Term Project (25%)

Check D2L for details on due date

Documentation

Your documentation is a key deliverable for this project. You must provide a UML Diagram for each class and pseudocode for your methods. In addition, your code must include appropriate JavaDoc blocks as has been discussed in class and that you have used for other assignments when it is submitted.

UML Diagram

You must provide a UML diagram for your classes including the class variables, variable types and methods. Indicate private (-) versus public (+) in your diagram. You can refer to Chapter 3 for information on UML diagrams and Appendix B for a sample. Include your file in your Dropbox submission.

Pseudocode

Prior to writing your methods and classes, create the pseudocode for your methods in readable English. Your pseudocode should describe how you plan to implement your solution. Remember that pseudocode is not true code and so don't spend time structuring it using Java syntax. You may find that you ultimately alter your implementation when you code your solution which is acceptable. Include your file in your Dropbox submission.

Your Code

For this project you will create a simple store inventory system. This will require you to use several of the concepts you learned throughout this course.

You will need to create the following classes. Note that the minimum methods are listed, but your implementation will likely require you to create additional methods. Remember to keep the concept of single responsibility in mind to ensure your methods don't get overly complicated. Move code to other methods where it makes sense.

Your method class variables should be private and your code should use getters/setters (accessor/mutator methods) where needed.

`class Store`

This class contains information about your store. It will include the `name` of the store as well as `items` that are available for purchasing. You must use an Array List to store the items.

Minimum Methods:

- Constructor that accepts the `name` field and stores it

- `displayInventory()` that prints the full store inventory (see Appendix A for sample) Note that the location information from your enum type is used here.
- `displayAvailableInventory()` that prints the available store inventory

`class Item`

This class will contain information about a particular item in your store including a `manufacturer`, `model`, `price`, `quantity` and `type`. For the `type` you must use an Enum Type (see section 8.9). Each `item` must be associated with a `store`.

For your enum type, each constant should have "name" and "storageLocation" field associated with it.

Minimum Methods:

- Constructor that accepts `manufacturer`, `model`, `price`, `quantity`, `type` and `store` and stores them.

`class Customer`

This class will contain information about a customer who shops at your store. This information will include the `firstName`, `lastName`, `emailAddress`, `phoneNumber` and a history of `invoices`. The history of invoices must use an Array List to store the invoices so that it can be dynamically resized. Each `customer` must be associated with a `store`.

Minimum Methods:

- Constructor that accepts `firstName`, `lastName`, `emailAddress`, `phoneNumber` and stores them.
- `addInvoice(Invoice invoice)` will add an invoice to a customer's invoices list
- `displayProfile()` will print a customer's profile (see Appendix A for a sample)

`class Invoice`

This class will contain information about a particular sale. The information that is tracked with this class includes the `itemsSold`, the `customer` and the `store`. `itemsSold` must use an Array List to store the items so that it can be dynamically resized.

Minimum Methods:

- Constructor that accepts a `customer` and `store` object and stores them
- `addItem(Item item)` will add an item to the invoice's list of items. This must ensure the quantity of the item is greater than 0 or it should throw an exception and not attempt to add it. Once an item is added to an invoice, the store inventory must be updated to reflect the sale.
- `displayInvoice()` will print an invoice (see Appendix A for a sample)

`class StoreTest`

You will use this class to test your other classes. You can use the standard `main` method for this.

Testing

To ensure your classes, methods and relationships are working properly, the following tests will need to be run. This testing should be in your `StoreTest` class. You can create methods inside of this to keep things simple, but your `main(String[] args)` method should call any methods that are required to run the full test.

1. Create two store objects – "My First Store" and "My Second Store"
2. Create 10 items for each store and add those to the inventory of each store. You can create any items you want with a variety of quantity numbers all greater than 10.
3. Run the `displayInventory()` for each store and confirm the layout in your output matches the layout in Appendix A (the items, quantity and price do not have to match the sample).
4. Create two customers for each store with unique first and last names, e-mail addresses and phone numbers.
5. Select one customer from one store and create a new invoice and use the `java.security.SecureRandom` API to randomly add 5 items from the store to the invoice using the `addItem()` method. Ensure the store quantity is being updated to reflect the change in quantity when added to an invoice. You must use a try-catch block when adding items.
6. Run the `displayInvoice()` method and check that the output matches the provided sample
7. Create a second invoice and add one item using the `addItem()` method. You must use a try-catch block when adding items.
8. Run the `displayProfile()` method on your customer and check that the output matches the provided sample.
9. Add a new item to one of your stores and set the quantity to 0.
10. Run the `displayAvailableInventory()` for the store and confirm the layout in your output matches the layout in Appendix A (the items, quantity and price do not have to match the sample) and that the 0 item is not displayed.
11. Select a different customer from the same store as used in #9 and create a new invoice. Use a for loop to add all your store items the invoice. Check to ensure you do get an exception for the 0 quantity item and that it is not a part of your invoice by running the `displayInvoice()` method. You must use a try-catch block when adding items.

Evaluation

Ensure you submit your project on time. Late submissions will have marks deducted:

- Late by one class day 10% penalty
- Late by two class days 20% penalty
- Late by three class days 40% penalty

Your project will **not be accepted** after the third day.

Rubric

Documentation	
UML Documentation includes all classes and variables	0.25
UML Documentation includes all methods	0.25
UML Documentation includes private/public indication for information	0.25
Pseudocode covers all methods and classes	1
Pseudocode accurately conveys intended solution	1
Javadoc block provided for all classes and includes @author and @version elements	1
Javadoc block provided for all methods and includes appropriate @param and @return elements.	1
Store Class	
The store class contains a name field and items are stored in an Array List	0.5
Full store inventory display of all items is in the format specified	1
Available store inventory displays only items where quantity > 0 in the format specified	0.5
The inventory display is only coded once and referenced by the two display methods	0.5
Class variables are private and getter/setter methods exist	0.25
Item Class	
The item class contains all of the required fields	0.5
The item class uses an enum type for the type field as specified	1.5
The item class contains a store object associated with it	0.5
Class variables are private and getter/setter methods exist	0.25
Customer Class	
The customer class contains all of the required fields	0.5
The customer class has an Array List of invoices	1
The customer class has a displayProfile method that outputs the customer profile in the format specified	1.5
The customer store contains a store object associated with it	0.5
Class variables are private and getter/setter methods exist	0.25
Customer class implements the addInvoice() method	1.00
Invoice Class	

The invoice class contains all of the required fields	0.50
The invoice class has an Array List of items	1.00
The invoice class contains a store object associated with it	0.50
The invoice class contains a customer object associated with it	0.50
Class variables are private and getter/setter methods exist	0.25
The invoice class implements the addItem method which throws an exception if 0 and updates the store inventory quantity	2.00
StoreTest Class	
Two stores created with 10 items each	1.00
Two customers for each store created with random names, email and phone	0.50
Invoice for one customer created with 5 random items using SecureRandom	1.50
Second customer invoice created for same customer with one item	0.50
Added 0 quantity item	0.25
Create new invoice for second customer using for loop to add all items to invoice including 0 quantity item	0.50
Adding item uses a try-catch block for 0 item	1.00
	25

Appendix A: Sample Outputs

Store Inventory Sample:

```
------------------------------------------------------------------------
Store Name: My First Store
------------------------------------------------------------------------
Manufacturer          Model                       Type     Qty    Price
------------------------------------------------------------------------
Apple                 iPhone 11 64GB              PHN       13      679
    Location: Storage Shelf 17-3
Apple                 iPhone 12 64GB              PHN       18      799
    Location: Storage Shelf 12-3
Apple                 iPhone 13 Pro 256GB         PHN        7     1089
    Location: Storage Shelf 11-8
Samsung               Galaxy Z Fold3 5G           PHN        0     2269
    Location: Storage Shelf 8-12
Google                Pixel 6 128GB               PHN       22      799
    Location: Storage Shelf 2-17
Belkin                iPhone 12 Case              ACC       12       29
    Location: Storage Shelf 7-4
------------------------------------------------------------------------
```

Sample Customer Profile:

```
------------------------------------------------------------------------
Name: Cantwell, David                       Total Sales: 1627
E-mail: david.cantwell@cna.nl.ca            Store: My First Store
Phone: (709) 555-1212
------------------------------------------------------------------------
Invoice 1
Purchase Date: 31-Oct-2021
Items Purchased
    Apple iPhone 12 64GB                                          799
    Belkin iPhone 11 Case                                          29
    ---------------------------------------------------------------
    Total                                                         828
------------------------------------------------------------------------
Invoice 2
Purchase Date: 30-Sep-2020
Items Purchased
    Apple iPhone 12 64GB                                          799
    ---------------------------------------------------------------
    Total                                                         828
------------------------------------------------------------------------
```

Sample Invoice

```
------------------------------------------------------------------------
Store Name: My First Store
------------------------------------------------------------------------
Name: Cantwell, David
E-mail: david.cantwell@cna.nl.ca
Phone: (709) 555-1212
Purchase Date: 31-Oct-2021
------------------------------------------------------------------------

Apple iPhone 12 64GB                                              799
Belkin iPhone 11 Case                                              29


------------------------------------------------------------------------
Total                                                             828
------------------------------------------------------------------------
```

Appendix B: UML Sample