DIP222 Assignment 2

Due Date: ~~8th April 2020 by 2.00pm~~ Extended: 13th April 2020 by 12pm
Value: 10%

Assessment mode: Individual Assessment

Rationale

This assignment has been designed to give students the opportunity to demonstrate their skill in:

- Solving fairly complex problem involving the design of more than one user defined class.
- Using list to maintain a collection of objects and allow management of them.
- Managing an application which involves a collection of objects.
- Writing and using methods which enable objects to show desired behaviors.
- Using and complying with a supplied specifications for classes to be written.
- Retrieving data from and writing data to files.
- Use of sorting and searching processes.
- Using good programming style.

The assignment assesses the following learning outcomes:
CLO 3: Discuss elements of good programming style
CLO 4: Apply the concept of object orientation as an approach to data abstraction.
CLO 5: Write programs to solve basic computing problems.

Overview

For this assignment, you are going to develop an application which helps the telco manager to manage his/her collection of customers registered for a telco postpaid plan. A customer collection will hold and maintain a list of customers who had registered for the telco service, which may allow the telco manager to display summary information from this collection of customers.

The problem focuses on a container class of user-defined objects and the main features being assessed include your ability to handle several classes working together, the dynamic adding of new objects to a list and searching the list for particular objects.

Task

You are to write **two** source files as described below:

- The first file named **customerApp.py**, should contain the following two classes:
  - The first is a class called `Customer` which defines a simple object type representing a customer in the telco system.
  - The second class called `CustomerGroup` defines objects which are containers of `Customer` objects.
- The second file named **customerDriver.py** defines a Python application, with main method, which creates one `CustomerGroup` object and allows the various methods of `Customer` to be called. This class will be an interactive application using the keyboard and the screen to interact with a human operator. It will not do calculations itself but will immediately pass user inputs as arguments to methods of `CustomerGroup` class.

NOTE: The final application will only execute correctly when both files have been defined completely and correctly but don't wait until you have completely written the two classes before you start testing your code.

The files

The files you will require are:

Class Customer [saved in **customerApp.py]**

Each customer registered will have 4 instance variables:

- name (a string, restricted to a maximum of 20 characters)
- age (an integer)
- postpaid plan (a string, valid value would be 'sp'/ 'SP' for saver plan, 'ap'/ 'AD' for advanced plan and 'pp'/ 'PP' for premium plan)
- postpaid fee (a float, in RM)

The methods of class `Customer` should include:

- An initialiser (constructor) method (__init__) which accepts FOUR attributes for a customer - name, age, postpaid plan and postpaid fee. The method should initialise its object's attributes with these parametric values. Note that the age of customer must be valid (between 18 – 70, inclusive) and the at least one valid postpaid plan must be selected, user will be asked to input the postpaid plan again if the the value is not valid.
- A reader (getter) method for each of the four attributes. That is, a simple 'getter' method for the name, age, postpaid plan and postpaid fees.
- A writer (setter) method for each of the attributes. Note that the postpaid fees will only be changed if the `setValue` method receives a positive postpaid fee.
- A method named `data` which does not accept parameter, but return the amount of data for internet available. The internet data is calculated based on the following:

  Saver plan: multiply the postpaid fee with 0.9

  Advanced plan: multiply the postpaid fee with 1.2

  Premium plan: multiply the postpaid fee with 1.5

  For example a customer who paid RM50.00 in saver plan will have 45GB of internet data (50X0.9). A customer who paid RM65.00 in Premium plan will have 98GB of internet data (65X1.5 = 97.5(round up the value to round number only))
- A method __eq__ that takes another Customer as parameter, and returns true if both Customers are equal (same), false otherwise. Two customers are considered equal if they have the same postpaid plan.
- A method __lt__ that takes another Customer as parameter, and returns true if this customer is having internet data less than the other parametric customer.
- A method __le__ that takes another Customer as parameter, and returns true if this customer is having postpaid fee less than or equal to the parametric customer.
- A string method (__str__) which return a single string containing the details of a customer. Such a string can be formed by concatenating the values of the four attributes name, age, postpaid plan, and postpaid fee in the format:

  `<name> <age> <Postpaid plan> <postpaid fee>`

  One sample output is as shown below:

  `Jimmy 25 premium plan RM156.00`

It is recommended that once you have written the `Customer` class, you create a tiny program to test it. The testing program should be used to create one or two `Customer` objects and call some of the `Customer` methods. Run the test program to check your work.

Class CustomerGroup [written and saved in the same file as **Customer** in **customerApp.py**]

This class is to be defined in the same file as class Customer. It declares a class of object with maintains a list of `Customer` objects. It will contain methods which enable the list to show the appropriate behaviours as required by the menu.

The `CustomerGroup` class should have a group name (of type string), and a collection of `Customer` objects, no additional attribute is required.

The `CustomerGroup` class must also contain some methods which allow the collection of customers to be managed. The methods of class `customerGroup` should include:

1. An initialiser (constructor) (__init__) with one argument of type string, which is used to initialise the group's name. The constructor should also initialise an empty list for customers' subscription.
2. Getter (reader) methods for accessing the attributes, name and customer list, but only setter (writer) method for name.
3. A method named **`addCustomer`** which accepts as an argument an object of class Customer. This method will store a reference to this Customer object into the list
4. A method named **`noOfCustomer`** which returns the number of customers currently stored in the list.
5. A method named **`totalValue`** that does not accepts any argument, but returns the total postpaid subscription value.
6. A method named **`findCustomerByType`** which finds customer(s) with a particular type of postpaid plan. This method accepts a character as the type of postpaid plan. If there is no such postpaid plan in the collection, returns "not found", otherwise returns a string giving details of all the customer with the wanted postpaid plan. Details returned include name, age, and internet data only – one customer per line.
7. A method ~~**`findTotal`**~~ `displayCustomer` which display all the customers in the collection.
8. A method **`highestValue`** which finds and returns the detail of customer with the most internet data
9. A method named **`customerWithdraw`** that takes an integer representing the index of the customer in the list to unsubscribe from the telco plan, return False if it is not successfrnet .
10. A method **`saveToFile`** that accepts a string representing the filename to save, and all customers will be saved to a text file, one per line, with each attribute separated by comma.
11. A method **`loadFromFile`** that accepts a string representing the filename where the data is to load from.

When you have written the `CustomerGroup` class - test it by creating a `CustomerGroup` object and invoking the methods from a small test program.

customerDriver.py

The aim of this file is to provide a user-interface for a modest application which uses a `CustomerGroup` container class. The user-interface is written as a 'console' application using the normal screen and keyboard to interact with a user via a simple text-based menu.

The user-interface should create a single `CustomerGroup` object and provide a menu of choices to the user.

The Menu

```
Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
2 Display all customers
3 Display total value of all postpaid subscription
4 List number of customer in a particular group
5 Display customers with user-specified postpaid plan
6 Display customer with the most internet data in his/her plan
7 Remove customer, based on index
8 Read customer information from file
9 Write customer information to file
0 Quit

Your choice?
```

Sample Run

NOTE: I have truncated display of menu to save space! The menu should display on the screen in full each time it is displayed. User's input is in red, and bold.

```
Enter group's name: Star Customer

1 Add a customer
2 Display all customers
3 Display total value of all postpaid subscription
4 List number of customer in a particular group
5 Display customers with user-specified postpaid plan
6 Display customer with the most internet data in his/her plan
7 Remove customer, based on index
8 Read customer information from file
9 Write customer information to file
0 Quit

Your choice? 2
No customer registered in this group
```

```
Telco Subscription by Private Telco
-----------------------------------------------------------
1 Add a customer
:
7 Remove customer, based on index
8 Read customer information from file
9 Write customer information to file
0 Quit

Your choice? 7
No customer registered in this group


Telco Subscription by Private Telco
-----------------------------------------------------------
1 Add a customer
:
9 Write customer information to file
0 Quit

Your choice? 1
Customer name? James Waterman
Customer age? 35
Postpaid plan('SP/AP/PP')? A
Invalid type! Please enter again!
Postpaid plan('SP/AP/PP')? sP
Invalid type! Please enter again!
Postpaid plan('SP/AP/PP')? sp
Postpaid fee? 0
Invalid value! Please enter again!
Postpaid fee? -300
Invalid value! Please enter again!
Postpaid fee? 50
... Customer has been added successfully.


Telco Subscription by Private Telco
-----------------------------------------------------------
1 Add a customer
:
9 Write customer information to file
0 Quit

Your choice? 1
Customer name? Lee Kar Sing
Customer age? 45
Postpaid plan('SP/AP/PP')? PP
Postpaid fee? 65
... Customer has been added successfully.
```

```
Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
2 Display all customers
:
9 Write customer information to file
0 Quit

Your choice? 2
All customers in this group:
James Waterman 35 saver plan RM50.00
Lee Kar Sing 45 premium plan RM65.00


Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
3 Display total value of all postpaid subscription
:
9 Write customer information to file
0 Quit

Your choice? 3
Total value of all postpaid subscription is RM115 and total internet
data 143GB


Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
4 List number of customer in a particular group
:
9 Write customer information to file
0 Quit

Your choice? 4
Enter group name: Star Customer
Total customer in group 'Star Customer': 2


Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
5 Display customers with user-specified postpaid plan
:
9 Write customer information to file
0 Quit

Your choice? 5
Postpaid plan('SP/AP/PP')?: SP
Customer with saver plan:
James Waterman 35 45G
```

```
Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
6 Display customers with the most internet data in his/her account
:
9 Write customer information to file
0 Quit

Your choice? 6
Customer with the highest internet data:
Lee Kar Sing 45 premium plan RM65.00 98GB


Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
7 Remove customer, based on index
:
9 Write customer information to file
0 Quit

Your choice? 7
Which customer to withdraw? 0
Invalid index. Try again!


Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
7 Remove customer, based on index
:
9 Write customer information to file
0 Quit

Your choice? 7
Which customer to withdraw? 2
Customer with index 2:{Lee Kar Sing 45 premium plan RM65.00} has
been removed!


Telco Subscription by Private Telco
------------------------------------------------------------
1 Add a customer
:
8 Read customer information from file
9 Write customer information to file
0 Quit

Your choice? 9
Please enter filename to save to: star.txt
File saved successfully
```

```
Investment Scheme by Private Banking
------------------------------------------------------------
1 Add a customer
:
8 Read customer information from file
9 Write customer information to file
0 Quit

Your choice? 8
Please enter filename to load from: star.txt
File loaded successfully


Investment Scheme by Private Banking
------------------------------------------------------------
1 Add a customer
2 Display all customers
3 Display total value of all postpaid subscription
4 List number of customer in a particular group
5 Display customers with user-specified postpaid plan
6 Display customer with the most internet data in his/her plan
7 Remove customer, based on index
8 Read customer information from file
9 Write customer information to file
0 Quit

Your choice? 2
All customers:
James Waterman 35 saver plan RM50.00


Investment Scheme by Private Banking
------------------------------------------------------------
1 Add a customer
:
9 Write customer information to file
0 Quit

Your choice? 0

Thank you!
```

Documentation

- You should include comments in your code stating what each method does and explaining any complex sections of code.
- You should also include your student ID as comments within the code.
- You should of course use meaningful variable names so that your code is to some extent self-documenting.

Submission Requirements

- A cover-sheet stating your student number. **NOTE**: Do not put your name on the cover-sheet.
- Printouts of your source code using Courier-New 10 point size font. Make sure that your long Python statements, if any, do not have the second line printed starting from the left-hand margin. You must break the long statement in appropriate length.
- You are **NOT** allowed to print in **landscape** orientation.
- Printouts demonstrating real interaction between yourself and your application
- Hardcopy of the Turnitin report (Word document which contains the source codes)
- A compressed file containing all the source files, with extension **.py** is to be uploaded to elearning.
- Refer to Academic Integrity Policy in e-learning

If your program does not meet the requirements by the due date you should obtain help from the lecturer and notify the lecturer that you will submit the assignment late (marks will be deducted).

Note about testing and plagiarism

It is very important that you complete this assignment alone. You may of course obtain general assistance from the lecturing staff in the subject and your peers, but the coding must be carried out yourself. It is normally quite easy to detect when two or more students work together on their coding.