Faculty of Engineering, Science & Technology

IQRA UNIVERSITY



Data Structure and Algorithm

Project Report

[FALL-2023]

Name: MUHAMMAD ZARYAB

Roll no: 14444

Semester:4

Batch:

*Instructor Name:* Ghazala Shafi Sheikh

Acknowledgment

” The best teacher teaches from the heart not from the BOOK. This report consists of a project.”

*I would like to acknowledge and all thanks to my teacher who thought me this software and able me to use it and guide me through my project and help me so now I am able to use the software and perform some work related to this software.*

*Thank you*

*Miss Ghazala Shafi Sheikh*

***Chapter 1***

Introduction

***Chapter 2***

System Overview

***Chapter 3***

GUI Design

***Chapter 4***

Functionality

***Chapter 5***

DSA Concepts

***Chapter 6***

Implementation Details

***Chapter 7***

Conclusion

***Chapter 8***

Coding

***Chapter 9***

Screen Shots

**Project Report: Zeus Coffee Shop Application**

**1. Introduction**

The Zeus Coffee Shop Application is a JavaFX-based application designed to streamline the coffee ordering process. This report provides a detailed overview of the application's structure, features, implementation, and the incorporation of key Data Structures and Algorithms (DSA) concepts.

**2. System Overview**

The application consists of three main components: the login screen, menu exploration, and order processing. It maintains a menu of coffee items, handles user authentication, and manages orders. The underlying data structures ensure efficient processing and tracking of orders and inventory.

**3. GUI Design**

JavaFX is utilized for creating an intuitive and visually appealing GUI. The design includes input fields, buttons, and lists for seamless interaction. CSS styling is applied for a consistent and visually appealing user interface.

**3.1 Login Screen**

The login screen collects user credentials for authentication. It uses text fields for username and password, and a button triggers the login process.

**3.2 Coffee Shop Interface**

Upon successful login, users are presented with a comprehensive interface. It features two lists displaying the menu and current order. Buttons enable users to add items to the order, customize orders, and process the final order.

**4. Functionality**

The application offers various functionalities, including user authentication, menu exploration, order customization, and order processing.

**4.1 User Authentication**

Users, both administrators and regular users, can log in with their credentials. Admin access provides additional features like order customization.

**4.2 Menu Exploration**

The menu is initialized with various coffee items, each having a name and price. The menu is displayed using a ListView.

**4.3 Order Customization**

Users can customize orders, and administrators have additional privileges for advanced order customization.

**4.4 Order Processing**

Orders are processed, and the total amount is calculated. The order list is then cleared, and the inventory is updated.

**5. DSA Concepts**

The application incorporates the following Data Structures and Algorithms concepts:

**5.1 HashMap for Inventory**

A HashMap is used to efficiently store and manage the inventory of coffee items. Each item is associated with its quantity, allowing for quick retrieval and updates.

**5.2 ArrayList for Menu and Orders**

ArrayLists are employed to store menu items and order details. They provide dynamic resizing, essential for managing variable-sized lists.

**6. Implementation Details**

The application uses JavaFX for the GUI and incorporates various Java programming concepts. The Model-View-Controller (MVC) pattern is followed for a structured design.

**7. Conclusion**

The Zeus Coffee Shop Application not only provides an efficient and user-friendly solution for managing coffee orders but also showcases the application of fundamental Data Structures and Algorithms concepts. Its modular design, clear GUI, and integration of JavaFX elements contribute to a positive user experience.

**8. Coding**

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class ZeusCoffeeShopApp extends Application {

private ArrayList<MenuItem> menuItems = new ArrayList<>();

private ArrayList<OrderItem> orders = new ArrayList<>();

private Map<String, Integer> inventory = new HashMap<>();

private TextField usernameField;

private PasswordField passwordField;

private Button loginButton;

private ListView<String> menuListView;

private ListView<String> orderListView;

private Button addToOrderButton;

private Button customizeOrderButton;

private Button processOrderButton;

private Stage primaryStage;

private boolean isAdmin = false;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

this.primaryStage = primaryStage;

primaryStage.setTitle("Zeous Coffee Shop");

initializeMenu();

initializeInventory();

createLoginScreen();

}

private void createLoginScreen() {

VBox loginBox = new VBox(10);

loginBox.setPadding(new Insets(20, 20, 20, 20));

Label usernameLabel = new Label("Username:");

usernameField = new TextField();

Label passwordLabel = new Label("Password:");

passwordField = new PasswordField();

loginButton = new Button("Login");

loginButton.setOnAction(event -> handleLogin());

loginBox.getChildren().addAll(usernameLabel, usernameField, passwordLabel, passwordField, loginButton);

Scene loginScene = new Scene(loginBox, 300, 200);

loginScene.getStylesheets().add(getClass().getResource("black-theme.css").toExternalForm());

primaryStage.setScene(loginScene);

primaryStage.show();

}

private void initializeMenu() {

menuItems.add(new MenuItem("Espresso", 2.50));

menuItems.add(new MenuItem("Latte", 3.00));

menuItems.add(new MenuItem("Cappuccino", 3.50));

menuItems.add(new MenuItem("Mocha", 4.00));

menuItems.add(new MenuItem("Americano", 2.75));

}

private void initializeInventory() {

for (MenuItem menuItem : menuItems) {

inventory.put(menuItem.getName(), 10);

}

}

private void initializeCoffeeShopApp() {

BorderPane borderPane = new BorderPane();

menuListView = new ListView<>(FXCollections.observableArrayList(getMenuNames()));

orderListView = new ListView<>(FXCollections.observableArrayList(getOrderDetails()));

addToOrderButton = new Button("Add to Order");

customizeOrderButton = new Button("Customize Order");

processOrderButton = new Button("Process Order");

addToOrderButton.setOnAction(event -> handleAddToOrder());

customizeOrderButton.setOnAction(event -> handleCustomizeOrder());

processOrderButton.setOnAction(event -> handleProcessOrder());

VBox leftVBox = new VBox(new Label("Menu"), menuListView);

VBox rightVBox = new VBox(new Label("Order"), orderListView, addToOrderButton, customizeOrderButton, processOrderButton);

borderPane.setLeft(leftVBox);

borderPane.setRight(rightVBox);

Scene coffeeShopScene = new Scene(borderPane, 600, 400);

coffeeShopScene.getStylesheets().add(getClass().getResource("black-theme.css").toExternalForm());

primaryStage.setScene(coffeeShopScene);

primaryStage.show();

}

private ArrayList<String> getMenuNames() {

ArrayList<String> menuNames = new ArrayList<>();

for (MenuItem menuItem : menuItems) {

menuNames.add(menuItem.getName());

}

return menuNames;

}

private ArrayList<String> getOrderDetails() {

ArrayList<String> orderDetails = new ArrayList<>();

for (OrderItem orderItem : orders) {

orderDetails.add(orderItem.toString());

}

return orderDetails;

}

private void handleLogin() {

String username = usernameField.getText();

String password = passwordField.getText();

if (isValidAdmin(username, password)) {

isAdmin = true;

initializeCoffeeShopApp();

} else if (isValidUser(username, password)) {

isAdmin = false;

initializeCoffeeShopApp();

} else {

showAlert("Login Failed", "Invalid username or password. Please try again.");

}

}

private boolean isValidAdmin(String username, String password) {

// For simplicity, hardcode admin credentials

return username.equals("admin") && password.equals("adminpassword");

}

private boolean isValidUser(String username, String password) {

return username.equals("user");

}

private void handleAddToOrder() {

String selectedItem = menuListView.getSelectionModel().getSelectedItem();

if (selectedItem != null && inventory.get(selectedItem) > 0) {

orders.add(new OrderItem(selectedItem, 1, menuItems.stream()

.filter(item -> item.getName().equals(selectedItem))

.findFirst()

.orElse(null)));

updateOrderListView();

inventory.put(selectedItem, inventory.get(selectedItem) - 1);

}

}

private void handleCustomizeOrder() {

String selectedItem = orderListView.getSelectionModel().getSelectedItem();

if (selectedItem != null) {

OrderItem selectedOrderItem = orders.get(orderListView.getSelectionModel().getSelectedIndex());

if (isAdmin) {

showCustomizationDialog(selectedOrderItem);

} else {

showAlert("Access Denied", "You don't have permission to customize orders.");

}

}

}

private void handleProcessOrder() {

double totalAmount = calculateTotalAmount();

System.out.println("Total Amount: $" + totalAmount);

orders.clear();

updateOrderListView();

initializeInventory();

}

private double calculateTotalAmount() {

double total = 0;

for (OrderItem orderItem : orders) {

total += orderItem.getTotalPrice();

}

return total;

}

private void updateOrderListView() {

orderListView.setItems(FXCollections.observableArrayList(getOrderDetails()));

}

private void showCustomizationDialog(OrderItem orderItem) {

showAlert("Customization Options", "Implement customization options here.");

}

private void showAlert(String title, String content) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(content);

alert.showAndWait();

}

private static class MenuItem {

private String name;

private double price;

public MenuItem(String name, double price) {

this.name = name;

this.price = price;

}

public String getName() {

return name;

}

public double getPrice() {

return price;

}

}

private static class OrderItem {

private String name;

private int quantity;

private MenuItem menuItem;

public OrderItem(String name, int quantity, MenuItem menuItem) {

this.name = name;

this.quantity = quantity;

this.menuItem = menuItem;

}

public String getName() {

return name;

}

public int getQuantity() {

return quantity;

}

public double getTotalPrice() {

return quantity \* menuItem.getPrice();

}

@Override

public String toString() {

return String.format("%s x%d - $%.2f", name, quantity, getTotalPrice());

}

}

}

**9. Screen Shots**







