LAB ASSIGNMENT

Concept:Classes And Objects, Plymorphism, Inheritance

Objective: At the end of assignments, participants will able to implement Classes And Objects, Plymorphism, Inheritance.

Q1: Create a class named "***Order***" that performs order processing of a single item. It has following fields: customer name, customer number, quantity ordered, unit price.
Include setter and getter methods for each field.
The setter methods prompt the user for values for each field. Create a method(calculatePrice()) to calculate the total price (quantity x unit price) and another method to display total price calculated.
Create a subclass named "***ShippedOrder***" that overrides calculatePrice () by adding a shipping and handling charge of Rs 4.00 per item.
Write an application named "UseOrder" that instantiates an object of each of classe and display the results separately.

Q2 a) Create a class named Year that contains a data that holds the number of days in a year. Make a constructor that sets the number of days to 365. Include a getter method that displays the number of days.
Create another subclass named LeapYear. It calls Year's constructor and sets the number of days to 366.
Make a service class name UseYear that instantiates one object of each class and displays their data. Save the files as Year.java, LeapYear.java, and UseYear.java.

b) Add a method named daysElapsed() to the Year class you created in last Exercise 2(a).
The daysElapsed()method accepts two arguments representing a month and a day; it returns an integer indicating the number of days that have elapsed since January 1 of that year. For example, on March 3 in nonleap years, 61 days have elapsed (31 in January, 28 in February, and 2 in March).
Create a daysElapsed()method for the LeapYear class that overrides the method in the Year class. For example, on March 3 in a LeapYear, 62 days have elapsed (31 in January, 29 in February, and 2 in March).
Write an application named UseYear2 that prompts the user for a month and day, and calculates the days elapsed in a Year and in a LeapYear. Save the files as Year2.java, LeapYear2.java, and UseYear2.java.

Q3. Create a class named HotelRoom having roomNumber as integer and nightlyRentCharge as double.
Include getter methods for required field and a constructor that requires an integer argument representing the room number.
The constructor sets the room rate based on the room number; rooms numbered 299 and below are Rs 6900 per night, and others are Rs 8900 per night.
Create an extended class named Suite whose constructor requires a room number and adds a Rs 400 surcharge to the regular hotel room rate, which again is based on the room number.
Write an application named HotelRoomService that creates an object of each class, and demonstrate that all the methods work correctly.

Q4. Create a class named Vehicle that acts as a superclass for vehicle types. The Vehicle class contains private variables for the number of wheels and the average number of miles per gallon. It also contains a constructor with integer arguments for the number of wheels and average miles per gallon, and a toString()method that returns a String containing these values.
Create two subclasses, Car and MotorCycle, that extend the Vehicle class. Each subclass contains a constructor that accepts the miles-per-gallon value as an argument and forces the number of wheels to the appropriate value—2 for a MotorCycle and 4 for a Car. Write a UseVehicle class to instantiate the two Vehicle objects and print the objects' values.

Q5. Create a class named CollegeCourse that includes data fields that hold the department (for example, "ENG"), the course number (for example, 101), the credits (for example, 3), and the fee for the course (for example, Rs360). All of the fields are required as arguments to the constructor, except for the fee, which is calculated at $120 per credit hour.
Add a display()method that displays the course data. Create a subclass named LabCourse that adds $50 to the course fee. Override the parent class display()method to indicate that the course is a lab course and to display all the data. Write an application named UseCourse that prompts the user for course information. If the user enters a class in any of the following departments, create a LabCourse: BIO, CHM, CIS, or PHY. If the user enters any other department, create a CollegeCourse that does not include the lab fee. Then display the course data. Save the files as CollegeCourse.java, LabCourse.java, and UseCourse.java.