The University of the West Indies, St. Augustine
COMP 3607 Object Oriented Programming II
2019/2020 Semester 1
Assignment 2

Due Date: October 30, 2019 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.

Patrons pay for items or combos by making a cash payment based on the amount of money they have on hand or by making a card payment for the exact amount with a valid credit card (based on expiry date). After a payment is validated, the patron is given a receipt for their purchase. The receipt lists the number of bills and coins returned for cash payments. For card payments, the receipt indicates whether the payment was approved or not.

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

```
public interface Snack{
        public int 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 getDetails( );    // returns the string representation of a Snack
}
```

An **Item** is a concrete class that implements the Snack interface. It represents a single snack that may be purchased from the vending machine. Item IDs start from 100 and are incremented by 100 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 items plus the cost of any sub-combos. Combo IDs start at 100000 and are incremented by 100000 thereafter. 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.

A **Payment** interface specifies a method **validatePayment(Order order): boolean** which works for both **Cash Payments** and **Card Payments**. Card payments are valid if the date on the card has not expired. Cash payments are valid if the cash supplied is more than or equal to the payment due for an order. For cash payments exceeding the total amount, the vending machine returns changes in the form of $20, $10, $5 or $1 bills or 25, 10, 5, 1 cent coins as necessary. For example, if a Patron has $100.00 cash on hand, and the cost of his snacks is $45.00 then the change due would be $55.00. The vending machine therefore dispenses two $20 bills, one $10 bill, and one $5 bill. The number of bills to be returned is calculated starting from the highest to the lowest denomination.

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
100	Popcorn	$10.00
200	Nachos	$10.00
300	Pizza	$15.00
400	Fries	$5.00
500	Hotdog	$10.00
600	Cake	$5.00
700	Soda	$5.00
800	Coffee	$5.00
900	Water	$5.00

Snack Combos

ID	Name	Snacks by ID
100000	Combo1	100, 500,700
200000	Combo2	400,900
300000	Combo3	800,800,600,600
400000	Combo4	300,700,200000
500000	Combo5	200,700,900,100000,200000

Orders

Type	Cash supplied / Card Expiry Date	Snacks by ID
CASH	100.00	500000
CASH	50.00	100, 300
CASH	5.00	500
CARD	2018-11-25	100
CARD	2018-01-01	400000, 100

Assignment Tasks:

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

(b) Code
Write Java code to create a working solution based on your design in part (a). Document your program appropriately and ensure that any data files you use conform to the sample lists on page 2.
Name your main class **Cafeteria.java.**

Submission Instructions
Your student ID must be documented in your code/documents. Upload a zipped file of your submission to the myElearning course page by the deadline.