****

**TEB 1043 / TFB 1033**

**Object-Oriented Programming**

**FINAL PROJECT GROUP JAVA CHIP**

**Lecturer: Dr Nordin Zakaria**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **ID No** | **Program** |
| 1 | INSYIRAH BINTI HAMZAH | 24000157 | Bachelor in Computer Science |
| 2 | HADHINA SYAHIRA BINTI MUHAMMAD HILMI | 24000291 | Bachelor in Computer Engineering |
| 3 | NUR ALYA SYAZWINA BINTI SAMSUL BAHRIN | 24000088 | Bachelor in Computer Engineering |
| 4 | NUR FATIHAH BINTI MOHD NOOR | 24000227 | Bachelor in Computer Science |

1. **INTRODUCTION**
2. Purpose

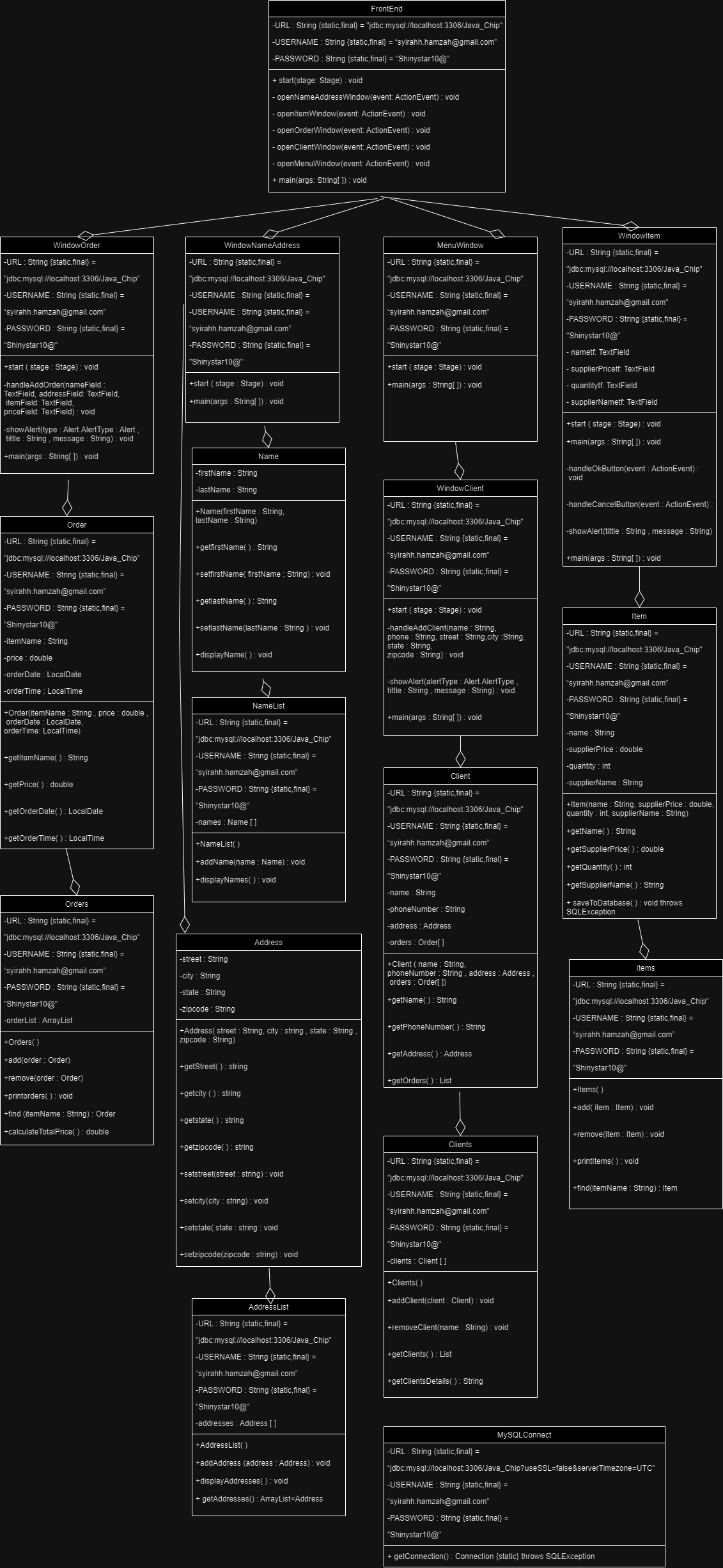
The purpose of this proposal is to outline the development of an object-oriented coffee shop management system named "Java Chip." This system aims to streamline the operations of a coffee shop, enhancing efficiency, customer experience, and overall business management through the use of modern software development practices.

1. Problem Statement

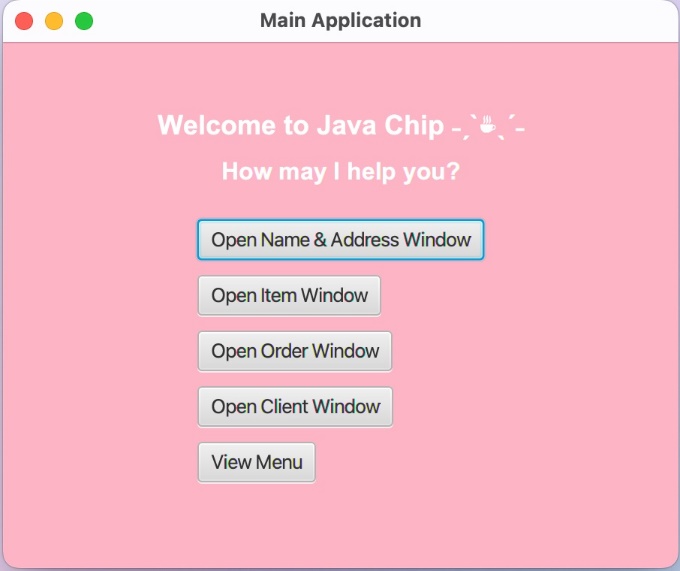
Running a coffee shop involves managing numerous tasks such as order processing, inventory management, staff scheduling, and customer loyalty programs. Traditional manual methods or disjointed software solutions can lead to inefficiencies, errors, and customer dissatisfaction. Java Chip aims to address these challenges by providing a unified and robust management system.

1. Solution

Java Chip will be developed as an object-oriented software application, leveraging the principles of encapsulation, inheritance, and polymorphism to create a modular, maintainable, and scalable system. This system will include features such as order management, inventory tracking, employee scheduling, customer management, and sales reporting. By integrating these functionalities into a single platform, Java Chip will improve operational efficiency and enhance the customer experience.

1. **UML DIAGRAM**
2. **CODING DESCRIPTION**
3. Frontend
4. Main Application

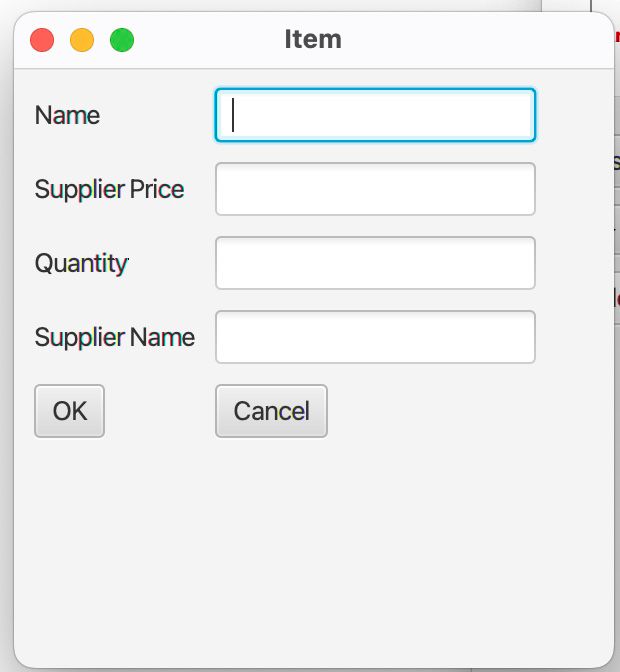
The interface that the user will see when opening the application.



*User’s interface*

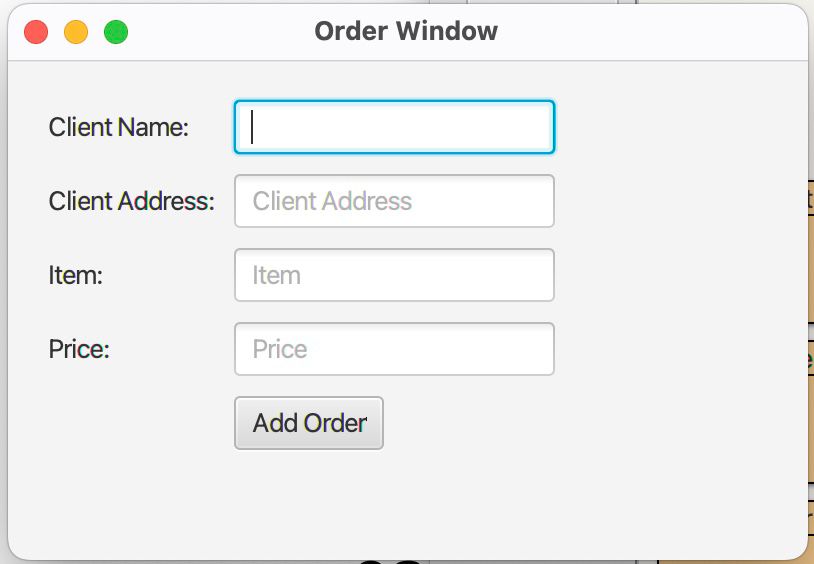
This allows users to directly choose which page they want to redirect to. Order window, item window, client window, name & address window or menu window.

1. Item window



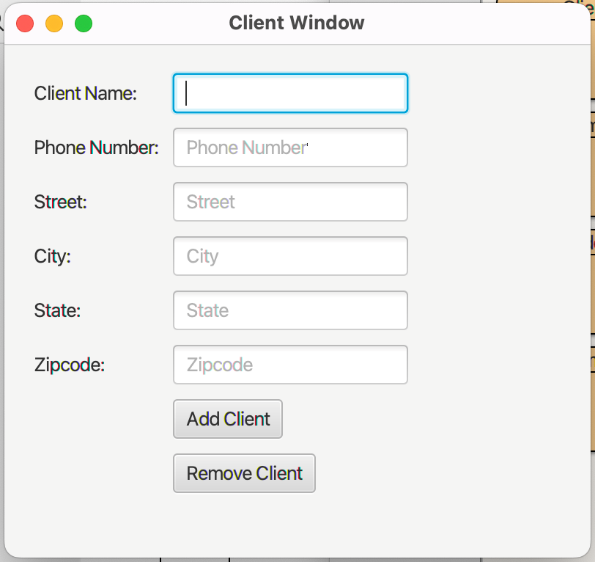
After clicking “Open Item Window”, users will see this window. They can fill up the details needed.

1. Order window



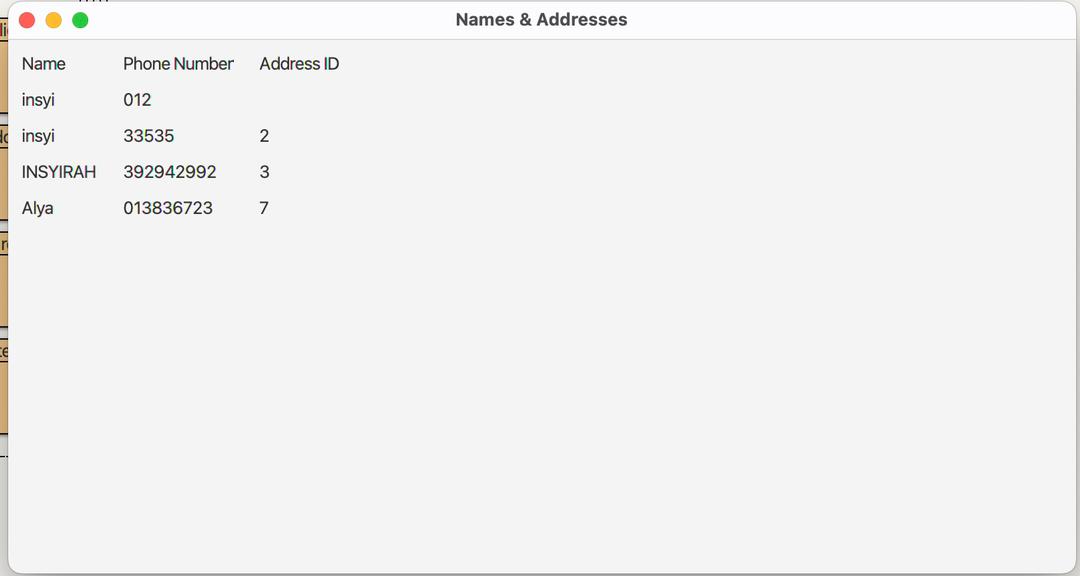
If users choose to click “Open Order Window”, they will be redirected to this window. Users can fill up the details needed regarding order.

1. Client window



If users click the “Open Client Window” button, this window will appear. Users can fill up the client details, add, and remove it into the client list database.

1. Name & Address window



If users click the “Open Name & Address Window” button, this window will appear. This list is used to store and manage contact details for clients.

1. Menu window



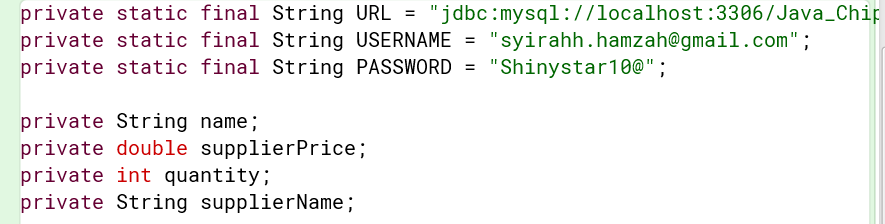
If users click the “View Menu” button, this window will appear. User may choose to order.

1. Backend

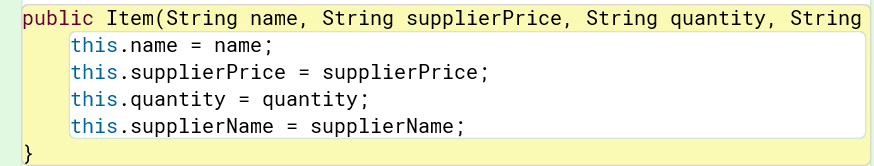
The data and infrastructure that makes the coffee shop application works.

1. Item class

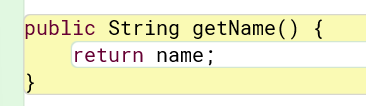
In this class, we build the backend of the Item Window. Each object will have these variables to store its name, supplier price, quantity, and supplier name.



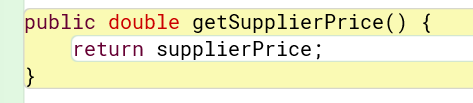
These are the attributes or properties of the Item class.



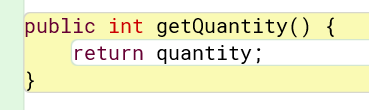
This is a constructor method and used to initialize the name, supplier price, quantity and supplier name.



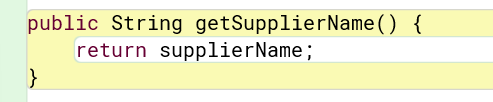
These methods allow external code to set the values of the name and return the current values when called.



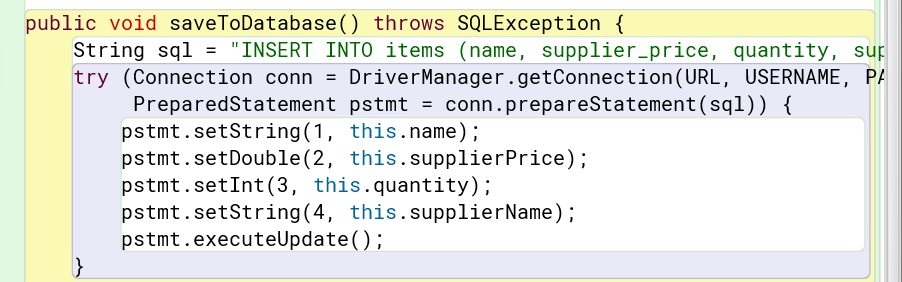
These methods allow external code to set the values of the supplier price and return the current values when called.



These methods allow external code to set the values of the quantity and return the current values when called.

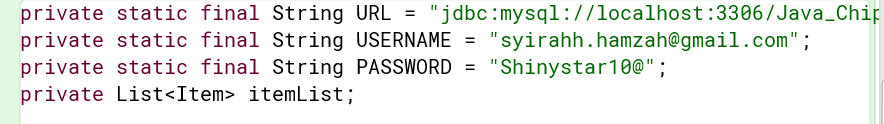


These methods allow external code to set the values of the supplier name and return the current values when called.

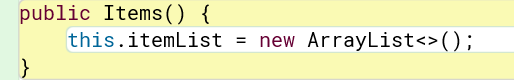


The saveToDatabase method adds data to a database. The methods set the values for each data using variable name, supplierPrice, quantity and supplierName.

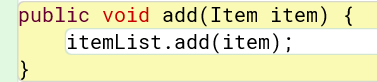
1. Items class



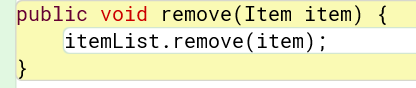
Declare a variable named ‘itemList’ and it will hold a collection of ‘Item’ objects



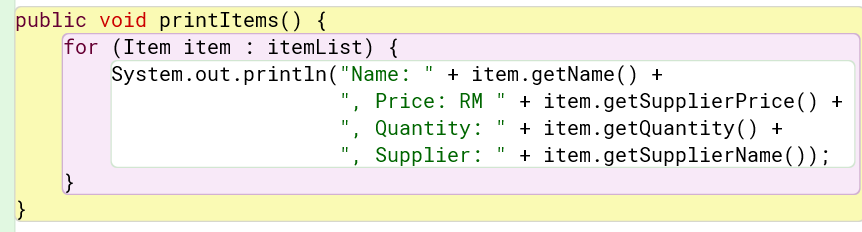
This constructor is used to initialize itemList as a type of ArrayList. This creates an empty list that will store ‘Item’ objects.



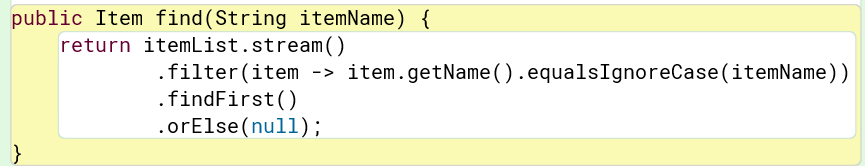
This method is used to add new value of an Item to the itemList.



This method is used to remove the value of Item from the itemList.



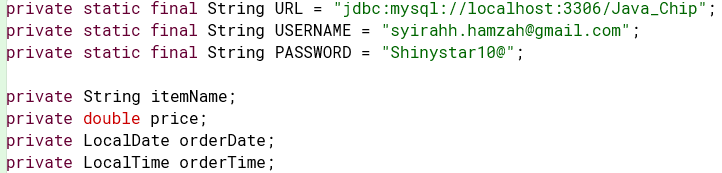
This method is used to print new Item to the console.

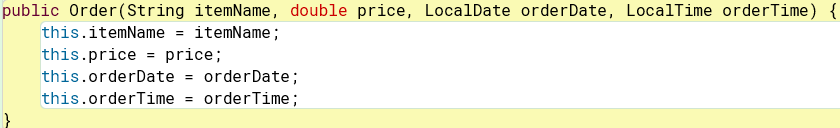


This method is used to find itemName in the itemList.

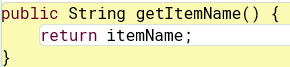
1. Order class

In this class, we build the backend of the Order Window. This class serves to hold data of item, price, date, and time of order taken.

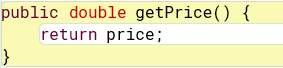
The first three lines of code define constants for connecting to a MySQL database using JDBC (Java Database Connectivity). The next lines are to declare the date type of the attributes.



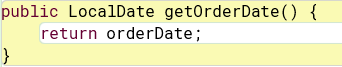
This constructor is used to initialize the item, price, date, and time attributes. As well as to allow pass value from users to the parameters in the ().



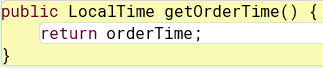
This method is used to retrieve value of itemName from the user.



This method is used to set value of the price and retrieve value of price from the user.

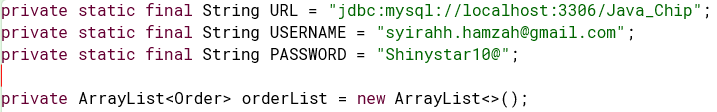


This method is used to retrieve value of date from the user.



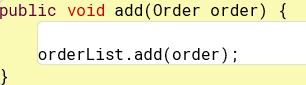
This method is used to retrieve value of time from the user.

1. Orders class

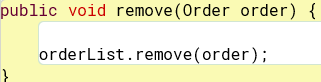
The first three lines of code define constants for connecting to a MySQL database using JDBC (Java Database Connectivity). Declaring orderlist as an array of list that holds the values of Order.



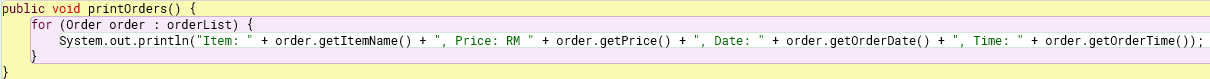
This default constructor serves to create instances of Orders class.



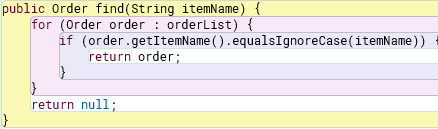
This method is used to add the new value of order into the orderList.



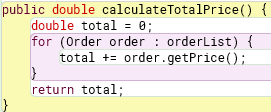
This method is used to remove the existing value of order from the orderList.



This method is used to compare the recently order input added with the existing data in orderList. If there is no match, this method will display the order added.



This method is used to find the name of order added in the orderList by comparing it with the existing itemName.

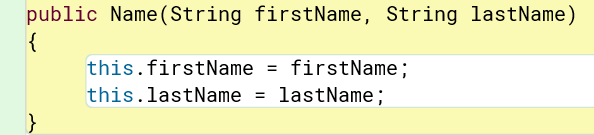


This method is used to calculate the total price of the orders and display it on Order Window.

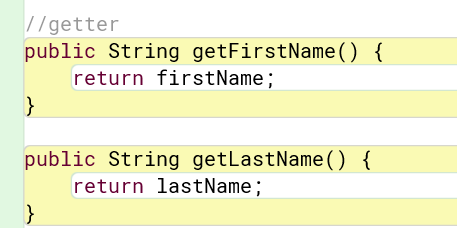
1. Name class



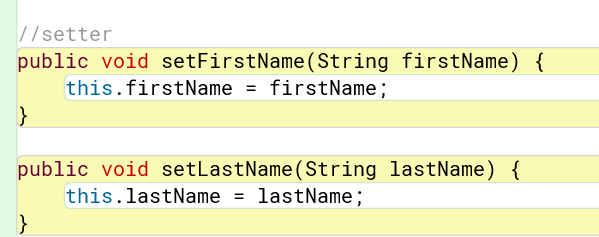
This line declares instances variable firstname and lastname of type String.



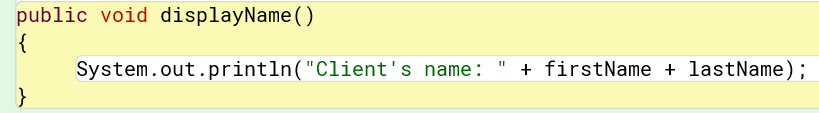
This is a constructor method and used to initialize the firstname and lastname.



These getter methods allow external code to set the values of the firstname and lastname and retrieve the value of firstname and lastname from the user.

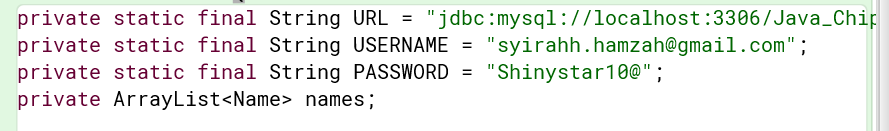


Setter methods are used to set or update the value of the firstname and lastname.

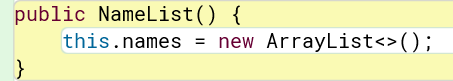


This method prints the firstname and lastname of the client to the console.

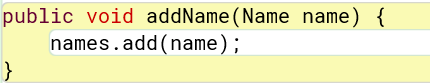
1. NameList class



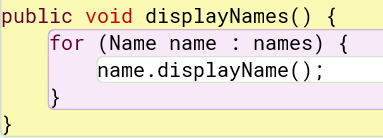
Declare a private variable named ‘names’ and it will hold a collection of ‘Name’ objects.



Declare a constructor for the nameList class and initialize ‘names’ as a type of ArrayList. This creates an empty list that will store ‘Name’ objects.

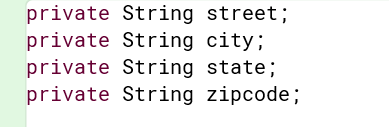


This method is used to add new Name to the names list.

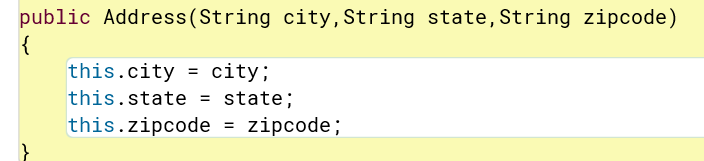


This method is used to show the name of the clients

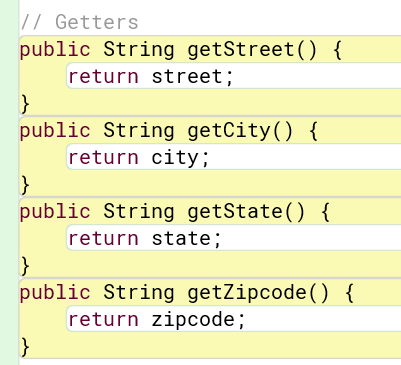
1. Address class



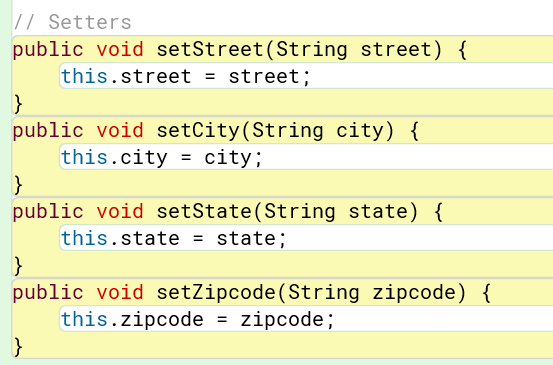
This line declares instances variable city, state and zipcode of type String.



This is a constructor method and used to initialize the city, state and zipcode.

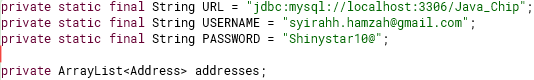


These getter methods allow external code to set the values of the street, city, state and zipcode and retrieve the value of street, city, state and zipcode from the user.



Setter methods are used to set or update the value of the street, city, state and zipcode.

1. AddressList class



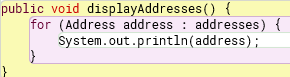
The first three lines of code define constants for connecting to a MySQL database using JDBC (Java Database Connectivity). Declaring addresses as an array of list that holds the values of Address.



This constructor is used to initialize addresses as an ArrayList.



This method is used to add address input into the addresses arraylist.

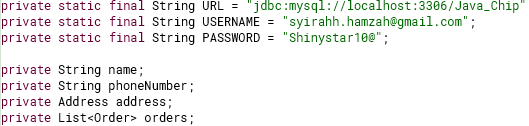


This method is used to display the address after comparing it with the existing address in addresses arraylist.

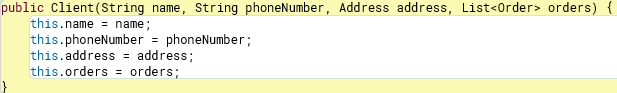


This method is used to retrieve the list of address in addresses.

1. Client class



The first three lines of code define constants for connecting to a MySQL database using JDBC (Java Database Connectivity). The next lines are to declare the date type of the attributes.

This constructor is used to pass value from input through the parameters and to initialize the attributes.



This method is used to retrieve the name value.



This method is used to retrieve the phoneNumber value.



This method is used to retrieve the address value.



This method is used to retrieve the value orders.

1. Clients class



The first three lines of code define constants for connecting to a MySQL database using JDBC (Java Database Connectivity). Declaring clients as an array of list that holds the values of Client.



This constructor is used to initialize clients as an arraylist.



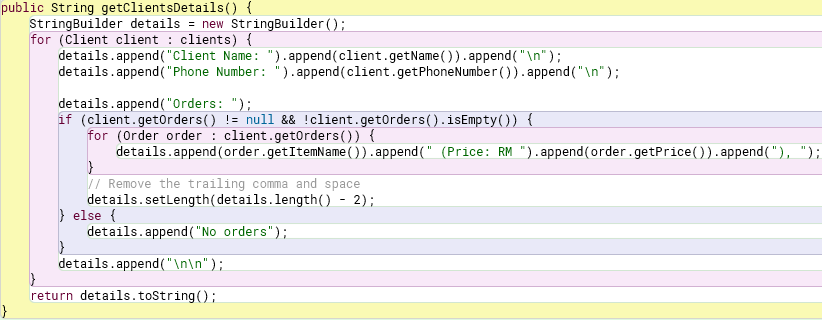
This method is used to add client input into the clients arraylist.



This method is used to remove client from the clients arraylist if same as name.



This method is used to return the new list of the clients.

This method is used to return a detailed string representation of all clients and their orders.

1. Database Table

A database table is a collection of data organized in rows and columns. Each table represents a specific type of data. Tables are fundamental to the structure of a relational database and are used to store and manage data efficiently.

1. Address



This table allows the user to manage information by editing item details, duplicating entries, or deleting them as needed. The interface is typical for managing a list of addresses, including cities, zip codes and streets.

1. Clients



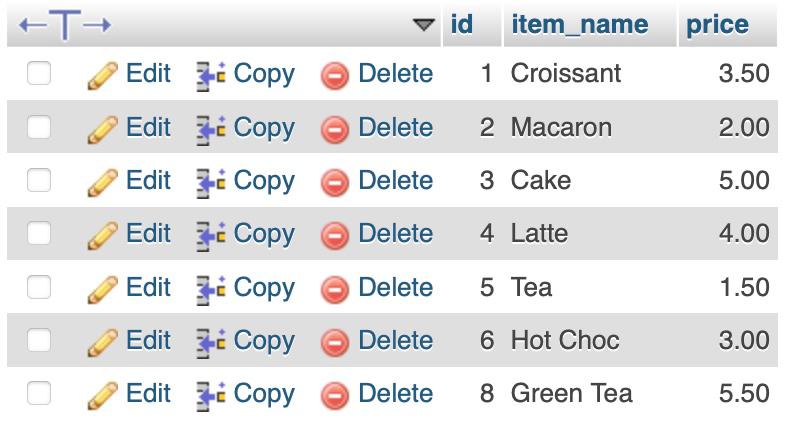
This table allows the user to manage contact information by editing details, duplicating entries, or deleting them as needed. The interface is typical for managing a list of contacts, including phone numbers and associated addresses. The presence of NULL in the address\_id column indicates that some entries may not have an address assigned.

1. Items



This table allows the user to manage inventory by editing item details, duplicating entries, or deleting them as needed. The interface is typical for managing a list of products or supplies, including tracking supplier information and stock quantities.

1. Menu



This table allows the user to manage items by editing their details, duplicating entries, or deleting them as needed. The interface is typical for managing a menu or product list in a small business, such as a cafe or bakery.