SOFTEWARE APPLICATION PROGRAMMING
PRACTICAL SESSION 7

1. Create a CarRental class with attributes days_rented and car_type, and a method calculate_cost to compute the rental cost based on these rates: Economy ($10/day), Standard ($20/day), and Luxury ($40/day). If the rental period exceeds 3 days, the method should apply a 5% discount to the total cost. Write a program to prompt the user for the number of rental days and car type, create an instance of the CarRental class, invoke the calculate_cost method, and display the total rental cost.

2. Create a Workout class with attributes exercise_type, duration, and weight, and a method calculate_calories to compute calories burned using the formulas: Running (0.2 × weight × duration), Cycling (0.04 × weight × duration), and Yoga (0.02 × weight × duration). Write a program to prompt the user for the type of exercise, duration, and weight, create an instance of the Workout class, and invoke the calculate_calories method. Display the total calories burned and a warning if the duration is below 5 minutes, indicating that the workout is unproductive.
3. Create a ConcertTicket class with attributes ticket_type and ticket_count, and a method calculate_total to compute the cost based on ticket prices: Normal ($20) and VIP ($40). Apply a 10% discount if more than 4 tickets are purchased. Write a program to prompt the user for the ticket type and number of tickets, create an instance of the ConcertTicket class, and invoke the calculate_total method. Display the final cost and, if the total exceeds $300, congratulate the user on being a premium customer.
4. Create a LoanApplication class with attributes annual_income, credit_score, and loan_amount, and a method check_approval to determine if the user qualifies for a loan. The method should approve the loan if the annual income is at least $10,000, the credit score is 500 or above, and the loan amount is less than four times the annual income. Write a program to prompt the user for their annual income, credit score, and loan amount, create an instance of the LoanApplication class, and invoke the check_approval method to display whether the loan is approved or denied.
5. Create a FlightBooking class with attributes destination, travel_class, and days_in_advance, and a method calculate_cost to compute the ticket price based on these rates: Domestic ($100 for Economy, $300 for Business) and International ($600 for Economy, $1,000 for Business). Apply a 10% discount if the booking is made 20 or more days in advance. Write a program to prompt the user for their destination, travel class, and days in advance, create an instance of the FlightBooking class, and invoke the calculate_cost method to display the final ticket cost.
6. Create a DeliveryCost class with attributes weight and destination, and a method calculate_cost to compute delivery charges based on these rules: Local ($2 for up to 4 kg, $4 for up to 10 kg, $10 for heavier packages) and International ($10 for up to 4 kg, $40 for up to 15 kg, $80 for heavier packages). Display a warning if the weight exceeds 4 kg, marking it as undeliverable. Write a

program to prompt the user for the weight and destination, create an instance of the DeliveryCost class, and invoke the calculate_cost method to display the cost or the warning.

7. Create a class called Book to represent a book. A Book should include four pieces of information as instance variables-a book name, an ISBN number, an author name and a publisher. Your class should have a constructor that initializes the four instance variables. Provide a mutator method and accessor method (query method) for each instance variable. Inaddition, provide a method named getBookInfo that returns the description of the book as a String (the description should include all the information about the book). You should use this keyword in member methods and constructor. Write a test application named BookTest to create an array of object for 30 elements for class Book to demonstrate the class Book's capabilities.
8. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables—a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.
9. You are tasked with creating a Customer class for a banking application to manage customer information, including attributes for customerName (String), customerAddress (String), and phone (String). The class must include a default constructor to initialize these attributes with default values and a parameterized constructor to set them using given values. Implement getter and setter methods for each attribute, ensuring the setters validate input, such as checking for a valid 10-digit phone number using a helper method isValidPhone(String phone). Use encapsulation by keeping attributes private and exposing access through public methods. Add method overloading by creating two versions of displayCustomerInfo()—one to display all customer details and another to optionally exclude the phone number when a boolean parameter (includePhone) is false. Finally, create a BankApp test class to: 1) instantiate a customer using the default constructor and display default information, 2) instantiate another customer with valid details via the parameterized constructor, 3) attempt to set an invalid phone number and observe behavior, 4) test both versions of displayCustomerInfo(), and 5) validate phone numbers using the isValidPhone method, demonstrating expected outputs at each step.