Assignment 2: Oil Sales Simulator

- Download Template: Assignment02
- Upload: Zip folder on LMS with name and student ID printed on the console
- Submission deadline: June 16, 2023 11:59 pm (midnight)
- Late Submissions will be accepted until June 18, 2023 11:59pm (midnight)

In this assignment, you have to create a simulator for oil sales based on three categories: Gasoline, Diesel and Biofuel. Each category has a different price, taxation or discount that should be evaluated using polymorphism. The base class Oil stores common information to the three derived classes. The sales are further distributed by region/districts. In each region you must generate random sales per category and use multithreading to synchronously add sales to each category. The input and output of the assignment should use file handling. The description and tasks for each class are mentioned below:

1. Oil class

- Create a Oil class that stores information about each candidate. Candidate class has 3 instance variables.
    - String name: name of oil category
    - int numSales: Number of sales of the oil type
    - Sales[] sales: array of sales for each category
    - int costPerLiter: per liter cost of oil category
- Create a constructor that takes two arguments (string name, int maxSales, int cpl).
    - Set the first argument name to the name of the candidate.
    - After creating an array of Sale type, set the size as the second argument, maxSales. Indicates the maximum number of sales of the oil type.
    - Set the costPerLiter to cpl
- toString() method returns a concatenated string of name and numSales in the following format.

```
Name: GASOLINE
Total Sales: 1448
```

- totalSales() method receives int regionNum as an input and returns the number of sales in the region.
- totalCost() method receives int regionNum as an input and returns total cost of oil sales in the region.
- totalQuantity() method receives int regionNum as an input and returns the total quantity in liters of sold oil in the region.
- Create an abstract getPrice method that takes quantity as parameter.
- The Oil class implements Comparable interface. Comparison between oil types is based on numSales, the number of sales per type, and sorting is done in descending order.
- Create a synchronized addSale method that adds each new sale to the array of sales.

2. Biofuel class

- Create the Biofuel class that inherits from the Oil class
- Create the constructor assigns the parameters to the super class and the discount variable to the derived class
- Write a toString() method that returns a string in the following format

```
Name: BIOFUEL
Total Sales: 1421
Discount: 1000
Price: 1000
```

- Create a getPrice method that takes an int quantity and computes the cost of the biofuel. In case of biofuel, there are no additional taxes and a discounted amount of 1000 is provided. The get price method should return the total price based on quantity with the reduction of the discount.

3. Diesel class

- Create the Diesel class that inherits from the Oil class
- Create the constructor assigns the parameters to the super class and the vat and envtax variable to the derived class
- Write a toString() method that returns a string in the following format

```
Name: DIESEL
Total Sales: 1431
VAT: 10%
EnvTax: 20%
Price: 1560
```

- Create a getPrice method that takes an int quantity and computes the cost of the diesel. In case of diesel, there are additional percentages of taxes: vat and environment taxes (envtax) are imposed on the per liter cost of the fuel. The get price method should return the total price based on quantity times the total cost per liter including vat and envtax tax percentages of the per liter cost.

4. Gasoline class

- Create the Gasoline class that inherits from the Oil class
- Create the constructor assigns the parameters to the super class and the vat variable to the derived class
- Write a toString() method that returns a string in the following format

```
Name: GASOLINE
Total Sales: 1448
VAT: 10%
Price: 1760
```

- Create a getPrice method that takes an int quantity and computes the cost of the gasoline. In case of gasoline, only vat is imposed on the per liter cost of the fuel. The get price method should return the total price based on quantity times the total cost per liter including vat percentage of the per liter cost.

5. Sale class

- The Sale class is already created. Sale class has three instance variables (int regionNum: region number, int liters: quantity in liters and int cost: cost of oil). The constructor assigns the values of each instance variable based in the input parameters. This class is used by the Oil class as an array to monitor the sales.

6. Region class

- Create a Region class that stores information about different districts. The Region class should use threading, it must implement the Runnable interface or extend the Thread class.
- The Region class has four instance variables.
    - String regionName: District name
    - int regionNum : region number
    - int overallSales: total sales of the region
    - Oil[] oil : An array of categories / types of oil
- Create a constructor that receives all instance variables as arguments.
- Create the generateSales() method. The method should be implemented as follows:
    - Through this method, sales in a specific region are generated randomly and assigned randomly to one of the three categories.
        - The overallSales of a region should be assigned randomly one by one to any one of the three categories: DIESEL, GASOLINE and BIOFUEL.
        - The quantity of fuel sold in each sale should be generated randomly between 0 to 10 liters using Math.random().
        - The prices of each category based on the quantity should be calculated according to their respective getPrice(int quantity) methods.
        - The sale should be then added to its respective oil category using the addSale(Sale s) method
- Lastly the run() method generates the sales in the thread.

7. OilSim class

- Create an OilSim class to be used as an oil sales simulator. The class reads the input text file, saves it, and runs the oil sales simulation. After that, the execution result is saved to another text file.
- OilSim class has 4 instance variables.
    - string outputFile: path of output file
    - int overallSales: total sales of all regions (maximum number of sales)
    - Oil[] oiltypes: list of categories of oil
    - Region[] regions: list of districts/regions
- Create a constructor that takes two arguments (String inputFile, String outputFile).
    - String inputFile: Input file path
    - String outputFile: output file path
    - The constructor must perform the following operations.
        - Set String outputFile among instance variables.

- Read the inputFile.
- The following operations are performed through the received inputFile.
    - Set overallSales among instance variables.
    - Create a Oil class object and add it to the oil array among instance variables.
    - Create a Region class object and add it to the regions array among instance variables.
    - According to the input file format read the contents of the three oil types : GASOLINE, DIESEL and BIOFUEL and assign their relevant objects to the oil array. Note that the values next to each category are in a given order.
        - GASOLINE <costperliter> <vat %>
        - DIESEL <costperliter> <vat %> <envtax %>
        - BIOFUEL <costperliter> <discount>

```
TOTALSALES 4300
OIL 3
GASOLINE 1600 10
DIESEL 1200 10 20
BIOFUEL 2000 1000
REGIONS 5
Seoul 1 1500
Daegu 2 700
Daejeon 3 300
Gwangju 4 800
Busan 5 1000
```

    - Make sure to do FileNotFound exception handling
- Create the saveData() method that sorts the oiltypes based on total sales and writes the output file. You should use the respective toString() methods of oil types to write in the format shown below. For each region write the total regional sales, total cost and total quantity sold of each category using the totalSales(), totalCost() and totalQuantity() methods.

Name: GASOLINE
Total Sales: 1448
VAT: 10%
Price: 1760

Seoul:
Regional Sales: 511
Total Cost: 3905440
Total Quantity: 2219

Daegu:
Regional Sales: 238
Total Cost: 1958880
Total Quantity: 1113

Daejeon:
Regional Sales: 104
Total Cost: 814880
Total Quantity: 463

Gwangju:
Regional Sales: 280
Total Cost: 2238720
Total Quantity: 1272

Busan:
Regional Sales: 315
Total Cost: 2335520
Total Quantity: 1327

==========

Name: DIESEL
Total Sales: 1431
VAT: 10%
EnvTax: 20%
Price: 1560

Seoul:
Regional Sales: 496
Total Cost: 3165600
Total Quantity: 2308

Daegu:
Regional Sales: 236
Total Cost: 1366080
Total Quantity: 992

Daejeon:
Regional Sales: 97
Total Cost: 588240
Total Quantity: 428

Gwangju:
Regional Sales: 251
Total Cost: 1569000
Total Quantity: 1143

Busan:
Regional Sales: 351
Total Cost: 2176440
Total Quantity: 1585

==========

```
Name: BIOFUEL
Total Sales: 1421
Discount: 1000
Price: 1000

Seoul:
Regional Sales: 493
Total Cost:     2229000
Total Quantity: 2229

Daegu:
Regional Sales: 226
Total Cost:     993000
Total Quantity: 993

Daejeon:
Regional Sales: 99
Total Cost:     465000
Total Quantity: 465

Gwangju:
Regional Sales: 269
Total Cost:     1215000
Total Quantity: 1215

Busan:
Regional Sales: 334
Total Cost:     1446000
Total Quantity: 1446

================================
```

- the runSimulation() method is already. This method starts the oil sales simulation.
    - It start all created regions and waits until all threads started for the regions are finished. When all threads are finished, the saveData() method is called to record the simulation results in the output file.

8. SimTest class

- The class that executes the main method
- Input file path and output file path are fixed as constants.
- Receive input file path and output file path as arguments, create OilSim class object, and call runSimulation() method.

Since the sales are assigned randomly the results may differ from the given output images. To make sure your results are calculated correctly you can compute manually and compare. Following are the points you should check:

- The sum of Total Sales of each category should be equal to the TOTALSALES in the input file.
- The sum of Regional Sales in each category must be equal to the Total Sales of that category.
- The discount has same units as price so it should be directly subtracted while the vat and envtax are percentages that must be computed based on the costPerLiter. Therefore the Price per category in the output file is after adding taxes or reducing discounts and are

different from the cost mentioned in the input file. You should devise the formula carefully.
Eg. In case of Gasoline, the cost per liter is 1600 and vat is 10%. So after adding taxes the price per liter increases to 1600 + 160 = 1760.

- The total cost in each category in each region can be checked by multiplying the Price per liter to the totalQuantity.