Course duration

- 5 days

Course Benefits

- 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, and use the logging API appropriately.
- Use Java as a functional language, making appropriate choices of tools including inner classes, functional interfaces, method references, and lambda expressions.
- Use the Stream API for efficient processing of data sets.

Available Delivery Methods

Public Class
Public expert-led online training from the convenience of your home, office or anywhere with an internet connection. Guaranteed to run .

Private Class
Private classes are delivered for groups at your offices or a location of your choice.

Course Outline

1. Review of Java Fundamentals
    1. The Java Architecture
    2. Forms for Java Software
    3. Three Platforms
    4. The Java Language
    5. Numeric Types
    6. Characters and Booleans
    7. Enumerations
    8. Object References
    9. Strings and Arrays
    10. Conditional Constructs

    11. Looping Constructs
    12. Varargs
2. Object-Oriented Software
    1. Complex Systems
    2. Abstraction
    3. Classes and Objects
    4. Responsibilities and Collaborators
    5. UML
    6. Relationships
    7. Visibility
3. Classes and Objects
    1. Java Classes
    2. Constructors and Garbage Collection
    3. Naming Conventions and JavaBeans
    4. Relationships Between Classes
    5. Using this
    6. Visibility
    7. Packages and Imports
    8. Overloading Methods and Constructors
    9. JARs
4. Inheritance and Polymorphism in Java
    1. UML Specialization
    2. Extending Classes
    3. Using Derived Classes
    4. Type Identification
    5. Compile-Time and Run-Time Type
    6. Polymorphism
    7. Overriding Methods
    8. The @Override Annotation
    9. Superclass Reference
5. Using Classes Effectively
    1. Class Loading
    2. Static Members
    3. Statics and Non-Statics
    4. Static Initializers
    5. Static Imports
    6. Prohibiting Inheritance
    7. Costs of Object Creation
    8. Strings and StringBuffers
    9. Controlling Object Creation
    10. Understanding Enumerated Types
    11. Stateful and Behavioral Enumerations
6. Interfaces and Abstract Classes
    1. Separating Interface and Implementation
    2. UML Interfaces and Realization
    3. Defining Interfaces
    4. Implementing and Extending Interfaces

    5. Abstract Classes
7. Collections
    1. Dynamic Collections vs. Arrays
    2. UML Parameterized Type
    3. Generics
    4. Using Generics
    5. The Collections API
    6. The Collection<E> and List<E> Interfaces
    7. The ArrayList<E> and LinkedList<E> Classes
    8. Looping Over Collections: Iterable<E>
    9. Collecting Primitive Values: Auto-Boxing
    10. Using Wildcards with Generic Types
    11. Iterators and the Iterator<E> Interface
    12. Maps and the Map<K,V> Interface
    13. Sorted Collections
    14. The SortedSet<E> and SortedMap<K,V> Interfaces
    15. The Collections Class Utility
    16. Algorithms
    17. Conversion Utilities
8. Exception Handling and Logging
    1. Reporting and Trapping Errors
    2. Exception Handling
    3. Throwing Exceptions
    4. Declaring Exceptions per Method
    5. Catching Exceptions
    6. The finally Block
    7. Catch-and-Release
    8. Chaining Exceptions
    9. try-with-resources
    10. Logging
    11. The Java SE Logging API
    12. Loggers
    13. Logging Levels
    14. Handlers
    15. Configuration
    16. Best Practices
9. Nested Classes
    1. Nested Classes
    2. Static Classes
    3. Inner Classes
    4. Relationship with the Outer Object
    5. Local Classes
    6. Enclosing Scope
    7. Anonymous Classes
10. Functional Programming
    1. Passing Behavior as a Parameter
    2. Inner Classes

    3. Functional Interfaces
    4. Built-In Functional Interfaces
    5. Lambda Expressions
    6. Scope and Visibility
    7. Deferred Execution
    8. Method References
    9. Creational Methods
    10. Designing for Functional Programming
    11. Default Methods
11. Streams
    1. The Stream Processing Model
    2. Streams
    3. Relationship to Collections
    4. Advantages and Disadvantages
    5. Iterating, Filtering, and Mapping
    6. Primitive-Type Streams
    7. Aggregate Functions and Statistics
    8. Sorting
    9. Generating, Limiting, and Reducing
    10. Finding and Matching
    11. Grouping
    12. Flattening and Traversing
    13. Sequential vs. Parallel Processing

Class Materials

Each student will receive a comprehensive set of materials, including course notes and all the class examples.

Class Prerequisites

Experience in the following *is required* for this Java class:

- 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.

Prerequisite Courses

Courses that can help you meet these prerequisites:

- Introduction to Java Training
- Object-Oriented Analysis and Design (OOAD) Training with UML

Follow-on Courses

- Advanced Java Programming
- Object-Oriented Analysis and Design (OOAD) Training with UML