TEST-7

SET-1

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 classe

code:

public class Main {

public static class Employee {

private String name;

private double salary;

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

public String getName() {

return name;

}

public double getSalary() {

return salary;

}

public String toString() {

return "Employee [Name=" + name + ", Salary=" + salary + "]";

}

}

public static class Manager extends Employee {

private String department;

public Manager(String name, double salary, String department) {

super(name, salary);

this.department = department;

}

public String getDepartment() {

return department;

}

public String toString() {

return "Manager [Name=" + getName() + ", Salary=" + getSalary() + ", Department=" + department + "]";

}

}

public static void main(String[] args) {

Employee emp = new Employee(" vishnu", 50000);

System.out.println(emp);

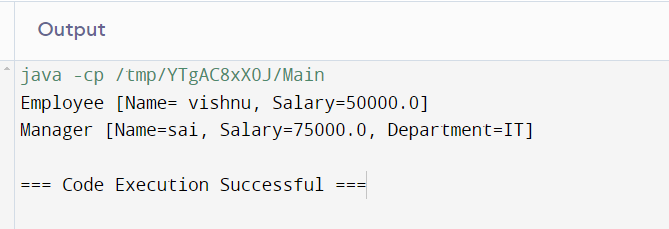
Manager mgr = new Manager("sai", 75000, "IT");

System.out.println(mgr);

}

}

Output:



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.

Code:

public class Main {

public static class person {

private String name;

private int age;

public person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public double getage() {

return age;

}

public String toString() {

return "Employee [Name=" + name + ", Salary=" + age + "]";

}

}

public static class student extends person {

private String department;

public student(String name, int age, String department) {

super(name, age);

this.department = department;

}

public String getDepartment() {

return department;

}

public String toString() {

return "Manager [Name=" + getName() + ", Salary=" + getage() + ", Department=" + department + "]";

}

}

public static void main(String[] args) {

person emp = new person(" vishnu", 15);

System.out.println(emp);

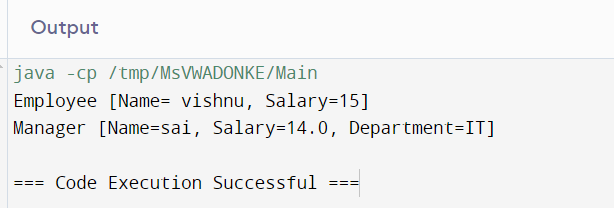
student mgr = new student("sai", 14, "IT");

System.out.println(mgr);

}

}

Ouput:



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.

Code:

public class Main {

public static class Vehicle {

public void move() {

System.out.println("The vehicle is moving");

}

}

public static class Car extends Vehicle {

public void move() {

System.out.println("The car is driving");

}

}

public static class Bicycle extends Vehicle {

public void move() {

System.out.println("The bicycle is pedaling");

}

}

public static void main(String[] args) {

Vehicle vehicle = new Vehicle();

vehicle.move();

Car car = new Car();

car.move();

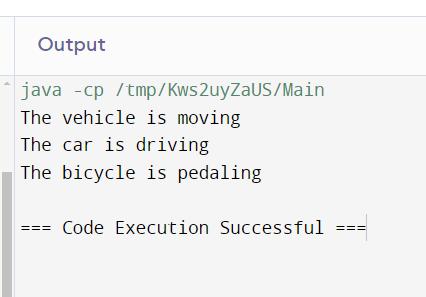
Bicycle bicycle = new Bicycle();

bicycle.move();

}

}

Output:



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.

Code:

public class Main {

public static abstract class Shape {

public abstract double calculateArea();

public abstract double calculatePerimeter();

}

public static 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;

}

}

public static 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);

}

}

public static class Triangle extends Shape {

private double sideA;

private double sideB;

private double sideC;

public Triangle(double sideA, double sideB, double sideC) {

this.sideA = sideA;

this.sideB = sideB;

this.sideC = sideC;

}

public double calculateArea() {

double s = (sideA + sideB + sideC) / 2;

return Math.sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));

}

public double calculatePerimeter() {

return sideA + sideB + sideC;

}

}

public static void main(String[] args) {

Shape circle = new Circle(5);

Shape rectangle = new Rectangle(4, 6);

Shape triangle = new Triangle(3, 4, 5);

System.out.println("Circle Area: " + circle.calculateArea());

System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

System.out.println("Rectangle Area: " + rectangle.calculateArea());

System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());

System.out.println("Triangle Area: " + triangle.calculateArea());

System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());

}

}

Output:

