**Java with ChatGPT & Me**

1. **History of Java**

Java is a programming language (Application); used to perform specific tasks. The program consists of a set of instructions (code) about the specific operations.

* Java was invented in the year **1995** by **James Gosling** and his team.
* Initially, it was invented to communicate with T.V.
* The initial name of Java was **OAK** – Symbol of Growth.
* In the year 1996, OAK was renamed as Java, as a programming language, and released the 1st version, J.D.K 1.0

1. **What are the features of Java?**

**ChatGPT:**  
Java has several key features that make it a popular programming language. Here are some of them:

1. **Platform Independence** – Java follows the "Write Once, Run Anywhere" (WORA) principle. It uses the Java Virtual Machine (JVM), which allows Java programs to run on any operating system without modification.
2. **Object-Oriented** – Java is based on object-oriented programming (OOP) principles, such as encapsulation, inheritance, polymorphism, and abstraction, which make code reusable and maintainable.
3. **Simple and Easy to Learn** – Java eliminates complex features from C++ (like pointers and multiple inheritance) and provides a cleaner syntax.
4. **Secure** – Java has built-in security features like bytecode verification, a security manager, and automatic memory management to prevent vulnerabilities.
5. **Robust and Reliable** – Java handles exceptions and errors effectively, reducing the chances of crashes. Its memory management and garbage collection help prevent memory leaks.
6. **Multithreading** – Java allows concurrent execution of multiple tasks using threads, which improves performance in applications like gaming and web servers.
7. **High Performance** – Although Java is an interpreted language, Just-In-Time (JIT) compilation optimizes performance by converting bytecode into native machine code at runtime.
8. **Distributed Computing** – Java supports networking and distributed applications through APIs like RMI (Remote Method Invocation) and CORBA.
9. **Automatic Garbage Collection** – Java automatically manages memory by removing unused objects, reducing manual effort for developers.
10. **Rich API and Libraries** – Java provides extensive built-in libraries for networking, I/O, database management, and more, making development easier.
11. **Number Conversions**

Number conversion is the process of converting a number from one numerical system to another.

| **Number System** | **Base** | **Digits Used** | **Example** |
| --- | --- | --- | --- |
| **Binary** | 2 | 0, 1 | 1011₂ |
| **Octal** | 8 | 0-7 | 57₈ |
| **Decimal** | 10 | 0-9 | 47₁₀ |
| **Hexadecimal** | 16 | 0-9, A-F | 2F₁₆ |

1. **Levels of Languages**

**ChatGPT:**  
In computing, programming languages are categorized into different levels based on their abstraction from machine code. The main levels are:

**1. Low-Level Languages**

These languages are closer to hardware and provide minimal abstraction. They are fast but difficult to understand and write.

* **Machine Language (Binary Code)**: The lowest level, consisting of 0s and 1s, directly executed by the CPU.
* **Assembly Language**: Uses symbolic codes (mnemonics) instead of binary, making it slightly easier to read and write. It requires an assembler to convert it into machine code.

**2. Middle-Level Languages**

These languages balance low-level hardware control and high-level abstraction. They allow interaction with hardware while providing some simplicity.

* **Examples**: C, C++
* C is often called a "middle-level" language because it provides both low-level memory manipulation and high-level programming features.

**3. High-Level Languages**

These languages are closer to human language, making programming easier. They are portable and independent of hardware architecture.

* **Examples**: Java, Python, C#, JavaScript
* These languages use compilers or interpreters to convert code into machine-executable instructions.

**4. Very High-Level Languages**

These languages are even more abstract and specialized for specific tasks.

* **Examples**: SQL (for databases), MATLAB (for numerical computing), Prolog (for AI and logic programming).

1. **Java Development Kit?**

**ChatGPT:**  
The **Java Development Kit (JDK)** is a software package that provides everything needed to develop, compile, and run Java applications. It includes essential tools and components for Java programming.

**Components of JDK**

1. **Java Compiler (javac)** – Converts Java source code (.java files) into bytecode (.class files) that the JVM can execute.
2. **Java Runtime Environment (JRE)** – Includes the Java Virtual Machine (JVM) and libraries required to run Java programs.
3. **Java Virtual Machine (JVM)** – Executes Java bytecode, making Java programs platform-independent.
4. **Development Tools** – Includes debugging tools (jdb), documentation generator (javadoc), and other utilities.
5. **Java Standard Library** – A collection of pre-built classes and APIs (such as java.util, java.io, java.net, etc.) for common programming tasks.

**JDK Versions**

There are different JDK distributions, such as:

* **Oracle JDK** – Official version from Oracle, available under a commercial license.
* **OpenJDK** – Open-source implementation of Java, free to use.
* **Other JDKs** – Various vendors provide their own JDKs, like Amazon Corretto, AdoptOpenJDK, and IBM JDK.

**JDK vs. JRE vs. JVM**

* **JDK** = JRE + development tools (compiler, debugger, etc.).
* **JRE** = JVM + standard libraries (needed to run Java applications).
* **JVM** = Executes Java bytecode on a specific machine.

**User: *What is Java Interpreter?***

**ChatGPT:**  
In Java, the **Java Interpreter** is a part of the **Java Virtual Machine (JVM)** that executes Java bytecode line by line. It reads and translates the compiled **.class** files (which contain Java bytecode) into machine code at runtime.

**How It Works**

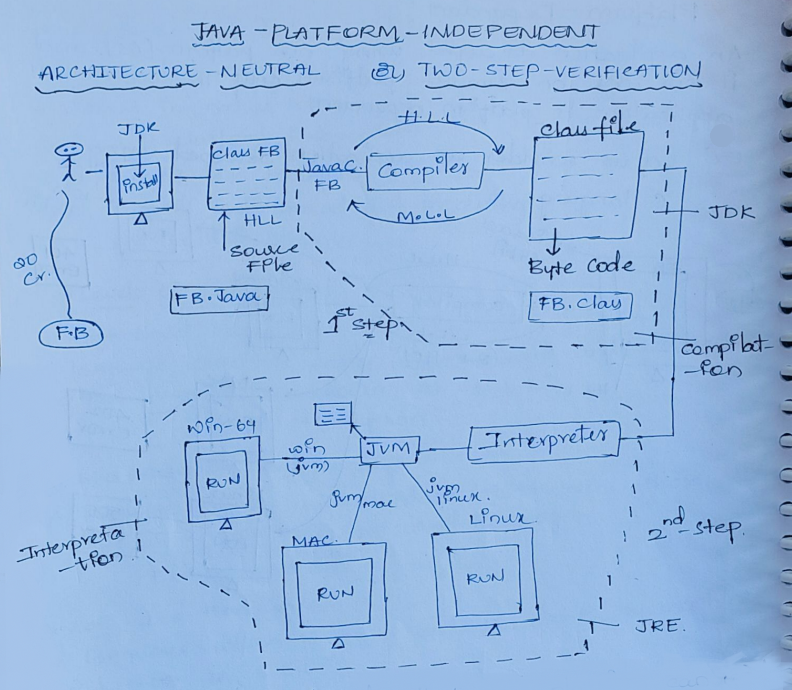
1. The Java source code (.java file) is compiled by the **Java Compiler (javac)** into bytecode (.class file).
2. The **Java Interpreter** (inside the JVM) reads the bytecode and executes it one instruction at a time.

**Key Features of Java Interpreter**

* It enables **platform independence** by running the same Java bytecode on different operating systems.
* Slower than compiled execution because it processes instructions one by one.
* Works alongside the **Just-In-Time (JIT) Compiler**, which converts frequently used bytecode into native machine code for better performance.

**JVM Execution Process**

1. **Interpretation** (executing bytecode line by line).
2. **JIT Compilation** (optimizing performance by compiling frequently executed bytecode into native machine code).
3. **Java Platform Independent**



**Note:** Java is platform independent, but it depends on the JVM.

Structure of Java Program:

// 1. Package declaration (optional)

package mypackage;

// 2. Import statements (optional)

import java.util.Scanner;

// 3. Class definition

public class MyProgram {

// 4. Main method - Entry point of the program

public static void main (String[] args) {

Method Body/ Body Context

} - main method end

The area outside the main method and inside the class is called the **Global Area**

**public** – Access Modifier

**static** – Access Specifier

**void** – Return Type

**(String [] Args)** – (Formal Arguments)

} – class end

1. **Tokens in Java Programming**

A token is the smallest unit in a Java program that the compiler recognizes. Java programs consist of different types of tokens, including **keywords**, **identifiers**, **literals**, operators, **separators**, and comments.

1. **Keywords:** Keywords are pre-defined words that have a specific meaning given by Oracle, whose meaning cannot be changed by the end-user.

* In Java, we have 60+ keywords, and they must start with a lowercase.

Ex: int, class, public, static, void, double, float..

1. **Data Literals:** Data is a raw factor of an object that is are fixed value and used directly in the code.

* Numbers (Whole, Float), Character, Strings, Boolean

1. **Identifiers:** These are the names given to Java members like class name, methods, and variables.

Rules:

* Keywords cannot be used as identifiers.
* They cannot start with digits.
* Only ‘**$**’ and ‘**\_**’ are used as identifiers in the special characters category.
* Spaces are not used as identifiers.
* **Conventions**: Conventions are not mandatory, but as good programmers, we have to follow conventions. Conventions are for industry practice.

1. **Separators:** They are used to differentiate the blocks.

* **i.e., {} [] ; ()**

1. **Data Types and Variables**

Data types are used to reserve memory space by creating a container or memory allocation by specifying two kinds of information

i.e., **Type of data** and **Size of data**

We have two types of Data Types:

* **Primitive DT**: They are used to store a single data value.
* **Non-Primitive DT**: They are used to store multi-value data. Ex: String – group of characters.

| **Category** | **Data Type** | **Size** | **Example Value** |
| --- | --- | --- | --- |
| **Integer Types** | byte | 1 byte | 100 |
|  | short | 2 bytes | 32000 |
|  | int | 4 bytes | 2147483647 |
|  | long | 8 bytes | 9223372036854775807L |
| Floating-Point | float | 4 bytes | 3.14f |
|  | double | 8 bytes | 3.1415926535 |
| Character | char | 2 bytes | 'A' |
| Boolean | boolean | 1 bit | true / false |
| **Reference Types** | String | Varies | "Hello" |
|  | Array | Varies | {1, 2, 3} |
|  | Class | Varies | Person p = new Person(); |

**Variables:** They are the names given to the memory container to store the values for future use.

1. Variable Declaration: Data Type and Variable Name – **int** number
2. Variable Initialization: Variable Name and Assignment Operator and Value/Expression – Ex: number = 10 String = “Java”
3. Variable Declaration and Initialization in a single statement:

Data Type - Variable Name – Assignment Operator – Value/Expression **‘;’ ended with semicolon**

Ex: int a = 10; String s = “Java”;

Classification of Variables:

Variables are classified based on the following types:

* Based on Data Types:

**Primitive Variables** and **Non-Primitive Variables**

* Based on Class Area:

**Local Variables** and **Global Variables**

1. **Command Prompt (CMD) Commands**

**D: -** Directly changing the drive to D drive destination.

**mkdir name\_of\_folder –** Creating a new folder.

**cd folder\_name –** Changing the directory to the given destination.

**javac filename.java** – Compilation of source code file.

**java filename –** Interpretation, running the class file: generates output.

**cd.. –** to go back to the previous directory.

**dir –** To display all the directories.

**cls –** To clear the cmd.

**mkdir name name name –** creating multiple directories.

**cd../.. -**  Go back to the previous multiple times mentioned (twice).

**rmdir name –** Removing/Deleting the directory.

Program: **To print the minimum and maximum values of all the data types.**

Class program {

Public static void main (string args []) {

System.out.println (“Byte minimum value:” + Byte.MIN\_VALUE);

System.out.println (“Byte maximum value:” + Byte.MAX\_VALUE);

short

char

int

long

float

double

Boolean

}

}