# What is Java?

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

# Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995. James 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.

# Platform: 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.

# Features of the Java?

# Java is a popular and widely-used programming language known for its platform independence, simplicity, robustness, and versatility. Here are some of the key features of Java:

# Platform independence: Java is designed to be "write once, run anywhere" (WORA). It achieves this through its "Java Virtual Machine" (JVM), which allows Java programs to be executed on any platform that has a compatible JVM, regardless of the underlying hardware and operating system.

# Object-oriented: Java is a fully object-oriented programming language. It follows the object-oriented programming paradigm and supports concepts such as classes, objects, inheritance, encapsulation, and polymorphism.

# Simple and easy-to-learn: Java was designed to be simple and easy to pick up for developers. Its syntax is similar to C/C++, making it familiar to many programmers.

# Robust and secure: Java's robustness comes from its strong type-checking, automatic memory management through garbage collection, and exception handling mechanisms. The language also has built-in security features to prevent unauthorized access to resources.

# Multi-threaded and concurrent: Java supports multithreading, allowing developers to create multi-threaded applications that can perform multiple tasks concurrently. This is particularly useful for performance optimization and handling concurrent operations.

# Platform libraries (APIs): Java comes with a vast standard library (Java API) that provides many pre-built classes and methods for various tasks like I/O, networking, database connectivity, GUI development, etc. This helps in speeding up development and reduces the need for writing low-level code from scratch.

# Performance: While Java might not be as fast as some lower-level languages like C or C++, it still offers good performance due to modern JVM optimizations and just-in-time (JIT) compilation.

# Architecture-neutral: Java code is compiled into an intermediate form called bytecode, which is architecture-neutral. This bytecode can be run on any JVM, making it platform-independent.

# Garbage collection: Java's automatic garbage collection takes care of memory management, freeing developers from explicitly managing memory allocation and deallocation.

# Networking capabilities: Java has strong support for networking, making it easy to develop network-based applications, like web servers, web clients, etc.

# Open-source: Java itself, along with much of its ecosystem, is open-source, which encourages community collaboration, rapid development, and extensive third-party libraries.

# Community support: Java has a massive and active community of developers, which means there is a wealth of online resources, tutorials, and forums available for learning and problem-solving.

# Overall, Java's combination of platform independence, object-oriented approach, robustness, and community support has made it a popular choice for a wide range of applications, from desktop software to web development, mobile apps, enterprise solutions, and beyond.

# JDK, JRE, JVM in Java?

# JDK (Java Development Kit):

# The JDK stands for Java Development Kit. It is a software package that includes all the tools and utilities required for Java application development. JDK provides a complete set of development tools, including the Java compiler (javac), the Java Runtime Environment (JRE), and various utilities like debugger, profiler, documentation generator, etc.

# Developers use the JDK to write, compile, and debug Java programs. It contains the necessary libraries, header files, and development tools to create Java applications. If you want to develop Java applications, you need to install the JDK on your system.

# JRE (Java Runtime Environment):

# The JRE stands for Java Runtime Environment. It is a runtime environment that is required to run Java applications on a computer. JRE includes the Java Virtual Machine (JVM) and the core class libraries that are necessary for running Java programs.

# End-users who only need to run Java applications (not develop them) should have JRE installed on their machines. JRE allows users to execute Java programs without having to worry about the development tools.

# JVM (Java Virtual Machine):

# The JVM stands for Java Virtual Machine. It is a crucial part of the Java Runtime Environment (JRE) and JDK. The primary function of the JVM is to execute Java bytecode, which is an intermediate representation of Java source code.

# When you compile a Java source code file (.java) using the Java compiler (javac), it generates bytecode (.class files). The JVM is responsible for interpreting and executing this bytecode on the specific operating system and hardware. It provides platform independence by allowing the same Java bytecode to run on any system with a compatible JVM.

# To summarize:

# JDK: Used for Java application development, includes JRE, and development tools like compiler, debugger, etc.

# JRE: Used for running Java applications, includes JVM and core class libraries.

# JVM: Executes Java bytecode and provides platform independence by interpreting bytecode on different systems.

# In many cases, when you install the JDK, it will also include the JRE components, so you have everything needed for both development and execution of Java applications.

# WHATS IS , public static void main(String[] args) ?

# In Java programming, public static void main(String[] args) is a special method that serves as the entry point of a Java application. When you run a Java program, the Java Virtual Machine (JVM) looks for this method and starts executing the code inside it. Let's break down each part of this signature.

# public: This is an access modifier, which means the main method can be accessed from outside the class. It allows the JVM to call this method from any other class.

# static: This is a keyword that indicates the main method belongs to the class itself, not to an instance (object) of the class. This is necessary because the JVM calls the main method without creating an object of the class containing the main method.

# void: This keyword indicates that the main method does not return any value. In Java, a method with a return type of void means it doesn't produce a result when executed.

# main: This is the name of the method. It is the standard name recognized by the JVM as the entry point of the Java program.

# (String[] args): This is the parameter list of the main method. Parameters are inputs to the method. In this case, the main method takes a single parameter, which is an array of strings called args. The args parameter allows you to pass command-line arguments to your Java program when you run it from the command line.

# SAMPLE CODE TO PRINT SOMETHING?

# public class PrintSomething {

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

# // Replace "Hello, World!" with whatever you want to print

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

# }

# }

# In this code, we have a class named PrintSomething with a main method. The main method is the entry point of the program, where the execution begins. Inside the main method, we use System.out.println() to print the text "Hello, World!" to the console.

# Compile and run the above code, and it will print "Hello, World!" on the console.

# Java data types?

# In Java, data types define the type and size of values that can be stored in variables. Java has two categories of data types: primitive data types and non primitive data types or reference data types .

# Primitive Data Types:

# Primitive data types are the basic data types provided by Java, and they are not objects. They are used to store simple values. Java has eight primitive data types:

# byte: 8-bit signed integer. Range: -128 to 127.

# short: 16-bit signed integer. Range: -32,768 to 32,767.

# int: 32-bit signed integer. Range: -2^31 to 2^31 - 1.

# long: 64-bit signed integer. Range: -2^63 to 2^63 - 1.

# float: 32-bit floating-point number. Suitable for representing decimal numbers with some loss of precision.

# double: 64-bit floating-point number. Suitable for representing decimal numbers with higher precision.

# char: 16-bit Unicode character. Represents a single character like 'a', 'B', '$', etc.

# boolean: Represents a true or false value.

# Non Primitive Data Types:

# Classes , Interfaces, Arrays