Restaurant Ordering System Java OOP Final Project

Lecturer:
D4017 JUDE JOSEPH LAMUG MARTINEZ, MCS

Arranged by:
2602158626 - Adyatama Mahabarata

3rd June 2023
Object Oriented Programming
Computer Science Faculty
Binus International University 2022/2026
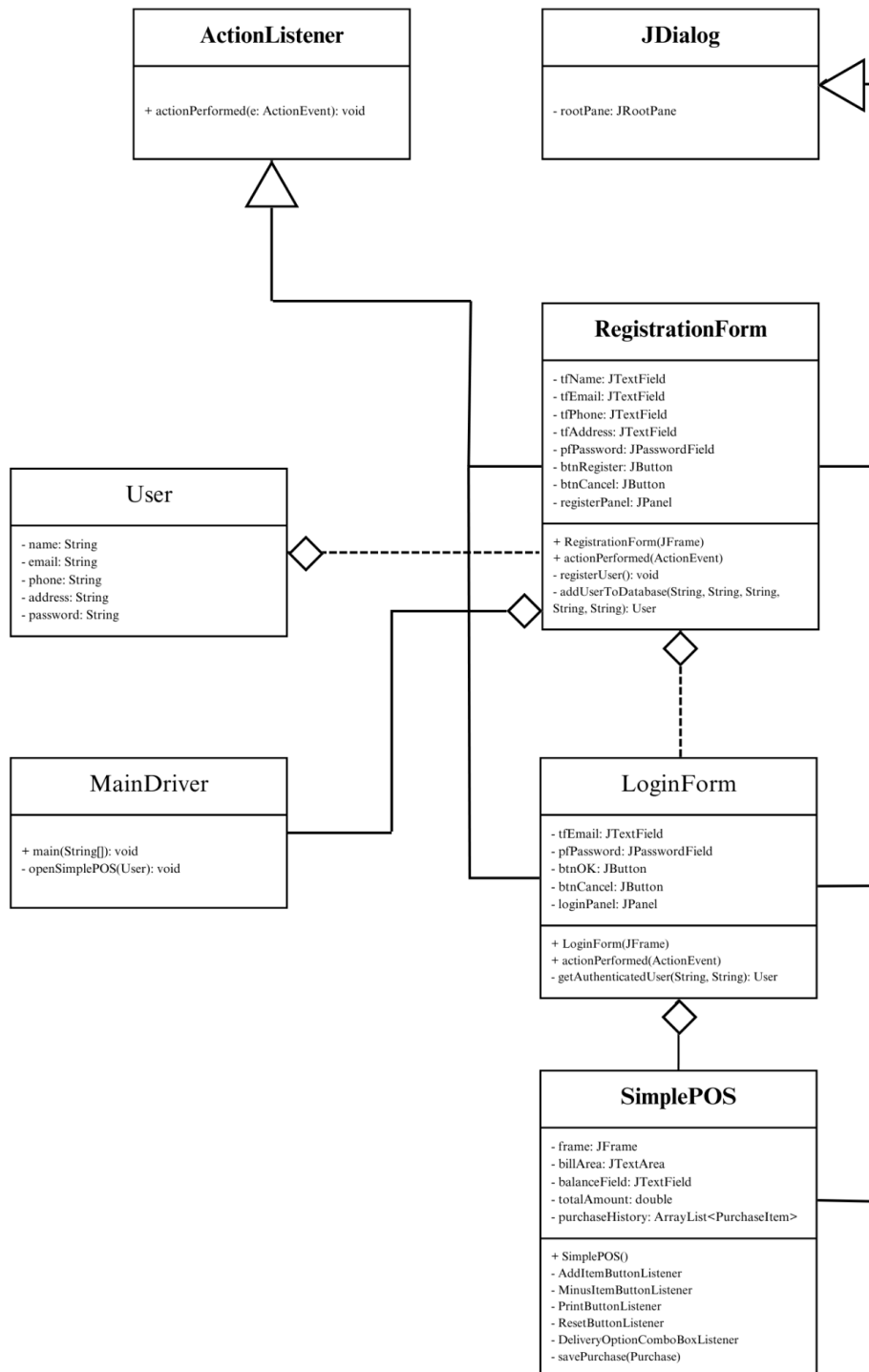
Table of Contents

Chapter 1 - Brief Description of Project Specification

The code that is used on this project is a Java language programming that consists of registration form, login form, and a simplePOS system that represents a restaurant ordering system. From the three main code files, the first one being the registration form, it allows the users to create an account by entering their information, such as name, email, address, and password. After the user registers the data will be saved and stored into a MySQL database. After a user registers, if the user decides to close the application they can choose the login form where they can just input their email and password to log them in to the application. The data will be validated through the MySQL database to ensure the email and passwords validity. If the email and password matches from the data stored in the MySQL database, the user then the login will be considered a success. Upon successful registration or login from the user, the user will be redirected to another frame where it will ask the user if they want to log in with the account they inputted and if they say yes they will then be redirected again to the restaurant ordering system called simplePOS.

The last part of the code, the SimplePOS system, implements a basic point of sale functionality. It simulates a restaurant ordering system, allowing users to perform actions such as adding items to an order, removing items, printing the order, calculating totals and change, and handling delivery options. The SimplePOS system is designed to facilitate the ordering process in a restaurant setting. It provides a straightforward interface for adding items to a customer's order and handling the necessary calculations for payment. The system can also accommodate delivery orders, offering options to specify delivery addresses and calculate any additional charges. In summary, the code provides a basic framework for a restaurant management system. It covers user registration, login authentication, and a simplified point of sale functionality. By integrating with a database and implementing the necessary logic, the application offers a practical solution for managing user accounts and processing restaurant orders.

Chapter 2 – Class Diagram

From the code this is the Class Diagram:

ActionListener

+ actionPerformed(e: ActionEvent): void

JDialog

- rootPane: JRootPane

RegistrationForm

- tfName: JTextField
- tfEmail: JTextField
- tfPhone: JTextField
- tfAddress: JTextField
- pfPassword: JPasswordField
- btnRegister: JButton
- btnCancel: JButton
- registerPanel: JPanel

+ RegistrationForm(JFrame)
+ actionPerformed(ActionEvent)
- registerUser(): void
- addUserToDatabase(String, String, String, String, String): User

User

- name: String
- email: String
- phone: String
- address: String
- password: String

MainDriver

+ main(String[]): void
- openSimplePOS(User): void

LoginForm

- tfEmail: JTextField
- pfPassword: JPasswordField
- btnOK: JButton
- btnCancel: JButton
- loginPanel: JPanel

+ LoginForm(JFrame)
+ actionPerformed(ActionEvent)
- getAuthenticatedUser(String, String): User

SimplePOS

- frame: JFrame
- billArea: JTextArea
- balanceField: JTextField
- totalAmount: double
- purchaseHistory: ArrayList<PurchaseItem>

+ SimplePOS()
- AddItemButtonListener
- MinusItemButtonListener
- PrintButtonListener
- ResetButtonListener
- DeliveryOptionComboBoxListener
- savePurchase(Purchase)

4

In this project, there are several classes involved in the restaurant ordering system. The explanation for the classes and their relationships based on the UML diagram:

1. **MainDriver**:
   - It contains the **main** method and serves as the entry point for the application.
   - It has an association on the **RegistrationForm** class.
   - It interacts with the user through **JOptionPane** dialogs to prompt for registration or login.
2. **RegistrationForm**:
   - It extends **JDialog** and serves as a registration form for creating a new user account.
   - It has a composition relationship with the **JTextField**, **JPasswordField**, JButton, and **JPanel** classes.
   - It has a dependency on the **User** class for user registration.
3. **LoginForm**:
   - It extends **JDialog** and serves as a login form for authenticating users.
   - It has a composition relationship with the **JTextField**, **JPasswordField**, **JButton**, and **JPanel** classes.
4. **User**:
   - It represents a user of the system.
   - It has attributes such as **name**, **email**, **phone**, **address**, and **password**.
   - It is used for user authentication and registration in the **LoginForm** and **RegistrationForm** classes.
5. **SimplePOS**:
   - It represents the main GUI application for the restaurant ordering system.
   - It extends **JFrame** and displays the graphical user interface for the POS system.
   - It has a dependency on the **JButton** and **JLabel** classes for UI components.
6. **JDialog**:
   - It is a class provided by **Java Swing library** for creating dialog boxes in a graphical user interface.
   - It is used as a base class for **RegistrationForm** and **LoginForm** to display the registration and login forms.
   - It has a dependency on the **JButton**, **JTextField**, **JPasswordField**, and **JPanel** classes for UI components.
7. **ActionListener** (Interface):
   - It is an interface provided by the Java Swing library.
   - It is implemented by the **LoginForm** and **RegistrationForm** classes to handle button click events.

- The **actionPerformed** method is defined in this interface and overridden in the implementing classes.

These classes have various dependencies and relationships, dependency, and inheritance, which are represented in the UML diagram.

Chapter 3 – Explanation of the Algorithms

In the code that is used for this final project, there are many algorithms that are utilized and here is the explanations of the main algorithms used in this project:

From the SimplePOS code file:

1. **Constructor for the SimplePOS**: The SimplePOS class serves as a main dialog for the POS system. In the constructor, it initializes the main application frame (JFrame), sets its properties, and creates various components such as labels, buttons, and panels to construct the user interface.

2. **AddItemButtonListener**: This is an inner class that implements the ActionListener interface. It handles the event when an item button is clicked. When an item button is clicked, the listener retrieves the item name and price, updates the total amount, appends the item details to the bill area, and adds the item to the purchase history.

3. **MinusItemButtonListener**: This is another inner class implementing ActionListener to handle the event when the minus button is clicked for an item. It subtracts the item price from the total amount, removes the item details from the bill area, and removes the item from the purchase history.

4. **PrintButtonListener**: This inner class handles the event when the print button is clicked. It retrieves the current date and time, parses the balance from the balance field, validates the balance, and compares it with the total amount. If the balance is sufficient, it calculates the change, constructs the bill text, displays the bill in a dialog, clears the bill area and balance field, and saves the purchase information.

5. **ResetButtonListener**: This inner class handles the event when the reset button is clicked. It clears the bill area, sets the total amount to zero, clears the balance field, and clears the purchase history.

6. **DeliveryOptionComboBoxListener**: This inner class implements ActionListener to handle the event when the delivery option is selected from the combo box. It retrieves the selected option and displays it in a dialog.

7. **savePurchase**: This method is called from the PrintButtonListener to save the purchase details. In this project code, it simply prints the purchase details to the console. In a real application, you would typically save the purchase information to a database or file.

8. **PurchaseItem**: This is a record (introduced in Java 14) representing a purchased item. It holds the item name and price.

9. **Purchase**: This is another record representing a complete purchase. It contains the date and time, a list of purchased items, the total amount, the balance, and the change. The toString() method is overridden to provide a string representation of the purchase.

10. **main:** The main method is where the application is launched. It uses SwingUtilities.invokeLater to ensure the GUI is created and updated on the event dispatch thread, which is required for Swing applications.

From the RegistrationForm code file:
1. **tfName, tfEmail, tfPhone, tfAddress**: These are JTextField components for capturing the user's name, email, phone number, and address, respectively.

2. **pfPassword**: This is a JPasswordField component for securely capturing the user's password.

3. **btnRegister, btnCancel**: These are JButton components for the register and cancel actions, respectively.

4. **registerPanel**: This is a JPanel that serves as the container for the registration form.

5. **RegistrationForm(JFrame parent)**: This is the constructor of the RegistrationForm class. It sets up the dialog window, sets the title, sets the content pane to the registerPanel, defines the size, sets the window as modal, centers the window on the parent frame, and sets the default close operation. It also adds ActionListener instances to the btnRegister and btnCancel buttons.

6. **actionPerformed(ActionEvent e)**: This method is implemented from the ActionListener interface, but it is not used in this code.

7. **registerUser()**: This method is called when the register button is clicked. It retrieves the entered values from the text fields and password field. It checks if any field is empty, displays an error message if so, and returns. If all fields are filled, it attempts to add the user to the database by calling the addUserToDatabase() method. If the user is added successfully, the registration form is closed. Otherwise, an error message is displayed.

8. **addUserToDatabase(String name, String email, String phone, String address, String password)**: This method attempts to add the user's information to a database. It establishes a connection to the MySQL database using the provided URL, username, and password. It prepares an SQL INSERT statement with placeholders for the user's details and executes it by setting the values using the PreparedStatement object. If the insertion is successful, it creates a User object with the entered details. Finally, it closes the statement and connection. If any exception occurs during the database operations, it prints the stack trace.

9. **main(String[] args)**: This is the entry point of the program. It creates an instance of the RegistrationForm, retrieves the user object, and prints a success message if the user is not null (indicating successful registration) or a cancellation message otherwise.

From the LoginForm code file:
1. **tfEmail**: This is a JTextField component for capturing the user's email.

2. **pfPassword**: This is a JPasswordField component for securely capturing the user's password.

3. **btnOK, btnCancel**: These are JButton components for the OK and cancel actions, respectively.

4. **loginPanel**: This is a JPanel that serves as the container for the login form.

5. **LoginForm(JFrame parent)**: This is the constructor of the LoginForm class. It sets up the dialog window, sets the title, sets the content pane to the loginPanel, defines the size, sets the window as modal, centers the window on the parent frame, and sets the default close operation. It also adds ActionListener instances to the btnOK and btnCancel buttons.

6. **actionPerformed(ActionEvent e)**: This method is implemented from the ActionListener interface, but it is not used in this code.

7. **getAuthenticatedUser(String email, String password)**: This method attempts to authenticate the user by checking the entered email and password against the database. It establishes a connection to the MySQL database using the provided URL, username, and password. It prepares an SQL SELECT statement with placeholders for the email and password and executes it by setting the values using the PreparedStatement object. If a matching user is found in the database, it creates a User object with the retrieved details. Finally, it closes the statement and connection. If any exception occurs during the database operations, it prints the stack trace.

8. **main(String[] args)**: This is the entry point of the program. It creates an instance of the LoginForm, retrieves the user object, and prints the corresponding message based on its value. If the user is authenticated successfully, it prints the user's details.

In summary, the code shows and demonstrates the implementation of a restaurant ordering system with user registration and login functionality using Java Swing and MySQL database interactions.

Chapter 4 – Screenshots of the Program Application

These are the screenshots of the restaurant ordering system:
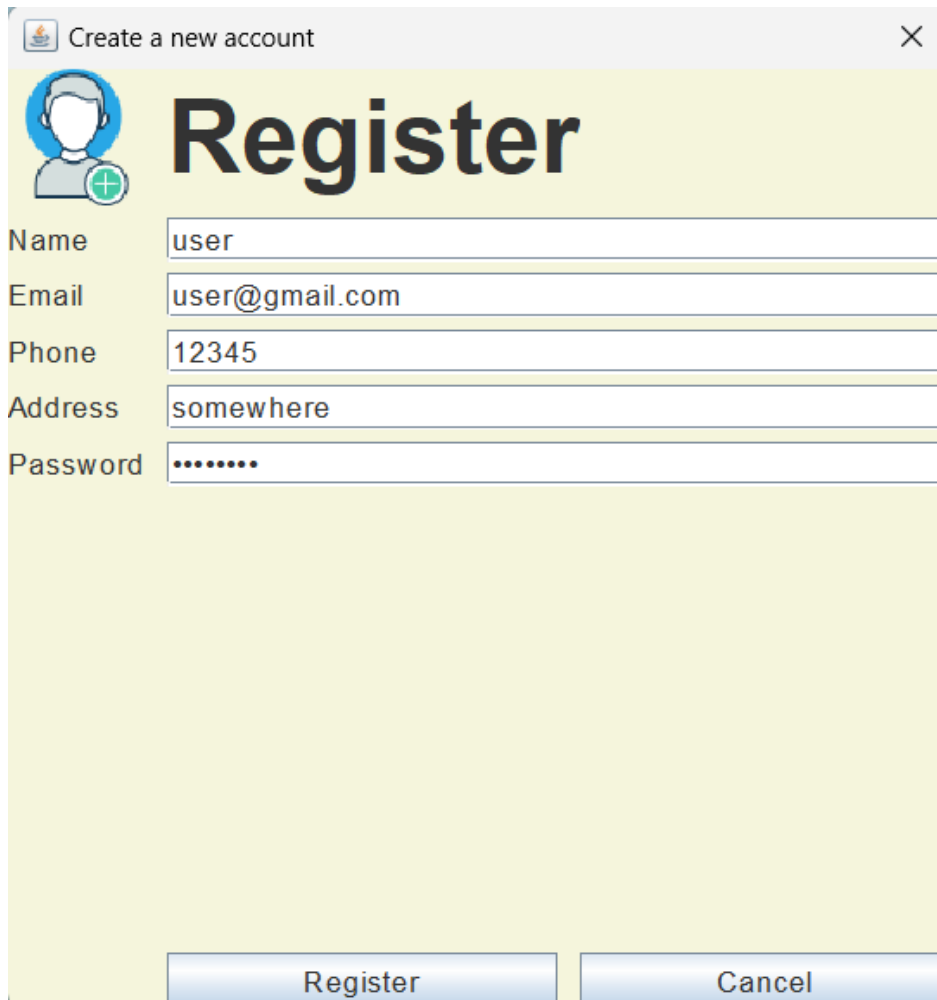
In this picture the prompt will appear when the user runs the restaurant ordering system app and then it will ask the user to either Register or Login.
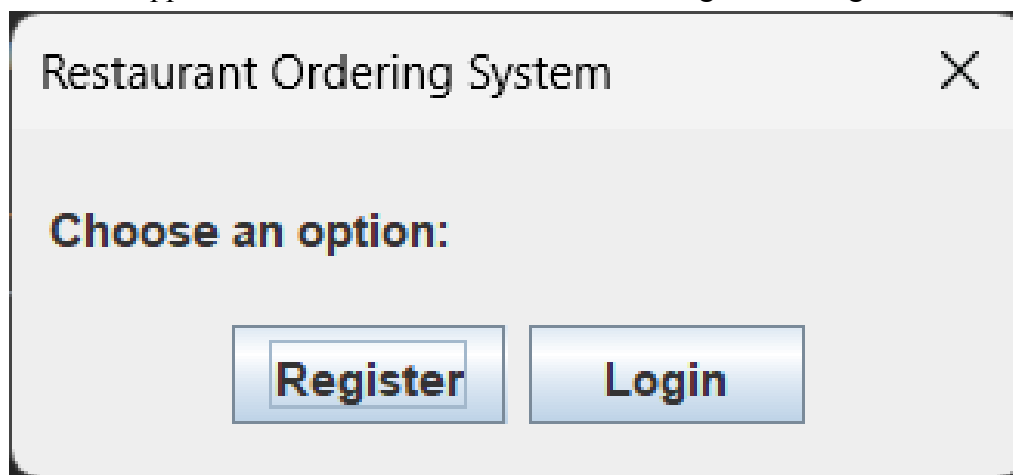
In this picture after the user chooses the option Register, it will redirect them to the Register page where the user can input the data to Register.

In this picture after the user input their data they have the option to Register by pressing the Register button or cancel button. If they press the Register button they will then be redirected to another confirmation prompt and their data will be updated into the MySQL database. And if they don't want to Register they can press the cancel button where they will then be closing the program.

In this picture after the user Registers they will receive a prompt if they want to be registered as the registered data that they have inputted and if they press the yes button they will then be redirected to the restaurant ordering system. If they press the no button it will then close the program but still saves the users data.
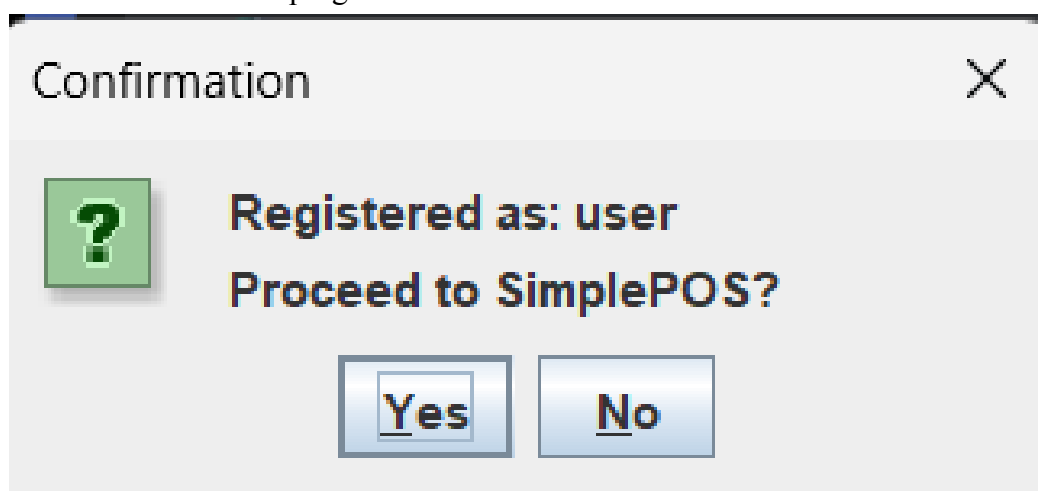
In this picture it shows the MySQL database where it will store the user data that registers from the Registration Form.
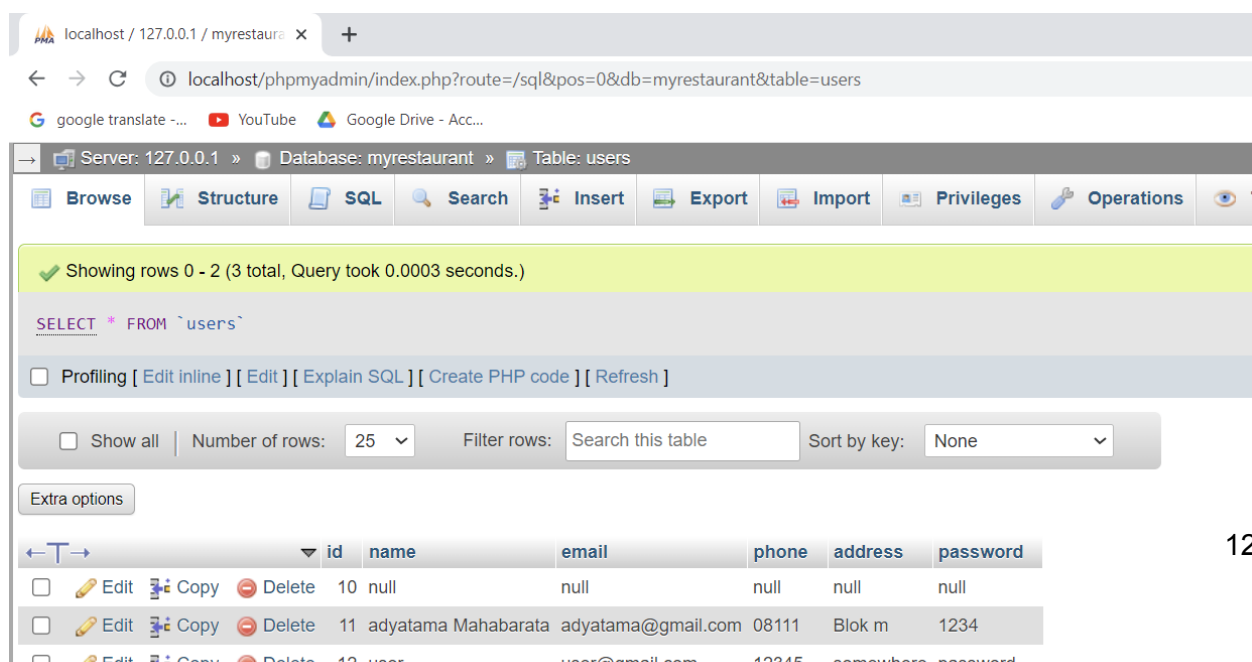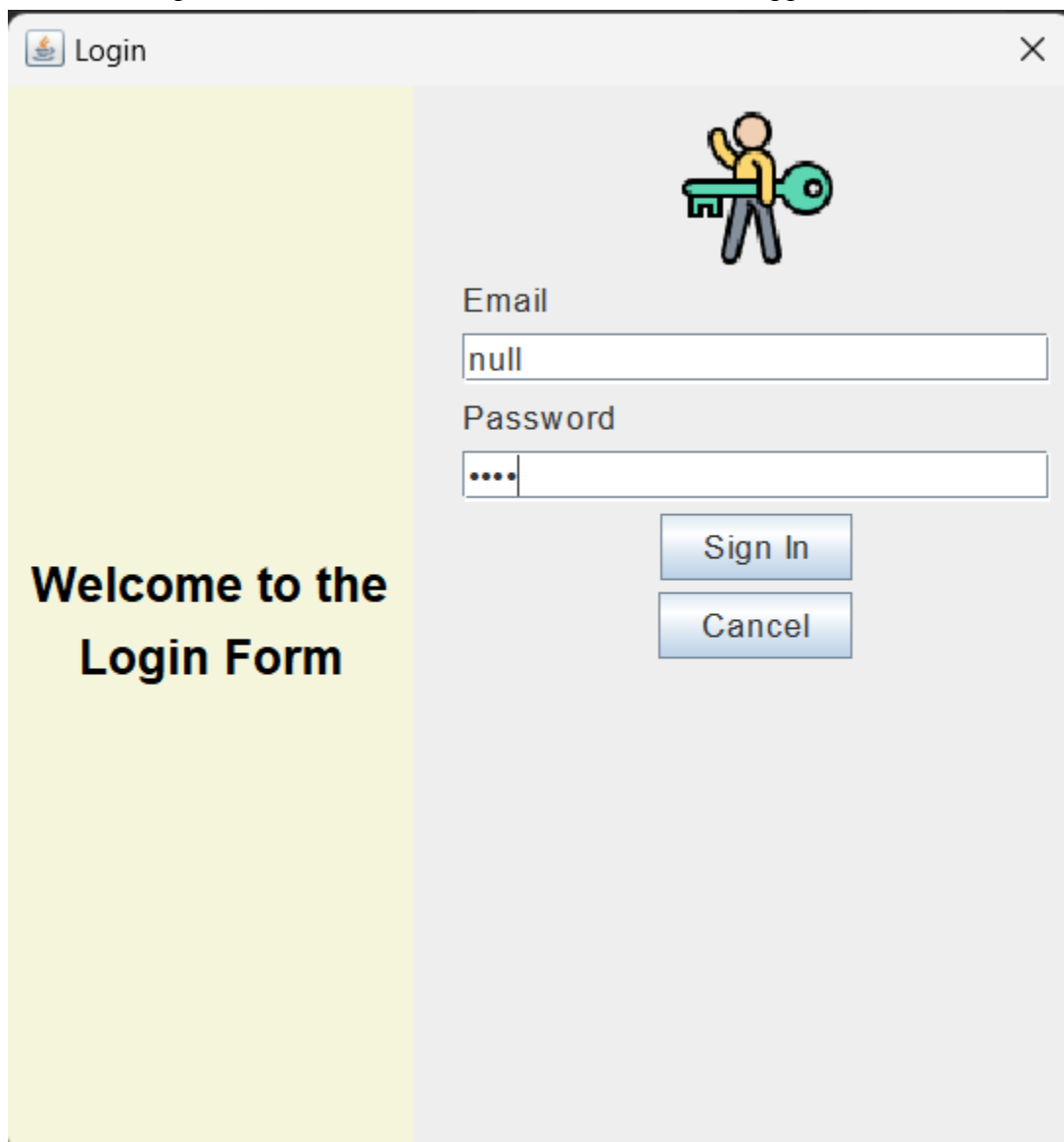
In this picture the Login Form will appear if the user chooses to Login instead of register and they can add the data where they can input the user information into the Login Form.

In this picture after the user inputs their user information they can choose to press the sign in button to Login and get redirected to a confirmation prompt. And if they choose not to and presses the cancel button it will then close the application.
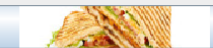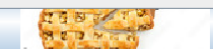
In this picture after the user Logins they will receive a prompt if they want to be logged as the registered data in the MySQL database that they have inputted and if they press the yes button they will then be redirected to the restaurant ordering system. If they press the no button it will then close the program but still saves the users data.
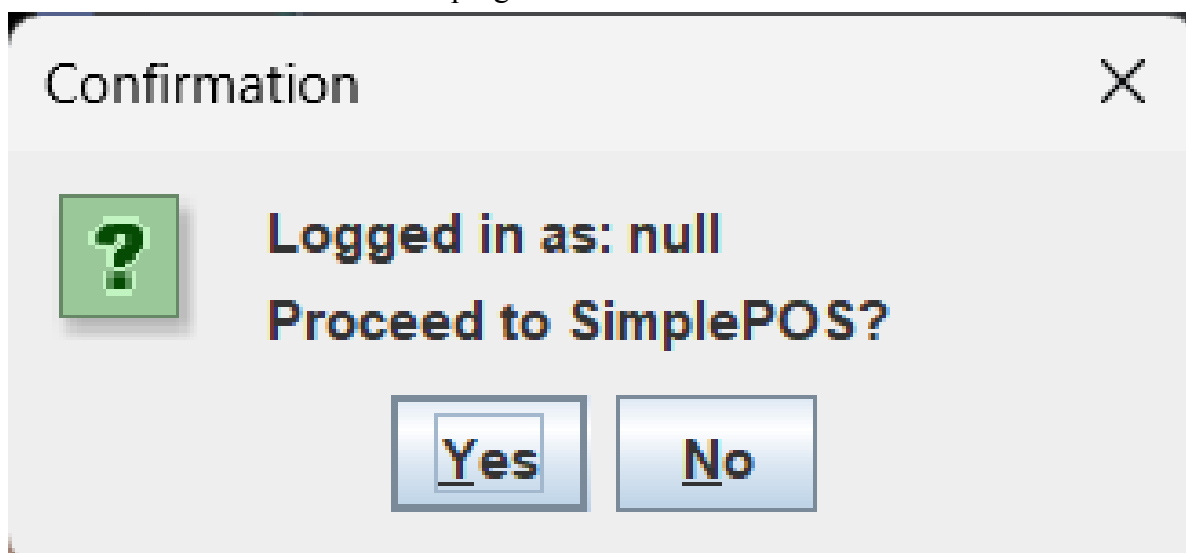
In this picture when the user is redirected to the Restaurant Ordering System they have the choice to choose as much food as they want as long as they have sufficient balance.
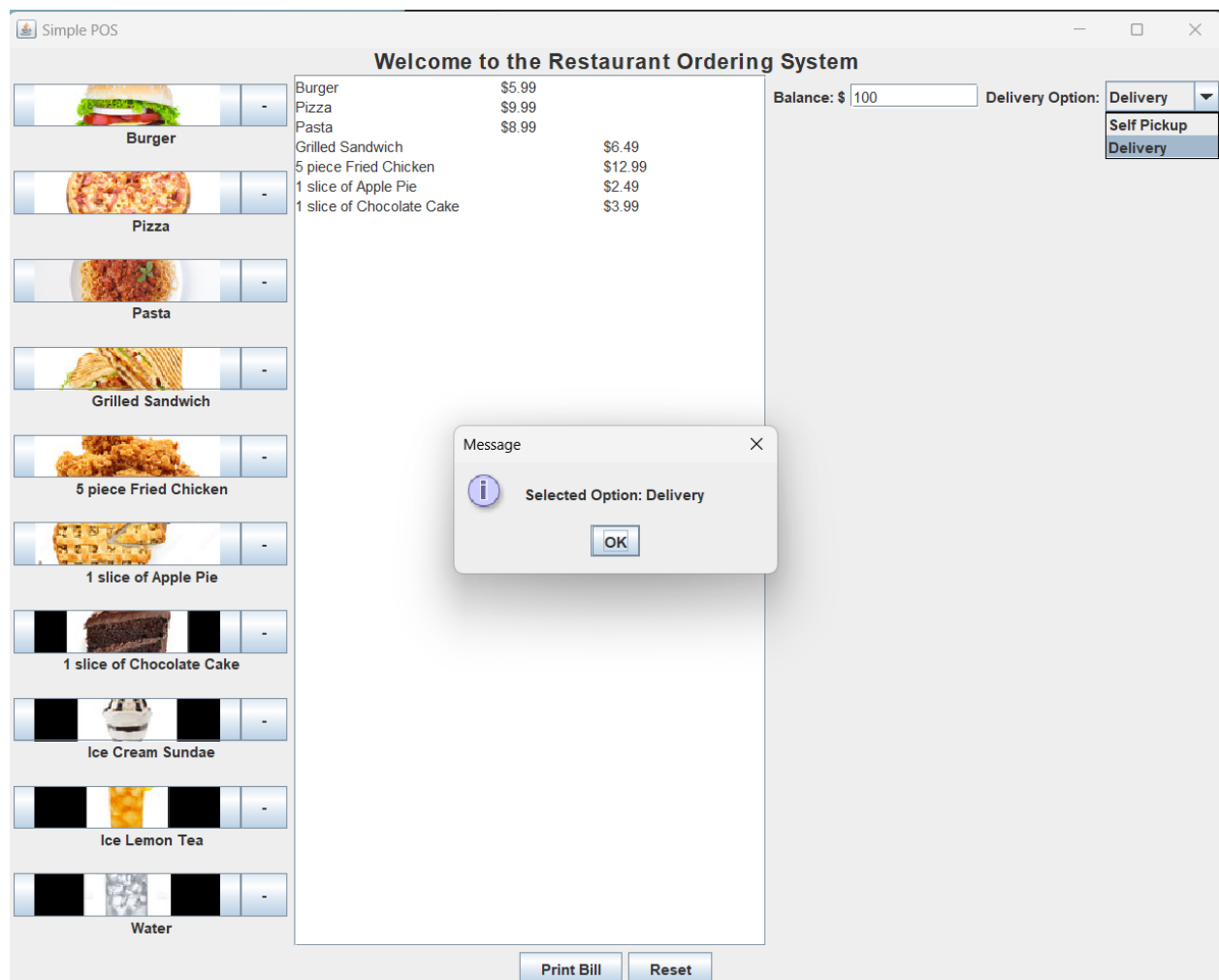
In this picture the user can also choose the delivery option be it self pickup or delivery.

In this picture after the user is done choosing all the things they want they can press the print bill button then it will show the bill to the user.

Message

========== Bill ==========
Date/Time: 15/06/2023 19:25:28

Burger$5.99
Pizza$9.99
Pasta$8.99
Grilled Sandwich$6.49
5 piece Fried Chicken$12.99
1 slice of Apple Pie$2.49
1 slice of Chocolate Cake$3.99

Total Amount: $50.93000000000001
============================
Balance: $100.0
Change: $49.06999999999999

OK
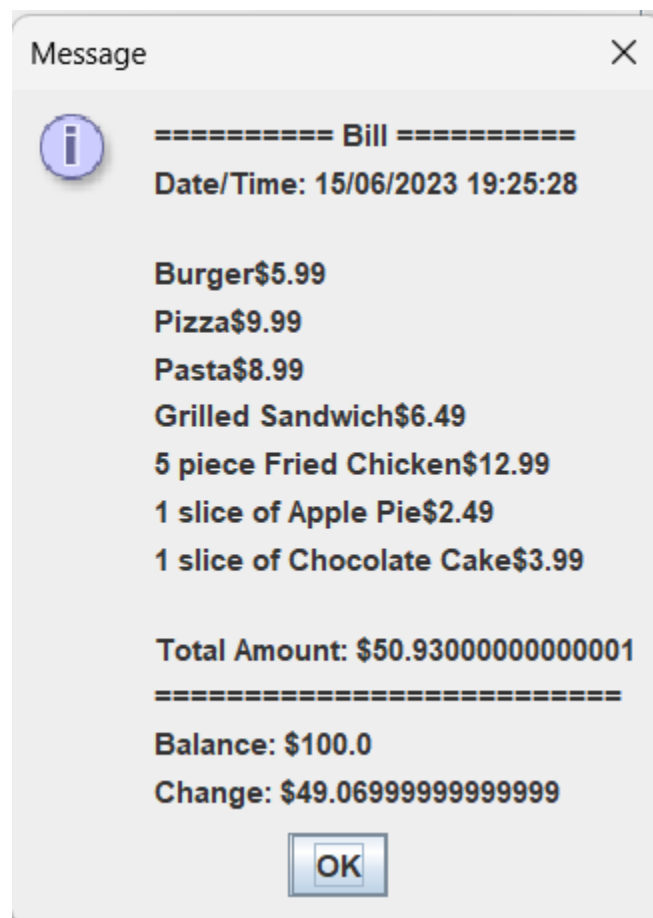
Chapter 5 – Lesson Learned/Reflection

From this final project, I can learn the usage of many new libraries and functions to create a working registration form, a login form, and a simplePOS system that represents a restaurant ordering system. It uses many imports to make it function properly, and not only that, I also have to download MySQL to store the data of the registered users using the application. This is a whole new experience for me, as I had to integrate myself to understand this language,

namely Java. I am proud of this experience, as I had many successful endeavors and also many failures making this program, but in the end I have completed this program to my satisfaction. It is also important to note that this code is nowhere near sophisticated, as I believe that this program can be learned easily with many guides on the internet, even for someone like me who is not familiar with the codes that make the program work.

Chapter 6 – Source Code and Video Demo

Source Code Link: https://github.com/AdyatamaMH/OOP-Final-Project

Video Demo Link:
https://drive.google.com/file/d/1zER-ForsFYwKsJ5pol-e_3-UKO0WqHE9/view?usp=sharing