Saaili Amborkar

Java Basics & OOPs Assignment Questions

Java Basics

1. What is Java? Explain its features.

ANS: Java is a popular and easy-to-learn programming language developed in 1995 by Sun Microsystems (now Oracle). It is used to build apps, websites, and software.

Key Features are:

- Simple – Easy to learn and use.
- Object-Oriented – Code is clean and reusable.
- Platform Independent – Runs on any device with JVM.
- Secure & Robust – Safe and handles errors well.
- Portable – Write once, run anywhere.
- Multithreaded – Can do many tasks at once.
- High Performance – Fast due to JIT compiler.

2. Explain the Java program execution process.

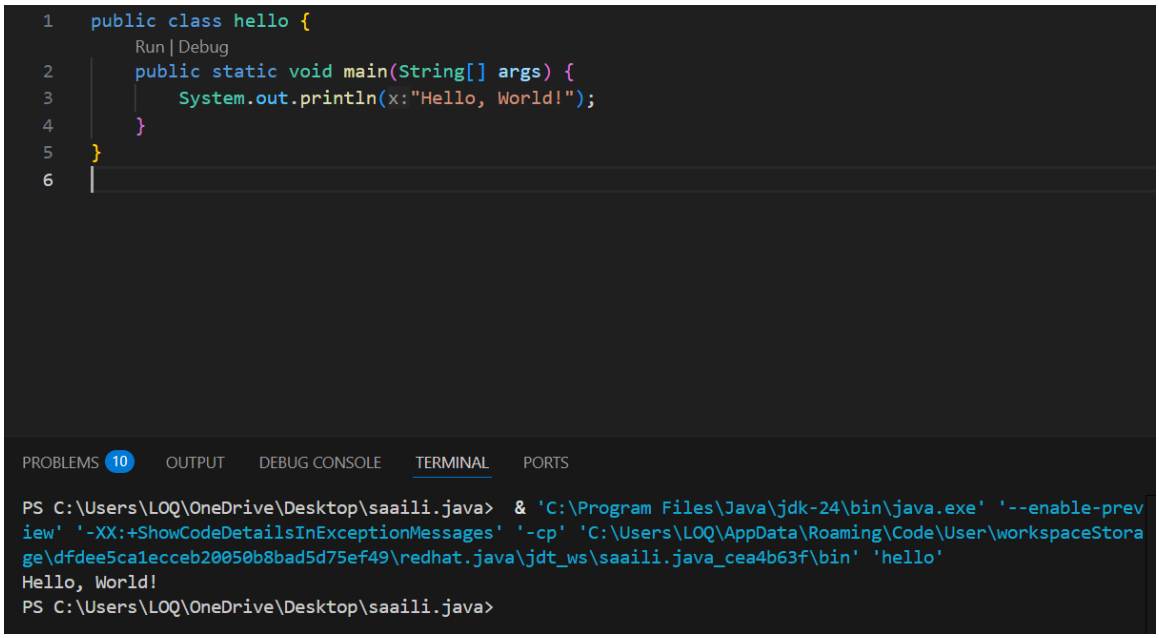
ANS: To run a Java program, we first write the code in a file like Demo.java. Then, the compiler changes it into bytecode. The Java Virtual Machine (JVM) reads this bytecode and runs the program. Finally, we see the output on the screen.

Steps: Write code → Compile → JVM runs → See output

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

ANS:

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



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

ANS: In Java, data types tell the computer what kind of value a variable can hold. There are two main types of data types:

1. Primitive Data Types (Basic types)

These are built-in and used to store simple values.

- Non-Numeric Types:
  1. boolean – Stores true or false.
  2. char – Stores a single character (like 'A').
- Numeric Types:
  1. Integer Types (whole numbers):

byte, short, int, long

Example: int age = 20;

2. Floating Point Types (decimal numbers):

float, double

Example: float price = 99.99f;

2. Non-Primitive Data Types (Reference types)

These are used for more complex data.

- String – Stores text.
- Array – Stores a list of values.
- etc. – Includes classes, interfaces, and more.

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

ANS: The difference between JDK, JRE, and JVM are:

1. JVM (Java Virtual Machine):

- It runs Java programs.
- It reads bytecode and converts it into machine code.
- It is part of JRE and JDK.

  Think of it as the engine that runs your Java program.

2. JRE (Java Runtime Environment):

- It provides the environment to run Java programs.
- It includes the JVM + libraries needed to run programs.

  (It's for running Java code, not writing it.)

3. JDK (Java Development Kit):

- It is used to write and run Java programs.
- It includes the JRE, JVM, and tools like the compiler (javac).

  (It's for developers who want to write and run Java code.)

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

ANS : In Java, a variable is a name given to a memory location that stores a value.

It is used to store data that can change while the program runs.

Types of Variables in Java:

1. **Local Variable** – Declared inside a method.
2. **Instance Variable** – Declared inside a class but outside methods.
3. **Static Variable** – Declared using static keyword, shared by all objects of the class.

CODE:

```
public class variable {
    int age = 20;           // instance variable
    static String name = "Saaili";  // static variable

    public void display() {
        int marks = 90;     // local variable
        System.out.println("Name: " + name);
```

```
        System.out.println("Age: " + age);
        System.out.println("Marks: " + marks);
    }

    public static void main(String[] args) {
        variable obj = new variable();
        obj.display();
    }
}
```

Output:

```
saaili.java_cea4b63f\bin' 'variable'
Name: Saaili
Age: 20
Marks: 90
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

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

ANS: The different types of operators in Java are:

1. **Arithmetic Operators**

```
ArithmeticDemo.java > ArithmeticDemo > main(String[])
1   public class ArithmeticDemo {
        Run | Debug
2       public static void main(String[] args) {
3           int a = 10;
4           int b = 20;
5
6           // Arithmetic Operations
7           System.out.println("Addition: " + (a + b));
8           System.out.println("Subtraction: " + (a - b));
9           System.out.println("Multiplication: " + (a * b));
10          System.out.println("Division: " + (a / b));
11          System.out.println("Modulus: " + (a % b));
12
13
14
15

PROBLEMS 12   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

iew' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStora
ge\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'ArithmeticDemo'
Addition: 30
Subtraction: -10
Multiplication: 200
Division: 0
Modulus: 10
```

2. Relational Operators

relational.java > relational > main(String[])

```java
public class relational {

    Run | Debug
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        System.out.println("a == b: " + (a == b));
        System.out.println("a != b: " + (a != b));
        System.out.println("a > b: " + (a > b));
        System.out.println("a < b: " + (a < b));
        System.out.println("a >= b: " + (a >= b));
        System.out.println("a <= b: " + (a <= b));
    }


```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-prev
iew' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStora
ge\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'relational'
a == b: false
a != b: true
a > b: false
a < b: true
a >= b: false
a <= b: true
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

3. Logical Operators

logical.java > logical > main(String[])

```java
public class logical {
    Run | Debug
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        // Logical AND (&&)
        System.out.println((a < b) && (b > 15));

        // Logical OR (||)
        System.out.println((a > b) || (b == 20));

        // Logical NOT (!)
        boolean result = (a > b);
        System.out.println(!result);
}
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-prev
iew' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStora
ge\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'logical'
true
true
true
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

4. Assignment Operators

```java
public class assignment {
    Run | Debug
    public static void main(String[] args){

        int a = 100;
        int b = 200;
        a += b;
        System.out.println("The value of a is: " + a);

        int c = 50;
        c %= 2;
        System.out.println("The value of c is: " + c);

    }
}
```

```
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'assignment'
The value of a is: 300
The value of c is: 0
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

5. Bitwise Operators

```java
Bitwise.java > Bitwise > main(String[])
public class Bitwise {
    Run | Debug
    public static void main(String[] args) {
        int a = 5;
        int b = 3;
        System.out.println("a & b: " + (a & b));
        System.out.println("a | b: " + (a | b));
        System.out.println("a ^ b: " + (a ^ b));
        System.out.println("~a: " + (~a));
        System.out.println("a << 1: " + (a << 1));
        System.out.println("a >> 1: " + (a >> 1));
    }
}
```

```
PROBLEMS 12  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-prev
iew' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStora
ge\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Bitwise'
a & b: 1
a | b: 7
a ^ b: 6
~a: -6
a << 1: 10
a >> 1: 2
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

6. Ternary Operator

```java
Ternary.java > Ternary > main(String[])
public class Ternary {

    Run | Debug
    public static void main(String[] args) {
        int age = 18;
        String result = (age >= 18) ? "You are an adult" : "You are a minor";

        System.out.println(result);
    }

}
```

```
PROBLEMS 12  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-prev
iew' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStora
ge\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Ternary'
You are an adult
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

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

ANS: Control statements in Java are used to make decisions in a program. They help the program choose different actions based on conditions.

1) **if** is use when you have one condition.

2) **if-else** is use when you have two possible choices.

3) **switch** is use when checking multiple values.

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

ANS:

oddeven.java > oddeven

```java
public class oddeven{
    Run | Debug
    public static void main(String[] args) {

        int a=45;

        if(a % 2 == 0) {
            System.out.println(x:"a is even");
        } else {
            System.out.println(x:"a is odd");
        }
    }
}
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Fil
odeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roami
f49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'oddeven'
a is odd
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

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

ANS:

1. In a **while loop**, the condition is checked before the loop runs. If the condition is false at the start, the loop will not run at all.

```java
public class whileloop {
    Run | Debug
    public static void main(String[] args) {
        int i = 1;

        while (i <= 5) {
            System.out.println(x:"Saaili");
            i++;
        }
    }
}
```

```
PROBLEMS 14  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-prev
iew' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStora
ge\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'whileloop'
Saaili
Saaili
Saaili
Saaili
Saaili
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

2. In a **do-while loop**, the code runs at least once, and then the condition is checked. Even if the condition is false, the loop will run one time.

```java
public class Dowhile {
    Run | Debug
    public static void main(String[] args) {
        int i = 1;

        do {
            System.out.println(x:"Saaili");
            i++;
        } while (i <= 5);
    }
}
```

```
PROBLEMS 14  OUTPUT  DEBUG CONSOLE  TERMINAL

TERMINAL
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>  & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-preview' '-X
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb2005
d5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Dowhile'
Saaili
Saaili
Saaili
Saaili
Saaili
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

Object-Oriented Programming (OOPs)

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

ANS: Main Principles of OOP in Java are:

1. **Encapsulation** – Binding data and methods into one unit (class).

2. **Inheritance** – One class can inherit properties from another class.

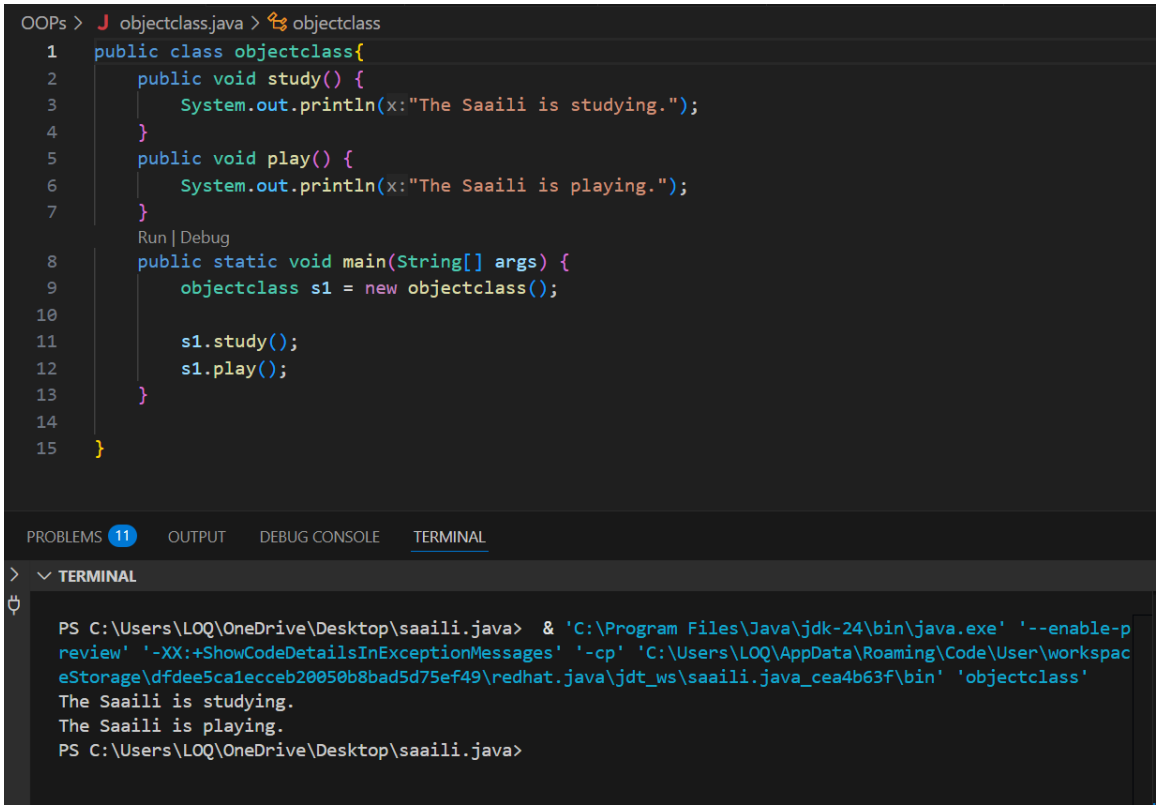
3. **Polymorphism** – One task can be performed in different ways.

4. **Abstraction** – Hiding internal details and showing only important features.

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

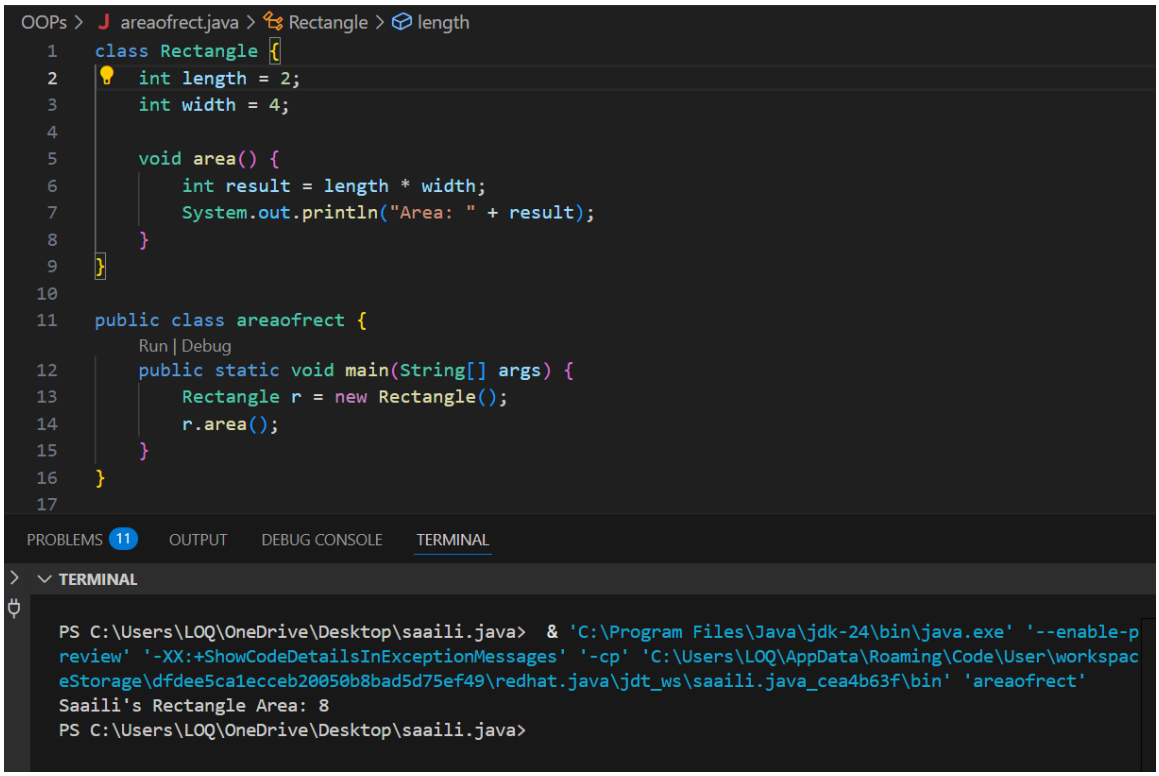
ANS: 1) Class: A blueprint for creating objects.

2)Object: A real-world entity created using a class.



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

ANS:

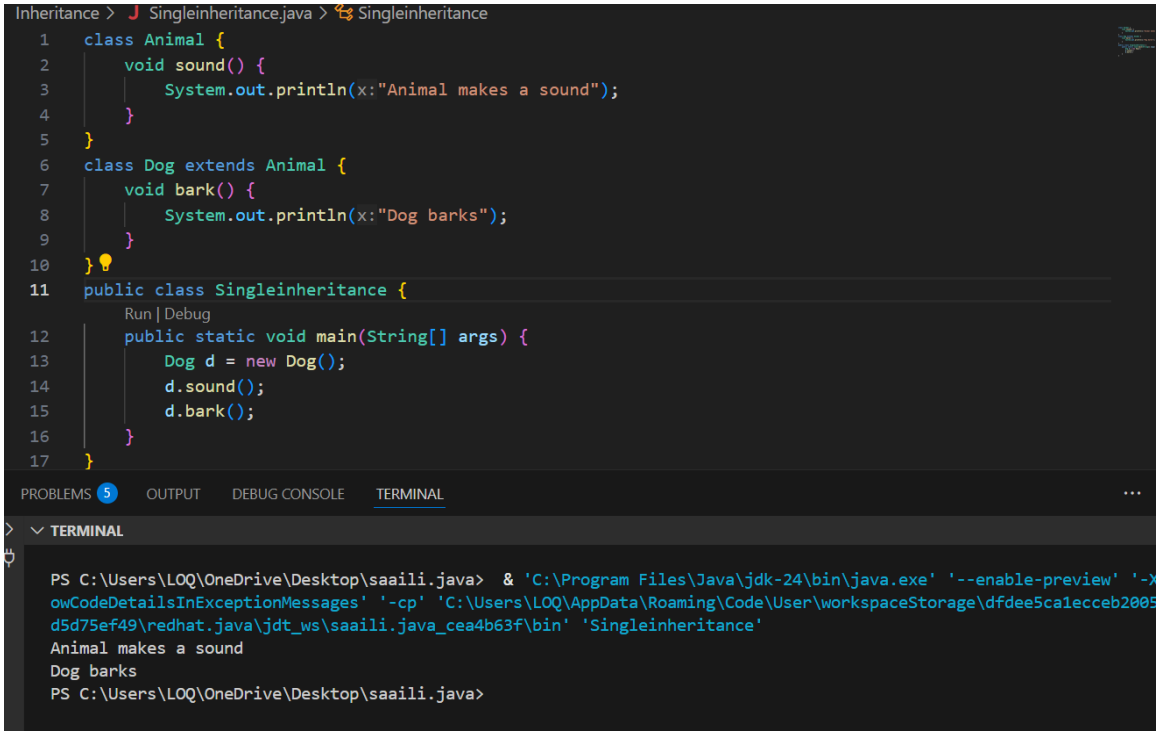


4. Explain inheritance with real-life example and Java code.

ANS: Inheritance is one of the main principles of Object-Oriented Programming (OOP).

It allows one class (called the child or subclass) to use the properties and methods of another class (called the parent or superclass).

In simple words, Inheritance means “acquiring the features of another class.”



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

ANS: Polymorphism means "many forms" — it allows the same method or object to behave differently based on the context.

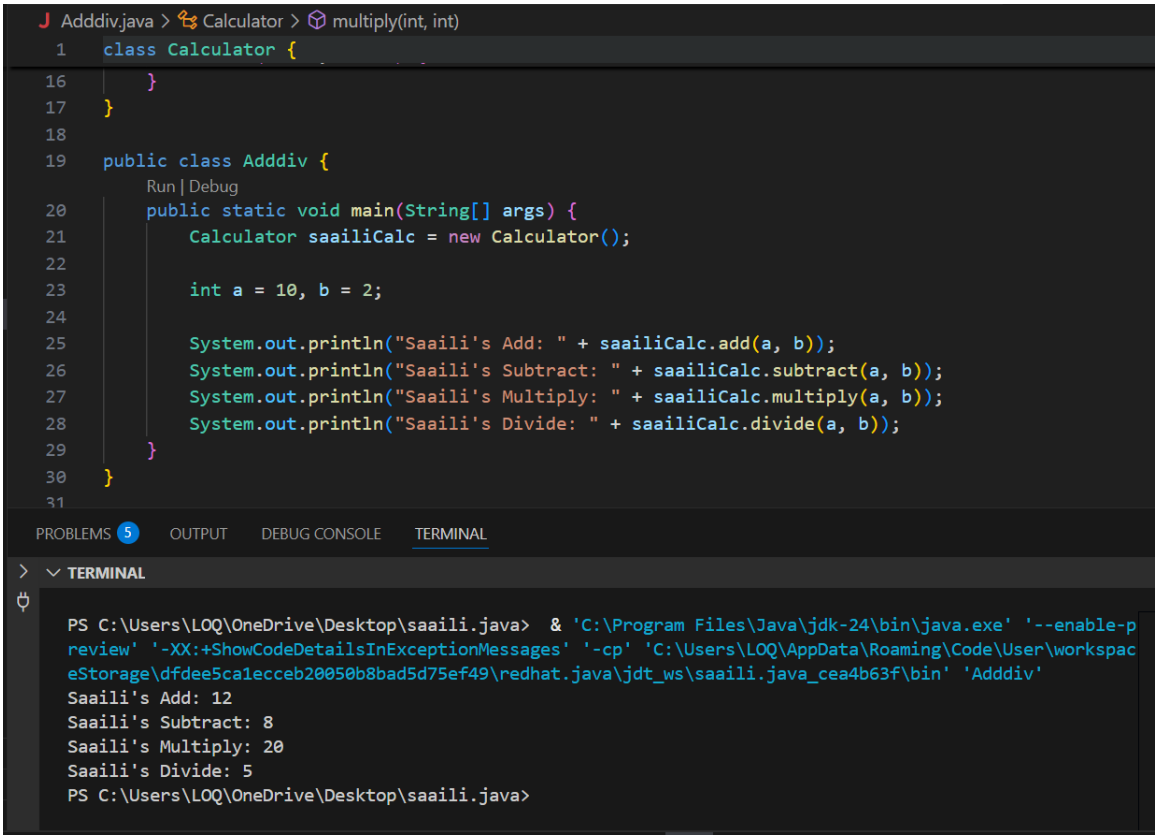
In Java, polymorphism helps you perform one action in different ways.

1. **Compile-Time Polymorphism (Method Overloading)**
    - Same method name with different parameters.
    - Decided at compile time.
2. **Run-Time Polymorphism (Method Overriding)**
    - Child class provides its own version of a method from the parent class.
    - Decided at runtime.

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

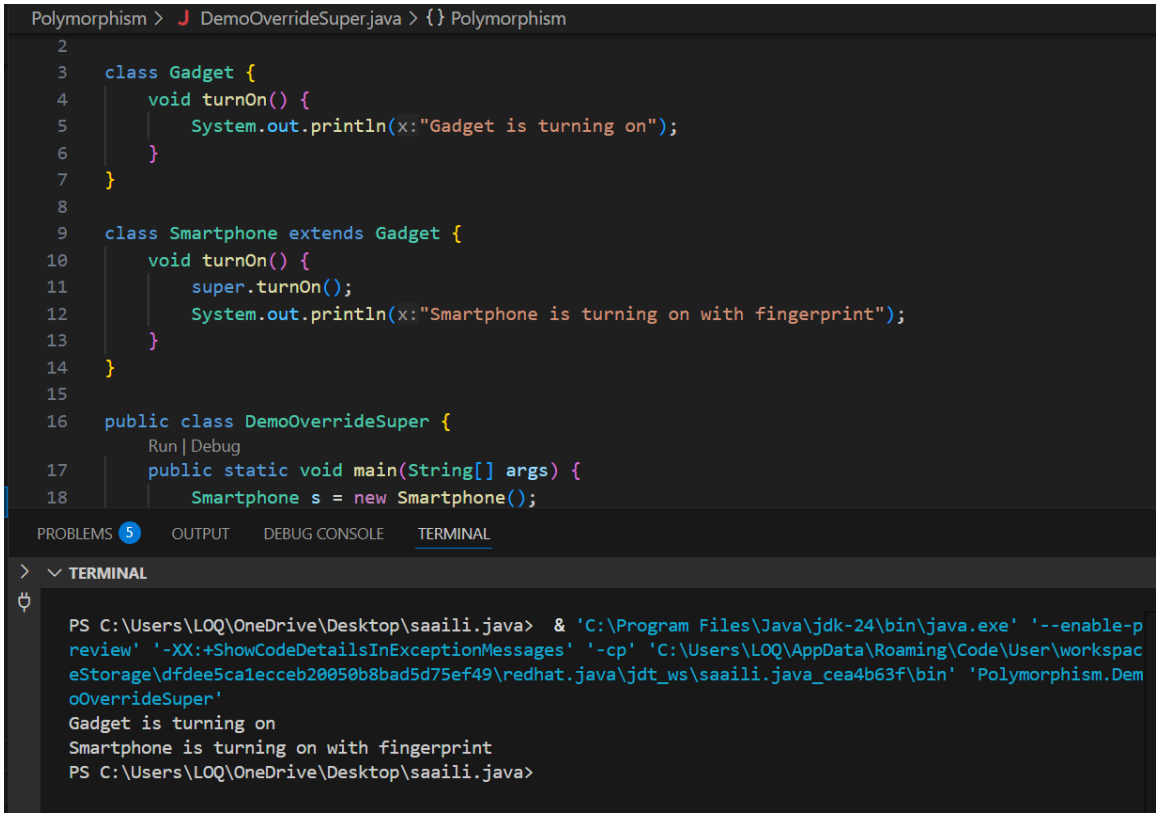
ANS: Method Overloading : It means having multiple methods with the same name but with different parameters (number, type, or order) in the same class.

It is a type of compile-time polymorphism in Java.



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

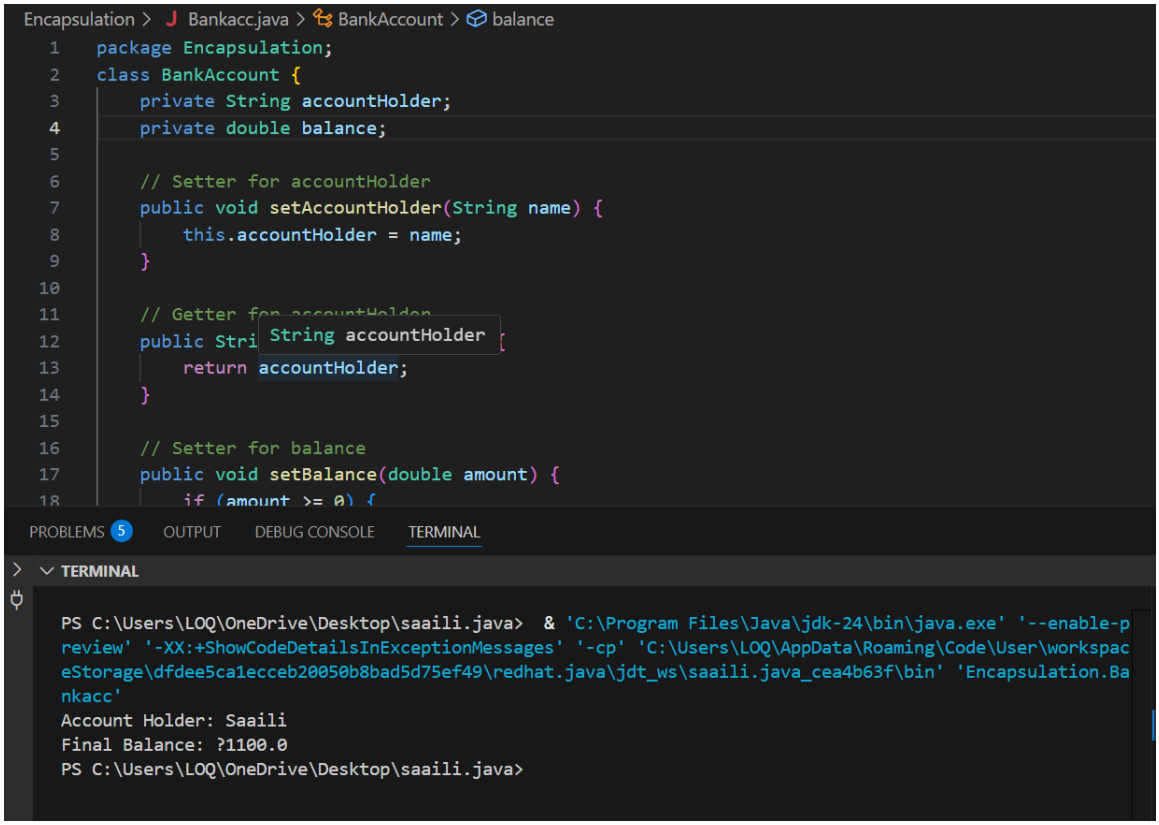
It allows the child class to provide its own version of a method that it inherited from the parent class.



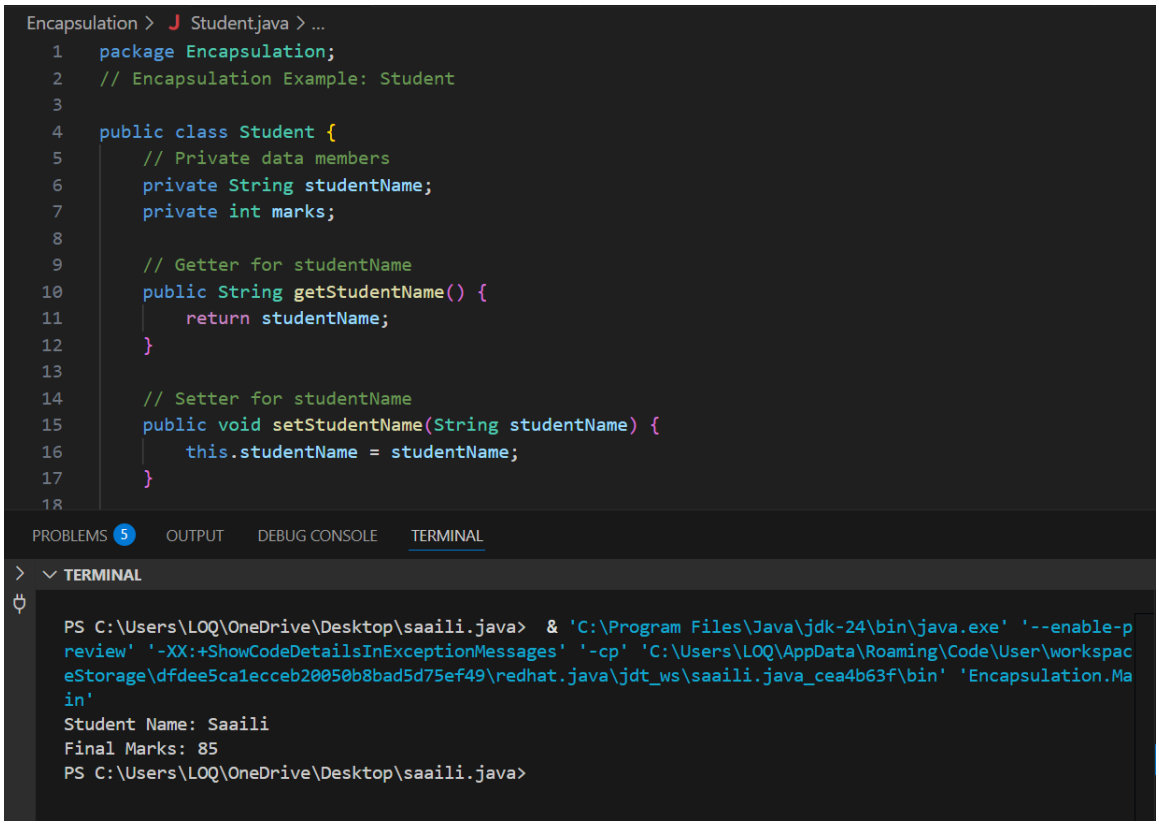
7. What is encapsulation? Write a program demonstrating encapsulation.

ANS: Encapsulation is a mechanism in Java where the data (variables) of a class is hidden from other classes and can be accessed only through the methods (getters and setters) of that class.

Example1:


Encapsulation > Bankacc.java > BankAccount > balance
1 package Encapsulation;
2 class BankAccount {
3 private String accountHolder;
4 private double balance;
5
6 // Setter for accountHolder
7 public void setAccountHolder(String name) {
8 this.accountHolder = name;
9 }
10
11 // Getter for accountHolder
12 public Stri String accountHolder
13 return accountHolder;
14 }
15
16 // Setter for balance
17 public void setBalance(double amount) {
18 if (amount >= 0) {
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Encapsulation.Bankacc'
Account Holder: Saaili
Final Balance: ?1100.0
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>

Example2:


Encapsulation > Student.java > ...
1 package Encapsulation;
2 // Encapsulation Example: Student
3
4 public class Student {
5 // Private data members
6 private String studentName;
7 private int marks;
8
9 // Getter for studentName
10 public String getStudentName() {
11 return studentName;
12 }
13
14 // Setter for studentName
15 public void setStudentName(String studentName) {
16 this.studentName = studentName;
17 }
18
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Encapsulation.Main'
Student Name: Saaili
Final Marks: 85
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>

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

ANS: Abstraction in Java is a process of hiding the implementation details and showing only the essential features of an object.It helps reduce complexity and allows the programmer to focus on what the object does instead of how it does it.

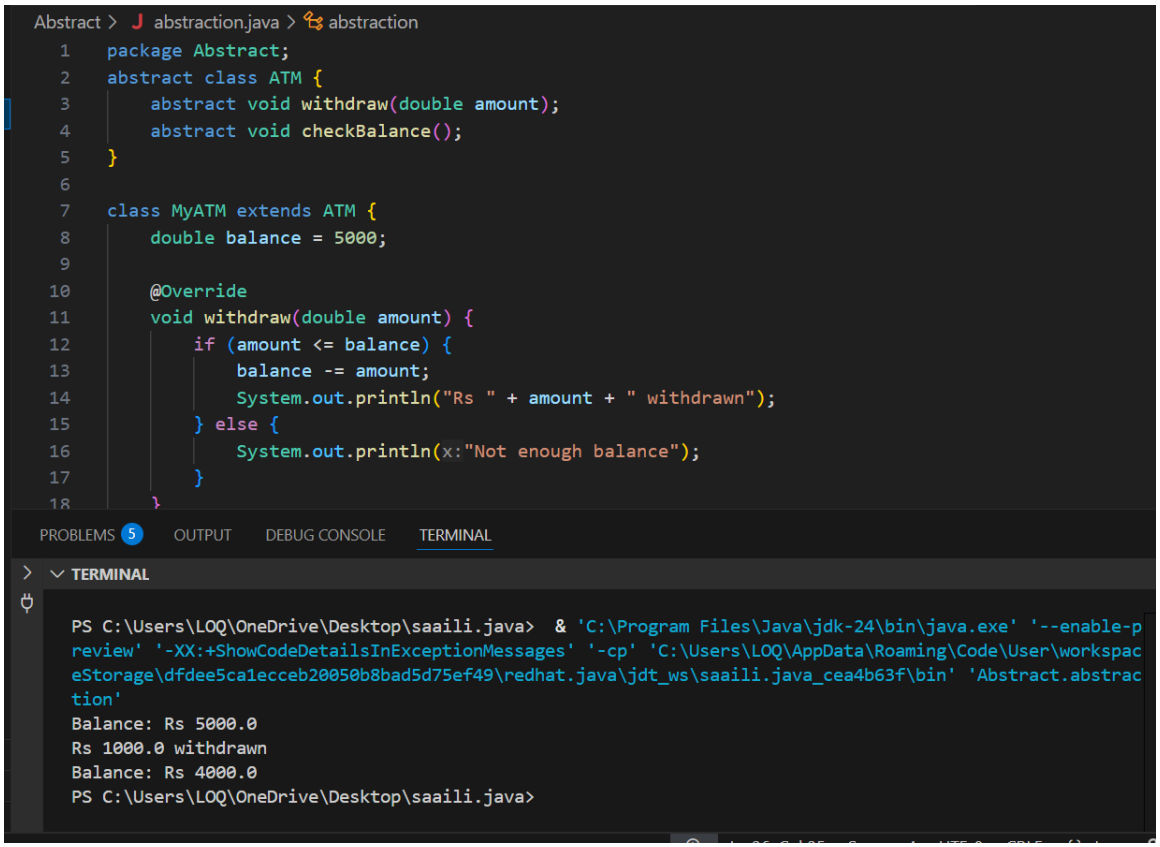
Abstraction in Java is achieved in two ways:

1. **Using Abstract Classes**
2. **Using Interfaces**

9. Explain the difference between abstract class and interface.

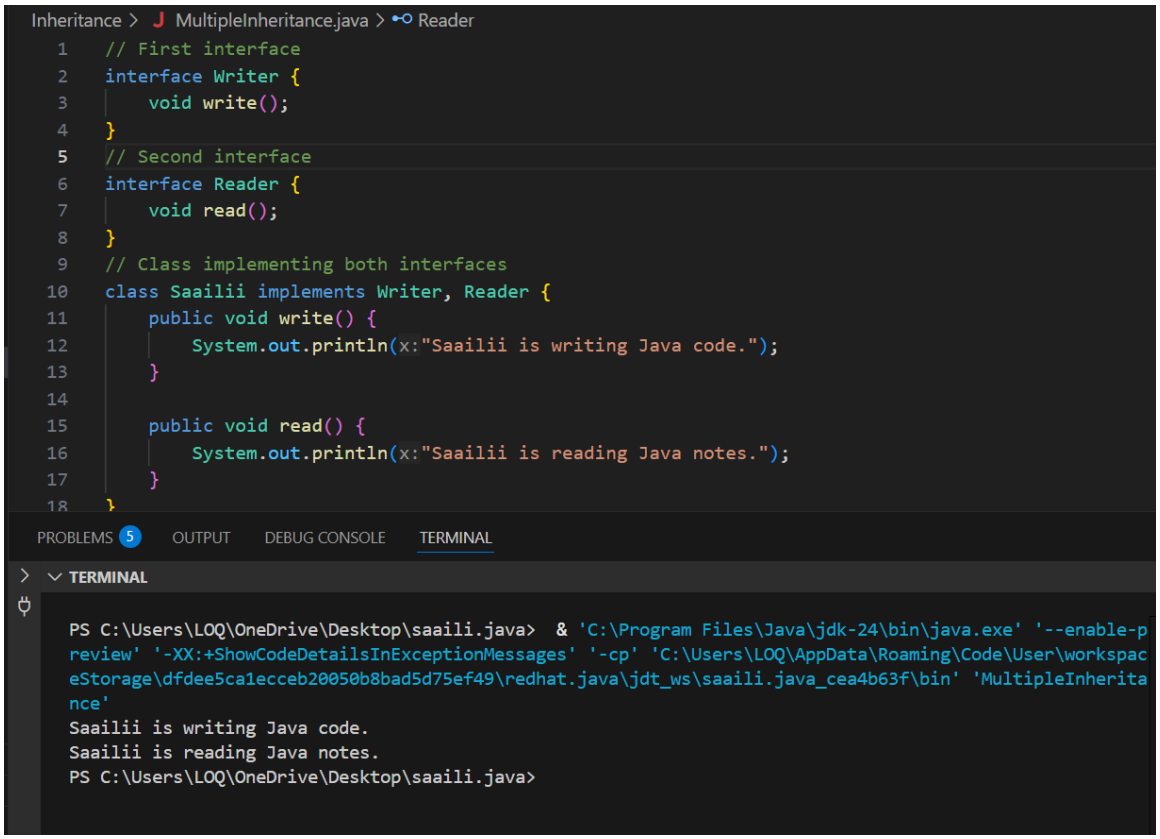
ANS: 1) Abstract class

1. Declared using abstract keyword.
2. Can have both abstract and normal methods.
3. Cannot create object of abstract class.
4. Can have variables and constructors.
5. Used for code sharing in related classes.
6. Supports single inheritance only.
7. Gives partial abstraction (not 100%).



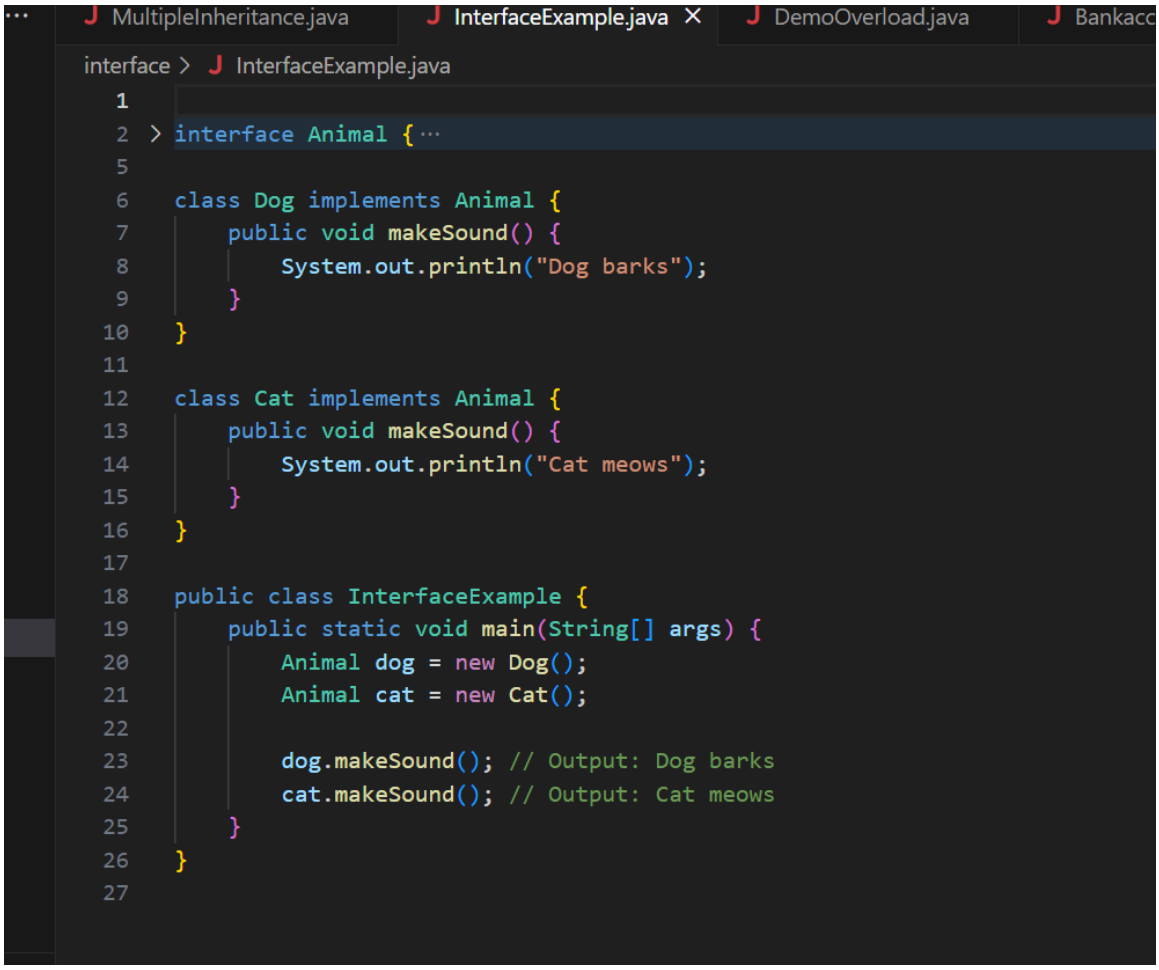
2) Interface

1. Declared using `interface` keyword.
2. Can only have abstract methods (Java 7), and default/static methods (Java 8+).
3. Cannot create object of an interface.
4. All variables are public, static, and final by default.
5. Used to define a contract or behavior.
6. Supports multiple inheritance.
7. Provides 100% abstraction (in Java 7).



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

ANS:


MultipleInheritance.java
InterfaceExample.java
DemoOverload.java
Bankacc
interface > InterfaceExample.java
1
2 interface Animal {···
5
6 class Dog implements Animal {
7 public void makeSound() {
8 System.out.println("Dog barks");
9 }
10 }
11
12 class Cat implements Animal {
13 public void makeSound() {
14 System.out.println("Cat meows");
15 }
16 }
17
18 public class InterfaceExample {
19 public static void main(String[] args) {
20 Animal dog = new Dog();
21 Animal cat = new Cat();
22
23 dog.makeSound(); // Output: Dog barks
24 cat.makeSound(); // Output: Cat meows
25 }
26 }
27