

CMS 270: Object Oriented Design and Development
Group Project

Rollins Cafe Menu Viewer

Dallas Holm -

Report, MenuItem, Add/Remove Functions

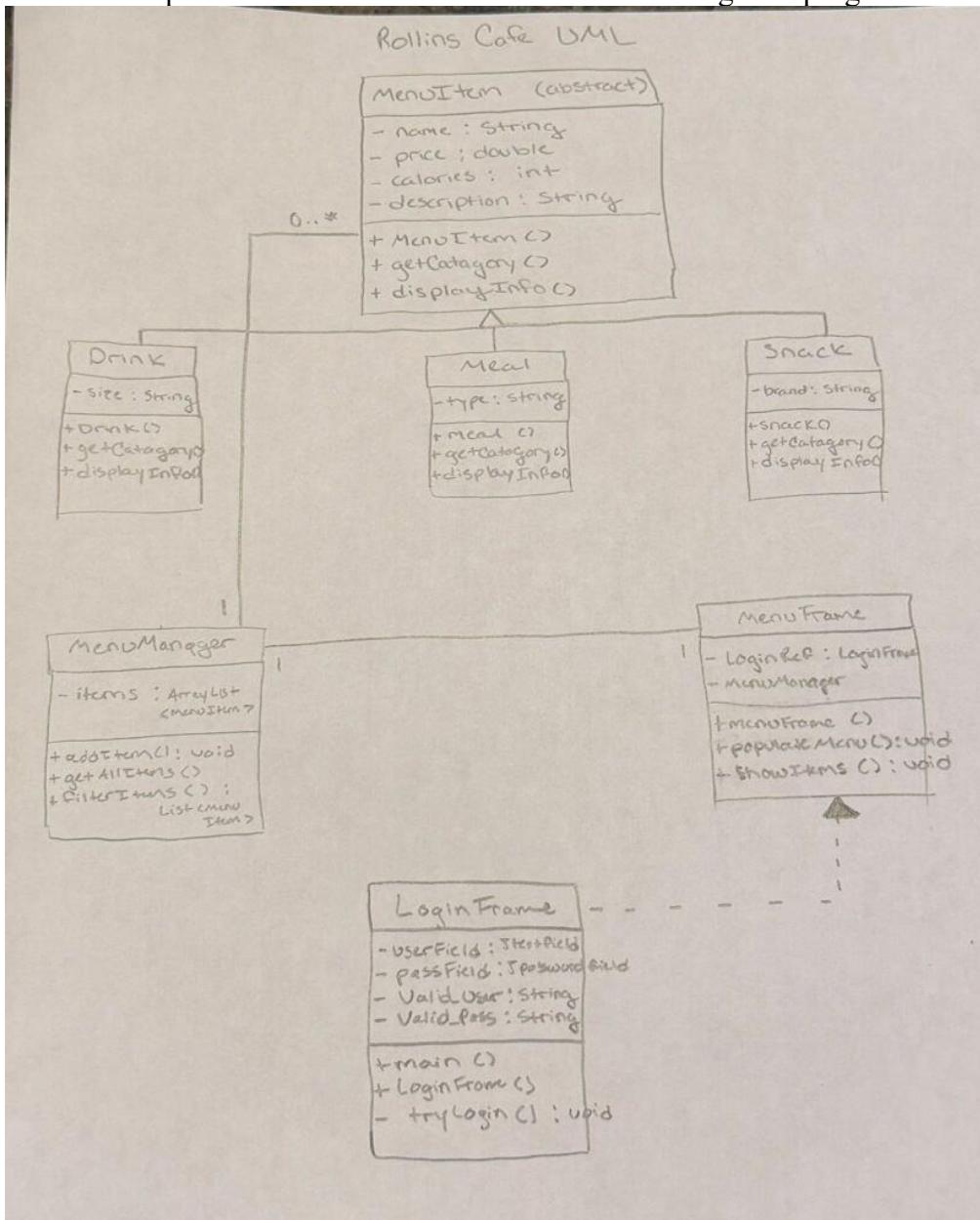
Dylan Lasensky -

UML, LoginFrame, MenuFrame, MenuManager

Luke Bowen -

Presentation, Drink, Meal, Snack

Our program is a menu viewer for a fictional Rollins Cafe. Displaying menu items as well as important information such as price, calories, description, and extra info that changes based on category of drink, meal, or snack. Its purpose is to display menu items for consideration. We considered a menu viewer to be a more creative choice for programs relating to our college. Luke came up with the idea when we first started deciding on a program to do.



Our program consists of seven different classes which includes `LoginFrame.java`, `MenuFrame.java`, `MenuManager.java`, `MenuItem.java`, `Drink.java`, `Meal.java`, and `Snack.java`. `MenuItem.java` acts as the abstract class which is inherited by `Drink.java`, `Meal.java`, and `Snack.java`. Establishing the private variables `String name`, `double price`, `int calories`, and `String description`. Also setting up the `displayInfo()` function which shows the related info for the specific item through encapsulation and polymorphism since each category changes what

displayInfo() returns. MenuManager.java sets up the ArrayList for all the items from each category. Also setting up getters for getting all items and getting specific categories.

LoginFrame.java acts as the starter and presents the user with a login panel which requires a username (admin) and password (pass123) to access the MenuFrame.java. Greeting the user with a message panel saying “Welcome to RollinsCafe!” once the proper login is entered. Once you get past the Login, MenuFrame.java is called and displays a scrollable table of menu items. Each menu item shows the variables established in MenuItem.java but also another variable which varies from Size, Brand, and Type depending on the related category. Clicking the “All” button shows all items. The “Snacks” button shows all snack items, the “Meals” button shows all meal items, and the “Drinks” button shows all drink items.

Clicking the “Add” button will display a separate panel labelled as “Add Menu Item.” Presenting the user with several text fields, allowing them to input important information for the item they wish to add to the list. The “Category” box lists the three options for the user to select and based on their choice, the “Size/Type/Brand” will print either of the three labels before the user’s input. And Lastly, the “Add” button on this panel will send the described item into the displayed table on the main panel. Closing the “Add Menu Item” panel with the added item reflected in the main panel list. Warning the user with a “Invalid Input!” if any input fields are not filled.

Clicking the “Logout” button will return the user to the login panel. The username and password fields still filled in for ease of access. If the user logs in once more, they’ll find that the MenuFrame.java has refreshed, allowing the user to restart the menu list if they wished to.

Clicking on the “Remove” button opens a separate panel labeled as “Remove Item.” Asking the user to select an item from the box which lists all items from the selected category. So, if the list is currently displaying only snacks, it will reflect the list of snacks. Lastly, is the “Remove” button, which will remove the selected item in the box. Closing the “Remove Item” panel with the selected item removed.

Challenges we faced throughout our process consisted of scheduling problems, github complications (mainly for me, Dallas Holm), and coding hurdles. Finding times to meet was difficult, seeing how each of us had other classes which conflicted with each other. We were able to get around it with Zoom calls but I found complications with github since both me and Lasensky set up the group project on github. I was able to get some assistance in adjusting to Lasensky’s repository, but there remains issues with the java filing which makes the github code unable to run despite the code running perfectly on my other workspace. We are getting around the issue by emailing the working code.

When it comes to coding issues, I faced complications when I wanted to add in the “Add” function to the “Add Menu Item” panel. When adding another item, I needed a way to distinguish and treat each category differently, like with the established list of items Luke made. So, I figured out with the help of AI that the best option would be to set up a scrollable box using JComboBox to list each category and allow one to be selected. After that, I needed to establish different cases for each category, so the added item will be assigned to that category. The JComboBox also helped display the list on the “Remove Item” panel, allowing the user to easily scroll through the list and select which item to remove.

Another coding problem I found was that after removing an item from the list, if I swapped focused categories, the item(s) removed would repopulate. In order to make my remove function more permanent, I needed to have it affect the ArrayList directly. I added removeItem() to delete the items I select in the Remove Item panel. I also had to add more to the Action Listener of the Remove button so it would push the selected item as variable toRemove to removeItem() for thorough removal.

Another coding challenge was found by Dylan when he was designing the MainFrame.java and LoginFrame.java. The GUI components gave him trouble lining up the layout and class relations. He had to think about how data is passed between the classes. This also made drawing up the UML diagram another challenge since he had to connect the dots while avoiding overlap.

For me, Dallas Holm, I was originally tasked with filling out the MenuItem.java, Snack.java, Meal.java, and Drink.java. But due to the github issues, Luke added it as well. I then added the Add and Remove functions to MainFrame.java and MenuManager.java, enabling the removal and addition of items from the menu. As well as typing up this report.

Luke added in the Snack.java, Meal.java, and Drink.java. Setting up the defining categories for each menu item. He filled in the default menu items which populate on the menu after getting past the login panel. He did the project presentation slides.

Dylan did a lot of heavy lifting by setting up MenuManager.java, LoginFrame.java, and MainFrame.java. Setting up the backbone for the working menu and login panel. Made the main menu an easy to use application that allowed the user to filter the list based on category. He did the UML diagram as well.