Description

“ChefSol”



So, basically, we have incorporated the well-known Uber system with Chefs. We are using four classes, namely:

* Chef
* User
* Admin
* Payment

Starting with **Chef class**, the chef class contains information of the Chef that are registered with our platform, that is the ChefSol. The informations are mainly the city they reside in, their name, user id, password, their charges per hour, the cuisine they cook, and their availability status. There are multiple methods to get and set these values. Constructors too are used. Apart from these general methods, there are methods of displaying their information which can be called by Chef class, as well as the Admin and User class.

The purpose of the "CHEF" class is to represent a chef and provide methods for managing chef-related information. It allows for creating chef objects with various attributes such as name, availability, city, cuisine, experience, hourly rate, ID, and password.

The class provides methods to set and get values for these attributes, such as **set\_name()**, **set\_availability()**, **set\_city()**, **set\_cuisine()**, **set\_experience()**, **set\_hourly\_rate()**, **set\_ID()**, **set\_password()**, and their corresponding getter methods.

The class also includes methods to view and display chef details. The **get\_chef\_details()** method displays the information of a chef object, and **view()** and **view3()** methods allow viewing the details of all chefs or specific chefs based on certain criteria (e.g., cuisine, city, availability). The **update()** method is used to update the attributes of a chef object.

Additionally, the class provides functionality to write chef information to a file (**write()**) and read chef information from a file to perform operations such as viewing and updating (**view2()**).

Overall, the class encapsulates the attributes and behaviors of a chef, allowing for creating, managing, and interacting with chef objects in a structured manner.

Moving on to the **User class**, the User class handles the customers that visit our platform. The purpose of the "USER" class is to represent a user and provide methods for managing user-related information. It contains attributes such as name, contact number, city, ID, password, wallet balance, and a pointer to a payment object.

The class includes various methods to perform operations related to user management. Some of the important methods are:

**USER()**: The constructor that initializes the attributes of the user object with default values.

**get\_wallet()**: Returns the wallet balance of the user.

**updateWallet()**: Allows the user to update their wallet balance by adding a specified amount.

**payment()**: Handles the payment process by interacting with a chef object.

**get\_user\_details()**: Displays the details of the user object.

**view3()**: Allows viewing user details based on various criteria such as name, contact number, and city.

**view2()**: Checks if the provided ID and password match the user's credentials and returns true or false accordingly.

**getID()** and **getPassword()**: Getter methods to retrieve the user's ID and password.

**setID()**: Sets the user's ID.

**rent\_chef()**: Facilitates the process of booking a chef by interacting with a chef object.

**check\_chef()**: Helps the user in selecting a chef based on cuisine preference and city.

**manage\_wallet()**: Placeholder method for managing the user's wallet balance.

Setter methods (**set\_name()**, **set\_number()**, **set\_city()**, **set\_password()**, **set\_wallet()**) to update the respective attributes of the user.

File handling methods (**write()**, **view()**, **update\_wallet\_file()**) to write user information to a file, view user details from a file, and update the user's wallet balance in the file.

Overall, the class encapsulates the attributes and behaviors of a user, allowing for creating, managing, and interacting with user objects in a structured manner.

Now, talking about the use of **Admin class**, it is the root that has the command over the Users, as well as the Chefs. The purpose of the **ADMIN** class given above is to provide functionality related to managing coupons. The class represents an administrator and includes methods to add coupons, delete coupons, and read coupons from a file. It also includes getter methods to access the administrator's username and password.

The class has private member variables **user\_name** and **password** to store the credentials of the administrator. The constructor initializes these variables with specific values.

The **add\_coupon** method allows the administrator to add a new coupon code to a file named "coupon.txt". The method prompts the administrator to enter a coupon code in a specific format and appends it to the file.

The **delete\_coupon** method allows the administrator to delete a coupon code from the "coupon.txt" file. The method prompts the administrator to enter the coupon code they wish to delete, and it creates a temporary file ("tempfile.txt") to store the contents of the original file excluding the specified coupon code. Finally, it replaces the original file with the updated contents from the temporary file.

The **read\_coupon** method has two different functionalities based on the **flag** parameter. If **flag** is 0, it reads and displays all the coupon codes from the "coupon.txt" file. If **flag** is 1, it searches for a specific coupon code (**x**) in the file and returns the corresponding discount percentage. If the coupon code is not found, it returns 0.

The getter methods **get\_user\_name** and **get\_password** allow access to the administrator's username and password outside the class.

The purpose of the **PAYMENT class** given above is to handle the payment process for chef bookings made by users. The class includes methods for processing payments, printing receipts, applying coupons, and checking wallet balances.

The class has private member variables such as **user** (of type USER), **chef** (of type CHEF), **time\_of\_order**, **hours**, **amount**, **payment\_method**, **a1** (of type ADMIN), and **discount**. These variables store relevant information related to the payment process.

The **wallet\_process** method is responsible for checking the user's wallet balance and performing the necessary transaction. It takes an **ID** parameter and reads the user's information from a file. If the user's wallet balance is sufficient for the payment amount, it updates the wallet balance and returns **true** to indicate a successful transaction. Otherwise, it returns **false**.

The **print\_receipt** method prints a receipt for the chef booking. It displays information such as the chef's name, ID, cuisine, payment method, hours rented, total amount, and order time.

The **process\_payment** method handles the overall payment process. It takes a **CHEF** object and an **ID** parameter. It prompts the user to enter the number of hours they want to book the chef for, calculates the total amount based on the hourly rate, and offers the option to apply a coupon for a discount. It then prompts the user to select a payment method: credit/debit card, wallet, or cash on delivery. If the user chooses the wallet option, it calls the **wallet\_process** method to check the wallet balance and perform the transaction. Finally, it calls the **print\_receipt** method to generate and display the receipt.

The **apply\_coupon** method allows the user to apply a discount coupon to the booking. It prompts the user to input a discount code, calls the **read\_coupon** method of the **ADMIN** class to validate the coupon, and returns the discount percentage.

The **USER** class also includes a **payment** method that creates an instance of the **PAYMENT** class (**p1**) and calls its **process\_payment** method to initiate the payment process for a chef booking.

Apart from all these essential classes, we have tried to make an attractive entrance and exit to our program. Mainly the function printHeading() and the commands that display GoodBye when the person exits the code.

Our main function interacts with the user of this program, may it be Customer, Admin, or Chef, and the user can perform functionalities accordingly. The user is given a menu driven interface asking the user which functionalities he/she wants to perform. We have used a while loop, so that the program does not end unless the user has finished all the tasks he/she wanted to do.

That’s all about the ChefSol briefly. We would be happy to give you a demo of our ChefSol for better understanding. Thank you.