Assignment 1
Grocery Distribution

CSIS 3475

Assignment

- Download **Assignment 1.zip** and import it into an Eclipse workspace using the standard instructions.
- The following packages are included
  - GroceryDistribution – this contains the assignment template
  - StackPackage – from class exercises
  - QueuePackage – from class exercises
  - QueueDemoPackage – from class exercises
- Complete the classes that implement StackInterface and QueueInterface
  - ArrayStack, LinkedStack, VectorStack in StackPackage
  - ArrayQueue, LinkedQueue, TwoPartCircularLinkedQueue, LinkedDeque in QueuePackage
  - Test the implementations using the demo or driver programs provided.
  - You should have already completed these as a part of in-class work. Simply copy your completed code from the CSIS3475 project into the appropriate files.
  - Make sure your name is in the @author field in each file that you complete.
- Complete the tasks that follow implementing a grocery distribution system. Template code files are in the GroceryDistribution package.
- Submit the completed projects using the standard submission instructions
  - Export the projects to a zip archive named **Assignment 1 YourName.zip** where YourName must be your first initial and last name.
  - You MUST use the submission instructions exactly or you will lose marks.
  - You MUST name the archive correctly or you will lose marks.
  - For example, for Michael Hrybyk
    - Assignment 1 MHrybyk.zip

Grocery Distribution overview

- Trucks with grocery items are loaded in Calgary, Toronto and California.
- Each grocery item has a destination: Surrey, New Westminster, or Vancouver.
- The trucks drop off all items at the BC Warehouse.
- New trucks with a destination of Surrey, New Westminster, or Vancouver are then loaded from the warehouse with items matching the destination.

Trucks and the warehouse

Item : String
Size : Int
[Destination : String or Enum?
Source : String or Enum?

- A truck is a stack.
  - As items are loaded, they are placed at the front of the truck.
  - When items are unloaded, the items at the back of the truck are taken off first.
  - LIFO
- The warehouse is a queue
  - As items are taken off the trucks, they are placed at the front of the warehouse.
  - Items are taken from the front of the warehouse and placed on trucks heading for the item's destination.
  - FIFO

Program logic

- A CSV (comma separated file) **GroceryItems.csv** is provided.
  - Each item in the file has a name, item weight in kilograms, origin, and destination.
  - Note the first line in the file must be skipped as it is a header (not data)
- Each item needs to be placed in a truck associated with its origin. → TRUCK( )
  - For example, all items with an origin of Calgary should be loaded in a separate truck.
- Trucks from Calgary, Toronto, and California arrive at the warehouse in that order, and unload their items.
- After all items have been unloaded at the warehouse, they are then loaded onto trucks headed for their destination: Surrey, New Westminster, or Vancouver.
- Once at their destination, the items are unloaded.

Tasks - classes

Item.
size
Destination
origin.

- Create the GroceryItem class
  - Make sure it has a toString() override method returns a string in the format equivalent to that found in the Output.txt file.
    - GroceryItem [itemName=Nantucket Apple Juice, itemWeight=53, destination=New Westminster, origin=Calgary]
- Create a **generic** Truck<T> class
  - It must use one of the classes that implements StackInterface<T>.
    - Hint: use inheritance
  - It should have strings for origin and destination

Tasks – main program **SendStockToStores**

Calg

- Create three trucks, one for each origin.
- Read the CSV file. For each item in the file, create a GroceryItem object and place it in the correct origin truck.
- Display the items in each origin truck.
- Create a warehouse object. This is a queue.
- For each origin truck, unload the items. Place each item in the warehouse object (queue).
- Display all the items in the warehouse
- Create three more trucks, one for each destination.
- For each item in the warehouse, load it into the correct destination truck.
- Display all the items in each truck
- Unload the items in each truck, displaying each item as it is unloaded.

Testing

- Make sure your code works with each Stack and Queue implementation
  - Truck class (Truck.java)
    - ArrayStack
    - LinkedStack
    - VectorStack
  - Warehouse in main program (SendStockToStores.java)
    - ArrayQueue
    - LinkedQueue
    - TwoPartCircularLinkedQueue
  - Note: although **LinkedDeque must be completed**, you do not need to include it in the SendStockToStores.java program.
- Do this by adding lines of code and commenting out the ones not being tested. See example below.
- **Output must correspond exactly to what is found in Output.txt**

```
QueueInterface<GroceryItem> warehouse = new ArrayQueue<>();
//QueueInterface<GroceryItem> warehouse = new LinkedQueue<>();
//QueueInterface<GroceryItem> warehouse = new TwoPartCircularLinkedQueue<>();
```

Grading

Item	Marks
Project properly named and submitted	.2
All code properly formatted and commented	.2
All Stack and Queue implementations completed and tested in demo programs	1
GroceryItem and Truck classes correctly created. Truck class uses StackInterface.	.6
SendStockToStores (main program code) correctly completed and produces proper output.	2.5
All Stack and Queue implementations tested in SendStockToStores and Truck.	.5
Total	5