**Understanding Programming, Computers, and Java #day1 class notes**

**1. What is the Purpose of a Programming Language?**

The **main purpose of any programming language** is to **solve problems** by providing a set of instructions that a computer can understand and execute. Programming languages serve as a bridge between **human logic** and **computer processing capabilities**.

**2. Components of a Computer**

A **computer system** comprises various hardware components known as **peripheral devices** which help users interact with the system:

* **Input Devices**: Keyboard, Mouse
* **Output Devices**: Monitor, Printer
* **Processing Unit**: CPU (Central Processing Unit)

**3. What is a Processor?**

The **CPU or processor** is the **brain of the computer**. It processes instructions, performs calculations, and manages data. Applications like Notepad, Word, or graphics software ultimately rely on the processor to function.

**4. Operating Systems and Interaction**

An **Operating System (OS)** acts as an intermediary between the **user and the processor**. Common operating systems include:

* **Windows**
* **MacOS**
* **Linux**

Earlier, systems used **command-line interfaces** like **DOS** and **Command Prompt**, which are less user-friendly. Modern OSes provide a **graphical user interface (GUI)** for easier interaction.

**5. Accessing Resources on Windows**

To install or use software on Windows:

* Open a **browser**
* Search and **download the required application**

For example, if you want to check whether a student is **pass or fail**, you'll need **computing capabilities**—either by interacting directly with the processor or through a **programming language** that abstracts the complexity.

**6. Why Programming Languages Exist**

Computers only understand **binary language** (0s and 1s), which is unreadable to humans. Hence, we use **high-level programming languages** like:

* Java, C, C++, Python, .NET
* Deprecated: COBOL, Pascal

These languages help us write human-readable code that can be converted into machine-understandable form

**7. Why Choose Java?**

**Advantages of Java:**

* **OOP (Object-Oriented Programming) support**
* **Platform independence**
* **Garbage collection**
* **No direct pointer usage**

**Example:** Choosing a phone with more features for a slightly higher price—Java offers more features compared to C or C++.

**8. Comparing C, C++, and Java**

| **Language** | **OOP Support** | **Platform Independence** | **Use Case** |
| --- | --- | --- | --- |
| C | ❌ | ❌ | System Programming |
| C++ | ✅ | ❌ | Performance-Critical Applications |
| Java | ✅ | ✅ | Web and Enterprise Applications |
| Python | ✅ | ✅ | Web, AI, Data Science |

**9. What is a Platform?**

A **Platform = Operating System + Processor**

Programming languages interact with the platform by:

* Writing **source code**
* Compiling it to machine-understandable code

**10. Keywords in Programming**

**Keywords** are reserved words used in a programming language. They have predefined meanings and can’t be used for variable names (e.g., if, else, class). Although English is most commonly used, some environments allow localization.

**11. Source Code and Compilation**

* **Source Code**: Human-readable code written in a programming language.
* **Compiler**: Translates source code into **binary code** (0s and 1s).
* Compilers are **platform independent**, but the **output they generate may depend on the target platform**.

**12. Challenges in Platform-Specific Compilation**

Different OS and processor combinations (e.g., Linux + Intel vs. Mac + M1 chip) may cause **compatibility issues** during compilation, especially in languages like C or C++.

**13. Java’s Unique Compilation Process**

Java uses the principle **“Compile Once, Run Anywhere”**:

1. Write Java code → .java file
2. Compile → Bytecode (.class file)
3. Run on any platform with a **JVM (Java Virtual Machine)**

**14. What is Bytecode?**

* Intermediate code generated after compilation
* **Platform-independent**
* Executed by JVM

**Java Compilation Flow:**

plaintext

CopyEdit

| Source Code (.java) |

↓

| Compiler |

↓

| Bytecode (.class) |

↓

| JVM (Windows/Linux/Mac) |

↓

| Machine Code |

**15. Interpreter in Java**

The **interpreter** (part of JVM) executes **bytecode line-by-line**, translating it into **machine code** during runtime.

**16. JVM, javac, JDK, and JRE**

* **JVM**: Executes bytecode (Platform Dependent)
* **javac**: Java compiler that creates bytecode (Platform Independent)
* **JDK**: Java Development Kit = JRE + Development tools (includes javac)
* **JRE**: Java Runtime Environment = JVM + Java Class Libraries

**17. What are Libraries?**

**Libraries** are pre-written code collections that perform specific tasks—saving time and effort during development.

**Important Summary Points**

1. Java is platform-independent because bytecode runs on a platform-specific JVM
2. C++ allows the use of pointers; Java doesn’t
3. Bytecode is the output of the Java compiler
4. JVM is platform-dependent (different for each OS)
5. Compiler (javac) converts source code into bytecode before execution
6. Java follows: Compilation → Bytecode → JVM interpretation
7. Java compiler is *not* included in JRE
8. JRE is a part of JDK
9. javac and java can be used from any directory if added to PATH
10. Libraries provide reusable, task-specific code

**Interview Questions Expected**

**1. Q: Why is Java considered platform-independent?**

**A:** Java is platform-independent because it compiles source code into **bytecode**, which is the same on all systems. This bytecode runs on a **platform-specific JVM**, allowing Java to “compile once, run anywhere.”

**2. Q: What is the role of JVM in Java?**

**A:** The **Java Virtual Machine (JVM)** is responsible for converting bytecode into machine code that the specific OS and processor can understand. Although Java itself is platform-independent, the **JVM is platform-dependent**.

**3. Q: What is the difference between JDK, JRE, and JVM?**

**A:**

* **JDK**: Development kit (includes javac, JRE, tools)
* **JRE**: Runtime environment (includes JVM and class libraries)
* **JVM**: Executes bytecode (part of JRE)

**4. Q: What makes Java different from C and C++?**

**A:** Java supports **OOP**, is **platform-independent**, uses **garbage collection**, and doesn’t allow **pointers**—unlike C and C++, which are platform-dependent and allow low-level memory manipulation.

**5. Q: What is Bytecode in Java?**

**A:** Bytecode is the **intermediate code** generated after compiling a Java program. It is **platform-independent** and executed by the **JVM** on any system.

**6. Q: What is the role of a compiler in Java?**

**A:** The **Java compiler (javac)** translates the source code (.java file) into **bytecode** (.class file). This bytecode is **platform-independent**, meaning it can run on any system that has a compatible **JVM**. The compiler itself is **platform-independent**, but it produces bytecode that is later interpreted by a **platform-dependent** JVM.

**7. Q: What is the difference between source code and bytecode?**

**A:**

* **Source Code**: Human-readable code written by the developer in Java (.java files)
* **Bytecode**: Intermediate, platform-independent code generated after compilation (.class files) that can be executed by the JVM

**8. Q: What is meant by “Java does not support pointers”?**

**A:** Java **does not allow direct access to memory** using pointers, unlike C or C++. This is done to **enhance security and memory safety**, prevent memory leaks, and make garbage collection more effective. Java uses **references** instead of pointers to access objects.

**9. Q: Why is JVM considered platform-dependent?**

**A:** While Java bytecode is platform-independent, the **JVM itself is platform-dependent** because it needs to be specifically implemented for each OS and hardware combination (e.g., Windows JVM, Mac JVM, Linux JVM) to convert bytecode into **machine code**.

**10. Q: What is the importance of JDK in Java development?**

**A:** The **JDK (Java Development Kit)** is essential for developing Java applications. It includes:

* The **Java compiler (javac)**
* The **JRE (Java Runtime Environment)**
* Development tools like javadoc, javap, and jar

It is not possible to write and compile Java code without the JDK.