The University of the West Indies, St. Augustine
COMP 3607 Object Oriented Programming II
2021/2022 Semester 1
Assignment 2

Release Date: October 18, 2021
Due Date: October 29, 2021 at 11:50 p.m.

Overview:
Patrons of a cinema may buy snack items or combos from concession vending machines. A combo may consist of two or more snack items offered together at a discounted price. A combo may also be made up of other combos.

A **Snack** interface is defined as follows:

```
public interface Snack{
    public String getID( );              // returns the ID of a Snack
    public String getName( );            // returns the name of a Snack
    public double getPrice( );           // returns the price of a Snack
    public int getNumItems( );           // returns the number of items making up a Snack
    public String toString( );       // returns the string representation of a Snack with ID, name and price
}
```

Figure 1. Snack Interface Specifications

An **Item** is a concrete class that implements the Snack interface. It represents a single snack that may be purchased from the vending machine. An item ID starts at 1 and is represented as a 3 character string e.g. 001. It is incremented by 1 thereafter.

A **Combo** is a concrete class that implements the Snack interface. It represents a combination of snacks that may be packaged together and offered for sale as a single snack. Use an appropriate collection to maintain a list of the combined snacks. A method must be provided to add snacks to a Combo. The price of a Combo is calculated as 80% of the net price of its snacks. Combo IDs start at 1 but are represented as 5 character strings and the numeric part is incremented by 1 thereafter. Example: 00001. This class may or may not implement the Comparable interface as per the collection you use.

An **Order** class encapsulates the important details of a patron's order. It also lists the snack IDs that make up the order. This class is used to generate a receipt using the toString( ) method.

A **VendingMachine** class processes orders and prints the receipts for each order. It stores lists of items and combos.

Sample lists: (to be loaded using the text files supplied).

Snack Items

ID	Name	Price
001	Popcorn	$10.00
002	Nachos	$10.00
003	Pizza	$15.00
004	Fries	$5.00
005	Hotdog	$10.00
006	Cake	$5.00
007	Soda	$5.00
008	Coffee	$5.00
009	Water	$5.00

Snack Combos

ID	Name	Snacks by ID
00001	Combo1	001, 005, 007
00002	Combo2	004,009
00003	Combo3	008,008,006,006
00004	Combo4	003,007,00002
00005	Combo5	002,007,009,00001,00002

Orders

ID	Type	Snacks by ID
1	CASH	00005
2	CASH	001, 003
3	CASH	005
4	CARD	001
5	CARD	00004, 001

Assignment Tasks:

(a) Design
Draw a complete class diagram that models a solution for the scenario above using the Composite design pattern.

(b) Code
Write Java code to create a working solution based on your design in part (a) that uses the Composite Design Pattern. Document your program appropriately and ensure that any data files you use conform to the sample lists provided on myElearning and on this assignment description. You may include additional private methods and attributes as appropriate however the Snack interface must conform to the signatures listed in Figure 1.

Name your main class **Cafeteria.java.**

Submission Instructions

- Ensure your student ID is documented in your submission.
- State any assumptions that you make. Cite any sources used as appropriate.
- Label all answers and diagrams appropriately based on the question part answered.
- Upload your submission for (a) as a PDF file to the myElearning course page by the deadline.
- Upload a zipped file of your solution for (b) as a Visual Studio Code project to the myElearning course page by the deadline.
- If you are using your credit day(s), please indicate on your submission.
- Sign and include the plagiarism declaration form. This is an individual assignment that should be completed on your own.