EFFECTIVELY USING JAVA PACKAGES AND FEATURES

General:

This intermediate to advanced 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.

For the most part, this course is independent of the Java development environment that is being used. Although the only requirement is access to the Java Development Kit, the course is best taught using a more formal Integrated Development Environment (IDE) such as Eclipse or RAD.

Objectives:

Upon successful completion of this course, the student will be able to:

- Use reflection and introspection to control the publishing and discovery of the properties, events, and methods of Java classes (`java.lang.reflect` package)
- Use the enhanced capabilities initially provided with Java 5 to write more type safe code
- Use the classes and interfaces that comprise the Collections Framework (`java.util` package)
- Process databases using JDBC (`java.sql` package)
- Create, control, and synchronize threads
- Create and use inner classes and nested classes
- Describe and use functional interfaces and lambda expressions
- Describe and use the networking related classes (`java.net` package)
- Create client/server programs including a chat room application

Audience:

Technical Users, Applications Programmers, and Systems Programmers.

Prerequisites:

Introduction to Java course or equivalent experience with basic Java programming.

Duration:

Five (5) days including any optional review time. The course includes classroom lecture and more than 50 percent hand-on lab sessions.

PanTech
908-281-6759

EFFECTIVELY USING JAVA PACKAGES AND FEATURES
COURSE OUTLINE

I. **REVIEW OF JAVA BASICS**

A. OO Concepts
B. Classes and Interfaces
C. IO Package

II. **JAVABEANS, REFLECTION, AND INTROSPECTION**

A. JavaBean Requirements
B. Determining Type of an Object
C. Reflection Overview and Uses
D. Reflection Issues
E. Reflection Using Class Object
F. Invoking Methods Using Reflection
G. Methods Available on the Class Object
H. Creating a New Instance
I. Introspection
J. Customizing BeanInfo
K. Invoking Methods Using Introspection
L. Using Reflection to Access Properties
M. Review of equals Method

*III. **TYPE SAFETY ENHANCEMENTS***

A. Annotations
   1. Standard Annotations
   2. User Defined Annotations
   3. Reflection Annotation Information
B. The enum Data Type
C. Generics
D. Autoboxing
E. Methods Having Variable Parameters
F. Assertions

IV. **COLLECTIONS FRAMEWORK**

A. What is the Collections Framework
B. Simple Arrays and Arrays of Objects
C. Legacy Container Classes
D. Collections Framework Overview
E. Collections Interfaces
   1. Lists
   2. Sets
   3. Queues
F. Map Interfaces
G. Interface Implementations
   1. Lists
   2. Sets
   3. Queues
   4. Maps
H. Iterators

V. **JDBC**

A. What is JDBC
B. JDBC Drivers
C. Accessing the Database
   1. Loading Driver
   2. Connecting to Data Source
   3. Creating Statements
   4. Executing Statements
D. Processing Result Sets
   1. Cursor Positioning
   2. Column Retrieval
   3. Updating Result Sets
E. Connection Pooling
F. Processing Errors and Warnings
G. Using Prepared Statements
H. Using Stored Procedures
I. Metadata
   1. Result Set Metadata
   2. Database Metadata
J. Transaction Processing
K. Isolation Levels
L. SQL Batches

VI. **THREADS**

A. Definition of Thread
B. Creating Threads
C. Naming Threads
D. Data Sharing Among Threads
    1. Local Data
    2. Instance Data
    3. Class Data
    4. volatile Keyword
E. Thread States
F. Thread Priority
    1. Min, Max, and Norm Priorities
    2. Preemptive Thread Priority
    3. setPriority Method
    4. getPriority Method
G. Piping Data between Threads
H. Coordination and Controlling Threads
I. Synchronizing Threads
    1. Why Synchronization is Needed
    2. Producer/Consumer Model
    3. synchronized Keyword
    4. Thread Synchronization Methods

VII. **INNER AND NESTED CLASSES**

A. What are inner classes
    1. Benefits of Inner Classes
    2. Terminology
    3. Restrictions
    4. Syntax
B. Types of inner classes
    1. Member inner class
    2. Local inner class
    3. Anonymous inner class
    4. Nested top level class

VIII. **LAMBDA EXPRESSIONS**

A. Functional interfaces
B. Interface methods
    1. default interface methods
    2. static interface methods
C. Lambda expressions
    1. Uses for lambda expressions
    2. Lambda expression syntax
    3. Lambda expression vs. anonymous inner class
D. Method & constructor references

IX. **NETWORKING**

A. Overview of `java.net` package
B. Format of URL
C. URL class
D. URLConnection class
E. InetAddress class
F. Definitions
    1. Client
    2. Server
    3. Port
    4. Socket
G. TCP/IP Protocols
H. Socket class
    1. Writing client side applications
    2. Read from Socket
    3. Write to Socket
I. ServerSocket class
    1. Writing server side applications
    2. Daemon servers
    3. Multi-threaded servers