**Name: Anamika Dhar**  **Id: 221-15-4855**

|  |
| --- |
| Food  -name:String  -price:double  +getName():String  +getPrice():double |

**UML:**

|  |
| --- |
| *Intro*  *+intro\_*text*():void* |

|  |
| --- |
| Restaurant  -menu:Map<Integer,Food>  -orderHistoryFile:String  +Restaurant(orderHistoryFile:String)  +addFood(id:int,food:Food):void  +displayMenu():void  +getFoodById(foodId:int):Food  placeOrder():void  +saveOrderHistoryToFile(filename:String,  customerName:String,orderdetails:string):void  +displayOrderHistory(filename:String):void  +deleteOrderHistory(filename:String):void  +intro\_text():void |

|  |
| --- |
| Main  -MENU\_FILE:String  -ORDER\_HISTORY\_FILE:String  +main(String[]):void |

**Code:**

// imported all libraries

import java.io.\*;

import java.time.Duration;

import java.time.LocalDateTime;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

import java.util.ArrayList;

import java.util.List;

// Abstract class for providing introduction text

abstract class Intro {

public abstract void intro\_text();

}

// Serializable class representing a food item

class Food {

private String name;

private double price;

public Food(String name, double price) {

this.name = name;

this.price = price;

}

public String getName() {

return name;

}

public double getPrice() {

return price;

}

}

// Restaurant class representing a restaurant with menu and order history functionality

class Restaurant extends Intro {

private Map<Integer, Food> menu; // Map to store menu items

private String orderHistoryFile; // File name for order history

public Restaurant(String orderHistoryFile) {

menu = new HashMap<>(); // Initialize the menu map

this.orderHistoryFile = orderHistoryFile; // Set the order history file name

}

// Add a food item to the menu

public void addFood(int id, Food food) {

menu.put(id, food);

}

// Display the menu with food items and their prices

public void displayMenu() {

System.out.println("Menu:");

for (Map.Entry<Integer, Food> entry : menu.entrySet()) {

int foodId = entry.getKey();

Food food = entry.getValue();

System.out.println(foodId + ". " + food.getName() + " - $" + food.getPrice());

}

}

// Get a food item from the menu by its ID

public Food getFoodById(int foodId) {

return menu.get(foodId);

}

// Place an order by taking input from the user and saving order history

public void placeOrder() {

Scanner scanner = new Scanner(System.in);

double totalPrice = 0;

System.out.print("Enter your name: ");

String customerName = scanner.nextLine();

while (true) {

System.out.print("Enter the food ID (0 to exit): ");

int foodId = scanner.nextInt();

if (foodId == 0) {

break;

}

System.out.print("Enter the quantity: ");

int quantity = scanner.nextInt();

scanner.nextLine(); // Consume newline character

Food food = getFoodById(foodId);

if (food != null) {

totalPrice += food.getPrice() \* quantity;

String orderDetails = "Food: " + food.getName() + ", Quantity: " + quantity;

saveOrderHistoryToFile(orderHistoryFile, customerName, orderDetails);

} else {

System.out.println("Invalid food ID. Please try again.");

}

}

System.out.println("Order placed for " + customerName);

System.out.println("Total Price: $" + totalPrice);

LocalDateTime orderTime = LocalDateTime.now();

LocalDateTime deliveryTime = orderTime.plusMinutes(30);

Duration waitingTime = Duration.between(orderTime, deliveryTime);

long minutes = waitingTime.toMinutes();

System.out.println("Estimated waiting time: " + minutes + " minutes.");

}

// Save the order history to a file

public void saveOrderHistoryToFile(String filename, String customerName, String orderDetails) {

try (PrintWriter writer = new PrintWriter(new FileWriter(filename, true))) {

writer.println("Customer: " + customerName);

writer.println(orderDetails);

writer.println();

System.out.println("Order history saved to file: " + filename);

} catch (IOException e) {

System.out.println("Error saving order history to file: " + e.getMessage());

}

}

// Display the order history from a file

public void displayOrderHistory(String filename) {

try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {

System.out.println("Order History:");

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

} catch (IOException e) {

System.out.println("Error reading order history file: " + e.getMessage());

}

}

// Delete the order history file

public void deleteOrderHistory(String filename) {

try (PrintWriter writer = new PrintWriter(new FileWriter(filename))) {

System.out.println("Order history deleted.");

} catch (IOException e) {

System.out.println("Error deleting order history: " + e.getMessage());

}

}

// Implementation of the abstract method in the Intro class to provide introduction text

public void intro\_text() {

System.out.println("-------------Welcome to our Restaurant!---------------");

System.out.println("------------Enjoy your dining experience!-------------");

System.out.println("------------------------------------------------------");

}

}

// Main class for running the restaurant application

public class Main {

private static final String ORDER\_HISTORY\_FILE = "order\_history.txt";

public static void main(String[] args) {

Restaurant restaurant = new Restaurant(ORDER\_HISTORY\_FILE);

// Add food items to the menu

restaurant.addFood(1, new Food("Burger", 5.99));

restaurant.addFood(2, new Food("Pizza", 8.99));

restaurant.addFood(3, new Food("Salad", 6.99));

restaurant.addFood(4, new Food("Pasta", 7.99));

restaurant.addFood(5, new Food("Sandwich", 4.99));

// Add more food items...

Scanner scanner = new Scanner(System.in);

// while option user in the system

while (true) {

// Options for the user

List<String> options = new ArrayList<>();

options.add("Order Food");

options.add("View Previous Order History");

options.add("Delete All Order History");

options.add("Exit");

restaurant.intro\_text();

for (int i = 0; i < options.size(); i++) {

System.out.println((i + 1) + ". " + options.get(i));

}

System.out.println("------------------------------------------------------");

System.out.println("------------------------------------------------------");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline character

switch (choice) {

case 1:

restaurant.displayMenu();

restaurant.placeOrder();

break;

case 2:

restaurant.displayOrderHistory(ORDER\_HISTORY\_FILE);

break;

case 3:

restaurant.deleteOrderHistory(ORDER\_HISTORY\_FILE);

break;

case 4:

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

}