-7

Специальность: 09.02.07 Информационные системы и программирование

Выполнил:

Студент группы Y2333

Сааль С. Ф.

ОТЧЕТ

о лабораторной работе № 8

по теме: Классы

по дисциплине: Разработка программных модулей

Санкт-Петербург 2020

Проверил:

Антонов. М.Б.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УНИВЕРСИТЕТ ИТМО

Факультет среднего профессионального образования

1. Цель и задачи

Цель: реализация работы с классами, освоение основных принципов ООП.

1. Задание

Написать программу учета заявок на обмен квартир и поиска вариантов обмена.

Каждая заявка содержит сведения о двух квартирах требуемой (искомой) и имеющейся. Сведения о каждой квартире содержат: количество комнат, площадь, этаж, район.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

* ввод заявки на обмен;
* поиск в картотеке подходящего варианта: при совпадении требований и предложений по количеству комнат и этажности и различии по показателю «площадь» в пределах 10% выводится соответствующая карточка и удаляется из списка, в противном случае поступившая заявка включается в картотеку;
* вывод всей картотеки.

Для хранения данных картотеки использовать контейнерный класс list.

1. КОд программы

Файл main.cpp:

/\*\*

\* Точка входа

\* @file main.com

\* @author Сааль Степан

\*/

#include <iostream>

#include "Menu.h"

/\*\*

\* Вывод меню

\* @return

\*/

int main() {

auto \*menu = new Menu;

menu->show();

return 0;

}

Файл Menu.h:

/\*\*

\* Меню для картотеки

\* @file Menu.h

\* @author Сааль Степан

\*/

#ifndef LAB8\_MENU\_H

#define LAB8\_MENU\_H

#include <iostream>

#include "FileCabinet.h"

#include "FlatApplication.h"

#include "Flat.h"

#include <string>

using namespace std;

/\*\*

\* Меню

\*/

class Menu {

public:

/\*\* Создание новой картотеки \*/

Menu() {

fileCabinet = new FileCabinet();

}

/\*\*

\* Показать главное меню

\*/

void show() {

cout << endl;

cout << "1. Show file-cabinet" << endl;

cout << "2. New request" << endl;

cout << "0. Exit" << endl;

mainMenuAction();

}

private:

/\*\* Картотека \*/

FileCabinet \*fileCabinet;

/\*\*

\* Действия главного меню

\*/

void mainMenuAction() {

int action;

cin >> action;

if (action == 1) {

showAction();

} else if (action == 2) {

newRequest();

} else if (action == 0) {

cout << "Bye!";

return;

} else {

cout << "Wrong action, try again:" << endl;

}

show();

}

/\*\*

\* Показать картотеку

\*/

void showAction() {

cout << "File-cabinet:" << endl << endl;

fileCabinet->show();

cout << "END" << endl << endl;

}

/\*\*

\* Записать данные о квартире

\* @return

\*/

Flat \*createFlat() {

int rooms, floor;

float area;

string distinct;

cout << "Input rooms:" << endl;

cin >> rooms;

cout << "Input floor:" << endl;

cin >> floor;

cout << "Input area:" << endl;

cin >> area;

cout << "Input distinct:" << endl;

cin >> distinct;

return new Flat(rooms, area, floor, distinct);

}

/\*\*

\* Записать данные о требовании и предложении

\* @return

\*/

FlatApplication \*createApplication() {

Flat \*offer, \*requirement;

cout << "Offer:" << endl;

offer = createFlat();

cout << "Requirement:" << endl;

requirement = createFlat();

return new FlatApplication(requirement, offer);

}

/\*\*

\* Создать новый запрос

\*/

void newRequest() {

FlatApplication \*application = createApplication();

Flat \*response = fileCabinet->newRequest(application);

if (response == nullptr) {

cout << "No suitable apartments, you are added to the list." << endl;

} else {

cout << "You made an exchange! You received:" << endl;

cout << "rooms:" << response->getRooms()

<< " floor: " << response->getFloor()

<< " area: " << response->getArea()

<< " distinct: " << response->getDistrict() << endl << endl;

}

}

};

#endif //LAB8\_MENU\_H

Файл FileCabinet.h:

/\*\*

\* Картотека

\* @file FileCabinet.h

\* @author Сааль Степан

\*/

#ifndef LAB8\_FILECABINET\_H

#define LAB8\_FILECABINET\_H

#include "Flat.h"

#include "FlatApplication.h"

#include <list>

#include <iostream>

using namespace std;

/\*\*

\* Картотека

\*/

class FileCabinet {

public:

/\*\*

\* Новый запрос

\* @param application

\* @return

\*/

Flat \*newRequest(FlatApplication \*application) {

auto it = applications.begin();

for (FlatApplication\* item : applications) {

if (item->isEqual(application)) {

applications.erase(it);

return item->getOffer();

}

advance(it, 1);

}

applications.push\_back(application);

return nullptr;

}

/\*\*

\* Показать картотеку

\*/

void show() {

for (FlatApplication\* item : applications) {

cout << "Offer => rooms: " << item->getOffer()->getRooms()

<< " floor: " << item->getOffer()->getFloor()

<< " area: " << item->getOffer()->getArea()

<< " distinct: " << item->getOffer()->getDistrict() << endl;

cout << "Requirement => rooms: " << item->getRequirement()->getRooms()

<< " floor: " << item->getRequirement()->getFloor()

<< " area: " << item->getRequirement()->getArea()

<< " distinct: " << item->getRequirement()->getDistrict() << endl << endl;

}

}

private:

/\*\* Список запросов \*/

list<FlatApplication\*> applications = {};

};

#endif //LAB8\_FILECABINET\_H

Файл FlatApplication.h:

/\*\*

\* Требование и предложение

\* @file FlatApplication.h

\* @author Сааль Степан

\*/

#ifndef LAB8\_FLATAPPLICATION\_H

#define LAB8\_FLATAPPLICATION\_H

#include "Flat.h"

/\*\*

\* Требование и предложение

\*/

class FlatApplication {

public:

FlatApplication(Flat \*requirement, Flat \*offer) : requirement(requirement), offer(offer) {};

~FlatApplication() {

delete requirement;

delete offer;

}

/\*\*

\* Проверка, подходят ли два запроса друг к другу

\* @param another

\* @return

\*/

bool isEqual(FlatApplication \*another)

{

return requirement->isEqual(another->getOffer()) && offer->isEqual(another->getRequirement());

}

/\*\*

\* Геттер для требования

\* @return

\*/

Flat \*getRequirement() const {

return requirement;

}

/\*\*

\* Геттер для предложения

\* @return

\*/

Flat \*getOffer() const {

return offer;

}

private:

Flat \*requirement;

Flat \*offer;

};

#endif //LAB8\_FLATAPPLICATION\_H

Файл Flat.h:

/\*\*

\* Квартира

\* @file Flat.h

\* @author Сааль Степан

\*/

#ifndef LAB8\_FLAT\_H

#define LAB8\_FLAT\_H

#include <string>

#include <cmath>

using namespace std;

/\*\*

\* Квартира

\*/

class Flat {

public:

/\*\*

\* Запись всех данных о квартире

\* @param rooms

\* @param area

\* @param floor

\* @param district

\*/

Flat(int rooms, float area, int floor, const string &district) :

rooms(rooms), area(area), floor(floor), district(district) {};

/\*\*

\* Проверка, схожи ли квартиры по требованиям

\* @param another

\* @return

\*/

bool isEqual(const Flat \*another) {

return rooms == another->getRooms() && floor == another->getFloor() &&

abs(1 - area / another->getArea()) < 0.1;

}

int getRooms() const {

return rooms;

}

float getArea() const {

return area;

}

int getFloor() const {

return floor;

}

const string &getDistrict() const {

return district;

}

private:

int rooms;

float area;

int floor;

string district;

};

#endif //LAB8\_FLAT\_H

В качестве результата работы программы представлены скриншоты результата выполнения программы на рисунках 1-6:

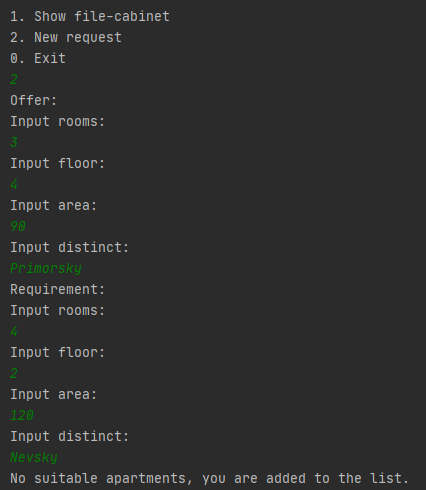


Рисунок 1

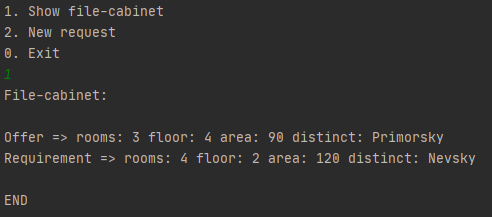


Рисунок 2

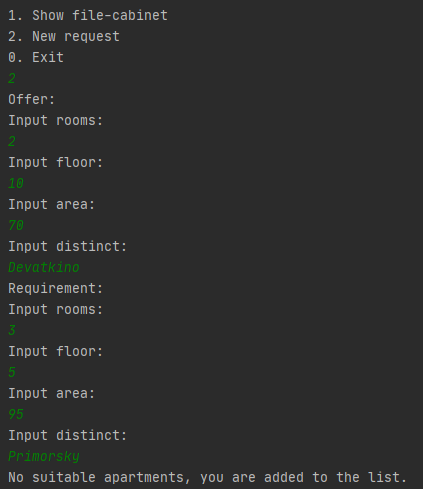


Рисунок 3

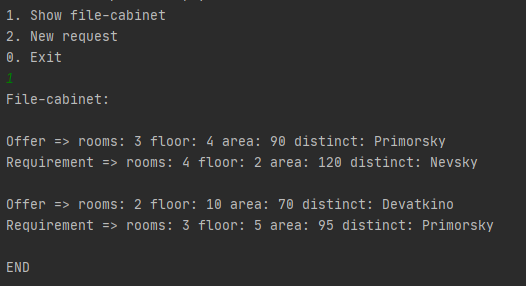


Рисунок 4

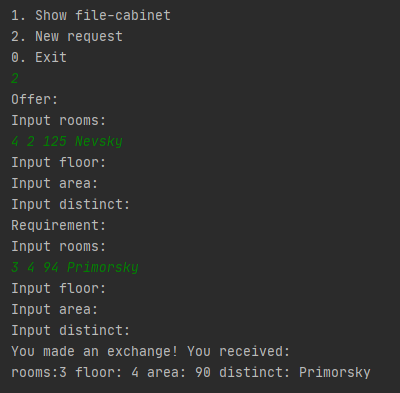


Рисунок 5

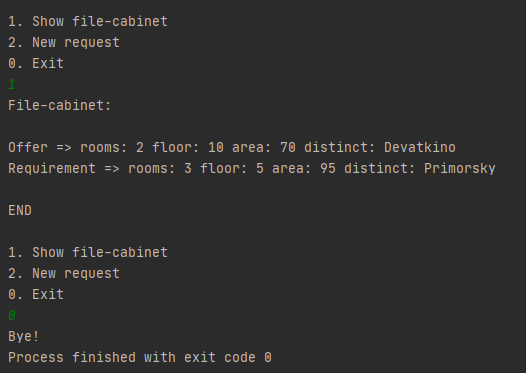


Рисунок 6

1. Диаграммы классов

Диаграмма классов представлена на рисунке 7.

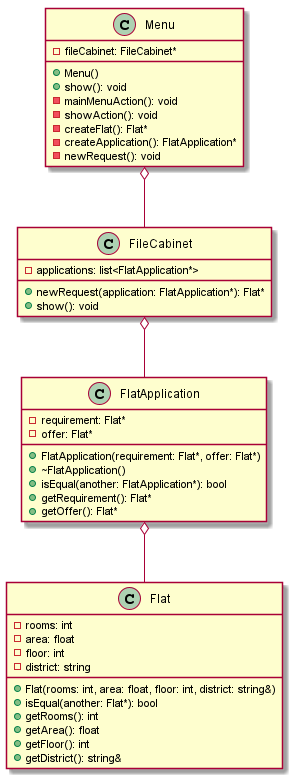


Рисунок 7

# ВЫВОД

В процессе выполнения лабораторной работы был получен опыт работы с классами ООП.