AVERAGE LEARNER ASSIGNMENT

Student Name: Rohan Kumar Shah **UID: 22BCS11018**
Branch: CSE **Section: IOT-642 -B**
Semester: 6th **DOP: 10/04/2025**
Subject: Java **Subject Code:22CSH-359**

PROBLEM:1

Aim: Develop a Java program showcasing the concept of inheritance. Create a base class and a derived class with appropriate methods and fields.

CODE:

```
class Animal {
    String name;

    void eat() {
        System.out.println(name + " is eating.");
    }

    void sleep() {
        System.out.println(name + " is sleeping.");
    }
}
class Dog extends Animal {
    void bark() {
        System.out.println(name + " is barking.");
    }
}
public class InheritanceExample {
    public static void main(String[] args) {
        Dog myDog = new Dog();       // Create object of Dog
        myDog.name = "Buddy";        // Set name

        myDog.eat();                 // Call inherited method
        myDog.sleep();               // Call inherited method
        myDog.bark();                // Call Dog's own method
    }
}
```

OUTPUT:

PROBLEM:2

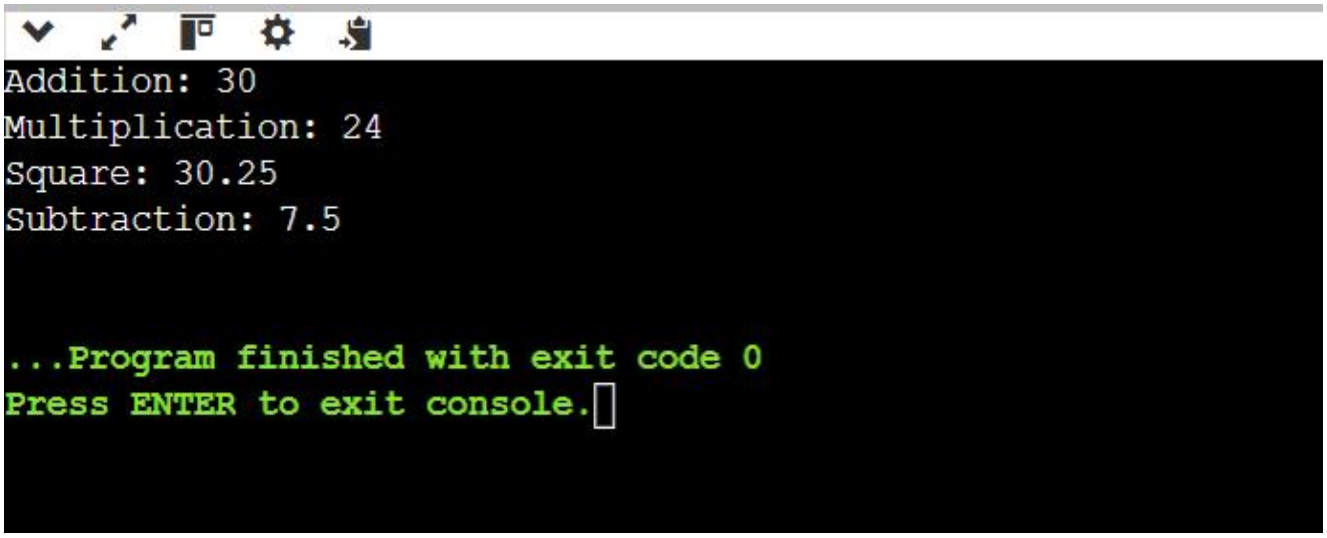
Aim: Implement a Java program that uses method overloading to perform different mathematical operations.

CODE:

```
class Calculator {
    int calculate(int a, int b) {
        return a + b;
    }

    int calculate(int a, int b, int c) {
        return a * b * c;
    }

    double calculate(double a) {
        return a * a;
    }

    double calculate(double a, double b, boolean subtract) {
        if (subtract) {
            return a - b;
        }
        return 0.0;
    }
}
public class MethodOverloadingExample {
    public static void main(String[] args) {
```

```
        Calculator calc = new Calculator();

        System.out.println("Addition: " + calc.calculate(10, 20));
        System.out.println("Multiplication: " + calc.calculate(2, 3, 4));
        System.out.println("Square: " + calc.calculate(5.5));
        System.out.println("Subtraction: " + calc.calculate(10.5, 3.0, true));
    }
}
```

OUTPUT:



PROBLEM:3

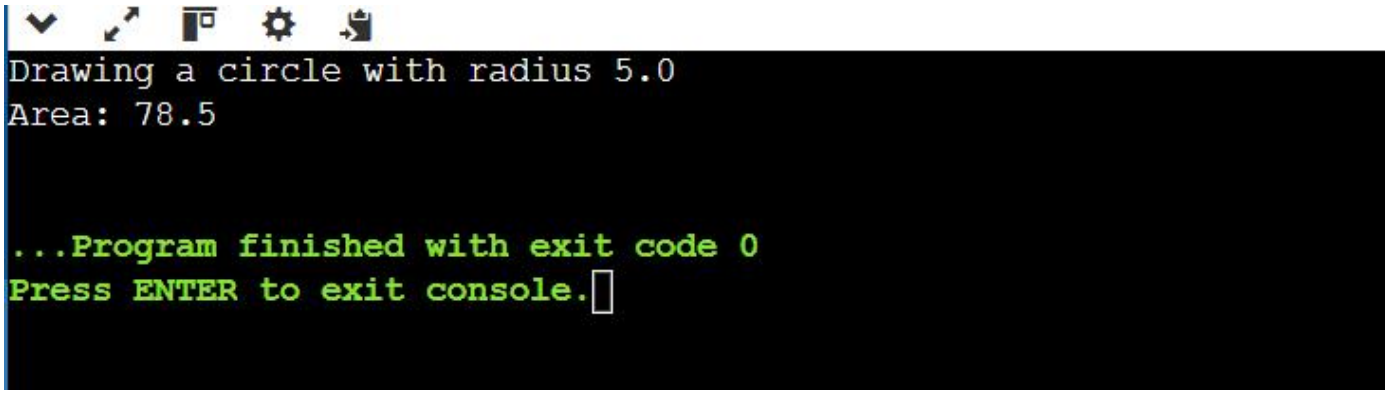
Aim: Define an interface in Java and create a class that implements it, demonstrating the concept of abstraction.

CODE:

```
interface Shape {
    void draw();
    double getArea();
}
class Circle implements Shape {
    double radius;

    Circle(double r) {
        radius = r;
    }
```

```
    public void draw() {
        System.out.println("Drawing a circle with radius " + radius);
    }
    public double getArea() {
        return 3.14 * radius * radius;
    }
}
public class InterfaceExample {
    public static void main(String[] args) {
        Shape s = new Circle(5.0);

        s.draw();
        System.out.println("Area: " + s.getArea());
    }
}
```

OUTPUT:



PROBLEM:4

Aim: Create a custom exception class in Java. Write a program that throws this custom exception in a specific scenario.

CODE:

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

```
public class CustomExceptionExample {
    static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age is below 18. Not allowed!");
        } else {
            System.out.println("Age is valid. Access granted.");
        }
    }

    public static void main(String[] args) {
        try {
            checkAge(16);  // Change the age to test different cases
        } catch (InvalidAgeException e) {
            System.out.println("Caught Exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Caught Exception: Age is below 18. Not allowed!


...Program finished with exit code 0
Press ENTER to exit console.
```

PROBLEM:5

5. Explain the difference between the throw and throws keywords in Java. Provide examples illustrating their usage.

Throw	Throws
1. Used to **actually throw** an exception. 2. It is used to inside a method/block. 3. Eg. throw new Exception("Message");	1. Used to **declare** that a method might throw an exception 2. It is used to method declaration/signature. 3. Eg. void method() throws Exception

Using Throw

```
public class ThrowExample {
    public static void main(String[] args) {
        int age = 15;

        if (age < 18) {
            throw new ArithmeticException("Access denied - You must be 18 or older.");
        }

        System.out.println("Access granted.");
    }
}
```

Using Throws

```
public class ThrowsExample {

    static void checkNumber(int num) throws Exception {
        if (num < 0) {
            throw new Exception("Negative numbers not allowed.");
```

```
        }
        System.out.println("Number is valid: " + num);
    }

    public static void main(String[] args) {
        try {
            checkNumber(-5);  // Will throw exception
        } catch (Exception e) {
            System.out.println("Caught Exception: " + e.getMessage());
        }
```