Advanced Programming
Assignment - 4 (*with Bonus)

Name : Saanvi Singh
Roll No. : 2023454
Dated : 23-Nov-2024

HOW TO RUN CODE

1. Download the zip folder provided and extract all files
2. In IntelliJ, open a New Project, set it up.
3. Replace the src directory in this project with the above downloaded src directory
4. Compile and run the Main class

ASSUMPTIONS

1. User will provide correct type of input (integer for integer, etc)
2. Admin will handles orders in the chronology/priority in which it is shown in the pending orders' list
3. One particular order will be using only one menu, different orders can have different menus
4. On writing D, price will be sorted in descending order, anything else will result in ascending order
5. On writing C, order will be marked as completed, anything else will change it to next status
6. Replicating previous order will insert each item of that order into customer's cart in 1 quantity, if the item still exists and is available
7. Customer will provide review for only the items they have purchased
8. Any quantity being asked from user will be inputted as positive integer
9. Rating provided in review is between 0 and 5
10. If a customer orders first and then upgrades to VIP, the previous orders are not subject to be given priority in completion
11. Changes made to the database of customer, admin and menu will be saved between session using IO stream and file handling
12. GUI and CLI will not run simultaneously, but they will run in the same session
13. JUnit testing has to be done separately at the beginning

COLLECTIONS

Admin

1. orderQueue : TreeSet<Order>
   Maintains a sorted order of elements using compareTo method in Order to facilitate the prioritization of VIP and first come first serve otherwise policy
2. orderMap : HashMap<String,Order>
   Maps OrderID to Order to easily access order through its ID
3. completedOrder : ArrayList<Order>
   Maintains chronological order list of completed orders
4. cancelledOrder : ArrayList<Order>
   Maintains chronological order list of canceled orders

Cart

1. cartList : HashMap<FoodItem,Integer>
   Maps FoodItem to its Quantity added in the cart

Customer

1. customerList : HashMap<String,Customer>
   Maps CusID to Customer to easily access customer through its ID
2. notifications : Stack<String>
   Maintains last-in first-out policy of notifications
3. readNotifs : ArrayList<String>
   Maintains chronological order of notifications read
4. orderHistory : HashMap<String,Order>
   Maps OrderID to Order to easily access order through its ID

FoodItem

1. orderMap : HashMap<String,Order>
   Maps OrderID to Order to easily access order through its ID
2. cartList : HashSet<Cart>
   Prevents duplication when adding same item twice separately into a cart
3. reviews : ArrayList<Review>
   Maintains chronological order of reviews provided

Menu

1. itemList : HashMap<String,FoodItem>
   Maps ItemID to FoodItem to easily access item through its ID
2. deletedItems : HashMap<String,FoodItem>
   Maps ItemID to FoodItem to easily access item through its ID
3. category : HashSet<String>
   Prevents duplication when adding same category in multiple items

Order

1. statusMap : ArrayList<String>
   Maintains order list of status to be accessed through fixed integers

2. orderList : HashMap<FoodItem,Integer>
   Maps FoodItem to its Quantity added in the cart

UPDATES

GUI

Implemented using Java Swing library. Once the program runs, the user is asked whether to show the CLI or the GUI. If the user selects GUI, a pop-up screen appears which has buttons for “View Menu” and “View Pending Orders”. Buttons are also provided to navigate back and forth between these screens. Once done, the user can close the GUI, and will again be provided with the option to choose between CLI and GUI.

CLI

Any changes made through CLI will be reflected in the GUI when the GUI is opened the next time.

Serialization using IOStream

The user history, order history, as well as other relevant information are being saved in .ser type files at the end of each session, by creating FileOutputStream. In the next session, these files are being restored using FileInputStream, and the session continues from where it stopped last time.

JUnit Testing

Two testing classes are present, one for testing InvalidLogin exceptions and one for testing OutofStock item exceptions. Both have multiple tests to verify the correctness of the program. These can be run separately through the test/java directory.