Computer Science

COMPSCI 101 S1 C - Assignment 1

Due Date: Thursday 16th May 2011 at 4:30pm

100 marks in total = 5% of the final grade

Assessment

- Due: 16th May, 2013 (4:30 pm)
- Worth: 5% of your final mark

Files

- A copy of this handout, as well as relevant files for questions in this assignment can be obtained from the 101 assignments page on the web:

  `http:// www.cs.auckland.ac.nz/courses/compsci101s1c/assignments/`

- Using the web drop box (`https://adb.ec.auckland.ac.nz/adb/`), you must submit the following files for this assignment:
  - `Application.java, GroceryStore.java, Stock.java, Cart.java, SalesItem.java, Keyboard.java, A2.txt`

Aims of the assignment

- solving problems using if statements, loops, classes, one-dimensional arrays of objects, etc.

Warning

- The work done on this assignment must be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone for help. Under no circumstances should you take or pay for an electronic copy of someone else's work, modify it, and submit it as your own.
- Penalties for copying will be severe – to avoid being caught copying, don't do it.
- To ensure you are not identified as cheating you should follow these points:
  - Always do individual assignments by yourself.
  - Never show your code to another person.
  - Never put your code in a public place (e.g., forum, your web site).
  - Never leave your computer unattended. You are responsible for the security of your account.

Your name, UPI and other notes

- In *this assignment* your UPI **must** appear in the output for the program.
- In this handout, the UPI `abcd001` has been used to illustrate this in the examples, however make sure that for the programs **you** submit, **your** UPI is displayed.
- All files should also include your name and ID in a comment at the beginning of the file.
- All your files should be able to be compiled without requiring any editing.
- All your files should include good layout structure.

A Simple Grocery Store Program *(100 Marks)*

In this assignment, you are going to implement a simple grocery store program. Your program should display the items on sale, assist the shopping process such as add/remove items to/from the shopping cart and check out the purchase. On completion of a checkout, the program should produce a bill on screen, listing the items bought and the amount to be paid for the purchases.

The items on sale are provided in a file called "`stock.txt`". As seen below, each line holds the information for an item including the item code, the description, the price, and the quantity of the item available. The information in the file is read into the program and stored in an object of the "`Stock`" class using an array named "`items`" (this piece of code is given to you). Please remember that if an item is purchased (by a customer), its quantity value represented by the "`quantity`" field of the "`SalesItem`" object should be decreased accordingly. When the program exists, it will save the updated stock items (with the new quantity values, if changed) into a file called "`stock2.txt`" (this piece of code is also given to you). Please DO NOT modify the content of the "`stock.txt`" file.

```
1001,Fresh toast bread white (700g),3.99,20
1002,Low-fat milk (2 liter),4.80,10
1003,V-energy drink,2.75,10
1004,Fresh garlic (450g),1.98,5
1005,Coca-Cola (300 ml),2.50,10
1006,Pineapple,3.60,6
1007,Mango,1.89,4
1008,Snickers chocolate bar,1.80,20
1009,Broccoli,1.47,11
1010,Washed Potato (2.5kg),2.98,7
1011,Good-morning cereal,5.60,10
1012,Rose apple (1.5kg bag),4.98,5
1013,Avocado (4pk),4.99,5
1014,Bananas (850g bag),2.96,4
1015,Kiwi fruit green (1kg),2.45,10
1016,Rock melon,7.98,2
1017,Lettuce,2.99,12
1018,Chocolate block (200g),3.59,10
```

You are provided with the following java files:

- `Application.java` – contains the `Application` class to be executed.
- `Keyboard.java` – contains the `Keyboard` class to read input from the keyboard.
- `SalesItem.java` – contains the `SalesItem` class to store the information of a sale item.
- `Stock.java` – contains the `Stock` class TO BE COMPLETED so that it displays the items on sale and finds a particular item based on the item code.
- `Cart.java` – contains the shopping `Cart` class TO BE COMPLETED so that it adds and removes items to the purchase and produces the final bill when checking out.
- `GroceryStore.java` – contains the `GroceryStore` class TO BE COMPLETED so that it displays the main and sub menus of the program and calls the methods defined in the `Stock` and shopping `Cart` classes to fulfill the functions of online shopping.

You need to understand all the files, but the only files that you are allowed to modify are the `Stock.java`, `Cart.java` and `GroceryStore.java` files. There are four methods defined in the `Stock` class as follows.

Methods	**Functionalities**
`public void displayItems()`	This method displays all the items on sale on the screen. Each individual item contains the item code, description, price and quantity.
`public SalesItem findItem(String itemCode)`	This method looks up the items array and finds the item matching

	the given item code. The method returns the item found (or null, if not found).
`public void loadItems(String filename)` (Implementation is **GIVEN** to you.)	This method reads the stock list file (given by the file name) that contains the information about each individual item on sale, such as the item code, description, price, and quantity, and stores the item information into the items array. The method also updates the total number of items on sale.
`public void saveItems(String filename)` (Implementation is **GIVEN** to you.)	This method saves the information of all the items on sale into a text file (indicated by the file name). The format of the text file is the same as the stock list file, which contains the information about each individual item, such as the item code, description, price, and quantity.

There are four methods defined in the shopping `Cart` class as follows.

Methods	**Functionalities**
`public void addItem(SalesItem item)`	This method adds an item to the shopping cart. The item is stored in the array purchases and the number of items in purchases increases by 1.
`public SalesItem deleteItem(String itemCode)`	This method removes an item from the shopping cart based on the given item code. The item is removed from the array purchases and the number of items in purchases decreases by 1. The method returns the item deleted, or null (if the item code is not found).
`public void deleteAll()`	This method removes all items in the shopping cart by deleting all the elements stored in the purchases array and set the number of items in purchases to zero.
`public void checkOut()`	This method scans through the purchases array and displays the items bought with their price. The method also calculates and displays the total amount due.

There are four methods defined in the `GroceryStore` class as follows.

Methods	**Functionalities**
`public void start()`	This method manages the top and sub-level menus of the system, and calls the corresponding methods defined in the `Stock` and `Cart` classes under the different menu options to achieve the desired functionalities of the system.

`private void topMenu()`	This method displays the top-level menu, which has the options of printing the items on sale, shopping online and exiting the system.
`private void subMenu()`	This method displays the second level menu, which has the options of adding or removing item to or from the shopping cart, checking out the shopping cart, and exiting the shopping without buying.
`private int getChoice(int lower, int upper)`	This method obtains a correct user input option. It checks whether an input is within the range of the indexes of the menu items (indicated by the lower and upper values). The method returns the user input as an integer.

Complete the required methods accordingly to achieve the functions of the program. **IMPORTANT**: Test each method after you have completed it and make sure that the program runs as described.

1. Assign your own UPI to the UPI constant defined in the `start` method of the `GroceryStore` class. `[5 marks]`

2. Display the top and second level menus and get the user's input choice. `[25 marks]`

There are two levels of menus in the program. The top-level menu consists the options such as showing the information of all items on sale, shopping online and exiting the program. Its exact layout should have the same format as shown below.

```
-------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
-------------------------------------------------------------
```

The second-level menu is used when a user selects the '`Start to shop online`' option. It consists the options such as adding or removing an item to/from the shopping cart, checking out the shopping cart or exiting the shopping without buying. Its exact layout should have the same format as shown below.

```
-------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
-------------------------------------------------------------
```

To display the above two menus, you should implement the `topMenu()` and `subMenu()` methods in the `GroceryStore` class accordingly and use them in the `start()` method.

A user's input is received from the keyboard and input into the program. Its exact layout should have the same format as shown below.

```
Please enter your choice:
```

If a user inputs an invalid menu option number, the program should detect this and continue asking for correct choice. Its exact layout should have the same format as shown below.

```
Please enter your choice: 5
Invalid option, please try again!
Please enter your choice:
```

To obtain the user's input choice, you should implement the `getChoice(int lower, int upper)` method in the `GroceryStore` class and use it in the `start()` method. Note that the input parameter variables '`lower`' and '`upper`' in the method represent the lower and upper bounds of the menu index numbers. If an input is outside this range, the method should continuously require a valid input.

The `start` method in the `GroceryStore` class manages the control flow between the two menus. The flows are described as follows.

- Option one of the top-level menu leads to displaying all the items on sale and then returning to the same top-level menu;
- Option two of the top-level menu leads to the second-level menu;
- Option three of the top-level menu leads to exiting from the program;
- Option one or two of the second-level menu leads to adding or removing an items to or from the shopping cart and then returning to the same second-level menu;
- Option 3 or 4 of the second-level menu leads to checking out or discarding the shopping cart and then returning back to the top-level menu.

An example is shown as follows.

```
==============================================================
------This is a simple grocery store program by abcd001.------
--------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
--------------------------------------------------------------
Please enter your choice: 2
--------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
--------------------------------------------------------------
Please enter your choice: 4
All items are cleared from the shopping cart.
--------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
--------------------------------------------------------------
Please enter your choice: 3
--------------------------------------------------------------
---------------Thank you for shopping with us!----------------
==============================================================
```

3. Display all the items on sale. `[10 marks]`

If the first option of the top-level menu, i.e., '`Show the list of items on sale`', is selected, the program should display the list of items on sale and then return to the main menu. Its exact layout should have the same format as shown below.

```
===============================================================
------This is a simple grocery store program by abcd001.------
---------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
---------------------------------------------------------------
Please enter your choice: 1
---------------------------------------------------------------
-----------------List of the items on sale--------------------
---------------------------------------------------------------
1001,Fresh toast bread white (700g),3.99,20
1002,Low-fat milk (2 liter),4.80,10
1003,V-energy drink,2.75,10
1004,Fresh garlic (450g),1.98,5
1005,Coca-Cola (300 ml),2.50,10
1006,Pineapple,3.60,6
1007,Mango,1.89,4
1008,Snickers chocolate bar,1.80,20
1009,Broccoli,1.47,11
1010,Washed Potato (2.5kg),2.98,7
1011,Good-morning cereal,5.6,10
1012,Rose apple (1.5kg bag),4.98,5
1013,Avocado (4pk),4.99,5
1014,Bananas (850g bag),2.96,4
1015,Kiwi fruit green (1kg),2.45,10
1016,Rock melon,7.98,2
1017,Lettuce,2.99,12
1018,Chocolate block (200g),3.59,10
---------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
---------------------------------------------------------------
Please enter your choice:
```

To display the items on sale, you should implement the `displayItems()` method in the `Stock` class and use it in the `start()` method of the `GroceryStore` class under the correct menu option.

Note that there is a `toString()` method defined in the `SalesItem` class that you can directly use, which prints out a `SalesItem` object with the same format as shown above. When implementing the `displayItems()` method, you basically need to loop through the array `items` starting from index 0 to `size`-1 and print out each element in the array one by one. Please also note that you should only display the items with a quantity value greater than zero. That is, if an item is out of stock (`quantity` is 0), it should not be listed as available for purchasing.

4. Add an item to the shopping cart. `[20 marks]`

If the first option of the second-level menu, i.e., '`Add an item to the shopping cart`, is selected, the program should allow the user to add an item to the shopping cart and then return to the same second-level menu. The user inputs the item code, and the program retrieves the item from the stock and adds it to the shopping cart. The program also displays the description of the item being added. Its exact layout should have the same format as shown below.

```
===============================================================
------This is a simple grocery store program by abcd001.------
---------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
```

```
---------------------------------------------------------------
Please enter your choice: 2
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
---------------------------------------------------------------
Please enter your choice: 1
---------------------------------------------------------------
Enter item code to buy: 1015
Item - Kiwi fruit green (1kg) is added to the shopping cart.
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
---------------------------------------------------------------
Please enter your choice:
```

To achieve the above task, you should implement the `findItem(String itemCode)` method in the `Stock` class and the `addItem(SalesItem item)` method in the `Cart` class and use them in the `start()` method of the `GroceryStore` class under the correct menu option.

The `findItem` method loops through the array of `items` and finds the item that has the same item code value and returns it (or returns null, if not found). If the item is found in stock, the program (in the `start` method) calls the `addItem` method in the `Cart` class to add the item into the shopping cart.

The `addItem` method appends this item to the end of the `purchases` array and increases the total number of items in the shopping cart by 1. Please also note that if an item is successfully added to the shopping cart, its `quantity` value should be decreased by 1. This means that one of such item is sold and the available amount in quantity is one less.

In the case of the entered item code cannot be matched to an item in the stock, the program (in the `start` method) should give an error message and continue with another prompt for typing an item code. For example, if the user enters the item code "4627", another prompt for typing an item code should be shown. Note that your program should have the exact same input and output format as below.

```
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
---------------------------------------------------------------
Please enter your choice: 1
---------------------------------------------------------------
Enter item code to buy: 4627
Invalid item code, please try again!
Enter item code to buy:
```

Apart from the above item code not found case, there is another situation where an adding of item to the shopping cart cannot be succeed. That is, if an item is found by the code and however its quantity value is zero, the program should display the item is out of stock. Its exact layout should have the same format as shown below.

```
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
```

```
---------------------------------------------------------------
Please enter your choice: 1
---------------------------------------------------------------
Enter item code to buy: 1016
Sorry, item 1016 is out of stock. Please select a different item.
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
---------------------------------------------------------------
Please enter your choice:
```

Note that in the above case, after trying to add an out of stock item to the shopping cart, the control flow goes back to the second-level menu.

5. Remove an item from the shopping cart. [10 marks]

If the second option of the second-level menu, i.e., '`Delete an item from the shopping cart`', is selected, the program should allow the user to remove an item from the shopping cart and then return to the same second-level menu. The user inputs the item code, and the program retrieves the item from the shopping cart and deletes it. The program also displays the description of the item being deleted. Its exact layout should have the same format as shown below.

```
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
---------------------------------------------------------------
Please enter your choice: 2
---------------------------------------------------------------
Enter item code to delete: 1004
Item - Fresh garlic (450g) is removed from the shopping cart.
---------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
---------------------------------------------------------------
Please enter your choice:
```

To achieve the above task, you should implement the `deleteItem(String itemCode)` method in the `Cart` class and use it in the `start()` method of the `GroceryStore` class under the correct menu option.

The `deleteItem` method loops through the array of `purchases` and finds the item that has the same item code value and deletes it (or report an error message, if not found). If an item is removed from the shopping cart, the `quantity` value of the item needs to be increased by 1. This indicates that a purchase has been reversed and one more such item is now available on sale.

For removing an item from the `purchases` array, it can be effective done by shifting all the elements on the right of the item (to be deleted, say it's at index `i`) to the left by 1. This can be easily achieved by using a `for` loop structure with a starting index from `i+1` to `size`-1. After deletion, the total number of elements in the `purchases` array indicated by the variable `size` should be decreased by 1.

In the case of the entered item code cannot be matched to an item in the shopping cart, the program should report an error message. Note that your program should have the exact same input and output format as in the following.

```
--------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
--------------------------------------------------------------
Please enter your choice: 2
--------------------------------------------------------------
Enter item code to delete: 1002
This item is not in the shopping cart.
--------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
--------------------------------------------------------------
Please enter your choice:
```

6. Clear all the items in the shopping cart. `[7 marks]`

If the fourth option of the second-level menu, i.e., '`Exit without buying`', is selected, the program should allow the user to remove all the items from the shopping cart without buying and then return back to the top-level menu. Its exact layout should have the same format as shown below.

```
--------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
--------------------------------------------------------------
Please enter your choice: 4
All items are cleared from the shopping cart.
--------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
--------------------------------------------------------------
Please enter your choice:
```

To achieve the above task, you should implement the `deleteAll()` method in the `Cart` class and use it in the `start()` method of the `GroceryStore` class under the correct menu option.

The `deleteAll` method loops through the array of `purchases` and deletes each element. It also sets the total numbers of items in the shopping cart to zero. For each item removed from the shopping cart, the `quantity` value of the item needs to be increased by 1. This indicates that a purchase has been reversed and one more such item is now available on sale.

7. Checkout items in the shopping cart. `[8 marks]`

If the third option of the second-level menu, i.e., '`Check out the shopping cart`', is selected, the program should print out the bill with all the items in the shopping cart and the total amount due and then return back to the top-level menu. Its exact layout should have the same format as shown below.

```
--------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
```

```
--------------------------------------------------------------
Please enter your choice: 3
--------------------------------------------------------------
Checking out items ...
-----------------------------Bill-----------------------------
Low-fat milk (2 liter)/$4.80
Fresh garlic (450g)/$1.98
Rose apple (1.5kg bag)/$4.98
--------------------------------------------------------------
Amount due: $11.76
--------------------------------------------------------------
1. Show the list of items on sale.
2. Start to shop online.
3. Exit the system.
--------------------------------------------------------------
Please enter your choice:
```

To achieve the above task, you should implement the `checkOut()` method in the `Cart` class and use it in the `start()` method of the `GroceryStore` class under the correct menu option.

The `checkout` method will go through the `purchases` array and output the items bought along with their price. It will also calculate the sum of purchases so far and at the end of the checkout print out the total amount due. To format a double variable, e.g., `amountDue`, as money, you could use `"$"+ new DecimalFormat("0.00").format(amountDue).`

SUMMARY OF SUBMISSION INSTRUCTIONS:

You should submit the following files for this assignment through the web dropbox at (https://adb.ec.auckland.ac.nz/adb/).

- `Application.java` - the Java application
- `GroceryStore.java` – the online shopping program
- `Stock.java` – the class that represents the stock of items
- `Cart.java` - the class that represents the shopping cart
- `SalesItem.java` – the class that represents the information of a sale item
- `Keyboard.java` - the class to read input from the Keyboard
- `A1.txt` – a text file containing your feedback on the assignment (see below).

```
CompSci 101 S1 2013
===================
Feedback on Assignment One
--------------------------
Name:
Login:
ID:

All the work done in this assignment is my own work.

How much time did the assignment take overall?

What areas of the assignment did you find easy?

What areas of the assignment did you find difficult?

Which topics of the course did the assignment most help you
understand?

Any other comments:
```

Summary of marking details

Your UPI must appear in the output. If it does not appear in the output, you will forfeit the marks for that question.

The Online Shopping Program	***100 Marks***
You must submit the files `Application.java, GroceryStore.java, Stock.java, Cart.java, SalesItem.java, Keyboard.java, A1.txt.`	
Documentation/ Neatness/ Style – e.g., comment at top of the program, indentation, variable and method naming conventions, use of methods to structure program.	10
Defining a constant that represents your UPI and using it in the output of the program.	5
Correctly implement the `topMenu` method in the `GroceryStore` class, which displays the main menu.	5
Correctly implement the `subMenu` method in the `GroceryStore` class, which displays the second-level menu.	5
Correctly implement the `getChoice` method in the `GroceryStore` class, which reads user's input options with error checking.	10
Implement the `start` method in the `GroceryStore` class, which manages the top and sub-level menus with correct control flows.	5
Correctly implement the `displayItems` method in the `Stock` class, which prints out all the item on sale. And call the method in the `start` method of the `GroceryStore` class under the correct menu option.	10
Correctly implement the `findItem` method in the `Stock` class, which locates an item on sale based on the item code. And call the method in the `start` method of the `GroceryStore` class under the correct menu option.	10
Correctly implement the `addItem` method in the `Cart` class, which adds an item to the shopping cart. And call the method in the `start` method of the `GroceryStore` class under the correct menu option.	10
Correctly implement the `deleteItem` method in the `Cart` class, which removes an item from the shopping cart. And call the method in the `start` method of the `GroceryStore` class under the correct menu option.	10
Correctly implement the `deleteAll` method in the `Cart` class, which removes all the items in the shopping cart. And call the method in the `start` method of the `GroceryStore` class under the correct menu option.	7
Correctly implement the `checkOut` method in the `Cart` class, which prints out the items bought with prices and calculates the total amount due. And call the method in the `start` method of the `GroceryStore` class under the correct menu option.	8
Submit your completed feedback for asssignment 1 in `A1.txt`.	5