EASTERN INTERNATIONAL UNIVERSITY **Practice Assignment – Quarter 1, 2024-2025**

SCHOOL OF COMPUTING **Course Name:** OOD

AND INFORMATION TECHNOLOGY **Course Code:** CSE 203

Student's Full Name:

Lab 3

Student ID:

Question 1. RetailItem Class
Write a class named **RetailItem** that holds data about an item in a retail store. The class should have the following fields:

- **description**: The `description` field references a `String` object that holds a brief description of the item.
- **unitsOnHand**: The `unitsOnHand` field is an `int` variable that holds the number of units currently in inventory.
- **price**: The `price` field is a `double` that holds the item's retail price.

Write a constructor that accepts arguments for each field, appropriate mutator methods that store values in these fields, and accessor methods that return the values in these fields. Once you have written the class, write a separate program that creates three **RetailItem** objects and stores the following data in them:

Description	Units on Hand	Price
Item #1: Jacket	12	59.95
Item #2: Designer Jeans	40	34.95
Item #3: Shirt	20	24.95

Question 2. CashRegister class
Write a **CashRegister** class that can be used with the **RetailItem** class that you wrote in Question 1. The **CashRegister** class should simulate the sale of a retail item. It should have a constructor that accepts a **RetailItem** object as an argument. The constructor should also accept an integer that represents the quantity of items being purchased. In addition, the class should have the following methods:

- The **getSubtotal** method should return the subtotal of the sale, which is the quantity multiplied by the price. This method must get the price from the **RetailItem** object that was passed as an argument to the constructor.
- The **getTax** method should return the amount of sales tax on the purchase. The sales tax rate is 6 percent of a retail sale.
- The **getTotal** method should return the total of the sale, which is the subtotal plus the sales tax.

Demonstrate the class in a program that asks the user for the quantity of items being purchased, and then displays the sale's subtotal, amount of sales tax, and total.

Question 3. Area Class
Write a class that has three overloaded static methods for calculating the areas of the following geometric shapes:

- circles
- rectangles
- cylinders

Here are the formulas for calculating the area of the shapes.

- **Area of a circle**: Area=πr^2 where π is `Math.PI` and r is the circle's radius.
- **Area of a rectangle**: Area=Width×Length
- **Area of a cylinder**: Area=$\pi r^2 h$ where π is `Math.PI`, r is the radius of the cylinder's base, and h is the cylinder's height.

Because the three methods are to be overloaded, they should each have the same name but different parameter lists. Demonstrate the class in a complete program.

Question 4. Car Instrument Simulator
For this assignment, you will design a set of classes that work together to simulate a car's fuel gauge and odometer. The classes you will design are the following:

- **The `FuelGauge` Class**: This class will simulate a fuel gauge. Its responsibilities are as follows:
  - To know the car's current amount of fuel, in gallons.
  - To report the car's current amount of fuel, in gallons.
  - To be able to increment the amount of fuel by 1 gallon. This simulates putting fuel in the car. (The car can hold a maximum of 15 gallons.)
  - To be able to decrement the amount of fuel by 1 gallon, if the amount of fuel is greater than 0 gallons. This simulates burning fuel as the car runs.
- **The `Odometer` Class**: This class will simulate the car's odometer. Its responsibilities are as follows:
  - To know the car's current mileage.
  - To report the car's current mileage.
  - To be able to increment the current mileage by 1 mile. The maximum mileage the odometer can store is 999,999 miles. When this amount is exceeded, the odometer resets the current mileage to 0.
  - To be able to work with a **`FuelGauge`** object. It should decrease the **`FuelGauge`** object's current amount of fuel by 1 gallon for every 24 miles traveled. (The car's fuel economy is 24 miles per gallon.)

Demonstrate the classes by creating instances of each. Simulate filling the car up with fuel, and then run a loop that increments the odometer until the car runs out of fuel. During each loop iteration, print the car's current mileage and amount of fuel.

Question 5. Parking Ticket Simulator

For this assignment, you will design a set of classes that work together to simulate a police officer issuing a parking ticket. You should design the following classes:

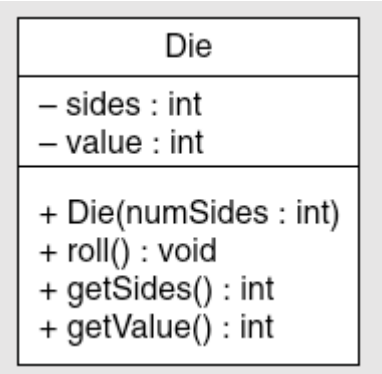
- **The ParkedCar Class**: This class should simulate a parked car. The class's responsibilities are as follows:
  - To know the car's make, model, color, license number, and the number of minutes that the car has been parked.
- **The ParkingMeter Class**: This class should simulate a parking meter. The class's only responsibility is as follows:
  - To know the number of minutes of parking time that has been purchased.
- **The ParkingTicket Class**: This class should simulate a parking ticket. The class's responsibilities are as follows:
  - To report the make, model, color, and license number of the illegally parked car.
  - To report the amount of the fine, which is $25 for the first hour or part of an hour that the car is illegally parked, plus $10 for every additional hour or part of an hour that the car is illegally parked.
  - To report the name and badge number of the police officer issuing the ticket.
- **The PoliceOfficer Class**: This class should simulate a police officer inspecting parked cars. The class's responsibilities are as follows:
  - To know the police officer's name and badge number.
  - To examine a **ParkedCar** object and a **ParkingMeter** object and determine whether the car's time has expired.
  - To issue a parking ticket (generate a **ParkingTicket** object) if the car's time has expired.

Write a program that demonstrates how these classes collaborate.

Question 6. The Dice class

Dice traditionally have six sides, representing the values 1 through 6. Some games, however, use specialized dice that have a different number of sides. For example, the fantasy role-playing game Dungeons and Dragons® uses dice with four, six, eight, ten, twelve, and twenty sides.

Suppose you are writing a program that needs to roll simulated dice with various numbers of sides. A simple approach would be to write a Die class with a constructor that accepts the number of sides as an argument. The class would also have appropriate methods for rolling the die and getting the die's value. Figure below shows the UML diagram for such a class.



Implement the class and demonstrate it in a complete program.

Hint: You can use the Random class in the roll method

Question 7. First to One Game

This game is meant for two or more players. In the game, each player starts out with 50 points, as each player takes a turn rolling the dice; the amount generated by the dice is subtracted from the player's points. The first player with exactly one point remaining wins. If a player's remaining points minus the amount generated by the dice results in a value less than one, then the amount should be added to the player's points. (As an alternative, the game can be played with a set number of turns. In this case, the player with the number of points closest to one, when all rounds have been played, wins.)

Write a program that simulates the game being played by two players. Use the Die class that was presented in Question 6 to simulate the dice. Write a Player class to simulate the players.

Submit: Must include:

- a pdf file containing your information (student id, name), and images of the class diagrams, codes, answers for questions (if any)
- and all source code files (if any)

in a zipped (.zip or *.rar) file to moodle