

Foodpanda

**Online food ordering application**

Section 2G

Daaim Ali 22K-4363

Abdul Ahad 22K-4221

Sohaib Baig 22k-4204

Submitted To: Miss Sumaiyah Zahid



Introduction:

Our food ordering program, developed as part of the Object-Oriented Programming course, is a GUI based application that streamlines the process of ordering food. This program utilizes the principles of object-oriented programming, with a particular focus on inheritance, to create a modular and extensible solution.

The primary objective of our program is to provide a user-friendly interface for customers to browse through a variety of food items, customize their orders, and place them conveniently. By employing object-oriented techniques, we have ensured that the program is easily maintainable, scalable, and adaptable to future enhancements.

HOW TO RUN THE PROGRAM:

1. Run the **main.exe** file located in the **Executable** folder:

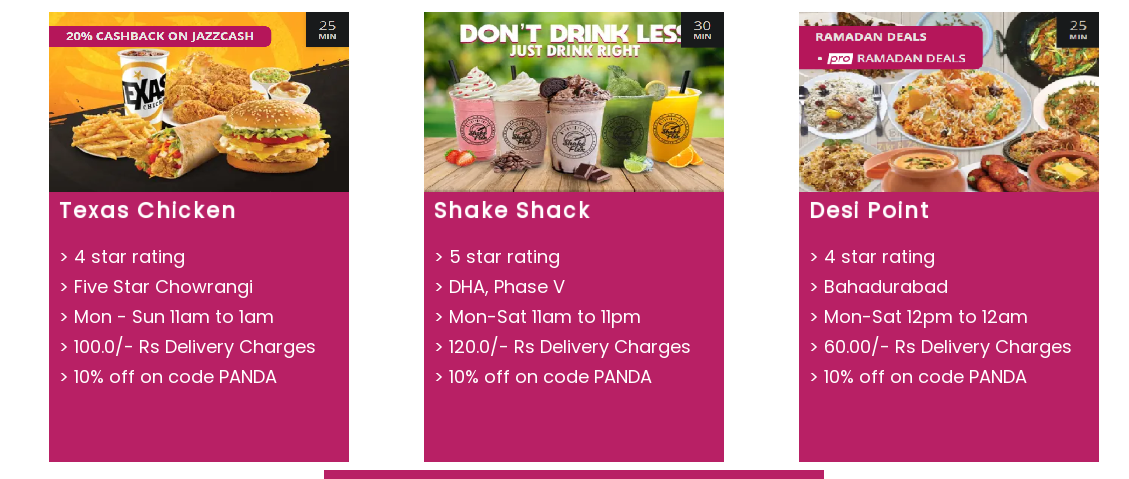

1. To locate your customer bills, open the **Database** folder in the **Executable** folder:



**Functionality:**

**The functionality of our food ordering program can be summarized into several distinct steps:**

1. Firstly, the program allows users to select their desired restaurant from a list of available options. This selection determines the menu and food items that will be displayed to the user.

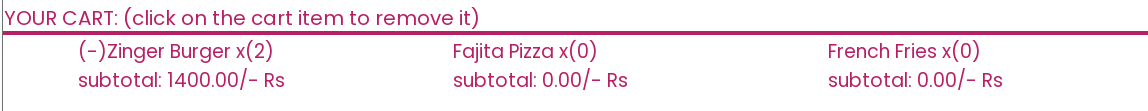


1. Once a restaurant is chosen, the user can browse through the menu and select their preferred food items. The program enables them to add these items to their virtual shopping cart.

Eg:- Texas Chicken



1. The shopping cart provides users with the flexibility to update their order at any time. They can add or remove items, modify quantities, and see the updated total price. This feature ensures that users have complete control over their order before proceeding to checkout.



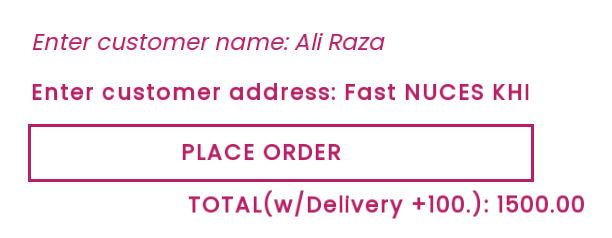
1. To enhance the user experience, our program incorporates a coupon feature. Users can enter a coupon code, which, if valid, will apply a discount to the total price. This feature adds an element of savings and incentivizes users to make use of available coupons.



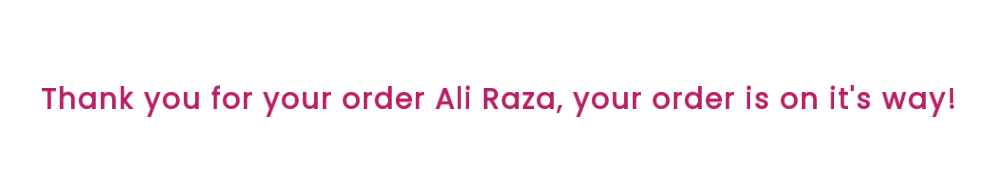
1. Throughout the ordering process, the program continuously displays the total price in the cart. This allows users to have a clear understanding of their accumulated costs and make informed decisions accordingly.

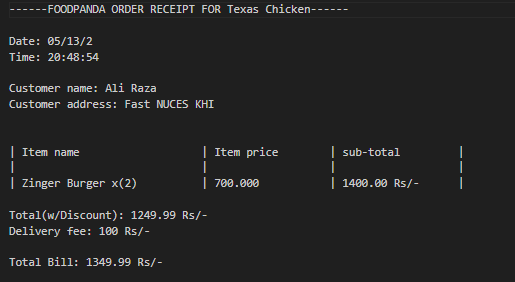


1. Upon reaching the checkout stage, the program prompts the user to provide their name and address. This information is essential for processing the order and ensuring accurate delivery.

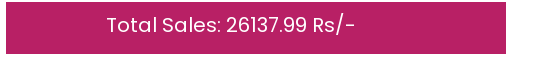


1. Once the user has entered their details, they can press the "Place Order" button. At this point, the program generates a bill by utilizing file handling techniques. This bill contains all the necessary order details, including the selected food items and quantities, the time & date of the order, prices, the final total, Restaurant’s name and the user's name and address.



Sample order bill:  


1. Upon each order placement, the total of that order(excluding the delivery) is added to the Total sales of the app. The total sales are saved in a totaldb.bin file so that the total amount is not lost when the app closes. Upon startup the total stored in totaldb.bin is fetched and displayed on the home screen. If the file is not found, it is created and the total is set to 0.


* In summary, our food ordering program offers the following functionality: selecting a restaurant, choosing food items and adding them to the cart, updating the cart, applying coupon codes for discounts, displaying the total price in real-time, entering name and address for delivery, generating a bill and storing the total sales amount through file handling, and finally, placing the order. These features combine to create a user-friendly and efficient food ordering experience.

OOP Implementation:

Total Classes: **17**

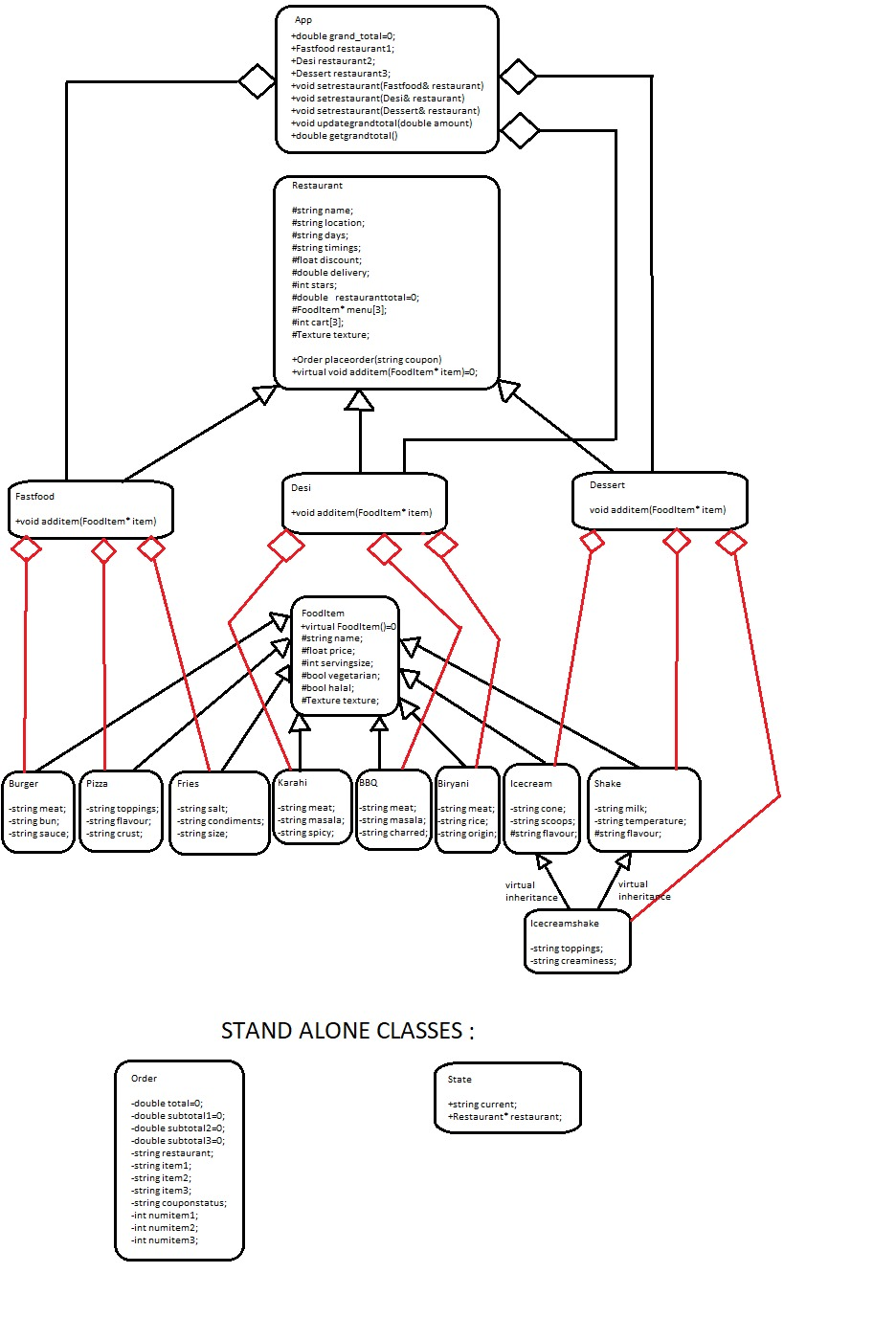
Class Names:

| App  Food Item  Order  Restaurant (Virtual)  Burger  Pizza  Fries  Fast Food  Karahi  BBQ | Biryani  Desi  Icecream  Shake  Icecream Shake  Dessert  State |
| --- | --- |

Inheritance:

* App <>- Dessert, Desi, FastFood
* Restaurant <- Dessert, Desi, FastFood
* FoodItem <- Burger, Fries, Pizza, BBq, Karahi, Biryani, Icecream, shake, IcecreamShake

Class Diagram:



Diamond Problem:

Since there were some common properties between the **Ice-cream**, **Shake**, and **Icecream-shake** classes, so in order to minimize code duplication we made the **Icecream-shake** class inherit from both **Icecream** and **Shake** classes, this consequently resulted in a diamond problem since both these parent classes inherited from a common grand-parent, the FoodItem class.





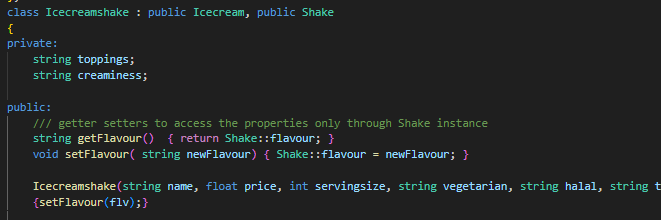


Solution:

* Using virtual inheritance we gave the grand-parent(**FoodItem**)’s properties to the **Icecream-shake** class easily without getting an “ambiguous reference to property” error.
* We also gave the common properties of the **Icecream** and **Shake** classes to the **Icecream-Shake** class and we set/accessed these properties through the instance of Shake object using the global scope resolution operator.





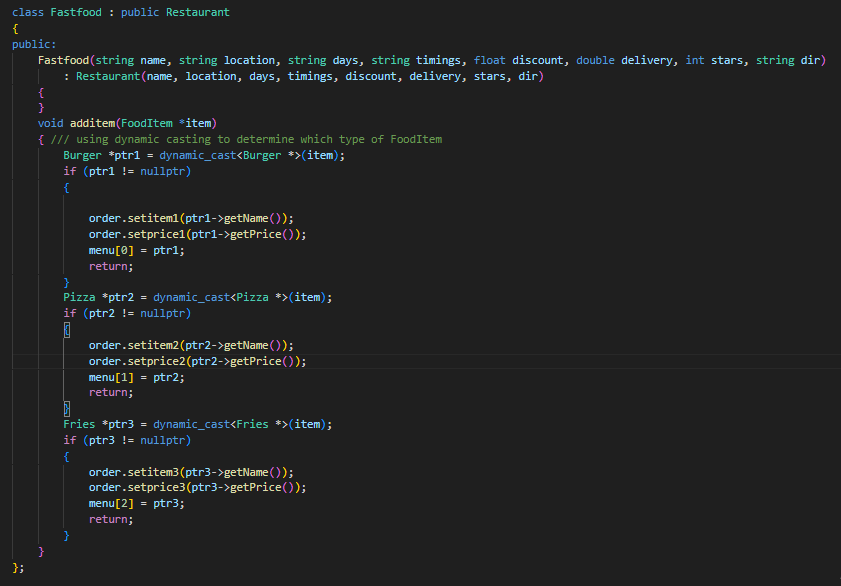


DYNAMIC CASTING

**We have utilized dynamic casting in a pure virtual function of the base Restaurant class, called additem, and the unique implementation of this method is defined in each of the Restaurant TYPE classes; Dessert, FastFood, and Desi.**

This method takes in a Pointer of the base classFoodItem, from which every child food class inherits, such as the Burger class. In this way the function can accept a pointer of any type of food item.

Further on, it declares a pointer of each of the corresponding food items of the restaurant, eg:- for the FastFood class’s implementation, the pointers of Burger, Pizza, and Fries classes would be made and then the received argument pointer would be dynamic casted into each of these pointers.

All the cases where a null pointer was returned will be ignored and in the one case in which a correct pointer will be returned, that pointer will be stored in the menu of the restaurant.  
  


***-An example implementation of the additem function in the FastFood class.***

The Graphical User Interface:

**We have utilized the SFML library to program the GUI of our Foodpanda application.**

* The library files can be found in the lib & include folders in the src folder:



* The library uses a RenderWindow class’s object to instantiate a window to display the contents on.
* SFML uses a continuous while loop to update/populate the window contents after each iteration.
* We have used the objects of the basic shape classes such as RectangleShape class, and other utility classes such as Text/Texture/Image etc. for each of the details you may observe in the app.
* STATE MANAGEMENT:

In order to logically and feasibly decide on runtime as to what content should be displayed, we have created a state class with properties such as the current state and the chosen restaurant, which enables the program to display the details accordingly. With the help of this functionality we have easily implemented a navigation functionality so that the user can easily go back and forth between multiple pages.

**Final Remarks:**

**Overall SFML was a very simple and intuitive GUI library to work with, and it enabled us to program this app with a visually appealing and user friendly interface.**

**REFERENCES & LINKS:**

* [**https://www.sfml-dev.org/**](https://www.sfml-dev.org/)
* [**https://www.youtube.com/playlist?list=PL6xSOsbVA1eaJnHo\_O6uB4qU8LZWzzKdo**](https://www.youtube.com/playlist?list=PL6xSOsbVA1eaJnHo_O6uB4qU8LZWzzKdo)
* **Images copyrights owned by: (**[**https://www.foodpanda.pk/**](https://www.foodpanda.pk/)**)**