 introduces a 4-day version of the course suitable for non-C programmers desiring an introductory course prior to taking a rigorous curriculum designed for C/C++ programmers.

Revision 1.0 is the initial revision, a 5-day course designed for new programmers.





Course Overview and Philosophy

Java is a popular and powerful language. Although comparatively simple in its language structure, there are a number of subtleties that can trip up new and less experienced programmers. It is based on C, and the terse nature of C may be unfamiliar to students with some previous experience programming in languages such as COBOL or Visual Basic. And, of course, Java is object-oriented.

This course explicitly targets new and less experienced programmers, providing them with a thorough step-by-step introduction to Java programming. It lays a firm foundation for further study of Java. There are a large number of example programs and many labs.

An important thrust of this course is to teach programming from an object-oriented perspective. It is often difficult for programmers trained originally in a procedural language to start "thinking in objects." This course introduces object-oriented concepts very early, and Java is developed in a way that leverages its object orientation. Most of the course emphasizes simple classes without inheritance. The last chapter introduces inheritance and polymorphism, along with interfaces and collections.

The course deliberately targets a development environment that is stripped down to a JDK and a text editor. There is a very simple directory structure, with all files for each example in a directory for that specific example. The program can then be built and run by two simple commands at the command prompt:

```
build  
run TheProgram
```

where **TheProgram.class** is the class file containing the **main()** method.

Thus the course focuses on programming and not on any complexities of the environment.





Timeline

Allow plenty of time for the labs—give your students a chance to sink their teeth into them and perform experiments on their own. That is what will help them to really learn. It would be better to de-emphasize some material at the end than slight the foundations.

Day 1

Chapter 1	What is Java?
Chapter 2	First Java Programs
Chapter 3	Introduction to Objects
Chapter 4	Data Types and Operators (spans days)

Day 2

Chapter 4	Data Types and Operators (spans days)
Chapter 5	Logical Operations
Chapter 6	Loops and Program Flow (spans days)

Day 3

Chapter 6	Loops and Program Flow (spans days)
Chapter 7	Objects and Classes
Chapter 8	Characters and Strings

Day 4

Chapter 9	Modular Programming
Chapter 10	Arrays

Day 5

Chapter 11	Bit Operations
Chapter 12	Exception Handling and More Program Flow
Chapter 13	Advanced Java Features





The Eclipse Overlay

Some students and instructors may prefer to install an IDE and use it in working through the course exercises. Capstone Courseware provides an optional package of workspace and project files for Eclipse 3.0.2 for this course. (See the course Setup Guide for download URLs.) Instructors, use this package on your own initiative and at your own risk. You should have experience yourself with Eclipse before using the overlay package in the classroom. The workspace and projects have been tested lightly with the course but are not part of the standard product.

That said, this overlay should save a good deal of work for those who wish to use Eclipse instead of the text editor and command-line tools that are standard for the course. See the file `c:\Capstone\JavaIntro\Eclipse\ReadMe.html` for general notes on how to use the Eclipse overlays for Capstone courses. Be prepared to walk students through the first few exercises in Eclipse; the notes in this file are for experienced Eclipse users, and will not be clear to many students on their own.

One foible of Eclipse is that workspaces cannot be arbitrarily relocated without confusing the workbench a bit. To wit, if you choose not to install the workspace to the default, `c:\Capstone`, you will probably find that projects either don't build when they should, or that when you try to run them the Eclipse launcher "can't find the main class." The bottom line is the projects all need to be refreshed if they are opened from a path other than the one at which they were last open. So in this case the best process is to have everyone begin by opening, refreshing, and then closing all projects; this should clean up any odd behavior due to the "surprise" location.

Again, Capstone Courseware can only offer complete technical support on the standard course, and while we hope this overlay is convenient, it is not as thoroughly tested as the core lab image at this time. If a given exercise is giving trouble, please be certain to build and run it from the command line, using the SDK tools as prescribed in the student guide, before contacting Capstone.

We always welcome feedback on our courseware, and especially with this new undertaking we would appreciate whatever comments and criticism you might have. Eclipse, like all IDEs, tries to promote ease of use by providing many different ways of doing the same thing; this is convenient for users but does leave a lot of questions as to what's best practice. At this time we believe consensus is still forming around a number of basic practices, and we'd like to hear how you use Eclipse for training situations and how this overlay works out for you. Please contact Will Provost at provost@capstonecourseware.com.





Chapter Notes

Chapter 1. What is Java?

As its name suggests this chapter is intended as a high level answer to the question of just what Java is. There are many facets to Java, many of which are not addressed in detail in this course. This chapter will at least ensure students obtain a basic acquaintance with the main aspects of Java. There is a short mention of object-oriented concepts at the beginning of the chapter. Try not to get drawn into adlibbing a detailed introduction to objects at this point. It will use up time. Objects are introduced very fully later in the course.

Chapter 2. First Java Programs

This is a very key chapter where students will get practice writing and running tiny Java programs. The basic structure of a Java program is introduced, and this brings the class concept. Students are being led gradually into object-oriented concepts, which will be introduced in Chapter 3 and covered more thoroughly in Chapters 7 and 13.

Besides introducing Java programs, this chapter introduces basic programming concepts such as variables, assignment and simple operators. Along with the ability to do output, the student has all that is needed to write simple Java programs to do useful calculations.

Chapter 3. Introduction to Objects

This chapter begins the serious discussion about objects and classes, and the concept of object-oriented programming. Although a thorough discussion of objects and classes is left to Chapters 7 and 13, by the end of this chapter the students have a good enough understanding of the concepts so they understand the reason for some of the syntax.

Input is introduced via an **InputWrapper** class. Besides overcoming a hurdle (interactive keyboard input in Java is rather difficult!), this class illustrates in a very practical way one of the benefits of object-oriented programming – the ability to create reusable code components.

Chapter 4. Data Types and Operators

This chapter covers the integer and floating point data types. It also covers simple operators and introduces another reusable class – the library Math class. Following the general course philosophy, data is introduced before control.

Chapter 5. Booleans and Enumerations

This chapter introduces the **boolean** data type and logical operations. Simple if tests are introduced in conjunction with boolean. The chapter continues with the switch



statement, and presentation of break as it relates to the switch statement. The if tests are the most important, practical part of the chapter.

A shorter, second section moves from booleans as two-value sets to the general idea of enumerated types. This is treated fairly lightly in this course: we introduce the concept here and have a brief exercise in it at the end of the chapter. For most of the rest of the course we rely on simple strings and string comparison, since full-feature use of Java 5 enums requires a number of other intermediate skills, such as statics and collections.

Chapter 6. Loops and Program Flow

The introduction of loops completes the student's toolkit for writing simple computer programs that can do calculations, make decisions, and perform operations repetitively. The block construct is introduced, and the key ideas of structured programming are covered. The student is thus prepared for a transition from structured programming to object-oriented programming.

Chapter 7. Objects and Classes

This chapter begins the actual object-oriented *programming* in the course. Classes are first introduced as a way to define structured data. You can create new data types not built into the language. **String** is offered as an illustration (and will be covered in the next chapter). This chapter covers the key concept of a "reference" in Java, and the fact that declaring an object does not create it – something with which even C++ programmers can have a little trouble at first. The concept of garbage collection is covered.

The Java **toString()** method is introduced and used extensively. This is a very nice feature of Java, that makes it easy to construct little exerciser programs for unit testing Java classes.





Chapter 8. Characters and Strings

This chapter covers the remaining primitive data type, **char**, and the very important library **String** class. Some of the important methods of **String** are outlined. An exerciser program **StrDemo** is created. Such a program is easy to write and can give the student practice in exercising the features of a class.

When discussing the command-processing loop at the end of the chapter, you might want to point out the possibility of using an enumeration to define all legal commands. Matching user input to enum values is perhaps the weakest feature of the native enumerated type; still there are a couple possible benefits here: avoiding mistyped strings in the code, and avoiding duplication in the usage statement, by iterating over the **valueSet** to list all legal commands. (A rigorous approach here would probably involve stateful enums or a map from command to implementation, but for this intro course that's a bit too heavy.)

Chapter 9. Modular Programming

The student knows enough Java now that things can easily get out of hand. The **StrDemo** program is presented as an illustration of a monolithic program, which can just "grow". It is very easy to incrementally add functionality, and before you know you have a class with a very large method – the antithesis of object oriented programming! This chapter introduces the idea of "modular programming" (in a loose sense) as a preferred alternative to monolithic programs. At this point we are not really dealing with true objects, as we are not using instantiation. In Java this kind of modularity can be implemented through static methods.

The chapter discusses top-down versus bottom-up programming (pointing out there is a role for both in modern software practice). Data passing in methods is passed – both returning an object as a return type and also passing data as input parameters. Call-by-value versus call-by-reference is discussed. The concept of a holder class is introduced for getting data back from a method call. Nested classes are introduced in a simple way.

Chapter 10. Arrays

This is the key chapter that is required before anything really "interesting" can be done. The array is a fundamental data structure in Java, but is very different from arrays in other languages. This complexity needs to be stressed. Arrays are illustrated via another exerciser program **ArrayDemo** that is similar to **StrDemo**. There is a short lab extending **ArrayDemo** and several longer labs.





Chapter 11. Bit Operations

This chapter introduces the bitwise operations, including shifting and masking. This low-level material is essential background for the beginning programmer. Hexadecimal notation is included, which should be a review for experienced programmers. This is a short chapter, and can go very quickly, especially for the experienced programmers.

Chapter 12. Exception Handling and More Control Flow

Exceptions and exception handling is a very important topic and fundamental to any serious Java program, so it should be mentioned in even an introductory course. I/O exceptions and runtime exceptions are discussed, as well as how the programmer can approach each of these types of exceptions. We do not cover creating our own exceptions, rather emphasizing handling exceptions that we are likely to encounter. A lab applies some of these concepts.

The **break** and **continue** statements and **do** loop are also demonstrated here, with examples showing how and when they should be used.

Chapter 13. Advanced Java Concepts

This chapter is intended to be an overview, as well as a teaser for advanced features that the programmer will need to learn later to be proficient in Java. An introduction to inheritance, polymorphism, and interfaces is covered, with examples. After these topics, we also introduce Collections. Several simple labs are provided.





Feedback

We truly do welcome feedback, both of a specific nature (pointing out mistakes) and general suggestions. For the former sending email with a numbered list of corrections would be most helpful.

Please send feedback to:

Will Provost
Capstone Courseware
<mailto:provost@capstonecourseware.com>
www.capstonecourseware.com

