



Capstone Courseware, LLC

33 Boylston Street
Jamaica Plain, MA 02130

877-227-2477
capstonecourseware.com

106. Advanced Java Programming

Version 5.0

This course provides advanced training in developing software using the Java 2 Platform, Standard Edition, or J2SE. It is intended for students with solid experience in structured and object-oriented Java programming, including use of the Collections API and exception handling. 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 5.0 version of the Java language and Core API. (Java 5.0 is also known as Java 1.5, as this revision effects a transition to a new numbering scheme for the Java environment.) It has been rebuilt thoroughly, not only to cover new 5.0 language features, but also to comb out old programming techniques in favor of new ones. For training within the Java 1.4 environment, please see version 1.4 of this course; 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".)

The course is organized into five modules. The first covers several general-purpose topics: using Java-5.0 generics, writing multi-threaded applications, the Reflection API and annotations, and network programming using sockets. Then the course takes up the challenge of building multi-tier applications using the standard Java platform. Multi-tier development most frequently uses the Java 2 Platform, Enterprise Edition, or J2EE, and we have a wide range of courses in that area. However it is quite possible to build lightweight multi-tier applications using only J2SE, and for some purposes the J2EE is more trouble than it's worth.

So the course looks at each of the traditional tiers and the J2SE APIs most suited to developing them: the Java Foundation Classes, or JFC, for building graphical user interfaces (GUIs) for the presentation tier; Java RMI as a way of distributing Java objects in a separate business tier; and JDBC for the persistence tier. A J2SE multi-tier application provides a comprehensive case study that illustrates multi-tier architecture, design patterns, and best practices, and also provides a few challenge labs. Students will complete the course prepared to build distributed Java applications and to pursue JFC, RMI, or JDBC in greater depth.

Prerequisites



106. Advanced Java Programming

Outline

- Solid Java programming experience is essential -- especially object-oriented use of the language. Language features and techniques that are integral to some lab exercises include interfaces and abstract classes, threading, generics and collections, and recursive methods. Course 103, "Java Programming," is excellent preparation.





Learning Objectives

- Make effective use of Java generic types.
- Write multi-threaded Java applications.
- Use the Reflection API for highly generic tasks, discovery, or code-generation.
- Use standard annotations and develop custom annotations to express meta-data in Java source files.
- Communicate between processes using network sockets.
- Understand the roles of JFC, RMI, JDBC, and other Core API packages in the classic multi-tier architecture for distributed systems.
- Understand the basics of the JFC architecture.
- Build complex GUIs using various JFC controls.
- Understand the relationship between RMI and various J2EE technologies such as JNDI, EJB, and CORBA.
- Implement simple RMI clients and servers.
- Connect to a database using JDBC and perform a simple query.
- Update relational data using JDBC to execute updates, inserts and deletes.
- Use prepared statements to produce reusable database queries and optimize execution time.
- Use callable statements to access database procedures.
- Use scrollable and updatable results sets for more robust solutions.
- Use commit, rollback, and savepoint to build transactional systems.
- Use batch processing for efficient handling of large datasets.
- Use the Java 5.0 Core API and related tools to develop robust multi-tier applications.

Timeline: 5 days.

IDE Support: Eclipse 3.2

In addition to the primary lab files, an optional overlay is available that adds support





for Eclipse 3.2. 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.





Module 1. Advanced Java

Chapter 1. Generics

- Using Generics
- Type Erasure
- Type Boundaries
- Wildcards
- Generic Methods
- Strengths and Weaknesses of Generics
- Legacy Code and Generics

Chapter 2. Threads

- Java Thread Model
- Creating and Running Threads
- Manipulating Thread State
- Thread Synchronization
- Volatile Fields vs. Synchronized Methods
- wait and notify
- join and sleep
- The Concurrency API
- Atomic Operations

Chapter 3. Reflection

- Uses for Meta-Data
- The Reflection API
- The Class<T> Class
- The java.lang.reflect Package
- Reading Type Information
- Navigating Inheritance Trees
- Dynamic Instantiation
- Dynamic Invocation
- Reflecting on Generics

Chapter 4. Annotations

- Aspect-Oriented Programming and Java
- The Annotations Model
- Annotation Types and Annotations





Built-In Annotations
Annotations vs. Descriptors (XML)

Chapter 5. Sockets

The OSI Reference Model
Network Protocols
The Socket Class
The ServerSocket Class
Connecting Through URL Objects
HTTP and Other TCP Servers
Datagram Clients and Servers
Non-Blocking Sockets





Module 2. J2SE Case Study

Chapter 1. Overview

- Three Tiers for J2EE
- Three Tiers for J2SE
- The Case Study
- Design Patterns
- Domain and Service Models

Chapter 2. The Presentation Tier

- The Standalone/Client Application
- JDesktopPane and JInternalFrame
- Adapting JList, JTable, and JTree to Services
- Presentation-Tier Patterns

Chapter 3. The Business Tier

- Distributing the Application
- A Chain of Services
- Logging
- Business-Tier Patterns
- Designing for Latency

Chapter 4. The Persistence Tier

- A Database is Not a Persistence Tier!
- Persistence Frameworks
- Persistent-Object Strategies
- Persistence-Tier Patterns
- Caching





Module 3. The Java Foundation Classes

Chapter 1. Introduction to JFC

- Abstract Windowing Toolkit Basics
- Simple Layout Management
- Simple Event Handling
- Lightweight Controls
- JFC Feature Set
- JFC Architecture and Relationship to AWT

Chapter 2. JFC Application Design

- Role of a JFrame
- Building a Frame-Based JFC Application
- Panes
- Using Dialogs

Chapter 3. JFC Components

- JFC Component Class Hierarchy
- JComponent Features
- Simple Control Types
- Text Components
- Menus
- Managing Look and Feel





Module 4. Remote Method Invocation

Chapter 1. RMI Architecture

- Motivation for RMI
- RMI, EJB, and CORBA
- RMI Architecture
- Lifetime of a Remote Method Invocation
- Registries
- Naming and URL Resolution
- Interface Design
- The Remote Interface
- Implementation Classes
- The RemoteObject and RemoteServer Classes
- The UnicastRemoteObject Class
- Server Implementation
- Using the Registry
- Client Implementation
- Code Deployment

Chapter 2. Practical RMI

- RMI Marshaling
- Passing Objects
- The Factory Pattern
- Serialization vs. Remote Reference
- Designing for Latency
- The Transfer Object Pattern
- Controlling Object Location
- Exception Handling





Module 5. Java Database Connectivity

Chapter 1. Database and SQL Fundamentals

- Relational Databases and SQL
- Database, Schema, Tables, Columns and Rows
- SQL Versions and Vendor Implementations
- DDL -- Creating and Managing Database Objects
- DML -- Retrieving and Managing Data
- Sequences
- Stored Procedures
- Using SQL Terminals

Chapter 2. JDBC Fundamentals

- What is the JDBC API?
- JDBC Drivers
- Making a Connection
- Creating and Executing a Statement
- Retrieving Values from a ResultSet
- SQL and Java Datatypes
- Creating and Updating Tables
- Handling SQL Exceptions and Proper Cleanup
- Handling SQLWarning

Chapter 3. Advanced JDBC

- SQL Escape Syntax
- Using Prepared Statements
- Using Callable Statements
- Scrollable Result Sets
- Updatable Result Sets
- Transactions
- Commits, Rollbacks, and Savepoints
- Batch Processing

Chapter 4. Introduction to Row Sets

- Row Sets in GUI and J2EE programming
- Advantages of RowSets
- RowSet Specializations





Using CachedRowSets

Appendix A. Learning Resources

System Requirements

Hardware Requirements (Minimum)

500 MHz, 256 meg RAM, 500 meg disk space.

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 J2SE 1.4 SDK.

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.

