Java Basics & OOPs Assignment Questions

# 1. Java Basics

1. What is Java? Explain its features.

Ans : Java is a high-level**,** object-oriented programming language developed by SunMicrosystems in 1995, now owned by Oracle Corporation. It is widely used for building web applications, mobile apps (especially Android), desktop software, enterprise solutions, and more.

**1. Platform Independent**

Java is platform independent because it uses the Java Virtual Machine (JVM). When a Java program is compiled, it is converted into bytecode, which is not platform-specific.

**2. Object-Oriented**

Java is a fully object-oriented language. This means that everything in Java is treated as an object, which represents real-world entities..

**3. Simple**

Java is simple to learn and use because its syntax is clear and easy to understand.

**4. Secure**

Java is a secure language. It runs inside a virtual machine which acts as a sandbox and prevents unauthorized access to system resources.

**7. Portable**

Java is portable because the compiled bytecode can run on any system with a JVM.

1. Explain the Java program execution process.

Write the Code

• Open Notepad and write your Java code.

• Save the file as hello.java

Compile the Code

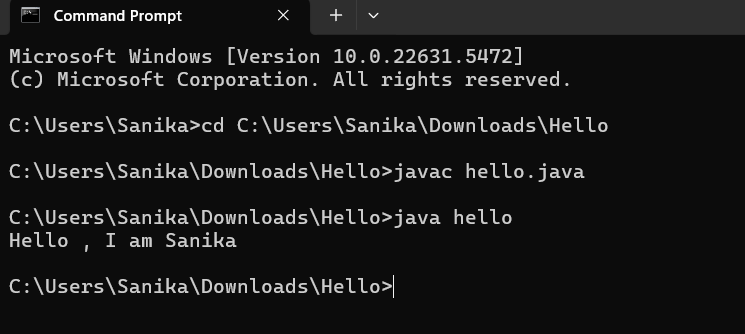
• Open Command Prompt (cmd).

• Navigate to the folder where Hello.java is saved using cd command.

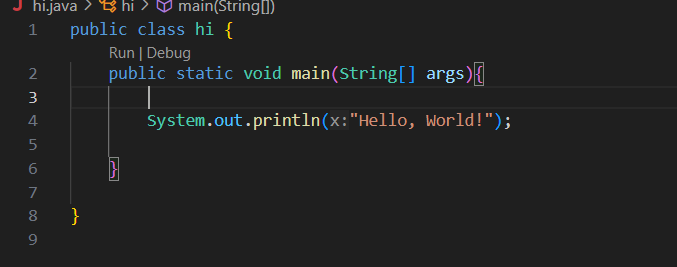
• USE Command as Javac hello.java

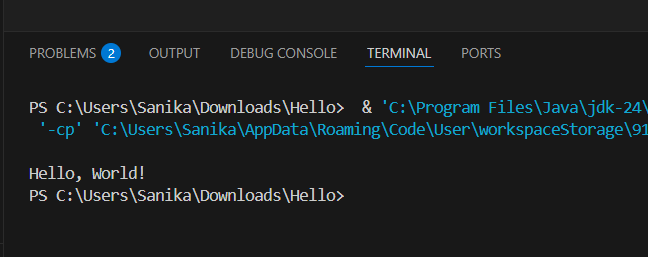
• This creates Hello.class (bytecode file).

Run the Program • Run the compiled class using the Java interpreter: • Use command as java Hello



Write a simple Java program to display 'Hello World'.





1. What are data types in Java? List and explain them.

Ans : Data types in Java define the type of data a variable can store. Java is a strongly typed language, meaning every variable must be declared with a data type before use.

1. Primitive Data Types These are the basic, built-in types. Java has 8 primitives:

• byte: Small integers, saves memory. ………Range: -128 to 127.

• short: Slightly larger integers. ……………Range: -32,768 to 32,767.

• int: Default for integers…………..Range: about ±2 billion.

• long: Used for very large integers………….. Ends with L.

• float: Decimal numbers (less precision)…………. Ends with f.

• double: Default for decimal numbers (more precision).

• char: Stores a single character (e.g., 'A', '5').

• boolean: Stores true or false values.

2. Non-Primitive (Reference) Data Types These are more complex types that refer to objects in memory:

• String – Stores a sequence of characters.

• Arrays – Stores multiple values of the same type.

• Classes/Objects – User-defined data types with methods and properties.

• Interfaces – Used to achieve abstraction and multiple inheritance.

1. What is the difference between JDK, JRE, and JVM?

Ans :

. JVM (Java Virtual Machine)

It runs the compiled Java bytecode (.class files).

➢ JVM makes Java platform-independent (same code runs on Windows, Linux, etc.).

➢ It handles memory, garbage collection, and execution.

Java program runner.

2. JRE (Java Runtime Environment)

➢ It contains the JVM + required libraries to run Java programs.

➢ You can run Java applications but cannot develop or compile code with it.

Ready-to-run Java setup.

JDK (Java Development Kit)

➢ It includes JRE + development tools like javac (compiler), debugger, etc.

➢ You need JDK to write, compile, and run Java programs.

Full Java software development kit.

1. What are variables in Java? Explain with examples.

Ans :

In Java, a **variable** is a **container that holds data** that can be changed during the execution of a program. Every variable in Java must have a **data type**, which defines the type of data it can store.

**1. Local Variable**

* Declared inside a method or block.
* Only accessible within that method or block.
* Not automatically initialized.

**2. Instance Variable**

* Declared inside a class but outside any method.
* Belongs to each object (instance) of the class.
* Has default values if not initialized.

**3. Static Variable**

* Declared using the static keyword.
* Belongs to the class rather than objects.
* Shared among all instances of the class.

1. What are the different types of operators in Java?

Ans :

In Java, **operators** are special symbols used to perform **operations** on variables and values. They are essential for performing calculations, comparisons, and logical decisions in a program.

Java provides several types of operators:

Ans:- 1. Arithmetic Operators Used to perform basic math operations like addition, subtraction, multiplication, division, and modulus.

• Examples: +, -, \*, /, %

• Also includes increment (++) and decrement (--).

• Example: a + b, a++

2. Relational Operators Used to compare two values and return true or false.

• Examples: == (equal), != (not equal), >, =, <=

• Example: a > b returns true if a is greater than b.

3.Logical Operators Used to combine multiple boolean expressions.

• Examples: && (AND), || (OR), ! (NOT)

• Example: (a > b) && (a < c) checks if both conditions are true.

4. Assignment Operators o Used to assign values to variables and modify them.

• Examples: =, +=, -=, \*=, /=, %=

• Example: c += 5 means c = c + 5. 5.

5.Bitwise Operators Operate on bits of integers. Useful for low-level programming.

• Examples: & (AND), | (OR), ^ (XOR), ~ (NOT), << (left shift), >> (right shift), >>> (unsigned right shift)

• Example: d & e performs AND on each bit of d and e

6. Ternary Operator A shorthand for if-else statements.

• Syntax: condition ? valueIfTrue : valueIfFalse

• Example: int max = (a > b) ? a : b;

7. Instanceof Operator Checks whether an object is an instance of a specific class or interface.

• Returns true or false.

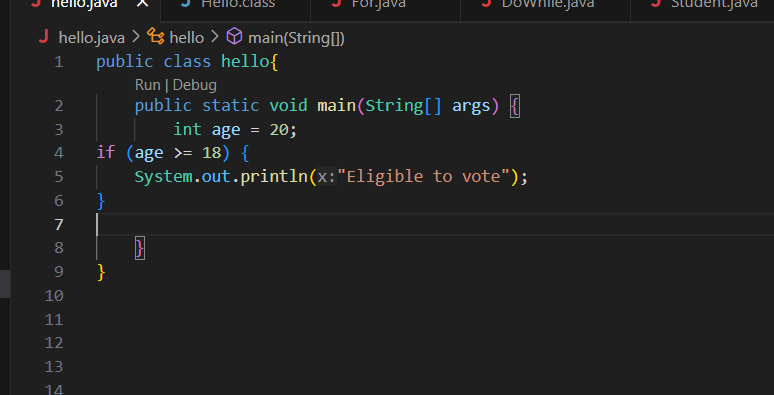
• Example: str instanceof String returns true if str is a String object.

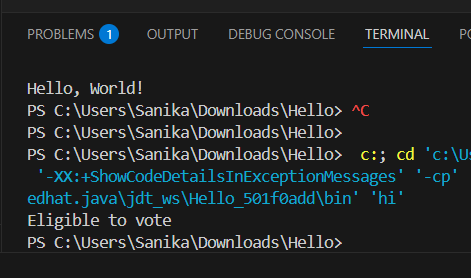
1. Explain control statements in Java (if, if-else, switch).

Ans :

**1. if Statement**

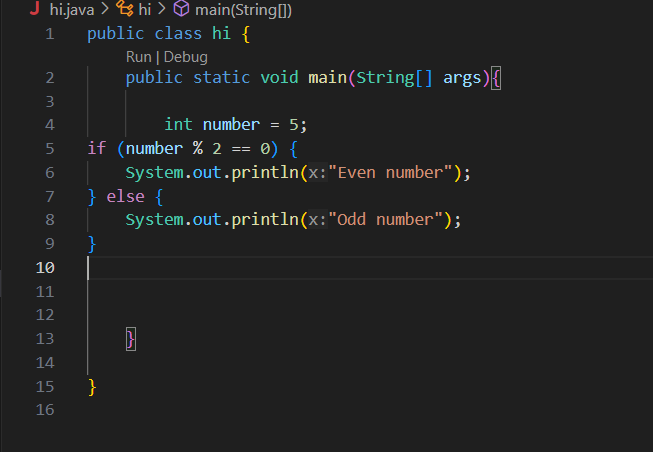
The if statement executes a block of code **only if** the specified condition is true.

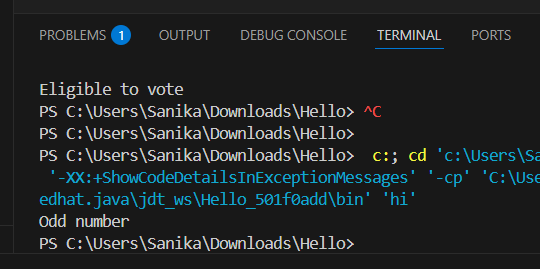




**2. if-else Statement**

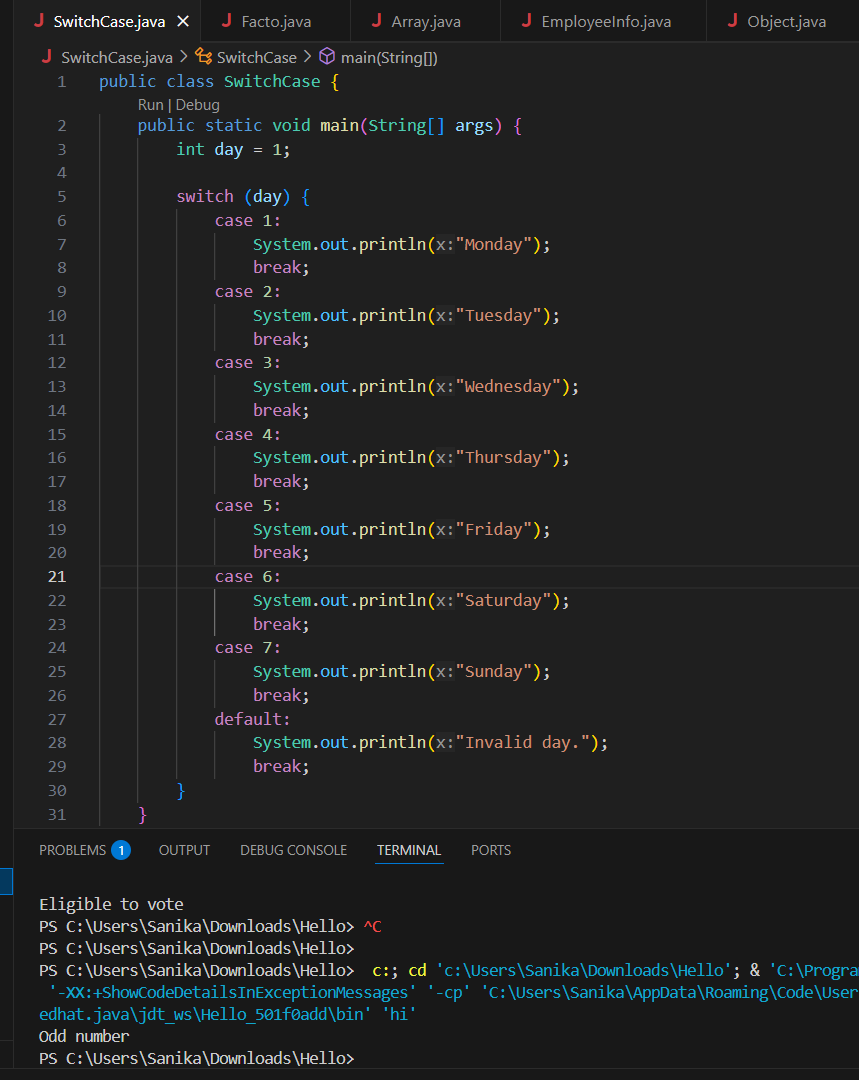
The if-else statement executes one block if the condition is **true**, and another block if the condition is **false**.



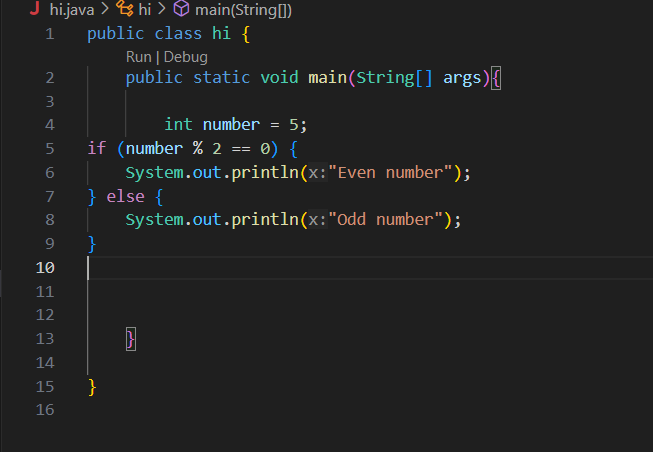


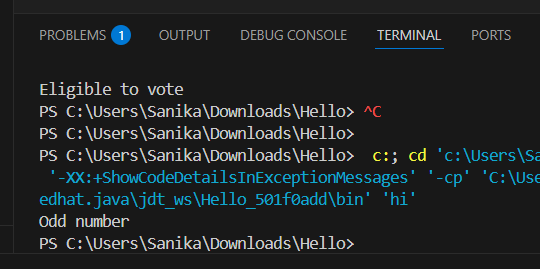
**3. switch Statement**

The switch statement is used to select one of many blocks of code to execute based on the value of a variable.



1. Write a Java program to find whether a number is even or odd.





1. What is the difference between while and do-while loop?

 **Condition Checking**:

* In a while loop, the condition is checked **before** the loop body executes.
* In a do-while loop, the condition is checked **after** the loop body executes.

 **Minimum Execution**:

* A while loop **may not execute** the loop body if the condition is false initially.
* A do-while loop **always executes at least once**, even if the condition is false.

 **Loop Type**:

* while is an **entry-controlled** loop.
* do-while is an **exit-controlled** loop.

 **Use Case**:

* Use while when the loop should run **only if** the condition is true.
* Use do-while when the loop must run **at least once**, regardless of the condition.

 **Syntax Difference**:

* while (condition) { // code } checks the condition first.
* do { // code } while (condition); runs the code first, then checks the condition.

# 2. Object-Oriented Programming (OOPs)

1. What are the main principles of OOPs in Java? Explain each.

Ans :

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of **"objects"**. Java is a purely object-oriented language (except for primitive types), and it follows the four main principles of OOP:

**1. Encapsulation**

**Definition**:  
Encapsulation is the process of **wrapping data (variables)** and **code (methods)** into a single unit called a **class**, and restricting direct access to some of the object's components.

**Benefits**:

* Improves security by hiding the internal state.
* Promotes modularity and maintainability.

**2. Inheritance**

**Definition**:  
Inheritance is the mechanism in Java by which one class **acquires the properties and behaviors** of another class using the extends keyword.

**Benefits**:

* Promotes code reusability.
* Establishes a relationship between parent and child classes.

**3. Polymorphism**

**Definition**:  
Polymorphism means **"many forms"**. It allows one interface to be used for a **different underlying form (data types or methods)**.

Types of Polymorphism:

* **Compile-time (Method Overloading)**
* **Runtime (Method Overriding)**

**Benefits**:

* Increases flexibility and scalability.
* Supports dynamic behavior in programs.

2. What is a class and an object in Java? Give examples.

Ans :

**Class in Java**

A **class** in Java is a **blueprint or template** for creating objects. It defines **properties (fields)** and **behaviors (methods)** that the objects created from the class will have.

**Example:**

class Car {

String color;

int speed;

void drive() {

System.out.println("Car is driving");

}

}

**Object in Java**

An **object** is a **real-world entity** created from a class. It has **state (data)** and **behavior (methods)** as defined in the class.

public class Main {

public static void main(String[] args) {

Car myCar = new Car(); // Object created

myCar.color = "Red";

myCar.speed = 120;

myCar.drive(); // Calling method

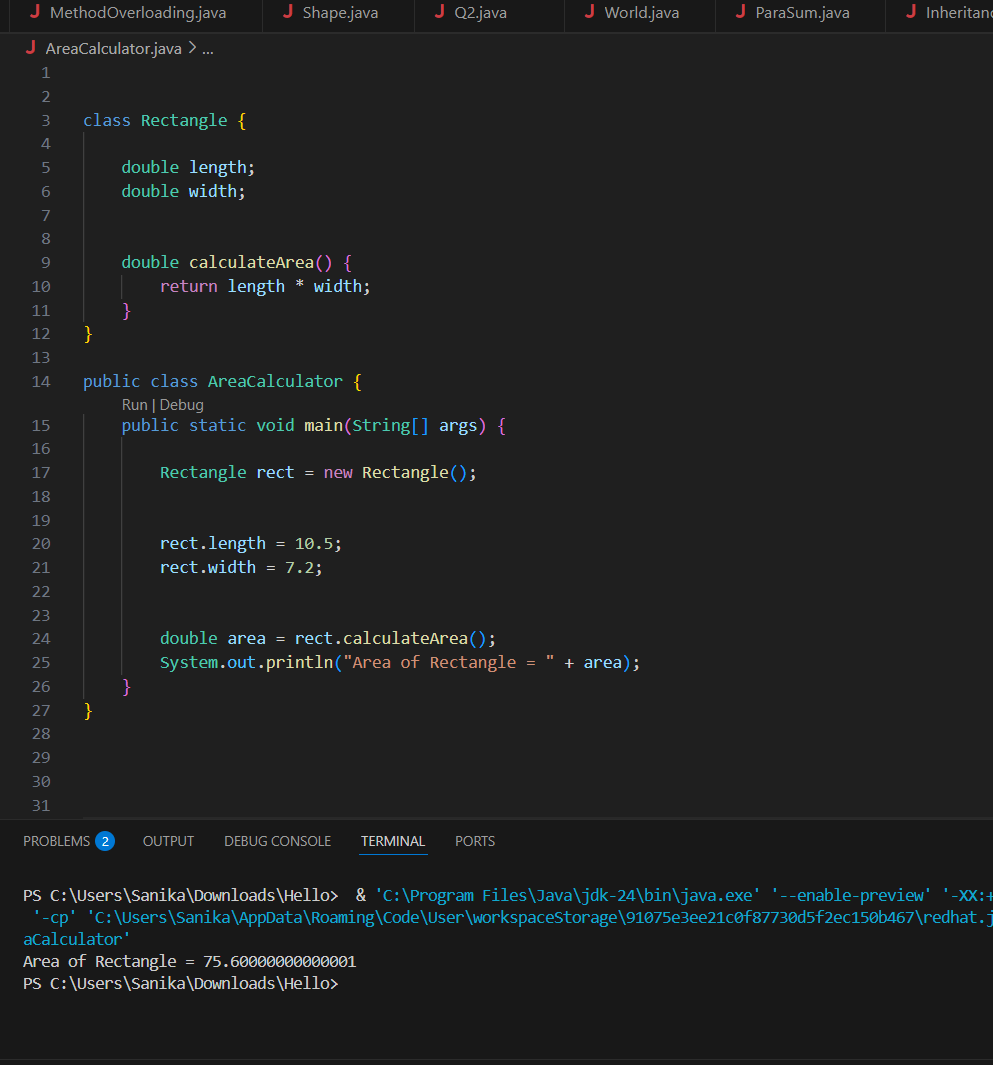
System.out.println("Color: " + myCar.color);

System.out.println("Speed: " + myCar.speed);

}

}

3. Write a program using class and object to calculate area of a rectangle.



1. Explain inheritance with real-life example and Java code.

**Inheritance** is a core principle of Object-Oriented Programming that allows one class (child/subclass) to **inherit the properties and behaviors** of another class (parent/superclass).

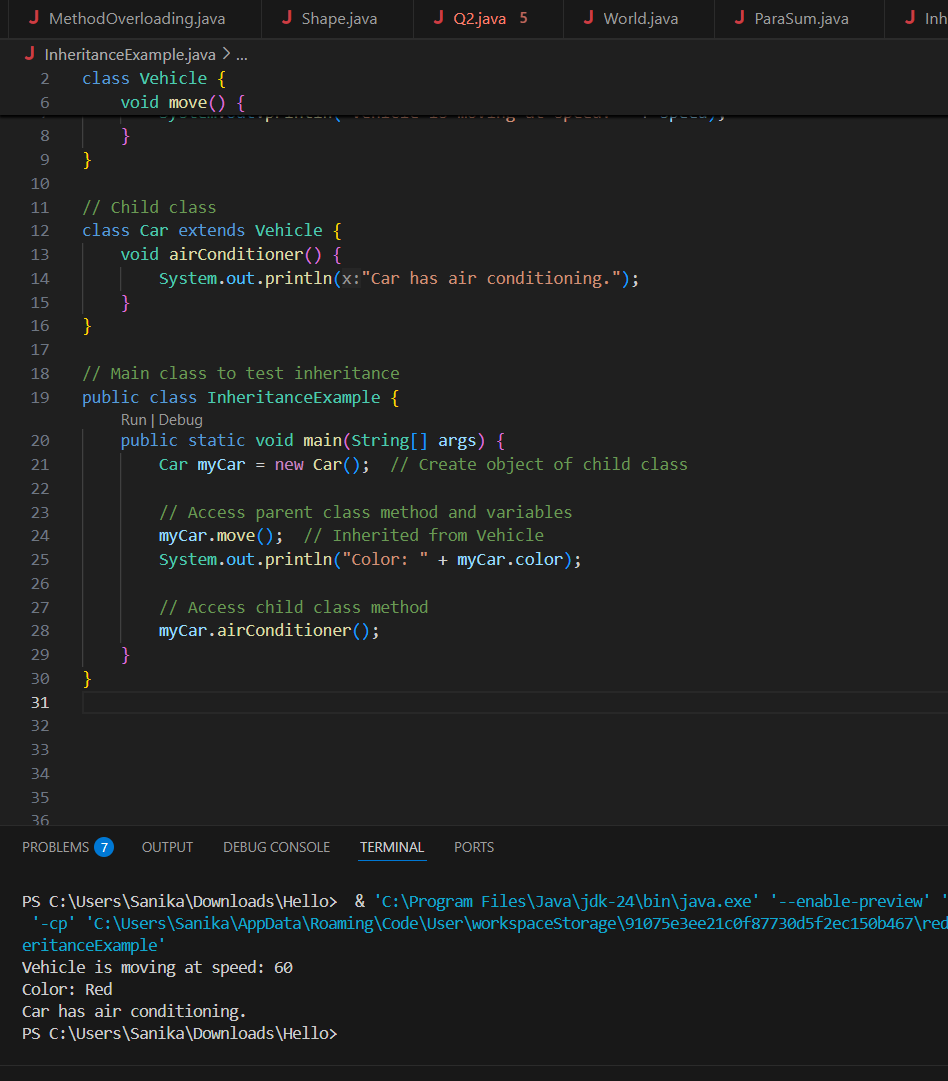
* It promotes **code reusability**.
* Implemented in Java using the extends keyword.

Imagine a **general class** Vehicle. All vehicles have **common features** like speed, color, and the ability to move().

Now, a **Car** is a type of Vehicle that also has its own specific features, like airConditioner().

So:

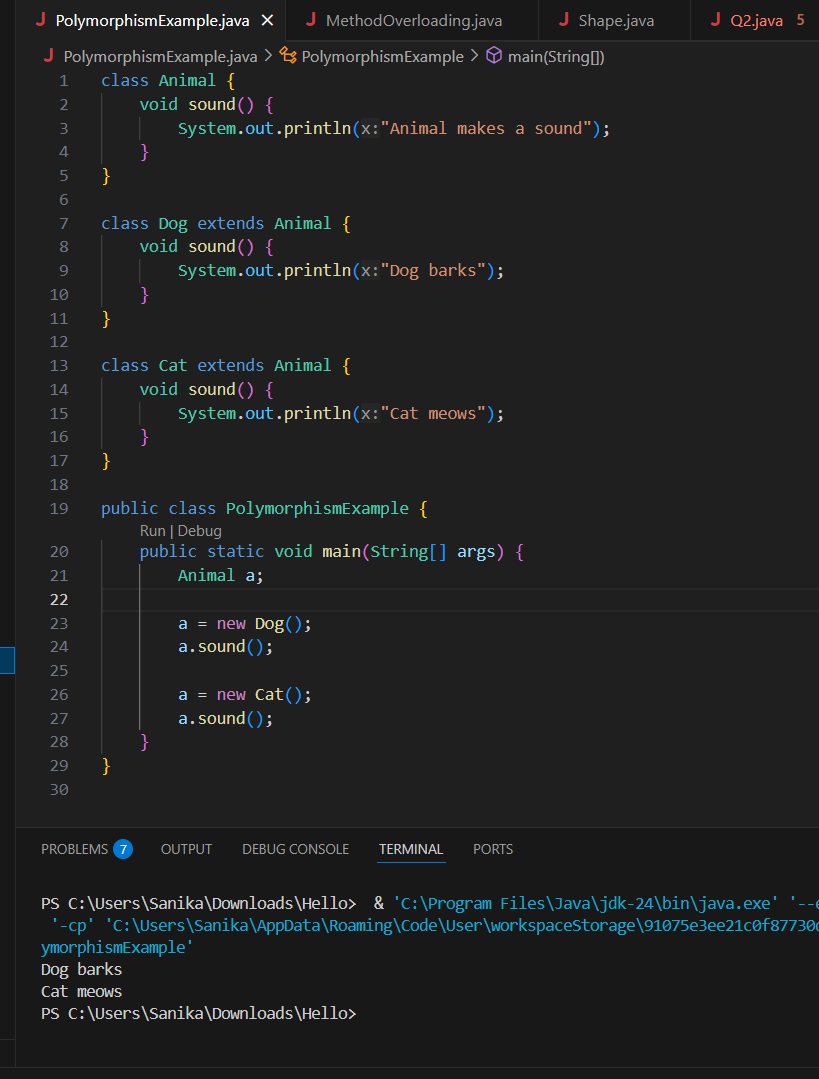
* Vehicle is the **parent class**.
* Car is the **child class** that **inherits** from Vehicle.



1. What is polymorphism? Explain with compile-time and runtime examples.

Ans :

Polymorphism means "many forms." It allows one interface (method name) to behave differently based on context. Java supports two types: 1. Compile-Time Polymorphism (Method Overloading) • Same method name, different parameters • Decided during compilation.

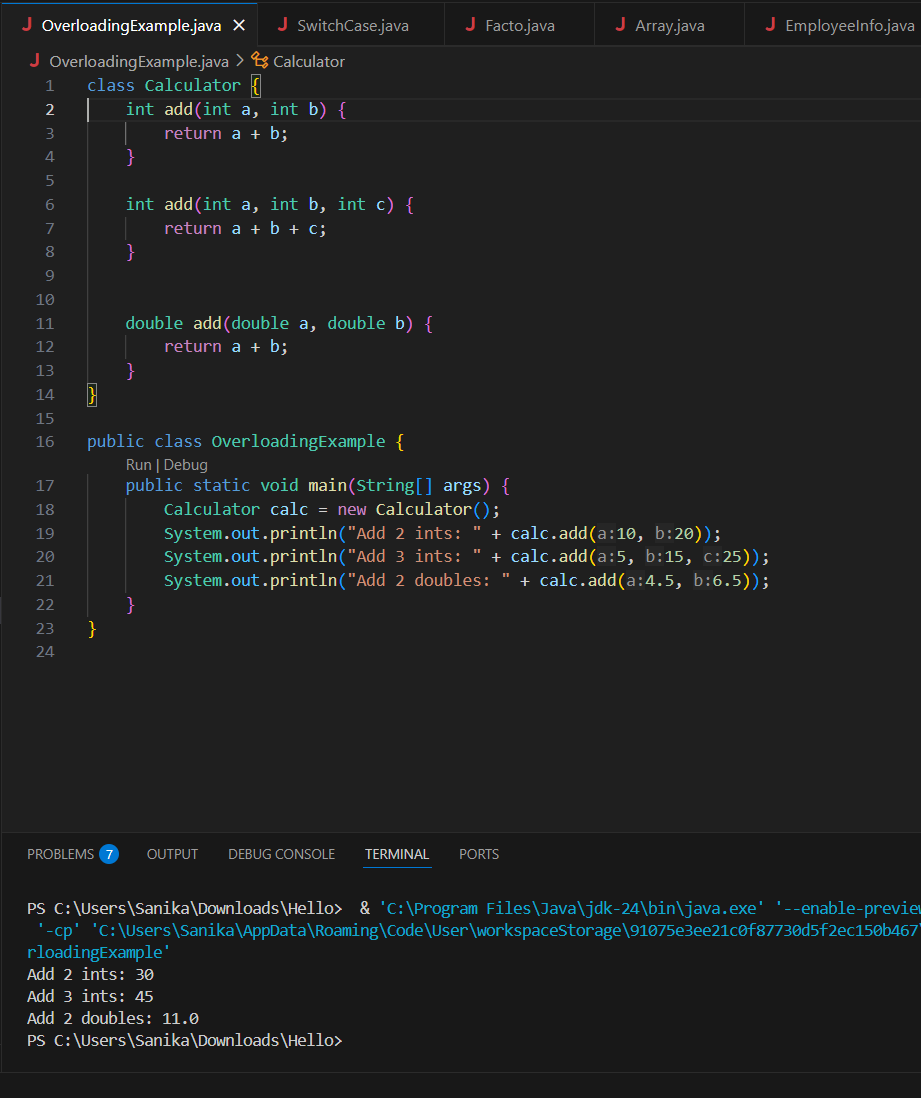


1. What is method overloading and method overriding? Show with examples.

Ans :

Method overloading means defining **multiple methods with the same name** but with **different parameters** (type, number, or order) **within the same class**.

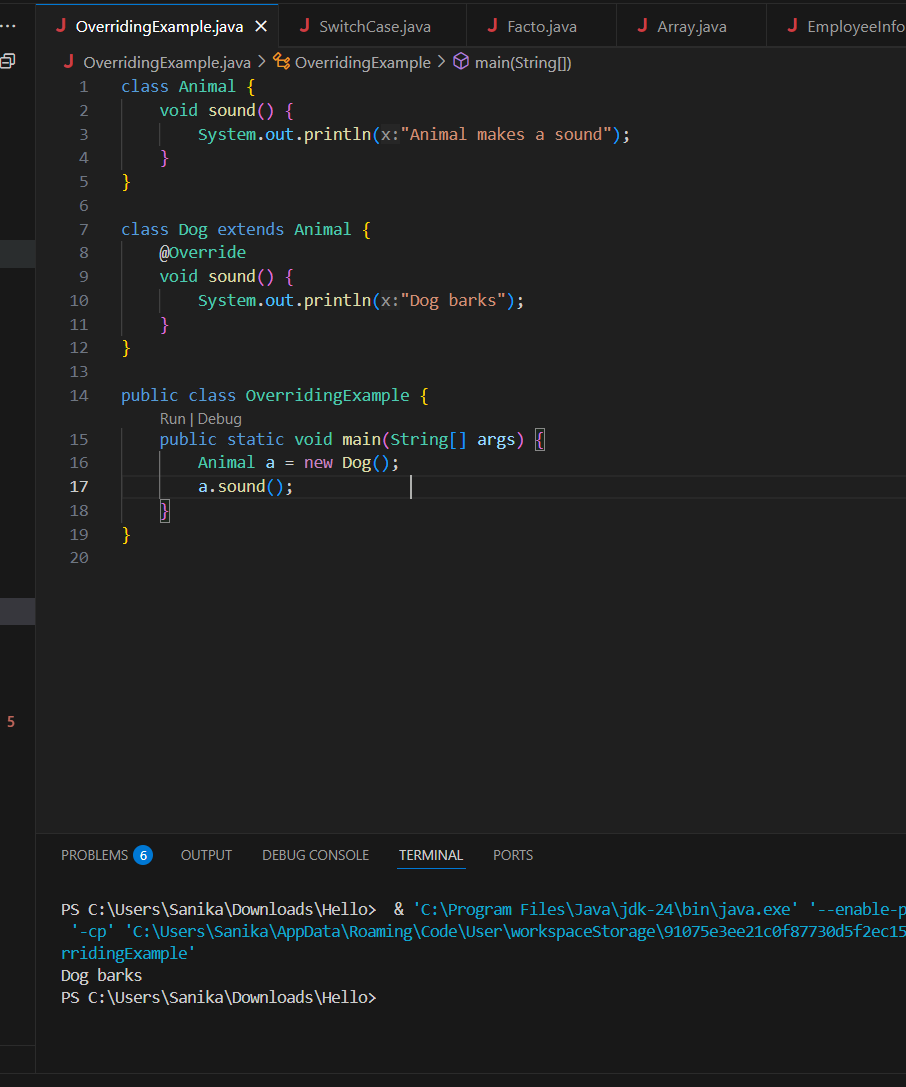
**Resolved at**: Compile-time.



**Method Overriding**

**Definition**:  
Method overriding means **redefining a method** of the **parent class** in a **child class** with the **same method name, return type, and parameters**.

**Resolved at**: Runtime



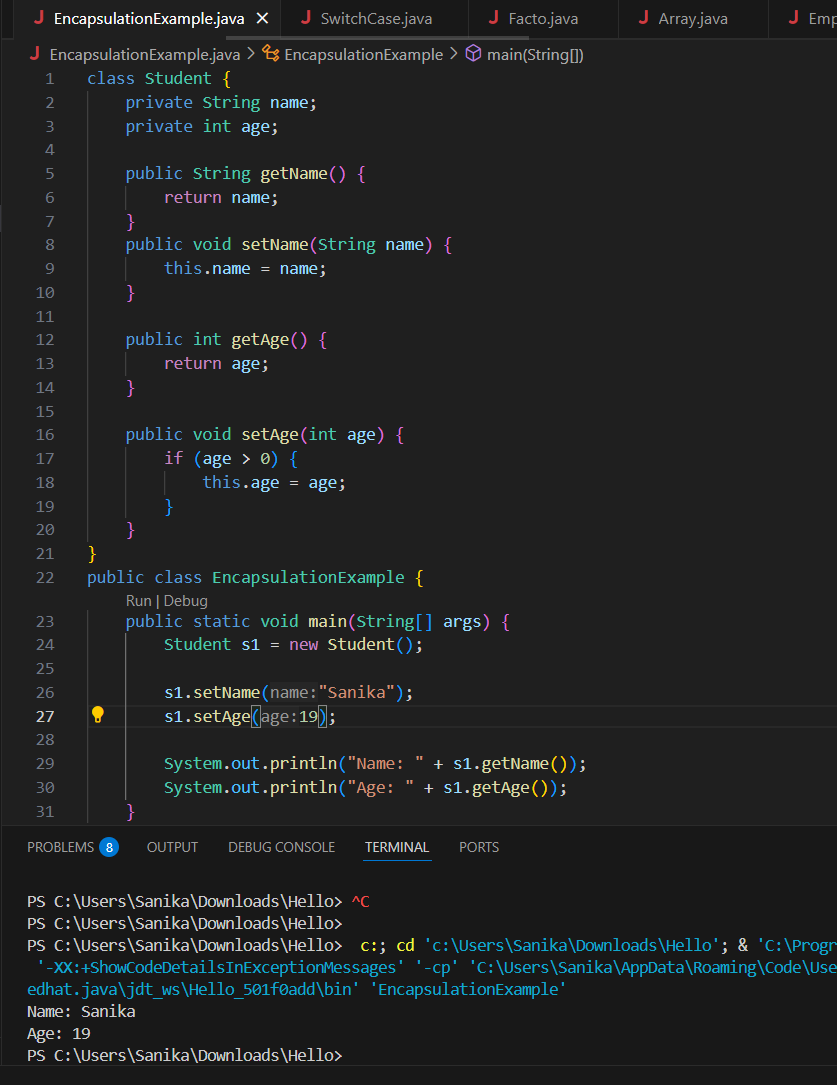
1. What is encapsulation? Write a program demonstrating encapsulation.

Ans :

**Encapsulation** is one of the four fundamental **Object-Oriented Programming (OOP)** principles in Java. It is the concept of **binding data (variables)** and **methods (functions)** that operate on that data into a single unit — a **class** — and **restricting direct access** to some of the object's components.

In Java, encapsulation is achieved by:

1. Making the class variables **private**.
2. Providing **public getter and setter methods** to access and modify them



1. What is abstraction in Java? How is it achieved?

Ans :

**Abstraction** in Java is the concept of **hiding the internal implementation details** and showing **only the essential features** of an object.

It allows the programmer to focus on **what** an object does rather than **how** it does it.

* When you drive a car, you use the **steering wheel, brake, and accelerator**, but you don’t need to know how the **engine** works internally.
* Similarly, in Java, abstraction hides the complex logic and shows only relevant functionalities.
* Abstraction in Java is achieved using:

| **Tool** | **Description** |
| --- | --- |
| **Abstract Class** | A class that cannot be instantiated. Can have both abstract and concrete methods. |
| **Interface** | A reference type that can contain only abstract methods (Java 7) or default/static methods too (Java 8+). |

9. Explain the difference between abstract class and interface.

Ans :

| **Feature** | **Abstract Class** | **Interface** |
| --- | --- | --- |
| **Definition** | A class that cannot be instantiated and may contain both abstract and concrete methods. | A collection of abstract methods (and default/static methods from Java 8 onward). |
| **Keyword Used** | abstract | interface |
| **Method Types** | Can have abstract and non-abstract methods | Only abstract methods (Java 7); can have default & static methods (Java 8+) |
| **Variables** | Can have instance variables (with any access modifier) | Only constants: public static final variables |
| **Constructor** | Can have a constructor | Cannot have a constructor |
| **Access Modifiers** | Can use public, protected, private | Methods are public abstract by default |
| **Inheritance Type** | Supports **single inheritance** only | Supports **multiple inheritance** via interfaces |
| **When to Use** | When classes are **closely related** and share base functionality | When you want to **define a contract** for unrelated classes |
| **Example Usage** | abstract class Animal { abstract void sound(); } | interface Shape { void draw(); } |

1. Create a Java program to demonstrate the use of interface.

