Delivery Service

Generated by Doxygen 1.9.8

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 abstract Class Reference

Public Member Functions

- **Employee** (const string &name, const string &position, const double &salary)
- string **getName** () const
- void **setName** (const string &name)
- string **getPosition** () const
- void **setPosition** (const string &position)
- double **getSalary** () const
- void **setSalary** (const double &salary)
- bool **getBusy** () const
- void **setBusy** (const int &busy)
- vector< **Order** > & **getOrders** ()
- virtual void **recieveAnOrder** (**Order** &order)=0
- virtual void **toWork** ()=0

Protected Attributes

- string **name_**
- string **position_**
- double **salary_**
- bool **busy_**
- vector< **Order** > **orders**

The documentation for this class was generated from the following file:

- Employee.h

4.2 Client Class Reference

Public Member Functions

- **Client** ()

  *Default constructor of the **Client** (p. 8) class.*
- **Client** (const string &name, const string &address, const string &number)

  *Constructor of the **Client** (p. 8) class.*
- string **getName** () const

  Get the client's name.
- void **setName** (const string &name)

  Set the client's name.
- string **getAddress** () const

  Get the client's address.
- void **setAddress** (const string &address)

  Set the client's address.
- string **getNumber** () const

  Get the client's phone number.
- void **setNumber** (const string &number)

  Set the client's phone number.
- void **takeDish** (const **Dish** &dish)

  Take a dish.
- void **placeAnOrder** (**DeliveryService** &delServ, bool manual)

  Place an order.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Client() [1/2]

```
Client::Client ( )
```

Default constructor of the **Client** (p. 8) class.

The default constructor initializes a **Client** (p. 8) object with a default name, address, and phone number.

4.2.1.2 Client() [2/2]

```
Client::Client (
                 const string & name,
                 const string & address,
                 const string & number )
```

Constructor of the **Client** (p. 8) class.

The constructor initializes a **Client** (p. 8) object with a given name, address, and phone number.

Parameters

name	The client's name.
address	The client's address.
number	The client's phone number.

4.2.2 Member Function Documentation

4.2.2.1 getAddress()

```
string Client::getAddress ( ) const
```

Get the client's address.

Returns

The client's address.

4.2.2.2 getName()

```
string Client::getName ( ) const
```

Get the client's name.

Returns

The client's name.

4.2.2.3 getNumber()

```
string Client::getNumber ( ) const
```

Get the client's phone number.

Returns

The client's phone number.

4.2.2.4 placeAnOrder()

```
void Client::placeAnOrder (
             DeliveryService & delServ,
             bool manual )
```

Place an order.

Places an order with a delivery service. If the order is manual, the client is asked to choose a restaurant and dishes. If the order is not manual, a dish is chosen from the first restaurant's menu.

Parameters

delServ	The delivery service.
manual	Whether the order is manual.

4.2.2.5 setAddress()

```
void Client::setAddress (
            const string & address )
```

Set the client's address.

Parameters

address	The new address of the client.

4.2.2.6 setName()

```
void Client::setName (
            const string & name )
```

Set the client's name.

Parameters

name	The new name of the client.

4.2.2.7 setNumber()

```
void Client::setNumber (
            const string & number )
```

Set the client's phone number.

Parameters

number	The new phone number of the client.

4.2.2.8 takeDish()

```
void Client::takeDish (
            const  Dish & dish )
```

Take a dish.

Adds a dish to the client's dishes.

Parameters

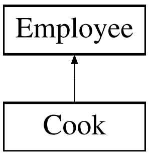
dish	The dish to take.

The documentation for this class was generated from the following files:

- Client.h
- **Client.cpp**

4.3 Cook Class Reference

Inheritance diagram for Cook:



Public Member Functions

- **Cook** ()

  *Constructor of the **Cook** (p. 11) class.*
- **Cook** (const string &name, const double &salary, const string &position)

  *Constructor of the **Cook** (p. 11) class.*
- void **recieveAnOrder** (**Order** &order) override

  Receive an order for preparation.
- void **toWork** () override

  Start working on orders.
- vector< **Dish** > **passOnDishes** ()

  Pass on prepared dishes.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Cook() `[1/2]`

```
Cook::Cook ( )
```

Constructor of the **Cook** (p. 11) class.

The default constructor initializes a **Cook** (p. 11) object with a given name, position, and salary.

4.3.1.2 Cook() `[2/2]`

```
Cook::Cook (
             const string & name,
             const double & salary,
             const string & position = "Cook" )
```

Constructor of the **Cook** (p. 11) class.

The constructor initializes a **Cook** (p. 11) object with a given name, position, and salary.

Parameters

name	The cook's name.
salary	The cook's salary.
position	The cook's position.

4.3.2 Member Function Documentation

4.3.2.1 passOnDishes()

```
vector< Dish > Cook::passOnDishes ( )
```

Pass on prepared dishes.

Returns a vector containing all prepared dishes.

Returns

A vector containing all prepared dishes.

4.3.2.2 recieveAnOrder()

```
void Cook::recieveAnOrder (
              Order & order )  [override]
```

Receive an order for preparation.

If the number of orders is less than the maximum, then the order is added to the list of orders. Otherwise, a message is displayed that the cook has received the maximum number of orders.

Parameters

order	The order to prepare.

4.3.2.3 toWork()

```
void Cook::toWork ( )  [override]
```

Start working on orders.

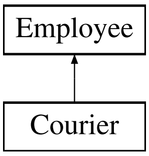
The cook starts preparing dishes for each order in the list of orders. After completing the preparation of each order, the order status is set to "ready", and a message is displayed about the order readiness. After completing work on all orders, the cook's busy status is set to "not busy".

The documentation for this class was generated from the following files:

- Cook.h
- **Cook.cpp**

4.4 Courier Class Reference

Inheritance diagram for Courier:



Public Member Functions

- **Courier** ()

  *Constructor of the **Courier** (p. 13) class.*
- **Courier** (const string &name, const double &salary, const string &position)

  *Constructor of the **Courier** (p. 13) class.*
- void **recieveAnOrder** (**Order** &order) override

  Receive an order for delivery.
- void **putInDishes** (const vector< **Dish** > &preparedDishes)

  Put dishes in the courier's bag.
- void **toWork** () override

  Start working on delivering orders.

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Courier() [1/2]

```
Courier::Courier ( )
```

Constructor of the **Courier** (p. 13) class.

The default constructor initializes a **Courier** (p. 13) object with a given name, position, and salary.

4.4.1.2 Courier() [2/2]

```
Courier::Courier (
             const string & name,
             const double & salary,
             const string & position = "Courier" )
```

Constructor of the **Courier** (p. 13) class.

The constructor initializes a **Courier** (p. 13) object with a given name, position, and salary.

Parameters

name	The courier's name.
salary	The courier's salary.
position	The courier's position.

4.4.2 Member Function Documentation

4.4.2.1 putInDishes()

```
void Courier::putInDishes (
            const vector< Dish > & preparedDishes )
```

Put dishes in the courier's bag.

Adds dishes to the courier's bag.

Parameters

preparedDishes	The dishes to add to the bag.

4.4.2.2 recieveAnOrder()

```
void Courier::recieveAnOrder (
             Order & order )  [override]
```

Receive an order for delivery.

If the number of orders is less than the maximum and the number of dishes in the order does not exceed the remaining space in the bag, then the order is added to the list of orders. Otherwise, a message is displayed that the courier's bag is full or the courier has received the maximum number of orders.

Parameters

order	The order to deliver.

4.4.2.3 toWork()

```
void Courier::toWork ( )  [override]
```

Start working on delivering orders.

The courier starts delivering dishes for each order in the list of orders. After delivering each dish, the order status is set to "delivered", and a message is displayed about the order delivery. After delivering all dishes for all orders, the courier's busy status is set to "not busy".

The documentation for this class was generated from the following files:

- Courier.h
- **Courier.cpp**

4.5 DeliveryService Class Reference

Public Member Functions

- **DeliveryService** ()

  *Default constructor of the **DeliveryService** (p. 15) class.*
- **DeliveryService** (const string &name)

  *Constructor of the **DeliveryService** (p. 15) class.*
- string **getName** () const

  Get the delivery service's name.
- void **setName** (const string &name)

  Set the delivery service's name.
- vector< **Restaurant** * > **getRestaurants** ()

  Get the list of restaurants.
- void **addRestaurant** (**Restaurant** &restaurant)

  Add a restaurant to the delivery service.
- void **addAnEmployee** (**Courier** &courier)

  Add a courier to the delivery service.
- vector< Employee * > & **getCouriers** ()

  Get the list of couriers.
- void **sentInRestaurant** (size_t index, **Order** &order)

  Send an order to the restaurant.

4.5.1 Constructor & Destructor Documentation

4.5.1.1 DeliveryService() [1/2]

```
DeliveryService::DeliveryService ( )
```

Default constructor of the **DeliveryService** (p. 15) class.

The default constructor initializes a **DeliveryService** (p. 15) object with a default name.

4.5.1.2 DeliveryService() [2/2]

```
DeliveryService::DeliveryService (
             const string & name )
```

Constructor of the **DeliveryService** (p. 15) class.

The constructor initializes a **DeliveryService** (p. 15) object with a given name.

Parameters

name	The delivery service's name.

4.5.2 Member Function Documentation

4.5.2.1 addAnEmployee()

```
void DeliveryService::addAnEmployee (
              Courier & courier )
```

Add a courier to the delivery service.

Adds a courier to the list of couriers of the delivery service.

Parameters

courier	The courier to add.

4.5.2.2 addRestaurant()

```
void DeliveryService::addRestaurant (
              Restaurant & restaurant )
```

Add a restaurant to the delivery service.

Adds a restaurant to the list of restaurants of the delivery service.

Parameters

restaurant	The restaurant to add.

4.5.2.3 getCouriers()

```
vector< Employee * > & DeliveryService::getCouriers ( )
```

Get the list of couriers.

Returns

The list of couriers.

4.5.2.4 getName()

```
string DeliveryService::getName ( ) const
```

Get the delivery service's name.

Returns

The delivery service's name.

4.5.2.5 getRestaurants()

```
vector< Restaurant * > DeliveryService::getRestaurants ( )
```

Get the list of restaurants.

Returns

The list of restaurants.

4.5.2.6 sentInRestaurant()

```
void DeliveryService::sentInRestaurant (
             size_t index,
              Order & order )
```

Send an order to the restaurant.

Sends an order to the restaurant and adds it to the courier's list of orders.

Parameters

index	The index of the restaurant.
order	The order to send.

4.5.2.7 setName()

```
void DeliveryService::setName (
             const string & name )
```

Set the delivery service's name.

Parameters

name	The new name of the delivery service.

The documentation for this class was generated from the following files:

- DeliveryService.h
- **DeliveryService.cpp**

4.6 Dish Class Reference

Public Member Functions

- **Dish** ()

*Constructor of the **Dish** (p. 17) class.*

- **Dish** (const string &name, const double &price)

  *Constructor of the **Dish** (p. 17) class.*
- string **getName** () const

  Get the name of the dish.
- void **setName** (const string &name)

  Set the name of the dish.
- double **getPrice** () const

  Get the price of the dish.
- void **setPrice** (const double &price)

  Set the price of the dish.
- bool **operator==** (const **Dish** &dish) const

  *Comparison operator for two **Dish** (p. 17) objects.*

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Dish() [1/2]

```
Dish::Dish ( )
```

Constructor of the **Dish** (p. 17) class.

The constructor initializes a **DishDataBase** (p. 21) object and gets a random dish from the database. It also sets the dish price to 10.

4.6.1.2 Dish() [2/2]

```
Dish::Dish (
             const string & name,
             const double & price )
```

Constructor of the **Dish** (p. 17) class.

The constructor initializes a **Dish** (p. 17) object with a given name and price.

Parameters

name	Name of the dish.
price	Price of the dish.

4.6.2 Member Function Documentation

4.6.2.1 getName()

```
string Dish::getName ( ) const
```

Get the name of the dish.

Returns

A string containing the name of the dish.

4.6.2.2 getPrice()

```
double Dish::getPrice ( ) const
```

Get the price of the dish.

Returns

Price of the dish.

4.6.2.3 operator==()

```
bool Dish::operator== (
             const  Dish & dish ) const
```

Comparison operator for two **Dish** (p. 17) objects.

The operator compares the names of the dishes of two **Dish** (p. 17) objects.

Parameters

dish	**Dish** (p. 17) object to compare.

Returns

Returns true if the dish names match, otherwise returns false.

4.6.2.4 setName()

```
void Dish::setName (
             const string & name )
```

Set the name of the dish.

Parameters

name	Name of the dish.

4.6.2.5 setPrice()

```
void Dish::setPrice (
             const double & price )
```

Set the price of the dish.

Parameters

price	Price of the dish.

The documentation for this class was generated from the following files:

- Dish.h
- **Dish.cpp**

4.7 DishDataBase Class Reference

Public Member Functions

- **DishDataBase** ()

  Constructor of the ***DishDataBase*** *(p. 21) class.*
- std::string **getRandomDish** ()

  Get a random dish.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 DishDataBase()

```
DishDataBase::DishDataBase ( )
```

Constructor of the **DishDataBase** (p. 21) class.

The constructor initializes the vector dishNames with dish names and selects a random dish.

4.7.2 Member Function Documentation

4.7.2.1 getRandomDish()

```
std::string DishDataBase::getRandomDish ( )
```

Get a random dish.

Returns

A string containing the name of the random dish.

The documentation for this class was generated from the following files:

- DishDB.h
- DishDB.cpp

4.8 Menu Class Reference

Public Member Functions

- **Menu** ()

  *Constructor of the **Menu** (p. 22) class.*
- **Menu** (const size_t &size)

  *Constructor of the **Menu** (p. 22) class.*
- **Dish getDish** (size_t i) const

  Get a dish by index.
- void **addDish** (const **Dish** &dish)

  Add a dish to the menu.
- void **addDish** (const string &name, const double &price)

  Add a dish to the menu.
- void **delDish** (const **Dish** &dish)

  Remove a dish from the menu.
- void **delDish** (const string &name)

  Remove a dish from the menu.
- void **delDish** (const size_t &number)

  Remove a dish from the menu.
- void **show** ()

  Display the menu.

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Menu() [1/2]

```
Menu::Menu ( )  [default]
```

Constructor of the **Menu** (p. 22) class.

Default constructor.

4.8.1.2 Menu() [2/2]

```
Menu::Menu (
             const size_t & size )
```

Constructor of the **Menu** (p. 22) class.

The constructor initializes a menu of a given size.

Parameters

size	Size of the menu.

4.8.2 Member Function Documentation

4.8.2.1 addDish() [1/2]

```
void Menu::addDish (
            const  Dish & dish )
```

Add a dish to the menu.

Parameters

dish	The dish to add.

4.8.2.2 addDish() [2/2]

```
void Menu::addDish (
            const string & name,
            const double & price = 10 )
```

Add a dish to the menu.

Parameters

name	Name of the dish.
price	Price of the dish.

4.8.2.3 delDish() [1/3]

```
void Menu::delDish (
            const  Dish & dish )
```

Remove a dish from the menu.

Parameters

dish	The dish to remove.

4.8.2.4 delDish() [2/3]

```
void Menu::delDish (
            const size_t & number )
```

Remove a dish from the menu.

Parameters

number	Number of the dish to remove.

4.8.2.5 delDish() [3/3]

```
void Menu::delDish (
             const string & name )
```

Remove a dish from the menu.

Parameters

name	Name of the dish to remove.

4.8.2.6 getDish()

```
 Dish Menu::getDish (
             size_t i ) const
```

Get a dish by index.

Parameters

i	Index of the dish.

Returns

The dish at the given index.

4.8.2.7 show()

```
void Menu::show ( )
```

Display the menu.

Displays all the dishes in the menu.

The documentation for this class was generated from the following files:

- Menu.h
- **Menu.cpp**

4.9 Order Class Reference

Public Member Functions

- **Order** (**Client** &client)

  *Constructor of the **Order** (p. 24) class.*
- bool **getStatus** () const

  Get the status of the order.

- void **setStatus** (bool status)

  Set the status of the order.
- size_t **getNumber** () const

  Get the order number.
- **Client** ∗ **getClient** () const

  Get the client for whom the order was created.
- vector< **Dish** > **getDishes** ()

  Get all dishes in the order.
- void **addDish** (const **Dish** &dish)

  Add a dish to the order.
- void **delDish** (const **Dish** &dish)

  Remove a dish from the order.
- bool **operator==** (const **Order** &other) const

  *Comparison operator for two **Order** (p. 24) objects.*

4.9.1 Constructor & Destructor Documentation

4.9.1.1 Order()

```
Order::Order (
              Client & client )
```

Constructor of the **Order** (p. 24) class.

The constructor initializes an **Order** (p. 24) object with a given client.

Parameters

client	The client for whom the order is created.

4.9.2 Member Function Documentation

4.9.2.1 addDish()

```
void Order::addDish (
             const  Dish & dish )
```

Add a dish to the order.

Parameters

dish	The dish to add to the order.

4.9.2.2 delDish()

```
void Order::delDish (
             const  Dish & dish )
```

Remove a dish from the order.

Parameters

dish	The dish to remove from the order.

4.9.2.3 getClient()

```
Client * Order::getClient ( ) const
```

Get the client for whom the order was created.

Returns

The client for whom the order was created.

4.9.2.4 getDishes()

```
vector< Dish > Order::getDishes ( )
```

Get all dishes in the order.

Returns

A vector containing all the dishes in the order.

4.9.2.5 getNumber()

```
size_t Order::getNumber ( ) const
```

Get the order number.

Returns

Order (p. 24) number.

4.9.2.6 getStatus()

```
bool Order::getStatus ( ) const
```

Get the status of the order.

Returns

Order (p. 24) status.

4.9.2.7 operator==()

```
bool Order::operator== (
             const Order & other ) const
```

Comparison operator for two **Order** (p. 24) objects.

The operator compares the numbers of two **Order** (p. 24) objects.

Parameters

other	**Order** (p. 24) object to compare.

Returns

Returns true if the numbers match, otherwise returns false.

4.9.2.8 setStatus()

```
void Order::setStatus (
            bool status )
```

Set the status of the order.

Parameters

status	New order status.

The documentation for this class was generated from the following files:

- Order.h
- **Order.cpp**

4.10 Restaurant Class Reference

Public Member Functions

- **Restaurant** ()

  *Default constructor of the **Restaurant** (p. 27) class.*
- **Restaurant** (string name, **Menu** menu, double capital, size_t rating)

  *Constructor of the **Restaurant** (p. 27) class.*
- string **getName** () const

  Get the restaurant's name.
- void **setName** (const string &name)

  Set the restaurant's name.
- float **getRating** () const

  Get the restaurant's rating.
- void **setRating** (const float &rating)

  Set the restaurant's rating.
- **Menu** **getMenu** () const

  Get the restaurant's menu.
- void **setMenu** (const **Menu** &menu)

  Set the restaurant's menu.
- double **getCapital** () const

  Get the restaurant's capital.
- void **setCapital** (const double &capital)

Set the restaurant's capital.

- vector< Employee * > & **getCooks** ()

  Get the list of cooks.
- void **revieveAnOrder** (const **Order** & **Order**)

  Receive an order.
- void **addAnEmployee** (**Cook** &cook)

  Add an employee to the restaurant.
- void **orderDistribution** ()

  Distribute orders.
- void **dishTransfer** (vector< Employee * > &couriers)

  Transfer dishes to couriers.
- void **showInfo** () const

  Show restaurant information.

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Restaurant() [1/2]

```
Restaurant::Restaurant ( )
```

Default constructor of the **Restaurant** (p. 27) class.

The default constructor initializes a **Restaurant** (p. 27) object with a default name, menu, capital, and rating.

4.10.1.2 Restaurant() [2/2]

```
Restaurant::Restaurant (
             string name,
              Menu menu,
             double capital,
             size_t rating = 0 )
```

Constructor of the **Restaurant** (p. 27) class.

The constructor initializes a **Restaurant** (p. 27) object with a given name, menu, capital, and rating.

Parameters

name	The restaurant's name.
menu	The restaurant's menu.
capital	The restaurant's capital.
rating	The restaurant's rating.

4.10.2 Member Function Documentation

4.10.2.1 addAnEmployee()

```
void Restaurant::addAnEmployee (
              Cook & cook )
```

Add an employee to the restaurant.

Adds cooks to the restaurant.

Parameters

cook	The cook.

4.10.2.2 dishTransfer()

```
void Restaurant::dishTransfer (
            vector< Employee * > & couriers )
```

Transfer dishes to couriers.

Transfers prepared dishes to couriers.

Parameters

couriers	The couriers.

4.10.2.3 getCapital()

```
double Restaurant::getCapital ( ) const
```

Get the restaurant's capital.

Returns

The restaurant's capital.

4.10.2.4 getCooks()

```
vector< Employee * > & Restaurant::getCooks ( )
```

Get the list of cooks.

Returns

The list of cooks.

4.10.2.5 getMenu()

```
Menu Restaurant::getMenu ( ) const
```

Get the restaurant's menu.

Returns

The restaurant's menu.

4.10.2.6 getName()

```
string Restaurant::getName ( ) const
```

Get the restaurant's name.

Returns

The restaurant's name.

4.10.2.7 getRating()

```
float Restaurant::getRating ( ) const
```

Get the restaurant's rating.

Returns

The restaurant's rating.

4.10.2.8 orderDistribution()

```
void Restaurant::orderDistribution ( )
```

Distribute orders.

Distributes orders among cooks.

4.10.2.9 revieveAnOrder()

```
void Restaurant::revieveAnOrder (
            const Order & order )
```

Receive an order.

Adds an order to the list of orders.

Parameters

order	The order to receive.

4.10.2.10 setCapital()

```
void Restaurant::setCapital (
            const double & capital )
```

Set the restaurant's capital.

Parameters

capital	The new capital of the restaurant.

4.10.2.11 setMenu()

```
void Restaurant::setMenu (
             const  Menu & menu )
```

Set the restaurant's menu.

Parameters

menu	The new menu of the restaurant.

4.10.2.12 setName()

```
void Restaurant::setName (
             const string & name )
```

Set the restaurant's name.

Parameters

name	The new name of the restaurant.

4.10.2.13 setRating()

```
void Restaurant::setRating (
             const float & rating )
```

Set the restaurant's rating.

Parameters

rating	The new rating of the restaurant.

4.10.2.14 showInfo()

```
void Restaurant::showInfo ( ) const
```

Show restaurant information.

Outputs information about the restaurant.

The documentation for this class was generated from the following files:

- Restaurant.h
- **Restaurant.cpp**

Chapter 5

File Documentation

5.1 Client.cpp File Reference

File containing the implementation of the **Client** (p. 8) class.

```
#include "Client.h"
#include "DeliveryService.h"
```

5.1.1 Detailed Description

File containing the implementation of the **Client** (p. 8) class.

Author

Verkovich E.V.

Date

November 26, 2023

5.2 Client.h

```
#pragma once

#include <iostream>
#include <string>
#include "Dish.h"
#include "Order.h"


using namespace std;
class DeliveryService;
class Restaurant;
class Client {
private:
    string name_;
    string address_;
    string number_;
    vector<Dish>dishes;

    //Restaurant* ChooseRestaurant(DeliveryService& delServ, size_t i);
public:
    Client();
    Client(const string& name, const string& address, const string& number);

    string getName() const;
    void setName(const string& name);

    string getAddress() const;
    void setAddress(const string& address);

    string getNumber() const;
    void setNumber(const string& number);

    void takeDish(const Dish& dish);

    void placeAnOrder(DeliveryService& delServ, bool manual);
};
```

5.3 Cook.cpp File Reference

File containing the implementation of the **Cook** (p. 11) class.

```
#include "Cook.h"
```

5.3.1 Detailed Description

File containing the implementation of the **Cook** (p. 11) class.

Author

Verkovich E.V.

Date

November 26, 2023

5.4 Cook.h

```
#pragma once

#include <vector>
#include <iostream>
#include <utility>
#include <string>
#include <algorithm>
#include <memory>
#include "Employee.h"

#define MROT 500

using namespace std;

class Cook : public Employee {
private:
    const size_t maxOrderQuantity_ = 10;
    vector<Dish>preparedDishes;

public:
    Cook();
    Cook(const string& name, const double& salary, const string& position);

    void recieveAnOrder(Order& order) override;
    void toWork() override;

    vector <Dish> passOnDishes();
};
```

5.5 Courier.cpp File Reference

File containing the implementation of the **Courier** (p. 13) class.

```
#include "Courier.h"
#include "DeliveryService.h"
```

5.5.1 Detailed Description

File containing the implementation of the **Courier** (p. 13) class.

Author

Verkovich E.V.

Date

November 26, 2023

5.6 Courier.h

```
#pragma once

#include "Employee.h"
#include "Order.h"
#include "Client.h"
#include <memory>

#define MROT 500

using namespace std;

class Courier : public Employee {
private:
    const size_t maxOrderQuantity_ = 10;
    const size_t bagSize_ = 10;
    vector<Dish>bag;


    bool findDish(const Dish& dish) const;
    Dish takeOut(const Dish& dish);

public:
    Courier();
    Courier(const string& name, const double& salary, const string& position);

    void recieveAnOrder(Order& order) override;

    void putInDishes(const vector<Dish>& preparedDishes);

    void toWork() override;
};
```

5.7 Delivery.h

```
#pragma once
#include <vector>
#include <iostream>
#include <utility>
#include <string>
#include <algorithm>
#include <memory>
#include "DishDB.h"

```

```
#define MROT 500
using namespace std;

class Client;
class DeliveryService;
















```

5.8 DeliveryService.cpp File Reference

File containing the implementation of the **DeliveryService** (p. 15) class.

```
#include "DeliveryService.h"
```

5.8.1 Detailed Description

File containing the implementation of the **DeliveryService** (p. 15) class.

Author

Verkovich E.V.

Date

November 26, 2023

5.9 DeliveryService.h

```
#pragma once

#include "Restaurant.h"

#define MROT 500

using namespace std;


class DeliveryService {
private:
    string name_;
    vector<Restaurant*> restaurants;
    vector<Employee*> couriers;

public:
    DeliveryService();
    DeliveryService(const string& name);

    string getName() const;
    void setName(const string& name);

    vector<Restaurant*> getRestaurants();
    void addRestaurant(Restaurant& restaurant);

    void addAnEmployee(Courier& courier);
    vector<Employee*>& getCouriers();

    void sentInRestaurant(size_t index, Order& order);
};
```

5.10 Dish.cpp File Reference

File containing the implementation of the **Dish** (p. 17) class.

```
#include "Dish.h"
```

5.10.1 Detailed Description

File containing the implementation of the **Dish** (p. 17) class.

Author

Verkovich.E.V.

Date

November 26, 2023

5.11 Dish.h

```
#pragma once
#include <string>
#include "DishDB.h"

using namespace std;
class Dish {
private:
    string name_;
    double price_;

public:
    Dish();
    Dish(const string& name, const double& price);

    string getName() const;
    void setName(const string& name);
    double getPrice() const;
    void setPrice(const double& price);

    bool operator==(const Dish& dish) const;
};
```

5.12 DishDB.h

```
#pragma once

#include <string>
#include<vector>

class DishDataBase {
public:
    DishDataBase();
    std::string getRandomDish();

private:
    std::vector<std::string> dishNames;
    size_t randomDishIndex;
    std::string randomDish;
};

```

5.13 Employee.cpp File Reference

File containing the implementation of the Employee class.

```
#include "Employee.h"
```

5.13.1 Detailed Description

File containing the implementation of the Employee class.

Author

Verkovich E.V.

Date

November 26, 2023

5.14 Employee.h

```
#pragma once

#include "Order.h"

using namespace std;

class Employee abstract {
protected:
    string name_;
    string position_;
    double salary_;
    bool busy_; //0 --  , 1 --
    vector<Order>orders;
public:
    Employee(const string& name, const string& position, const double& salary);

    string getName() const;
    void setName(const string& name);

    string getPosition() const;
    void setPosition(const string& position);

    double getSalary() const;
    void setSalary(const double& salary);

    bool getBusy()const;
    void setBusy(const int& busy);

    vector<Order>& getOrders();

    virtual void recieveAnOrder(Order& order) = 0;
    virtual void toWork() = 0;
};
```

5.15 Menu.cpp File Reference

File containing the implementation of the **Menu** (p. 22) class.

```
#include "Menu.h"
#include <iostream>
#include <vector>
```

5.15.1 Detailed Description

File containing the implementation of the **Menu** (p. 22) class.

Author

Verkovic E.V.

Date

November 26, 2023

5.16 Menu.h

```
#pragma once
#include "Dish.h"
#include <iostream>
#include <vector>
using namespace std;

class Menu {
private:
    vector<pair<size_t, Dish» dishes;
    bool isSameInMenu(const Dish& dish);

public:
    Menu();
    Menu(const size_t& size);

    Dish getDish(size_t i) const;

    void addDish(const Dish& dish);
    void addDish(const string& name, const double& price);
    void delDish(const Dish& dish);
    void delDish(const string& name);
    void delDish(const size_t& number);

    void show();
};
```

5.17 Order.cpp File Reference

File containing the implementation of the **Order** (p. 24) class.

```
#include "Order.h"
#include "Client.h"
```

5.17.1 Detailed Description

File containing the implementation of the **Order** (p. 24) class.

Author

Verkovich E.V.

Date

November 26, 2023

5.18 Order.h

```
#pragma once

#include <iostream>
#include <memory>
#include <algorithm>
#include "Dish.h"

using namespace std;

class Client;
class Order {
private:
    static size_t currentNumber_;
    size_t number_;
    bool status_;    //0 -- (), 1 -- , 2 --  , 3 --
    vector <Dish> dishes;
    Client*client_;

public:
    Order(Client& client);
    bool getStatus() const;
    void setStatus(bool status);

    size_t getNumber() const;

    Client* getClient() const;

    vector<Dish> getDishes();

    void addDish(const Dish& dish);
    void delDish(const Dish& dish);

    bool operator==(const Order& other)const;
};
```

5.19 Restaurant.cpp File Reference

File containing the implementation of the **Restaurant** (p. 27) class.

```
#include "Restaurant.h"
```

5.19.1 Detailed Description

File containing the implementation of the **Restaurant** (p. 27) class.

Author

Your Name

Date

November 26, 2023

5.20 Restaurant.h

```
#pragma once

#include "Menu.h"
#include "Cook.h"
#include "Courier.h"


#define MROT 500

using namespace std;

class Restaurant {
private:
    string name_;
    float rating_;
    Menu menu_;
    double capital_;

    vector<Employee*> cooks;
    vector<Order>orders;

    bool checkOrderMatch(Employee* cook, Employee* courier);
public:
    Restaurant();
    Restaurant(string name, Menu menu, double capital, size_t rating);

    string getName() const;
    void setName(const string& name);

    float getRating() const;
    void setRating(const float& rating);

    Menu getMenu() const;
    void setMenu(const Menu& menu);

    double getCapital()const;
    void setCapital(const double& capital);

    vector<Employee*>& getCooks();

    void revieveAnOrder(const Order& Order);

    void addAnEmployee(Cook& cook);

    void orderDistribution();

    void dishTransfer(vector<Employee*>& couriers);

    void showInfo()const;
};
```

Index