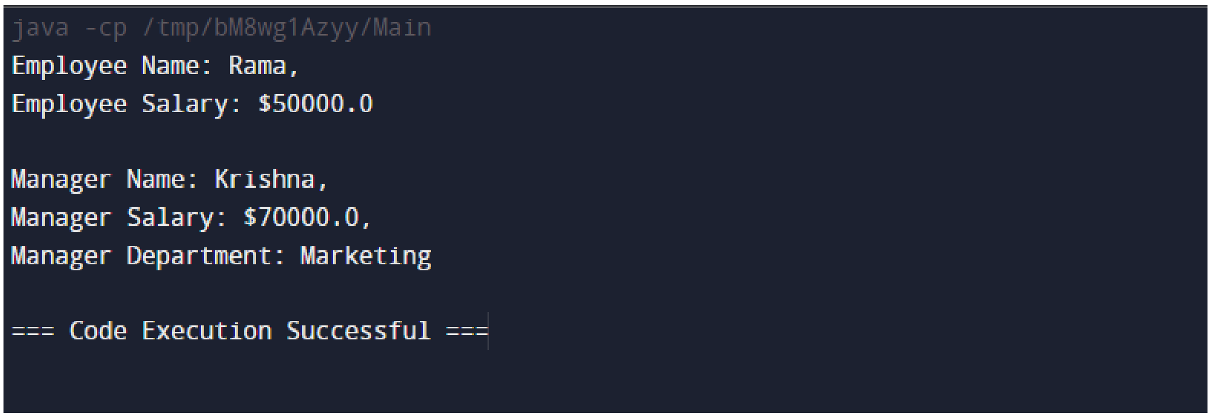
1. **Create a class Employee with properties name and salary, and a constructor to initialize these properties. Create a subclass Manager that adds a property department and a constructor to initialize all properties. Demonstrate creating instances of both classes.**

Program:

```
class Employee {
    String name;
    double salary;
    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
}
class Manager extends Employee {
    String department;
    public Manager(String name, double salary, String department) {
        super(name, salary);
        this.department = department;
    }
}
public class Main {
    public static void main(String[] args) {
        Employee emp = new Employee("Rama", 50000.0);
        Manager manager = new Manager("Krishna", 70000.0, "Marketing");
        System.out.println("Employee Name: " + emp.name + ",\nEmployee 
Salary: $" + emp.salary);
        System.out.println("\nManager Name: " + manager.name + ", 
\nManager Salary: $" + manager.salary + ", \nManager Department: " + 
manager.department);
```

```
    }
}
```

Output:



```
java -cp /tmp/bM8wg1Azyy/Main
Employee Name: Rama,
Employee Salary: $50000.0

Manager Name: Krishna,
Manager Salary: $70000.0,
Manager Department: Marketing

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

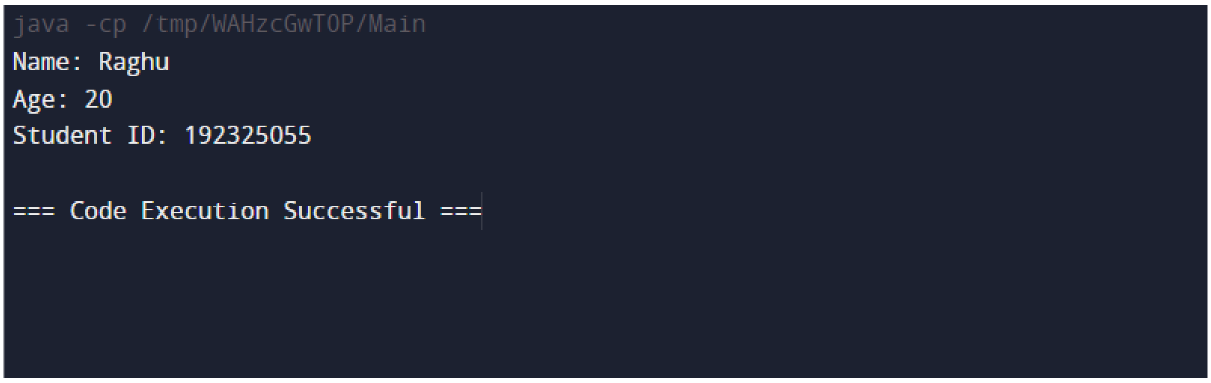
2. **Create a superclass Person with properties name and age, and a method displayInfo(). Create a subclass Student that adds a property studentId and overrides the displayInfo() method. Use the super keyword to call the superclass method.**

Program:

```
class Person {
    String name;
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

```
class Student extends Person {
    int studentId;
    public Student(String name, int age, int studentId) {
        super(name, age);
        this.studentId = studentId;
    }
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Student ID: " + studentId);
    }
}
public class Main {
    public static void main(String[] args) {
        Student student = new Student("Raghu", 20, 192325055);
        student.displayInfo();
    }
}
```

Output:



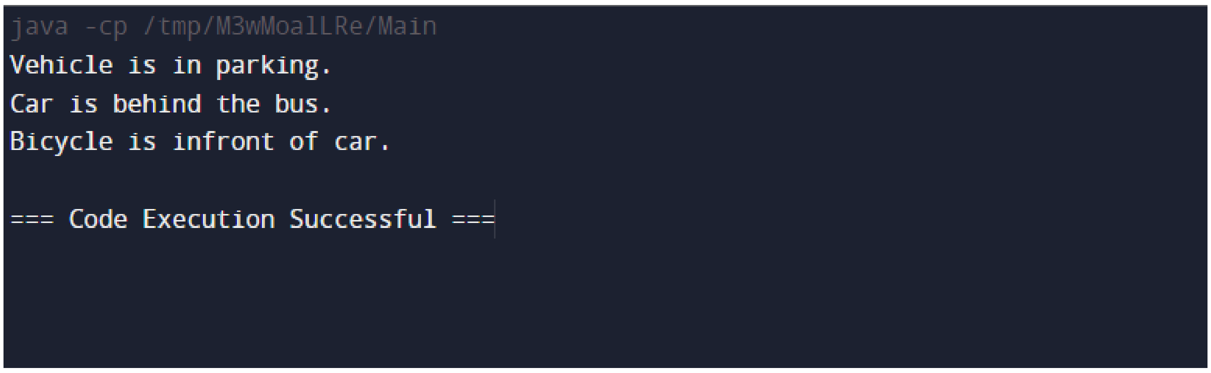
```
java -cp /tmp/WAHzcGwTOP/Main
Name: Raghu
Age: 20
Student ID: 192325055

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

3. **Create a class Vehicle with a method move(). Create subclasses Car and Bicycle, each overriding the move() method to provide specific implementations. Demonstrate the use of overridden methods.**

Program:

```
class Vehicle {
    public void move() {
        System.out.println("Vehicle is in parking.");
    }
}
class Car extends Vehicle {
    public void move() {
        System.out.println("Car is behind the bus.");
    }
}
class Bicycle extends Vehicle {
    public void move() {
        System.out.println("Bicycle is infront of car.");
    }
}
public class Main {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle();
        Car car = new Car();
        Bicycle bicycle = new Bicycle();
        vehicle.move();
        car.move();
        bicycle.move();
    }
}
```

Output:



```
java -cp /tmp/M3wMoalLRe/Main
Vehicle is in parking.
Car is behind the bus.
Bicycle is infront of car.

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

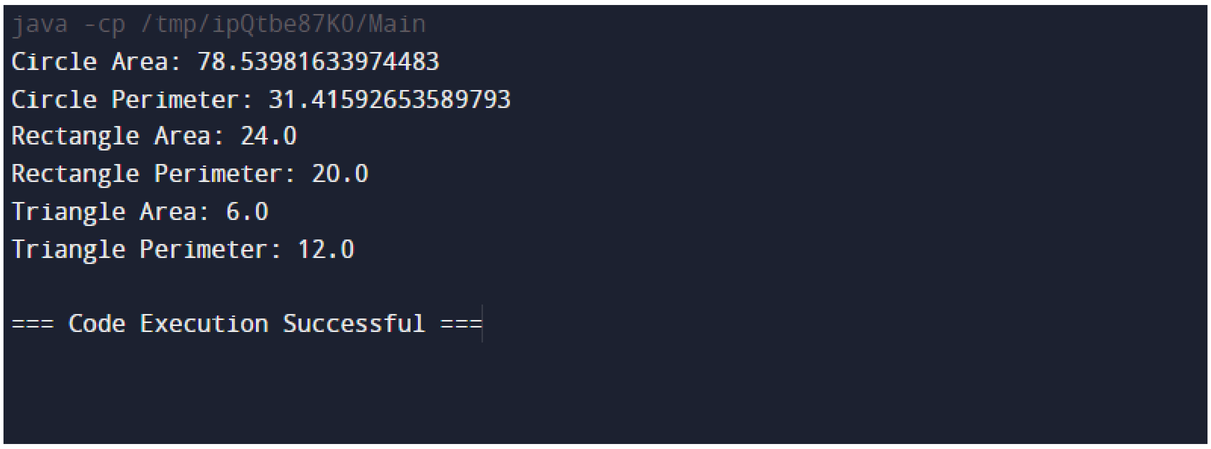
4. Design a class hierarchy for shapes in Java. Include an abstract class Shape with methods calculateArea() and calculatePerimeter(). Implement subclasses such as Circle, Rectangle, and Triangle that extend Shape and provide specific implementations for area and perimeter calculations.

Program:

```
abstract class Shape {
    public abstract double calculateArea();
    public abstract double calculatePerimeter();
}
class Circle extends Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}
```

```java
class Rectangle extends Shape {
    private double length;
    private double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double calculateArea() {
        return length * width;
    }
    public double calculatePerimeter() {
        return 2 * (length + width);
    }
}
class Triangle extends Shape {
    private double side1;
    private double side2;
    private double side3;
    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }
    public double calculateArea() {
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }
    public double calculatePerimeter() {
        return side1 + side2 + side3;
    }
}
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

        Rectangle rectangle = new Rectangle(4, 6);
        System.out.println("Rectangle Area: " + rectangle.calculateArea());
        System.out.println("Rectangle Perimeter: " +
rectangle.calculatePerimeter());

        Triangle triangle = new Triangle(3, 4, 5);
        System.out.println("Triangle Area: " + triangle.calculateArea());
        System.out.println("Triangle Perimeter: " +
triangle.calculatePerimeter());
    }
}
```

Output:



```
java -cp /tmp/ipQtbe87K0/Main
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Rectangle Area: 24.0
Rectangle Perimeter: 20.0
Triangle Area: 6.0
Triangle Perimeter: 12.0

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