```
#include <iostream>
#include "User.h"
#include "Admin.h"
#include "IncorrectInput.h"

using namespace std;

int main() {

    system("clear");
    int choice = 100  , x ;
    User user;
    Admin admin;
    while(choice != 0){
        while(1) {
            cout << endl  << "Menu for:" << endl;
            cout << "1 - admin" << endl;
            cout << "2 - user" << endl;
            cout << "0 - end of program" << endl;

            try {
                if (!(cin >> choice))
                    throw IncorrectInput("Sorry, enter int!");
                break;
            }
            catch (IncorrectInput err) {
                err.Display();
                rewind(stdin);
                cin.clear();
                continue;
            }
        }
        switch (choice) {
            case 1:{admin.CheckAdmin();break;}
            case 2:
            {
               int a = 1;
                while(a != 0){
                    while(1) {

                        cout << endl << endl << "Menu user:" << endl;
                        cout << "1 - log in account" << endl;
                        cout << "2 - create new client" << endl;
                        cout << "0 - main menu" << endl;

                        try {
                            if (!(cin >> x))
                                throw IncorrectInput("Sorry, enter
int!");

                            break;
                        }
                        catch (IncorrectInput err){
                            err.Display();
                            rewind(stdin);
                            cin.clear();
                            continue;
                        }
                    }
                    switch (a) {
                        case 1:
```

```
                        {
                            user.User();
                            break;
                        }

                        case 2:
                        {
                            user.NewUser();
                            break;
                        }

                        case 0:
                        {
                            a = 0;
                            break;
                        }

                        default:
                        {
                            continue;
                        }
                    }
                }
                break;
            }
            case 0:
                break;

            default:
            {
                continue;
            }
        }
    }
return 0;
}
#ifndef Services_h
#define Services_h

#include <iomanip>
#include <iostream>

using namespace std;

class Services
{
protected:
    string _name;
    double _price;

public:
    void ServicesMenu()
    {
        int _number = 1;

        while (_number != 0)
        {
            cout << "Services Menu:" << endl;
            cout << "1 - Autoservice" << endl;
            cout << "2 - Body Work" << endl;
            cout << "3 - Tire Fitting" << endl;
            cout << "4 - Diagnostics" << endl;
            cout << "5 - Carwash" << endl;
            cout << "0 - Exit to main menu " << endl;

            cin >> _number;
        }
    }
};

#endif /* Services_h */

#ifndef Autoservice_h
#define Autoservice_h

#include <string>
#include <iostream>
#include "Services.h"

using namespace std;

class AutoService: public Services {// Автосервис
protected:
    int DisplayAutoservice()
    {
        int _number = 100;

        while (_number != 0)
        {
            cout << "Autoservice Menu:" << endl;
            cout << "1 - Filters" << endl;
            cout << "2 - Engine" << endl;
            cout << "3 - BrakeSystem" << endl;
            cout << "0 - Exit to main menu " << endl;

            cin >> _number;

            switch(_number)
            {
                case 1:

            }
        }
    };

class Filters :public AutoService{
    class OilFilterReplacement{
    public:
        OilFilterReplacement(){
            _name = "Oil Filter Replacement";
            _price = 690;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~OilFilterReplacement(){}
    };
    class AirFilterReplacement{
    public:
        AirFilterReplacement(){
            _name = "Air Filter Replacement";
            _price = 890;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~AirFilterReplacement(){}
    };
};

class Engine :public AutoService{
    class EngineDiagnostics{
    public:
        EngineDiagnostics(){
            _name = "Engine Diagnostics";
            _price = 1390;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~EngineDiagnostics(){}
    };
    class ExtendedEngineDiagnostics{
    public:
        ExtendedEngineDiagnostics(){
            string _name = "Extended Engine Diagnostics";
            double _price = 2490;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~ExtendedEngineDiagnostics(){}
    };
};

class BrakeSystem :public AutoService{
    class BrakeFluidReplacement{
    public:
        BrakeFluidReplacement(){
            _name = "Brake Fluid Replacement";
            _price = 890;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~BrakeFluidReplacement(){}
    };
    class BrakePadReplacement{
    public:
        BrakePadReplacement(){
            _name = "Brake Pad Replacement";
            _price = 890;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~BrakePadReplacement(){}
    };
};

#endif /* Autoservice_h */

#ifndef BodyRepair_h
#define BodyRepair_h

#include <string>
#include <iostream>
#include "Services.h"

class BodyRepair :public Services{    // Кузовной ремонт

    };

class GlassWork :public BodyRepair{
    class FrontGlassReplacement{
    public:
        FrontGlassReplacement(){
            _name = "Front Glass Replacement";
            _price = 2800;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~FrontGlassReplacement(){}
    };
```

```
};
class BodyWork :public BodyRepair{
    class AntiCorrosionProtection{
    public:
        AntiCorrosionProtection(){
            _name = "Anti-corrosion protection";
            _price = 9300;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~AntiCorrosionProtection(){}
    };
};

#endif /* BodyRepair_h */

#ifndef TireFitting_h
#define TireFitting_h

#include <string>
#include <iostream>
#include "Services.h"

using namespace std;

class TireFitting : public Services{   // Шиномонтаж
private:
    int TireSize;
    cout << "Enter your tire size:" << endl;
    cin >> TireSize;
    if(TireSize > 19){
        _price += 500;
    }
    class LowProfile{
    public:
        LowProfile(){
            _name = "Low Profile Tire Fitting";
            _price = 3280;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~LowProfile(){}
    };
    class RunFlat{
    public:
        RunFlat(){
            string _name = "Run Flat Tire Fitting";
            double _price = 4860;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~RunFlat(){}
    };
};

#endif /* TireFitting_h */


#ifndef Diagnostics_h
#define Diagnostics_h

#include <string>
#include <iostream>
#include "Services.h"

//using namespace std;

class Diagnostics :public Services{   // Диагностика

    class EngineDiagnostics{
    public:
        EngineDiagnostics(){
            _name = "Engine Diagnostics";
            price = 1390;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~EngineDiagnostics(){}
    };
    class BrakeSystemDiagnostics{
    public:
        BrakeSystemDiagnostics(){
            string _name = "Brake System Diagnostics";
            double _price = 690;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~BrakeSystemDiagnostics(){}
    };
};
```

```
#endif /* Diagnostics_h */

#ifndef CarWash_h
#define CarWash_h

#include <string>
#include <iostream>
#include "Services.h"

//using namespace std;

class CarWash :public Services{   // Автомойка

    };

class Complex :public CarWash{

    class StandartCarWash{
    public:
        StandartCarWash(){
            _name = "Standart wash";
            _price = 450;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~StandartCarWash(){}
    };

    class LuxeCarWash{
    public:
        LuxeCarWash(){
            _name = "Luxe wash";
            _price = 900;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~LuxeCarWash(){}
    };

};

class DryCleaning :public CarWash{

    class StandartDryCleaning{
    public:
        StandartDryCleaning(){
            _name = "Standart Dry Cleaning";
            _price = 900;

        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~StandartDryCleaning(){}
    };
    class LuxeDryCleaning{
    public:
        LuxeDryCleaning(){
            _name = "Luxe Dry Cleaning";
            _price = 1800;
        }
        string ServiceName()
        {
            return _name;
        }
        double ServicePrice()
        {
            return _price;
        }
        ~LuxeDryCleaning(){}
    };

};

#endif /* CarWash_h */

#ifndef User_111_h
#define User_111_h

#include <vector>
#include <iomanip>
#include <fstream>
#include "Services.h"
#include "Exception.h"
#include "IncorrectInput.h"

class User: public Client {
private:
private:
    string Name;
    string Surname;
    string Phoneumber;
    string Car;
    string ServiceName;
    string PasswordAccount;
    Service *service;                   // Tarif *tarif
public:
    User(){}
    ~User(){}

    void NewUser(){
        User newUser;                                   // Client
person1;
        newUser.AddUser(newUser);
        int number = 100;
        while (number != 0){
            while(true) {

                cout << "New user menu:" << endl;
                cout << "1 - display user" << endl;
                cout << "2 - add car " << endl;
                cout << "3 - add phone number" << endl;
                cout << "4 - save change in base and go to main menu" <<
endl;

                try{
                    if(!(cin >> number))
                        throw MyException("Error, enter int.");
                    break;
                }
                catch(MyException& ex){
                    ex.show();
                    rewind(stdin);
                    cin.clear();
                    continue;
                }
            }
            switch (number) {

                case 1:
                {
                    DisplayUser();                  //PrintViewClient();
                    newUser.GetUser();
                    break;
                }

                case 2:{ system("clear");newUser.SetCar();break;}

                case 3:{ system("clear");newUser.SetPhoneNumber();break;}

                case 4:{
                    system("clear");
                    if(CheckIsEmptyUser(newUser))
                        newUser.WriteFile(newUser);
                    number = 0;
                    break;
                }
                default:{continue;}
            }
        }
    }
    void PrintViewClient(){
        cout <<
"************************************************************************
*********************************" << endl
        cout << "User: " << endl;
        cout << "Phone Number" << endl;
        cout << "Services" << endl;
        cout <<
"**********************************************************************
********************************" << endl
    }
    void User(){                            // ForBasicUser(){
        system("clear");

        vector<User> users = LoadPeopleInVector();
        string CheckPassword;
        int index;
        int number = 100;

        cout << "Enter password:";
        cin >> CheckPassword;

        for (int i = 0; i < users.size(); ++i) {
            if( persons[i].GetPassport() == PasportID) {
                if (persons[i].GetPassword() == CheckPassword)
                    index = i;
            }
        }

        if(!persons.empty()) {
                if (persons[index].GetPassword() == CheckPassword) {
                    cout << "Continue." << endl;
                    DisplayUser();
                    persons[index].GetClient();
                }
             else {
                cout << endl << endl << "Wrong password";
                number = 0;
            }
        }
        while (number != 0){
            while(true) {

                cout << "User menu:" << endl;
                cout << "1 - display user" << endl;
                cout << "2 - change phone number" << endl;
                cout << "3 - change car" << endl;
                cout << "4 - save changes and go to main menu" << endl;
                cout << "5 - delete account" << endl;

                try {
                    rewind(stdin);
                    cin.clear();
                    if (!(cin >> choice))
                        throw IncorrectInput("Error, enter int.");
                    break;
                }
                catch (IncorrectInput &err) {
                    err.display();
                    continue;
                }
            }
            switch (number) {

                case 1:
                {
                    DisplayUser();                  // PrintViewClient --
DisplayUser
                    users[index].GetUser();
                    break;
                }

                case 2:
                {
                    system("clear");
                    cout << "Change phone number." << endl;
                    users[index].SetPhoneNumber();
                    break;
                }

                case 3:
                {
                    system("clear");
                    cout << "Change car ." << endl;
                    users[index].SetCar();
                    break;
                }

                case 4:
                {
                    system("clear");
                    WtiteUsersToFile(users);
                    number = 0;
                    break;
                }

                case 5:{
                    system("clear");
                    DeleteUser(users[index]);
                    choice = 0;
                    break;
                }
                default:{
                    continue;
                }
            }
        }
        persons.clear();
    }

    void DeleteUser(User DeleteUser){                  // (Client
personDelete){

        vector<Client> users = AddUsersToVector(); // persons -- users
        string Path = "Clients.txt";
        fstream fout;
        fout.open(Path, ios::trunc | ios::out | ios::in);
        int count;

        try {
            if (!fout.is_open())
                throw IncorrectFileOpen("Files is not opened!");

        }
        catch (IncorrectFileOpen& ex){
            ex.show();
            exit(1);
        }

        for (int i = 0; i < users.size(); ++i) {

            if(users[i].GetPhoneNumber() != userDelete.GetPhoneNumber()){
                fout << users[i];
            }
            else{
                    cout << endl<< endl <<"User successfully deleted!" <<
endl;
            }
        }
        fout.close();
        users.clear();
    }

    void WriteUsersToFile(vector<Client> users){

        string Path = "Users.txt";
        fstream fin;
        fin.open(Path,  ios::trunc | ios::out);

        try {
            if (!fin.is_open())
                throw InCorrectOpenFiles("Files is not open!");

        }
        catch (InCorrectOpenFiles& ex){
            ex.show();
            exit(1);
        }
        for (int i = 0; i < users.size(); ++i) {

            fin << users[i];
        }
        fin.close();
        users.clear();
    }

public:
    {
        Surame = "empty";
        Name = "empty";
        Car = "empty";
        PhoneNumber = "empty";
    }

    void AddUser(User &newUser) {            // void CreateClient(Client
&person1) {

        cout<< endl << "Ading new user:" << endl;
        newUser.SetUser();
        cout << endl << endl;
        cout << "Choose service:" << endl;
        ServicesList();

        while (true) {
            newUser.setService();
            break;
        }
        SetServices();
    }
    void SetName(){

        rewind(stdin);
        cin.clear();
        cout << "Enter your name:";
        getline(cin,Name);
    }

    void Surname(){

        rewind(stdin);
        cin.clear();
        cout << "Enter your surname:";
        getline(cin,Surname);
    }

    void SetPhoneNumber() {
        cout << "Enter your phone number:" << endl;
        rewind(stdin);
        cin.clear();
        getline(cin,PhoneNumber);
    }

    void Car() {
        cout << "Enter your car:" << endl;
        rewind(stdin);
        cin.clear();
        getline(cin,Car);
    }

    void SetPassword(){

        cout << "Enter your password:";
        rewind(stdin);
        cin.clear();
        cin >> PasswordAccount;
    }
    vector<User> AddUserToVector(){
        vector<User> users;
        User a;
        string Path = "Users.txt";
        fstream fin;
        fin.open(Path);

         while (true) {
             try {
                 if (!fin.is_open())
                     throw IncorrectFileOpen("File error!");
                 break;
             }
             catch (IncorrectFileOpen& err){
                 err.show();
                 continue;
             }
         }
        while(!(fin.eof())){
            fin >> a;
```

```
            users.push_back(a);
        }
        fin.close();

        return users;
    }

    void SetUser() {
        SetName();
        SetSurName();
        SetPassword();
        SetPhoneNumber();
        SetCar();
    }

     void ServicesList(){

        cout << "Services:" << endl;
        Service* service;
        cout << endl << endl;

        service = new OilFilterReplacement;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new AirFilterReplacement;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new EngineDiagnostics;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new ExtendedEngineDiagnostics;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new BrakeFluidReplacement;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new BrakePadReplacement;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new FrontGlassReplacement;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new AntiCorrosionProtection;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new LowProfile;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new RunFlat;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;
        service = new EngineDiagnostics;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new BrakeSystemDiagnostics;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new StandartCarWash;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new LuxeCarWash;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new StandartDryCleaning;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

        service = new LuxeDryCleaning;
        cout << service->ServiceName() << endl;
        cout <<"\tPrice: " << service->ServicePrice()  << " rub" << endl;

    }

    void setServices() {
        string ServiceName;
        int number = 1;
        while (number != 0) {
            cout << "Choose services:";
            cin >> ServiceName;
            if ( ServiceName == "Oil Filter Replacement") {
                service = new OilFilterReplacement();
                ServiceName = "Oil Filter Replacement";
                number = 0;
            } else if (SefviceName =="AirFilterReplacement") {
                service = new AirFilterReplacement();
                NameTarif = "AirFilterReplacement";
                number = 0;
            }
            else if (SefviceName =="EngineDiagnostics") {
                service = new EngineDiagnostics();
                NameTarif = "EngineDiagnostics";
                number = 0;
            }
            else if (SefviceName =="ExtendedEngineDiagnostics") {
                service = new ExtendedEngineDiagnostics();
                NameTarif = "ExtendedEngineDiagnostics";
                number = 0;
            }
            else if (SefviceName =="BrakeFluidReplacement") {
                service = new BrakeFluidReplacement();
                NameTarif = "BrakeFluidReplacement";
                number = 0;
            }
            else if (SefviceName =="BrakePadReplacement") {
                service = new BrakePadReplacement();
                NameTarif = "BrakePadReplacement";
                number = 0;
            }
            else if (SefviceName =="FrontGlassReplacement") {
                service = new FrontGlassReplacement();
                NameTarif = "FrontGlassReplacement";
                number = 0;
            }
            else if (SefviceName =="AntiCorrosionProtection") {
                service = new AntiCorrosionProtection();
                NameTarif = "AntiCorrosionProtection";
                number = 0;
            } else if (SefviceName =="LowProfile") {
                service = new LowProfile();
                NameTarif = "LowProfile";
                number = 0;
            } else if (SefviceName =="RunFlat") {
                service = new RunFlat();
                NameTarif = "RunFlat";
                number = 0;
            }
            else if (SefviceName =="EngineDiagnostics") {
                service = new EngineDiagnostics();
                NameTarif = "EngineDiagnostics";
                number = 0;
            }
            else if (SefviceName =="BrakeSystemDiagnostics") {
                service = new BrakeSystemDiagnostics();
                NameTarif = "BrakeSystemDiagnostics";
                number = 0;
            }
            else if (SefviceName =="StandartCarWash") {
                service = new StandartCarWash();
                NameTarif = "StandartCarWash";
                number = 0;
            }
            else if (SefviceName =="LuxeCarWash") {
                service = new LuxeCarWash();
                NameTarif = "LuxeCarWash";
                number = 0;
            }
            else if (SefviceName =="StandartDryCleaning") {
                service = new StandartDryCleaning();
                NameTarif = "StandartDryCleaning";
                number = 0;
            }
            else if (SefviceName =="LuxeDryCleaning") {
                service = new LuxeDryCleaning();
                NameTarif = "LuxeDryCleaning";
                number = 0;
        }
    }
    string GetPassword(){
        return PasswordAccount;
    }

    string GetName(){
        return FirstName;
    }

    string GetSurame(){
        return LastName;
    }

    string GetCar(){
        return Car;
    }

    void GetUser() {

        cout << "* " << setw(30) << left << this->Name + " " + this-
>LastName
            << "* " << setw(13) << left << this->Phoneumber
            << "* " << setw(40) << left << this->ServicName << endl;
    cout <<
"**************************************************************************
***************************************************************************
********************************" << endl;
    }

    string GetService() {
        return NameTarif;
    }

    string GetPhoneumber() {
        return Number;
    }

    void WriteFile(User user) {

        string Path = "Users.txt";
        ofstream fout;
        fout.open(Path, ofstream::app);

        try {
            if (!fout.is_open())
                throw IncorrectFileOpen("Files is not opened.");

        }
        catch (IncorrectFileOpen& err){
            err.display();
            exit(1);
        }
        fout << user;
        fout.close();
    }
    friend ostream& operator << (std::ostream &os, User &u);
    friend istream& operator >> (std::istream& in, User& u);
};
std::ostream& operator << (std::ostream &os, User &u)
{
    os <<"\n"<< u.Name << " " << u.Surname << " " << u.PasswordAccount <<
" " << u.ServiceName<< " " << u.PhoneNumber << " " << u.Car;
    return os ;
}
std::istream& operator >> (std::istream& in, User& u)
{
    in >> u.Name >> u.Surname >> u.PasswordAccount >> u.ServiceName >>
p.PhoneNumber >> u.Car;
    return in;
}
```

```
#endif /* User_111_h */

#ifndef Admin_h
#define Admin_h

#include "Services.h"
#include "User.h"
#include "Exception.h"
#include "InCorrectInput.h"
#include "Container.h"
#include <iostream>

using namespace std;

class Admin: public User {

private:
    string Password = "qwertyadmin2022";

public:

    Admin(){};
    ~Admin(){};

    void AdminLogin(){

        string _password;
        cout << endl << endl << "Enter password for admin:" ;
        cin >> _password;

        if(Password == _password){
            system("clear");
            ForAdmin();
        }
        else
            cout << endl << "Incorrect password!" << endl;
    }

     void Admin() {
        int number = 1;
        User user;

        while (number != 0) {
            while (1) {
                cout << endl << endl << "Menu for admin:" << endl;
                cout << "1 - Display all users" << endl;
                cout << "2 - Delete user by phone number" << endl;
                cout << "3 - Find user by phonne number" << endl;
                cout << "0 - main menu" << endl;

                try {
                    rewind(stdin);
                    cin.clear();
                    if (!(cin >> choice))
                        throw IncorrectInput("Error, enter int.");
                    break;
                }
                catch (IncorrectInput &err) {
                    err.display();
                    continue;
                }
            }
            switch (choice) {

                case 1:
                {
                    system("clear");
                    FileAdmin();
                    break;
                }
                case 2:
                {
                    system("clear");
                    DeleteFileAdmin();
                    break;
                }
                case 3:
                {
                    system("clear");
                    FileAdminPhoneNumber();
                    break;
                }
                case 0:
                {
                    system("clear");
                    choice = 0;
                    break;
                }
                default:{continue;}
            }
        }
    }

     void FileAdmin(){

        User a;
        List<User> users;
        string Path = "Users.txt";
        fstream fin;
        fin.open(Path);

         try {
             if (!fin.is_open())
                 throw IncorrectFileOpen("Files is not opened!");
         }
         catch (IncorrectFileOpen& err){
             err.display();
             exit(1);
         }
        fin.close();

         if(!users.isEmpty()){
             DisplayUsers();
             for (int i = 0; i < users.getSize(); ++i) {
                 users[i].GetUser();
             }
         }
         users.clear();
     }
```

```
 void FileAdminPhoneNumber(){

    User a;
    string fileidentif;

    cout << "Enter number for search:";
    cin >> fileidentif;

    string Path = "Clients.txt";
    fstream fin;
    fin.open(Path);

     try {
         if (!fin.is_open())
             throw IncorrectFileOpen("Files is not opened!");
     }
     catch (IncorrectFileOpen& err){
         err.display();
         exit(1);
     }
    while(!(fin.eof())){
        fin >> a;
        if(x.GetPhoneNumber() == fileidentif){
            DisplayUsers();
           a.GetUser();
        }
    }
    fin.close();
}

 void DeleteFileAdmin(){

    vector<User> users;
    string PhoneNumber;
    cout << "Enter phone number to delete:";
    cin >> PhoneNumber;
    User a;
    string Path = "Users.txt";
    fstream fin;
    fin.open(Path);

     try {
         if (!fin.is_open())
             throw IncorrectFileOpen("Files is not opened.");
     }
     catch (IncorrectFileOpen& err){
         err.display();
         exit(1);
     }
     int count = 0;
    while(!(fin.eof())){
        fin >> a;
        if(strcmp(a.GetPhoneNumber() == PhoneNumber)){
            count += 1;
            continue;
        }else{
            users.push_back(x);
        }
    }
```

```
        fin.close();
          cout << endl<< endl << "Client successfully deleted!" << endl;
        fstream fout;
        fout.open(Path, ios::trunc | ios::out | ios::in);

         try {
             if (!fout.is_open())
                 throw IncorrectFileOpen("Files is not opened.");
         }
         catch (IncorrectFileOpen& err){
             err.display();
             exit(1);
         }
        for (int i = 0; i < users.size(); ++i) {

            fout << users[i];
        }
        fout.close();
        users.clear();
    }
};

#endif /* Admin_h */

#ifndef Exception_h
#define Exception_h

#include <iostream>

using namespace std;

class Exception{

public:
    string ErrorMessage;

public:
    Exception() {
        ErrorMessage = "Error!";
    }
    Exception(string messg)
    {
        ErrorMessage = messg;
    }
    ~Exception() {}

     void DisplayMessage()
    {
        cout << endl << endl << "Error: " << ErrorMessage << endl <<
endl;
    }
};

#endif /* Exception_h */

#ifndef IncorrectFileOpen_h
#define IncorrectFileOpen_h

#include "Exception.h"
```

```
class InCorrectFileOpen : public Exception {

public:

    InCorrectFileOpen(string ErrorMessage){

        this->ErrorMessage = ErrorMessage;
    }
};
#ifndef IncorrectInput_h
#define IncorrectInput_h

#include "Exception.h"

class IncorrectInput: public Exception {

public:
    IncorrectInput(string ErrorMessage){
        this->ErrorMessage = ErrorMessage;
    }
};
#ifndef IncorrectPasswordLength_h
#define IncorrectPasswordLength_h

#include "Exception.h"

class IncorrectPasswordLength: public Exception {

public:
    IncorrectPasswordLength(string ErrorMessage) {

        this->ErrorMessage = ErrorMessage;
    }
};
#ifndef Container_h
#define Container_h

#include <iostream>

using namespace std;

template <typename T>
class List {

public:

    List();
    ~List();
    void push_back(T data);
    int getSize();
    T& operator[](const int index);
    void clear();
    bool isEmpty();

private  :

    template<typename T1>
    class Node {

    public:
        Node* next;
        T1 data;
        Node(T1 data = T(), Node* next = nullptr) {
            this->data = data;
            this->next = next;
        }
    };
    int size;
    Node<T>* first;
    Node<T>* last;

};

template <typename T>
List<T>::List() {
    size = 0;
    first = last = nullptr;
}

template <typename T>
void List<T>::push_back(T data) {
    if (first == nullptr) {
        try {
            first = new Node<T>(data);
            last = first;
        }
        catch (bad_alloc& e) {
            cout << e.what() << endl;
        }
    }
    else {

        if (last->next == nullptr) {
            try {
                last->next = new Node<T>(data);
                last = last->next;
            }
            catch (bad_alloc& e) {
                cout<<e.what()<<endl;
            }
        }
    }
    size++;
}


template <typename T>
int List<T>::getSize() {
    return size;
}

template<typename T>
T& List<T>::operator[](const int index) {
    int counter = 0;
    Node<T>* current = this->first;
    while (current != nullptr) {

        if (counter == index) {
            return current->data;
        }
        current = current->next;
        counter++;
    }

    return current->data;
}

template<typename T>
void List<T>::clear() {
    Node<T>* temp;
    while (size) {
        temp = first;
        first = first->next;
        delete temp;
        size--;
    }
}

template<typename T>
bool List<T>::isEmpty() {
    if (this->size == 0)
        return true;
    else
        return false;
}

template<typename T>
List<T>::~List() {
    this->clear();
}

#endif /* Container_h */
```