Week 4

Programs on Classes and Objects

Write a Java program that defines a class Rectangle with attributes w (width) and h (height). Implement a member function set_values() to set the width and height, ensuring that they are nonnegative. If either value is negative, output an error message. Implement another member function area() to calculate and return the area of the rectangle. Create an object of this class, prompt the user to input the width and height of the rectangle, set the values using the set_values() function, and print its area.

PROGRAM:

```
import java.util.Scanner;
class Rectangle {
    private double width;
    private double height;

    public Rectangle() {
        this.width = 0;
        this.height = 0;
    }

    public void set_values(double w, double h) {
        if (w < 0 || h < 0) {
            System.out.println("Error: Width and height must be nonnegative.");
        } else {
            this.width = w;
            this.height = h;
        }
    }

    public double area() {
        return this.width * this.height;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Rectangle rect = new Rectangle();

        System.out.print("Enter the width of the rectangle: ");
        double width = scanner.nextDouble();

        System.out.print("Enter the height of the rectangle: ");
        double height = scanner.nextDouble();
```

```
        rect.set_values(width, height);

        if (width >= 0 && height >= 0) {
            System.out.println("The area of the rectangle is: " + rect.area());
        }

        scanner.close();
    }
}
```

OUTPUT:

```
Enter the width of the rectangle: 5
Enter the height of the rectangle: 2
The area of the rectangle is: 10.0
```

2. Create a Java program to manage information about different vehicles. Your program should:

- **Define a Vehicle class with instance variables for make, model, year, and price.**
- **Implement constructors for the Vehicle class:**
  - **A no-argument constructor that initializes default values.**
  - **A constructor that accepts parameters for make, model, year, and price.**
- **Include methods in the Vehicle class to display vehicle details.**
- **In the main method, create an array of Vehicle objects using different constructors and display their details.**

PROGRAM:

```
class Vehicle {
    private String make;
    private String model;
    private int year;
    private double price;

    public Vehicle() {
        this.make = "Unknown";
        this.model = "Unknown";
        this.year = 0;
        this.price = 0.0;
    }
```

```
    public Vehicle(String make, String model, int year, double price) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.price = price;
    }

    public void displayDetails() {
        System.out.println("Make: " + this.make);
        System.out.println("Model: " + this.model);
        System.out.println("Year: " + this.year);
        System.out.println("Price: $" + this.price);
        System.out.println();
    }

    public static void main(String[] args) {
        Vehicle[] vehicles = new Vehicle[3];

        vehicles[0] = new Vehicle();

        vehicles[1] = new Vehicle("Toyota", "Camry", 2020, 24000.0);
        vehicles[2] = new Vehicle("Honda", "Civic", 2018, 18000.0);

        for (Vehicle vehicle : vehicles) {
            vehicle.displayDetails();
        }
    }
}
```

OUTPUT:

```
Make: Unknown
Model: Unknown
Year: 0
Price: $0.0

Make: Toyota
Model: Camry
Year: 2020
Price: $24000.0

Make: Honda
Model: Civic
Year: 2018
Price: $18000.0
```

Programs on Static Variables and methods

1.Create a Java program to manage student enrollment in a university. Your program should:

- **Use a static variable to keep track of the total number of enrolled students.**
- **Provide a static method to increment the student count when a new student enrolls.**
- **Provide static methods to get the total number of students and to reset the student count.**
- **Create a Student class with a constructor that calls the method to increment the student count.**
- **Demonstrate the use of these static methods in a main method by enrolling students and displaying the total count.**

PROGRAM:

```
class Student {
    private static int totalStudents = 0;

    private String name;

    public Student(String name) {
        this.name = name;
        incrementStudentCount();
    }

    private static void incrementStudentCount() {
        totalStudents++;
    }

    public static int getTotalStudents() {
        return totalStudents;
    }

    public static void resetStudentCount() {
        totalStudents = 0;
    }

    public static void main(String[] args) {
        Student s1 = new Student("Alice");
        Student s2 = new Student("Bob");
        Student s3 = new Student("Charlie");

        System.out.println("Total enrolled students: " + Student.getTotalStudents());

        Student.resetStudentCount();
```

```
        System.out.println("Total enrolled students after reset: " + Student.getTotalStudents());

        Student s4 = new Student("David");
        Student s5 = new Student("Eva");

        System.out.println("Total enrolled students: " + Student.getTotalStudents());
    }
}
```

OUTPUT:

```
java -cp /tmp/MhxOhTGSDi/Student
Total enrolled students: 3
Total enrolled students after reset: 0
Total enrolled students: 2
```

2. Design a class Employee with private member variables id, name, and salary, representing the employee ID, name, and salary, respectively. Implement the following functionalities: A constructor to initialize the employee ID, name, and salary. A static member variable totalEmployees to keep track of the total number of employees. A static member function getTotalEmployees() to return the total number of employees. A member function display() to display the details of the employee.

Write the Java code for the Employee class along with the implementation of the static member variable and function.

PROGRAM:

```
class Employee {
    private int id;
    private String name;
    private double salary;

    private static int totalEmployees = 0;

    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
        incrementEmployeeCount();
    }
```

```
    public static int getTotalEmployees() {
        return totalEmployees;
    }

    private static void incrementEmployeeCount() {
        totalEmployees++;
    }

    public void display() {
        System.out.println("Employee ID: " + this.id);
        System.out.println("Employee Name: " + this.name);
        System.out.println("Employee Salary: $" + this.salary);
    }

    public static void main(String[] args) {
        Employee emp1 = new Employee(1, "Alice", 50000);
        Employee emp2 = new Employee(2, "Bob", 60000);
        Employee emp3 = new Employee(3, "Charlie", 70000);

        emp1.display();
        System.out.println();
        emp2.display();
        System.out.println();
        emp3.display();
        System.out.println();

        System.out.println("Total Employees: " + Employee.getTotalEmployees());
    }
}
```

OUTPUT:

```
java -cp /tmp/eFgEFInUMc/Employee
Employee ID: 1
Employee Name: Alice
Employee Salary: $50000.0

Employee ID: 2
Employee Name: Bob
Employee Salary: $60000.0

Employee ID: 3
Employee Name: Charlie
Employee Salary: $70000.0

Total Employees: 3
```

Programs on Method Overloading and Constructor overloading

1. Create a Java program to manage hotel reservations. Your program should:

- Define a Reservation class with instance variables for reservation ID, guest name, room type, and number of nights.
- Implement overloaded constructors for the Reservation class:
  - A no-argument constructor that initializes default values.
  - A constructor that accepts parameters for reservation ID and guest name.
  - A constructor that accepts parameters for reservation ID, guest name, and room type.
  - A constructor that accepts parameters for reservation ID, guest name, room type, and number of nights.
- Include methods in the Reservation class to display reservation details and calculate the total cost.
- In the main method, create multiple Reservation objects using different constructors, calculate costs, and display their details.

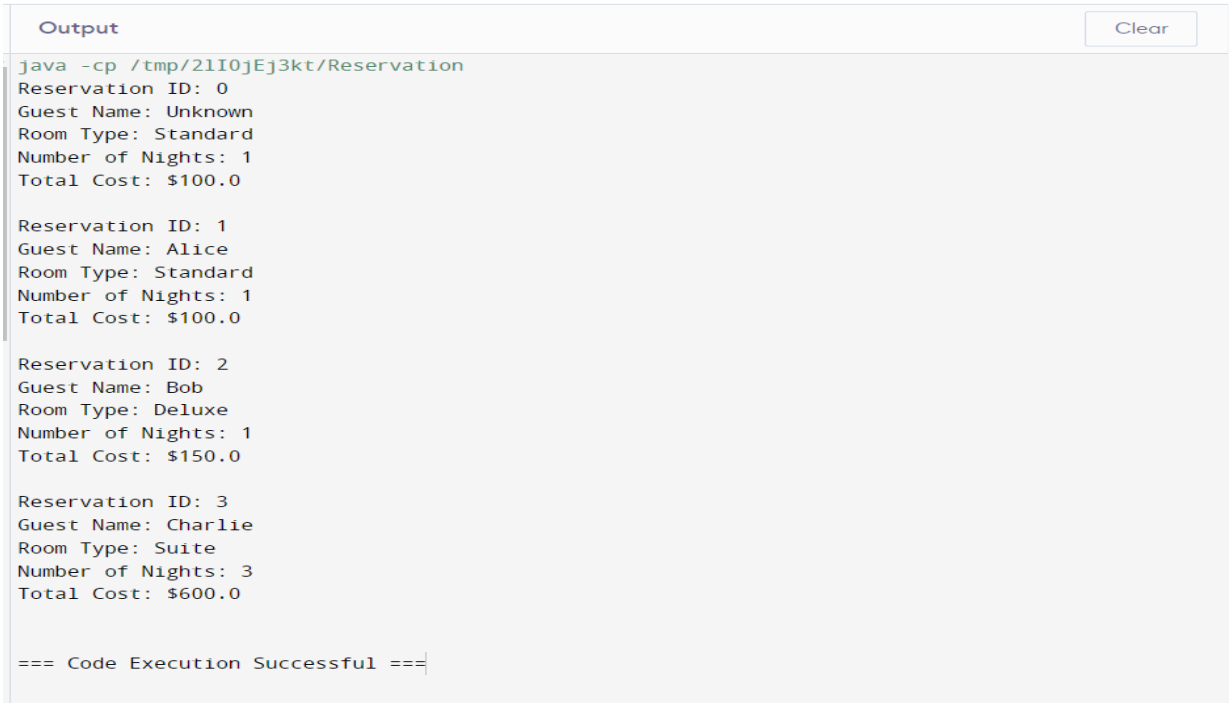
PROGRAM:

```
class Reservation {
    private int reservationId;
    private String guestName;
    private String roomType;
    private int numberOfNights;

    private static final String DEFAULT_ROOM_TYPE = "Standard";
    private static final int DEFAULT_NUMBER_OF_NIGHTS = 1;

    private static final double STANDARD_RATE = 100.0;
    private static final double DELUXE_RATE = 150.0;
    private static final double SUITE_RATE = 200.0;

    public Reservation() {
        this.reservationId = 0;
        this.guestName = "Unknown";
        this.roomType = DEFAULT_ROOM_TYPE;
        this.numberOfNights = DEFAULT_NUMBER_OF_NIGHTS;
    }

    public Reservation(int reservationId, String guestName) {
        this.reservationId = reservationId;
        this.guestName = guestName;
        this.roomType = DEFAULT_ROOM_TYPE;
        this.numberOfNights = DEFAULT_NUMBER_OF_NIGHTS;
    }
```

```
    public Reservation(int reservationId, String guestName, String roomType) {
        this.reservationId = reservationId;
        this.guestName = guestName;
        this.roomType = roomType;
        this.numberOfNights = DEFAULT_NUMBER_OF_NIGHTS;
    }

    public Reservation(int reservationId, String guestName, String roomType, int numberOfNights)
{
        this.reservationId = reservationId;
        this.guestName = guestName;
        this.roomType = roomType;
        this.numberOfNights = numberOfNights;
    }

    public void display() {
        System.out.println("Reservation ID: " + this.reservationId);
        System.out.println("Guest Name: " + this.guestName);
        System.out.println("Room Type: " + this.roomType);
        System.out.println("Number of Nights: " + this.numberOfNights);
        System.out.println("Total Cost: $" + calculateTotalCost());
        System.out.println();
    }

    public double calculateTotalCost() {
        double rate = 0.0;
        switch (this.roomType.toLowerCase()) {
            case "standard":
                rate = STANDARD_RATE;
                break;
            case "deluxe":
                rate = DELUXE_RATE;
                break;
            case "suite":
                rate = SUITE_RATE;
                break;
            default:
                System.out.println("Invalid room type. Defaulting to Standard rate.");
                rate = STANDARD_RATE;
                break;
        }
        return rate * this.numberOfNights;
    }

    public static void main(String[] args) {
```

```
        Reservation res1 = new Reservation();
        Reservation res2 = new Reservation(1, "Alice");
        Reservation res3 = new Reservation(2, "Bob", "Deluxe");
        Reservation res4 = new Reservation(3, "Charlie", "Suite", 3);

        res1.display();
        res2.display();
        res3.display();
        res4.display();
    }
}
```

OUTPUT:



2.Write a Java program that processes orders in an e-commerce application using method overloading. Implement methods to:

- **Process an order given the item ID and quantity.**
- **Process an order given the item ID, quantity, and discount code.**
- **Process an order given the item ID, quantity, discount code, and express shipping option.**
- **Process an order given the item ID, quantity, discount code, express shipping option, and gift wrapping option.**
- **Process an order for multiple items given an array of item IDs and their corresponding quantities.**

PROGRAM:

```
class EcommerceApplication {
   public void processOrder(String itemId, int quantity) {
      System.out.println("Processing order for item ID: " + itemId);
      System.out.println("Quantity: " + quantity);
      double totalCost = calculateTotalCost(itemId, quantity);
      System.out.println("Total Cost: $" + totalCost);
      System.out.println();
   }

   public void processOrder(String itemId, int quantity, String discountCode) {
      System.out.println("Processing order for item ID: " + itemId);
      System.out.println("Quantity: " + quantity);
      System.out.println("Discount Code: " + discountCode);
      double totalCost = calculateTotalCost(itemId, quantity);
      totalCost = applyDiscount(totalCost, discountCode);
      System.out.println("Total Cost after discount: $" + totalCost);
      System.out.println();
   }

   public void processOrder(String itemId, int quantity, String discountCode, boolean
expressShipping) {
      System.out.println("Processing order for item ID: " + itemId);
      System.out.println("Quantity: " + quantity);
      System.out.println("Discount Code: " + discountCode);
      System.out.println("Express Shipping: " + (expressShipping ? "Yes" : "No"));
      double totalCost = calculateTotalCost(itemId, quantity);
      totalCost = applyDiscount(totalCost, discountCode);
      if (expressShipping) {
         totalCost += 15.0; // Adding a flat express shipping fee
      }
      System.out.println("Total Cost after discount and shipping: $" + totalCost);
      System.out.println();
   }

   public void processOrder(String itemId, int quantity, String discountCode, boolean
expressShipping, boolean giftWrapping) {
      System.out.println("Processing order for item ID: " + itemId);
      System.out.println("Quantity: " + quantity);
      System.out.println("Discount Code: " + discountCode);
      System.out.println("Express Shipping: " + (expressShipping ? "Yes" : "No"));
      System.out.println("Gift Wrapping: " + (giftWrapping ? "Yes" : "No"));
      double totalCost = calculateTotalCost(itemId, quantity);
      totalCost = applyDiscount(totalCost, discountCode);
      if (expressShipping) {
```

```
            totalCost += 15.0;
        }
        if (giftWrapping) {
            totalCost += 5.0;
        }
        System.out.println("Total Cost after discount, shipping, and wrapping: $" + totalCost);
        System.out.println();
    }

    public void processOrder(String[] itemIds, int[] quantities) {
        System.out.println("Processing order for multiple items.");
        double totalCost = 0.0;
        for (int i = 0; i < itemIds.length; i++) {
            System.out.println("Item ID: " + itemIds[i] + ", Quantity: " + quantities[i]);
            totalCost += calculateTotalCost(itemIds[i], quantities[i]);
        }
        System.out.println("Total Cost for all items: $" + totalCost);
        System.out.println();
    }

    private double calculateTotalCost(String itemId, int quantity) {
        // For simplicity, assume each item costs $20.0
        return quantity * 20.0;
    }

    private double applyDiscount(double totalCost, String discountCode) {
        // For simplicity, assume a flat 10% discount for any discount code
        return totalCost * 0.9;
    }

    public static void main(String[] args) {
        EcommerceApplication app = new EcommerceApplication();

        app.processOrder("A123", 2);
        app.processOrder("B456", 3, "DISCOUNT10");
        app.processOrder("C789", 1, "DISCOUNT10", true);
        app.processOrder("D012", 4, "DISCOUNT10", true, true);

        String[] itemIds = {"A123", "B456", "C789"};
        int[] quantities = {2, 3, 1};
        app.processOrder(itemIds, quantities);
    }
}
```

OUTPUT:

```
Output
java -cp /tmp/4YJuWIrsQa/EcommerceApplication
Processing order for item ID: A123
Quantity: 2
Total Cost: $40.0

Processing order for item ID: B456
Quantity: 3
Discount Code: DISCOUNT10
Total Cost after discount: $54.0

Processing order for item ID: C789
Quantity: 1
Discount Code: DISCOUNT10
Express Shipping: Yes
Total Cost after discount and shipping: $33.0

Processing order for item ID: D012
Quantity: 4
Discount Code: DISCOUNT10
Express Shipping: Yes
Gift Wrapping: Yes
Total Cost after discount, shipping, and wrapping: $92.0

Processing order for multiple items.
Item ID: A123, Quantity: 2
Item ID: B456, Quantity: 3
Item ID: C789, Quantity: 1
Total Cost for all items: $120.0
```