**Java Core**

Java is a class-based, **object-oriented programming language** that is designed to have as few implementation dependencies as possible. It is intended to let application developers **Write Once and Run Anywhere (WORA)**, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

Java is known for its simplicity, robustness, and security features, making it a popular choice for enterprise-level applications. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is similar to C/C++.

## Key Features of Java

### **1. Platform Independent**

**Compiler** converts source code to [**byte code**](https://www.geeksforgeeks.org/byte-code-in-java/)and then the JVM executes the bytecode generated by the compiler. This byte code can run on any platform be it Windows, Linux, or macOS which means if we compile a program on **Windows**, then we can run it on**Linux** and vice versa. Each operating system has a different**JVM**, but the output produced by all the OS is the same after the execution of the **byte code**. That is [**why we call java a platform-independent language.**](https://www.geeksforgeeks.org/java-platform-independent/)

### **2. Object-Oriented Programming**

**Java**is an [**object-oriented language**](https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/), promoting the use of**objects** and **classes**. Organizing the program in the terms of a collection of objects is a way of object-oriented programming, each of which represents an instance of the class.

The**four main concepts of Object-Oriented programming** are:

* [***Abstraction***](https://www.geeksforgeeks.org/abstraction-in-java-2/)
* [***Encapsulation***](https://www.geeksforgeeks.org/encapsulation-in-java/)
* [***Inheritance***](https://www.geeksforgeeks.org/inheritance-in-java/)
* [***Polymorphism***](https://www.geeksforgeeks.org/polymorphism-in-java/)

### 3. Simplicity

[**Java’s syntax**](https://www.geeksforgeeks.org/java-basic-syntax/) is simple and easy to learn, especially for those familiar with **C** or **C++**. It eliminates complex features like pointers and multiple inheritances, making it easier to ***write, debug,***and ***maintain code.***

### 4. Robustness

**Java language is robust which means reliable**. It is developed in such a way that it puts a lot of effort into checking errors as early as possible, that is why the java compiler is able to detect even those errors that are not easy to detect by another programming language. The main features of java that make it robust are garbage collection, exception handling, and memory allocation.

### 5. Security

**In java, we don’t have pointers**, so we cannot access [**out-of-bound arrays**](https://www.geeksforgeeks.org/array-index-out-of-bounds-exception-in-java/)i.e it shows **ArrayIndexOutOfBound Exception** if we try to do so. That’s why several security flaws like stack corruption or buffer overflow are impossible to exploit in Java. Also, java programs run in an environment that is independent of the **os(operating system)** environment which makes java programs more secure.

### **6.** **Distributed**

**We can create distributed applications using the java programming language.** Remote Method Invocation and Enterprise Java Beans are used for creating distributed applications in java. The java programs can be easily distributed on one or more systems that are connected to each other through an internet connection.

### **7.** **Multithreading**

[**Java supports multithreading**](https://www.geeksforgeeks.org/multithreading-in-java/), enabling the**concurrent execution** of multiple parts of a program. This feature is particularly useful for applications that require high performance, such as games and real-time simulations.

### 8. Portability

As we know, java code written on one machine can be run on another machine. The platform-independent feature of java in which its platform-independent bytecode can be taken to any platform for execution makes java portable. [**WORA(Write Once Run Anywhere)**](https://www.geeksforgeeks.org/why-is-java-write-once-and-run-anywhere/)makes java application to generates a [**‘.class’ file**](https://www.geeksforgeeks.org/java-class-file/) that corresponds to our applications(program) but contains code in binary format. It provides architecture-neutral ease, as bytecode is independent of any machine architecture. It is the primary reason java is used in the enterprising IT industry globally worldwide.

### **9. High Performance**

**Java architecture** is defined in such a way that it reduces overhead during the runtime and at some times java uses[**Just In Time (JIT) compiler**](https://www.geeksforgeeks.org/just-in-time-compiler/)where the compiler compiles code on-demand basis where it only compiles those methods that are called making applications to execute faster.

## Essential Java Terminologies You Need to Know

Before learning Java, one must be familiar with these common terms of Java.

### **1. Java Virtual Machine(JVM)**

The [**JVM**](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) is an integral part of the**Java platform**, responsible for executing Java bytecode. It ensures that the output of Java programs is consistent across different platforms.

* Writing a program is done by a java programmer like you and me.
* The compilation is done by the **JAVAC** compiler which is a primary Java compiler included in the Java development kit (JDK). It takes the Java program as input and generates bytecode as output.
* In the Runningphase of a program,**JVM** executes the bytecode generated by the compiler.

The Java Virtual Machine (JVM) is designed to run the bytecode generated by the Java compiler.Each operating system has its own version of the JVM, but all JVMs follow the same rules and standards. This means Java programs can run the same way on any device with a JVM, regardless of the operating system**.** This is why Java is called a platform-independent language.

### **2. Bytecode**

**Bytecode** is the intermediate representation of Java code, generated by the Java compiler. It is platform-independent and can be executed by the JVM.

### **3. Java Development Kit(JDK)**

While we were using the term[**JDK**](https://www.geeksforgeeks.org/jdk-in-java/) when we learn about bytecode and JVM. So, as the name suggests, it is a complete Java development kit that includes everything including **compiler,** **Java Runtime Environment (JRE)**, J**ava Debuggers**, J**ava Docs**, etc. For the program to execute in java, we need to install JDK on our computer in order to create, compile and run the java program.

### **4. Java Runtime Environment (JRE)**

JDK includes JRE. [**JRE**](https://www.geeksforgeeks.org/jre-in-java/) installation on our computers allows the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet support, and plugins. For running the java program, a computer needs JRE.

### **5. Garbage Collector**

In Java, programmers can’t delete the objects. To delete or recollect that memory JVM has a program called [**Garbage Collector**](https://www.geeksforgeeks.org/garbage-collection-java/). Garbage Collectors can recollect the objects that are not referenced. So**Java** makes the life of a programmer easy by **handling memory management**. However, **programmers should be careful about their code whether they are using objects that have been used for a long time**. Because Garbage cannot recover the memory of objects being referenced.

### **6. ClassPath**

The **[Classpath](https://www.geeksforgeeks.org/classpath-in-java/)**is the file path where the java runtime and Java compiler look for **.class** files to load. By default, JDK provides many libraries. If you want to include external libraries they should be added to the classpath.

Basically everything in java is represented in Class as an object including the main function.

## Advantages of Java

* **Platform independent:**Java code can run on any platform that has a**Java Virtual Machine (JVM)** installed, which means that applications can be written once and run on any device.
* **Object-Oriented:** Java is an object-oriented programming language, which means that it follows the principles of encapsulation, inheritance, and polymorphism.
* **Security:** Java has built-in security features that make it a secure platform for developing applications, such as automatic memory management and type checking.
* **Large community:** Java has a large and active community of developers, which means that there is a lot of support available for learning and using the language.
* **Enterprise-level applications:**Java is widely used for developing enterprise-level applications, such as web applications, e-commerce systems, and database systems.

## Disadvantages of Java

1. **Performance:**Java can be slower compared to other programming languages, such as C++, due to its use of a virtual machine and automatic memory management.
2. **Memory management:** Java’s automatic memory management can lead to slower performance and increased memory usage, which can be a drawback for some applications.

**1. How Java Works?**

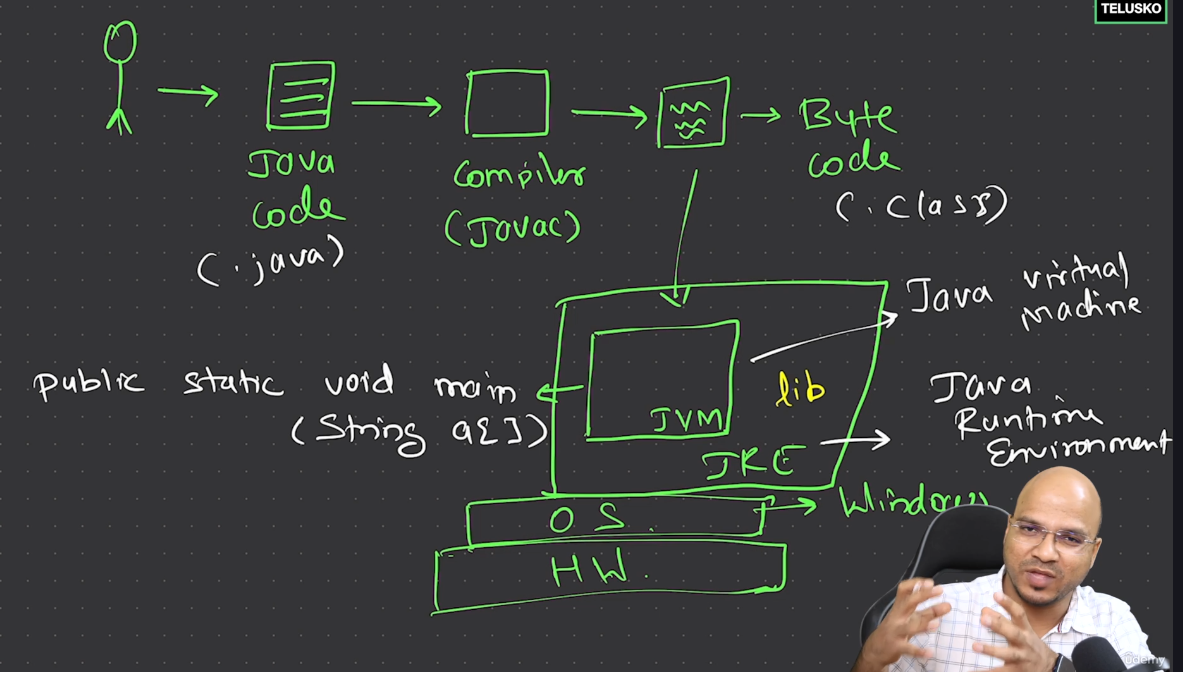
public class \_1\_Hello {

    public static void main(String[] args) {

        System.out.println("Hello, World!");

    }

}

****

### **Java Execution Process (Explained with Diagram)**

#### 1. **Writing Code**

* You write Java code (.java file), e.g., \_1\_Hello.java.
* This file contains the main() method, which is the entry point for execution.

#### 2. **Compiling Code**

* You use the **Java Compiler (javac)** to compile the code:

bash

CopyEdit

javac \_1\_Hello.java

* This converts the code into **Bytecode** (.class file) which is platform-independent.

#### 3. **Bytecode**

* The compiled .class file contains instructions in Bytecode format.
* This bytecode is not directly executed by the operating system.

#### 4. **JRE (Java Runtime Environment)**

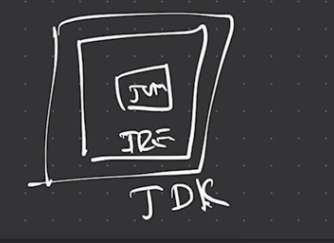
* The JRE contains:
  + **JVM (Java Virtual Machine)** — executes bytecode.
  + **Libraries (lib)** — support runtime functionalities.

#### 5. **JVM**

* The **JVM** reads the .class bytecode and **interprets or JIT-compiles** it to native code.
* The native code is then executed on the **underlying OS (like Windows)**, which runs on **hardware (HW)**.

#### 6. **Output**

* The output "Hello, World!" is printed to the console via System.out.println

****

**2. Variables**

variables are containers that store data in memory. Variable: is used initialize and store data.

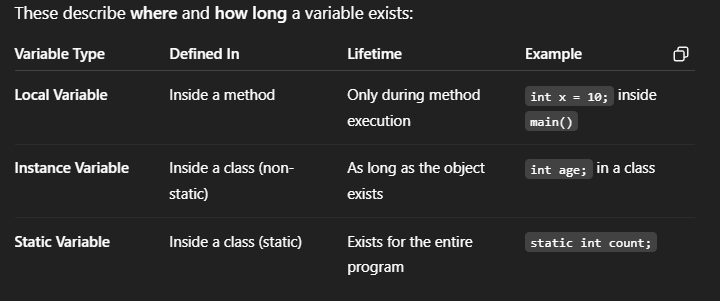
**Key Components of Variables in Java:**

A variable in Java has three components, which are listed below:

* **Data Type:**Defines the kind of data stored (e.g., int, String, float).
* **Variable Name:**A unique identifier following Java naming rules.
* **Value:** The actual data assigned to the variable.

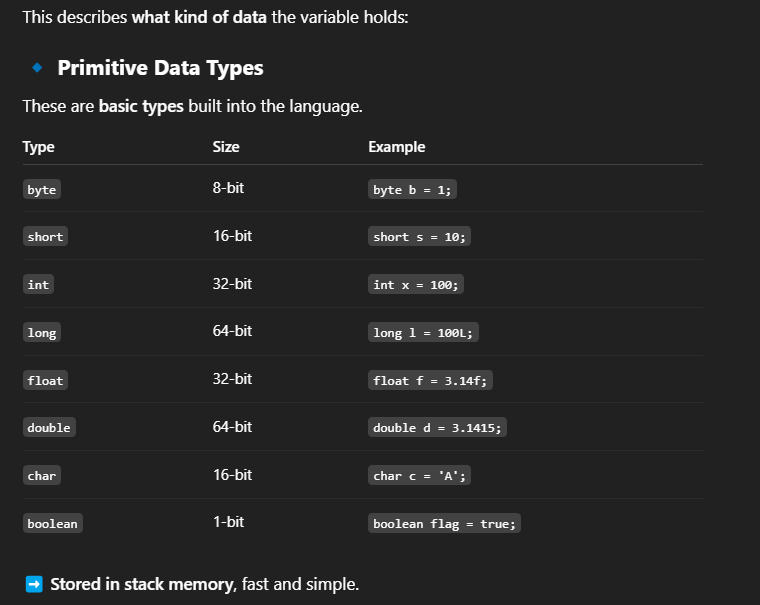
**There are three types of variables in Java:**

**1. Types of Variables in Java (Based on Scope)**

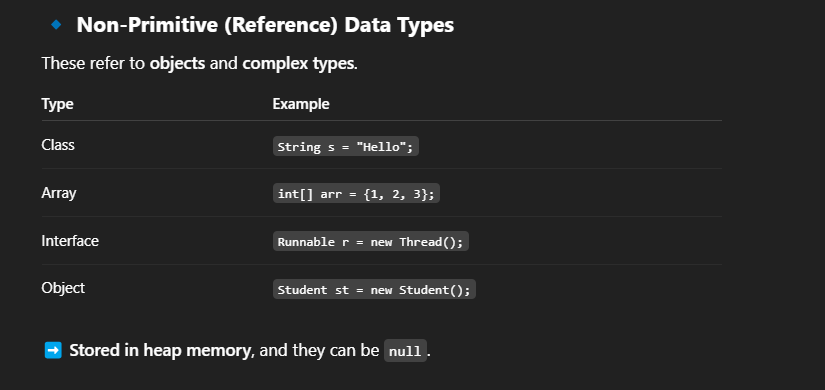
****

**2. Types of Data in Java (Based on Nature)**

**>** Primitive Data Types

****

* + Non-Primitive (Reference) Data Types





### Quick Notes:

* **Local variables** must be initialized before use.
* **Instance variables** are tied to objects (new keyword).
* **Static variables** are shared across all instances of the class.
* **Primitive types** hold the actual value.
* **Non-Primitive types** hold references (pointers) to objects in memory.

**Type casting in Java:**

**Implicit casting(widening)**

Byte -> short->int->long->float->double ->

byte = 8bit (integer type)

short = 16bit (integer type)

int = 32bit(integer type)

long = 64bit(integer type)

float= 32 bit (floating type)

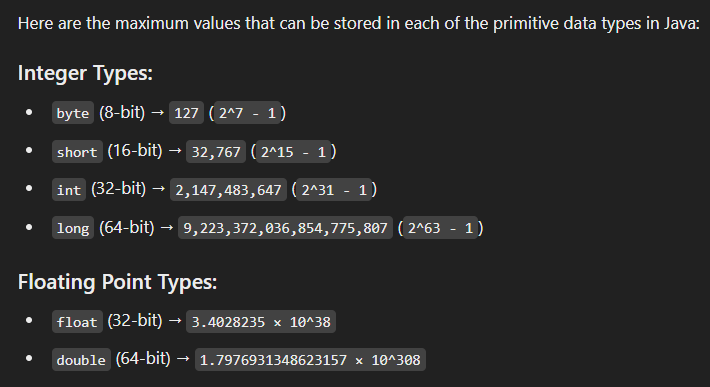
double = 64 bit(floating type)

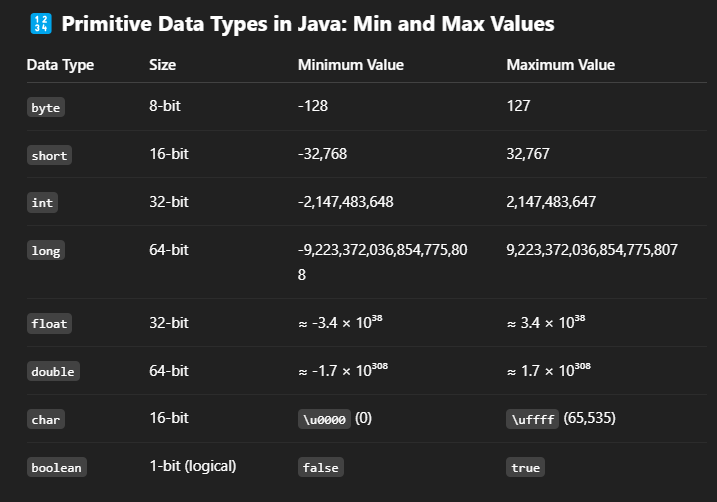
char = 16bit (UNICODE)

Boolean = True/False

**Explicitly casting (Narrowing)**

double -> float-> long -> int -> short -> byte ->



****

**3. Data types:**