

# 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");  
    }  
}
```

```
1 public class hello {
2     Run | Debug
3     public static void main(String[] args) {
4         System.out.println("Hello, World!");
5     }
6 }
```

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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' 'hello'
Hello, World!
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

#### 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:**

```

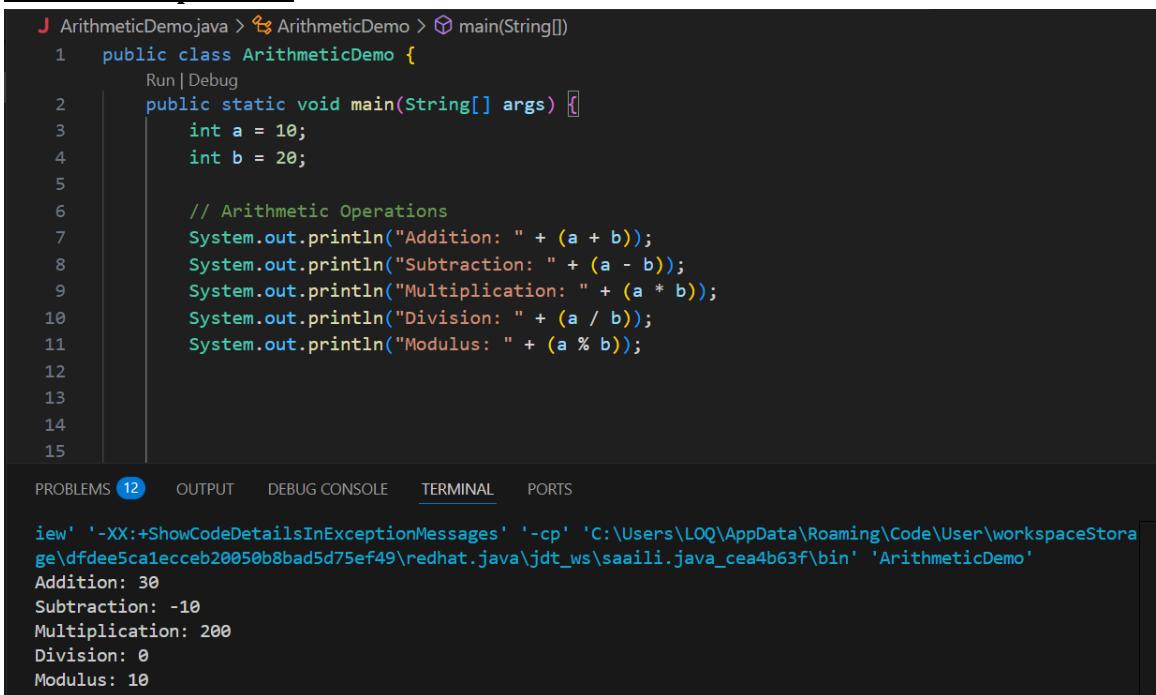
saaili.java_cea4b63f\bin' 'variable'
Name: Saaili
Age: 20
Marks: 90
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>

```

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

**ANS:** The different types of operators in Java are:

### 1. Arithmetic Operators



```

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

```

PROBLEMS 12    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```

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

```

## 2. Relational Operators

The screenshot shows a Java code editor with a dark theme. A Java file named `relational.java` is open. The code defines a class `relational` with a `main` method. Inside the `main` method, variables `a` and `b` are assigned values 10 and 20 respectively. Then, six `System.out.println` statements are used to print the results of various relational comparisons between `a` and `b`. The output terminal shows the results of these prints.

```
J relational.java > ↵ relational > ⚙ main(String[])
1  public class relational {
2
3      Run | Debug
4      public static void main(String[] args) {
5          int a = 10;
6          int b = 20;
7
8          System.out.println("a == b: " + (a == b));
9          System.out.println("a != b: " + (a != b));
10         System.out.println("a > b: " + (a > b));
11         System.out.println("a < b: " + (a < b));
12         System.out.println("a >= b: " + (a >= b));
13         System.out.println("a <= b: " + (a <= b));
14     }
15 }
```

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-preview' '-X:ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\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

The screenshot shows a Java code editor with a dark theme. A Java file named `logical.java` is open. The code defines a class `logical` with a `main` method. Inside the `main` method, variables `a` and `b` are assigned values 10 and 20 respectively. The code then demonstrates three logical operations: AND (`&&`), OR (`||`), and NOT (`!`). The output terminal shows the results of these logical expressions.

```
J logical.java > ↵ logical > ⚙ main(String[])
1  public class logical {
2
3      Run | Debug
4      public static void main(String[] args) {
5          int a = 10;
6          int b = 20;
7
8          // Logical AND (&&)
9          System.out.println((a < b) && (b > 15));
10
11         // Logical OR (||)
12         System.out.println((a > b) || (b == 20));
13
14         // Logical NOT (!)
15         boolean result = (a > b);
16         System.out.println(!result);
17     }
18 }
```

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-preview' '-X:ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'logical'
true
true
true
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

#### **4. Assignment Operators**

The screenshot shows a Java code editor with a dark theme. A Java file named `assignment.java` is open. The code demonstrates assignment operators:

```
J assignment.java > assignment
1 public class assignment {
2     Run | Debug
3     public static void main(String[] args){
4         int a = 100;
5         int b = 200;
6         a += b;
7         System.out.println("The value of a is: " + a);
8
9         int c = 50;
10        c %= 2;
11        System.out.println("The value of c is: " + c);
12
13    }
14
15 }
```

Below the code editor, there is a terminal window showing the execution of the Java program:

```
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-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**

```
J Bitwise.java > ⚙ Bitwise > ⚡ main(String[])
1  public class Bitwise {
2      Run | Debug
3      public static void main(String[] args) {
4          int a = 5;
5          int b = 3;
6          System.out.println("a & b: " + (a & b));
7          System.out.println("a | b: " + (a | b));
8          System.out.println("a ^ b: " + (a ^ b));
9          System.out.println("~a: " + (~a));
10         System.out.println("a << 1: " + (a << 1));
11         System.out.println("a >> 1: " + (a >> 1));
12     }
13 }
```

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-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\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

```
J Ternary.java > ⚙ Ternary > ⚡ main(String[])
1  public class Ternary {
2
3      Run | Debug
4      public static void main(String[] args) {
5          int age = 18;
6          String result = (age >= 18) ? "You are an adult" : "You are a minor";
7
8          System.out.println(result);
9      }
10 }
11 }
```

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-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\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:**

The screenshot shows a Java code editor with a dark theme. The code in the editor is:

```
J oddeven.java > oddeven
1 public class oddeven{
2     Run | Debug
3     public static void main(String[] args) {
4         int a=45;
5
6         if(a % 2 == 0) {
7             System.out.println("a is even");
8         } else {
9             System.out.println("a is odd");
10        }
11    }
12 }
```

Below the code editor, there is a terminal window showing the output of the program:

```
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.

A screenshot of a Java code editor showing a file named `whileloop.java`. The code contains a simple `while` loop that prints the word "Saaili" five times. Below the code, a terminal window shows the output of running the program.

```
J whileloop.java > ...
1  public class whileloop {
2      Run | Debug
3      public static void main(String[] args) {
4          int i = 1;
5
6          while (i <= 5) {
7              System.out.println("Saaili");
8              i++;
9          }
10 }
11 
```

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-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\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.

A screenshot of a Java code editor showing a file named `Dowhile.java`. The code contains a `do-while` loop that prints the word "Saaili" five times. Below the code, a terminal window shows the output of running the program.

```
J Dowhile.java > Dowhile
1  public class Dowhile {
2      Run | Debug
3      public static void main(String[] args) {
4          int i = 1;
5
6          do {
7              System.out.println("Saaili");
8              i++;
9          } while (i <= 5);
10 }
11 
```

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' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb20050b8bad5d75ef49\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.

```
OOPS > J objectclass.java > ↵ objectclass
 1  public class objectclass{
 2      public void study() {
 3          System.out.println("The Saaili is studying.");
 4      }
 5      public void play() {
 6          System.out.println("The Saaili is playing.");
 7      }
 8      Run | Debug
 9      public static void main(String[] args) {
10          objectclass s1 = new objectclass();
11
12          s1.study();
13          s1.play();
14      }
15  }
```

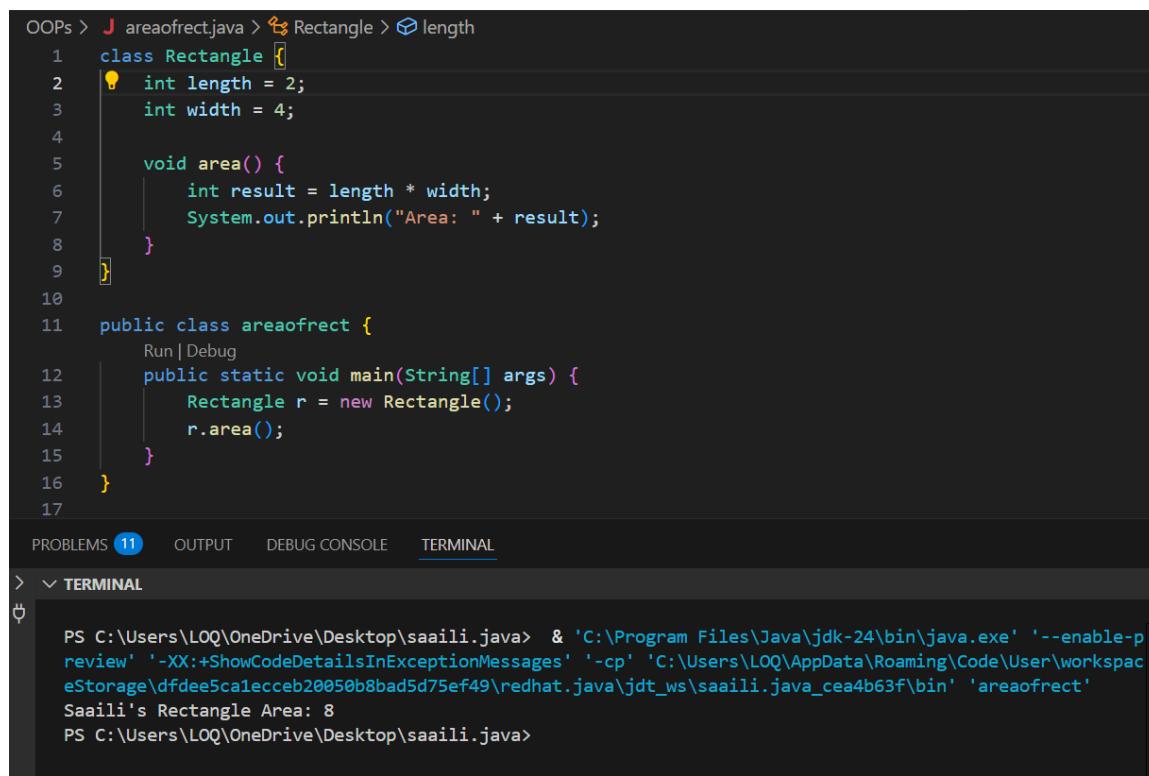
PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL

> ⇕ TERMINAL

PS C:\Users\LOQ\OneDrive\Desktop\saailli.java & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-p  
review' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspac  
eStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt\_ws\saailli.java\_cea4b63f\bin' 'objectclass'  
The Saaili is studying.  
The Saaili is playing.  
PS C:\Users\LOQ\OneDrive\Desktop\saailli.java>

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

## ANS:



The screenshot shows a Java code editor interface with a dark theme. At the top, there's a breadcrumb navigation bar: OOPs > areaofrect.java > Rectangle > length. Below this is the Java code for the Rectangle class and its main method:

```
1  class Rectangle {
2      int length = 2;
3      int width = 4;
4
5      void area() {
6          int result = length * width;
7          System.out.println("Area: " + result);
8      }
9  }
10
11 public class areaofrect {
12     Run | Debug
13     public static void main(String[] args) {
14         Rectangle r = new Rectangle();
15         r.area();
16     }
17 }
```

Below the code editor are tabs for PROBLEMS (11), OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the command-line output of running the program:

```
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\dfdee5caleccb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'areaofrect'
Saaili's Rectangle Area: 8
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

## 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.”

```
Inheritance > J Singleinheritance.java > Singleinheritance
1  class Animal {
2      void sound() {
3          System.out.println("Animal makes a sound");
4      }
5  }
6  class Dog extends Animal {
7      void bark() {
8          System.out.println("Dog barks");
9      }
10 }
11 public class Singleinheritance {
12     Run | Debug
13     public static void main(String[] args) {
14         Dog d = new Dog();
15         d.sound();
16         d.bark();
17     }
18 }
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    ...

> < TERMINAL

```
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-preview' '-XshowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspaceStorage\dfdee5ca1ecceb2005\d5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Singleinheritance'
Animal makes a sound
Dog barks
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

## 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)

- o Same method name with different parameters.
- o Decided at compile time.

### 2. Run-Time Polymorphism (Method Overriding)

- o Child class provides its own version of a method from the parent class.
- o 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.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a breadcrumb navigation bar: J Adddiv.java > Calculator > multiply(int, int). Below this is the code editor area containing two classes:

```
1  class Calculator {  
16 }  
17 }  
18  
19 public class Adddiv {  
    Run | Debug  
20     public static void main(String[] args) {  
21         Calculator saailiCalc = new Calculator();  
22  
23         int a = 10, b = 2;  
24  
25         System.out.println("Saaili's Add: " + saailiCalc.add(a, b));  
26         System.out.println("Saaili's Subtract: " + saailiCalc.subtract(a, b));  
27         System.out.println("Saaili's Multiply: " + saailiCalc.multiply(a, b));  
28         System.out.println("Saaili's Divide: " + saailiCalc.divide(a, b));  
29     }  
30 }  
31
```

Below the code editor are tabs for PROBLEMS (5), OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the output of the Java application:

```
> ▾ TERMINAL  
ψ  
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-p  
review' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workpac  
eStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Adddiv'  
Saaili's Add: 12  
Saaili's Subtract: 8  
Saaili's Multiply: 20  
Saaili's Divide: 5  
PS C:\Users\LOQ\OneDrive\Desktop\saaili.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.

```
Polymorphism > J DemoOverrideSuper.java > {} Polymorphism
2
3  class Gadget {
4      void turnOn() {
5          System.out.println("Gadget is turning on");
6      }
7  }
8
9  class Smartphone extends Gadget {
10     void turnOn() {
11         super.turnOn();
12         System.out.println("Smartphone is turning on with fingerprint");
13     }
14 }
15
16 public class DemoOverrideSuper {
17     Run | Debug
18     public static void main(String[] args) {
19         Smartphone s = new Smartphone();
}
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-p
review' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workpac
eStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Polymorphism.Dem
oOverrideSuper'
Gadget is turning on
Smartphone is turning on with fingerprint
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

## 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 > J 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 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
ψ
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-p
review' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspac
eStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Encapsulation.Ba
nkacc'
Account Holder: Saaili
Final Balance: ?1100.0
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

## Example2:

```
Encapsulation > J 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
ψ
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '--enable-p
review' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LOQ\AppData\Roaming\Code\User\workspac
eStorage\dfdee5ca1ecceb20050b8bad5d75ef49\redhat.java\jdt_ws\saaili.java_cea4b63f\bin' 'Encapsulation.M
ain'
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%).

The screenshot shows a Java code editor with the following code:

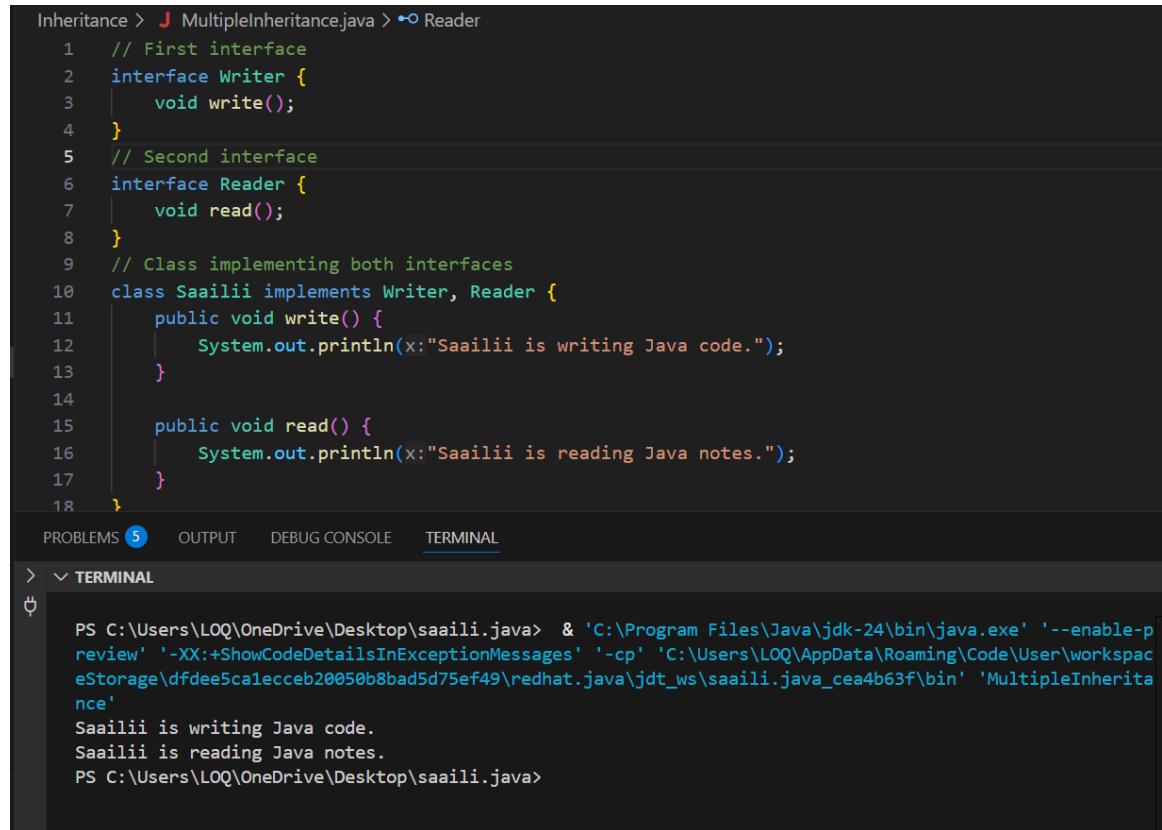
```
Abstract > J abstraction.java > abstraction
1 package Abstract;
2 abstract class ATM {
3     abstract void withdraw(double amount);
4     abstract void checkBalance();
5 }
6
7 class MyATM extends ATM {
8     double balance = 5000;
9
10    @Override
11    void withdraw(double amount) {
12        if (amount <= balance) {
13            balance -= amount;
14            System.out.println("Rs " + amount + " withdrawn");
15        } else {
16            System.out.println("Not enough balance");
17        }
18    }
}
```

Below the code editor, there is a terminal window showing the output of running the code:

```
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' 'Abstract.abstraction'
Balance: Rs 5000.0
Rs 1000.0 withdrawn
Balance: Rs 4000.0
PS C:\Users\LOQ\OneDrive\Desktop\saaili.java>
```

## 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).



```
Inheritance > J MultipleInheritance.java > Reader
1 // First interface
2 interface Writer {
3     void write();
4 }
5 // Second interface
6 interface Reader {
7     void read();
8 }
9 // Class implementing both interfaces
10 class Saailii implements Writer, Reader {
11     public void write() {
12         System.out.println("Saailii is writing Java code.");
13     }
14
15     public void read() {
16         System.out.println("Saailii is reading Java notes.");
17     }
18 }
```

PROBLEMS 5    OUTPUT    DEBUG CONSOLE    TERMINAL

> < TERMINAL

```
PS C:\Users\LOQ\OneDrive\Desktop\saailii.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\saailii.java_cea4b63f\bin' 'MultipleInheritance'
Saailii is writing Java code.
Saailii is reading Java notes.
PS C:\Users\LOQ\OneDrive\Desktop\saailii.java>
```

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

**ANS:**

The screenshot shows a Java code editor with multiple tabs at the top. The active tab is "InterfaceExample.java". The code itself is as follows:

```
1 interface > J InterfaceExample.java ...
2 > interface Animal { ...
3
4     class Dog implements Animal {
5         public void makeSound() {
6             System.out.println("Dog barks");
7         }
8     }
9
10    class Cat implements Animal {
11        public void makeSound() {
12            System.out.println("Cat meows");
13        }
14    }
15
16    public class InterfaceExample {
17        public static void main(String[] args) {
18            Animal dog = new Dog();
19            Animal cat = new Cat();
20
21            dog.makeSound(); // Output: Dog barks
22            cat.makeSound(); // Output: Cat meows
23        }
24    }
25
26 }
27
```