**Java** was developed by James Gosling, who is known as the father of Java, in 1995

**What is Java?**

- Java technology is both a programming language and a platform.

**Java Programming language :**

* Java programming is a *high-level language* that can be characterised by all of the following buzzwords:

1. Simple
2. Object oriented
3. Distributed
4. Multithreaded
5. Dynamic
6. Architecture neutral
7. Portable
8. High performance
9. Robust
10. Secure

* Java programming language platform from *Sun Microsystems*.
  + Your programming language is ***object oriented***, yet it's still dead ***simple***.
  + Your development cycle is much ***faster*** because Java technology is ***interpreted***. The compile-link-load-test-crash-debug cycle is obsolete—now you just compile and run.
  + Your applications are ***portable*** across multiple platforms. Write your applications once, and you never need to port them--they will run without modification on multiple OS and hardware architectures.
  + Your applications are ***robust*** because the ***Java runtime environment manages memory*** for you.
  + Your interactive graphical applications have ***high performance*** because multiple concurrent threads of activity in your application are supported by the ***multithreading*** built into the Java programming language and runtime platform.
  + Your applications are ***adaptable*** to changing environments because you can ***dynamically*** download code modules from anywhere on the network.
  + Your end users can trust that your applications are ***secure***, even though they're downloading code from all over the Internet; ***the Java runtime environment has built-in protection against viruses and tampering***.

***Design requirements of the Java programming language*** are driven by the ***nature of the computing environments*** in which software must be deployed.

***IMAGE FROM ORACLE SITE REPRESENTING DEVELOPMENT PROCESS USING JAVA*** 



***Java VM*** is available on different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.

**The Java Platform**

A ***platform*** is the ***hardware or software environment in which a program runs***. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The ***Java platform*** differs from most other platforms in that it's a ***software-only platform that runs on top of other hardware-based platforms***.

The Java platform has two components:

* The *Java Virtual Machine*
* The *Java Application Programming Interface* (API)

As a ***platform-independent*** environment, the Java platform can be a ***bit slower than native code***. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

# **What Can Java Technology Do?**

The general-purpose, high-level Java programming language is a powerful software platform. Every full implementation of the Java platform gives you the following features:

* **Development Tools**: The development tools provide everything you'll need for compiling, running, monitoring, debugging, and documenting your applications. As a new developer, the main tools you'll be using are the javac compiler, the java launcher, and the javadoc documentation tool.
* **Application Programming Interface (API)**: The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access, and more. The core API is very large.
* **Deployment Technologies**: The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying your applications to end users.
* **User Interface Toolkits**: The JavaFX, Swing, and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).
* **Integration Libraries**: Integration libraries such as the Java IDL API, JDBC API, Java Naming and Directory Interface (JNDI) API, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.

# **How Will Java Technology Change My Life?**

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program written in C++.
* **Write better code**: The Java programming language encourages good coding practices, and automatic garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans™ component architecture, and its wide-ranging, easily extendible API let you reuse existing, tested code and introduce fewer bugs.
* **Develop programs more quickly**: The Java programming language is simpler than C++, and as such, your development time could be up to twice as fast when writing in it. Your programs will also require fewer lines of code.
* **Avoid platform dependencies**: You can keep your program portable by avoiding the use of libraries written in other languages.
* **Write once, run anywhere**: Because applications written in the Java programming language are compiled into machine-independent bytecodes, they run consistently on any Java platform.
* **Distribute software more easily**: With Java Web Start software, users will be able to launch your applications with a single click of the mouse. An automatic version check at startup ensures that users are always up to date with the latest version of your software. If an update is available, the Java Web Start software will automatically update their installation.

Checklist to write Java Program:

* Java SE Development kit
* IDE(Integrated Development Environment) / Text Editor

\* IDE provides options to compile and run Program.

Programs written through text editors should be saved with *.java* extension and executed on terminal using below commands

* *javac* – used to *compile* java file (Eg : javac a.java)
* *java* – used to *execute*(interpret) byte code (Eg : java a)

Both the compiler (javac) and launcher (java) are case-sensitive

Sample Java program :

class HelloWorldApp {

public static void main(String[] args) {

System.out.println("Hello World!"); // Display the string.

}

}

T he "Hello World!" application consists of three primary components:

* source code comments ( // )
* HelloWorldApp class definition
* Main method

Source Code comments:

* Comments get ignored by compiler but are useful for other programmers to understand your code.
* Java has 3 types of comment styles:
  + Multiline comments : /\* to \*/
  + Documentation : /\*\* doc comments \*/. Used by Javadoc tool to generated documentation for code.
  + Single line comments : //comment.

HelloworldApp class Definition:

* The keyword class begins the class definition for a class named name, and the code for each class appears between the opening and closing curly braces.

Main method:

* **main** represents the starting point of the program.
* **public** keyword is an access modifier which represents visibility. It means it is visible to all.
* **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.
* **void** is the return type of the method. It means it doesn't return any value.
* **String[] args** is used for command line argument.
* **System.out.println()** is used to print statement. Here, System is a class, out is the object of PrintStream class, println() is the method of PrintStream class.

**Observations:**

1. The modifiers public and static can be written in either order (public static or static public), but the convention is to use public static.
2. You can name the argument anything you want, but most programmers choose "args" or "argv".
3. Argument array is the mechanism through which the runtime system passes information to your application. For example: java MyApp arg1 arg2

Questions :

**Question** 1 :

When you compile a program written in the Java programming language, the compiler converts the human-readable source file into platform-independent code that a Java Virtual Machine can understand. What is this platform-independent code called?

**Ans** : Bytecode

**Question 2**:

Which of the following is *not* a valid comment:

a. /\*\* comment \*/ b. /\* comment \*/ c. /\* comment d. // comment

**Ans** : c

**Question 3:**

What is the first thing you should check if you see the following error at runtime:

Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorldApp.java.

**Ans**: Check your classpath. Your class cannot be found.

**Question 4**:

What are the valid signatures of the main method?

**Ans** :

public static void main(String[] args){}

public static void main(String args[]) {}

public static void main(String… args) {}

public static void main(String []args) {}

static public void main(String[] args) {}

public static final void main(String[] args) {}

final public static void main(String[] args) {}

final strictfp public static void main(String[] args) {}

**Question 5**:

What are the invalid signatures of the main method?

**Ans** :

public void main(String[] args) {}

static void main(String[] args) {}

public void static main(String[] args) {}

abstract public static void main(String[] args) {}

**Question 6:**

When declaring main method which modifier must come first , public or static?

**Ans** : They can be in any order, but the convention is public static.

**Question 7**:

What are the parameters of main method?

**Ans :** They represent command line arguments. They take array type arguments.