**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 the App system reads in from a file default items and are display to the user to add them to the 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 remove 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 item to the System: In this case, the system will require information from the user such as: Enter the item name, Enter a description for the item, Enter the item’s price and Enter the quantity available in the system.
2. Add item to the cart: In this case, the system displays all of the items in the app system and will require the user to “Enter the name of the item”. The item will be added to the cart by search for that item in the item list the app system has available. If the item is already in the cart than the quantity is increase by one.
3. Display Cart: All the items in the cart are displayed, alone with the sub-total of the item, tax which is the sub-total \* 0.05, and the total which is the sub-total + tax
4. Display System: displays all the items that are available in the system
5. Remove item from cart: In this case, the system will ask the user for the name of the item they want to remove
6. Remove item from system: In this case, the system will ask the user for the name of the item they want to remove
7. Quit: This option terminates the program

**Requirement 1:**

**Model Class:**

In the given package, create a class item with the private member variables specified in **TABLE 1**. These private members must have **GETTERS** and **SETTERS** methods.

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

**TABLE 1:**

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

The following constructor must be implemented:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Class Name** | **Constructor Name** | **Input Parameters** |
| 1. | Item | public Item () – This constructor initializes the quantity to 1 | None |

**Requirement 2:**

**Sample.txt and TheSystem:**

**Take a second to look a sample.txt; Notice that there is a single white space for items that have a name consist of two words and there is two white space to separate the data of one item.**

In the given package, create a class **TheSystem** with the private member variable specified in **TABLE 2**. This private member must have default **SETTER** method and a custom **GETTER** method.

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.

|  |  |  |
| --- | --- | --- |
| **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:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Class Name** | **Constructor Name** | **Input Parameters** |
| 1. | TheSystem | public TheSystem () – This constructor initializes the itemCollection member field. It then check if the appsystem is invoking the constructor (getClass().getSimpleName().equals("AppSystem")), if it is true then the itemCollection is feed in what the sample.txt file contains. Otherwise, it would be initialize and just be empty. Recommended: When reading from the sample.txt file, read each line and the do the following line: String[] lines = line.split(“\\s “); This would the string where it sees more the one white space. | None |

The following methods must be implemented:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Class Name** | **Method Name** | **Input Parameters** |
| 1 | HashMap<String, Item> | TheSystem | getItemCollection() – This method would create a new HashMap<String, Item>, then it adds every key and item in the original itemCollection to the new HashMap<String, Item>. Finally, return the new HashMap<String, Item>, | None |
| 2 | void | TheSystem | setItemCollection() – this method takes a HashMap as a parameter and assigns it to the original itemCollection | HashMap<String, Item> cols |
| 3 | Boolean | TheSystem | checkAvailability() – This method takes as parameter an Item object and a Integer which represents how many times a single item has been added to the cart. Then it checks if the item.getQuantity() + current is greater than the item.getAvailableQuatity(). If it is, then display the following message. “System is unable to add [item’s quantity] [item’s name]”. “System can only add [calculate available item] [item’s name]. Finally return false. Otherwise return true. | Item item, Integer current |
| 4 | Boolean | TheSystem | add() – This method takes as a parameter Item object and checks if the item is already in the collection. If it is, then check if there is available item, otherwise don’t add the item. If the item is not in the collection, just add it to the collection. | Item item |
| 5 | Item | TheSystem | Remove() – This method takes as a parameter the item name to be remove. First check if the item is in the collection, if it is, then remove it and return the Item object being remove. If is not in the collection then return null | String itemName |

**Requirement 3:**

**AppSystem:**

In the given package, create a class **AppSystem**.

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

The following methods must be implemented:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Class Name** | **Method Name** | **Input Parameters** |
| 1 | void | AppSystem | display() – This method takes no parameter and it would display every item in the App system. | None |
| 2 | Boolean | AppSystem | add() – This method takes Item Object as a parameters. It checks if the item is already in the system. If it is, then display a message “Item [Item’s system] is already in the system” and return false. If is not then add it and return true. | Item item |

**Requirement 4:**

**CartSystem:**

In the given package, create a class **CartSystem**.

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

The following methods must be implemented:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Class Name** | **Method Name** | **Input Parameters** |
| 1 | void | CartSystem | display() – This method takes no parameter and it would display every item in the Cart system, alone with the sub-total of the item, tax which is the sub-total \* 0.05, and the total which is the sub-total + tax | None |

**Optional Bonus Requirement 5:**

If an item is removed from the System, but it was already added to the cart. It should also be removed from the cart.

**MainEntryPoint class**

This class provides the complete logic that makes every component work together. Take the time to review the logic provide 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.

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 Really Good 12.3 20

2 Fried Chicken Fried Really Good 12.3 20

3 Hunger Burger Really Good 12.3 20

4 Salad Really Good 15.5 30

* **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:

Jugo

Enter a description for the item:

bueno

Enter the item's price:

10

Enter the quantity available in the System:

20

Jugo

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 Really Good 12.3 20

2 Jugo bueno 10.0 20

3 Fried Chicken Fried Really Good 12.3 20

4 Hunger Burger Really Good 12.3 20

5 Salad Really Good 15.5 30Enter the name of the item

Jugo

* **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 Really Good 12.3 20

2 Jugo bueno 10.0 20

3 Fried Chicken Fried Really Good 12.3 20

4 Hunger Burger Really Good 12.3 20

5 Salad Really Good 15.5 30Enter the name of the item

Jugo

* **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

Jugo bueno 10.0 1

Sub Total: 10.0

Tax: 0.5

Total: 10.5

Enter the name of the item

Jugo

* **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!!