МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №1

«Взаимодействие классов»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

Кустов Максим Александрович

Череповец, 2024 год

**Цель работы:**

* изучить способы взаимодействия классов;

**Требования к разработке:**

1. Каждый класс должен быть оформлен в отдельных файлах: заголовочный (.h) и файл с кодом (.cpp).
2. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
3. Придерживаться принципа DRY (Don’t repeat yourself).
4. Обязательно наличие комментариев.

**Задание на лабораторную работу:**

Необходимо создать еще один класс по своему варианту (вариант №14 – устройства передачи данных) и реализовать его взаимодействие с исходным классом, созданным в предыдущих лабораторных работах.

**Текст программы**

Заголовочный файл **Queue.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

template <class T>

class Queue {

private:

struct Node { //узел очереди

T data;

Node\* next;

};

Node\* head;

Node\* tail;

int size = 0;

public:

Queue() { //конструктор

head = NULL;

tail = NULL;

}

bool empty() { //проверка очереди на пустоту

return head == NULL;

}

void add(T value) { //добавление элемента в очередь

if (empty()) { //если она пустая

head = new Node;

head->data = value;

head->next = NULL;

tail = head;

size++;

}

else { //если она не пустая

tail->next = new Node;

tail = tail->next;

tail->data = value;

tail->next = NULL;

size++;

}

}

T del() { //взятие и удаление элемента из очереди

if (empty()) {

cout << "Queue is empty" << endl;

}

else {

T d = head->data;

Node\* tmp = head;

head = head->next;

delete(tmp);

return d;

}

}

void nullQueue() { //обнуление очереди

Node\* tmp;

while (!empty()) {

tmp = head;

head = head->next;

delete(tmp);

}

}

bool operator >= (const Queue& other) {

return this->size >= other.size;

}

bool operator < (const Queue& other) {

return this->size < other.size;

}

int getSize() { //геттер для логина

return size;

}

friend ostream& operator<<(ostream& del, Queue& q) {

while (!q.empty())

del << q.del() << endl;

return del;

}

void arr(T\* a, int& i) {

while (!empty()) {

a[i] = del();

i++;

}

}

void print\_NewOld() {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

i--;

for (i; i >= 0; i--)

cout << networks[i] << endl;

}

void print\_AZ() {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

string k;

for (i = 0; i < size; i++)

for (int j = i; j < size; j++) {

if (networks[i] > networks[j]) {

k = networks[i];

networks[i] = networks[j];

networks[j] = k;

}

}

for (i = 0; i < size; i++)

cout << networks[i] << endl;

}

void print\_ZA() {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

string k;

for (i = 0; i < size; i++)

for (int j = i; j < size; j++) {

if (networks[i] < networks[j]) {

k = networks[i];

networks[i] = networks[j];

networks[j] = k;

}

}

for (i = 0; i < size; i++)

cout << networks[i] << endl;

}

void search(string name) {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

int k2 = 0;

for (i = 0; i < size; i++) {

int k1 = 0;

string s;

for (int l = 0; l < name.length(); l++) {

s = networks[i];

if (s[l] == name[l]) k1++;

}

if (k1 == name.length()) {

if (k2 == 0) cout << endl << "Найденные сети:" << endl;

cout << networks[i] << endl;

k2++;

}

}

if (k2 == 0) cout << "Ничего не найдено!" << endl;

}

void search(int number) {

int\* networks = new int[size];

int i = 0, k = 0;

arr(networks, i);

for (i = 0; i < size; i++) {

if (networks[i] == number) k++;

}

if (k == 0) cout << "Подключений к сети №" << number << " не было" << endl;

else cout << "К сети №" << number << " подключались " << k << " раз(а)" << endl;

}

~Queue() { //деструктор

}

};

Заголовочный файл **WiFi.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

enum security { WEP, WPA, WPA2, WPA3, NoProtection };

class Site;

class WiFi {

private:

string login, password;

static string work;

security protection;

int speed, radius, traffic;

double frequency, time;

public:

bool operator > (const WiFi& other) {

return this->speed > other.speed;

}

bool operator < (const WiFi& other) {

return this->speed < other.speed;

}

bool operator == (const WiFi& other) {

return this->speed == other.speed;

}

int operator + (const WiFi& other) {

return this->traffic + other.traffic;

}

WiFi(string valueLogin, string valuePassword, security valueProtection, int valueSpeed, int valueRadius, double valueFrequency) {

this->login = valueLogin; //конструктор со всеми параметрами

this->password = valuePassword;

this->protection = valueProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi() {

this->login = "Неопределенная сеть"; //конструктор по умолчанию

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = 0;

this->time = rand() / 100.0;

}

WiFi(string valueLogin) {

this->login = valueLogin; //конструктор с 1 параметром

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(double valueFrequency, string valueLogin) {

this->login = valueLogin; //конструктор с 2 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(int valueSpeed, int valueRadius, string valueLogin) {

this->login = valueLogin; //конструктор с 3 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(const WiFi& obj) {

this->login = obj.login; //копирующий конструктор

this->password = obj.password;

this->protection = obj.protection;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getLogin() { //геттер для логина

return login;

}

string getPassword() { //геттер для пароля

return password;

}

int getTraffic() {

return traffic;

}

double getTime() {

return time;

}

string getProtection() { //геттер для протокола безопасности

switch (protection) {

case WEP: return "WEP";

case WPA: return "WPA";

case WPA2: return "WPA2";

case WPA3: return "WPA3";

case NoProtection: return "Без защиты";

default: return "N/A";

}

}

template<class T>

T sum(T a, T b) {

return a + b;

}

friend ostream& operator<<(ostream& info, WiFi& other) { //вывод информации о сети

info << "Вы успешно подключились к WiFi-сети " << other.login << endl << endl;

info << "Информация о сети" << endl;

info << "Скорость: " << other.speed << " Мбит/с" << endl;

info << "Частота: " << other.frequency << " ГГц" << endl;

info << "Радиус покрытия: " << other.radius << " м" << endl;

info << "Протокол безопасности: " << other.getProtection() << endl;

info << "Cеть работает: " << other.work << endl;

info << "Передано данных: " << other.traffic << " Мб" << endl;

info << "Время соединения: " << other.time << " сек" << endl;

info << endl << "=======================" << endl << endl;

return info;

}

void openSite(Site& site); //функция, взаимодействующая с объектом класса Site

~WiFi() { //деструктор

}

};

Заголовочный файл **Site.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

enum protocol { http, https, NoProtocol };

class WiFi;

class Site {

private:

string name, type, address;

static string work;

double time;

protocol connect;

public:

Site(string valueName, string valueType, string valueAddress, protocol valueConnect) {

this->name = valueName; //конструктор со всеми параметрами

this->type = valueType;

this->address = valueAddress;

this->connect = valueConnect;

this->time = rand() / 1000.0;

}

Site() {

this->name = "Неизвестный сайт"; //конструктор по умолчанию

this->type = "Неизвестный тип";

this->address = "about:blank";

this->connect = NoProtocol;

this->time = rand() / 1000.0;

}

Site(string valueAddress) {

this->name = "Неизвестный сайт"; //конструктор c 1 параметром

this->type = "Неизвестный тип";

this->address = valueAddress;

this->connect = NoProtocol;

this->time = rand() / 1000.0;

}

Site(string valueAddress, protocol valueConnect) {

this->name = "Неизвестный сайт"; //конструктор c 2 параметрами

this->type = "Неизвестный тип";

this->address = valueAddress;

this->connect = valueConnect;

this->time = rand() / 1000.0;

}

Site(string valueName, string valueAddress, protocol valueConnect) {

this->name = "Неизвестный сайт"; //конструктор c 3 параметрами

this->type = "Неизвестный тип";

this->address = valueAddress;

this->connect = valueConnect;

this->time = rand() / 1000.0;

}

Site(const Site& obj) {

this->name = obj.name; //копирующий конструктор

this->type = obj.type;

this->address = obj.address;

this->connect = obj.connect;

this->time = rand() / 1000.0;

}

string getName() { //геттер для названия сайта

return name;

}

string getWork() { //геттер для статуса работы сайта

return work;

}

double getTime() { //геттер для времени загрузки сайта

return time;

}

friend ostream& operator<<(ostream& info, Site& other) {

switch (other.connect) {

case NoProtocol: info << other.name << " - " << other.address << " (" << other.type << ")";

break;

case http: info << other.name << " - " << " http://" << other.address << " (" << other.type << ")";

break;

case https: info << other.name << " - " << " https://" << other.address << " (" << other.type << ")";

break;

}

return info;

}

void statusWiFi(WiFi& wifi); //функция, взаимодействующая с объектом класса WiFi

~Site() { //деструктор

}

};

Файл с кодом **лб1.cpp**:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "WiFi.h"

#include "Queue.h"

#include "Site.h"

using namespace std;

string WiFi::work = "Да";

string Site::work = "Да";

void WiFi::openSite(Site& site) { //функция класса WiFi, взаимодействующая с объектом класса Site

cout << "Сообщение от провайдера: Вы успешно зашли на сайт " << site.getName() << " через Wi-Fi сеть " << getLogin() << endl;

cout << "Сайт работает: " << site.getWork() << endl;

cout << "Время загрузки сайта: " << site.getTime() << " мс" << endl << endl;

}

void Site::statusWiFi(WiFi& wifi) { //функция класса Site, взаимодействующая с объектом класса WiFi

cout << "Сообщение от сайта: Подключение к сайту " << getName() << " осуществляется через Wi-Fi сеть " << wifi.getLogin() << endl;

cout << "Протокол безопасности сети: " << wifi.getProtection() << endl << endl;

}

void NetworkError(int err) { //функция вывода ошибок

switch (err) {

case 1: cout << "Такой WiFi-сети нет!" << endl << endl;

break;

case 2: cout << "Пароль неверный!" << endl << endl;

break;

case 3: cout << "Такого варианта нет!" << endl << endl;

break;

case 4: cout << "Такого сайта нет!" << endl << endl;

}

}

WiFi authorization(int number, int& traffic, double& time, string password, WiFi& Network, Queue<string>& q, Queue<string>& error, Queue<int>& q\_num, int& correct) {

if (password == Network.getPassword()) { //если пароль верный, то выводится информация о сети

WiFi Network\_Connected(Network);

q.add(Network.getLogin());

q\_num.add(number);

correct++;

cout << Network;

traffic = Network.sum(traffic, Network.getTraffic());

time = Network.sum(time, Network.getTime());

return Network\_Connected;

}

else {

NetworkError(2); //если пароль неверный, выводится сообщение об ошибке

WiFi Network\_Disconnected(Network);

error.add(Network.getLogin());

return Network\_Disconnected;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(0));

int number, number\_site, count = 0, traffic = 0, correct = 0;

double time = 0;

string password, name;

Queue<string> q, error;

Queue<int> q\_num;

while (true) {

WiFi Network\_1("CHSU\_FREE", "0", security::NoProtection, 100, 70, 5); //описание доступных WiFi-сетей

WiFi Network\_2("CHSU\_WORK", "1234567890", security::WPA3, 120, 50, 5);

WiFi Network\_3("IPhone", "ghfdgldklkl", security::WPA2, 80, 40, 2.4);

WiFi Network\_4("Xiaomi", "h5hjg54g5", security::WPA2, 70, 40, 2.4);

WiFi Network\_5;

WiFi Network\_6(Network\_1);

WiFi Network\_7(2.4, "Rostelecom");

WiFi Network\_8(100, 60, "Beeline");

WiFi Network\_9("iZet");

Site Site\_1("Google", "Поисковик", "www.google.com", protocol::https);

Site Site\_2("Яндекс", "Поисковик", "www.ya.ru", protocol::https);

Site Site\_3("ВКонтакте", "Социальная сеть", "www.vk.com", protocol::https);

Site Site\_4("YouTube", "Видеохостинг", "www.youtube.com", protocol::https);

Site Site\_5;

Site Site\_6(Site\_1);

Site Site\_7("example.com");

Site Site\_8("www.edu.chsu.ru", protocol::http);

Site Site\_9("ЧГУ", "www.chsu.ru", protocol::https);

cout << "Доступные сети: " << endl; //вывод доступных сетей на экран

cout << "1. " << Network\_1.getLogin() << endl;

cout << "2. " << Network\_2.getLogin() << endl;

cout << "3. " << Network\_3.getLogin() << endl;

cout << "4. " << Network\_4.getLogin() << endl;

cout << "5. " << Network\_5.getLogin() << endl;

cout << "6. " << Network\_6.getLogin() << endl;

cout << "7. " << Network\_7.getLogin() << endl;

cout << "8. " << Network\_8.getLogin() << endl;

cout << "9. " << Network\_9.getLogin() << endl << endl;

cout << "0. " << "Вывести историю подключения к сетям" << endl;

if (count == 0) {

cout << endl << "Рекомендации: " << endl;

cout << "Выбирая между сетями " << Network\_1.getLogin() << " и " << Network\_2.getLogin() << " рекомендуем выбрать сеть ";

if (Network\_1 > Network\_2) cout << Network\_1.getLogin();

else if (Network\_1 < Network\_2) cout << Network\_2.getLogin();

cout << ", т.к. её скорость выше" << endl;

if (Network\_5 == Network\_9) cout << "Скорость у сетей " << Network\_5.getLogin() << " и " << Network\_9.getLogin() << " одинакова" << endl;

if (Network\_3 == Network\_4) cout << "Скорость у сетей " << Network\_3.getLogin() << " и " << Network\_4.getLogin() << " одинакова" << endl;

cout << "С помощью сетей " << Network\_7.getLogin() << " и " << Network\_8.getLogin() << " можно передать " << Network\_7 + Network\_8 << " Мб данных" << endl;

}

cout << endl << "Введите номер сети, к которой хотите подключиться: "; //выбор сети

cin >> number;

count++;

if (number < 0 || number > 9) { //если введен номер несуществуещей сети, выводится сообщение об ошибке

NetworkError(1);

continue;

}

else if (number == 0 && !q.empty()) {

int num;

if (q >= error) cout << endl << "Было совершено " << q.getSize() << " удачных подключений и " << error.getSize() << " неудачных" << endl;

if (q < error) cout << endl << "Было совершено " << error.getSize() << " неудачных подключений и " << q.getSize() << " удачных" << endl;

cout << "Всего было передано " << traffic << " Мб данных" << endl;

cout << "Общее время соединения: " << time << " сек" << endl;

cout << endl << "Порядок вывода:" << endl; //вывод доступных сетей на экран

cout << "1. " << "В хронологическом (от старых подключений к новым)" << endl;

cout << "2. " << "В хронологическом (от новых подключений к старым)" << endl;

cout << "3. " << "В алфавитном (A - Z)" << endl;

cout << "4. " << "В обратном алфавитному (Z - A)" << endl;

cout << "5. " << "Поиск по названию" << endl;

cout << "6. " << "Поиск по номеру" << endl;

cout << endl << "Выберите порядок: ";

cin >> num;

if (num < 1 || num > 6) {

NetworkError(3);

continue;

}

if (num >= 1 && num <= 4) cout << endl << "История подключений: " << endl;

switch (num) {

case 1: cout << q;

break;

case 2: q.print\_NewOld();

break;

case 3: q.print\_AZ();

break;

case 4: q.print\_ZA();

break;

case 5: {

cout << endl << "Введите название сети: ";

cin >> name;

cin.get();

q.search(name);

}

break;

case 6: {

cout << endl << "Доступные сети: " << endl; //вывод доступных сетей на экран

cout << "1. " << Network\_1.getLogin() << endl;

cout << "2. " << Network\_2.getLogin() << endl;

cout << "3. " << Network\_3.getLogin() << endl;

cout << "4. " << Network\_4.getLogin() << endl;

cout << "5. " << Network\_5.getLogin() << endl;

cout << "6. " << Network\_6.getLogin() << endl;

cout << "7. " << Network\_7.getLogin() << endl;

cout << "8. " << Network\_8.getLogin() << endl;

cout << "9. " << Network\_9.getLogin() << endl;

cout << endl << "Введите номер сети: ";

int number2;

cin >> number2;

cin.get();

q\_num.search(number2);

}

break;

}

return 0;

}

else if (number == 0 && q.empty()) {

cout << "Подключений не было" << endl;

return 0;

}

cout << "Введите пароль: "; //ввод пароля

cin >> password;

cin.get();

WiFi Network\_Connected;

correct = 0;

switch (number) {

case 1: Network\_Connected = authorization(number, traffic, time, password, Network\_1, q, error, q\_num, correct);

break;

case 2: Network\_Connected = authorization(number, traffic, time, password, Network\_2, q, error, q\_num, correct);

break;

case 3: Network\_Connected = authorization(number, traffic, time, password, Network\_3, q, error, q\_num, correct);

break;

case 4: Network\_Connected = authorization(number, traffic, time, password, Network\_4, q, error, q\_num, correct);

break;

case 5: Network\_Connected = authorization(number, traffic, time, password, Network\_5, q, error, q\_num, correct);

break;

case 6: Network\_Connected = authorization(number, traffic, time, password, Network\_6, q, error, q\_num, correct);

break;

case 7: Network\_Connected = authorization(number, traffic, time, password, Network\_7, q, error, q\_num, correct);

break;

case 8: Network\_Connected = authorization(number, traffic, time, password, Network\_8, q, error, q\_num, correct);

break;

case 9: Network\_Connected = authorization(number, traffic, time, password, Network\_9, q, error, q\_num, correct);

break;

}

while (correct == 1) {

cout << "Доступные сайты: " << endl; //вывод доступных сетей на экран

cout << "1. " << Site\_1 << endl;

cout << "2. " << Site\_2 << endl;

cout << "3. " << Site\_3 << endl;

cout << "4. " << Site\_4 << endl;

cout << "5. " << Site\_5 << endl;

cout << "6. " << Site\_6 << endl;

cout << "7. " << Site\_7 << endl;

cout << "8. " << Site\_8 << endl;

cout << "9. " << Site\_9 << endl << endl;

cout << "0. Отключиться от Wi-Fi сети " << Network\_Connected.getLogin() << endl << endl;

cout << "Выберите сайт: ";

cin >> number\_site;

switch (number\_site) {

case 1: {

Network\_Connected.openSite(Site\_1);

Site\_1.statusWiFi(Network\_Connected);

}

break;

case 2: {

Network\_Connected.openSite(Site\_2);

Site\_2.statusWiFi(Network\_Connected);

}

break;

case 3: {

Network\_Connected.openSite(Site\_3);

Site\_3.statusWiFi(Network\_Connected);

}

break;

case 4: {

Network\_Connected.openSite(Site\_4);

Site\_4.statusWiFi(Network\_Connected);

}

break;

case 5: {

Network\_Connected.openSite(Site\_5);

Site\_5.statusWiFi(Network\_Connected);

}

break;

case 6: {

Network\_Connected.openSite(Site\_6);

Site\_6.statusWiFi(Network\_Connected);

}

break;

case 7: {

Network\_Connected.openSite(Site\_7);

Site\_7.statusWiFi(Network\_Connected);

}

break;

case 8: {

Network\_Connected.openSite(Site\_8);

Site\_8.statusWiFi(Network\_Connected);

}

break;

case 9: {

Network\_Connected.openSite(Site\_9);

Site\_9.statusWiFi(Network\_Connected);

}

break;

case 0: {

cout << "Вы отключились от Wi-Fi сети " << Network\_Connected.getLogin() << endl << endl;

correct--;

}

break;

default: NetworkError(4);

break;

}

}

}

system("pause");

return 0;

}

**Схема классов**



**Описание программы**

В программу был добавлен класс **Site**, объекты которого являются сайтами, к которым подключается устройство через Wi-Fi сети – объекты класса **WiFi**.

В закрытой части класс содержит следующие поля:

* **name** – строка типа **string**, название сайта
* **type** - строка типа **string**, тип сайта
* **address** - строка типа **string**, адрес сайта
* **work** - строка типа **string**, статическая переменная, показывает, работает ли сайт, всегда имеет значение *«Да»*
* **time** – число типа **double**, время загрузки сайта
* **connect** – переменная-перечисление типа **protocol**, протокол сайта, может принимать значения *http*, *https* и *NoProtocol*

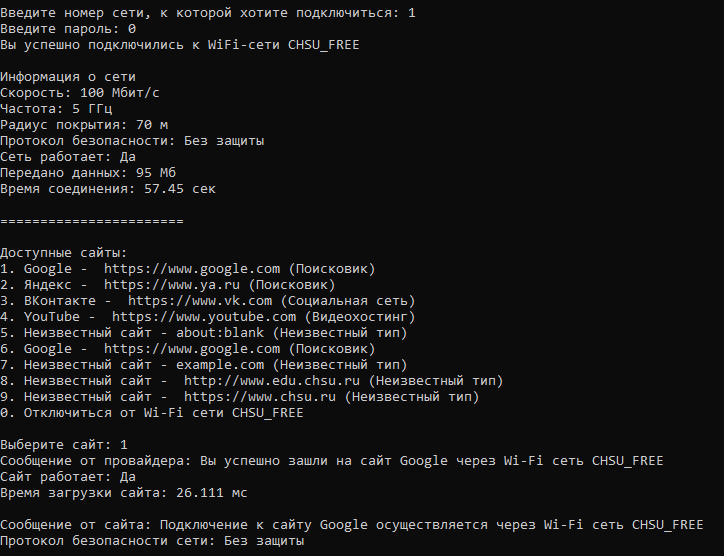
Открытая часть класса содержит следующие функции:

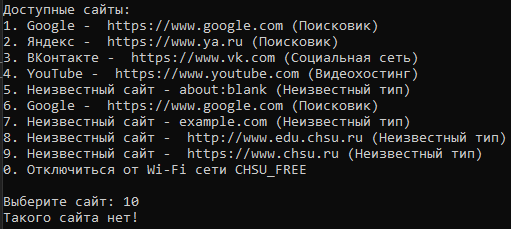
* Конструкторы: со всеми параметрами, по умолчанию, с 1, 2 и 3 параметрами и копирующий.
* Геттеры: для переменных **name**, **work** и **time**
* Перегрузка оператора **<<**
* Деструктор
* Функция **statusWiFi**, параметром которой является объект класса **WiFi**. В этой функции происходит взаимодействие классов. Функция представляет собой сообщение от сайта о Wi-Fi сети, через которую произведен вход на сайт

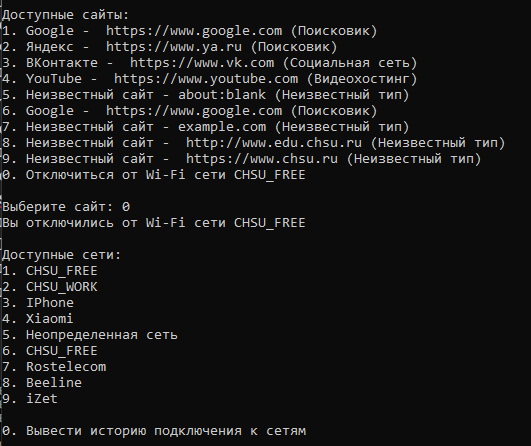
В класс **WiFi** добавлена функция **openSite**, параметром которой является объект класса **Site**. В этой функции происходит взаимодействие классов. Функция представляет собой сообщение от провайдера о сайте, на который пользователь зашел через Wi-Fi сеть.

Теперь после подключения к Wi-Fi сети программа не предлагает сразу же снова подключаться к сети. Вместо этого она выдает список сайтов (объектов класса **Site**, их было реализовано 9 штук) и предлагает ввести число – номер сайта. После выбора номера сайта выполняются функции **openSite** и **statusWiFi**, в которых классы взаимодействуют друг с другом. После подключения к сайту программа предлагает подключиться к другому сайту. При вводе цифры 0 происходит отключение от Wi-Fi сети, после чего программа предлагает подключиться к другой Wi-Fi сети либо вывести историю подключений к сетям.

**Результаты тестов**







**Вывод**

В ходе выполнения лабораторной работы были изучены способы взаимодействия классов.

Был создан еще один класс. В обоих классах были реализованы функции, взаимодействующие с объектами другого класса.