



Capstone Courseware, LLC

33 Boylston Street
Jamaica Plain, MA 02130

877-227-2477
capstonecourseware.com

104. Intermediate Java Programming

Version 6.0

This course teaches programming in the Java language -- i.e. the Java Standard Edition platform. It is intended for students with previous Java experience or training, who already know the fundamentals of the Java architecture and basic procedural programming. This course provides 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 key parts of the Java SE Core API, including collections, exception-handling, logging, streams, and object serialization. The course software also includes an optional overlay of workspace and project files to support use of the Eclipse IDE in the classroom. (This requires that the instructor be experienced in use of Eclipse and able to walk students through basic tasks in the IDE.)

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".)

Prerequisites

- Students must be able to write, compile, test, and debug simple Java programs, using structured programming techniques, strong data types, and flow-control constructs such as conditionals and loops. Course 102 is ideal preparation for this course.



Learning Objectives

- Chiefly, learn to program effectively in the Java language.
- 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.
- Use the standard logging API to write diagnostic information at runtime.
- 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.

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. Review of Java Fundamentals

- The Java Architecture
- Forms for Java Software
- Three Platforms
- The Java Language
- Numeric Types
- Characters and Booleans
- Enumerations
- Object References
- Strings and Arrays
- Conditional Constructs
- Looping Constructs
- Varargs

Chapter 2. Object-Oriented Software

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

Chapter 3. 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 4. 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 5. 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 6. Interfaces and Abstract Classes

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

Chapter 7. 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 8. 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 9. Inner Classes

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

Chapter 10. The Java Streams Model

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

Chapter 11. Working with Files

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





Chapter 12. Advanced Stream Techniques

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

Chapter 13. 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 14. 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

Hardware Requirements (Minimum)

500 MHz, 128 meg RAM, 500 meg disk space.





104. Intermediate Java Programming

Outline

Hardware Requirements (Recommended)

1.5 GHz, 512 meg RAM, 1 gig disk space.

Operating System

Tested on Windows XP Professional. Course software should be viable on all systems which support a Java 6 Developer's Kit.

Network and Security

Limited privileges required -- please see our standard security requirements at <http://capcourse.com/Guides/Security.html>.

Software Requirements

All free downloadable tools.

