**What is Java?**

* Java is an object-oriented programming language.
* It treats everything as an object in the real world.
* It is a high-level programming language.
* It is robust and very secure.
* It uses English language.
* Java can be run in windows, mac and OS and various platforms.
* In note pad we write the programme and is executed using command
* prompt.
* More than 3 billion devices run java programme.
* Java was designed for embedded system.

**History of java.**

* Java was developed by sun microsystems(which is now the subsidiary of Oracle) in the year 1995.
* Games Gosling is **known** as the father of Java.
* Before Java, its name was oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

**Features of Java.**

1. [Simple](https://www.javatpoint.com/features-of-java#Simple)
2. [Object-Oriented](https://www.javatpoint.com/features-of-java#Object-Oriented)
3. [Portable](https://www.javatpoint.com/features-of-java#Portable)
4. [Platform independent](https://www.javatpoint.com/features-of-java#Platform-independent)
5. [Secured](https://www.javatpoint.com/features-of-java#Secured)
6. [Robust](https://www.javatpoint.com/features-of-java#Robust)
7. [Architecture neutral](https://www.javatpoint.com/features-of-java#Architecture-neutral)
8. [Interpreted](https://www.javatpoint.com/features-of-java#Interpreted)
9. [High Performance](https://www.javatpoint.com/features-of-java#High-Performance)
10. [Multithreaded](https://www.javatpoint.com/features-of-java#Multithreaded)
11. [Distributed](https://www.javatpoint.com/features-of-java#Distributed)
12. Dynamic.

**Simple**

* java is very easy to learn, and its syntax is simple, clean and easy to understand.:
* Java syntax is based on C++ (so easier for programmers to learn it after C++).
* Java has removed many complicated and rarely used features, for example, explicit pointers, operator overloading, etc.
* There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

**Object oriented**

* Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language.
* Everything in Java is an object.
* Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

**Platform independent**

* Java is platform independent because it is different from other languages like c and c++ etc. which are compiled into platform specific machines while Java is a write once, run anywhere language.
* Java provides a software-based platform.
* Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc.
* Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere.

**Security**

* java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:
* No explicit pointer
* Java Programs run inside a virtual machine sandbox.

**Robust**

* The English meaning of Robust is strong.
* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
* There are exception handling and the type checking mechanism in Java. All these points make Java robust.

**Architecture neutral**

* Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
* In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But java occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

**Portable**

* Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.

It is still a little bit slower than a compiled language (e.g., C++).

Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

**Distributed**

* Java is distributed because it facilitates users to create distributed applications in Java.
* This feature of Java makes us able to access files by calling the methods from any machine on the internet.

**Multi-threaded**

* A thread is like a separate program, executing concurrently.
* We can write Java programs that deal with many tasks at once by defining multiple threads.
* The main advantage of multi-threading is that it doesn't occupy memory for each thread.
* It shares a common memory area.
* Threads are important for multi-media, Web applications, etc.

**Dynamic**

* Java is a dynamic language.
* It supports the dynamic loading of classes. It means classes are loaded on demand.
* It also supports functions from its native languages, i.e., C and C++.

**How to set path**

Download java. Open c drive -> program files -> java->jdk.11.0.17->bin and copy the path.

Go to My Computer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok

**Compile error.**

* Compile-time errors (or compilation errors) occur when the code you have written does not conform to the rules of the programming language and cannot be translated into machine code by the compiler.
* Examples of compile-time errors include syntax errors, type errors, and referencing undefined variables or methods. These errors are usually caught by the compiler during the compilation process, and the code will not be able to run until these errors are fixed.

**Runtime error**

* Runtime errors (or execution errors) occur when the code is being executed and something goes wrong at runtime.
* These errors are typically caused by logical errors in the code, such as dividing by zero, referencing null pointers, or going out of bounds of an array.
* Runtime errors are not caught by the compiler and can cause the program to crash or produce unexpected behaviour.