James Macartney

Project Report

Description:

For my project, the goal was to design and implement an inventory management system specifically for a restaurant, which included items like food and beverages. The project focused on using OOP principles, including key concepts such as Encapsulation and Inheritance. I developed three main classes: “Item,” “Inventory,” and “Report,” each responsible for a different aspect of the system. The “Item” class managed the categorization and naming of different products, while the “Inventory” class handled adding and removing items from stock. The “Report” class was used for additional functionality and tracking.

Instructions:

The program consists of three main classes: Inventory, Item, and Report, each with distinct roles and some inheriting from one another. The program also categorizes items into types like Food, Beverages, etc. You can view these categories by using the ‘list_items’ method from the Inventory class. To interact with the Inventory class, you need to create an object, which will allow you to access any of the four available methods. Each method requires a specific number of arguments to function. For example, the `add_new_item` and `remove_existing_item` methods require two arguments, while the `update_inventory` method requires one.

Similarly, the Item class follows a comparable structure, but with the added feature of private class attributes. Each method in this class also requires a certain number of arguments. The same approach is applied to the Report class.

Structure:

Class Inventory; no attributes

1. ***add-item method:***

2 arguments; item and category which needed to be added.

2. ***remove-item method:***

2 arguments; item and category which needed to be removed from

3. ***List_items method:***

No arguments; prints inventory items

4. ***Class Item:***

2 class attributes; item and category which needed to be added.

5. ***Set_catagory method:***

No arguments; prints category of object which is created from Item class.

6. ***Get_details method:***

No arguments; prints name of object which is created from Item class.

7. ***Get_details method:***

No arguments; prints name of object which is created from Item class.

Class Report; no attributes

1. Items_in method:

1 argument, the category which you want to check items for; prints out the # of items in a category.

2. Recipes method:

No arguments; prints a list of the recipes

3. Add_new_recipe method:

2 arguments; name and items of recipes prints the created recipes

Overview of Developed Classes:

Inventory Class:

The `Inventory` class manages the main functions of the inventory system, including adding, removing, and listing items. It organizes items into categories such as "foods" and "beverages." The class also includes a method, `update_inventory`, which allows the creation of a new category in the inventory.

Item Class:

The `Item` class deals with individual items and their attributes. It has three private variables—name, category, and expiration date—that cannot be changed directly by users of the program. The class contains methods to set these attributes, specifically `set_item_name`, `set_item_category`, and `set_item_expiry`, which assign values to the private variables.

Report Class:

The `Report` class provides details about the inventory. For example, the `check_item_count` method displays the number of items in a specific category, while the `list_recipes` method lists the available recipes. Additionally, the `add_new_recipe` method allows users to create new recipes by passing a list of ingredients.

Verification of Sanity of Code

Scenario 1: Adding a new item to a category

To verify the functionality of adding a new item to a category, we use the `add_item` method in the `InventoryManager` class. This method takes two arguments: the item you want to add and the category to which the item should be added.

For example, let's add a new item called 'smoothie' under the 'Beverages' category. The process involves calling the `add_item` method, and the system will check if the item already exists in the category. If not, it adds the item to the correct list (in this case, the `beverages_category` list).

```
24      # Adds a new item to the desired category
25      def add_item(self, item, category):
26          if category == 'Beverages' and item not in beverages_category:
27              beverages_category.append(item)
28              print(f"Item '{item}' added to Beverages category.")
29          elif category == 'Foods' and item not in food_category:
30              food_category.append(item)
31              print(f"Item '{item}' added to Foods category.")
32          else:
33              print("Item already exists or invalid category.")
```

```
>>> %Run inventoryjjm.py

  Welcome to the Inventory
  Before adding 'smoothie':
  Foods: ['Bread', 'Rice', 'Pasta', 'Chicken', 'Beef', 'Pork', 'Fish', 'Shrimp', 'Lamb', 'Eggs', 'Cheese', 'Milk', 'Yogurt', 'Butter', 'Olive Oil', 'Tomatoes', 'Lettuce',
  'Spinach', 'Carrots', 'Potatoes', 'Onions', 'Garlic', 'Peppers', 'Cucumbers', 'Apples', 'Bananas', 'Oranges', 'Strawberries', 'Grapes', 'Mangoes', 'Watermelon']
  Beverages: ['Water', 'Coca-Cola', 'Pepsi', 'Sprite', 'Fanta', 'Orange Juice', 'Apple Juice', 'Lemonade', 'Iced Tea', 'Hot Tea', 'Coffee', 'Espresso', 'Cappuccino', 'Lat
  te', 'Mocha', 'Hot Chocolate', 'Milk', 'Almond Milk', 'Soy Milk', 'Coconut Water', 'Energy Drink', 'Gatorade', 'Beer', 'Wine', 'Smoothie']
  Item 'smoothie' added to Beverages category.

  After adding 'smoothie':
  Foods: ['Bread', 'Rice', 'Pasta', 'Chicken', 'Beef', 'Pork', 'Fish', 'Shrimp', 'Lamb', 'Eggs', 'Cheese', 'Milk', 'Yogurt', 'Butter', 'Olive Oil', 'Tomatoes', 'Lettuce',
  'Spinach', 'Carrots', 'Potatoes', 'Onions', 'Garlic', 'Peppers', 'Cucumbers', 'Apples', 'Bananas', 'Oranges', 'Strawberries', 'Grapes', 'Mangoes', 'Watermelon']
  Beverages: ['Water', 'Coca-Cola', 'Pepsi', 'Sprite', 'Fanta', 'Orange Juice', 'Apple Juice', 'Lemonade', 'Iced Tea', 'Hot Tea', 'Coffee', 'Espresso', 'Cappuccino', 'Lat
  te', 'Mocha', 'Hot Chocolate', 'Milk', 'Almond Milk', 'Soy Milk', 'Coconut Water', 'Energy Drink', 'Gatorade', 'Beer', 'Wine', 'Smoothie', 'smoothie']
```

Scenario 2: Checking the Number of Items in a Category

To check the number of items in a specific category, I used the `check_item_count` method from the `Report` class. This method requires one argument: the category you want to check. For example, if I want to know how many items are in the 'Beverages' category, I call this method with 'Beverages' as the argument. The output will then display the current number of items in that category. Keep in mind that this number can change if you add or remove items later.

```
# Method to check the number of items in a category
def check_item_count(self, category):
    if category == 'Foods':
        count = len(food_items)
    elif category == 'Beverages':
        count = len(beverages_items)
    print(f"There are {count} items in the '{category}' category.")
    print("You may add or remove items as needed.")
```

Shell

```
>>> %Run inventoryjjm.py

  Inventory Management System Initialized
  'Smoothie' is already in the selected category or invalid category.

  Checking the number of items in the 'Beverages' category:
  There are 25 items in the 'Beverages' category.
  You may add or remove items as needed.

  Checking the number of items in the 'Foods' category:
  There are 31 items in the 'Foods' category.
  You may add or remove items as needed.
```

Conclusion

During this project, I faced the challenge of choosing the right data structure for managing inventory items. I decided to use lists because they are easy to work with and offer helpful built-in functions like `append`*, which makes adding items from the CSV files straightforward.*

Using Object-Oriented Programming (OOP) was a great way to organize the project. By applying encapsulation, I ensured that the item details (like name and category) couldn't be altered by users, keeping the data secure and consistent. This made the program easier to maintain.

Overall, this project helped me improve my understanding of OOP, and I learned how to structure programs using classes and objects. In the future, I could improve it by adding user input or more classes for different features.