# 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 will require 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 in the system.

2. Add an item to the cart: In this case, the system displays all of the available items in the app system and will require 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 list 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 will then calculate the of the sub-total by adding the total of the product of the price of each item time quantity of that particular item in the cart ( Sum(item's price * items quantity)).The sub-total should be displayed along with the tax, which is the sub-total * 0.05, and the total, which is the sub-total + tax for all items in the cart.
4. Display System: displays all the items that are available in the system
5. Remove an item from the cart: In this case, the system will ask the user for the name of the item they want to remove
6. Remove an item from the system: In this case, the system will ask the user for the name of the item they want to remove. f an item is removed from the System, 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:

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.

| 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:

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

**Requirement 2:**

**Sample.txt and TheSystem:**

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

In the given package, create a class called **TheSystem** with the private member variables specified in TABLE 2. These private members must have default **SETTER** methods and custom **GETTER** methods.

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.<br>It then checks if an instance of the class AppSystem | None |

| | | | |
|---|---|---|---|
| | | is invoking the constructor<br>by using<br>(getClass().getSimpleName().equals("AppSystem"))<br>. If an instance of the class AppSystem is the one invoking the constructor, then the itemCollection should be filled with the items contained in the sample.txt file. Otherwise, itemCollection should be initialized and just be empty.<br><br>Recommended: When reading from the sample.txt file, read each line and the use the following line: String[] lines = line.split("\\s "); This will split the line where it sees more the one white space. | |

| No. | Return Type | Class Name | Method Name | Input Parameters |
|---|---|---|---|---|
| 1 | HashMap<String, Item> | TheSystem | getItemCollection()– This method should create a new HashMap<String, Item>, then add every key and item in the original itemCollection to a new HashMap<String, Item>. Finally, it should return the new HashMap<String, Item> | None |
| 2 | void | TheSystem | setItemCollection() – this method should take a HashMap as a parameter and assigns it to the original itemCollection | HashMap<String, Item> cols |
| 3 | Boolean | TheSystem | checkAvailability() – This method takes Item object and an Integer (current) as its parameters, the integer represents how many times a single item has been added to the cart.Then it checks if the item's quantity (item.getQuantity() )+ current is greater than the item's available quantity (item.getAvailableQuatity()); If it is, then display the following message. "System is unable to | Item item, Integer current |

| | | | | |
|---|---|---|---|---|
| | | | add [item.getQuantity() + current] [item's name]". "System can only add [item's Quantity] [item's name]. Otherwise, if if the item's quantity (item.getQuantity() )+ current is lower than the item's available quantity (item.getAvailableQuatity()), return true. | |
| 4 | Boolean | TheSystem | add() – If it is already in the collection, then check if the item is available, If it is available, increase the quantity by one, otherwise, don't add the item. If the item is not already in the collection, just add it to the collection. | Item item |
| 5 | Item | TheSystem | Remove() – This method takes as a parameter the name of the item to be removed. First check if the item is in the collection, if it is, then removes 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 the App system.

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 should display every item in the App system. | None |
| 2 | Boolean | AppSystem | add() – This method takes an Item Object as a parameter. It checks if the item is already in the system. If it is, then display a message | Item item |

| | | | | |
|---|---|---|---|---|
| | | | "[Item's name] is already in the [name of the class calling this method]" and return false. If is not then add it and return true | |

**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 the cart.

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 displays every item in the Cart system, along with the subtotal(the sum of the [item's price * quantity] of the items in them cart), the tax which is the subtotal * 0.05, and the total which is the subtotal + 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 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.

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

Jugo 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

Jugo bueno 10.0 1
Sub Total: 10.0
Tax: 0.5
Total: 10.5
Enter the name of the item
Jugo

Jugo 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!!