Core Java Syllabus

Overview

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suite various types of platforms. Ex: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere.**

Java is:

- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architectural-neutral:** Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.
- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Course Goals

- Understand fundamentals of programming such as variable, conditional and iterative execution, method, etc.
- Understand fundamentals of object oriented programming in java, including defining classes, invoking method, using class libraries, etc.
- Be aware of important topics and principle of software development.
- Have the ability to write computer program to solve specific problem.
- Be able to use Java SDK environment to create, debug and run simple java program.

✓ .Introduction to java
  - Introduction
  - Basic concepts of OOPs
  - Java History
  - Java Feature
  - Comparison in Java and C++
  - Java Virtual Machine
  - Java Environment
  - Program

✓ Data types
  - Data types
  - Integer data type
  - Floating point data type
  - Character data type
  - Boolean data type
  - Mixing Data types
  - Variables
  - Variable name
  - Constants
  - Integer Constant
  - Real Constant
  - Character Constant

- String Constant
- Symbolic constant
- Backslash character constant
- Comments
- Command line arguments

✓ Operators
- Introduction
- Tokens in Java
- Identifiers
- Litrals
- Keywords
- Operator
- Arithmetic operators
- Logical operators
- Relational operators
- Assignment operators
- Conditional operators
- Increment and decrement operators
- Shift operator
- Separators

✓ Control Structure
- Introduction
- Control structure
- Selection Statement
- if statement
- Simple if statement
- The if...else statement
- Nesting of if-else statement
- switch statement
- Iteration Statement
- for loop
- while loop
- do-while loop
- Jump in Statement

✓ Classes and Objects, Constructors
- Objective
- Class
- Creating —main ll in a separate class
- Methods with parameters
- Methods with a Return Type
- Method Overloading
- Passing Objects as Parameters
- Passing Values to methods and Constructor:
- Access Modifiers

        - Public
        - Private
        - Protected
        - Default

- ✓ OOPS Features in Java
    - Inheritance
    - Encapsulation
    - Polymorphism
    - Abstraction
        - Abstract Class
        - Interfaces
            - Introduction
            - More about interface‘
            - Access
            - Multiple Inheritance
            - Interfaces and Abstract Classes
            - Inheritance within interfaces

- ✓ Nested Classes
    - Nested Class : What and Why?
    - Member Inner class
    - Anonymous Inner class
    - Local Inner class
    - static nested class
    - Nested Interface

- ✓ wrapper classes
    - Integer
    - Float
    - Double
    - Short
    - Byte etc

- ✓ AutoBoxing and UnBoxing

- ✓ String Handling
    - String : What and Why?
    - Immutable String
    - String Comparison
    - String Concatenation
    - Substring
    - Methods of String class
    - StringBuffer class
    - StringBuilder class
    - Creating Immutable class
    - toString method

- StringTokenizer class

✓ Exception Handling
- Exception Handling : What and Why?
- try and catch block
- Multiple catch block
- Nested try
- finally block
- throw keyword
- Exception Propagation
- throws keyword
- Exception Handling with Method Overriding
- Custom Exception

✓ Multithreading
- Multithreading : What and Why?
- Life Cycle of a Thread
- Creating Thread
- Thread Schedular
- Sleeping a thread
- Joining a thread
- Thread Priority
- Daemon Thread
- Thread Pooling
- Thread Group
- ShutdownHook
- Performing multiple task by multiple thread
- Garbage Collection
- Runnable class

✓ Synchronization
- Synchronization : What and Why?
- synchronized method
- synchronized block
- static synchronization
- Deadlock
- Inter-thread Communication
- Interrupting Thread

✓ Input and output
- FileOutputStream & FileInputStream
- ByteArrayOutputStream
- SequenceInputStream
- BufferedOutputStream & BufferedInputStream
- FileWriter & FileReader
- CharArrayWriter
- Input from keyboard by InputStreamReader
- Input from keyboard by Console
- Input from keyboard by Scanner

- PrintStream class
- PrintWriter class
- Compressing and Uncompressing File
- Reading and Writing data simultaneously
- DataInputStream and DataOutputStream
- StreamTokenizer class

✓ Serialization
- Serialization & Deserialization
- Serialization with IS-A and Has-A
- transient keyword

✓ Collection
- Collection Framework
- ArrayList class
- LinkedList class
- ListIterator interface
- HashSet class
- LinkedHashSet class
- TreeSet class
- PriorityQueue class
- ArrayDeque class
- Map interface
- HashMap class
- LinkedHashMap class
- TreeMap class
- Hashtable class
- Comparable and Comparator
- Properties class

✓ Generic
- Generic Class
- Generic Method
- Bounded Type Parameter