

problem 1.

class Restaurant

public:

vector<Employee> all_employees();

int all_money();

int avg_customer();

private:

Owner owner; vector<Consumer> cons;

vector<Employee> chefs;

vector<Employee> hosts;

vector<Employee> waiters;

class Consumer

public:

virtual int tip()=0; (pure virtual)

protected:

int size, time;

string table;

public

class walk-in
: public Consumer

public

class Reserved:
public Consumer

public

class Celebrity:
public Consumer

class Employee

public:

virtual int pay()=0; (pure virtual)

public

class Waiter:
public Employee

public

class Host:
public Employee

public

class Chef:
public Employee
private:
queue<string> int skill;
public:
void cook

private:

set<string> tables;

vector<Employee> chefs;

public:

int tip();

void get_order();

void give_order();

class Owner

Abstract classes: Consumer, Employee

Not Abstract classes: Restaurant, walk-in, Reserved, Celebrity, waiter, Host, Chef.

Explanation :

I chose Consumer and Employee to be abstract classes so that they can be a prototype for each class that inherit whichever are. All inherited classes will define the pure virtual functions inside themselves. I didn't specify them in the diagram b/c of redundancy. I added unique data members to each class (if necessary) to support all the functionalities required by the problem such as host retrieving orders from a table and bringing them to a chef. Waiters have chefs so that they can assign a chef to cook. Finally, the class Restaurant has owner, chefs, hosts, ~~and~~ waiters, and consumers as private members so that they can do any necessary operations.