Effectively Using Java Packages And Features

Duration: 5 Days *(Face-to-Face & Remote-Live)*, or 35 Hours *(On-Demand)*

Price: $2495 *(Face-to-Face & Remote-Live)*, or $1495 *(On-Demand)*

Discounts: We offer multiple discount options. Click here for more information.

Delivery Options: Attend face-to-face in the classroom or remote-live attendance.

Students Will Learn

- Using reflection and introspection to control the publishing and discovery of the properties, events, and methods of Java classes (`java.lang.reflect package`)
- Writing type safe and reliable code with the enhanced capabilities of Java
- Using the classes and interfaces that comprise the Collections Framework (`java.util package`)
- Processing databases using JDBC (`java.sql package`)
- Creating, controlling and synchronizing threads
- Creating and using inner classes and nested classes
- Using functional interfaces and lambda expressions
- Describing and using the networking related classes (java.net package)
- Creating client/server programs including a chat room application

Course Description

This intermediate level course is intended for programmers who already have a fundamental understanding of Java programming and some experience writing code. It provides additional insights and details regarding some of the more advanced and useful capabilities contained in the Java Programming Language and it's associated packages. Topics include reflection and JavaBeans, Java type safety enhancements, the Java Collections Framework, Java Database Connectivity (JDBC), multithreading, inner classes, lambda expressions and networking.

Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Prerequisites

Familiarity with and experience using Java. Completion of either the Learning to Program with Java, Java for COBOL Programmers, or Java Programming course, or equivalent Java programming experience.

Course Overview

JavaBeans, Reflection, and Introspection

- JavaBean Requirements
- Determining Type of an Object
- Reflection Overview and Uses
- Reflection Issues
- Reflection Using the Class Object
- Invoking Methods Using Reflection
- Methods Available on the Class Object
- Creating a New Instance
- Introspection
- Customizing `BeanInfo`
- Invoking Methods Using Introspection
- Using Reflection to Access Properties
- Review of `equals` Method

Type Safety Enhancements

- Annotations
  - Standard Annotations
  - User Defined Annotations
  - Reflection Annotation Information
- The `enum` Data Type
- Generics
- Autoboxing
- Methods Having Variable Parameters
- Assertions

Collections Framework

- What is the Collections Framework
- Simple Arrays and Arrays of Objects
- Legacy Container Classes
- Collections Framework Overview
- Collections Interfaces
  - Lists
  - Sets
  - Queues
- Map Interfaces
- Interface Implementations
  - Lists
  - Sets
  - Queues
  - Maps
- Iterators

JDBC

- What is JDBC
- JDBC Drivers
- Accessing the Database
  - Loading Driver
  - Connecting to Data Source
  - Creating Statements
  - Executing Statements
- Processing Result Sets
  - Cursor Positioning
  - Column Retrieval
  - Updating Result Sets
- Connection Pooling
- Processing Errors and Warnings
- Using Prepared Statements
- Using Stored Procedures
- Metadata
  - Result Set Metadata
  - Database Metadata
- Transaction Processing
- Isolation Levels
- SQL Batches

Threads

- Definition of a Thread
- Creating Threads
- Naming Threads
- Data Sharing Among Threads
  - Local Data
  - Instance Data
  - Class Data
  - `volatile` Keyword

Networking

- Overview of `java.net` Package
- Format of URL
- `URL` Class
- `URLConnection` Class
- `InetAddress` Class
- Definitions
  - Client
  - Server

- Thread States
- Thread Priority
  - Minimum. Maximum, Normal Priorities
  - Preemptive Thread Priority
  - `setPriority` Method
  - `getPriority` Method
- Piping Data Between Threads
- Coordination and Controlling Threads
- Synchronizing Threads
  - Why Synchronization is Needed
  - Producer/Consumer Example
  - `synchronized` Keyword
  - Thread Synchronization Methods

Inner and Nested Classes

- What Are Inner Classes
  - Benefits of Inner Classes
  - Terminology
  - Restrictions
  - Syntax
- Types of Inner Classes
  - Member Inner Class
  - Local Inner Class
  - Anonymous Inner Class
  - Nested Top Level Class
- Lambda Expressions
  - Functional Interfaces
  - Default Interface Methods
  - Static Interfaces Methods
  - Lambda Expression Uses
  - Lambda Expression Syntax
  - Method and Constructor References
  - Port
  - Socket
- TCP/IP Protocols
- Socket Class
  - Writing Client Side Applications
  - Read From Socket
  - Write to Socket
- `ServerSocket` Class
  - Writing Server Side Applications
  - Daemon Servers
  - Multi-Threaded Servers

Hands On Technology Transfer
The Best Way to Transfer Technology Skills

1 Village Square, Suite 8
14 Fletcher Street
Chelmsford, MA 01824