



Capstone Courseware, LLC

33 Boylston Street  
Jamaica Plain, MA 02130

877-227-2477  
capstonecourseware.com

## 103. Java Programming

### Version 6.0

This course teaches programming in the Java language -- i.e. the Java Standard Edition platform. It is intended for programmers with experience in languages other than Java, but who may or may not have any previous Java experience. It focuses on procedural and structured coding skills first, and then offers meticulous, in-depth coverage of object-oriented concepts and how to apply them to Java software design and development. The latter part of the course moves from these basic skills into key parts of the Java SE Core API, including collections, logging, streams, and object serialization. A final chapter introduces automated unit-testing practices using JUnit.

This revision of the course targets the 6.0 version of the Java language and Core API; but it is equally applicable to Java 5 and groups looking for Java training who know they'll be using Java 5 are encouraged to use this course. For training within the Java 1.4 environment, please see version 1.4.3 of this course, which works to the old version but looks ahead to some Java-5/6 language features; to read more about different versions of Java and for help deciding on which version of this course to use, see "Java Versions and Terminology Demystified".)

Students come to Java from a wide range of backgrounds, and this course is designed to be as flexible as possible over the upper end of that range. Specifically:

- Experienced C and C++ programmers will find this course a very good fit and if anything will find that they complete it in a little less than the full five-day timeline.
- Those with experience in languages less like Java, such as Visual Basic, ASP and other Web-scripting languages, and other pseudo-object-oriented languages may need more time in the early going, and this course covers its introductory topics in good depth and offers many optional and "challenge" labs to support this.
- Less experienced programmers or those coming from non-structured languages -- such as COBOL, PL/1, or 4GL tools -- will probably not cover the whole course in a week, and may want to pursue an abbreviated version at a slower pace. This too is quite feasible, but this audience may also want to consider our Course 102, "Introduction to Java Programming," for a more relaxed pace through the early material.



## Prerequisites

- No prior Java experience is required, but students must be experienced programmers in another third-generation (high-level) language. See the overview for suggestions about pace and scope for different backgrounds.





## Learning Objectives

- Chiefly, learn to program effectively in the Java language.
- Understand the Java software architecture, and the design decisions which make Java software portable, efficient, secure and robust.
- Learn how to configure a simple Java development environment.
- Know the grammar, data types and flow control constructs of the Java language for simple procedural programming.
- Understand Java as a purely object-oriented language, and implement software as systems of classes.
- Implement and use inheritance and polymorphism, including interfaces and abstract classes.
- Design appropriate exception handling into Java methods, and use the logging API appropriately.
- Understand the structure of streams in Java, and learn how to use streams to manage file I/O.
- Learn how to use Java Serialization to internalize and externalize potentially complex graphs of objects.
- Build unit tests for Java classes using JUnit.

**Timeline: 5 days.**

## IDE Support: Eclipse Europa

In addition to the primary lab files, an optional overlay is available that adds support for Eclipse Europa. Students can code and build all exercises from within the IDE. Most exercises can be tested from within the IDE as well, though some must be tested from the command line. See also our orientation to Using Capstone's Eclipse Overlays, and please be advised that this is an optional feature; it is not a separate version of the course, and the course itself does not contain explicit Eclipse-specific lab instructions.





## **Chapter 1. The Java Environment**

- Overview of Architecture
- Forms for Java Software
- J2SE, J2EE, and J2ME Platforms
- Java Virtual Machine
- The Core API
- Java Runtime Environment
- Java Developer's Kit
- Java Class Path
- Classes
- Built-In Streams and Command-Line Parameters

## **Chapter 2. Language Fundamentals**

- Source File Format
- Application Classes
- Code Grammar and Expressions
- Identifiers
- Literals
- Operators
- Calling Methods
- Variable Parameter Lists ("varargs")

## **Chapter 3. Data Types**

- Strict Type Checking
- Primitive Types
- Numeric Types
- Characters and Booleans
- Enumerations
- Type Conversion
- Formatted Output
- Object References
- Comparing and Assigning References
- Strings
- Arrays

## **Chapter 4. Flow Control**

- The main Method
- Calling and Returning from Methods
- Conditional Constructs





- Looping Constructs
- Processing Arrays
- Looping and Enumerations
- Processing Varargs
- The Flow-Control Operator
- Break and Continue
- Recursion

### **Chapter 5. Object-Oriented Software**

- Complex Systems
- Abstraction
- Classes and Objects
- Responsibilities and Collaborators
- UML
- Relationships
- Visibility

### **Chapter 6. Classes and Objects**

- Java Classes
- Constructors and Garbage Collection
- Naming Conventions and JavaBeans
- Relationships Between Classes
- Using this
- Visibility
- Packages and Imports
- Overloading Methods and Constructors
- JARs

### **Chapter 7. Inheritance and Polymorphism in Java**

- UML Specialization
- Extending Classes
- Using Derived Classes
- Type Identification
- Compile-Time and Run-Time Type
- Polymorphism
- Overriding Methods
- The @Override Annotation
- Superclass Reference

### **Chapter 8. Using Classes Effectively**





- Class Loading
- Static Members
- Statics and Non-Statics
- Static Initializers
- Static Imports
- Prohibiting Inheritance
- Costs of Object Creation
- Strings and StringBuffer
- Controlling Object Creation
- Understanding Enumerated Types
- Stateful and Behavioral Enumerations

### **Chapter 9. Interfaces and Abstract Classes**

- Separating Interface and Implementation
- UML Interfaces and Realization
- Defining Interfaces
- Implementing and Extending Interfaces
- Abstract Classes

### **Chapter 10. Collections**

- Dynamic Collections vs. Arrays
- UML Parameterized Type
- Generics
- Using Generics
- The Collections API
- The Collection<E> and List<E> Interfaces
- The ArrayList<E> and LinkedList<E> Classes
- Looping Over Collections: Iterable<E>
- Collecting Primitive Values: Auto-Boxing
- Using Wildcards with Generic Types
- Iterators and the Iterator<E> Interface
- Maps and the Map<K,V> Interface
- Sorted Collections
- The SortedSet<E> and SortedMap<K,V> Interfaces
- The Collections Class Utility
- Algorithms
- Conversion Utilities

### **Chapter 11. Exception Handling and Logging**

- Reporting and Trapping Errors





- Exception Handling
- Throwing Exceptions
- Declaring Exceptions per Method
- Catching Exceptions
- The finally Block
- Catch-and-Release
- Chaining Exceptions
- The J2SE Logging API
- Severity Levels
- Log Hierarchies

### **Chapter 12. Inner Classes**

- Passing Behavior
- Inner Classes in GUI Programming
- Named Inner Classes
- Outer Object Reference
- Static Inner Classes
- Anonymous Inner Classes

### **Chapter 13. The Java Streams Model**

- Delegation-Based Stream Model
- InputStream and OutputStream
- Media-Based Streams
- Filtering Streams
- Readers and Writers

### **Chapter 14. Working with Files**

- File Class
- Modeling Files and Directories
- File Streams
- Random-Access Files

### **Chapter 15. Advanced Stream Techniques**

- Buffering
- Data Streams
- Push-Back Parsing
- Byte-Array Streams and String Readers and Writers

### **Chapter 16. Java Serialization**





The Challenge of Object Serialization  
Serialization API  
Serializable Interface  
ObjectInputStream and ObjectOutputStream  
The Serialization Engine  
Transient Fields  
readObject and writeObject  
Externalizable Interface

### Chapter 17. Automated Unit Testing with JUnit

Automated Testing  
JUnit and Related Tools  
The @Test Annotation  
The Assert Class Utility  
Test Runners  
Lifecycle Methods

### Appendix A. Learning Resources

### Appendix B. Compatibility and Migration

- Compatibility: Compiler and Runtime
- Mixing 1.4, 5.0, and 6.0 Classes
- Compatibility with Generics: Type Erasure
- Compatibility with Enumerations and Varargs
- The @SuppressWarnings Annotation
- Migrating 1.4 Code to 5.0
- Runtime Type Safety with "Checked" Collections

### System Requirements

<b>Hardware Requirements (Minimum)</b>	500 MHz, 128 meg RAM, 500 meg disk space.
<b>Hardware Requirements (Recommended)</b>	1.5 GHz, 512 meg RAM, 1 gig disk space.
<b>Operating System</b>	Tested on Windows XP Professional. Course software should be viable on all systems which support a Java 6 Developer's Kit.
<b>Network and Security</b>	Limited privileges required -- please see our standard security requirements at <a href="http://capcourse.com/Guides/Security.html">http://capcourse.com/Guides/Security.html</a> .
<b>Software Requirements</b>	All free downloadable tools.

