МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет имени М. Т. Калашникова»

Институт «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Работа защищена с оценкой«\_\_\_\_\_\_\_\_\_\_»

Дата \_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: Информационная система для обеспечения базы данных учета недвижимости»

Выполнил

студент гр. Б04-191-3 П. С. Костицын

Руководитель

к.т.н., доцент В. Г. Власов

Рецензия:

степень достижения поставленной целиработы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc43564492)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc43564493)

[2. ОПИСАНИЕ ТРЕБОВАНИЙ 5](#_Toc43564494)

[3. УТОЧНЕНИЕ ТРЕБОВАНИЙ 6](#_Toc43564495)

[4. КЛАССЫ 10](#_Toc43564496)

[4.1 Класс Program 10](#_Toc43564497)

[4.2 Класс ResidentialFacility 10](#_Toc43564498)

[4.3 Класс PrivateHouse 10](#_Toc43564499)

[4.4 Класс GardenHouse 11](#_Toc43564500)

[4.5 Класс Multistory 11](#_Toc43564501)

[4.6 Класс Apartment 11](#_Toc43564502)

[4.7 Класс NonResidentialFacility 12](#_Toc43564503)

[4.8 Класс Warehouse 12](#_Toc43564504)

[4.9 Класс Shop 12](#_Toc43564505)

[4.10 Класс Bank 12](#_Toc43564506)

[4.11 Иерархия классов 13](#_Toc43564507)

[3. БАЗА ДАННЫХ 14](#_Toc43564508)

[4. КОНТРОЛЬНЫЙ ПРИМЕР 18](#_Toc43564509)

[ЗАКЛЮЧЕНИЕ 31](#_Toc43564510)

[ПРИЛОЖЕНИЕ 32](#_Toc43564511)

# ВВЕДЕНИЕ

В наше непростое время вопросы учета недвижимости очень важны и, соответственно, необходима информационная система для обеспечения базы данных учета недвижимости. Для всех людей будет намного проще следить за списком домов,квартир,банков,магазинов,складов если у них будет база данных.  
 В данном курсовом проекте разработана информационная система для обеспечения базы данных учета недвижимости, и сама база данных. Разработанная система должна обеспечить удобный доступ к информации о недвижимости.

# 1. ПОСТАНОВКА ЗАДАЧИ

В курсовой работе должна быть разработана информационная система для обеспечения базы данных учета недвижимости. Система должна учитывать жилые (многоэтажные дома, частные дома, садовые дома) и нежилые (гаражи, склады, магазины, отд.банков, подвалы, хоз.постройки, баня, сарай, курятник, и т. п.) постройки. Причем для каждой постройки должна быть сформирована своя структура. Например, для жилого многоэтажного дома:

1. дом

2. квартира

3. комната

4. санузел

5. прихожая

и т.п.

# 2. ОПИСАНИЕ ТРЕБОВАНИЙ

Программа должна реализовывать следующие запросы:

1.Список всех многоэтажных домов

2.Список квартир в заданном доме

3.Список площадей в заданной квартире заданного дома

4.Список всех частных домов с нежилыми помещениями, привязанными к этому дому

5.Список всех садовых домов с нежилыми помещениями, привязанными к этому дому

6.Список всех банков

7.Список адресов конкретного банка

8.Список всех магазинов

9.Список адресов конкретного магазина

10.Добавить магазин

11.Добавить банк

12.Список всех складов

13.Список квартир с заданным количеством комнат

14.Добавить склад

15.Список банков на заданной улице

0. Выход

Программа выполнена на языке C#, при создании базы данных использовалась MS SQL.

# 3. УТОЧНЕНИЕ ТРЕБОВАНИЙ

При запуске программы должно появляться меню, содержащее следующие пункты:

1.Список всех многоэтажных домов

2.Список квартир в заданном доме

3.Список площадей в заданной квартире заданного дома

4.Список всех частных домов с нежилыми помещениями, привязанными к этому дому

5.Список всех садовых домов с нежилыми помещениями, привязанными к этому дому

6.Список всех банков

7.Список адресов конкретного банка

8.Список всех магазинов

9.Список адресов конкретного магазина

10.Добавить магазин

11.Добавить банк

12.Список всех складов

13.Список квартир с заданным количеством комнат

14.Добавить склад

15.Список банков на заданной улице

0. Выход

При выборе первого пункта меню на экране должен появляться список всех многоэтажных домов в виде таблицы, содержащей столбцы: адрес, номер дома, кол-во этажей и кол-во квартир. Также должна появляться фраза «Нажмите любую клавишу, чтобы вернуться в меню». После нажатия клавиши, должно снова выводиться меню. При выборе второго запроса пользователю должно предлагаться ввести адрес и номер дома, после чего на экране должна появиться таблица квартир, содержащей столбцы: номер квартиры, этаж, площадь квартиры, кол-во комнат. Если дома по этому адресу не найдено, программа должна вывести «Дом по данному адресу не найден». Возврат в меню аналогичен первому запросу. При выборе пользователем третьего запроса, пользователю должно предлагаться ввести адрес дома, номер дома и номер квартиры, после чего на экране должна появиться таблица, содержащая следующие столбцы: площадь комнат, площадь санузла и площадь прихожей. В случае, если адрес или номер квартиры введен неправильно или такого дома или квартиры нет в базе данных, программа сообщает о том, что квартира не найдена. Возврат в меню аналогичен первому запросу. При выборе пользователем четвертого запроса, на экране должен появляться список всех частных домов в виде таблицы, содержащей столбцы: адрес, номер дома, кол-во этажей, площадь дома, наличие гаража, площадь гаража, наличие подвала, площадь подвала, высота подвала. Возврат в меню аналогичен первому запросу. При выборе пользователем пятого запроса, на экране должен появляться список всех садовых домов в виде таблицы, содержащей столбцы: адрес, номер дома, кол-во этажей, площадь дома, наличие сарая, площадь сарая, высота сарая, материал (из которого сделан сарай), наличие бани, площадь бани. Возврат в меню аналогичен первому запросу. При выборе пользователем шестого запроса, на экране должен появляться список всех банков в виде таблицы, содержащей столбцы: адрес, номер дома, территория деятельности, форма собственности, название. Возврат в меню аналогичен первому запросу. При выборе пользователем седьмого запроса, пользователю должно предлагаться ввести название банка, после чего на экране должна появиться таблица, содержащая следующие столбцы: адрес и номер дома. Если банк не найден, то программа должна вывести соответствующее сообщение. Возврат в меню аналогичен первому запросу. При выборе пользователем восьмого запроса, на экране должен появляться список всех магазинов в виде таблицы, содержащей столбцы: адрес, номер дома, название, площадь, специализация. Возврат в меню аналогичен первому запросу. При выборе пользователем девятого запроса, пользователю должно предлагаться ввести название магазина, после чего на экране должна появиться таблица, содержащая следующие столбцы: адрес и номер дома. Если магазин не найден, то программа должна вывести соответствующее сообщение. Возврат в меню аналогичен первому запросу. При выборе пользователем десятого запроса, пользователю должно предлагаться ввести адрес, номер дома, площадь, специализацию и название. После чего на экране должна появляться надпись «магазин успешно добавлен». При этом в таблице Shopsв базе данных должна появляться новая строка с введенными пользователем данными. Возврат в меню аналогичен первому запросу. При выборе пользователем одиннадцатого запроса, пользователю должно предлагаться ввести адрес, номер дома, площадь, территорию деятельности, форму собственности и название. После чего на экране должна появляться надпись «банк успешно добавлен». При этом в таблице Banksв базе данных должна появляться новая строка с введенными пользователем данными. Возврат в меню аналогичен первому запросу. При выборе пользователем двенадцатого запроса, на экране должен появляться список всех складов в виде таблицы, содержащей столбцы: адрес, номер, площадь, и высота. Возврат в меню аналогичен первому запросу. При выборе пользователем тринадцатого запроса, пользователю должно предлагаться ввести кол-во комнат. После чего на экране должен появляться список всех квартир в виде таблицы, содержащей столбцы: адрес, номер квартиры, этаж, площадь квартиры. В случае, если квартир с заданным количеством комнат не найдено, программа должна выдать сообщение «таких квартир не найдено». Возврат в меню аналогичен первому запросу. При выборе пользователем четырнадцатого запроса, пользователю должно предлагаться ввести адрес, номер, площадь и высоту. После чего на экране должна появляться надпись «склад успешно добавлен». При этом в таблице Warehousesв базе данных должна появляться новая строка с введенными пользователем данными. Возврат в меню аналогичен первому запросу. При выборе пользователем пятнадцатого запроса, пользователю должно предлагаться ввести адрес. После чего на экране должен появляться список всех банков, расположенных на этой улице, в виде таблицы, содержащей столбцы: адрес, номер дома, территория деятельности, форма собственности и название. В случае отсутствия банков на заданной улице, программа сообщает о том, что банков не найдено. Возврат в меню аналогичен первому запросу. При выборе пользователем пункта 0, программа завершает работу.

# 4. КЛАССЫ

# 4.1 Класс Program

Класс Program содержит такие глобальные переменные, как multList (список многоэтажных домов), apList (список квартир), privList (список частных домов), gardList (список садовых домов), warList (список складов), shopList (список магазинов), bankList (список банков), connectionString (строка подключения базы данных). Также в классе Program есть методы FillMultList(), FillApList(), FillPrivList(), FillGardList(), FillWarList(), FillShopList(), FillBankList(), которые заполняют соответствующие листы данными из базы данных. Кроме этого, присутствуют методы WriteHouse(), WriteApartment(), WriteAreas(), WritePrivateHouses(), WriteGardenHouses(), WriteBank(), WriteBankAddresses(), WriteShop(), WriteShopAddresses(), WriteWarehouse(), AddShop(), AddBank(), AddWareHouse(), которые выполняют все запросы в меню. Реализуются методы Menu() и Main().

# 4.2 Класс ResidentialFacility

Класс ResidentialFacility описывает жилой объект. В нем содержатся такие поля, как Address (адрес), Number (номер дома) и Floors (количество этажей). Эти характеристики имеет любой жилой объект. Также в классе содержится виртуальный метод GetNecessaryApartment(), который переопределяется в классе-наследнике.

# 4.3 Класс PrivateHouse

Класс PrivateHouse является наследником класса ResidentialFacility, а значит имеет те поля, которые содержит класс ResidentialFacility. Данный класс описывает объект частного дома. В нем содержатся такие поля, как HouseArea (площадь дома),Garage (наличие гаража), GarageArea (площадь гаража), Basement (наличие подвала), BasementArea (площадь подвала), BasementHeight (высота подвала).

# 4.4 Класс GardenHouse

Класс GardenHouse является наследником класса ResidentialFacility, а значит имеет те поля, которые содержит класс ResidentialFacility. Данный класс описывает объект садового дома. В нем содержатся такие поля, как HouseArea (площадь дома), Barn (наличие сарая), BarnArea (площадь сарая), BarnHeight (высота сарая, если сарай есть), BarnBuildingMaterial (материал, из которого постороен сарай), Bathhouse (наличие бани), BathhouseArea (площадь бани).

# 4.5 Класс Multistory

Класс Multistory является наследником класса ResidentialFacility, а значит имеет те поля, которые содержит класс ResidentialFacility. Данный класс описывает объект многоэтажного дома. В нем содержится поле CountApartment (количество квартир).

# 4.6 Класс Apartment

Класс Apartment является наследником класса Multistory, а значит имеет те поля, которые содержит класс Multistory. Данный класс описывает объект квартиры. В нем содержатся такие поля, как Floor (этаж), ApartmentNumber (номер квартиры), ApartmentArea (площадь квартиры), CountRoom (количество комнат), LivingArea (жилая площадь), BathroomArea (площадь санузла), HallwayArea (площадь гостиной). Также в классе переопределяется метод GetNecessaryApartment(), с помощью которого можно найти квартиры с заданным количеством комнат.

# 4.7 Класс NonResidentialFacility

Класс NonResidentialFacility описывает нежилой объект. В нем содержатся такие поля, как Address (адрес), Number (номер дома) и Area (площадь). Эти характеристики имеет любой нежилой объект. Также в классе содержится виртуальный метод GetBankOnStreet(), который переопределяется в классе-наследнике.

# 4.8 Класс Warehouse

Класс Warehouse является наследником класса NonResidentialFacility, а значит имеет те поля, которые содержит класс NonResidentialFacility. Данный класс описывает объект склада. В нем содержится поле Height (высота склада).

# 4.9 Класс Shop

Класс Shop является наследником класса NonResidentialFacility, а значит имеет те поля, которые содержит класс NonResidentialFacility. Данный класс описывает объект магазина. В нем содержатся такие поля, как Specialization (специализация магазина), Name (название магазина).

# 4.10 Класс Bank

Класс Bank является наследником класса NonResidentialFacility, а значит имеет те поля, которые содержит класс NonResidentialFacility. Данный класс описывает объект банка. В нем содержатся такие поля, как Territorial (территория деятельности банка), Property (форма собственности банка), Name (название банка). Также в классе переопределяется метод GetBankOnStreet(), с помощью которого можно найти банки, расположенные на заданной улице.

# 4.11 Иерархия классов

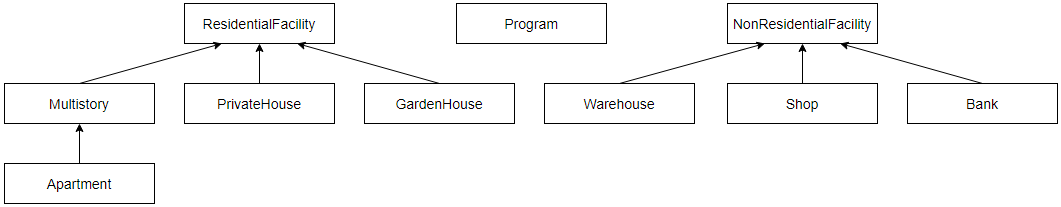


Рис. 1. Иерархия классов

# 3. БАЗА ДАННЫХ

В базе данных находится 7 таблиц, содержащих жилые и нежилые объекты. Данный код создает таблицы в базе данных.

usingSystem;

usingSystem.Collections.Generic;

using System.Data.Entity;

using System.Text;

namespace PavelKurs

{

publicclassMyDbContext : DbContext

{

publicMyDbContext() : base(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\79068\OneDrive\Документы\kurs.mdf;Integrated Security=True;Connect Timeout=30") { }

public DbSet<Multistory> multystories { get; set; }

public DbSet<Apartment> apartments { get; set; }

public DbSet<PrivateHouse> PrivateHouses { get; set; }

public DbSet<GardenHouse> gardenHouses { get; set; }

public DbSet<Warehouse> warehouses { get; set; }

public DbSet<Shop> shops { get; set; }

public DbSet<Bank> banks { get; set; }

}

}

На рис. 2 представлена таблица многоэтажных домов:

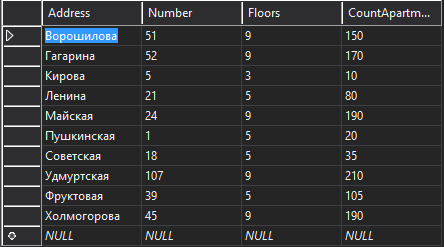


Рис. 2. Таблица многоэтажных домов

На рис. 3 представлена таблица квартир:

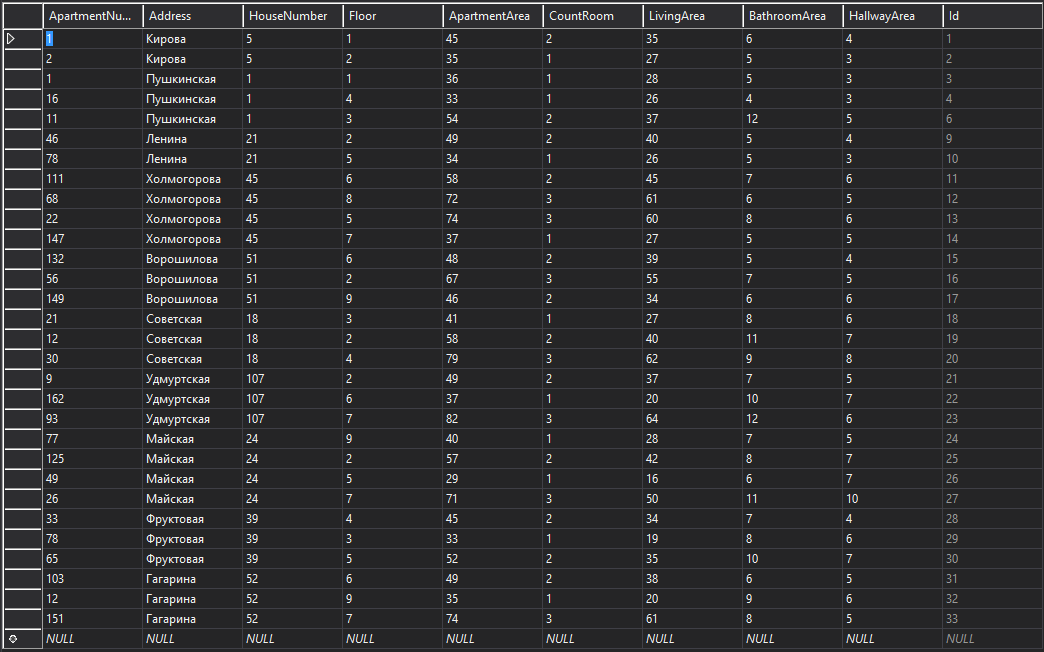


Рис. 3. Таблица квартир

На рис. 4 представлена таблица частных домов:

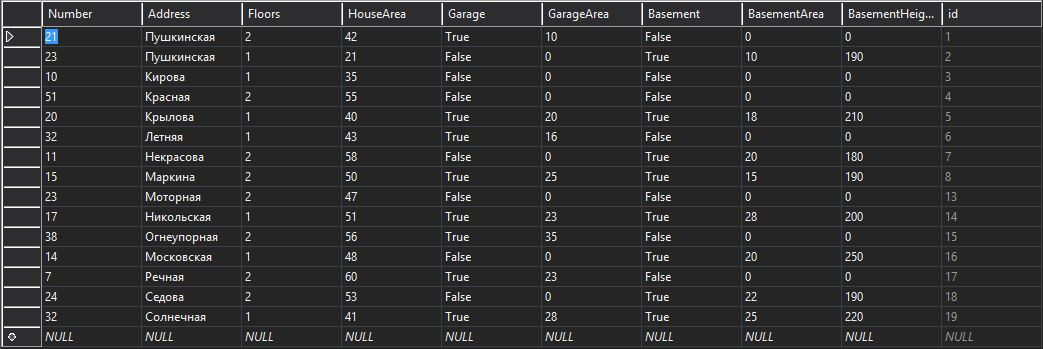


Рис. 4. Таблица частных домов

В столбце Garage (гараж)можно увидеть слова "False", это означает, что у данного частного дома нет гаража и соответственно в столбце GarageArea(площадь гаража) стоит число 0. Точно так же происходит в столбце Basement (подвал) - "False"означает отсутствие у данного частного дома подвала и в столбцах BasementArea (площадь подвала)иBasementHeight(высота подвала) стоит число 0.

На рис. 5 представлена таблица садовых домов:

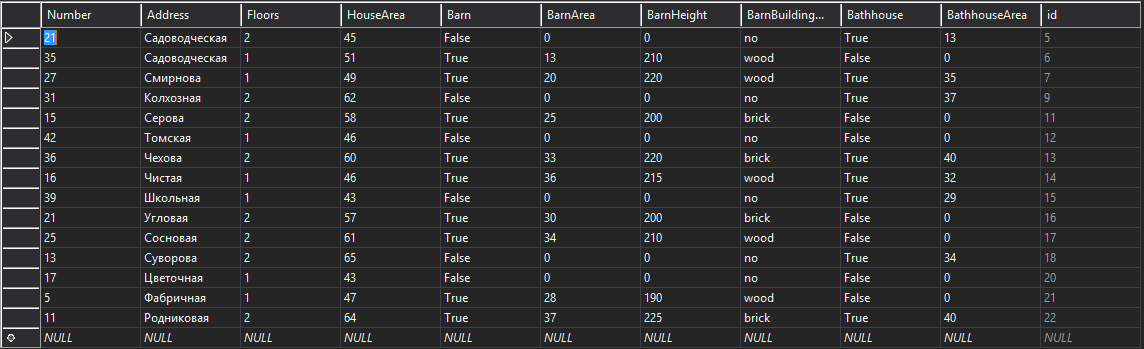


Рис. 5. Таблица садовых домов

В столбце Barn (сарай)можно увидеть слова "False", это означает, что у данного садового дома нет сарая и в столбцах BarnArea (площадь сарая) и BarnHeight (высота сарая)стоит число 0, а в столбце BarnBuildingMaterial (материал, из которого построен сарай) написано "no". Так же, если в столбце Bathhouse (баня)написано "False", это означает, что у данного садового дома нет бани и в столбце BathhouseArea (площадь бани) стоит число 0.

На рис. 6 представлена таблица магазинов:

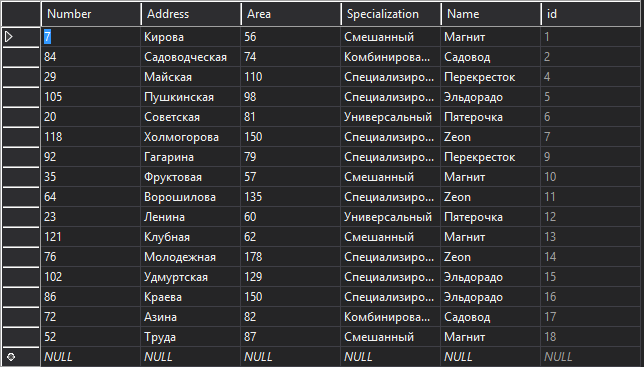


Рис. 6. Таблица магазинов

На рис. 7 представлена таблица банков:

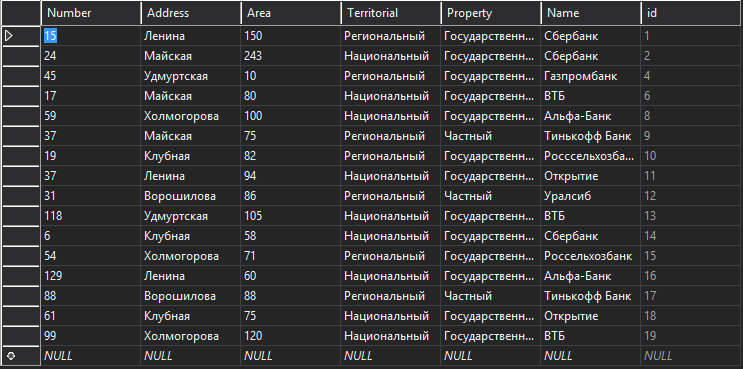


Рис. 7. Таблица банков

На рис. 8 представлена таблица складов:

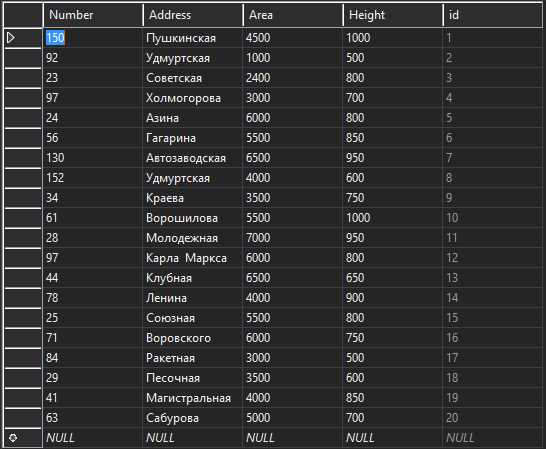


Рис. 8. Таблица складов

# 4. КОНТРОЛЬНЫЙ ПРИМЕР

На рис. 9 представлено главное меню программы:

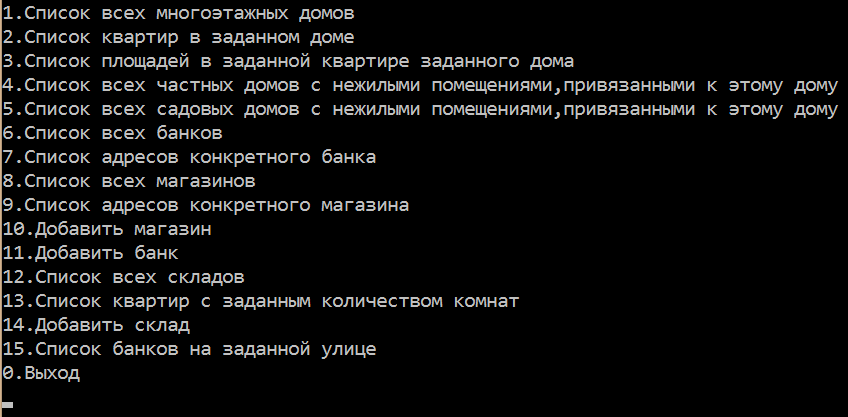


Рис. 9. Главное меню

На рис. 10 представлено выполнение первого запроса:

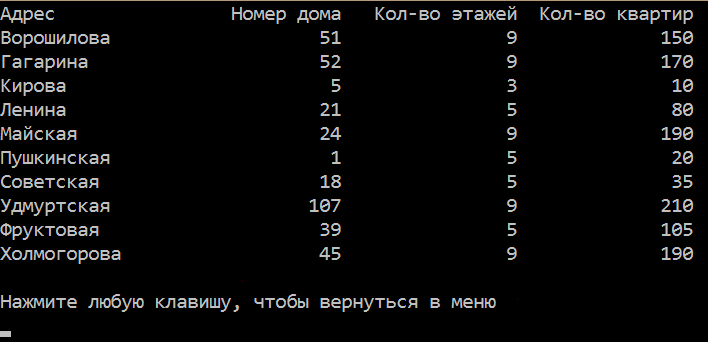


Рис. 10. Первый запрос

Результаты второго запроса можно посмотреть на рис. 11-13:

2 запрос.png

Рис. 11

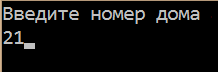


Рис. 12

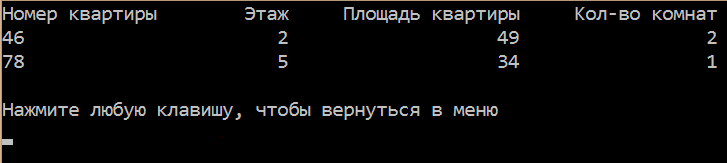


Рис. 13

В случае, если адрес введен неправильно или такого дома нет в базе данных, программа сообщает о том, что дом не найден:

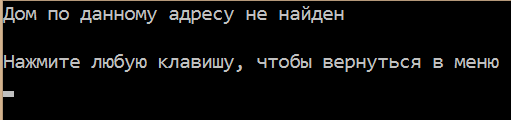


Рис. 14. Дом не найден

Результаты третьего запроса можно посмотреть на рис. 15-18:

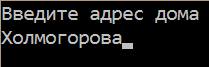


Рис. 15

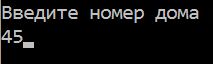


Рис. 16

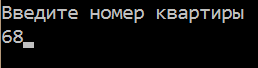


Рис.17

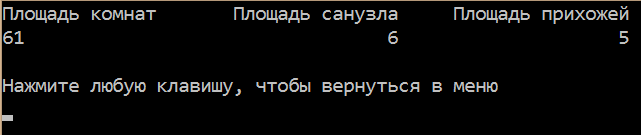


Рис. 18

В случае, если адрес или номер квартиры введен неправильно или такого дома или квартиры нет в базе данных, программа сообщает о том, что квартира не найдена:

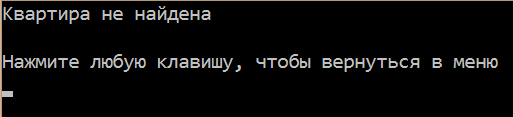


Рис. 19. Квартира не найдена

На рис. 20 представлено выполнение четвертого запроса:



Рис. 20. Четвертый запрос

На рис. 21 представлено выполнение пятого запроса:



Рис. 21. Пятый запрос

На рис. 22 представлено выполнение шестого запроса:

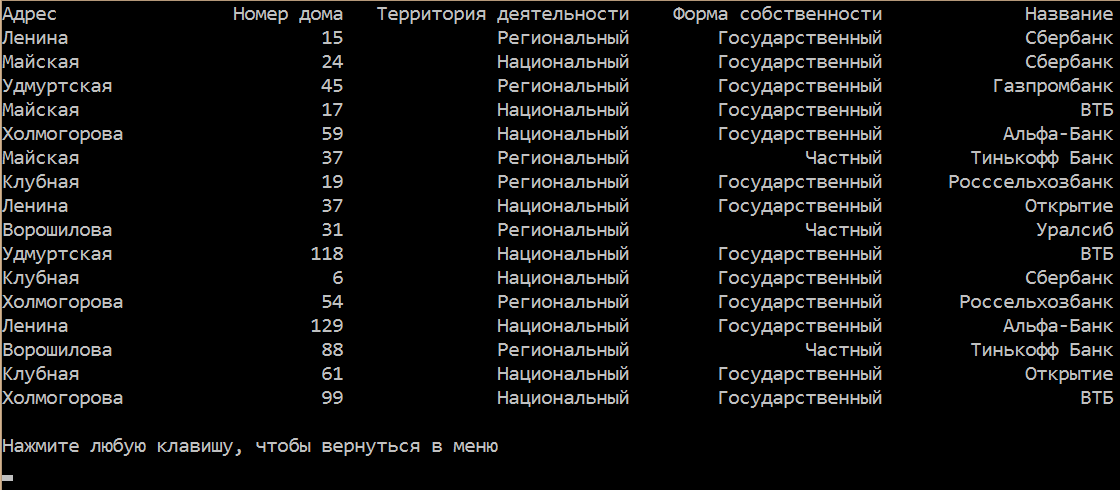


Рис. 22. Шестой запрос

Результаты седьмого запроса можно посмотреть на рис. 23-24:

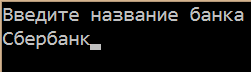


Рис. 23

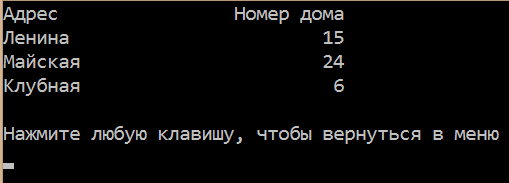


Рис. 24

В случае, банков с таким названием не найдено или название введено неправильно, программа сообщает о том, что банки не найдены:

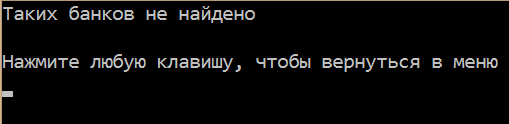


Рис. 25. Банки не найдены

На рис. 26 представлено выполнение восьмого запроса:

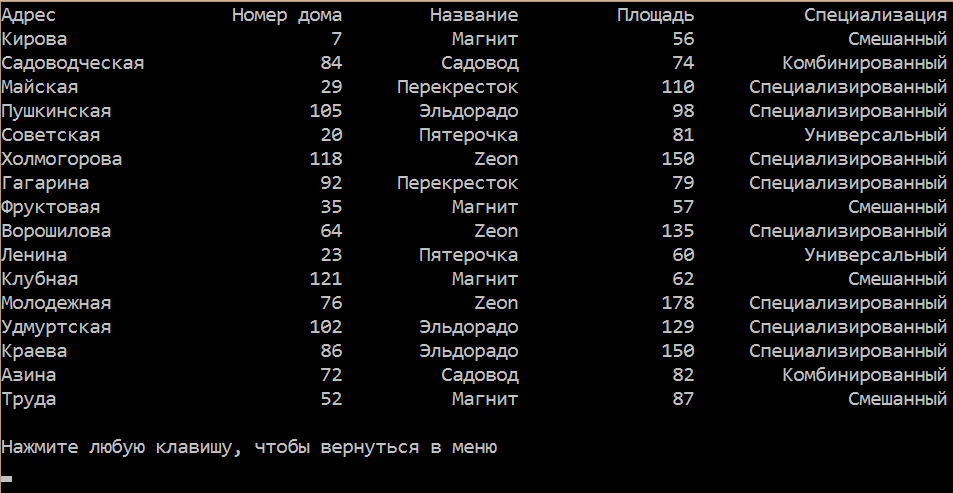


Рис. 26. Восьмой запрос

Результаты девятого запроса можно посмотреть на рис. 27-29:

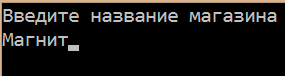


Рис. 27

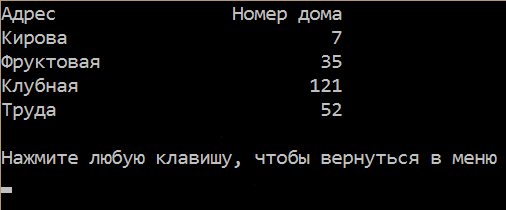


Рис. 28

В случае, магазинов с таким названием не найдено или название введено неправильно, программа сообщает о том, что магазины не найдены:

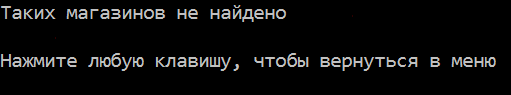


Рис. 29. Магазины не найдены

Результаты десятого запроса можно посмотреть на рис. 30-36:

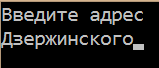


Рис. 30

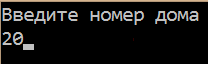


Рис. 31

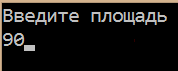


Рис. 32

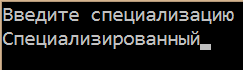


Рис. 33

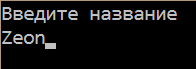


Рис. 34

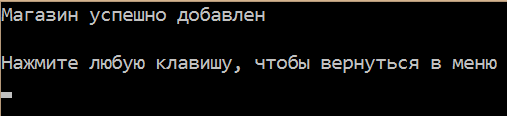


Рис. 35

На рис. 36 таблица с добавленным магазином:

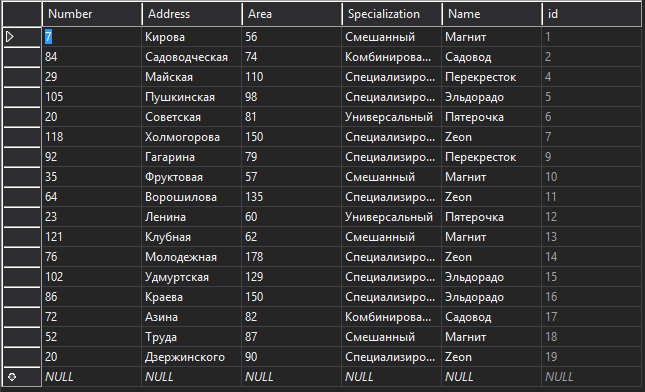


Рис. 36. Добавлен магазин

Результаты одиннадцатого запроса можно посмотреть на рис. 37-44:

11 запрос адрес.png

Рис. 37

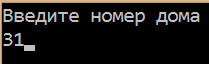


Рис. 38

11 запрос площадь.png

Рис. 39

11 запрос терр.png

Рис. 40

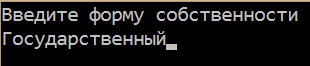


Рис. 41

11 запрос название.png

Рис. 42

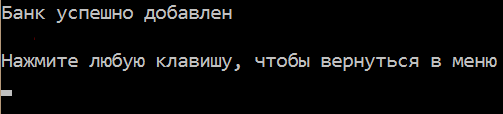


Рис. 43

На рис. 44 таблица с добавленным банком:

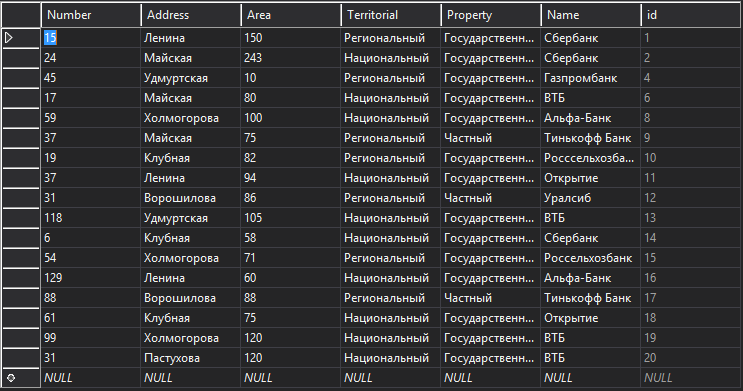


Рис. 44. Добавлен банк

На рис. 45 представлено выполнение двенадцатого запроса:

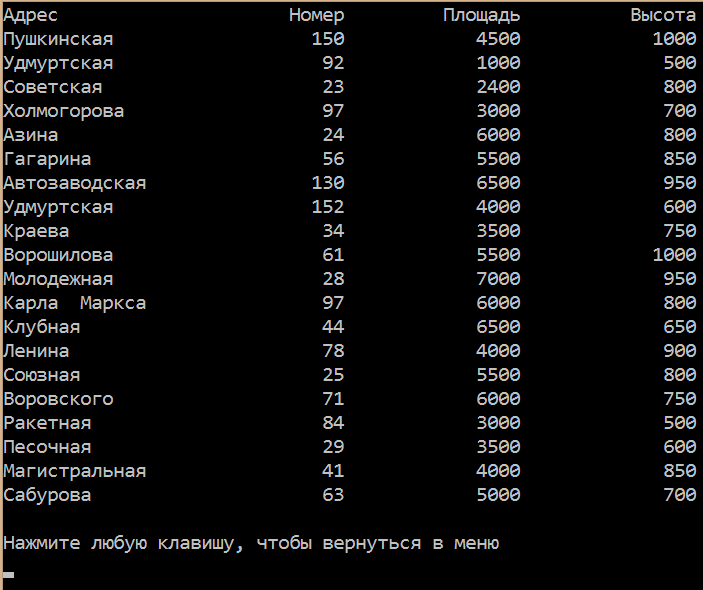


Рис. 45. Двенадцатый запрос

Результаты тринадцатого запроса можно посмотреть на рис. 46-48:

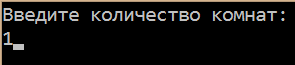


Рис. 46

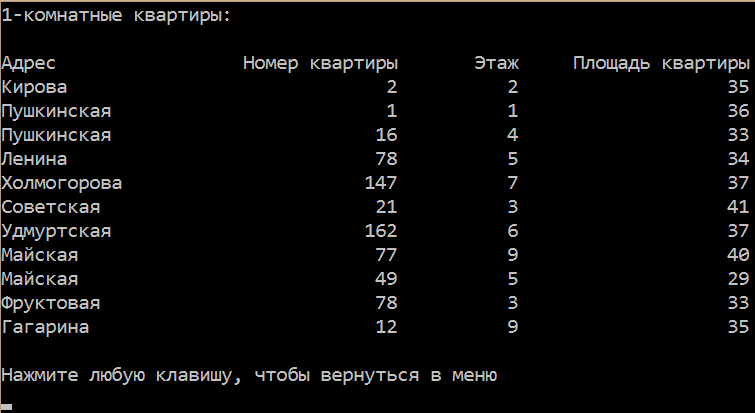


Рис. 47

В случае, если квартир с заданным количеством комнат не найдено, программа сообщает об этом:

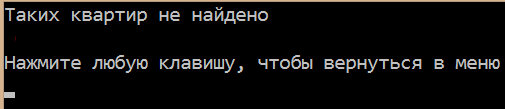


Рис. 48. Квартир не найдено

Результаты четырнадцатого запроса можно посмотреть на рис. 49-54:

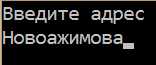


Рис. 49

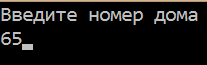


Рис. 50

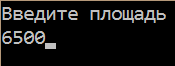


Рис. 51

14 запрос высота.png

Рис. 52

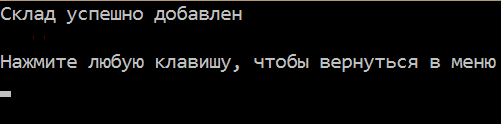


Рис. 53

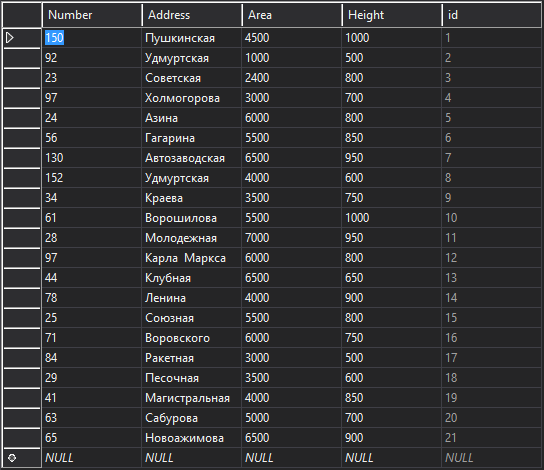


Рис. 54. Добавлен склад

Результаты пятнадцатого запроса можно посмотреть на рис. 55-57:

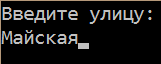


Рис. 55

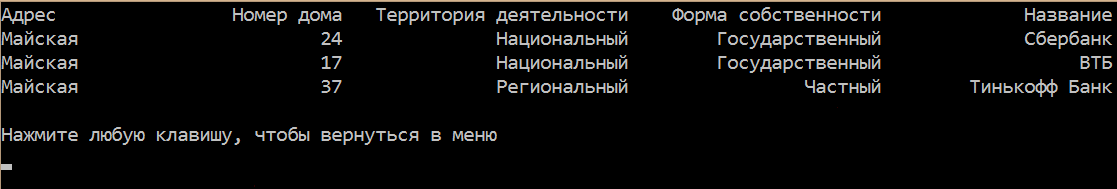


Рис. 56

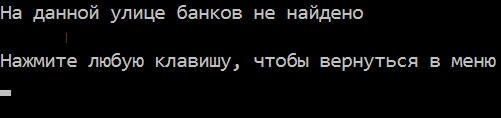


Рис. 57. Банков не найдено

# ЗАКЛЮЧЕНИЕ

Программа успешно реализована и работает в соответствии с поставленными задачами. В ней реализованы такие функции, как список всех домов, список квартир в заданном доме, список площадей в заданной квартире заданного дома, список всех частных домов с нежилыми помещениями, привязанными к этому дому, список всех садовых домов с нежилыми помещениями, привязанными к этому дому, список всех банков, список адресов конкретного банка, список всех магазинов, список адресов конкретного магазина, добавить магазин, добавить банк, список всех складов, список квартир с заданным количеством комнат, добавить склад, список банков на заданной улице. Для полноценной работы с программой был создан дружественный интерфейс, позволяющий пользователю быстро выполнять свои запросы. В программе реализованы такие принципы ООП, как инкапсуляция (были созданы классы на основе сущностей), наследование (например, класс Apartment является дочерним по отношению к классу Multistory, а класс Multistory, в свою очередь, является дочерним по отношению к классу ResidentialFacility) и полиморфизм (в базовых классах созданы виртуальные методы GetNecessaryApartment() и GetBankOnStreet() и переопределены в дочерних классах).

# ПРИЛОЖЕНИЕ

Листинг программы

usingSystem;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data;

using System.ComponentModel.DataAnnotations;

using System.Data.Entity;

using System.Data.SqlClient;

using System.IO;

using System.Xml.Serialization;

using System.Linq;

namespace PavelKurs

{

public class ResidentialFacility //жилой

{

public string Address;

public int Number;

public int Floors;

public virtual void GetNecessaryApartment(List<Apartment> apartmentList) { }

}

public class NonResidentialFacility //нежилой

{

public string Address;

public int Number;

public double Area;

public virtual void GetBankOnStreet(List<Bank> bankList) { }

}

public class Multistory : ResidentialFacility //многоэтажный

{

public int CountApartment;

}

public class Apartment : Multistory

{

public int Floor;

public int ApartmentNumber;

public double ApartmentArea;

public int CountRoom;

public double LivingArea;

public double BathroomArea;

public double HallwayArea;

public override void GetNecessaryApartment(List<Apartment> apartmentList)

{

Console.Clear();

Console.WriteLine("Введитеколичествокомнат:");

var count = Convert.ToInt32(Console.ReadLine());

var countFoundApartment = 0;

Console.Clear();

Console.WriteLine(count + "-комнатныеквартиры:\n");

Console.WriteLine("{0,-20} {1,15} {2,10} {3,20}", "Адрес", "Номерквартиры", "Этаж", "Площадьквартиры");

foreach (var i in apartmentList)

{

if (i.CountRoom == count)

{

countFoundApartment++;

Console.WriteLine("{0,-20} {1,15} {2,10} {3,20}", i.Address, i.ApartmentNumber, i.Floor, i.ApartmentArea);

}

}

if (countFoundApartment == 0)

{

Console.Clear();

Console.WriteLine("Такихквартирненайдено");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

}

}

public class PrivateHouse : ResidentialFacility //частныйдом

{

public double HouseArea;

public bool Garage;

public double GarageArea;

public bool Basement;

public double BasementArea;

public double BasementHeight;

}

public class GardenHouse : ResidentialFacility //садовыйдом

{

public double HouseArea;

public bool Barn;

public double BarnArea;

public double BarnHeight;

public string BarnBuildingMaterial;

public bool Bathhouse;

public double BathhouseArea;

}

public class Warehouse : NonResidentialFacility

{

public int Height;

}

public class Shop : NonResidentialFacility

{

public string Specialization;

public string Name;

}

public class Bank : NonResidentialFacility

{

public string Territorial; //(региональный, национальныйитд)

public string Property; //(государственный или частный)

public string Name;

public override void GetBankOnStreet(List<Bank> bankList)

{

Console.Clear();

Console.WriteLine("Введитеулицу:");

var street = Console.ReadLine();

var countFoundBank = 0;

Console.Clear();

Console.WriteLine("{0,-20} {1,10} {2,25} {3,22} {4,20}", "Адрес", "Номер дома", "Территория деятельности", "Форма собственности", "Название");

foreach (var i in bankList)

{

if (i.Address == street)

{

countFoundBank++;

Console.WriteLine("{0,-20} {1,10} {2,25} {3,22} {4,20}", i.Address, i.Number, i.Territorial, i.Property, i.Name);

}

}

if (countFoundBank == 0)

{

Console.Clear();

Console.WriteLine("Наданнойулицебанковненайдено");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

}

}

class Program

{

public static List<Multistory> multList = new List<Multistory>();

public static List<Apartment> apList = new List<Apartment>();

public static List<PrivateHouse> privList = new List<PrivateHouse>();

public static List<GardenHouse> gardList = new List<GardenHouse>();

public static List<Warehouse> warList = new List<Warehouse>();

public static List<Shop> shopList = new List<Shop>();

public static List<Bank> bankList = new List<Bank>();

public static string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\777\Downloads\kurs.mdf;Integrated Security=True;Connect Timeout=30";

public static void FillMultList()

{

multList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM Multistories";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var mlt = new Multistory

{

Floors = read.GetInt32(2),

Number = read.GetInt32(1),

Address = read.GetString(0),

CountApartment = read.GetInt32(3)

};

multList.Add(mlt);

}

}

}

public static void FillApList()

{

apList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM Apartments";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var ap = new Apartment

{

Number = read.GetInt32(2),

Address = read.GetString(1),

Floor = read.GetInt32(3),

ApartmentArea = read.GetDouble(4),

CountRoom = read.GetInt32(5),

LivingArea = read.GetDouble(6),

BathroomArea = read.GetDouble(7),

HallwayArea = read.GetDouble(8),

ApartmentNumber = read.GetInt32(0)

};

apList.Add(ap);

}

}

}

public static void FillPrivList()

{

privList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM PrivateHouses";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var ph = new PrivateHouse

{

Address = read.GetString(1),

Number = read.GetInt32(0),

Floors = read.GetInt32(2),

HouseArea = read.GetDouble(3),

Garage = read.GetBoolean(4),

GarageArea = read.GetDouble(5),

Basement = read.GetBoolean(6),

BasementArea = read.GetDouble(7),

BasementHeight = read.GetDouble(8)

};

privList.Add(ph);

}

}

}

public static void FillGardList()

{

gardList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM GardenHouses";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var gh = new GardenHouse

{

Number = read.GetInt32(0),

Address = read.GetString(1),

Floors = read.GetInt32(2),

HouseArea = read.GetDouble(3),

Barn = read.GetBoolean(4),

BarnArea = read.GetDouble(5),

BarnHeight = read.GetDouble(6),

BarnBuildingMaterial = read.GetString(7),

Bathhouse = read.GetBoolean(8),

BathhouseArea = read.GetDouble(9)

};

gardList.Add(gh);

}

}

}

public static void FillWarList()

{

warList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM Warehouses";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var wh = new Warehouse

{

Number = read.GetInt32(0),

Address = read.GetString(1),

Area = read.GetDouble(2),

Height = read.GetInt32(3)

};

warList.Add(wh);

}

}

}

public static void FillShopList()

{

shopList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM Shops";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var sh = new Shop

{

Number = read.GetInt32(0),

Address = read.GetString(1),

Area = read.GetDouble(2),

Specialization = read.GetString(3),

Name = read.GetString(4)

};

shopList.Add(sh);

}

}

}

public static void FillBankList()

{

bankList.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string sql = "SELECT \* FROM Banks";

SqlCommand com = new SqlCommand(sql, connection);

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

var bk = new Bank

{

Number = read.GetInt32(0),

Address = read.GetString(1),

Area = read.GetDouble(2),

Territorial = read.GetString(3),

Property = read.GetString(4),

Name = read.GetString(5)

};

bankList.Add(bk);

}

}

}

public static void Menu()

{

Console.WriteLine("1.Списоквсехдомов");

Console.WriteLine("2.Список квартир в заданном доме");

Console.WriteLine("3.Список площадей в заданной квартире заданного дома");

Console.WriteLine("4.Список всех частных домов с нежилыми помещениями,привязанными к этому дому");

Console.WriteLine("5.Список всех садовых домов с нежилыми помещениями,привязанными к этому дому");

Console.WriteLine("6.Список всех банков");

Console.WriteLine("7.Список адресов конкретного банка");

Console.WriteLine("8.Список всех магазинов");

Console.WriteLine("9.Список адресов конкретного магазина");

Console.WriteLine("10.Добавить магазин");

Console.WriteLine("11.Добавить банк");

Console.WriteLine("12.Список всех складов");

Console.WriteLine("13.Список квартир с заданным количеством комнат");

Console.WriteLine("14.Добавить склад");

Console.WriteLine("15.Список банков на заданной улице");

Console.WriteLine("0.Выход");

var key = Convert.ToInt32(Console.ReadLine());

switch (key)

{

case 1:

{

WriteHouse();

}

break;

case 2:

{

WriteApartment();

}

break;

case 3:

{

WriteAreas();

}; break;

case 4:

{

WritePrivateHouses();

}; break;

case 5:

{

WriteGardenHouses();

}; break;

case 6:

{

WriteBank();

}; break;

case 7:

{

WriteBankAddresses();

}; break;

case 8:

{

WriteShop();

}; break;

case 9:

{

WriteShopAddresses();

}; break;

case 10:

{

AddShop();

}; break;

case 11:

{

AddBank();

}; break;

case 12:

{

WriteWarehouse();

}; break;

case 13:

{

var apartment = new Apartment();

apartment.GetNecessaryApartment(apList);

Menu();

}; break;

case 14:

{

AddWareHouse();

}; break;

case 15:

{

var bank = new Bank();

bank.GetBankOnStreet(bankList);

Menu();

}; break;

}

}

public static void WriteHouse()

{

Console.Clear();

Console.WriteLine("{0,-20} {1,10} {2,15} {3,15}", "Адрес", "Номер дома", "Кол-во этажей", "Кол-во квартир");

foreach (var i in multList)

Console.WriteLine("{0,-20} {1,10} {2,15} {3,15}", i.Address, i.Number, i.Floors, i.CountApartment);

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteApartment()

{

Console.Clear();

Console.WriteLine("Введитеадресдома");

string adres = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите номер дома");

int num = Convert.ToInt32(Console.ReadLine());

var countAraptment = 0;

Console.Clear();

Console.WriteLine("{0,-20} {1,5} {2,20} {3,17}", "Номерквартиры", "Этаж", "Площадьквартиры", "Кол-вокомнат");

foreach (var i in apList)

{

if (i.Address == adres && i.Number == num)

{

countAraptment++;

Console.WriteLine("{0,-20} {1,5} {2,20} {3,17}", i.ApartmentNumber, i.Floor, i.ApartmentArea, i.CountRoom);

}

}

if (countAraptment == 0)

{

Console.Clear();

Console.WriteLine("Домподанномуадресуненайден");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteAreas()

{

Console.Clear();

Console.WriteLine("Введитеадресдома");

string adres = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите номер дома");

int num = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введитеномерквартиры");

int apNum = Convert.ToInt32(Console.ReadLine());

Console.Clear();

var found = false;

Console.WriteLine("{0,-20} {1,15} {2,20}", "Площадькомнат", "Площадьсанузла", "Площадьприхожей");

foreach (var i in apList)

{

if (i.Address == adres && i.Number == num && i.ApartmentNumber == apNum)

{

found = true;

Console.WriteLine("{0,-20} {1,15} {2,20} ", i.LivingArea, i.BathroomArea, i.HallwayArea);

}

}

if (!found)

{

Console.Clear();

Console.WriteLine("Квартираненайдена");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WritePrivateHouses()

{

Console.Clear();

Console.WriteLine("{0,-15}{1,10}{2,16}{3,15}{4,17}{5,17}{6,18}{7,18}{8,17}", "Адрес", "Номердома",

"кол-во этажей", "площадь дома", "наличие гаража", "площадь гаража",

"наличие подвала", "площадь подвала", "высота подвала");

foreach (var i in privList)

{

string gar;

string bas;

if (i.Garage == true) gar = "гаражесть";

else gar = "гаражанет";

if (i.Basement == true) bas = "подвалесть";

else bas = "подваланет";

Console.WriteLine("{0,-15}{1,10}{2,16}{3,15}{4,17}{5,17}{6,18}{7,18}{8,17}",

i.Address, i.Number, i.Floors, i.HouseArea, gar, i.GarageArea, bas, i.BasementArea, i.BasementHeight);

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteGardenHouses()

{

Console.Clear();

Console.WriteLine("{0,-20}{1,10}{2,15}{3,15}{4,15}{5,15}{6,15}{7,10}{8,15}{9,15}", "Адрес", "Номердома",

"Кол-во этажей", "Площадь дома", "Наличие сарая", "Площадь сарая", "Высота сарая", "Материал",

"Наличиебани", "Площадьбани");

foreach (var i in gardList)

{

string gar;

string bas;

string mat = null;

if (i.Barn == true) { gar = "сарайесть"; mat = i.BarnBuildingMaterial; }

else { gar = "сараянет"; mat = ""; }

if (i.Bathhouse == true) bas = "баняесть";

else bas = "банинет";

Console.WriteLine("{0,-20}{1,10}{2,15}{3,15}{4,15}{5,15}{6,15}{7,10}{8,15}{9,15}",

i.Address, i.Number, i.Floors, i.HouseArea, gar, i.BarnArea, i.BarnHeight, mat, bas, i.BathhouseArea);

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteBank()

{

Console.Clear();

Console.WriteLine("{0,-20} {1,10} {2,25} {3,22} {4,20}", "Адрес", "Номер дома", "Территория деятельности", "Форма собственности", "Название");

foreach (var i in bankList)

Console.WriteLine("{0,-20} {1,10} {2,25} {3,22} {4,20}", i.Address, i.Number, i.Territorial, i.Property, i.Name);

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteBankAddresses()

{

Console.Clear();

Console.WriteLine("Введите название банка");

string b = Console.ReadLine();

Console.Clear();

var countFoundBank = 0;

Console.WriteLine("{0,-20} {1,10}", "Адрес", "Номердома");

foreach (var i in bankList)

{

if (i.Name == b)

{

countFoundBank++;

Console.WriteLine("{0,-20} {1,10} ", i.Address, i.Number);

}

}

if (countFoundBank == 0)

{

Console.Clear();

Console.WriteLine("Такихбанковненайдено");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteShop()

{

Console.Clear();

Console.WriteLine("{0,-20} {1,10} {2,15} {3,15} {4,22}", "Адрес", "Номер дома", "Название", "Площадь", "Специализация");

foreach (var i in shopList)

Console.WriteLine("{0,-20} {1,10} {2,15} {3,15} {4,22}", i.Address, i.Number, i.Name, i.Area, i.Specialization);

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteShopAddresses()

{

Console.Clear();

Console.WriteLine("Введите название магазина");

string ad = Console.ReadLine();

Console.Clear();

var countFoundShop = 0;

Console.WriteLine("{0,-20} {1,10} ", "Адрес", "Номердома");

foreach (var i in shopList)

{

if (i.Name == ad)

{

countFoundShop++;

Console.WriteLine("{0,-20} {1,10}", i.Address, i.Number);

}

}

if (countFoundShop == 0)

{

Console.Clear();

Console.WriteLine("Такихмагазиновненайдено");

}

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void WriteWarehouse()

{

Console.Clear();

Console.WriteLine("{0,-20} {1,10} {2,15} {3,15}", "Адрес", "Номер", "Площадь", "Высота");

foreach (var i in warList)

Console.WriteLine("{0,-20} {1,10} {2,15} {3,15}", i.Address, i.Number, i.Area, i.Height);

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

Menu();

}

public static void AddShop()

{

string sql = "SELECT \* FROM Shops";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

var adapter = new SqlDataAdapter(sql, connection);

var ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

DataRow newRow = dt.NewRow();

Console.Clear();

Console.WriteLine("Введитеадрес");

newRow["Address"] = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите номер дома");

newRow["Number"] = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введитеплощадь");

newRow["Area"] = Convert.ToDouble(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введитеспециализацию");

newRow["Specialization"] = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введитеназвание");

newRow["Name"] = Console.ReadLine();

Console.Clear();

dt.Rows.Add(newRow);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(dt);

ds.Clear();

adapter.Fill(ds);

Console.WriteLine("Магазинуспешнодобавлен");

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

FillShopList();

Menu();

}

}

public static void AddBank()

{

string sql = "SELECT \* FROM Banks";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

var adapter = new SqlDataAdapter(sql, connection);

var ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

DataRow newRow = dt.NewRow();

Console.Clear();

Console.WriteLine("Введитеадрес");

newRow["Address"] = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите номер дома");

newRow["Number"] = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введитеплощадь");

newRow["Area"] = Convert.ToDouble(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введите Территорию деятельности");

newRow["Territorial"] = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите форму собственности");

newRow["Property"] = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введитеназвание");

newRow["Name"] = Console.ReadLine();

Console.Clear();

dt.Rows.Add(newRow);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(dt);

ds.Clear();

adapter.Fill(ds);

Console.WriteLine("Банкуспешнодобавлен");

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

FillBankList();

Menu();

}

}

public static void AddWareHouse()

{

string sql = "SELECT \* FROM Warehouses";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

var adapter = new SqlDataAdapter(sql, connection);

var ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

DataRow newRow = dt.NewRow();

Console.Clear();

Console.WriteLine("Введитеадрес");

newRow["Address"] = Console.ReadLine();

Console.Clear();

Console.WriteLine("Введите номер дома");

newRow["Number"] = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введитеплощадь");

newRow["Area"] = Convert.ToDouble(Console.ReadLine());

Console.Clear();

Console.WriteLine("Введитевысоту");

newRow["Height"] = Console.ReadLine();

Console.Clear();

dt.Rows.Add(newRow);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(dt);

ds.Clear();

adapter.Fill(ds);

Console.WriteLine("Складуспешнодобавлен");

Console.WriteLine("\nНажмите любую клавишу, чтобы вернуться в меню");

Console.ReadKey();

Console.Clear();

FillWarList();

Menu();

}

}

static void Main(string[] args)

{

FillMultList();

FillApList();

FillBankList();

FillGardList();

FillPrivList();

FillShopList();

FillWarList();

Menu();

}

}

}