2019

ADVANCED SOFTWARE ENGINEERING

COURSEWORK - STAGE 1 REPORT

GROUP 8 MEMBERS – ABHISHEK RB, SANJNA RAJESH, NIKHITHA PANICKER, ONKAR GIROTRA

CONTENTS

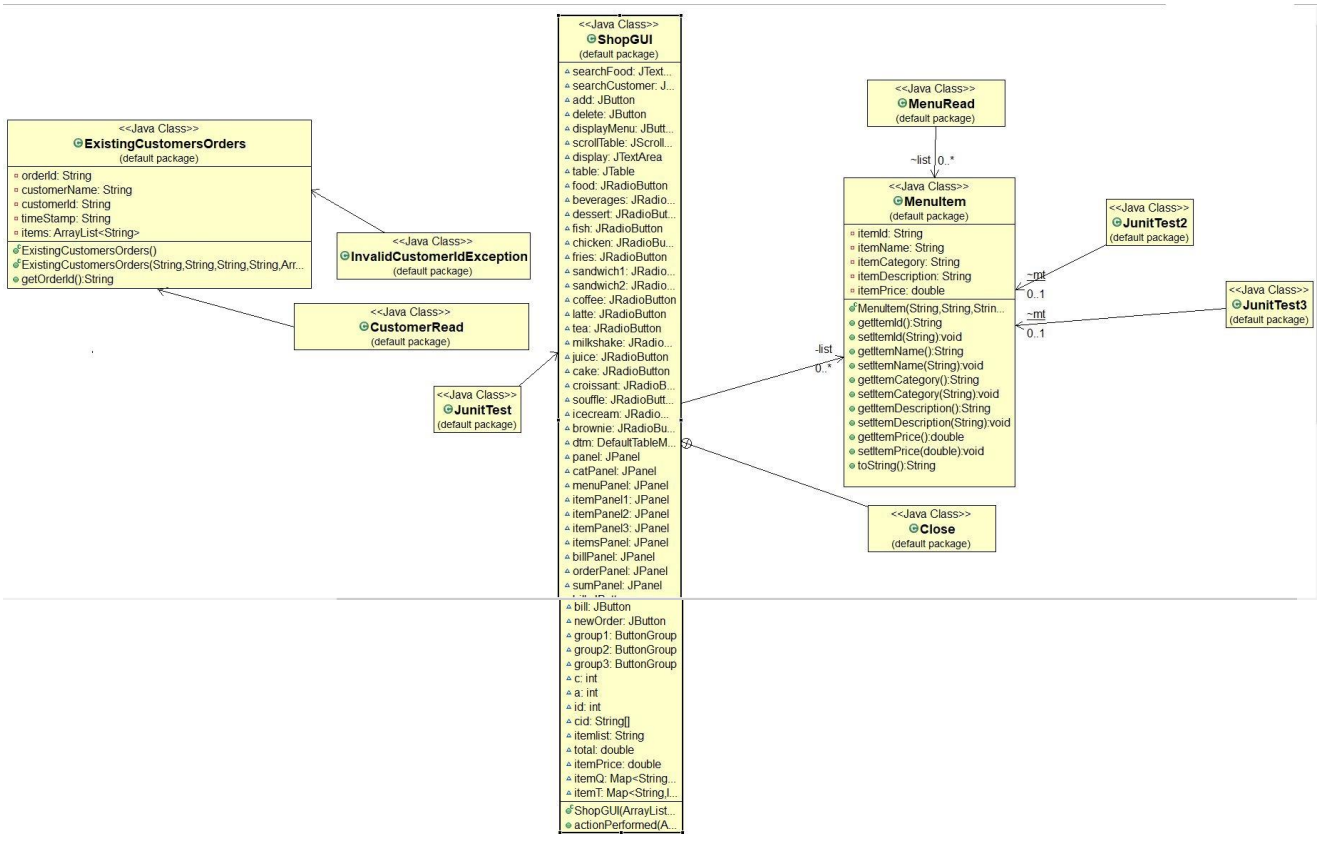
1 INTRODUCTION

The coursework deals with the development of a simulation for a coffee shop. This includes generating a menu, a list of orders, analyzing these orders and producing a result (price, discount, etc.) based on it. The programming language used is Java and the IDE used is eclipse.

2 SPECIFICATIONS

The program has successfully executed all the specifications mentioned in the coursework. There are no issues or bugs in the program.

3 CLASS DIAGRAM



4 DATA STRUCTURES

The data structures used includes ArrayList and linked LinkedHashMaps. The reason for using these data structres is as mentioned below:

- **ArrayList**

  - This is a dynamic array, as it can vary its sizes depending on the addition or removal of an element. Since the items in the food menu can be added or removed, the choice of ArrayList is best suited here.
  - The order of insertion of an element can be maintained [1].

- ***LinkedHashMap***

  - This is a category of HashMap [2].
  - LinkedHashMap has a key and a value pair. The key here is the item and the value is the quantity. When a particular item is added, the quantity of that item should increase and if the item is reduced, the quantity should reduce too. When the item is removed, only the quantity should reduce (if the quantity is more than 1). But in HashMap this feature is not supported and hence LinkedHashMap is best suited for here as it supports this feature.
  - LinkedHashMap provides the position of the key (item) from the selected row and this is checked with the HashMap and only the quantity will be changed as necessary.

- **Classes Used**

The classes implemented for various applications is as discussed.

- ***ShopGUI***

This class is for creating the GUI. This class allows addition and removal of items such as food, beverages and dessert to the order. It also produces a bill on clicking the generate bill button and it is written to a text file and it also includes the customer ID. The new order button, allows the user to input a new set of order and removes the previously ordered list from the GUI. Once the window is closed, all the orders including its quantity and the total price for all the orders are displayed as a pop up.

- ***MenuRead***

A text file containing the menu and the customer details are first created. The function of this class is to read the text file and get the details from it and store it in the ArrayList.

- ***MenuItem***

The function of this class is to initialize the variables such as item ID, category, name, price, description, etc., This class uses a constructor and the variables are initialized inside the constructor. With the help of get() and set() functions, the values for these variables set.

- ***InvalidCustomerIDException***

This class checks if a customer ID entered is invalid and if it is so then it produces an output that the ID is invalid.

- ***ExistingCustomersOrder***

This class includes the order ID, customer ID, customer name, time stamp and the ordered items. These details are stored in an ArrayList.

- ***CustomerRead***

This class reads the text file containing the customer details and stores it in an ArrayList.

- ***Test***

This is the main class where the ShopGUI, MenuRead and CustomerRead classes are called.

5 FUNCTIONALITY

The coffee shop has a menu with items in 3 categories i.e., food, beverage and dessert. Each of these have a unique ID. For each item in the food category, the ID starts with the letter "F" followed by a number, e.g.: "F110" for Fish Fingers. Similarly, for items under beverage the identifier is given by "B110" and for dessert as "D110". These details are stored on a text file and is accessed in the MenuRead class.

The customer order is placed on another text file and each customer has an ID, name, the items ordered, etc., Each customer has a unique customer ID and it is represented as "CID1234".

- **Discount**

The customer gets a discount of 5% when the total price goes above 50.

6 TESTING

The JUnit testing is used to check the accuracy of the code [3]. Each segment of the code performs a specific task and the role of JUnit testing is to check if that segment of the code is performing accurately.

Three JUnit tests were done. The first one is done on the ShopGUI class for checking the discount. The other two test are done on the get() –set() method, to check the item ID and item name.

- **Exceptions**

The two exceptions used are FileNotFound and IO exception and both of these are checked exceptions. Both the exceptions are used in the ShopGUI, CustomerRead and MenuRead class.

Since these classes require the data to be read from a text file or saved to a text file, if the file is not present then the FileNotFound exception will be shown.

IO exception is managed by the code itself. This exception is encountered when and IO task fails.

7 GUI

Figure 1 shows the GUI at the beginning of the application. Figure 2 shows the GUI when an order is placed. Figure 3 shows the dialog box pop up when the window is closed. The dialog box shows the total price for all the orders.

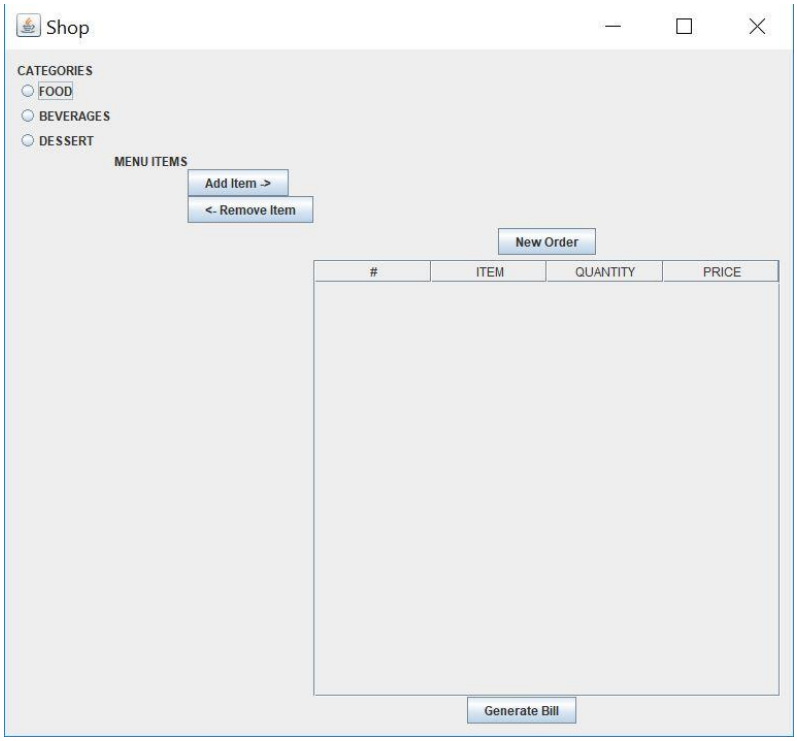


Figure 1: GUI

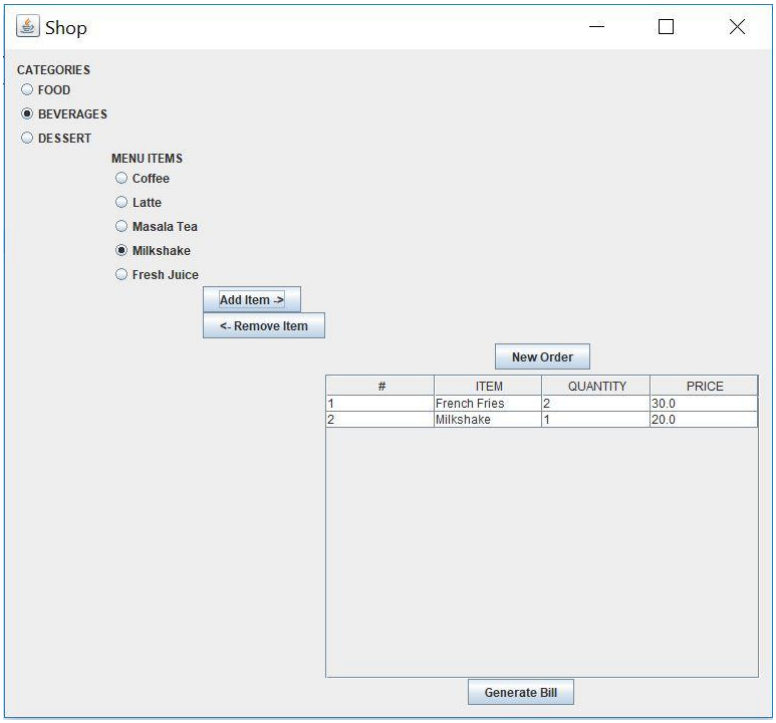


Figure 2: GUI after addition of items

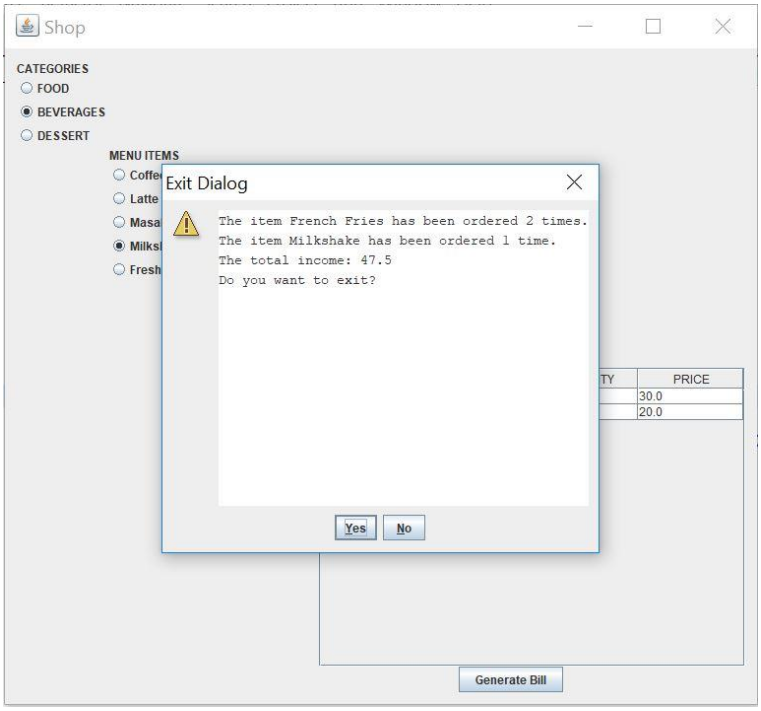
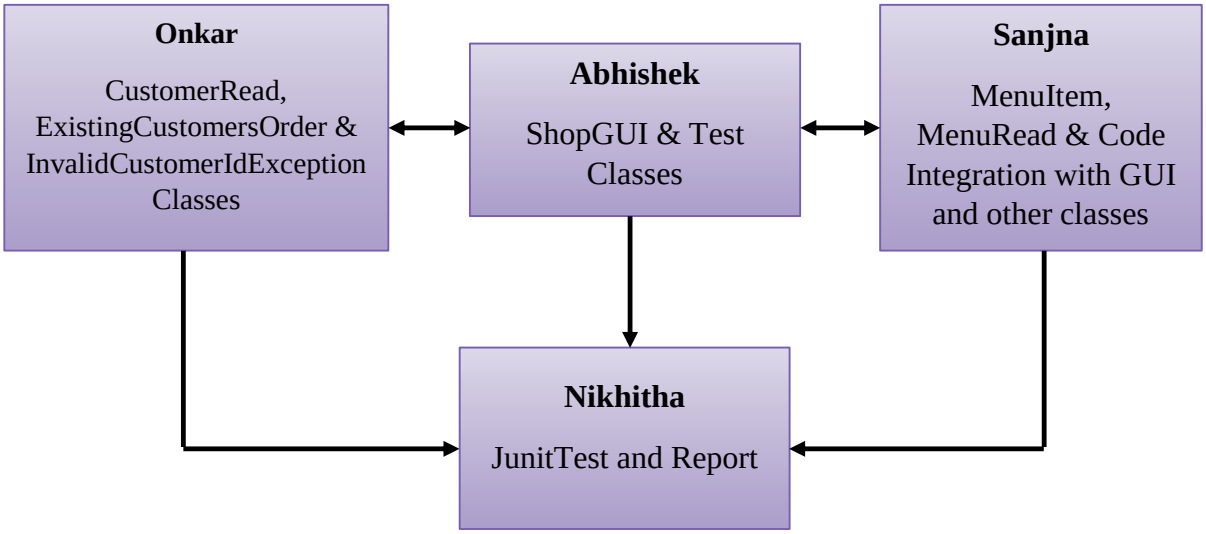


Figure 3: Dialog Box Pop Up

8 GROUP TASKS



9 GitHub Repository

https://github.com/abhirb/Coursework1-repo

10 References

[1]. https://www.callicoder.com/java-arraylist/

[2]. https://www.baeldung.com/java-linked-hashmap

[3]. https://stackoverflow.com/questions/10858990/why-use-junit-for-testing