CONFIDENTIAL

UTM UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING
Faculty of Engineering

FINAL EXAMINATION
SEMESTER 2, SESSION 2021 / 2022

PART B (PROBLEM SOLVING)

SUBJECT CODE	:	**SCSJ 2154**
SUBJECT NAME	:	**OBJECT ORIENTED PROGRAMMING**
SECTION	:	**(SECJ / SECV / SECB / SECR / SECP)**
DATE/DAY	:	**12 JULY 2022 (TUESDAY)**
TIME	:	**11.20 AM – 01.50 PM**

INSTRUCTIONS:

- You are given 2 HOURS 30 MINUTES to complete the exam inclusive the submission of your answers.
  - ✓ Download the question: 11.20 am – 11.30 am (10 minutes)
  - ✓ Answer the question: 11.30 am – 01.40 pm (2 hours)
  - ✓ Interim submission: 12.30 pm – 12.40 pm (10 minutes)
  - ✓ Final answer submission: 01.40 pm – 01.50 pm (10 minutes)

- A candidate who is suspected of cheating in examinations is liable to disciplinary action including (but not limited to) suspension or expulsion from the University. All materials and or devices which are found in violation of any examination rules and regulation will be confiscated.

IMPORTANT NOTES:

- All the **COMMENT STATEMENTS** in the submitted program **WILL NOT BE EVALUATED**.

SUBMISSION PROCEDURE:

- Compress all files using ***.zip format**. Only the compress file is required for the submission and the file shall be named as follows: ***InterimSCSJ2154_Name_ matricsNo_section.zip*** (for interim submission) and ***FinalAnsSCSJ2154_Name_ matricsNo_section.zip*** (for final submission).

Name	
I/C No.	
Year / Course	
Section	
Lecturer Name	

PROBLEM SOLVING (60 Marks)

Write a complete Java program based on the UML class diagram given in **Figure 1**. Your program should be able to produce the output shown in **Figure 3**.

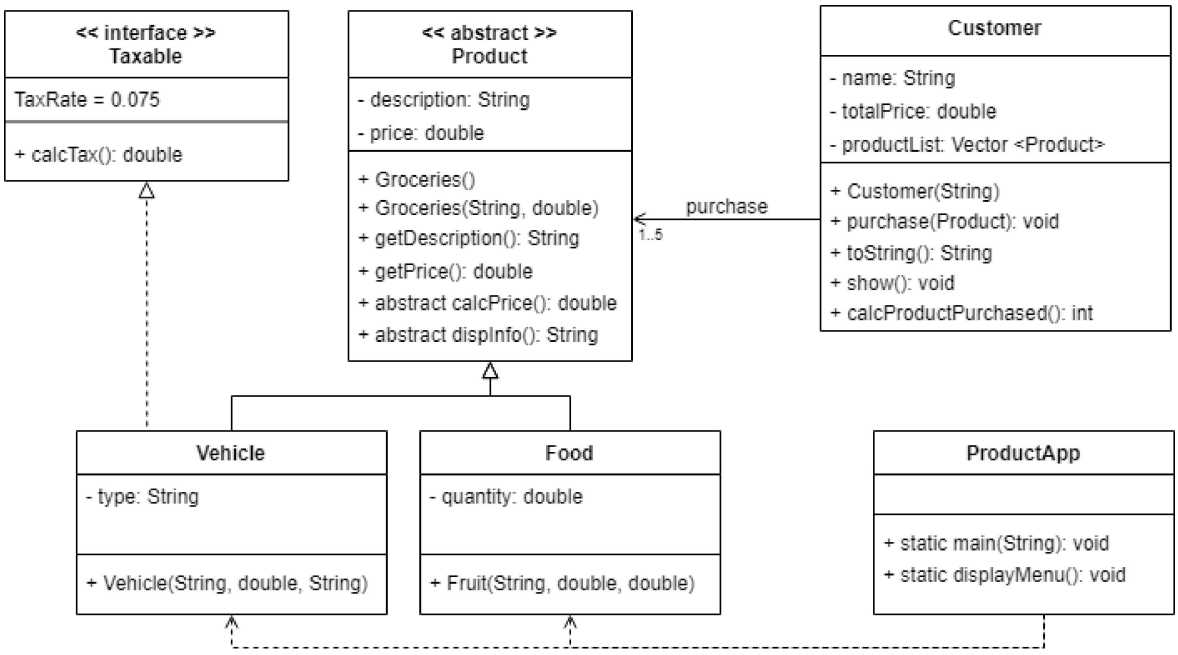

Figure 1: UML class diagram

Implement all the classes with the instance variables (attributes) and methods specified in the diagram. The purpose of each method is as the name implies, and some of them are further explained below. Write the program based on the following tasks:

(a) Write a class named **Customer**. The class provides the following instance/ member variables and methods: (11 Marks)

(i) Define all the attributes of the class. **Note:** You need to use the **Vector** (dynamic array) to declare **productList** attribute. (1 mark)

(ii) Define the constructor with arguments. Initialize all the attributes of the instance with the passed argument and/or zero value. Then, create the object from class **Vector** that is defined in Task a(i). (1 mark)

(iii) Define the **toString** method. The method returns the following sentence: "**<<<** `{Customer Name}'s shopping list` **>>>**". (1 mark)

(iv) Define the **calcProductPurchased** method. The method returns the number of products purchased by a customer. (1 mark)

(v) Define the **purchase** method. The method is used to add an object from the **Product** class to the **productList** array and calculate the total price for all of the products purchased by a customer. The object added to the array represents the product purchased by a customer. (2 marks)

(vi) Define the **show** method. The method is used to display the list and the total commission of products purchased by a customer (if any) in the following format (please refer to the sample output given in **Figure 3)**: (5 marks)

```
Number of products purchased: 2

Product 1
{Information for Product 1}

Product 2
{Information for Product 2}

TOTAL PRICE: RM##.##
```

(b) Write an interface named **Taxable** and declare one final static variable and one abstract method. (1 Mark)

(c) Write an abstract superclass named **Product**. The class provides the constructor with arguments that initialize all the member attributes to the values passed as arguments. It also provides the **getDescription** method, which returns the value of **description** member variable, as well as the **getPrice** method, which returns the value of **price** member variable. (4 Marks)

(d) Write a subclass **Vehicle** that implements the **Taxable** interface with the following codes: (8.5 Marks)

(i) Define the constructor with arguments that will initialize all of the class's member attributes, including the superclass's attributes. (1.5 marks)

(ii) Define the **calcTax** method, which will return the tax paid on each product purchased. (1 mark)

(iii) Define the **calcPrice** method, which will return the price of each product purchased, including taxes. (1 mark)

(iv) Define the **dispInfo** method, which will display the product's description and type, price before and after tax in RM, and tax levied in the following format (please refer to the sample output given in **Figure 3)**: (3.5 marks)

```
Description: ???? ????
```

```
Type: ???? ????
Price before tax: RM#####.##
Price after tax: RM#####.##
The tax levied: RM####.##
```

(e) Write a subclass **Food** with the following codes: (6.5 Marks)

(i) Define the constructor with arguments that will initialize all of the class's member attributes, including the superclass's attributes. (1.5 marks)

(ii) Define the **calcPrice** method, which will return the price of each product purchased, including taxes. (1 mark)

(iii) Define the **dispInfo** method, which will display the product's description and type, price per kg in RM, quantity and subtotal price in RM in the following format (please refer to the sample output given in **Figure 3)**: (3 marks)

```
Product: ???? ????
Price: RM##.## per kg
Quantity: #.#
Subtotal Price: RM##.##
```

(f) Write a driver class named **EidApp**. The class provides the following methods: (26.5 Marks)

(i) Define the **displayMenu** method that will provide the user a menu-driven interaction as follows (please refer to the sample output given in **Figure 3**):

```
========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task:
```

The definition for the **displayMenu** method is fully given in **Figure 2**.

```
1  public static void displayMenu() {
2    System.out.println("========== Menu ==========");
3    System.out.println("[1] Add Customer");
4    System.out.println("[2] Purchase Vehicle");
5    System.out.println("[3] Purchase Food");
6    System.out.println("[4] Display Shopping List");
7    System.out.println("[5] Exit");
8    System.out.println("==========================");
9    System.out.print("\nSelect task: ");
10 }
```

Figure 2: Menu-driven interaction method

(ii) Define the **main** method that will produce the output like the sample output given in **Figure 3** according to the following descriptions: (26 marks)

- Identify the suitable variables and/ or instances where appropriate.
- Create a `Scanner` object for input purpose.
- Ask the user to enter the chosen task. Handling the user input error using an **exception** approach. Note that, all the interactions shown in **Figure 3** are continuous in a single run. Note also that, the **bold** texts indicate input entered by the user.

[1] Add Customer *[1.5 marks]*

- Ask the user to enter the customer's name.
- Create a new **Customer** object.

[2] Purchase Vehicle *[5.5 marks]*

- If there is no customer, display an appropriate message, i.e. "`There is no customer. Please start by entering the customer's name.`".
- Ask the user to enter the product's description and type, and price in RM.
- Create a new **Vehicle** object. **Note:** You should use the polymorphism concept.
- Using an appropriate method defined above, insert the new **Vehicle** object into the **productList** array for the customer.

[3] Purchase Food *[6 marks]*

- If there is no customer, display an appropriate message, i.e. "`There is no customer. Please start by entering the customer's name.`".
- Ask the user to enter the product description, price per kg in RM, and quantity purchased (in kg).
- Create a new **Food** object. **Note:** You should use the polymorphism concept.
- Using an appropriate method defined above, insert the new **Food** object into the **productList** array for the customer.

[4] Display Shopping List *[6 marks]*

- If there is no customer, display an appropriate message, i.e. "`There is no customer. Please start by entering the customer's name.`".
- If there is no product purchased, display an appropriate message, i.e. "`No products were purchased!!`".
- Display the information of all the products purchased by the customer (please refer to the sample output given in **Figure 3**).

[5] Exit *[0.5 mark]*

- End the program by displaying an appropriate message, i.e. "**`Thank you! :)`**".

(g) The program is able to run, work, and display the output as required. (2.5 Marks)

```
========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 2

There is no customer. Please start by entering the customer's name.

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 3

There is no customer. Please start by entering the customer's name.

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 1

<<< Add Customer >>>
Name: Rashid Ali

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 4

No products were purchased!!

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
```

```
[4] Display Shopping List
[5] Exit
==========================

Select task: 2

<<< Purchase Vehicle >>>
Description: Perodua Alza
Type: Car
Price (in RM): 84500

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 3

<<< Purchase Food >>>
Description: Banana
Price (in RM): 4.5
Quantity (in kg): 5

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 4

<<< Rashid Ali's shopping list >>>

Number of products purchased: 2

Product 1
Description: Perodua Alza
Type: Car
Price before tax: RM84500.00
Price after tax: RM90837.50
The tax levied: RM6337.50

Product 2
Description: Banana
Price: RM4.50 per kg
Quantity: 5.0
Subtotal Price: RM22.50

TOTAL PRICE: RM90860.00


========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
```

```
[4] Display Shopping List
[5] Exit
==========================

Select task: 5

Thank you! :)
```

(a) Example output of the first run

```
========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 4

There is no customer. Please start by entering the customer's name.

========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 0

java.lang.Exception: Invalid Input!!
```

(b) Example output of the second run

```
========== Menu ==========
[1] Add Customer
[2] Purchase Vehicle
[3] Purchase Food
[4] Display Shopping List
[5] Exit
==========================

Select task: 6

java.lang.Exception: Invalid Input!!
```

(c) Example output of the third run

Figure 3: Expected output of the program