Introduction of JAVA

Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.

Introduction of JAVA

- 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]).
- Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.
- The latest release of the Java Standard Edition is Java SE 21.

Introduction of JAVA

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

Features of JAVA

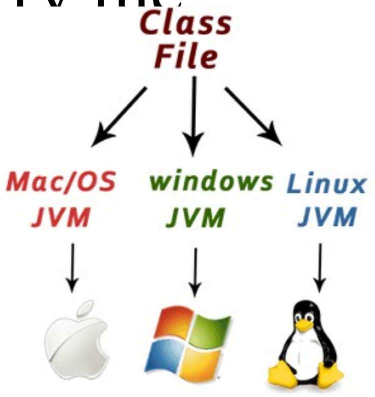
- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

Features of JAVA

- **Architecture-neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

"write once; run anywhere, any time, forever."

- **Portable** – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable.
- Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't requi implementation.



- **Robust** – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime (dynamic) checking.

  Java virtually eliminates these problems by managing memory allocation and deallocation for you.
- **Multithreaded** – With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

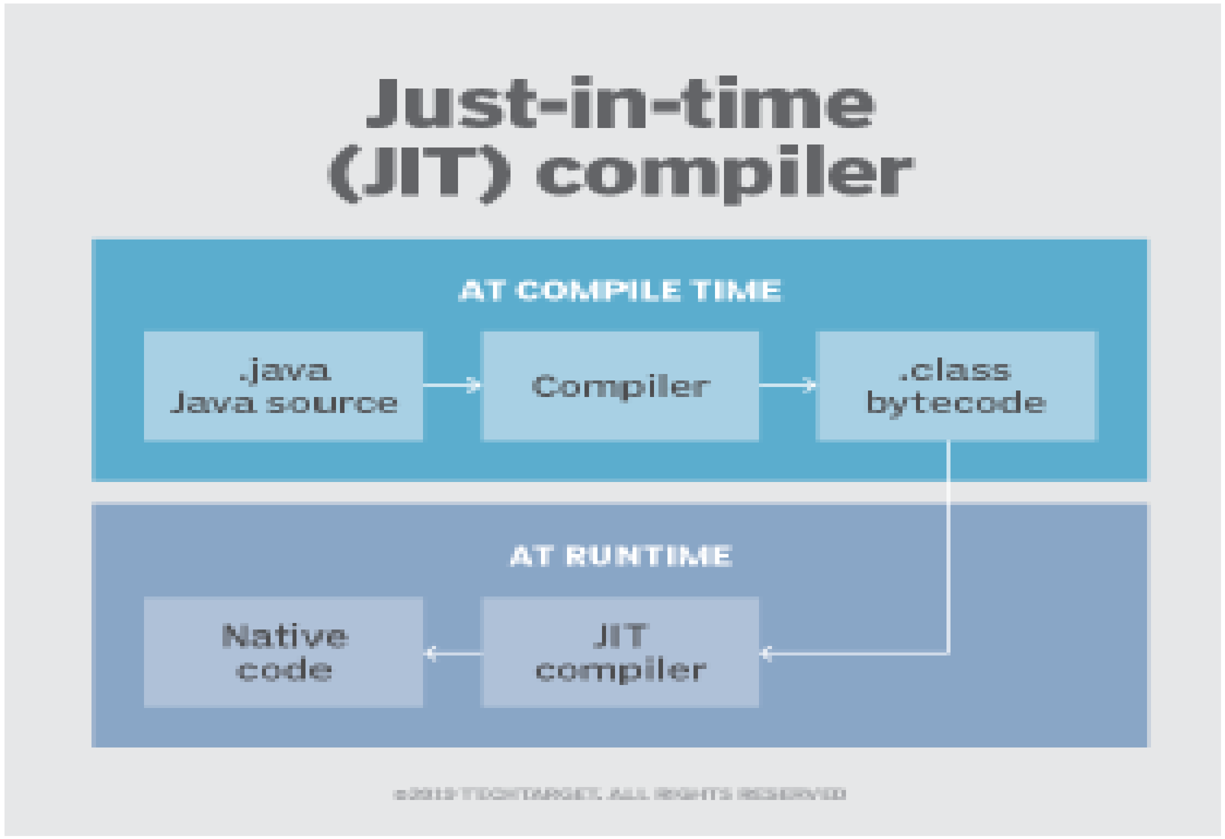
Features of JAVA

- **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 light-weight process. Source code is first compiled into a binary byte-code
- **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
- The Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance of Java™ applications by compiling bytecodes to native machine code at run time

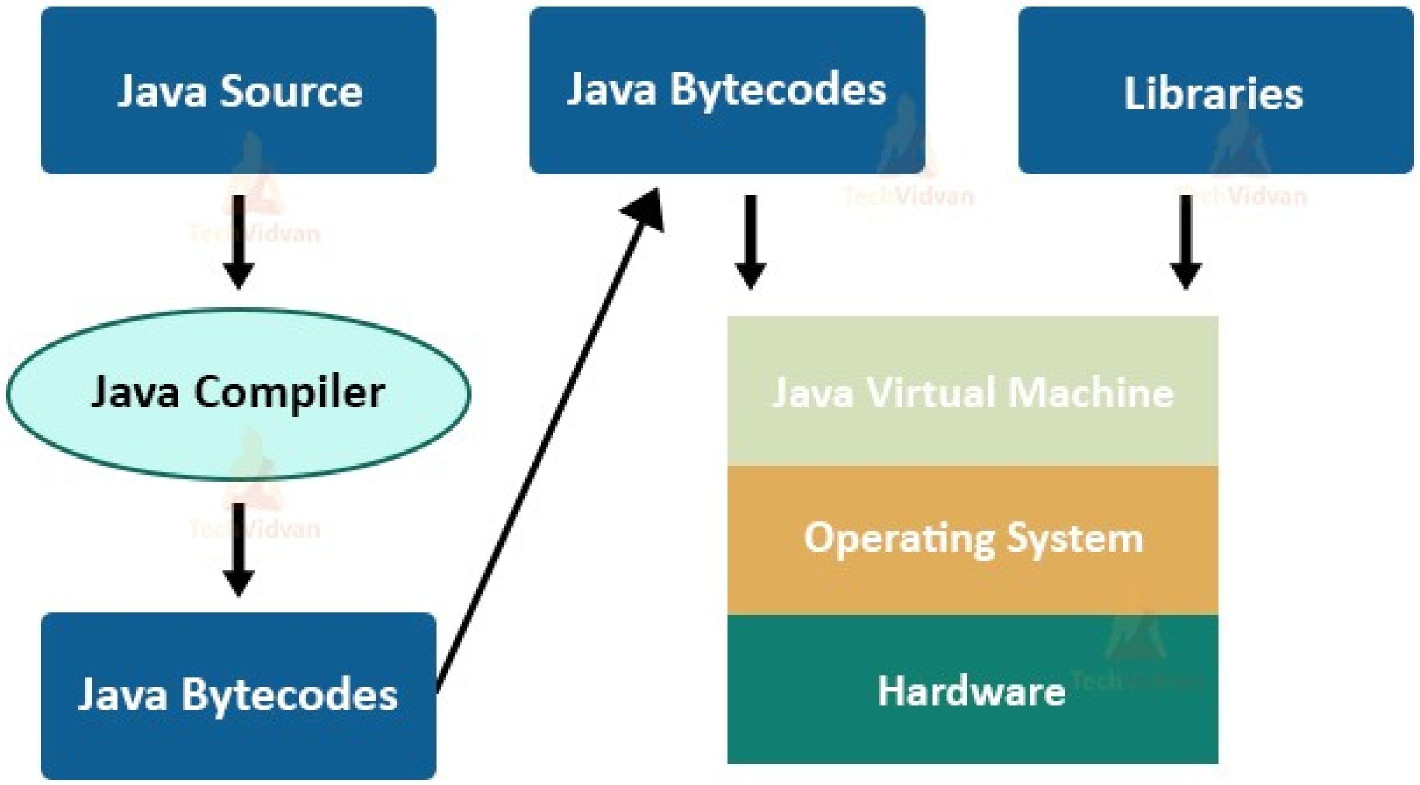
Features of JAVA

- **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.



A Java Virtual Machine (JVM) is a program that interprets Java bytecode to run as a program by providing a runtime environment that executes this process.

Working of JVM



History of Java

- James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects.
- The language, initially called 'Oak' after an oak tree that stood outside Gosling's office, also went by the name 'Greentalk' and ended up later being renamed as Java, from a list of random words.
- Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.
- On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).
- On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

How JAVA differs from C & C++

Java and C

- JAVA is Object-Oriented while C is procedural.
- Java is an Interpreted language while C is a compiled language.
- C is a low-level language while JAVA is a high-level language
- C uses the top-down approach while JAVA uses the bottom-up approach.
- The Memory Management (Garbage Collection) with JAVA & The User-Based Memory Management in C.
- Unlike C, JAVA does not support Preprocessors,
- Exception Handling in JAVA and the errors & crashes in C.

How JAVA differs from C & C++

Java and C++

- C++ supports multiple inheritance. Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
- C++ is mainly used for system programming. Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
- C++ supports operator overloading. Java doesn't support operator overloading.
- C++ uses compiler only. Java uses compiler and interpreter both.

How JAVA differs from C & C++

Java and C++

- C++ supports both call by value and call by reference. Java supports call by value only. There is no call by reference in java.
- C++ supports structures and unions. Java doesn't support structures and unions.
- Java Doesn’t have Destructor like C++   Instead Java Has finalize Method

Application

- Desktop Applications such as acrobat reader, media player, antivirus, etc.
- Web Applications such as irctc.co.in, javatpoint.com, etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc.

Types of Java Applications

- Standalone Application
- Web Application
- Enterprise Application
- Mobile Application

Java Platforms / Editions

- Java SE (Java Standard Edition)
- Java EE (Java Enterprise Edition)
- Java ME (Java Micro Edition)
- JavaFX (EFF-ECTS)

Java Environment

JDK contains

- Appletviewer (for viewing applets)
- javac (Java Compiler)
- java (Java Interpreter)
- javap (Java disassembler)
- javah (for C header files)
- javadoc (creating HTML documents)
- jdb (Java Debugger)

First Java Program

```
 public class MyFirstJavaProgram
{
            public static void main(String []args)
            {
                        System.out.println("Hello World");
               }
}
```

- The public keyword is an access modifier, which allows the programmer to
- control the visibility of class members.
- The keyword static allows main( ) to be called without having to instantiate a particular instance of the class. This is necessary since main( ) is called by the Java Virtual Machine before any objects are made
- Java is case-sensitive.
- , args receives any command-line arguments present when the program is executed.

Fundamental Concepts of JAVA

- Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts –
- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Dynamic Binding
- Message Passing

Fundamental Concepts of JAVA

- **Object** – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
- **Class** – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

Fundamental Concepts of JAVA

- **Classes in Java**
- A class is a blueprint from which individual objects are created.

```
    public class Dog
    {           String breed;
                String color;
                 void barking(){ }
                void hungry()
                { }
                 void sleeping()
                { }
  }
```

```
 Public class Student
  {
     Int rollno;
     String name;
      Int marks;
       Void displaydetail() {}
        Float cal_percentage(){}
}
```

Fundamental Concepts of JAVA

- A class can contain any of the following variable types.
- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

Fundamental Concepts of JAVA

- **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

Constructors

- When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.
- Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Object

- **Creating an Object**
- An object is created from a class. In Java, the new keyword is used to create new objects.
- There are three steps when creating an object from a class −
- **Declaration** − A variable declaration with a variable name with an object type.
- **Instantiation** − The 'new' keyword is used to create the object.
- **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

  ClassName object = new ClassName();

Object

- **Accessing Instance Variables and Methods**
- Instance variables and methods are accessed via created objects. To access an instance variable, following is the fully qualified path −

/* First create an object */

ObjectReference = new Constructor();

/* Now call a variable as follows*/ ObjectReference.variableName;

/* class method as follows */ ObjectReference.MethodName();

obj.length; obj.show();

Object

```
public class Hello
{
        void show()
        {
              System.out.println("Welcome to java");
         }
        public static void main(String[] args)
        {
              //creating an object using new keyword
              Hello obj = new Hello();
              //invoking method using the object
              obj.show();
        }
}
```

```
class Student{
    int id;//field or data member or instance variable
    String name;
    Void display()
     {

   System.out.println(id);//accessing member through reference variable

    System.out.println(name);
   }
    public static void main(String args[]){
    //Creating an object or instance
  Student s1=new Student();//creating an object of Student
    S1.display();
 }
}
```

- Java is a whole platform, with a huge library, containing lots of reusable code, and an execution environment that provides services such as security, portability across operating systems, and automatic garbage collection.
- Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

- Java has everything
  - a good language, a high-quality execution environment, and a vast library. That combination is what makes Java an irresistible proposition to so many programmers.