Mingyun Kim

Aryan Khadiri

Prajwal Pyakurel

CS151 Team Project Report

Link: https://github.com/mikim42/SJSU_CS151/tree/master/POS

POBest

Abstract:

In order to run any customer service based business like a store or a restaurant, there needs to be a device that contains a system, which takes the items ordered by customers as input, adds up the total price of the items, and provides options for checkout. In addition, the menu in the system needs to be dynamic, so that the user of the device can easily add and remove items, change prices, and add notes whenever there are special cases. Also, to have unique menu sets, it needs to interact with a database system to save/load menu sets. This project is about creating a POS system, POBest, based on Java Swing API, that has all the aforementioned functionalities. POBest is a simple,yet intuitive and effective system.

Description:

Objective: Our objective of creating POBest was to design a POS system that can add, and edit menu items in real time, read the items from a database, and add the items to a cart, from which checkout can be done.

Outcomes and functionality: We were successful in creating a system that has a dynamic menu, where adding, and editing can be done in real time, as well as the items are always saved in the database. We were also able to use the panels and frames effectively to perform the desired functions as mentioned above, using an attractive and intuitive user interface. The following are the functionalities of POBest:

1) Add items to the menu
2) Modify the items
3) Have the items saved in a database in real time
4) Add items to Cart
5) Checkout from the cart
6) View the total and add tips
7) Chooses payment methods

Components used:

- JPanel and JFrame: We used JPanel and JFrame to display the contents of the menu, the buttons, and take in user input.
- JButtons: Users can click on various buttons on the screen to select their choice, and then the program receives the input and processes the data.
- JTextField and JLabel: JTextField and JLabel components contain texts provide the users with instructions on choosing the right buttons and putting in inputs in the right areas.

- Database: The data is stored into a database(.txt file) in real-time, which prevents data loss if the program gets unexpectedly terminated. The database file is set to be a hidden file to be protected.

Implementation - The project implementation provides the details on how the specified problem was solved. This implementation will include diagrams, use cases, CRC cards, design patterns, programming methodologies, efforts to optimize the project code, and any difficulties encountered while implementing the design.

Design patterns:

MVC Architecture

We designed the system by using MVC design pattern and implemented Java GUI to interact with the users.

Model: The model part of the project contains the parts that store the menu items, and collections of items in the shopping cart. The classes that belong to Model are:

- Item
- Shopping Cart
- Database

View: The view part of the project contains the panels that are seen by the users, and buttons and textfields that the users can interact with. The classes of View are:

- Appframe

- CheckoutPanel
- ChangePanel
- CreditPanel
- EditAddedItemFrame
- ItemAdderPanel
- MenuCanvas
- MenuCanvasEditor
- SideStatusCanvas
- EditAddedItemFrame

Controller: The controller part communicates with the view and model parts, gets input from view, and manipulates the model based on user input. The controller of POBest contained the following classes:

- Main
- MenuItems

Iterator design Pattern:

We applied iterator design pattern in order to make our classes more reusable in case we changed our collection type from ArrayList to other type. The implementation is in the

following picture:

```
public void modifyItem(String item, String name, double price) {
    Iterator<Item> it = items.iterator();
    while (it.hasNext()) {
        Item tmp = it.next();
        if (tmp.getName().equals(name)) {
            tmp.setName(name);
            tmp.setPrice(price);
        }
    }
}

public Item findItem(String name) {
    Iterator<Item> it = items.iterator();
    while (it.hasNext()) {
        Item tmp = it.next();
        if (tmp.getName().equals(name))
            return tmp;
    }
    return null;
}

public void updateDB() {
    try {
        BufferedWriter writer = new BufferedWriter(new FileWriter(".db"));

        Iterator<Item> it = items.iterator();
        while (it.hasNext()) {
            Item tmp = it.next();
            writer.write(tmp.getName() + " " + tmp.getPrice() + "\n");
        }
```

Use Cases:

Add items in the menu

Edit the added items

Add items to the cart

Edit items added to the cart

Checkout form the cart

Choose tips and payment methods

Enter the total cash paid and view the change

CRC Cards:

MenuItems	
• MenuCanvas • Manipulate Items in the database	• MenuCanvas

ShoppingCart	
• Create ArrayList of items • add them to Cart	• SideStatusCanvas • EditAddedItemFrame • CheckoutPanel

SideStatusCanvas	JPanel
• Display items in cart • Edit Items in cart • CheckOut	• ShoppingCart • EditAddedItemFrame • CheckOutPanel

Item	
• Item Name • Price • Quantity • Note • Id	• MenuItems • ShoppingCart

ItemAdderPanel	JFrame
• Add Items to the Cart • Add notes to the items	• SideStatusCanvas • MenuCanvas

Main	
• Display JFrame of the Application	• AppFrame • ShoppingCart

MenuCanvas	JPanel
• MenuItems	• SIdeStatusCanvas • MenuCanvasEditor

MenuCanvasEditor	JPanel
• Items • Add new items • Change Price	• ShoppingCart • SideStatusCanvas • MenuItems

AppFrame	JFrame
• Create JFrame • Add MenuCanvas	• MenuCanvas • ShoppingCart

ChangePanel	JFrame
• Calculate change • Display change	• CheckoutPanel

CheckoutPanel	JFrame
• Display Total • Calculate Tips • Direct to ChangePanel or CreditPanel	• SideStatusCanvas • ChangePanel

CreditPanel	JFrame
• Obtain Credit Card information	• CheckoutPanel

EditAddedItemFrame	JFrame
• Edit items displayed on Cart	• SideStatusCanvas

Difficulties faced and Efforts to Optimize the Code:

The program did work well in the alpha phase, but we realized that there were some areas that could be improved.

1) Law of Demeter: We have reduced tight couplings as much as possible. To achieve that, we have made our classes as immutable as the functionality permitted. We also created get methods; we specifically created getTotal methods in all the panels that accessed the total.
2) Iterator Pattern: We used Iterator pattern so that the iteration through the database can be possible with any kind of collection.
3) Decimal Places: We encountered a problem where the values displayed contained more than three decimal places. We converted the values into Bigdecimal and used its function to display only upto three decimal places.

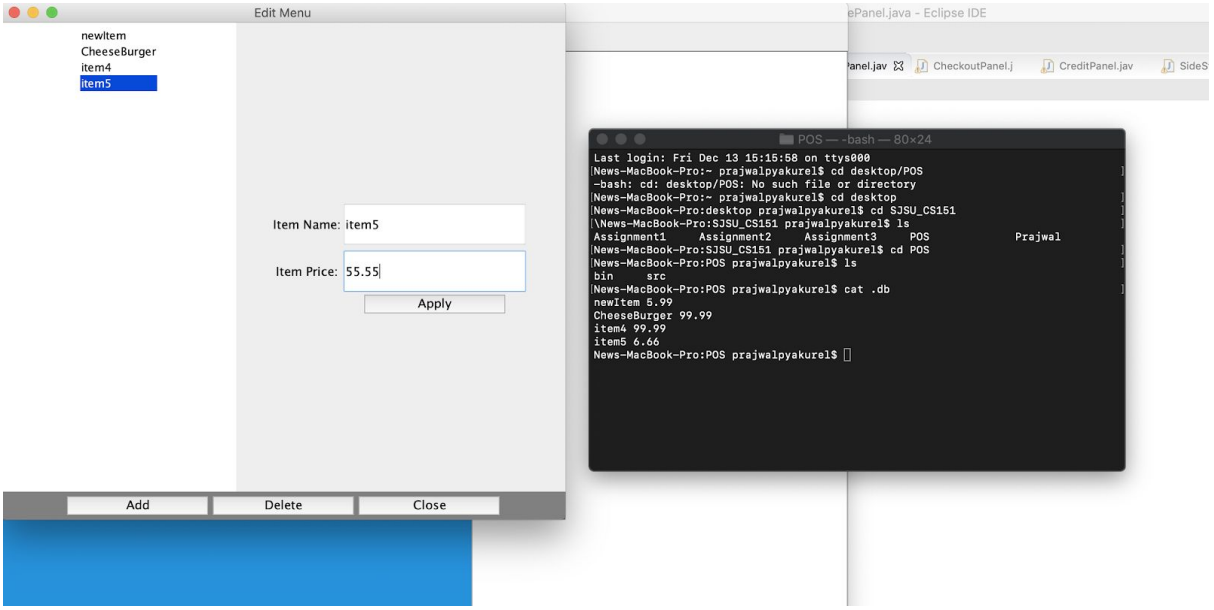
Experimental Evaluation

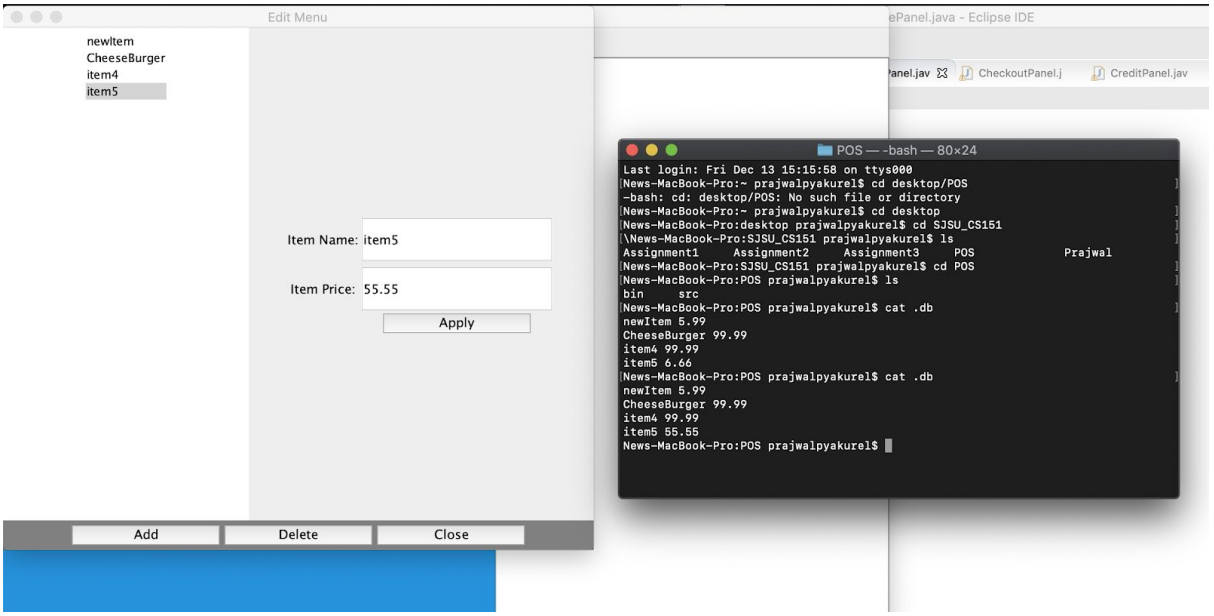
In order to test our prototype, performed all the tasks mentioned in the use cases. We got the results and improved our code to match our expectations. Some of the experiments were:

Database test: We checked if our application was adding, editing and deleting items while manipulating our database in real time. For that we added the items, deleted items, and edited items, and then viewed our database file. Its performance didn't match our expectations in the beginning, but we improved the code and made it work. The pictures below will explain further:

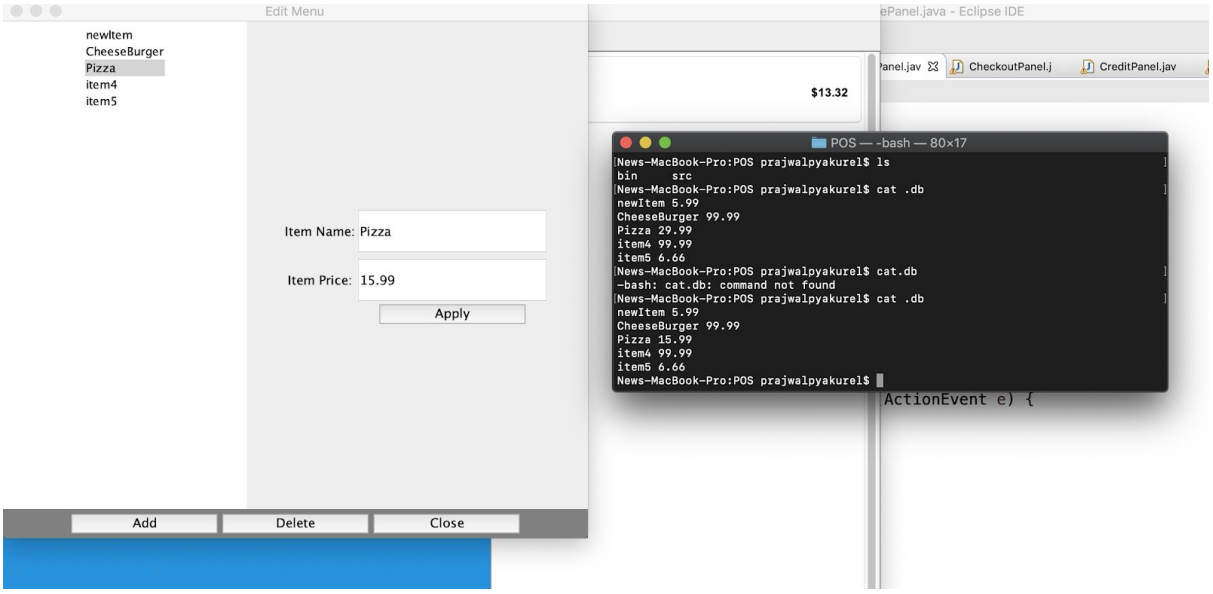
- We edited the price of item5, and found the changes in our database as shown in the following pictures. In the first picture, you can see the price of item6 is 6.66 in the database (right side terminal window). In the second picture, after the apply button

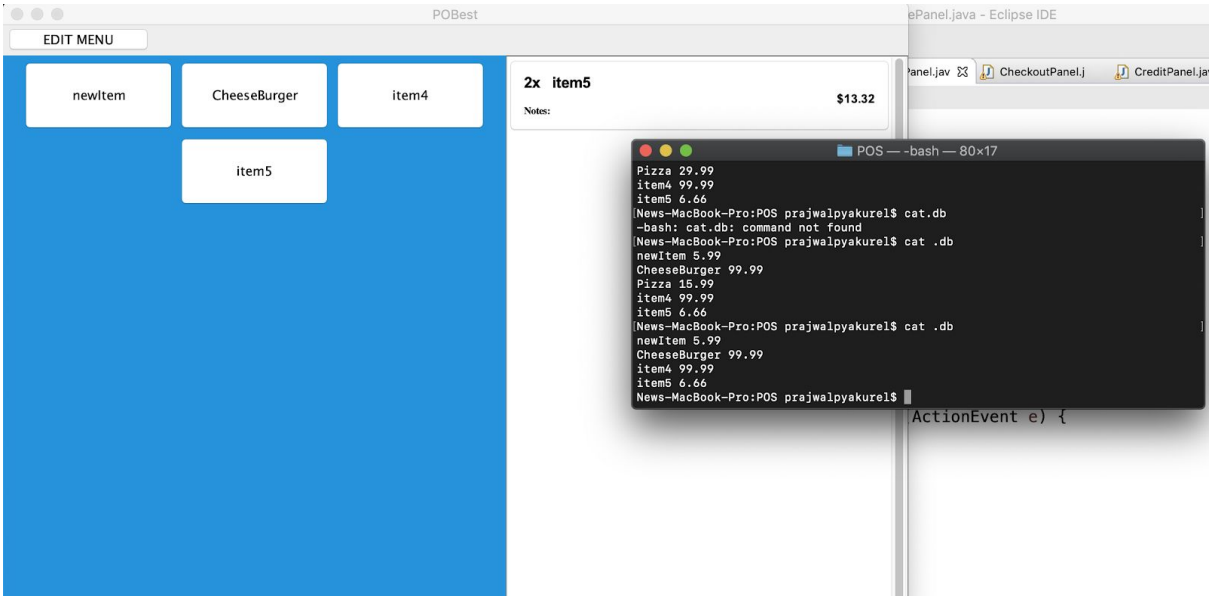
is clicked, the price changes to 55.55.





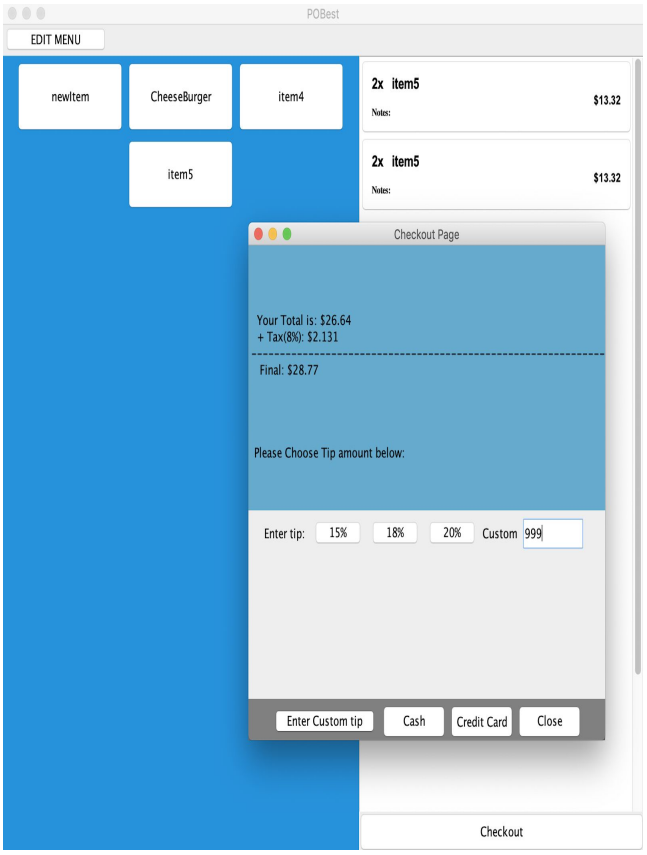
- Deleting the item: We deleted Pizza from the Edit Menu Panel. The first shows the database, and Edit Menu panel before Pizza was deleted. The second image shows the database and Menu Panel after Pizza was deleted.

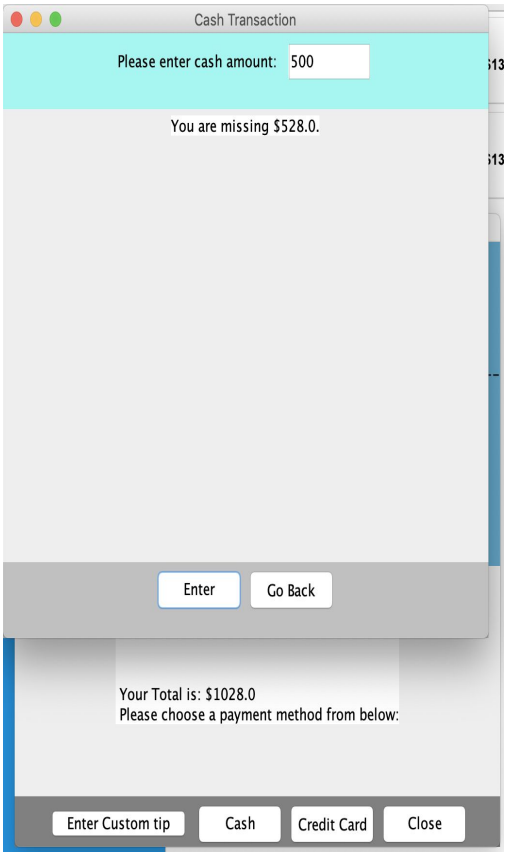


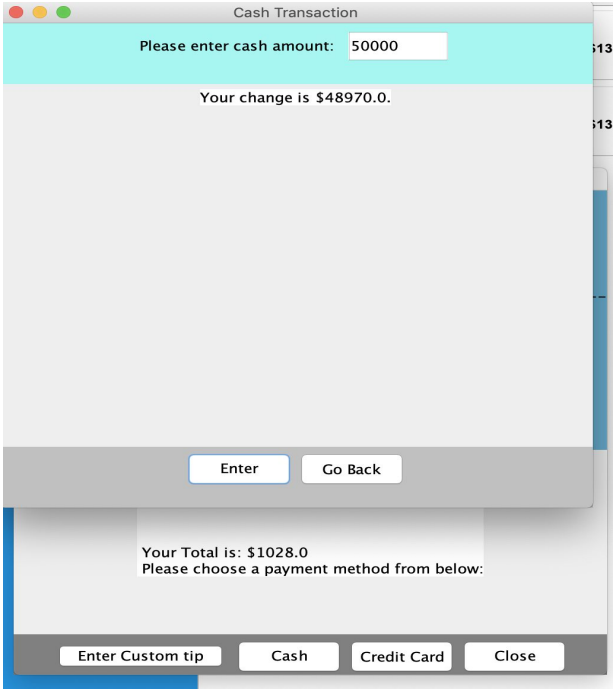


2) The second test that we performed was checking the value of total after adding tips, and checking out through changePanel.

- We tested with various values of custom tip amounts, and checked the Cash Transaction window if the results were satisfactory.







Summary - The project was successful in meeting all the expectations that we had set before we started writing the code for it. Our system, POBest, addresses all the use cases that we had outlined, and turned out to be a simple but effective application.

What we learned: We got the chance to implement the design patterns and programming techniques that we studied in class. We were able to sharpen our skills on object oriented programming techniques, Java programming language and Java GUI based on Swing API. We also learned how to create high quality code that contains reusable classes, and lesser areas for error.

An area for improvement would be adding navigation panel in the menu, that classifies the items based on their type content, or prices.

We can change the appearance of our application by adding images in the panels to make them look more modern and professional, as well as also experiment with other colors.