**Name : Mangesh Ghadwaje**

**Class : SYMCA**

**Features of Java**

* **Platform Independent**: Write once, run anywhere (WORA) using JVM.
* **Object-Oriented**: Encourages modularity and code reuse through classes and objects.
* **Simple**: Easy to learn with a syntax similar to C++.
* **Secure**: Provides built-in security features like bytecode verification and secure class loading.
* **Multithreaded**: Supports concurrent execution of multiple threads.
* **Robust**: Strong memory management and exception handling mechanisms.
* **High Performance**: Just-In-Time (JIT) compiler optimizes bytecode to machine code.
* **Distributed**: Supports networking and distributed computing.
* **Dynamic**: Supports dynamic loading of classes and libraries.

**JVM Architecture**

* **Class Loader Subsystem**: Loads, links, and initializes classes.
* **Runtime Data Areas**:
  + **Method Area**: Stores class structures like runtime constant pool, field, and method data.
  + **Heap Area**: Stores objects and arrays.
  + **Stack Area**: Stores local variables and partial results. Each thread has its own stack.
  + **Program Counter (PC) Register**: Contains the address of the JVM instruction currently being executed.
  + **Native Method Stack**: Contains all the native methods used in the application.
* **Execution Engine**:
  + **Interpreter**: Interprets bytecode into machine code.
  + **Just-In-Time (JIT) Compiler**: Compiles bytecode to native code for performance enhancement.
  + **Garbage Collector**: Manages memory by reclaiming unused objects.
* **Native Method Interface (JNI)**: Allows execution of native methods.

**JDK and Its Usage**

* **Java Development Kit (JDK)**: Provides tools necessary for Java development.
* **Key Tools**:
  + javac: Java compiler that compiles .java files into bytecode .class files.
  + java: Launches the Java application by invoking the JVM.
  + javadoc: Generates HTML documentation from Java source code.
  + jdb: Java debugger for finding and fixing bugs in Java programs.

**Structure of a Java Class**

java

Copy code

public class ClassName {

// Fields (Variables)

private int field1;

private String field2;

// Constructor

public ClassName(int field1, String field2) {

this.field1 = field1;

this.field2 = field2;

}

// Methods

public void method1() {

// Method implementation

}

public String method2() {

return field2;

}

// Main method (Entry point of the application)

public static void main(String[] args) {

ClassName obj = new ClassName(1, "Hello");

obj.method1();

System.out.println(obj.method2());

}

}

**Writing Your First Java Program**

**Example: Hello World**

java

Copy code

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

**About main() Method**

* **Syntax**: public static void main(String[] args)
* **Purpose**: Entry point for the application. The JVM invokes this method to start the program.
* **Modifiers**:
  + public: Accessible by any class.
  + static: Can be called without creating an instance of the class.
  + void: Does not return any value.
  + String[] args: Array of command-line arguments passed to the program.

**Constructor in Java**

* **Purpose**: Initializes a new object.
* **Syntax**:

java

Copy code

public ClassName(parameters) {

// Initialization code

}

* **Types**:
  + **Default Constructor**: No parameters, provided by the compiler if no other constructors are defined.
  + **Parameterized Constructor**: Accepts parameters to initialize fields.

**Character Set (Alphabets, Digits, Special Symbols)**

* **Alphabets**: A-Z, a-z
* **Digits**: 0-9
* **Special Symbols**: + - \* / = % ( ) { } [ ] ; : , . " ' < >

**Java Comments**

* **Single-line**: // This is a single-line comment
* **Multi-line**: /\* This is a multi-line comment \*/
* **Documentation**: /\*\* This is a documentation comment \*/

**Java Keywords**

* Reserved words with predefined meanings, e.g., class, public, static, void, int, if, else, etc.

**Java Datatype**

* **Primitive Datatypes**:
  + **byte**: 8-bit integer
  + **short**: 16-bit integer
  + **int**: 32-bit integer
  + **long**: 64-bit integer
  + **float**: 32-bit floating point
  + **double**: 64-bit floating point
  + **char**: 16-bit Unicode character
  + **boolean**: true or false

**Variable**

* **Syntax**: datatype variableName = value;
* **Example**: int age = 25;

**Constant**

* **Syntax**: final datatype CONSTANT\_NAME = value;
* **Example**: final int MAX\_AGE = 100;

**Java Literals**

* **Integer Literals**: 10, 0b1010 (binary), 012 (octal), 0xA (hexadecimal)
* **Floating-point Literals**: 10.5, 2.3e4
* **Character Literals**: 'A', '\\'
* **String Literals**: "Hello"
* **Boolean Literals**: true, false

**Demonstration and Lab Activities**

**Print "Hello, World!"**

java

Copy code

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

**Add Two Numbers**

java

Copy code

public class AddNumbers {

public static void main(String[] args) {

int num1 = 5;

int num2 = 10;

int sum = num1 + num2;

System.out.println("Sum: " + sum);

}

}

**Calculate Compound Interest**

java

Copy code

public class CompoundInterest {

public static void main(String[] args) {

double principal = 1000;

double rate = 5;

int time = 2;

double amount = principal \* Math.pow((1 + rate / 100), time);

double compoundInterest = amount - principal;

System.out.println("Compound Interest: " + compoundInterest);

}

}

**Calculate Power of a Number**

java

Copy code

public class PowerOfNumber {

public static void main(String[] args) {

int base = 2;

int exponent = 3;

int result = (int) Math.pow(base, exponent);

System.out.println("Power: " + result);

}

}

**Swap Two Numbers**

java

Copy code

public class SwapNumbers {

public static void main(String[] args) {

int a = 5;

int b = 10;

System.out.println("Before Swap: a = " + a + ", b = " + b);

int temp = a;

a = b;

b = temp;

System.out.println("After Swap: a = " + a + ", b = " + b);

}

}

**Exercise 1: Profit Calculation**

java

Copy code

public class ProfitCalculation {

public static void main(String[] args) {

double costPrice = 500;

double sellingPrice = 700;

double profit = sellingPrice - costPrice;

System.out.println("Profit: " + profit);

}

}

**Exercise 2: Profit and Loss**

java

Copy code

public class ProfitAndLoss {

public static void main(String[] args) {

double costPrice = 500;

double sellingPrice = 300;

if (sellingPrice > costPrice) {

double profit = sellingPrice - costPrice;

System.out.println("Profit: " + profit);

} else {

double loss = costPrice - sellingPrice;

System.out.println("Loss: " + loss);

}

}

}

**Java Output**

**Using System.out.println()**

* Prints a message with a newline at the end.
* **Example**: System.out.println("Hello, World!");

**Using System.out.print()**

* Prints a message without a newline.
* **Example**: System.out.print("Hello, World!");

**Using System.out.printf()**

* Formats and prints a message.
* **Example**: System.out.printf("Hello, %s!", "World");

**Difference between println(), print(), and printf()**

* **println()**: Adds a newline after the message.
* **print()**: Does not add a newline after the message.
* **printf()**: Formats the message according to the format specifiers.

**Formatting Output with printf()**

* **Format Rules**:
  + %[flags][width][.precision]conversion-character
  + **Flags**:
    - -: Left-justify
    - +: Include sign
    - 0: Pad with zeros
    - ,: Include grouping separator
  + **Width**: Minimum number of characters to be written.
  + **Precision**: Number of digits after the decimal point for floating-point numbers.

**Conversion Characters**

* %d: Integer
* %f: Floating-point
* %s: String
* %c: Character
* %b: Boolean

**Optional Modifiers**

* **Boolean Formatting**: %b
  + Example: System.out.printf("%b", true);
* **String Formatting**: %s
  + Example: System.out.printf("%s", "Hello");
* **Char Formatting**: %c
  + Example: System.out.printf("%c", 'A');
* **Number Formatting**:
  + **Integer Formatting**: %d
    - Example: System.out.printf("%d", 100);
  + **Float and Double Formatting**: %f
    - Example: System.out.printf("%.2f", 10.12345);

**Date and Time Formatting**

* **Time Formatting**: %t
  + Example: System.out.printf("%tT", new Date());
* **Date Formatting**: %t
  + Example: System.out.printf("%tD", new Date());