Course Project- IT 209
Simulating a Grocery Cart

Description

We will design a simple user cart for a grocery cart as you see in any online shopping. The purpose of this project is to demonstrate an acceptable level of expertise with the fundamental procedural and object-oriented concepts and Graphical User Interface (GUI) implementation techniques introduced and refined in lectures and labs during the course of the semester.

Recall that the acceptable resources for this assignment differs from those approved for lab assignments, and are limited to the class text, Python Library, Language and Tutorial references, lecture and lab slides/notes, lecture and lab instructors.

Please note that this is an individual project. Your final submissions will be checked for plagiarism and identified projects will receive a Zero automatically. The completed project must be submitted via Blackboard before the specified due date. If you have questions, use office hour from Instructor/GTA to obtain assistance.

Project Requirements

- The source code must be submitted in files named using the accepted format.
- The source code files must contain a file header using the accepted format. The header information must be complete and accurate.
- The source code file should use self-documenting code and additional comments (as required) to improve code readability and should use appropriate method and function header comments.

Project Submission

You need to submit the following files:

1. A class file containing all the class and methods (please see classes and methods section). The name of the file should follow this format: **Firstname_lastname_Project_CLASS.py**

2. A GUI file containing the GUI applications and class implementation. The name of the file should be in the following format: **Firstname_lastname_Project_GUI.py**

Program Structure

1. Program starts by reading from an input file containing the name, product category, price and unit of each item. The input file contains the following information separated by "|"

Name| Category| Price| Unit

```
Butter, salted|Dairy|2.99|lb
Butter, whipped, with salt|Dairy|3.99|lb
Cheese, american|Dairy|4.99|lb
Cheese, cheddar|Dairy|5.99|lb
Egg, white|Dairy|1.99|dozen
Egg, brown|Dairy|2.99|dozen
```

2. Once read, an item will be created and saved to a list of items per category.
3. A testing code is provided to check the class file. You need to comment the testing code out once done checking the class file.
4. Create a GUI file that imports the class file and uses the category list, item, and cart object to display information to the user and handle different operations.
5. As shown in #3 in Figure-1, the user has the option to order items from different category. Once a category is selected (#2 in Figure-1), the user will be directed to the list of items available to purchase.
6. Each list, loads the items from the corresponding category list. If user selects Dairy, then all dairy products will be loaded from the *dairy_items* that was created in the class file.
7. For each product, the GUI must display the following information to the user (Figure-1):
   a) A check box to select the item to purchase
   b) A label to show the name of the item
   c) An entry to enter the quantity of the item
   d) Another label to display the unit
8. Once the required item(s) are selected, the user will click on the "Add to Cart" button, which will display a *subtotal* label with the total cost for all selected items.
9. The user can choose more items or click on the "Checkout" button.
10. After checkout, the GUI must display all purchased items, break-down of each item and their quantity and price, tax and total.
11. A description of CLASS and GUI part is as follows:

SmartCart Class

It is a dict subclass. A SmartCart object contains all the cart items and quantity as user orders. It works like a dictionary, where each **key** is an item object and **value** are the quantity user orders per object.

Methods:	Description	Parameters	Return type
subtotal: None	Returns subtotal from the **SmartCart** object. Iterate over all the **dict** values and multiply with the item price to obtain subtotal (before applying tax)	None	float

Item class

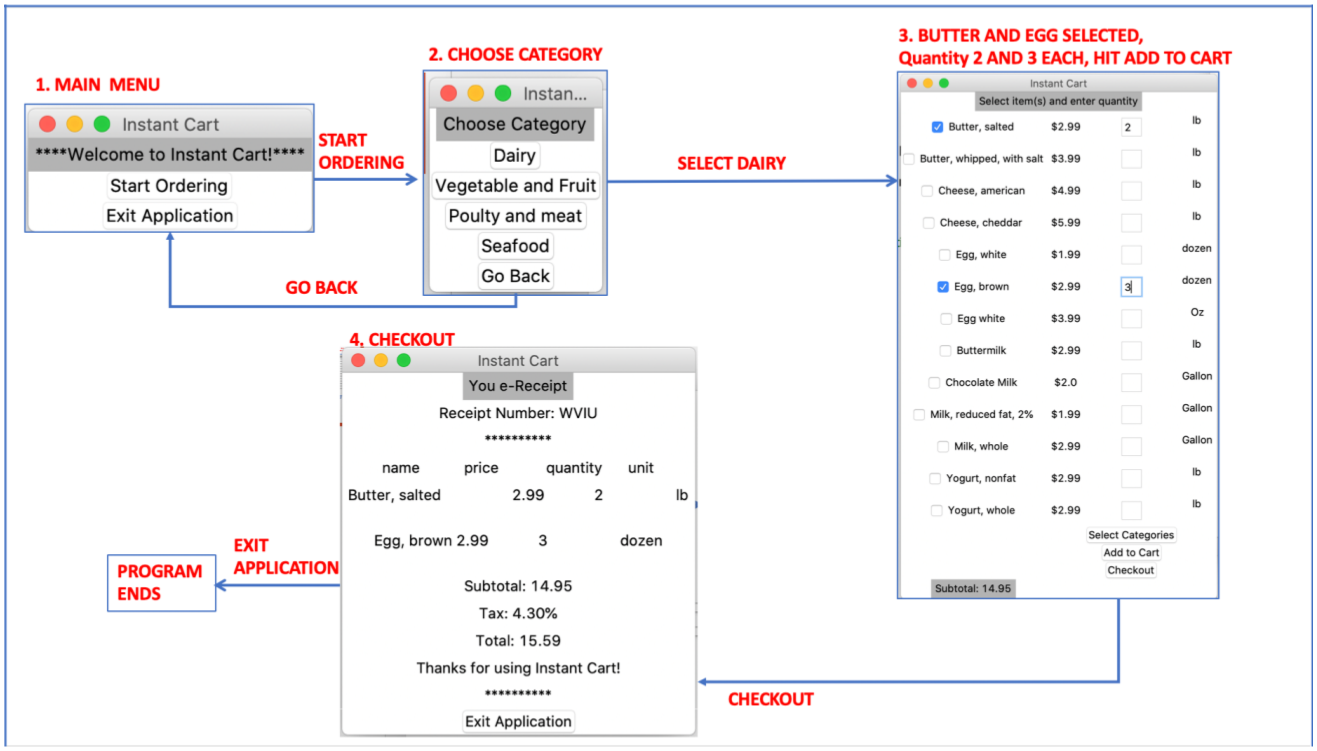
It contains four class variables to hold different types of items. Each of this class variable are simply python list to hold types of item objects. For this program you have dairy, vegetable and fruit, meat and seafood items.

Methods:	Description	Parameters	Return type
__init__()	Initialization method to initialize an item. Each item has four types of information: category, name, price and unit. See below for details**@@**. Depending on the item category, it will add it to the appropriate list.	Category, name, price, unit	None
get_category()	Returns category of an item	None	string
get_name()	Returns name of an item	None	string
get_unit()	Returns unit of an item. For example: some items are sold in pound while some per ounce etc.	None	string
get_price()	Returns prince of an item per unit	None	float
__str__()	String representation of an object. Returns category, name and unit.	None	string

@@

Attribute name	Description	type
category	Private attribute. Defines the category of an item. Valid categories: Dairy, Fruit and Vegetable, Poultry, Seafood	string
name	Private attribute. Name of an item	string
price	Private attribute. Price of an item per unit.	float
unit	Private attribute. Unit of an item. For example: lb, oz, each etc.	string.

Graphical User Interface



✓	Butter, salted	$2.99	2	lb
	Butter, whipped, with salt	$3.99		lb
	Cheese, american	$4.99		lb
	Cheese, cheddar	$5.99		lb
	Egg, white	$1.99		dozen
✓	Egg, brown	$2.99	3	dozen
	Egg white	$3.99		Oz
	Buttermilk	$2.99		lb
	Chocolate Milk	$2.0		Gallon
	Milk, reduced fat, 2%	$1.99		Gallon
	Milk, whole	$2.99		Gallon
	Yogurt, nonfat	$2.99		lb
	Yogurt, whole	$2.99		lb

```
Receipt Number: WVIU
**********
   name        price       quantity    unit
Butter, salted          2.99       2          lb

   Egg, brown 2.99        3              dozen

            Subtotal: 14.95
              Tax: 4.30%
             Total: 15.59
       Thanks for using Instant Cart!
              **********
```

***Figure-1**: Instant Cart – A simple grocery shopping. User needs to select an item and enter the quantity (#3) and hitting 'Add to Cart' button will add the items in the cart display the current subtotal. User can choose to select another category or checkout for final receipt.*

Description of the GUI Components

1. **Welcome screen (Main Menu):** This screen will be the first screen to display. It will contain following widgets:
   a. Welcome message: Label
   b. Start Ordering: Button – start the program, will take user to #2
   c. Exit Application: Button – exit the program

2. **Choose Category (Listed Categories):** If user presses start ordering then the program will display another frame with different categories of items. This window has following widgets:
   a. Choose Category: label
   b. Dairy: Button – will take to all dairy items (details in #3 below)
   c. Vegetable and Fruit: Button – will take to all vegetable and fruit items
   d. Poultry and Meat: Button – will take to all poultry and meat items
   e. Seafood: Button – will take to all seafood items
   f. Go Back: Button – will take to step 1 again

3. **Selected Category (e.g., Dairy):** User get to grocery by category of items. Selecting the category will take user to
   all the items available currently in the store. For example: Figure 1 shows all the dairy products available. This window contains the following widgets:
   a. Name of an item: check button – user selects which item they want to buy
   b. Price of an item: Label
   c. Quantity: Entry – user needs to enter quantity of an item
   d. Unit: Label – unit of an item, for example: lb, oz etc.
   e. Select Categories: Button - will take back to #2 Choose Categories
   f. Add to Cart: Button – Add selected items to cart i. Will display a current subtotal as label
   g. Checkout: Button – After adding to cart, user can directly go to checkout window (#4).

4. **Checkout:** Checkout window displays the summary of purchase with following widgets:
   a. Your e-receipt: Label
   b. Receipt Number: Label - Randomly generated by program
   c. Name Price Quantity Unit: Header Label
   d. Item purchased, price quantity, unit: Label
   e. Subtotal: Label
   f. Tax: Label
   g. Total: Label
   h. 'Thank you' message: Label
   i. Exit application: Button – exit the program

Grading Rubric

	Excellent	**Average**	**Needs Improving**	**Points**
Submission Details	File names and file header meet stated spec.	Either file name is incorrect or file header is missing sections/details.	Both file name and file header are missing or are incorrectly implemented.	___/ 5
Comments, Self-Documenting Code and Method headers (docstring)	Comments, self-documenting code and method headers clearly convey intent of code.	Comments, self-documenting code and method headers generally convey intent of code.	Comments, self-documenting code and method headers are largely missing or trivially implemented.	___/ 5
Class File	Class completely implements the attributes and behaviors required for the application.	Class partially implements the attributes and behaviors required for the application.	Class is not implemented or is trivially implemented (to the point everything is in one module)	___/ 10
GUI Class	Contains all specified GUI components and event model operations.	Contains GUI components and event model operations that significantly implement intended operations.	Contains some GUI components and event model operations but cart operation is incomplete.	___/ 10
Driver code	Correctly initializes application (using the appropriate classes).	Initializes application with minor issues.	Driver module not implemented or dysfunctional.	___/ 10
I/O	Specified I/O fully implemented.	Specified I/O is mostly implemented, with minor deficiencies.	I/O is significantly dysfunctional.	___/ 10
Overall Score				**___/50**