Bright Star Institute

302.335.8868

admissions@brightstarinstitute.com

Java™ Programming

Duration: 90 hours

Prerequisites: Basic programming skills in a structured language. Knowledge and experience with Object-Oriented Design (OOD) is helpful, but not required.

Description: This hands on course introduces experienced programmers to Java™ technology and Java programming techniques. The Java platform provides an object-oriented, portable and robust framework for application development. Included are core language concepts including fundamental data types, flow control, and standard function libraries. The course emphasizes object oriented programming and modular design to support distributed development environments. Included are the design of classes and objects, inheritance and polymorphism, and the details about creating programs for use on a distributed network, with emphasis on JSP, Servlets, and JDBC.The course also includes coverage of the Java Collections API, fundamental I/O, exceptions, and exception handling.

The course is designed to leverage the participants' existing programming skills and to highlight the new and extended features of the Java programming framework as compared to other common languages. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Overview

Introduction to Java

- Cornerstones of the Java Platform
- Java Advantages
- The Java Programming Language
- The Java Virtual Machine (JVM)
- Core Java Libraries
- Extension Libraries

Developing Software Using Java

- Applications, Applets, Web Components
- Java SE, Java EE, Java ME
- Installing the JDK
- Compiling and Running Java from the Command Line
- The `main()` Method
- `package` and `import` Statements
- JAR Files

- Class Loading and `CLASSPATH`
- Online API Documentation
- JDK Tools
- Java Integrated Development Environments (IDEs)

Java Syntax Fundamentals

- Comments
- Identifiers
- Reserved Words
- Classes
- Statements and Blocks
- Variables, Constants, Literals
- Scope of Variables
- Methods
- Method Overloading
- static Members
- Static Import
- Naming Conventions

Data Types and Operators

- Primitive Types
- Boolean, Integer, Floating-Point and Character Types
- Unicode Characters and Strings
- Type Conversion and Casting
- Expressions and Operators
- Arithmetic Operators
- Increment/Decrement Operators
- Division and Remainder Operators
- Assignment Operators
- Relational Comparison and Logical Operators
- Conditional Operator
- Bitwise Operators
- Order of Evaluation
- Operator Precedence and Associativity

Flow of Control

- `if/else` Statement
- Combining `ifs`
- `while` and `do/while` Loops
- `for` Loop and Loop Counters
- `break` and `continue`
- Break to Labeled Loops
- `switch` Statement
- `return` Statement
- Exit Status

Using Java Classes and Objects

- Classes as Data Types
- Objects and References
- Memory in the JVM
- Object Initialization
- Objects as Arguments to Methods
- Objects as Return Values
- Garbage Collection
- Primitive Wrapper Classes - Integer, Double, etc.
- Autoboxing and Unboxing

Strings

- String Manipulation
- `StringBuffer` and `StringBuilder`
- Simple Number/String Conversion

Arrays

- Declaring and Allocating Arrays
- Multi-Dimensional Array
- Array Literals
- The `java.util.Arrays` Class

- Command-Line Arguments
- Enhanced for Loop
- Arrays as Method Arguments
- Variable-Length Arglists (varargs)
- Autoboxing and varargs

Developing Java Classes

- Object-Oriented (OO) Concepts
- Methods, Member Variables
- Accessing Members
- Tight Encapsulation
- Access Control Modifiers
- Constructors and Finalizer
- Using `this`
- Class Variables - Static Members and Static Blocks
- Instance Variables
- Local Variables
- Variables and Initialization
- Inner Classes
- Anonymous Classes
- JavaBeans
- Driver Classes

Inheritance

- Extending Java Classes
- Accessing Superclass Constructors and Members
- Overriding Methods
- Abstract CLasses and Methods
- Polymorphism
- Overriding Methods of `java.lang.Object`
- `equals()`, `toString()`, `hashCode()`
- Final Classes and Methods
- Multiple Inheritance
- Interfaces
- Casting Object References
- Documenting Classes with the `javadoc` Utility
- Unit Testing

Type Safety

- Annotations
- Java SE Built-In Annotations
- Defining New Annotations
- Enumerated Types
- Constants and Constrained Values
- Defining and Declaring `enums`
- `enum` Values
- `enums` and `switch` Statements
- `values()` and `valueOf()`
- Generic Classes
- Generic Type Parameters
- Using Type Parameters `inClass`, Variable and Method Declarations
- Using a Generic Class
- Bounded Type Parameters

The Collections Framework

- The `java.util` Package
- Container Objects
- Arrays as Containers
- Legacy Container Classes - Vector, Hashtable, Enumeration
- Legacy Container Generic Forms
- Collections Interfaces - `Collection<E>`, `List<E>`, `Set<E>`, `SortedSet<E>`
- Map Interfaces - `Map<K,V>`
- Coding to the Interface
- `List<E>`, `Set<E>`, `Queue<E>` and `Map<K,V>` implementations
- Iterating Collections with the `Iterator<E>` Interface
- Collections and the Enhanced for Loop

- Choosing the Correct Implementation and Interface
- The `java.util.Collections` Utility Class
- Sorting Using the Comparable Interface

Exceptions and Exception Handling

- The Throwable Hierarchy: Error, `RuntimeException` and Checked Exception
- Methods that Throw Exceptions
- Handling Exceptions with `try-catch-finally` Blocks
- Application-Defined Exceptions
- Throwing an Exception
- Assertions
- Enabling Assertions at Run-Time

Basic Input and Output (I/O)

- The `java.io` Package
- Using Stream Classes
- Combining Streams
- `flush()` and `close()`
- Console Input and Output
- Navigating the File System
- File Streams
- Character File Input and Output
- Reader and Writer Interfaces
- `BufferedReader` and `BufferedWriter`
- Binary File I/O - `DataOutputStream` and `DataInputStream`
- Object Streams - `ObjectInputStream` and `ObjectOutputStream`
- Serialization and Versioning
- Random Access Files
- Formatted Input and Output
- Formatter
- Format specifiers, `printf()` and `format()`
- `java.text` Classes for Formatting Dates, Numbers, Currencies
- Input with Scanner

Network Programming

- The `java.net` Package
- IP Addresses and Port Numbers
- Client/Server Socket Programming
- `URL` and `URLConnection` Classes
- Communicating with Web Servers
- HTTP `GET` and `POST` Operations
- Posting to a Server-Side Program

Threads

- Life and States of a Thread
- Creating and Starting a Thread
- `java.lang.Runnable` and `java.lang.Thread`
- Stopping a Thread
- Inter-Thread Communication
- Thread-Safe Access to Shared Objects and Variables
- Synchronized Code
- Sleeping

- Interrupting a Blocked Thread
- `wait()`, `notify()`, `notifyAll()` Example
- Thread Scheduling
- Thread Groups
- Writing a Multithreaded Server

Java Database Connectivity

- The `java.sql` Package
- JDBC Architecture and Drivers
- SQL Exceptions
- `DriverManager`, `Connection`, `Statement` and `ResultSet` Interfaces
- Examining Database MetaData
- Basic Query and Update
- Improving Performance with `PreparedStatement` and `CallableStatement` Interfaces
- JDBC Transaction Management

Java Web Applications

- Java Enterprise Edition
- Java EE Application Servers
- Web Application Directory and WAR files
- Deploying a Web Application - The `web.xml` File
- Servlet Architecture
- The `javax.servlet` Package
- Servlet Classes and Interfaces
- Writing a Servlet
- `HttpServletRequest` and `HttpServletResponse`
- Handling HTML Forms
- Retrieving Request Parameters

JavaServer Pages (JSPs)

- JSP Lifecycle
- Elements of a JSP
- Directives, Declarative, Scriptlets
- Writing a JSP
- Objects Available in a JSP
- Repeated content in JSPs
- Translation-Time and Request-Time Includes
- Using JavaBeans in a JSP
- Session Management
- Mixing JSPs and Servlets
- Installing and Using Tag Libraries
- The JSP `taglib` Directive
- The Tag Library Descriptor

admissions@brightstarinstitute.com

Copyright© Bright Star Institute