**Cart System**

**[Strictly adhere to the object oriented programming specifications given in the problem statement. Template code is provided to ease the input output process. Template code will not compile. You need to fill in the missing code.]**

**Business Requirement:**

Your task is to create a basic Cart System where an App system reads from a file of default items and displays those items to the user in order to add them to their cart. A new item can be added at any time to the system (Only at runtime and it does not need to be saved in the file with default items). Items can be removed from the System as well as from the cart.

**Work-Flow:**

The Cart System starts by asking the user to select whether they would like to:

1. Add an item to the System: In this case, the system asks the user to enter the new item's name, a description for the new item, the new item’s price and the available quantity of the new item, finally the new item will be added to the app.
2. Add an item to the Cart: In this case, the system displays all of the available items in the app system and asks the user to enter the name of an item to be added to the cart. The item will be added to the cart by searching the name for that item in the collection of items available in the app system. If the item has already been added to the cart, the quantity of that item in the cart is increased by one.
3. Display Cart: All the items in the cart are displayed, the system then calculates the sub-total by calculating the total sum of the products of the price and quantity of each item in the cart (Sum (item's price \* item's quantity)). The sub-total is displayed along with the tax, which is (sub-total \* 0.05), and the total, which is (sub-total + tax) for all items in the cart.
4. Display System: The system displays all the items that are available in the app.
5. Remove an item from the cart: In this case, the system asks the user for the name of the item to be removed from the cart.
6. Remove an item from the system: In this case, the system asks the user for the name of the item to be removed from the app. If an item is removed from the app, but it was already added to the cart. It will also be removed from the cart.
7. Quit: This option terminates the program.

**Requirement 1:**

**Model Class:**

A class **Item** should contain the private instance variables specified in ***TABLE 1*** with associated **GETTERS** and **SETTERS**.

The purpose of the item class is to carry data related to an item.

***TABLE 1:***

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| String | itemName | Name of the item |
| String | itemDesc | Contains the item's description |
| Double | itemPrice | Contains a double value which represents the item’s price |
| Integer | quantity | Represents the number of items the user has added to the cart |
| Integer | availableQuantity | Represents the number of items available in the system |

The following constructor must be implemented:

|  |  |
| --- | --- |
| **Description** | **Input Parameters** |
| This constructor initializes quantity to 1 | None |

**Requirement 2:**

**sample.txt and TheSystem:**

sample.txt contains data in “double space” separated value format.

A class **TheSystem** should contain the private instance variable specified in ***TABLE 2*** with associated **GETTER** and **SETTER**.

The purpose of the **TheSystem** class is to maintain the list of items and the main logic of the system that is similar in the app and cart classes.

***TABLE 2:***

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| HashMap<String, Item> | itemCollection | Provides the list of items in the system or the cart depending on which class initiates it |

The following constructor must be implemented:

|  |  |
| --- | --- |
| **Description** | **Input Parameters** |
| This constructor initializes the itemCollection variable with an empty hashmap.  It then checks if the AppSystem is invoking the constructor (getClass().getSimpleName().equals("AppSystem")),  if so, it adds the items from the sample.txt file to the itemCollection.    Recommended: When reading from the sample.txt file,  read each line and the do the following line: String[] itemInfo = line.split("\s "); | None |

The following methods must be implemented:

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Description** | **Input Parameters** |
| Boolean | **checkAvailability()** – This method takes Item object as a parameter, then it checks if the item.getQuantity() is greater than or equal to the item.getAvailableQuatity(). If it is, display the following message.  "System is unable to add [item name] to the card. System only has [item available quantity] [item name]s." and return false. Otherwise, return true. | Item item |
| Boolean | **add()** – This method takes Item object as a parameter, checks:  1. if item is null, then returns false  2. If it is already in the collection and is available.  If so, the method increases the quantity by one and returns true.  3. If the item is not in the collection, the method adds the item to the collection and also returns true.  4. In all other cases, the method returns false. | Item item |
| Item | **remove()** – This method takes String itemName as a parameter, checks if the item is in the collection, if it is, then removes it and returns the Item object being removed. If is not in the collection then returns null. | String itemName |

**Requirement 3:**

**AppSystem:**

The purpose of the **AppSystem** class is to implement the logic related only to the App system.

The following methods must be implemented:

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Description** | **Input Parameters** |
| void | **display()** – This method takes no parameter and displays every item in the App system. | None |
| Boolean | **add()** – This method takes Item object as a parameter, checks:  1. if item is null, then returns false  2. If it is already in the collection, displays a message  "[Item name] is already in the App System" and returns false.  3. If the item is not in the collection, the method adds the item to the collection and returns true. | Item item |

**Requirement 4:**

**CartSystem:**

The purpose of the **CartSystem** class is to implement the logic related only to the cart.

The following methods must be implemented:

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Description** | **Input Parameters** |
| void | **display()** – This method takas no parameter and displays every item in the Cart system, along with the sub-total, tax, and total:  *sub-total* = the total sum of the products of the price and quantity of each item in the cart (Sum (item's price \* item's quantity)).  *tax* = sub-total \* 0.05  *total* = sub-total + tax | None |

**Optional Bonus Requirement 5:**

When an item is removed from the App system, automatically remove it also from the Cart if it is there.

**MainEntryPoint class**

This class provides the complete logic that makes every component work together. Take the time to review the logic provided and make sure you fully understand what is being implemented.

Follow the naming convention provided by this document and do not change the name or return type of any of the methods provided throughout the program. Do not change the name of any of the classes provided throughout this program.

Feel free to create any helper methods / variables if needed.

Try to align the outputs in the console so they are more user friendly.

**Sample Output:**

* **Display Cart – Cart initially empty**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

3

Item Name Item Description Item Price Available Quantity

Sub Total: 0.0

Tax: 0.0

Total: 0.0

* **Display System – Initially data**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

4

Item Name Item Description Item Price Available Quantity

1 pizza very cheesy 12.3 3

2 salad cobb salad 15.50 12

3 hunger burger huge patty 9.49 10

4 fried chicken so crispy 18.99 5

* **Add new item to System**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

1

Enter the item name:

orange juice

Enter a description for the item:

fresh

Enter the item's price:

4.99

Enter the quantity available in the System:

15

orange juice

Item successfully added

* **Add new item to Cart**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

2

Item Name Item Description Item Price Available Quantity

1 pizza very cheesy 12.3 3

2 salad cobb salad 15.50 12

3 hunger burger huge patty 9.49 10

4 fried chicken so crispy 18.99 5

5 orange juice fresh 4.99 15

Enter the name of the item  
orange juice

Item successfully added

* **Remove item from System**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

6

Item Name Item Description Item Price Available Quantity

1 pizza very cheesy 12.3 3

2 salad cobb salad 15.50 12

3 hunger burger huge patty 9.49 10

4 fried chicken so crispy 18.99 5

5 orange juice fresh 4.99 15

Enter the name of the item  
orange juice

orange juice was removed from the System

* **Remove item from Cart**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

5

Item Name Item Description Item Price Available Quantity

orange juice fresh 4.99 1

Sub Total: 4.99

Tax: 0.25

Total: 5.24

Enter the name of the item

orange juice

orange juice was removed from the cart

* **Quit Program**

Choose an action:

1. Add item to System

2. Add item to Cart

3. Display Cart

4. Display System

5. Remove item from Cart

6. Remove item from System

7. Quit

7

Byyyeee!!