The Introduction to Java training course provides students with a foundational knowledge of the Java platform and Java language required to build stand-alone Java applications. This course assumes students have a background in another object-oriented programming language, such as C++, or C#.

The Introduction to Java course prepares students to take the Sun Certified Java Associate and Sun Certified Java Programmer exams.

Course Objectives

At the end of the course, students will be able to:

- What is Java?
- Creating your first Java program
- Working with Identifiers, Keywords, and Data
- Types: Primitives, Strings, Objects, Primitive to Object conversion using Auto-boxing, Enumerations
- Controlling program flow with conditions and loops
- Object Oriented Programming with Java: Classes, Objects, States and Behaviors, and Packages
- Detecting and correcting errors through exception handling and assertions
- Managing and manipulating application data using arrays, Collections, and generics
- Parsing and processing files using java.io and regular expressions
- Packaging applications for deployment
- What is “advanced” core Java?
- Advanced OO concepts: Class partitioning for deployment, Inner Classes, Enumerations, Generics
- Customizing the Collections API through custom algorithms and data structures
- Creating multi-threaded programs: Threads, ThreadGroups, Mutual exclusion, Semaphores, etc.
- Interacting with the Virtual Machine through Shut-down hooks
- Forking and managing external processes
- Managing application preferences with the Preferences API
- Creating and invoking dynamic systems at run-time with Reflection and custom Annotations
- Leaving audit trails with the java.util.logging API
- Formatting and localizing complex data
- Synchronous TCP-based programming using java.io and java.net
- Object reference management through java.lang.ref

Target Audience
This course is intended for anyone who is interested in learning Java and J2EE. These developers will be likely to develop enterprise business solutions. Prior programming familiarity or experience in C, C++ preferred. However, it is not a must.

**Syllabus Overview**

**Introduction to Classes and Objects**
- Classes, Objects and Methods
- Object Instances
- Declaring and Instantiating a Java Object
- Declaring Methods
- *set* and *get* Methods
- Initiating Objects with Constructors
- Primitive Types vs. Reference Types

**Writing Methods (Functions)**
- Static vs. Dynamic Allocation
- Declaring Methods
- Declaring Methods with Multiple Parameters
- Method-Call Stack
- Scope of Declarations
- Argument Promotion and Casting
- Designing Methods for Reusability
- Method Overloading

**Deeper Into Classes and Objects**
- Controlling Access to Class Members
- Referencing the Current Object Using *this*
- Overloading Constructors
- Default and No-Argument Constructors
- Composition of Classes
- Garbage Collection and Destructors
- The *finalize* Method
- Static Class Members

**Increasing Convenience by Using Polymorphism**
- Purpose of Polymorphic Behavior
- The Concept of a Signature
- Abstract Classes and Methods
- *final* Methods and Classes
- Purpose of Interfaces
- Using and Creating Interfaces
- Common Interfaces of the Java API
Fundamental Searching and Sorting
- Introduction to Searching Algorithms
  - Linear Search
  - Binary Search
- Introduction to Sorting Algorithms
  - Selection Sort
  - Insertion Sort
  - Merge Sort

Exception Handling
- Types of Exceptions
- Exception Handling Overview
- Exception Class Hierarchy
- Extending Exception Classes
  - When to Throw or Assert Exceptions

Strings, Characters and Regular Expressions
- Fundamentals of Characters and Strings
  - String Class
  - String Operations
  - StringBuilder Class
  - Character Class
  - StringTokenizer Class
- Regular Expressions
  - Regular Expression Syntax
  - Pattern Class
  - Matcher Class

Flow Control
- Conditional Constructs
- Looping Constructs
  - Counter-Controlled Repetition
  - Sentinel-Controlled Repetition
- Nested Control Constructs
  - break and continue Statements
    - Structured Programming Best Practices

Arrays
- Purpose of Arrays
- Declaring and Instantiating Arrays
- Passing Arrays to Methods
- Multidimensional Arrays
- Variable-Length Argument Lists
- Using Command-Line Arguments
  - Using Environment Variables
Defining Classes Using Inheritance
- Superclasses and Subclasses
- Advantages of Using Inheritance
- protected Class Members
  - Constructors in Subclasses

Files and Streams
- Concept of a Stream
- Class File
- Sequential Access
- Object Serialization to/from Sequential Access Files
  - Additional java.io Classes

Fundamental Data Structures
- Dynamic Memory Allocation
- Linked Lists
- Stacks
- Queues
- Trees

The Collections Framework
- The java.util Package
- Container Objects
- Arrays as Containers
- Legacy Container Classes - Vector, Hashtable, Enumeration
- Legacy Container Generic forms (Java SE 5+)
- 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

Formatted Output
- printf Syntax
- Conversion Characters
- Specifying Field Width and Precision
- Using Flags to Alter Appearance
- Printing Literals and Escape Sequences
- Formatting Output with Class Formatter

Threads
- Life and States of a Thread
- Creating and Starting a Thread
Training Delivery Methods

Take this training with one of the following training delivery methods below:

- In-House Training
- Weekend Training
- Live Online Training
- Private 1-on-1 Training

Call us at 949-732-3105

Email us at training@ssinfotek.com