CORE JAVA

DAY-1

What is a Program?

A program is a set of instructions that a computer follows in order to perform a particular task.

What is programming language?

- A programming language is a set of instructions written by a programmer to deliver instructions to the computer to perform and accomplish a task.
- A programming language is a type of written language that tells computers what to do.
- Programming refers to a technological process for telling a computer which tasks to perform in order to solve problems.

Different types of programming languages and its definition.

Programming languages can mainly be classified as low-level and high-level programming languages.

Low-Level Language: A type of programming language that is close to the computer's own language. It is harder for people to read and write but allows direct control over the computer's hardware.

Low-level languages include assembly and machine languages.

- Machine language is directly understood by the computer's processing unit. A programmer will first write his code in a high-level language, then compile it into a machine-readable format where instructions are represented in binary.
- Assembly Language: A low-level programming language that uses short, readable codes instead of numbers. Each code represents a basic instruction for the computer's CPU, making it easier for humans to write and understand compared to machine language.

High-Level Language: A type of programming language that is more user-friendly and easier to read and write. It uses abstract terms and symbols closer to human languages, allowing programmers to write instructions without needing to manage hardware details directly. Examples include Python, Java, and C++.

Q. Difference between High-level language and Low-level Language.

Based on Programming Paradigm

1. **Procedural Languages:**
   - Focus on procedures or routines to operate on data. Examples include C, Pascal, and Fortran.
2. **Object-Oriented Languages:**
   - Based on the concept of objects, which are instances of classes. Examples include Java, C++, Python, and Ruby.
3. **Functional Languages:**
   - Treat computation as the evaluation of mathematical functions and avoid changing-state and mutable data. Examples include Haskell, Lisp, and Scala.
4. **Logic Programming Languages:**

    - Based on formal logic, they express facts and rules about problems within a system of formal logic. An example is Prolog.
5. **Scripting Languages:**
    - Often used for automating tasks, they are typically interpreted rather than compiled. Examples include JavaScript, Perl, and Python.

Based on Specific Use Cases

1. **Web Development Languages:**
    - Used for creating websites and web applications. Examples include HTML, CSS, JavaScript, and PHP.
2. **System Programming Languages:**
    - Used for system software development, such as operating systems and compilers. Examples include C and Rust.
3. **Database Query Languages:**
    - Used for querying and manipulating databases. Examples include SQL and PL/SQL.
4. **Scientific Computing Languages:**
    - Used for scientific computations and data analysis. Examples include MATLAB, R, and Python (with libraries like NumPy and SciPy).

Why we use JAVA?

why Java is exciting and why you should consider learning it:

1. **Build Anything, Run Anywhere**: Java can be used to create many types of applications, and once you write your Java program, it can run on any device that has Java installed. So, you can make something on your computer and have it work on your friend's computer, a server as well.
2. **Popular and In-Demand**: Java is one of the most popular programming languages in the world. Big companies like Google, Amazon, and Netflix use Java for their systems. Learning Java can open doors to many job opportunities.
3. **Great for Beginners**: Java's syntax (the way you write code) is straightforward and easy to understand. It’s a great language to start with if you’re new to programming because it teaches you good programming habits.
4. **Strong Future**: Java has been around for over 25 years and continues to grow and evolve. It’s not going away anytime soon, so skills in Java will be valuable for years to come.
5. **Fun Projects**: With Java, you can create exciting projects like Android apps, web applications, games, and even control hardware devices. Imagine building your own mobile app or game – Java can make it happen!

History:

 Origins and Development: Java was developed by James Gosling and his team at Sun Microsystems (later acquired by Oracle Corporation) in the early 1990s. The project initially started as a way to create software for consumer electronic devices, like set-top boxes.

 Release of Java 1.0: The first official version of Java, known as Java 1.0, was released by Sun Microsystems in 1995. This release included the Java Development Kit (JDK) 1.0, which allowed developers to write and compile Java programs.

 Expansion and Standardization: Throughout the late 1990s and early 2000s, Java gained widespread adoption in both enterprise and consumer applications. It was standardized through the Java Community Process (JCP), which allowed for contributions and updates from various stakeholders in the Java ecosystem.

 Enterprise Java and Web Applications: Java became particularly popular for building enterprise applications and web applications. Technologies like Java Enterprise Edition (Java EE) provided frameworks and libraries for developing scalable and robust server-side applications.

 Acquisition by Oracle: In 2010, Oracle Corporation acquired Sun Microsystems. Oracle continues to develop and support Java, releasing new versions and updates regularly.

 Java's Continuing Evolution: Java has evolved over the years with new features, performance improvements, and enhancements to support modern programming practices.

 Community and Ecosystem: Java has a vibrant community of developers, with extensive documentation, libraries, frameworks (like Spring), and tools (such as IntelliJ IDEA and Eclipse IDE) supporting Java development. It remains one of the most widely used programming languages worldwide.

Before learning Java, we are must be familiar with some basic terminologies :

 Java Virtual Machine (JVM): Imagine JVM as a translator for Java programs. It takes the code you write and translates it into a language (bytecode) that your computer can understand and execute. JVM ensures that your Java program can run on different types of computers without needing to rewrite it for each one.

 Bytecode: When you write a Java program, the compiler (Javac) turns it into bytecode. Bytecode is like a set of instructions that JVM can read and execute. It's stored in .class files and is the intermediary step between your readable code and the computer's machine language.

 Java Development Kit (JDK): JDK is a package of tools you need to develop Java applications. It includes the compiler (to turn your code into bytecode), Java Runtime Environment (JRE) which is needed to run Java programs, and other tools like debuggers and documentation.

 Java Runtime Environment (JRE): JRE includes everything needed to run Java applications on your computer. It includes the JVM that executes bytecode, libraries, and plugins. If you only want to run Java programs, you just need JRE installed.

 Garbage Collector: Java manages memory for you. When your program creates objects (like variables and data structures), JVM automatically allocates memory for them. Garbage Collector helps by reclaiming memory from objects that are no longer needed, making sure your program runs efficiently without memory leaks.

 ClassPath: ClassPath is like a set of instructions telling Java where to find classes (like files containing Java code). It includes paths to directories and libraries that your program needs. It's important when your program uses external libraries or resources.

Features of JAVA:

1. **Platform Independence**: Java programs can run on any device or operating system with a Java Virtual Machine (JVM). This is because Java code is compiled into bytecode, which can be executed on any system that has a JVM. This makes Java applications highly portable.
2. **Object-Oriented Programming**: Java is organized around objects, which are instances of classes. This approach promotes concepts like abstraction (hiding complex details), encapsulation (data hiding), inheritance (sharing properties between classes), and polymorphism (using a single interface for multiple data types).
3. **Simple**: Java avoids complex features found in other languages, like pointers and explicit memory management, making it easier to learn and use for beginners.
4. **Robust**: Java is designed to be reliable and detect errors early. Features like garbage collection (automatic memory management), exception handling (managing errors), and strong memory allocation contribute to its robustness.
5. **Secure**: Java's design eliminates certain security risks like buffer overflow by not using pointers. It also runs programs in a controlled environment (JVM), enhancing security.
6. **Distributed**: Java supports creating applications that can be distributed across multiple devices connected via a network. This is achieved through technologies like Remote Method Invocation (RMI) and Enterprise JavaBeans (EJB).
7. **Multithreading**: Java enables concurrent execution of multiple parts of a program, allowing better utilization of CPU resources and enhancing performance in multitasking environments.
8. **Portable**: Java's bytecode can be executed on any platform with a JVM, making it portable across different devices and operating systems without modification.
9. **High Performance**: Java optimizes performance with its architecture and can use Just-In-Time (JIT) compilation to improve execution speed by compiling code on-demand.
10. **Dynamic Flexibility**: Java's object-oriented nature allows flexibility in adding new classes or methods, and it supports integrating code from other languages like C or C++ through native methods.

Installation:

Download JDK (Version 22)

Download Visual Studio Code

Install Extension on VS Code: Extension pack for java

debugger for java

test runner for java

language support for java

maven for java

project manager for java

code runner