A Simple and Detailed Guide to Java Architecture and Core Concepts

1. Java Architecture Overview

Steps to Execute a Java Program:

1. Write the Program:
- We write the Java source code in any text editor like Notepad.
- Save the file with the same name as the class name and `.java` extension.
  Example: `ProgramName.java`

2. Compile the Program:
- Use the `javac` (Java Compiler) command to compile the code.
- It converts the source code into Bytecode (an intermediate code).
- Output is a `.class` file.

3. Execute the Program:
- Use the `java` command (Java Interpreter) to execute the `.class` file.
- It converts bytecode into machine code (platform-specific).

Hence, Java is known for "Write Once, Run Anywhere" (WORA).

Flowchart of Java Program Execution

```
 Source Code (Program.java)
        |
     javac        javac Program.java
        |
   Bytecode (Program.class)(intermediate code)
        |
     java         java Program
        |
   Machine Code (Executed Output)
```

2. Java Tools Overview

Java Tools Table:

Tool	Purpose
javac	Java compiler – compiles `.java` files into bytecode (`.class` files).
javadoc	Generates HTML documentation from Java source code comments.
javah (deprecated)	Was used to generate C header files for native methods.
javap	Disassembles a class file to show bytecode and method information.
jdb	Java debugger – used to identify and fix bugs in Java programs.
appletviewer	Executes Java applets without using a web browser (deprecated now).
JIT Compiler	Compiles bytecode into machine code at runtime for better performance.

3. JVM, JRE, JDK

- JDK includes:
  - JRE
    - JVM
  - Development tools (javac, javadoc, etc.)

Explanation in Simple English:

- JVM (Java Virtual Machine): Runs the bytecode and converts it to machine code.
- JRE (Java Runtime Environment): Includes JVM and required libraries to run Java programs.
- JDK (Java Development Kit): Full package including JRE + development tools for writing Java programs.

4. Java Core Packages

- java.util: Includes utility classes like ArrayList, Scanner, Collections, etc.
- java.awt: For GUI components like Button, Frame, Label, etc.
- java.lang: Automatically imported. Includes core classes like String, Math, Object, etc.
- java.io: For input and output (File handling, Streams).
- java.applet: For creating and running Java Applets (obsolete in modern Java).

5. Scanner Class

Why We Need Scanner?

It is used to take input from the user.

Methods in Scanner:

- nextInt() - Reads an integer
- nextFloat() - Reads a float
- nextLong() - Reads a long
- nextDouble() - Reads a double
- next() - Reads a single word (without spaces)
- nextLine() - Reads a full line (including spaces)
- next().charAt(0) - Reads a single character

Example Program Using Scanner:

```
import java.util.Scanner;

public class InputExample {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter int: ");
    int a = sc.nextInt();

    System.out.print("Enter float: ");
    float b = sc.nextFloat();

    System.out.print("Enter long: ");
    long c = sc.nextLong();

    System.out.print("Enter double: ");
    double d = sc.nextDouble();

    sc.nextLine(); // Consume leftover newline
```

```
        System.out.print("Enter string with spaces: ");
        String strLine = sc.nextLine();

        System.out.print("Enter single word: ");
        String str = sc.next();

        System.out.print("Enter character: ");
        char ch = sc.next().charAt(0);

        System.out.println("Output:");
        System.out.println(a + ", " + b + ", " + c + ", " + d + ", " + strLine + ", " + str + ", " + ch);
    }
}
```

Difference between next() and nextLine():

- next() reads only until space.
- nextLine() reads the entire line including spaces.

print vs println

- print() prints in the same line.
- println() prints and moves to the next line.

6. Core Java Concepts

What is a Class?

A class is a user-defined data type which can hold different data types and methods.

What is a Method?

A method is a block of code that performs a specific task.

What is a Constructor?

A constructor is a special method that initializes an object. It has the same name as the class and no return type.

public static void main(String[] args)

- public: Access modifier, visible everywhere.
- static: No object is needed to call main().
- void: It doesn’t return anything.
- main: Starting point of the program.
- String[] args: Used to accept command-line arguments.

Why Create Objects?

Objects are instances of classes. They allow us to use class variables and methods.

Example:

```
class Student {
    String name;
    int age;

    Student(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }

    public static void main(String[] args) {
        Student s1 = new Student("John", 20);
        s1.display();
    }
}
```