JAVA Assignment

By Gadhiya Chetan Bhaveshbhai 23BCP182 ---------- Prof. Dr. Nandini Modi

---

Practical-1_1.java:

Program :

```java
// 23BCP182-Chetan B.Gadhiya
class Lab1_1
{
    public static void main(String []args){
        System.out.println("WelCome TO JAVA!");
    }
}
```

Output :

```
WelCome TO JAVA!

=== Code Execution Successful ===
```

Practical-1_2.c :

Program :

```java
// 23BCP182-Chetan B.Gadhiya
class Lab1_2{
    public static void main(String[]args){
        int a=4;
        int b=5;
        int sum=a+b;
        System.out.println("Sum of a and b is : " + sum);

    }
}
```

```
Sum of a and b is : 9

=== Code Execution Successful ===
```

Practical-2_1.java :

```java
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
```

```
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-2_2.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
```

```
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-2_3.java :

```java
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
```

```
    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-2_4.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}
```

```java
// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-2_5.java :

```java
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
```

```java
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-2_6.java :

```java
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
```

```
    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
```

```
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-2_7.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
```

```
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-3_1.java :

```
// Interface Shape
interface Shape {
    double getArea();
}
```

```java
// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-3_2.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }
```

```
    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-3_3.java :

```
// Interface Shape
interface Shape {
    double getArea();
}
```

```java
// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-3_3.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }
```

```
    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-3_4.java :

```
// Interface Shape
interface Shape {
    double getArea();
}
```

```java
// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-3_6.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }
```

```
    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

Practical-4_1.java :

```
// Interface Shape
interface Shape {
```

```
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-4_2.java :

```java
// Final class (cannot be extended)
final class FinalClass {
    // Final method (cannot be overridden)
    final void display() {
        System.out.println("This is a final method.");
    }
}

// Attempting to extend FinalClass will result in a compilation
error
// class ExtendedClass extends FinalClass { }

// Using final with variables
public class FinalKeywordDemo {
    public static void main(String[] args) {
        final int x = 10; // Final variable
        // x = 20; // This will cause a compilation error

        // Final class instance
        FinalClass obj = new FinalClass();
        obj.display();
```

```
    }
}
```

```
This is a final method.

=== Code Execution Successful ===
```

Practical-4_3.java :

```java
// Parent class
class Bank {
    double getInterestRate() {
        return 0.0;
    }
}

// Child class 1
class SBI extends Bank {
    @Override
    double getInterestRate() {
        return 4.0; // SBI interest rate
    }
}

// Child class 2
class ICICI extends Bank {
    @Override
    double getInterestRate() {
        return 3.5; // ICICI interest rate
    }
}

// Child class 3
class HDFC extends Bank {
    @Override
    double getInterestRate() {
        return 4.5; // HDFC interest rate
    }
}

// Main class to test method overriding
public class MethodOverridingDemo {
    public static void main(String[] args) {
```

```
        Bank bank;

        bank = new SBI();
        System.out.println("SBI Interest Rate: " +
bank.getInterestRate());

        bank = new ICICI();
        System.out.println("ICICI Interest Rate: " +
bank.getInterestRate());

        bank = new HDFC();
        System.out.println("HDFC Interest Rate: " +
bank.getInterestRate());
    }
}
```

```
SBI Interest Rate: 4.0
ICICI Interest Rate: 3.5
HDFC Interest Rate: 4.5

=== Code Execution Successful ===
```

Practical-4_4.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}
```

```java
// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-5_1.java :

```java
// Parent class
class Parent {
    int x;

    Parent(int x) {
        this.x = x;
    }

    void display() {
        System.out.println("Parent x: " + x);
    }
}

// Child class
class Child extends Parent {
    int x;

    Child(int x, int y) {
        super(x); // Call Parent class constructor
        this.x = y; // Set Child class x
    }

    void display() {
        super.display(); // Call Parent class method
        System.out.println("Child x: " + x);
    }
}

// Main class to test 'super' and 'this'
public class SuperThisDemo {
    public static void main(String[] args) {
        Child child = new Child(10, 20);
        child.display();
    }}
```

```
Parent x: 10
Child x: 20

=== Code Execution Successful ===
```

Practical-5_2.java :

```java
public class StringRotationChecker {

    // Method to check if str2 is a rotation of str1
    public static boolean isRotation(String str1, String str2) {
        // Check if lengths are the same and neither string is
empty
        if (str1.length() != str2.length() || str1.isEmpty()) {
            return false;
        }

        // Concatenate str1 with itself
        String concatenated = str1 + str1;

        // Check if str2 is a substring of the concatenated string
        return concatenated.contains(str2);
    }

    public static void main(String[] args) {
        String str1 = "tip";
        String str2 = "pit";

        if (isRotation(str1, str2)) {
            System.out.println(str2 + " is not a rotation of " +
str1);
        } else {
            System.out.println(str2 + " is a rotation of " +
str1);
        }
    }
}
```

```
pit is a rotation of tip

=== Code Execution Successful ===
```

Practical-6_1.java :

```java
// Abstract class Shape
abstract class Shape {
    // Abstract method to calculate area
    abstract double area();
}

// Subclass Triangle
class Triangle extends Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement area() for Triangle
    double area() {
        return 0.5 * base * height;
    }
}

// Subclass Rectangle
class Rectangle extends Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement area() for Rectangle
    double area() {
        return length * width;
    }
}

// Subclass Circle
class Circle extends Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }
```

```
    // Implement area() for Circle
    double area() {
        return Math.PI * radius * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape triangle = new Triangle(10, 5);
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);

        System.out.println("Area of Triangle: " +
triangle.area());
        System.out.println("Area of Rectangle: " +
rectangle.area());
        System.out.println("Area of Circle: " + circle.area());
    }
}
```

```
Area of Triangle: 25.0
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985

=== Code Execution Successful ===
```

Practical-6_2.java :

```
// Abstract class Employee
abstract class Employee {
    String name;
    int id;

    Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    // Abstract methods
    abstract double calculateSalary();
    abstract void displayInfo();
}

// Subclass Manager
```

```java
class Manager extends Employee {
    double basicSalary;

    Manager(String name, int id, double basicSalary) {
        super(name, id);
        this.basicSalary = basicSalary;
    }

    // Implement calculateSalary() for Manager
    double calculateSalary() {
        return basicSalary + 0.1 * basicSalary; // Example bonus
    }

    // Implement displayInfo()
    void displayInfo() {
        System.out.println("Manager Name: " + name + ", ID: " + id
+ ", Salary: " + calculateSalary());
    }
}

// Subclass Programmer
class Programmer extends Employee {
    double basicSalary;

    Programmer(String name, int id, double basicSalary) {
        super(name, id);
        this.basicSalary = basicSalary;
    }

    // Implement calculateSalary() for Programmer
    double calculateSalary() {
        return basicSalary + 0.05 * basicSalary; // Example bonus
    }

    // Implement displayInfo()
    void displayInfo() {
        System.out.println("Programmer Name: " + name + ", ID: " +
id + ", Salary: " + calculateSalary());
    }
}

public class Main {
    public static void main(String[] args) {
        Employee manager = new Manager("Alice", 101, 50000);
        Employee programmer = new Programmer("Bob", 102, 40000);
```

```
        manager.displayInfo();
        programmer.displayInfo();
    }
}
```

```
Manager Name: Alice, ID: 101, Salary: 55000.0
Programmer Name: Bob, ID: 102, Salary: 42000.0

=== Code Execution Successful ===
```

Practical-6_3.java :

```
// Interface Shape
interface Shape {
    double getArea();
}

// Implementing Rectangle
class Rectangle implements Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Implement getArea() for Rectangle
    public double getArea() {
        return length * width;
    }
}

// Implementing Circle
class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    // Implement getArea() for Circle
    public double getArea() {
        return Math.PI * radius * radius;
    }
```

```
}

// Implementing Triangle
class Triangle implements Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement getArea() for Triangle
    public double getArea() {
        return 0.5 * base * height;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        Shape circle = new Circle(7);
        Shape triangle = new Triangle(10, 5);

        System.out.println("Area of Rectangle: " +
rectangle.getArea());
        System.out.println("Area of Circle: " + circle.getArea());
        System.out.println("Area of Triangle: " +
triangle.getArea());
    }
}
```

```
Area of Rectangle: 50.0
Area of Circle: 153.93804002589985
Area of Triangle: 25.0

=== Code Execution Successful ===
```

Practical-7_1.java :

```
public class TryCatchExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0; // This will cause
ArithmeticException
```

```
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: Division by zero
is not allowed.");
        }
        System.out.println("Program continues after exception
handling.");
    }
}
```

```
Exception caught: Division by zero is not allowed.
Program continues after exception handling.

=== Code Execution Successful ===
```

Practical-7_2.java :

```
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            System.out.println(numbers[5]); //
ArrayIndexOutOfBoundsException
            int result = 10 / 0; // ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException caught: " +
e.getMessage());
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException
caught: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("General Exception caught: " +
e.getMessage());
        }
    }
}
```

```
ArrayIndexOutOfBoundsException caught: Index 5 out of bounds for length 3

=== Code Execution Successful ===
```

Practical-7_3.java :

```
public class NestedTryExample {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            try {
```

```
                System.out.println(numbers[5]); //
ArrayIndexOutOfBoundsException
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Inner catch: Array index out
of bounds.");
            }
            int result = 10 / 0; // ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Outer catch: Division by zero.");
        }
    }
}
```

```
Inner catch: Array index out of bounds.
Outer catch: Division by zero.

=== Code Execution Successful ===
```

Practical-7_4.java :

```
class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}

public class BankingApplication {
    private double balance;

    public BankingApplication(double initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: Rs " + amount + ", Current
Balance: Rs " + balance);
    }

    public void withdraw(double amount) throws
InsufficientFundsException {
        if (amount > balance) {
            throw new InsufficientFundsException("Not Sufficient
Fund: Current Balance is Rs " + balance);
        }
        balance -= amount;
```

```
        System.out.println("Withdrawn: Rs " + amount + ",
Remaining Balance: Rs " + balance);
    }

    public static void main(String[] args) {
        BankingApplication account = new
BankingApplication(1000.00);

        try {
            account.withdraw(400.00);
            account.withdraw(300.00);
            account.withdraw(500.00); // Will throw exception
        } catch (InsufficientFundsException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

Practical-7_5.java :

```
public class FinallyBlockExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0; // This will cause
ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: " +
e.getMessage());
        } finally {
            System.out.println("Finally block always executes,
whether exception occurs or not.");
        }
        System.out.println("Program continues after try-catch-
finally.");
    }
}
```

```
Exception caught: / by zero
Finally block always executes, whether exception occurs or not.
Program continues after try-catch-finally.

=== Code Execution Successful ===
```

Practical-8_1.java :

```java
import java.io.*;
import java.util.Scanner;

public class FileStats {
    public static void main(String[] args) throws IOException {
        File file = new File("input.txt");
        Scanner scanner = new Scanner(file);

        int sentenceCount = 0, wordCount = 0, charCount = 0;

        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            charCount += line.length();
            wordCount += line.split("\\s+").length;
            sentenceCount += line.split("[.!?]").length;
        }
        scanner.close();
        System.out.println("Sentences: " + sentenceCount);
        System.out.println("Words: " + wordCount);
        System.out.println("Characters: " + charCount);
    }
}
```

Practical-8_2.java :

```java
import java.io.*;

public class ConvertToUpper {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new
FileReader("input.txt"));
        BufferedWriter writer = new BufferedWriter(new
FileWriter("output.txt"));

        String line;
        while ((line = reader.readLine()) != null) {
            writer.write(line.toUpperCase());
            writer.newLine();
        }
        reader.close();
        writer.close();
        System.out.println("File converted to uppercase and saved
to output.txt");
    }
}
```

Practical-8_3.java :

```
import java.io.*;
import java.util.HashSet;

public class RemoveDuplicateLines {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new
FileReader("input.txt"));
        BufferedWriter writer = new BufferedWriter(new
FileWriter("unique_output.txt"));
        HashSet<String> lines = new HashSet<>();
        String line;
        while ((line = reader.readLine()) != null) {
            if (lines.add(line)) { // Only add unique lines
                writer.write(line);
                writer.newLine();
            }
        }
        reader.close();
        writer.close();
        System.out.println("Duplicate lines removed and saved to
unique_output.txt");
    }
}
```

Practical-8_4.java :

```
import java.io.*;

class Student implements Serializable {
    String name;
    int id;

    public Student(String name, int id) {
        this.name = name;
        this.id = id;
    }
    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name;
    }
}
```

```java
public class StudentManager {
    public static void main(String[] args) throws IOException,
ClassNotFoundException {
        File file = new File("students.dat");

        // Add student
        Student student = new Student("John Doe", 101);
        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(student);
        oos.close();

        // Read student
        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Student readStudent = (Student) ois.readObject();
        ois.close();

        System.out.println("Read Student: " + readStudent);
    }
}
```

Practical-8_5.java :

```java
import java.io.*;

class Student {
    String name;
    int id;

    public Student(String name, int id) {
        this.name = name;
        this.id = id;
    }

    @Override
    public String toString() {
        return id + "," + name;
    }

    public static Student fromString(String data) {
        String[] parts = data.split(",");
        return new Student(parts[1], Integer.parseInt(parts[0]));
    }
```

```
}

public class StudentManagerBuffered {
    public static void main(String[] args) throws IOException {
        File file = new File("students.txt");

        // Write student
        Student student = new Student("Alice", 102);
        BufferedWriter writer = new BufferedWriter(new
FileWriter(file));
        writer.write(student.toString());
        writer.newLine();
        writer.close();

        // Read student
        BufferedReader reader = new BufferedReader(new
FileReader(file));
        String data = reader.readLine();
        reader.close();

        Student readStudent = Student.fromString(data);
        System.out.println("Read Student: " + readStudent);
    }
}
```

Practical-8_6.java :

```
import java.io.*;

public class FileManipulation {
    public static void main(String[] args) throws IOException {
        File inputFile = new File("input.txt");
        File outputFile = new File("output_processed.txt");

        FileReader reader = new FileReader(inputFile);
        FileWriter writer = new FileWriter(outputFile);

        int character;
        while ((character = reader.read()) != -1) {
            // Example transformation: Convert to lowercase
            writer.write(Character.toLowerCase(character));
        }

        reader.close();
        writer.close();
```

```
        System.out.println("File processed and saved to
output_processed.txt");
    }
}
```

Practical-9_1.java :

```
class MyThread extends Thread {
    @Override
    public void run() {
        System.out.println("Thread using Thread class is running:
" + Thread.currentThread().getName());
    }
}

class MyRunnable implements Runnable {
    @Override
    public void run() {
        System.out.println("Thread using Runnable interface is
running: " + Thread.currentThread().getName());
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread();
        thread1.start();

        Thread thread2 = new Thread(new MyRunnable());
        thread2.start();
    }
}
```

```
Thread using Thread class is running: Thread-0
Thread using Runnable interface is running: Thread-1

=== Code Execution Successful ===
```

Practical-9_2.java :

```
class SharedResource {
    private int counter = 0;

    public synchronized void increment() {
```

```
        counter++;
        System.out.println(Thread.currentThread().getName() + "
incremented counter to: " + counter);
    }
}

public class ThreadSynchronization {
    public static void main(String[] args) {
        SharedResource resource = new SharedResource();

        Runnable task = () -> {
            for (int i = 0; i < 5; i++) {
                resource.increment();
            }
        };

        Thread thread1 = new Thread(task, "Thread-1");
        Thread thread2 = new Thread(task, "Thread-2");

        thread1.start();
        thread2.start();
    }
}
```

```
Thread-1 incremented counter to: 1
Thread-1 incremented counter to: 2
Thread-1 incremented counter to: 3
Thread-1 incremented counter to: 4
Thread-1 incremented counter to: 5
Thread-2 incremented counter to: 6
Thread-2 incremented counter to: 7
Thread-2 incremented counter to: 8
Thread-2 incremented counter to: 9
Thread-2 incremented counter to: 10

=== Code Execution Successful ===
```

Practical-9_3.java :

```
class SharedQueue {
    private int value;
    private boolean isAvailable = false;
```

```
    public synchronized void produce(int val) {
        while (isAvailable) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        value = val;
        isAvailable = true;
        System.out.println("Produced: " + value);
        notify();
    }

    public synchronized void consume() {
        while (!isAvailable) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        System.out.println("Consumed: " + value);
        isAvailable = false;
        notify();
    }
}

public class InterThreadCommunication {
    public static void main(String[] args) {
        SharedQueue queue = new SharedQueue();

        Thread producer = new Thread(() -> {
            for (int i = 1; i <= 5; i++) {
                queue.produce(i);
            }
        });

        Thread consumer = new Thread(() -> {
            for (int i = 1; i <= 5; i++) {
                queue.consume();
            }
        });

        producer.start();
        consumer.start();
```

```
    }
}
```

```
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5

=== Code Execution Successful ===
```

Practical-9_4.java :

```
public class ThreadPriorityDemo {
    public static void main(String[] args) {
        Thread highPriority = new Thread(() -> {
            System.out.println(Thread.currentThread().getName() +
" is running with priority: " +
Thread.currentThread().getPriority());
        }, "HighPriorityThread");

        Thread lowPriority = new Thread(() -> {
            System.out.println(Thread.currentThread().getName() +
" is running with priority: " +
Thread.currentThread().getPriority());
        }, "LowPriorityThread");

        highPriority.setPriority(Thread.MAX_PRIORITY);
        lowPriority.setPriority(Thread.MIN_PRIORITY);

        lowPriority.start();
        highPriority.start();
    }
}
```

```
LowPriorityThread is running with priority: 1
HighPriorityThread is running with priority: 10

=== Code Execution Successful ===
```

Practical-9_5.java :

```java
import java.util.LinkedList;
import java.util.Queue;

class ProducerConsumer {
    private final Queue<Integer> queue = new LinkedList<>();
    private final int capacity;

    public ProducerConsumer(int capacity) {
        this.capacity = capacity;
    }

    public synchronized void produce(int value) throws
InterruptedException {
        while (queue.size() == capacity) {
            wait();
        }
        queue.add(value);
        System.out.println("Produced: " + value);
        notifyAll();
    }

    public synchronized void consume() throws InterruptedException
{
        while (queue.isEmpty()) {
            wait();
        }
        int value = queue.poll();
        System.out.println("Consumed: " + value);
        notifyAll();
    }
}

public class ProducerConsumerDemo {
    public static void main(String[] args) {
        ProducerConsumer pc = new ProducerConsumer(5);

        Thread producer = new Thread(() -> {
            for (int i = 1; i <= 10; i++) {
```

```
                try {
                    pc.produce(i);
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }
        }, "Producer");

        Thread consumer = new Thread(() -> {
            for (int i = 1; i <= 10; i++) {
                try {
                    pc.consume();
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }
        }, "Consumer");

        producer.start();
        consumer.start();
    }
}
```

```
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Produced: 5
Consumed: 1
Consumed: 2
Consumed: 3
Consumed: 4
Consumed: 5
Produced: 6
Produced: 7
Produced: 8
Produced: 9
Produced: 10
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Consumed: 10

=== Code Execution Successful ===
```

Practical-10_1.java :

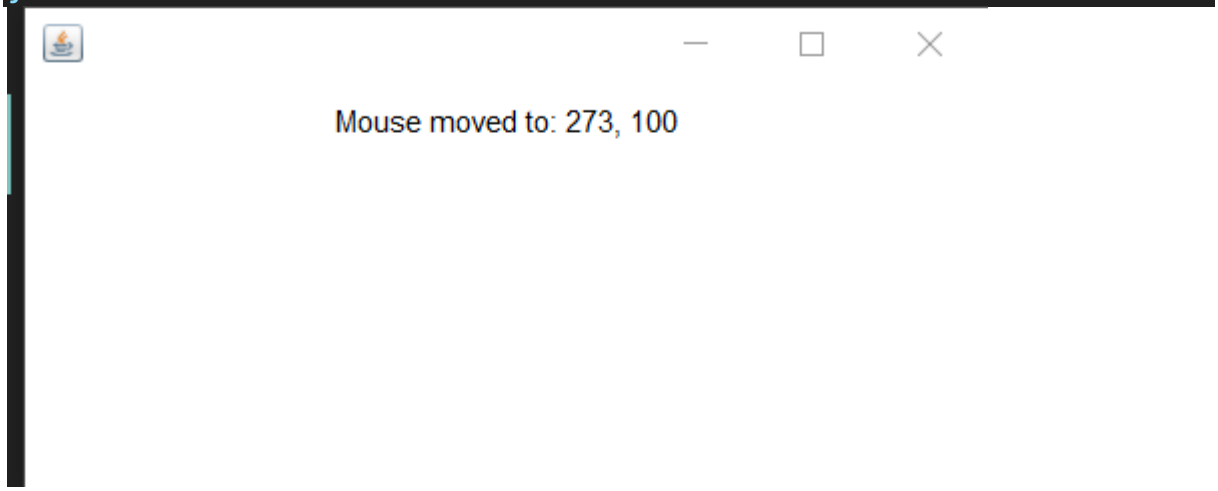
```java
import java.awt.*;
import java.awt.event.*;
public class WindowEventDemo extends Frame implements
WindowListener {
    public WindowEventDemo() {
        addWindowListener(this);
        setTitle("Window Event Demo");
        setSize(400, 200);
        setVisible(true);
    }

    @Override
    public void windowOpened(WindowEvent e) {
```

```
        System.out.println("Window opened");
    }

    @Override
    public void windowClosing(WindowEvent e) {
        System.out.println("Window closing");
        dispose();
    }

    @Override
    public void windowClosed(WindowEvent e) {
        System.out.println("Window closed");
    }

    @Override
    public void windowIconified(WindowEvent e) {
        System.out.println("Window minimized");
    }

    @Override
    public void windowDeiconified(WindowEvent e) {
        System.out.println("Window restored");
    }

    @Override
    public void windowActivated(WindowEvent e) {
        System.out.println("Window activated");
    }

    @Override
    public void windowDeactivated(WindowEvent e) {
        System.out.println("Window deactivated");
    }

    public static void main(String[] args) {
        new WindowEventDemo();
    }
}
```

Practical-10_2.java :

```
import java.awt.*;
import java.awt.event.*;

public class MouseEventDemo extends Frame implements
MouseListener, MouseMotionListener {
    Label label;

    public MouseEventDemo() {
        label = new Label("Perform mouse actions!");
        add(label);
        setLayout(new FlowLayout());
        setSize(400, 200);
        addMouseListener(this);
        addMouseMotionListener(this);
        setVisible(true);
    }

    @Override
    public void mouseClicked(MouseEvent e) {
        label.setText("Mouse clicked at: " + e.getX() + ", " +
e.getY());
    }

    @Override
    public void mouseEntered(MouseEvent e) {
        label.setText("Mouse entered");
    }

    @Override
```

```
    public void mouseExited(MouseEvent e) {
        label.setText("Mouse exited");
    }

    @Override
    public void mousePressed(MouseEvent e) {
        label.setText("Mouse pressed");
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        label.setText("Mouse released");
    }

    @Override
    public void mouseDragged(MouseEvent e) {
        label.setText("Mouse dragged to: " + e.getX() + ", " +
e.getY());
    }

    @Override
    public void mouseMoved(MouseEvent e) {
        label.setText("Mouse moved to: " + e.getX() + ", " +
e.getY());
    }

    public static void main(String[] args) {
        new MouseEventDemo();
    }
}
```



Practical-10_3.java :

```
import java.awt.*;
```

```java
import java.awt.event.*;

public class KeyEventDemo extends Frame implements KeyListener {
    Label label;

    public KeyEventDemo() {
        label = new Label("Press any key");
        add(label);
        addKeyListener(this);
        setSize(400, 200);
        setVisible(true);
    }

    @Override
    public void keyPressed(KeyEvent e) {
        label.setText("Key pressed: " + e.getKeyChar());
    }

    @Override
    public void keyReleased(KeyEvent e) {
        label.setText("Key released: " + e.getKeyChar());
    }

    @Override
    public void keyTyped(KeyEvent e) {
        label.setText("Key typed: " + e.getKeyChar());
    }

    public static void main(String[] args) {
        new KeyEventDemo();
    }
}
```

Key released: s

Practical-10_4.java :

```java
import javax.swing.*;
import java.awt.event.*;

public class SimpleGUIDemo {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Button and Label Demo");
        JLabel label = new JLabel("Click the button!");
        JButton button = new JButton("Click Me");

        label.setBounds(100, 50, 200, 30);
        button.setBounds(100, 100, 150, 30);

        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                label.setText("Button clicked!");
            }
        });

        frame.add(label);
        frame.add(button);
        frame.setLayout(null);
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```