Assignment 6

Question 1

You are working on an **e-commerce application** where you need to calculate the total price of an order. The system should handle different types of orders:

1. **Single Item**: The total price is calculated based on the price of a single item and its quantity.

2. **Multiple Items**: The total price is calculated based on multiple item prices and quantities.

3. **Discounted Order**: The total price includes a discount, so you need to apply a percentage discount on the total price.

Implement a method calculateTotalPrice() using **method overloading** to handle the following scenarios:

- Calculate the total price for a single item (quantity and price).
- Calculate the total price for multiple items (arrays of prices and quantities).
- Calculate the total price for an order that includes a discount (total price and discount percentage).

Question 2:

You are working on a **travel booking application** where users can book different types of transportation for their journey. The application should handle:

1. **Booking a flight**: The user provides the flight number, departure city, and arrival city.
2. **Booking a train**: The user provides the train number and the class of service (e.g., economy, business).
3. **Booking a cab**: The user provides the pickup location and destination, with an option to specify the number of passengers.

Implement a method bookTransport() using **method overloading** to handle the following scenarios:

- Book a flight by providing the flight number, departure city, and arrival city.
- Book a train by providing the train number and class of service.
- Book a cab by providing the pickup location and destination, with an optional parameter for the number of passengers.

Question 3

You are a student at NIT Calicut participating in a programming competition.

The task is to develop a string manipulation system to manage student names with various functionalities.

You need to create a base class **Student** that has a method **formatName(String name)** to reverse the order of characters in the student's name and contains additional methods for counting the number of vowels in the name and checking if the name is a palindrome (i.e., reads the same forwards and backwards).

Next, create a derived class **NITCalicutStudent** that overrides the **formatName(String name)** method to reverse the student's name and append " (NIT Calicut)" to the reversed name, while inheriting the other functionalities like vowel counting and palindrome checking from the **Student** class.

To demonstrate method overriding, create instances of both the **Student** and **NITCalicutStudent** classes, format the names, and display the results for both types of students, showing the number of vowels in the names and checking whether the names are palindromes.

Sample input

Rajesh

Computer Science

CS123

Sample output

For a **Student** with name **"Rajesh"**

Reversed Name: hsejaR

Vowel Count: 2

Palindrome Check: false

For a **NITCalicutStudent** with name **"Rajesh"**, department **"Computer Science"**, and roll number **"CS123"**:

Reversed Name with Tag: hsejaR (NIT Calicut)

Vowel Count: 2

Palindrome Check: false

Student Details: Rajesh - Department: Computer Science, Roll Number: CS12

Question 4

Develop an online food delivery system for a restaurant chain. The system must manage customer orders for regular customers and premium customers with different delivery options.

1. For **regular customers**, a delivery charge of Rs 5 is applied in addition to the cost of food.

2.For **premium customers**, delivery is free, and only the cost of food is considered.

You are required to:

- Create a base class **Customer** to handle general customer information and functionality.

- Create a derived class **PremiumCustomer** that overrides the method responsible for calculating the total cost, eliminating the delivery charge for premium customers.

Input Format:

- **Regular Customer**: The customer's name and the food cost.

- **Premium Customer**: The customer's name and the food cost.

Expected Output:

1. For a **regular customer**, display the name, food cost, and total order cost (including delivery charge).

2. For a **premium customer**, display the name, food cost, and total order cost (with free delivery).

Sample Input

Alice, 20.0 // Regular customer

Bob, 30.0 // Premium customer

Sample output

Customer: Alice

Total Order Cost: $25.0

Customer: Bob

Total Order Cost: $30.0

Question 5

You are supposed to build a **Payment System** for an e-commerce application, where:

Base Class: Payment

- This class contains two methods:
    - processPayment(): A method to process a generic payment.
    - getPaymentMethod(): A method to get the payment method used (e.g., credit card, debit card, etc.).
    - Provides a calculateTransactionFee() method that returns 0.0 for the base class, meaning no fee by default.

Subclass 1: CardPayment

- This subclass overrides three methods:
    - processPayment(): Implements card-specific payment processing.
    - getPaymentMethod(): Returns the payment method (e.g., "Credit Card").
    - calculateTransactionFee(double PaymentAmount): Calculates a 2% transaction fee on the payment amount.

Subclass 2: SecureCardPayment

- This subclass overrides three methods:
    - processPayment(): Adds security to the payment process and prevents further overriding of this method because this class provides the final, secure way to process payments.
    - getPaymentMethod(): Calls getPaymentMethod() from **CardPayment**.
    - **calculateTransactionFee()** add an additional **$1.00** security fee on top of the 2% transaction fee.

#Main function

```
public class Main {
   public static void main(String[] args) {
      // Create an instance of CardPayment
      CardPayment cardPayment = new CardPayment();
      System.out.println("Card Payment Method: " +
cardPayment.getPaymentMethod());
```

```
        cardPayment.processPayment();
        double cardFee = cardPayment.calculateTransactionFee(100.0);
        System.out.println("Card Payment Transaction Fee: $" + cardFee);
        System.out.println();
        // Create an instance of SecureCardPayment
        SecureCardPayment securePayment = new SecureCardPayment();
        System.out.println("Secure Payment Method: " +
securePayment.getPaymentMethod());
        securePayment.processPayment();
        double secureFee = securePayment.calculateTransactionFee(100.0);
        System.out.println("Secure Payment Transaction Fee: $" + secureFee);
    }
```

OUTPUT:

```
Card Payment Method: Credit Card
Processing card payment...
Card Payment Transaction Fee: $2.0

Secure Payment Method: Secure Credit Card
Processing card payment...
Adding extra security layers to card payment...
Secure Payment Transaction Fee: $3.0
```

2.

Question 6

Design a **Ticket Booking System** for events (like concerts, sports, etc.). The system has two classes:

1. **Base Class: TicketBooking**
    - Contains two methods:
        - bookTicket(): A method to handle the regular ticket booking process.
        - calculateBookingFees(): A method to calculate booking fees based on type of customer. In the base class, booking fees are **10% of the ticket price** for regular tickets.
        -
2. **Subclass: VIPBooking**
    - **Overrides** both methods:
        - bookTicket(): Implements a VIP-specific booking process.
        - calculateBookingFees(): Offers discounted booking fees for VIP members, charging **5% of the ticket price** instead of the regular 10%.

 # Main function

```
public class Main {
    public static void main(String[] args) {
        // Create an instance of TicketBooking (for regular customers)
        TicketBooking regularBooking = new TicketBooking();

        // Create an instance of VIPBooking (for VIP customers)
        VIPBooking vipBooking = new VIPBooking();

        // Process regular ticket booking
        System.out.println("=== Regular Ticket Booking ===");
        regularBooking.bookTicket();
        double regularFee = regularBooking.calculateBookingFees(200.0);
        System.out.println("Regular booking fee: $" + regularFee);
        System.out.println();
        // Process VIP ticket booking
```

```
        System.out.println("=== VIP Ticket Booking ===");
        vipBooking.bookTicket();
        double vipFee = vipBooking.calculateBookingFees(200.0);
        System.out.println("VIP booking fee: $" + vipFee);
    }
```

OUTPUT:

```
=== Regular Ticket Booking ===
Booking regular ticket...
Regular booking fee: $20.0

=== VIP Ticket Booking ===
Booking VIP ticket with additional perks...
VIP booking fee: $10.0
```

Question 7

Write a Java code for an academic management system for NIT Calicut.

The system should handle different types of people at the institute, including **Professor, Student**, and **AdministrativeStaff**.

Each of these types has common properties but also some specific properties and behaviors that differ.

The system should also be able to compute specific roles or responsibilities and calculate the workload or credits for each individual based on different criteria.

Do the following:

- Create an abstract class Person which has variable: *name*, *id, department* and method **getRole()** that will return the role of the person, **calculateWorkload()** that will compute the workload or credits depending on the type of person, **getDetails()** that returns person's name, ID, and department.

- Create an interface Responsibility with a method **listResponsibilities(**) that will be implemented by classes that provide specific responsibilities for each type of person.

- Create a class named Professor that extends Person and implements Responsibility. It has variables: *coursesTaught* - number of courses taught by the professor, *researchProjects* - number of research projects the professor is involved in. It has methods **getRole()** that returns "Professor", **listResponsibilities()** that prints the professor's responsibilities, such as "Teaching, Research, Mentoring Students", **calculateWorkload()** that calculates and returns the workload as:

  Workload=(coursesTaught×3)+(researchProjects×2) (where 3 is the credit for each course and 2 is the credit for each research project).

- Create a class named Student that extends Person. It should have variables *creditsEnrolled* - total credits enrolled in by the student, *gpa* - the GPA of the student. It has the method **getRole()** that returns "Student", **calculateWorkload()** that returns the creditsEnrolled.

- Create a class named AdministrativeStaff that extends Person and implements Responsibility. It should have variables *tasksHandled* - number of administrative tasks handled and methods **getRole()** that returns "Administrative Staff", **listResponsibilities()** method that prints responsibilities like "Managing Files, Organizing Events, Supporting Faculty", **calculateWorkload()** that calculates and returns the workload as: Workload=tasksHandled×1.5 (where 1.5 represents the workload credit per task).

- Create a main method in a class named NITCalicutManagement. Inside the main method create instances of Professor, Student, and AdministrativeStaff with appropriate details. Print out the details, roles, and workload of each person as shown below.

Output:

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

        Professor professor = new Professor("Dr. John Doe", "P123", "Computer
Science", 3, 2);
        Student student = new Student("Alice Smith", "S456", "Electrical
Engineering", 18, 3.8);
        AdministrativeStaff adminStaff = new AdministrativeStaff("Mr. Mark
Johnson", "A789", "Administration", 10);

        // Display details, role, and workload of each person

        System.out.println(professor.getDetails());
        System.out.println("Role: " + professor.getRole());
        System.out.println("Workload: " + professor.calculateWorkload());

         professor.listResponsibilities();

         System.out.println("\n" + student.getDetails());
         System.out.println("Role: " + student.getRole());
         System.out.println("Workload: " + student.calculateWorkload());

         System.out.println("\n" + adminStaff.getDetails());
         System.out.println("Role: " + adminStaff.getRole());
         System.out.println("Workload: " + adminStaff.calculateWorkload());

         adminStaff.listResponsibilities(); }
```

Question 8

Write a Java Program to build a ticket reservation system for different types of events. The system should support various event types and allow users to book and cancel tickets. Base class should be designed in such a way that it should impose implementation of certain functions in the specific event class.

Do the following:

1. All events should have the functions: *getEventName()* returns the name of the event, *getAvailableTickets()* returns the number of available tickets, *bookTickets(int numberOfTickets)* books a specified number of tickets if available and returns whether the booking was successful, *cancelTickets(int numberOfTickets)* cancels a specified number of tickets.

2. Specific events: **Concert** - represents a concert with properties such as *eventName*, *totalTickets*, and *performer*. **TheaterPlay** - represents a theater play with properties such as *eventName, totalTickets,* and *director*. **SportsGame** - represents a sports game with properties such as *eventName, totalTickets,* and *teamNames.* Each class should provide its own implementation of the methods defined in the Event interface. For example: Concert might display the performer in its details. TheaterPlay might display the director in its details. SportsGame might display the team names in its details.

3. Create a class EventManager with the following methods: *addEvent(Event event)* adds an event to the manager's list, *bookEventTickets(String eventName, int numberOfTickets)* attempts to book tickets for an event by its name, *cancelEventTickets(String eventName, int numberOfTickets)* cancels tickets for an event by its name and *displayAllEvents()* displays details of all events.

4. In the main method, create instances of Concert, TheaterPlay, and SportsGame. Add these events to an EventManager instance. Demonstrate booking and canceling tickets, and display event details as shown below.

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

    EventManager manager = new EventManager();

    Event concert = new Concert("Summer Music Festival", 100, "The Jazz
Band");
    Event play = new TheaterPlay("Shakespeare's Hamlet", 50, "John Doe");
    Event sports = new SportsGame("City Derby", 200, "Team A vs Team B");
```

```
        manager.addEvent(concert); manager.addEvent(play);
        manager.addEvent(sports); // Demonstrate booking and canceling tickets
        manager.bookEventTickets("Summer Music Festival", 10);
        manager.cancelEventTickets("Shakespeare's Hamlet", 5); // Display all
events
        manager.displayAllEvents();
}
```