**Final Project:**

***Restaurant Management***

This project is a partial user/customer-based project. It allows a user to choose between modifying the employee details as admin and ordering the food as a customer and calculating the final bill for the order. It also assigns every employee a different order which is supposed to prepared by a single employee.

**Classes:**

**Restaurant (**main class**):**  This class is just a base/starter class for the program. Basically, for creating the system and providing the user with options to continue as an admin or a customer.

**EmployeeList:** It’s a parent class for employee as a private member. Any function related to any modification to the employee list like adding new employee or removing the old ones, is included in this class.

**Employee**: It’s a class for taking details of employee from the user and creating a list using parameters as name, id, age, and salary.

**OrderList:** It’s a parent class for order as a private member. Adding order, removing order for a customer (from the list) are the main functions of this class.

**Order :** class creates a listfor orders, that the customer wants to order and all the details related to the order are taken as an input such as, order id, item ordered, cost of item, final bill calculation.

Key differences from Preliminary project:

* **Original Declaration:**

The basic idea was to create a code where the admin of a restaurant can manage the details of the employee and change or modify the list of employees working in a restaurant customer can order a food and generate a final bill.

#include <iostream>

class Restaurant {

private:

int choice;

public :

void select();

void print();

}

void select(){

//Select the choice for admin page or placing order

}

void print(){

//printing the choices and

}

class employee {

private :

char \*ename;

int eid;

double salary;

char \*position;

public :

void employee();

void add\_emp();

void delete\_emp();

}

void employee(){

//Details for the employee such as name, id , salary,etc

}

void add\_emp(){

//Adding new employee

}

void delete\_emp(){

//removing the employees

}

class order{

private :

int order\_id;

char \* order;

public :

void add\_order();

void remove\_order();

void print\_menu();

}

void add\_order(){

//Placing orders and addding to it

}

void remove\_order(){

//removing the orders

}

void print\_menu(){

//printing the menu

}

class Bill : public order{

private :

float price;

float tax;

public :

void print\_bill();

void calc\_bill();

}

void print\_bill(){

//printing the final order with bill

}

void calc\_bill(){

//calculate the bill

}

class Test{

private :

Restaurant R;

public :

Test();

}

Test(){

//This class is used to test the functionality of the project . It check if the public methods of Menu class wrks as intended by checking the output for various stages.

}

* **Updated Declaration:**

The actual code can perform various tasks like, adding the employee in the list, removing employee, taking orders, removing orders, assigning orders to one particular employee, printing final list of employee, orders and assigned orders.

#include <iostream>  
#include <cstring>  
using namespace std;  
class Restaurant {  
  
private:  
 int choice;  
  
  
public :  
  
 void select(int choice);  
 void print();  
  
 friend class employeeList;  
 friend class employee;  
 friend class orderList;  
 friend class order;  
 void emp\_detail();  
 void order\_detail();  
 void emp\_orders();  
};

class orderList : public Restaurant{  
private :  
 class order{  
 private:  
 int order\_id;  
 int quantity;  
 double order\_cost;  
 string item;  
 order \*next1;  
 public :  
 order(string item, int quantity, int order\_id, double order\_cost, order \*next1);  
 friend class employee;  
 friend class employeeList;  
 friend class orderList;  
 friend class Restaurant;  
 };  
 order \*head1=nullptr;  
public :  
 void add\_order(string item, int quantity, int order\_id, double order\_cost);  
 void remove\_order(int order\_id);  
 void assign\_orders();  
 void print\_orders();  
 //void order\_detail();  
 friend class order;  
 friend class employee;  
 friend class employeeList;  
  
};

class employeeList : public Restaurant{  
private :  
  
 class employee{  
 private:  
 string fname;  
 int eid;  
 double salary;  
 int age;  
 employee \*next;  
 orderList::order\* ord;  
 public:  
 employee(string fname, int eid, int age, double salary, employee\* next);  
 friend class Restaurant;  
 friend class employeeList;  
 friend class orderList;  
 };  
 employee\* head = nullptr;  
public :  
 employeeList() = default;  
 void add(string fname, int eid, int age, double salary);  
 void printEmp();  
  
 //void add(char \*fname, char \*eid, char\* position, double salary);  
 void delete\_emp(int eid);  
 friend class order;  
 friend class orderList;  
 friend class Restaurant;  
};

* **Changes to Bill class:**

The actual code all the functions mentioned above. But instead of generating a bill(mentioned in the preliminary project) it calculates the final amount and displays it with the order details. Instead of creating a whole class for final bill calculation, the calculations were made inside the function itself.

Bill Class(from preliminary code):

class Bill : public order{

private :

float price;

float tax;

public :

void print\_bill();

void calc\_bill();

}

void print\_bill(){

//printing the final order with bill

}

void calc\_bill(){

//calculate the bill

}

Final price calculation:

while(current){  
 cout << "Item Name: " << current->item << " Order Id: " << current->order\_id << " Quantity: " << current->quantity << " Price: " << (current->order\_cost)\*(current->quantity) << endl;  
 current = current->next1;  
}

* **Assigning Orders :**

A new function added to the code, which was not mentioned in the preliminary project is, that every single employee in the restaurant is assigned an order to prepare.

void orderList :: assign\_orders(){  
 employeeList :: employee \*current=employees->head;  
 orderList :: order \*current1=orders->head1;  
  
 while(current){  
 current->ord=current1;  
 current=current->next;  
 current1=current1->next1;  
 }  
}

void Restaurant :: emp\_orders(){  
 int choose=1;  
  
 employeeList::employee\* curr = employees->head;  
 cout<<"\n----------x------------x------------x------------\n";  
 switch(choose){  
 case 1:  
  
 while(curr){  
 cout << "Employee id: " << curr->eid <<" | "<< " order id : " << curr->ord->order\_id << " order : " << curr->ord->item;  
 cout<<" order price: " << (curr->ord->order\_cost)\*(curr->ord->quantity) << endl;  
 curr = curr->next;  
 }  
 break;  
 default:  
 break;  
 }  
  
}

* **Testing’s:**

The testings were included inside the functions. They are done during the flow of the code itself instead of a separate function or class.

A few examples:

switch(choose){

default:  
 cout<<"\nWrong choice!!! Logging out...\n";

}

if (head == nullptr) {  
 cout << "\n No employees to delete.\n";  
 return ;  
}

cout << "1.Add new employee\n";  
cout << "2.Delete employee\n";  
cout << "3.Print employees\n";  
cout << "Select an option (1, 2 or 3) :\n";  
cin >> choose;  
string name; int id, age; double sal;  
switch(choose){  
 case 1:  
 cout << "Enter the employee name: \n";  
 cin >> name;  
 cout << "Enter the id :\n";  
 cin >> id;  
 cout << "Enter the salary: \n";  
 cin >> sal;  
 cout << "Enter the age: \n";  
 cin >> age;  
 employees->add(name, id, age, sal);  
 break;  
 case 2:  
 cout<<"\nEnter the id of the employee to be removed: \n ";  
 cin>>id;  
 employees->delete\_emp(id);  
 break;  
 case 3:  
 employees->printEmp();  
 break;  
 default:  
 cout<<"\nWrong choice!!! Logging out...\n";  
 break;

* Termination of the program if wrong choice is selected;

Int main(){

int number=1;  
while(number==1 ||number==2 || number==3){  
 s.print();  
 cout<<"\n";  
 cout<<"Select an option :\n";  
 cin>>number;  
 s.select(number);  
}

}

Inheritance

Dependency

RESTAURANT

Int choice;

Void select(int choice);

Void print();

Friend class employeeList;

Friend class employee;

Friend class orderList;

Friend class order;

Friend class order;

Void emp\_detail();

Void order\_detail();

orderList

private:

class order

order \*head1=nullptr;

public:

void add\_order (string item, int quantitly, int order\_id, double);

order\_cost();

void remove\_order(int order\_id);

void assign\_order();

void print\_order();

friend class order;

friend class employee;

friend class employeeList;

Order

Private:

int order\_id;

int quantity;

double order\_cost;

string item;

order \*next1;

public:

order (string item, int quantity, int order\_id, double order\_cost, order \*next1);

friend class employee;

friend class employeeList;

friend class orderList;

friend class Restaurant;

EmployeeList

Private:

Class employee;

Employee \*head = nullptr;

Public:

employeeList() = default;

void add(string fname, int eid, int age, double salary);

void printEmpy();

void delete\_emp(int eid);

friend class order;

friend class orderList;

friend class Restaurant;

Employee

Private:

String fname;

Int eid;

Double salary;

Int age;

Employee \*next;

orderList::order \* ord;

public:

employee(string fname, int eid, int age, double salary, employee\* next);

friend class Restaurant;

friend class employeeList;

friend class orderList;