Samruddhi Jagdale

Java Basics & OOPs Assignment Answers

## Q1. What is Java? Explain its features.

**Java** is a high-level, object-oriented programming language developed by **Sun Microsystems** (now owned by **Oracle**). It is designed to be **platform-independent** using the concept of "**Write Once, Run Anywhere**".

#### ****Key Features of Java:****

* **Platform Independent:** Java code runs on any device with a JVM.
* **Object-Oriented:** Everything in Java is treated as an object.
* **Simple and Easy to Learn**
* **Secure:** Java provides a secure environment through a security manager and bytecode verification.
* **Robust:** Strong memory management and exception handling.
* **Multithreaded:** Supports multithreaded programming.
* **Portable:** Java programs can be moved from one platform to another without changes.
* **High Performance:** With Just-In-Time (JIT) compilers, performance is improved.

## Q2. Explain the Java program execution process.

### Step 1: Writing Java Code using Notepad

1. Open **Notepad**.
2. Type the JAVA code.
3. Save the file

### Step 2: Set Environment Variable for Java (One-time setup)

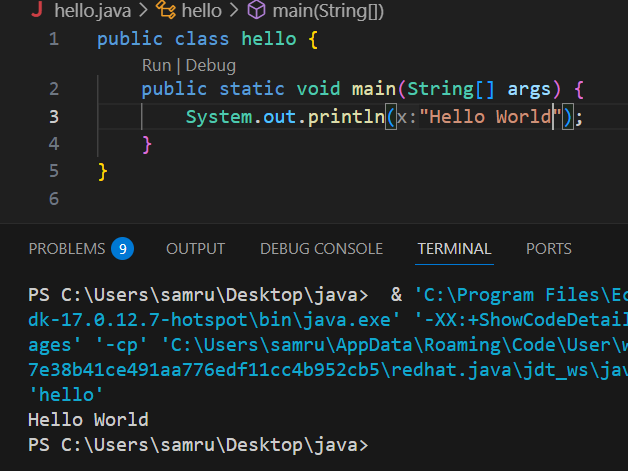
Required only if you're compiling Java from CMD.

1. Find where JDK is installed:
2. Set environment variable:

## Step 3: Compile and Run Java Program Using CMD

### Writing & Running Java in ****VS Code****

1. **Install VS Code**
2. Install **Java Extension Pack** from Extensions tab
3. Create a new file: HelloWorld.java
4. Write the same program as above
5. Click **Run** (or use Terminal)

**Q3. Write a simple Java program to display 'Hello World'.**

**Q4. What are data types in Java? List and explain them.**

Data types specify the type of data a variable can hold.  
- Primitive Data Types: byte, short, int, long, float, double, char, boolean  
- Non-Primitive Data Types: String, Arrays, Classes, Interfaces

**Q5. What is the difference between JDK, JRE, and JVM?**

- JVM: Executes bytecode   
- JRE: Provides libraries and JVM to run Java programs  
- JDK: Includes JRE + development tools (compiler, debugger)

**Q6. What are variables in Java? Explain with examples.**

Variables store data values.  
Example:  
int age = 25;  
String name = "Samruddhi";  
Types of variables:

* **Local**: inside methods
* **Instance**: non-static variables in a class
* **Static**: shared among all objects of a class

**Q7. What are the different types of operators in Java?**

Java supports several types of **operators**:

* **Arithmetic Operators**: +, -, \*, /, %
* **Relational (Comparison)**: ==, !=, >, <, >=, <=
* **Logical Operators**: &&, ||, !
* **Assignment Operators**: =, +=, -=, \*=, /=
* **Unary Operators**: ++, --, +, -
* **Bitwise Operators**: &, |, ^, ~, <<, >>

**Q8. Explain control statements in Java (if, if-else, switch).**

**Control Statements** control the flow of execution.

**if**

if (age > 18) {

System.out.println("Adult");

}

**if-else**

if (age > 18) {

System.out.println("Adult");

} else {

System.out.println("Not an adult");

}

**switch**

int day = 3;

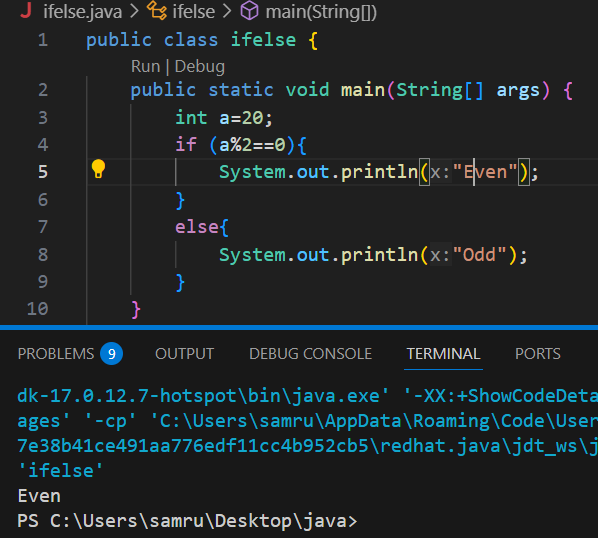
switch(day) {

case 1: System.out.println("Monday"); break;

case 2: System.out.println("Tuesday"); break;

default: System.out.println("Other Day");

}

**Q9. Write a Java program to find whether a number is even or odd**

**Q10. What is the difference between while and do-while loop?**

| **while loop** | **do-while loop** |
| --- | --- |
| Condition is checked before execution. | Condition is checked after execution. |
| May not run at all if the condition is false. | Runs **at least once** even if the condition is false. |

#### ****Example:****

// while loop

int i = 1;

while (i <= 5) {

System.out.println(i);

i++;

}

// do-while loop

int j = 1;

do {

System.out.println(j);

j++;

} while (j <= 5);

## Q11. What are the main principles of OOPs in Java? Explain each.

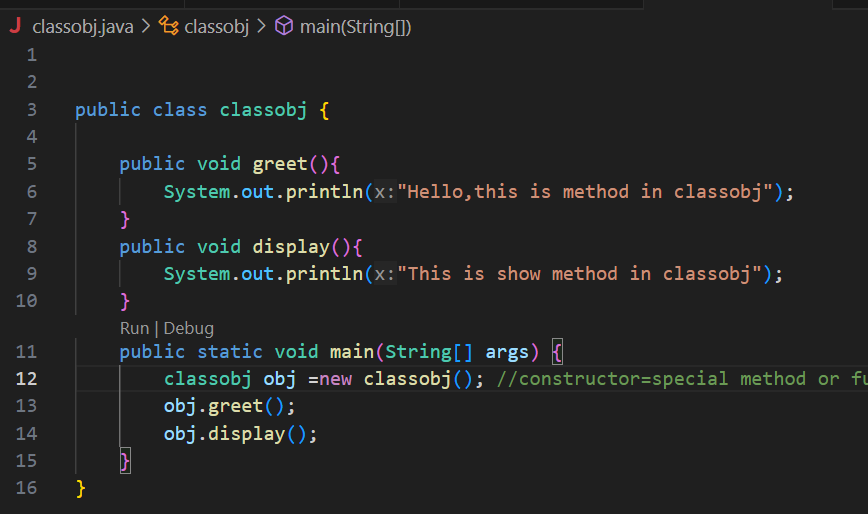
Java follows 4 main **Object-Oriented Programming** principles:

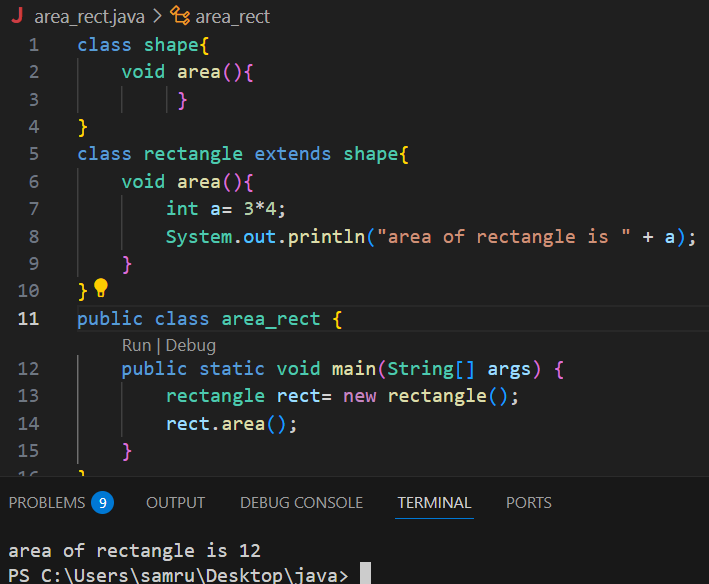
1. **Encapsulation** – Binding data (variables) and code (methods) together.
2. **Abstraction** – Hiding internal details and showing only essential features.
3. **Inheritance** – Acquiring properties of one class into another.
4. **Polymorphism** – Performing the same task in different ways.

## Q12. What is a class and an object in Java? Give examples.

**Class**: A blueprint for creating objects. It defines properties and behaviors.

**Object**: A real-world entity created from a class.

**Example:**

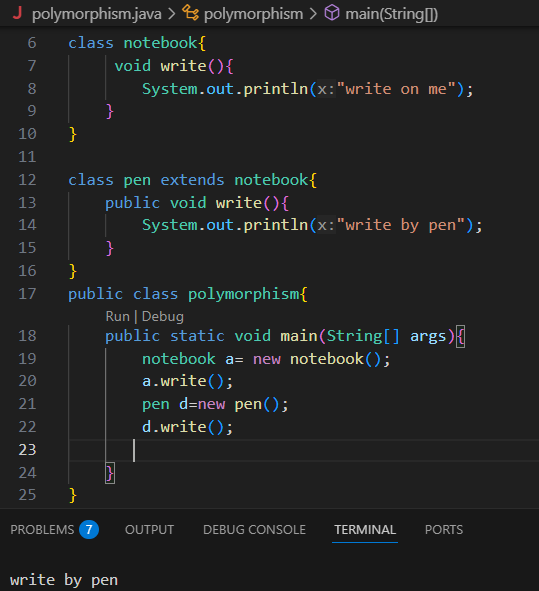
**Q13. Write a program using class and object to calculate area of a rectangle.**

Q14. Explain inheritance with real-life example and Java code.

## Real-life example: A Dog is a subclass of Animal. It inherits features like legs, eating, etc. Screenshot 2025-06-27 145401.png

## Q15. What is polymorphism? Explain with compile-time and runtime examples.

Polymorphism means many forms.  
- Compile-time (Method Overloading)  
- Runtime (Method Overriding)

Example:  


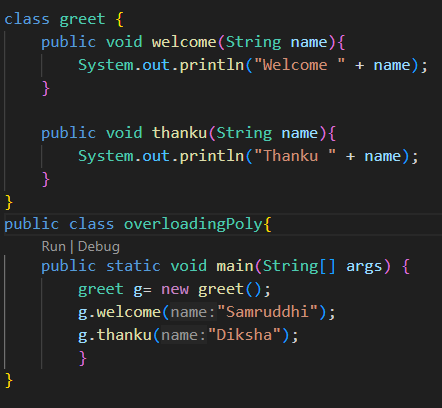
## Q16. What is method overloading and method overriding? Show with examples.

Overloading: Same method name, different parameters

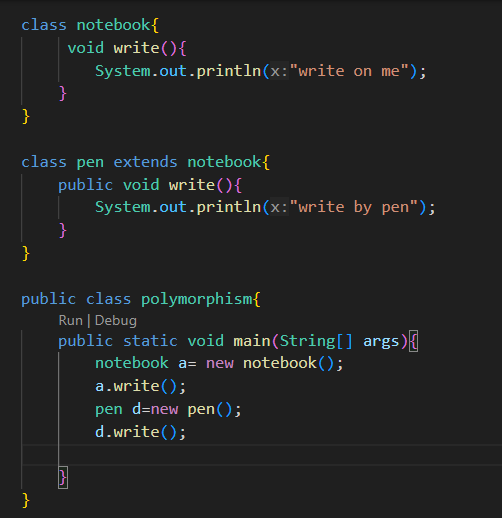
Overriding: Subclass provides specific implementation of parent method

**Example:**

**Overloading:**

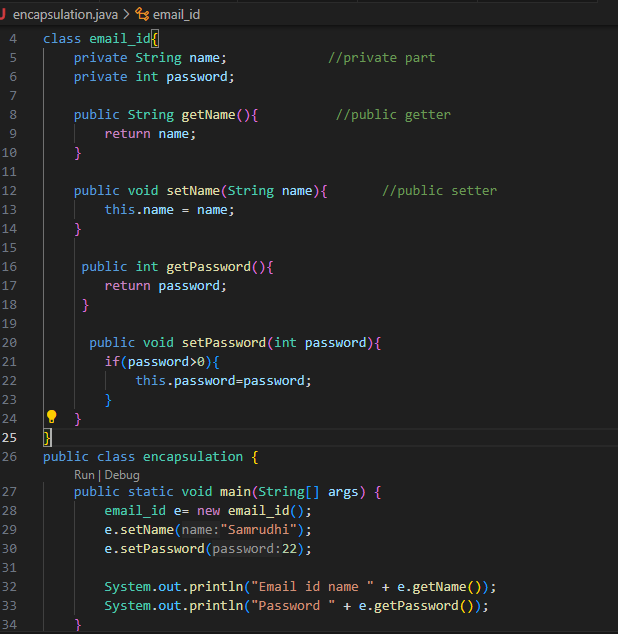
****

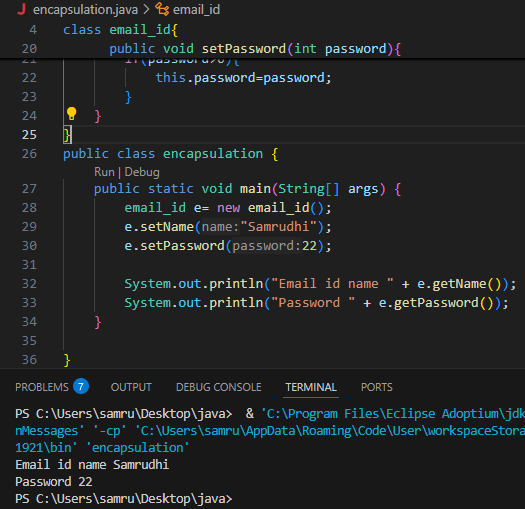
**Overriding:**



## Q17. What is encapsulation? Write a program demonstrating encapsulation.

**Encapsulation** means hiding internal data using **private variables** and accessing them via **public methods**.  
**Example:**



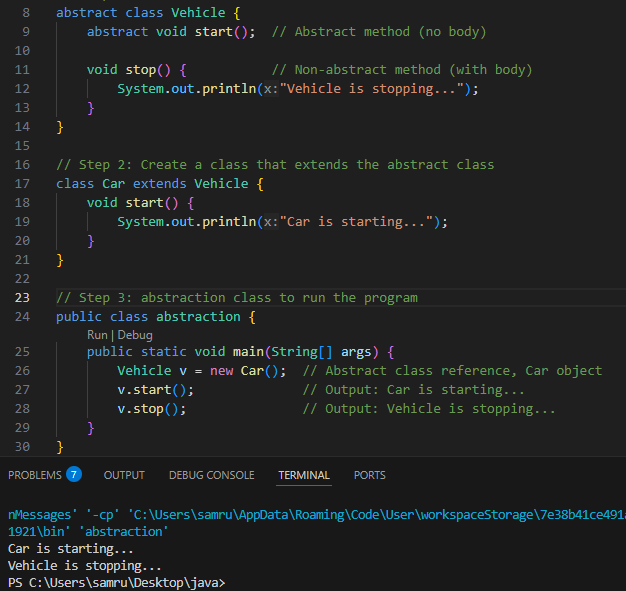


**Q18. What is abstraction in Java? How is it achieved?**

**Abstraction** hides implementation details and shows only essential features.

In Java, it is achieved using:

* **Abstract Classes**
* **Interfaces**



**Q19. Explain the difference between abstract class and interface.**

| **Feature** | **Abstract Class** | **Interface** |
| --- | --- | --- |
| Methods | Can have both abstract and concrete methods | All methods are abstract (Java 7), default/static allowed (Java 8+) |
| Variables | Can have variables | Only public static final variables |
| Inheritance | Can extend only one abstract class | Can implement multiple interfaces |
| Use Case | When base class provides common functionality | For complete abstraction |

## Q20. Create a Java program to demonstrate the use of interface.

