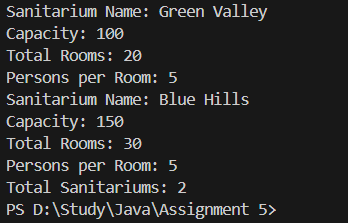
Name – Harshit Dudhani

MIS – 112316018

Java Lab

Assignment 5

1. Write program to create a class ‘Sanitarium’. Initialize its data members name,  
   capacity and total number of rooms using a constructor. Define method to find  
   number of persons can be allocated for one room. (NOTE: Make use of this and  
   static).  
     
   Code :
2. class Sanitarium {
3. private String name;
4. private int capacity;
5. private int totalRooms;
6. private static int totalSanitariums = 0;
7. public Sanitarium(String name, int capacity, int totalRooms) {
8. this.name = name;
9. this.capacity = capacity;
10. this.totalRooms = totalRooms;
11. totalSanitariums++;
12. }
13. public int personsPerRoom() {
14. if (totalRooms == 0) {
15. System.out.println("No rooms available.");
16. return 0;
17. }
18. return capacity / totalRooms;
19. }
20. public static int getTotalSanitariums() {
21. return totalSanitariums;
22. }
23. public void displayDetails() {
24. System.out.println("Sanitarium Name: " + this.name);
25. System.out.println("Capacity: " + this.capacity);
26. System.out.println("Total Rooms: " + this.totalRooms);
27. System.out.println("Persons per Room: " + this.personsPerRoom());
28. }
29. public static void main(String[] args) {
30. Sanitarium san1 = new Sanitarium("Green Valley", 100, 20);
31. Sanitarium san2 = new Sanitarium("Blue Hills", 150, 30);
32. san1.displayDetails();
33. san2.displayDetails();
34. System.out.println("Total Sanitariums: " + Sanitarium.getTotalSanitariums());
35. }
36. }

Output:  
  
   
  
  
2.    Write program to create a class ‘Salary’ which will contain data members basic  
pay, TA, DA, HRA. Make use of constructors to initialize these values for  
object. Calculate total salary of the employee using the method. (NOTE :Make  
use of this and static).  
  
Code:  
class Salary {

    private double basicPay;

    private double ta;

    private double da;

    private double hra;

    private static int employeeCount = 0;

    public Salary(double basicPay, double ta, double da, double hra) {

        this.basicPay = basicPay;

        this.ta = ta;

        this.da = da;

        this.hra = hra;

        employeeCount++;

    }

    public double calculateTotalSalary() {

        return this.basicPay + this.ta + this.da + this.hra;

    }

    public void displaySalaryDetails() {

        System.out.println("Basic Pay: " + this.basicPay);

        System.out.println("TA: " + this.ta);

        System.out.println("DA: " + this.da);

        System.out.println("HRA: " + this.hra);

        System.out.println("Total Salary: " + this.calculateTotalSalary());

    }

    public static int getEmployeeCount() {

        return employeeCount;

    }

    public static void main(String[] args) {

        Salary emp1 = new Salary(30000, 2000, 4000, 5000);

        Salary emp2 = new Salary(40000, 3000, 5000, 7000);

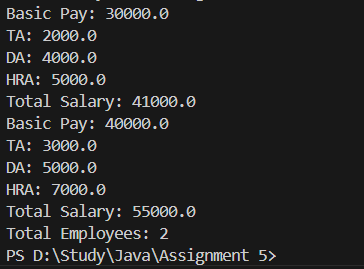
        emp1.displaySalaryDetails();

        emp2.displaySalaryDetails();

        System.out.println("Total Employees: " + Salary.getEmployeeCount());

    }

}

Output:  


   
  
  
3.    Write program to create a class ‘Institute’ having data members: total students,  
branches and percentage of total result. Input the data using static  
method. Accept this data for two objects. Create a static method  
containing two objects parameter and display the data of college whose  
result is better.  
  
Code:  
class Institute {

    private int totalStudents;

    private int branches;

    private double resultPercentage;

    public Institute(int totalStudents, int branches, double resultPercentage) {

        this.totalStudents = totalStudents;

        this.branches = branches;

        this.resultPercentage = resultPercentage;

    }

    public double getResultPercentage() {

        return this.resultPercentage;

    }

    public void displayInstituteDetails() {

        System.out.println("Total Students: " + this.totalStudents);

        System.out.println("Branches: " + this.branches);

        System.out.println("Result Percentage: " + this.resultPercentage + "%");

    }

    public static Institute betterResult(Institute inst1, Institute inst2) {

        return inst1.getResultPercentage() > inst2.getResultPercentage() ? inst1 : inst2;

    }

    public static void main(String[] args) {

        Institute inst1 = new Institute(500, 5, 89.5);

        Institute inst2 = new Institute(600, 6, 92.3);

        System.out.println("Details of Institute with Better Result:");

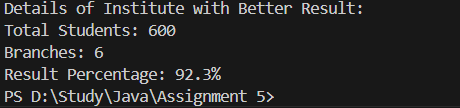
        Institute betterInstitute = Institute.betterResult(inst1, inst2);

        betterInstitute.displayInstituteDetails();

    }

}

Output:

  
   
  
  
4.    Write a program to overload the method ‘FindVolume()’ to find volume of cube, cone, cylinder and  
sphere. Initialize the objects using constructors. Make the use of this() and  
static variables.  
Code:  
public class FindVolume {

    private static final double PI = 3.14159;

    private static int objectCount = 0;

    private double volume;

    public FindVolume() {

        this.volume = 0;

        objectCount++;

    }

    public double FindVolume(double side) {

        volume = side \* side \* side;

        return volume;

    }

    public double FindVolume(double radius, double height) {

        volume = (1.0/3.0) \* PI \* radius \* radius \* height;

        return volume;

    }

    public double FindVolume(double radius, double height, String shape) {

        if (shape.equalsIgnoreCase("cylinder")) {

            volume = PI \* radius \* radius \* height;

        }

        return volume;

    }

    public double FindVolume(double radius, boolean isSphere) {

        if (isSphere) {

            volume = (4.0/3.0) \* PI \* radius \* radius \* radius;

        }

        return volume;

    }

    public static int getObjectCount() {

        return objectCount;

    }

    public static void main(String[] args) {

        FindVolume calc = new FindVolume();

        double cubeVolume = calc.FindVolume(5);

        System.out.println("Volume of cube: " + cubeVolume);

        double coneVolume = calc.FindVolume(3, 7);

        System.out.println("Volume of cone: " + coneVolume);

        double cylinderVolume = calc.FindVolume(4, 8, "cylinder");

        System.out.println("Volume of cylinder: " + cylinderVolume);

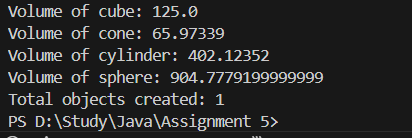
        double sphereVolume = calc.FindVolume(6, true);

        System.out.println("Volume of sphere: " + sphereVolume);

        System.out.println("Total objects created: " + FindVolume.getObjectCount());

    }

}

Output:  


5. Read the concepts of function/method overloading and operator overloading.

Method overloading is a feature that allows a class to have multiple methods with the same name but different parameters. It's a form of compile-time polymorphism (also called static polymorphism).

Methods can be overloaded in three ways:

1. By changing the number of parameters
2. By changing the data types of parameters
3. By changing the sequence/order of parameters

Important points about method overloading:

1. Return type alone is not sufficient for method overloading. The method signature must be different.
2. Method overloading is determined at compile time, not runtime.
3. Two overloaded methods cannot have the same parameter types in the same order.
4. Type promotion plays a role in overloading. If an exact match method is not found, Java automatically promotes the data type to the next higher type to find a match.
5. Both static and non-static methods can be overloaded.
6. Overloaded methods can have different access modifiers.
7. Overloaded methods can have different return types as long as the parameter list is different.

Benefits of method overloading:

* Increases program readability
* Reuses the same method name for related operations
* Reduces complexity in the code
* Eliminates the need to remember different method names

Java doesn’t support operator overloading but there are some other methods which we can use in palace of operator overloading they are:

1. Using Methods

* Instead of overloading + operator: use add(), addTo(), plus() methods
* Instead of overloading - operator: use subtract(), minus() methods
* Instead of overloading \* operator: use multiply(), times() methods

1. Using Factory Methods

* Static methods that create new objects
* Commonly used for operations that would use operators in other languages

1. Using Specialized Classes

* Java provides special classes for common operations
* StringBuilder for string concatenation
* BigInteger and BigDecimal for mathematical operations