Java Practice – Day 1 (Foundations)

1. Why Programming Exists

Before starting coding, we must first understand:

- Why programming is needed
- How programming came into existence
- Different levels of programming languages
- How a computer system works internally (memory & processor)

Programming exists to **instruct computers to solve problems**, automate tasks, and process data efficiently.

---

2. Heart of a Computer – Processor (CPU)

The most important component of any device (mobile, laptop, or computer) is the **processor (CPU)**.
Without a processor, a computer cannot perform any operation.

The processor:

- Executes instructions
- Performs calculations
- Controls all operations of the system

---

3. How Processors Work (0s and 1s)

Processors are built using **semiconductor technology**, mainly **transistors**.

A **transistor**:

- Is very fast
- Can represent two voltage states:
  - **High voltage → 1**
  - **Low voltage → 0**

Because of this, computers understand only **binary data (0 and 1)**.

---

4. Machine Level Language (MLL)

- Writing instructions in **binary form (0s and 1s)** is called **Machine Level Language**
- It is also known as **Binary Code**
- The processor can **directly understand only MLL**

 In early days, programmers wrote programs **only in machine language**, which was:

- Very difficult to write
- Hard to read
- Error-prone

---

5. Assembly Level Language (ALL)

To improve readability, **Assembly Level Language** was introduced.

Features:

- Uses symbolic instructions (like `ADD`, `MOV`)
- Easier than binary code
- Faster development compared to MLL

But:

- Processors still cannot understand it directly
- We need an **Assembler** to convert **Assembly → Machine Language**

---

6. High Level Language (HLL)

Later, **High Level Languages** were developed to make programming easy for humans.

Examples:

- C, C++, Java, Python

Features:

- Human-readable
- Easy to write and maintain
- Platform independent (in Java's case)

To execute HLL:

- A **Compiler** or **Interpreter** converts HLL into Machine Level Language

 Java uses both **Compiler + JVM**, which we'll discuss later.

---

7. Memory Components in a Computer

Now let’s understand **Hard Disk, RAM, Cache, and Processor**.

Early Computer Design

- Hard Disk → stores data
- Processor → executes instructions

Problem:

- Data transfer between Hard Disk and Processor was **slow**
- Reason:
    - Processor → **Semiconductor technology (very fast)**
    - Hard Disk → **Electro-mechanical / magnetic technology (slow)**

---

8. Why RAM Was Invented

To match the speed of the processor, **RAM (Random Access Memory)** was introduced.

RAM characteristics:

- Made of semiconductor technology
- Faster than hard disk
- Acts as a bridge between processor and storage
- Stores data and instructions **temporarily**

 Disadvantage:

- RAM is **volatile**
- Even a small power loss → **entire data is lost**

---

9. Role of Hard Disk (Secondary Memory)

To solve RAM’s volatility issue:

- **Hard Disk** is used for permanent storage

Features:

- Non-volatile (data stays even after power off)
- Cheaper than RAM
- Slower than RAM

 If you **save your work**, it is stored in hard disk and remains as long as you want.

How Data Travels: HDD / SSD → RAM → Processor

A computer **never executes data directly from HDD or SSD**.
Execution always happens **inside the processor**, but data must pass through **RAM first**.

Step-by-Step Data Flow

Step 1: Data Stored in HDD / SSD

- Programs (Java files, OS, applications) are stored permanently in **HDD or SSD**
- This storage is **non-volatile** but **slow** compared to RAM

Example:

- `HelloWorld.java`
- Java compiler
- Operating System files

Step 2: Program Loaded into RAM

When you **open or run a program**:

- Operating System copies required files from **HDD/SSD into RAM**
- Reason:
  - RAM is much faster than storage
  - Processor can communicate efficiently with RAM

□ **Processor cannot directly access HDD/SSD**

Step 3: Data Moves from RAM to Cache

- Frequently used instructions/data from RAM are copied into **Cache Memory**
- Cache is:
  - Very small
  - Extremely fast
  - Located closest to the processor

Step 4: Processor Executes Instructions

- Processor fetches instructions from:
    1. Cache (first priority)
    2. RAM (if not found in cache)
- Instructions are executed inside CPU using:
    - ALU (calculations)
    - Control Unit (decision making)

Complete Flow (One Line)

```
HDD / SSD → RAM → Cache → Processor (CPU)
```

Simple Real-Life Example

Imagine studying for an exam:

- **HDD/SSD** → Books in your cupboard
- **RAM** → Books on your table
- **Cache** → Notes in your hand
- **Processor** → Your brain

You **never read directly from the cupboard**—you bring books closer first.

Why This Design Is Important

- Storage is slow but permanent
- RAM is fast but temporary
- Cache is ultra-fast but very small
- Processor needs **speed**, not storage

This layered approach gives:

- ⚡ High performance
- □ Cost efficiency
- □ Better power management

Java-Specific Hint (Preview)

When you run a Java program:

1. `.class` file loads from **SSD/HDD**
2. JVM loads it into **RAM**
3. Frequently used bytecode moves to **cache**

4. CPU executes it

(We’ll cover this in detail in JVM topics)

How Data Travels Using BUS

Data does **not magically jump** from HDD/SSD to RAM or CPU.
It **travels through a communication system called the BUS**.

What is a BUS?

A **BUS** is a **set of electrical pathways (wires)** that allows components of a computer to communicate with each other.

It connects:

- **HDD / SSD**
- **RAM**
- **Processor (CPU)**
- **Cache & I/O devices**

Types of BUS (Important)

1. **Data Bus**
    - Transfers **actual data**
    - Example: program instructions, variables
2. **Address Bus**
    - Specifies **where** the data should be read from or written to
    - Example: RAM location
3. **Control Bus**
    - Carries **control signals**
    - Example: read, write, interrupt

Correct Data Flow with BUS

Step-by-Step:

1. Program is stored in **HDD / SSD**
2. OS sends a request via **Control Bus**
3. Data location is sent via **Address Bus**
4. Actual data moves via **Data Bus**
5. Data reaches **RAM**
6. Frequently used data moves from RAM → **Cache**

7. CPU fetches data via **Bus Interface Unit**

```
HDD / SSD
   ↓ (System Bus)
RAM
   ↓ (System Bus)
Cache
   ↓
CPU
```

Why BUS Is Important

- Processor speed is very high
- HDD/SSD speed is much lower
- BUS acts as a **bridge**
- Controls **how fast and safely** data moves

Why HDD Was Slow Earlier (Exam Point)

- HDD uses **electro-mechanical technology**
- Processor uses **semiconductor technology**
- BUS had to wait for HDD rotations
- This created a **speed mismatch**

➡ That's why **RAM + Cache** were introduced

10. Primary vs Secondary Memory

Memory Type	Example	Speed	Volatility
Primary Memory	RAM, Cache	Very Fast	Volatile
Secondary Memory	Hard Disk, SSD	Slow/Medium	Non-Volatile

---

11. Cache Memory (Very Important)

Cache memory is a small, ultra-fast memory located **very close to the processor**.

Purpose:

- Stores **frequently used data**
- Reduces access time
- Improves performance

Example:

Imagine:

- You open WhatsApp many times a day
- Frequently used data (icons, recent chats) is stored in **cache**
- So next time, the app opens faster

Similarly:

- If the processor needs the **same data again and again**
- That data is stored in **cache memory**
- Processor gets it quickly without going to RAM or Hard Disk

---

12. Memory Speed Order

From fastest to slowest:

1. **Cache Memory**
2. **RAM**
3. **SSD**
4. **Hard Disk**