Morning Session:

1. Introduction to Java

Historical Perspective

- Java was developed by **James Gosling** at Sun Microsystems in **1991** (initially called **Oak**).
- Officially released in **1995** by Sun Microsystems (later acquired by Oracle).
- Key Milestones:
  - **Java 1.0 (1996)**: First public release.
  - **Java 5.0 (2004)**: Introduced generics, enums, and enhanced for-loop.
  - **Java 8 (2014)**: Added **Lambda Expressions, Stream API**.
  - **Java 17 (2021)**: Latest LTS (Long-Term Support) version.

Core Applications

- **Desktop Applications**: JavaFX, Swing.
- **Web Applications**: Servlets, JSP, Spring Framework.
- **Mobile Applications**: Android (Java/Kotlin).
- **Enterprise Applications**: Banking, E-commerce (Java EE).
- **Big Data & Cloud**: Hadoop, Spark.

Java Ecosystem

- **JDK (Java Development Kit)**: Tools for development (compiler, debugger).
- **JRE (Java Runtime Environment)**: Runs compiled Java programs.
- **JVM (Java Virtual Machine)**: Executes bytecode (platform-independent).

2. Installing and Configuring the Java Development Kit (JDK)

Steps to Install JDK

1. Download JDK from Oracle's official website.

2. Run the installer and follow instructions.
3. Set **JAVA_HOME** environment variable.

- **Windows**:

```
set JAVA_HOME=C:\Program Files\Java\jdk-17
```

- **Linux/macOS**:

```
export JAVA_HOME=/usr/lib/jvm/jdk-17
```

4. Add **JDK/bin** to **PATH**:

```
export PATH=$PATH:$JAVA_HOME/bin
```

5. Verify installation:

```
java -version
javac -version
```

3. Writing, Compiling, and Executing Your First Java Program

Example: HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Steps to Compile & Run

1. Save as `HelloWorld.java`.
2. Compile:

```
javac HelloWorld.java
```

3. Run:

```
java HelloWorld
```

Output:

```
Hello, World!
```

4. Fundamental Data Types, Variables, and Operators

Primitive Data Types

Data Type	Size	Example
byte	1 byte	byte b = 100;
short	2 bytes	short s = 5000;
int	4 bytes	int num = 42;
long	8 bytes	long l = 999999999L;
float	4 bytes	float f = 3.14f;
double	8 bytes	double d = 99.99;
char	2 bytes	char c = 'A';
boolean	1 bit	boolean flag = true;

Variables & Naming Rules

- Must start with a letter, `_`, or `$`.
- Cannot use Java keywords (`int`, `class`).
- Example:

```
int age = 25;
String name = "Alice";
```

Operators

- **Arithmetic**: `+`, `-`, `*`, `/`, `%`

```
int sum = 10 + 5; // 15
```

- **Comparison**: `==`, `!=`, `>`, `<`

```
boolean isEqual = (10 == 5); // false
```

- **Logical**: `&&`, `||`, `!`

```
boolean result = (true && false); // false
```

Afternoon Session:

1. Object-Oriented Programming (OOP) Principles in Java

Four Pillars of OOP

1. **Encapsulation**: Hide internal state (private fields, public getters/setters).
2. **Inheritance**: Extend classes (`extends` keyword).
3. **Polymorphism**: Same method, different forms (method overriding).
4. **Abstraction**: Hide complexity (abstract classes, interfaces).

2. Defining Classes and Objects

Class (Blueprint)

```
class Car {
    String model;
    int year;

    void start() {
        System.out.println("Car started!");
    }
}
```

Object (Instance)

```
public class Main {
    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.model = "Tesla";
        myCar.year = 2023;
        myCar.start(); // Output: "Car started!"
    }
}
```

3. Method Encapsulation and `this` Keyword

Encapsulation Example

```
class Student {
```

```
    private String name;

    // Getter
    public String getName() {
        return name;
    }

    // Setter
    public void setName(String name) {
        this.name = name; // 'this' refers to current object
    }
}
```

Using this Keyword

```
Student s = new Student();
s.setName("Bob");
System.out.println(s.getName()); // "Bob"
```

4. Constructors for Object Initialization

Default & Parameterized Constructors

```
class Book {
    String title;

    // Default Constructor
    Book() {
        title = "Unknown";
    }

    // Parameterized Constructor
    Book(String t) {
        title = t;
    }
}
```

Usage

```
Book b1 = new Book();
Book b2 = new Book("Java Programming");
System.out.println(b1.title); // "Unknown"
```

```
System.out.println(b2.title); // "Java Programming"
```

5. Control Flow: Conditional Statements & Loops

If-Else Example

```
int age = 18;
if (age >= 18) {
    System.out.println("Adult");
} else {
    System.out.println("Minor");
}
```

For Loop Example

```
for (int i = 1; i <= 5; i++) {
    System.out.println(i); // Prints 1 to 5
}
```

While Loop Example

```
int count = 0;
while (count < 3) {
    System.out.println("Hello");
    count++;
}
```

Summary

- Java basics, installation, first program, data types, operators.
- OOP concepts, classes/objects, encapsulation, constructors, control flow.