Object-Oriented Programming Lab#6, Fall 2023

Today's Topics

- Class/Object, Constructor,
- package
- Array (Reference Type)
- ArrayList

ArrayLIst:

Action	**Code**
Creating an ArrayList	ArrayList<T> list = new ArrayList<T>();
Adding element to arraylist	list.add(T);
Accessing an element	T t = List.get(int index)
Size of arraylist	int len = list.size();

Online Store- Problem Description

Develop an application for an online store which will help **a store owner** to keep the record of its items/items and run the business. The store contains different types of items e.g., **Food items, clothing, electronics** and many more. For simplicity, you can work with just 3 types of items as mentioned in previous line. Each **item** has some common characteristics e.g., ***name, id, category, price*** and ***quantity***. Like all other online stores, you need to implement the following functionalities in your application.

1. User can browse through the items – View all items.
2. User can view the list of items of specific category e.g., all food items, all clothing etc.
3. User can view the details of a specific item.
4. Add item to store
5. Sell an item

Implementation

Note: The following design is just one possible option. You can modify it according to your need. You are free to add additional attributes, methods.

Possible classes:

1. **Item (**under **store** package**)**

   - Private Attributes: **name, id, category, price, quantity**
   - Constructor
   - Method:
     - Getter/~~setter~~ method for all attributes
     - public double getSalePrice(double ***saleAmount***)
       - The parameter "***saleAmount***" is in percentage. From the method return the discounted price after sale. For example, if the **price** of an item is 100 and ***saleAmount*** is 20 (in percent) then this method will return 80. Note: This method will not change the **price** attribute.
     - Implement the toString() method and return

2. **Shop** (under **store** package)

   - Attributes: name, ArrayList<Item> **items** or Item array
   - Constructor
   - Method:
     - private Item/int findItem(String id)
       - Search for the item in the **items** array using the **id**. If the item is found, return the item. Return null if the item is not found.
     - public void addItem(String **name,** String **id,** String **category,** double **price,** int **count**)
       - Call findItem using the id. If the item is found increase the **quantity** attribute of the product by the **count** amount. If the item is not available in the list, create an **Item**

object using the parameters and add the object to the array/arraylist **items**.

- public void viewItem(String id)
  - Search for the item using **findItem** method. If the item is found, print the item.
- public void viewItems()
  - Access each of the item in items arraylist and print those items.
- public void sellAnItem(String id, int quantity)
  - Call findItem method. If the item is found in the list, decrease the **quantity**. If the item is not found, a message should display
- public double getSalePriceOfAnItem(String id, double saleAmount)
  - Call findItem. If the item is found in the list, call **getSalePrice(..)** using the item. If the item is not found, a message should display

3. ShopApp (under **store.app** package**)**

- Add Main method. Inside the main, create an object of **Shop** class and then provide the following **menu** and call appropriate method using the **Shop** object. Take input from users as needed.
  (a) 1 to view all
  (b) 2 to view a specific item
  (c) 3 to add item
  (d) 4 to sell item
  (e) 5 to see sale price of an item
  (f) 0 to exit