МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Выполнила
Радиончик С.С.,
студентка группы ПО-5

Проверил
Крощенко А.А.,
ст. преп. Кафедры ИИТ,
«__» _____ 2021 г.

Брест, 2021

Цель работы: научиться создавать и использовать классы в программах на языке программирования C#.

Вариант 6.

Задание 1. Реализовать простой класс.
***Требования к выполнению*:**
• Реализовать пользовательский класс по варианту:
6*) **Множество вещественных чисел ограниченной мощности.** Предусмотреть возможность объединения двух множеств, вывода на печать элементов множества, а также метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволить создавать объекты с начальной инициализацией. Мощность множества задается при создании объекта. Реализацию множества осуществить на базе одномерного массива. Реализовать метод equals, выполняющий сравнение объектов данного типа.*
• Создать другой класс с методом main, в котором будут находится примеры использования пользовательского класса.
Для каждого класса:
• Создать поля классов
• Создать методы классов
• Добавьте необходимые get и set методы (по необходимости)
• Укажите соответствующие модификаторы видимости
• Добавьте конструкторы
• Переопределить методы toString() и equals().

Задание 2. Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных.
***Требования к выполнению*:**
• Данные для программы загружаются из файла (формат произволен). Файл создать и написать вручную.
6*) **Автоматизированная система аренды квартир.***
Составить программу, которая содержит информацию о квартирах, содержащихся в базе данных бюро обмена квартир. Сведения о каждой квартире (Room) содержат:
• количество комнат;
• общую площадь;
• этаж;
• адрес;
• цену аренды.
• сдается ли квартира.
Программа должна обеспечить:
• Формирование списков свободных занятых квартир;

• Поиск подходящего варианта (при равенстве количества комнат и этажа и различии площадей в пределах 10 кв. м.);
• Удаление квартиры из списка свободных квартир и перемещение в список сдаваемых квартир;
• Вывод полного списка.
• Список квартир, имеющих заданное число комнат;
• Список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;
• Список квартир, имеющих площадь, превосходящую заданную.

Выполнение:
Код программы.
1)
Set.cs

```
using System;
using System.Linq;

namespace Lab3_1
{
	class Set
	{
		public double[] set;

		public Set(int N)
		{
			this.set = new double[N];
		}

		public Set(double[] set)
		{
			this.set = new double[set.Length];
			this.set = set;
		}

		public Set CombineSet(Set addedSet)
		{
			Set combineSet = new Set(this.set);

			for(int i = 0; i < addedSet.set.Length; i++)
			{
				if (!combineSet.set.Any(x => x == addedSet.set[i]))
					combineSet.AddElement(addedSet.set[i]);
			}

			combineSet.set = Sort(combineSet.set);

			return combineSet;
		}

		public void Print()
		{
			Console.Write("Set: ");
			for(int i = 0; i < this.set.Length; i++)
			{
				Console.Write($"{this.set[i]}");
				if (i != this.set.Length - 1)
					Console.Write(", ");
			}
```

```
        Console.WriteLine();
    }

    public void AddElement(double element)
    {
        if(!this.IsSetElement(element))
        {
            Set changedSet = new Set(this.set.Length + 1);

            for(int i = 0; i < this.set.Length; i++)
            {
                changedSet.set[i] = this.set[i];
            }

            changedSet.set[this.set.Length] = element;

            this.set = Sort(changedSet.set);
        }
    }

    public void DeleteElement(double element)
    {
        if (this.IsSetElement(element))
        {
            Set changedSet = new Set(this.set.Length - 1);

            for (int i = 0, j = 0; i < this.set.Length; i++, j++)
            {
                if (this.set[i] == element)
                {
                    j--;
                }
                else
                    changedSet.set[j] = this.set[i];
            }

            this.set = changedSet.set;
        }
        else
            Console.WriteLine("Нет такого элемента");
    }

    public bool Equals(Set comparableSet)
    {
        if (this.set.Length != comparableSet.set.Length)
            return false;

        for (int i = 0; i < this.set.Length; i++)
            if (this.set[i] != comparableSet.set[i])
                return false;

        return true;
    }

    public override string ToString()
    {
        string result = "";

        foreach (double x in this.set)
            result += $"{x.ToString()} ";

        return result;
    }
```

```
        private bool IsSetElement(double element)
        {
            if (this.set.Any(x => x == element))
                return true;
            return false;
        }

        private double[] Sort(double[] set)
        {
            double temp;

            for(int i = 0; i < set.Length - 1; i++)
            {
                for (int j = i + 1; j < set.Length; j++)
                {
                    if(set[i] > set[j])
                    {
                        temp = set[i];
                        set[i] = set[j];
                        set[j] = temp;
                    }
                }
            }

            return set;
        }
    }
}
```

Program.cs

```
using System;

namespace Lab3_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Set set1 = new Set(new double[] { -2, 3.5, 4, 7 });
            Set set2 = new Set(set1.set);
            set1.Print();
            set2.Print();

            set1.AddElement(10);
            set1.Print();
            set2.DeleteElement(3.5);
            set2.Print();

            Console.WriteLine(set1.Equals(set2));

            Set set3 = set1.CombineSet(set2);
            set3.Print();

            Console.Write(set1.ToString());
        }
    }
}
```

2)

Room.cs

```
using System;

namespace Lab3_2
{
	class Room
	{
		public int Id { get; set; }
		public int RoomCount { get; set; }
		public int Area { get; set; }
		public int Floor { get; set; }
		public string Address { get; set; }
		public double RentalPrice { get; set; }
		public bool IsRent { get; set; }

		public void Print()
		{
			Console.WriteLine($"Id: {this.Id}");
			Console.WriteLine($"Number of rooms: {this.RoomCount}");
			Console.WriteLine($"Area: {this.Area}");
			Console.WriteLine($"Floor: {this.Floor}");
			Console.WriteLine($"Price: {this.RentalPrice}");
			Console.WriteLine($"Is for rent: {this.IsRent}\n");
		}
	}
}
```

DataBase.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Lab3_2
{
	class DataBase
	{
		public List<Room> rooms = new List<Room>();

		public DataBase FreeRoomsList()
		{
			DataBase freeRoom = new DataBase();

			foreach(var room in rooms)
			{
				if(room.IsRent)
					freeRoom.rooms.Add(room);
			}

			return freeRoom;
		}

		public DataBase OccupiedRoomsList()
		{
			DataBase occupiedRoom = new DataBase();

			foreach(var room in rooms)
			{
				if(!room.IsRent)
					occupiedRoom.rooms.Add(room);
			}

			return occupiedRoom;
```

```
        }

        public DataBase SuitableOption(int roomCount, int floor, double area)
        {
                DataBase suitableRoom = new DataBase();

                foreach (var room in rooms)
                {
                        if(room.RoomCount == roomCount && room.Floor == floor)
                        {
                                double areaDifference =  room.Area - area > 0 ? room.Area - area
: Math.Abs(room.Area - area);

                                if (areaDifference <= 10)
                                        suitableRoom.rooms.Add(room);
                        }
                }

                return suitableRoom;
        }

        public DataBase SuitableOptionWithCertainRoomsNumber(int roomCount)
        {
                DataBase suitableRoom = new DataBase();

                foreach (var room in rooms)
                {
                        if (room.RoomCount == roomCount)
                        {
                                suitableRoom.rooms.Add(room);
                        }
                }

                return suitableRoom;
        }

        public DataBase SuitableOptionWithCertainRoomsNumberAndFloor(int roomCount, int from,
int to)
        {
                DataBase suitableRoom = new DataBase();

                foreach (var room in rooms)
                {
                        if (room.RoomCount == roomCount && room.Floor >= from && room.Floor <=
to)
                        {
                                suitableRoom.rooms.Add(room);
                        }
                }

                return suitableRoom;
        }

        public DataBase SuitableOptionWithLargerRoomsArea(int area)
        {
                DataBase suitableRoom = new DataBase();

                foreach (var room in rooms)
                {
                        if (room.Area > area)
                                suitableRoom.rooms.Add(room);
                }

                return suitableRoom;
        }
```

```
        public void FromFreeToOccupiedList(Room room)
        {
            var relocatable = rooms.FirstOrDefault(e => e == room);
            relocatable.IsRent = false;
        }

        public void FromOccupiedToFreeList(Room room)
        {
            var relocatable = rooms.FirstOrDefault(e => e == room);
            relocatable.IsRent = true;
        }

        public void Print()
        {
            foreach (var room in rooms)
                room.Print();

        }
    }
}
```

Program.cs

```
using System;
using System.IO;
using ExcelDataReader;

namespace Lab3_2
{
    class Program
    {
        static void Main(string[] args)
        {
            DataBase data = new DataBase() { };

            string FilePath = "D://Rooms.xlsx";
            FileStream fileStream = File.Open(FilePath, FileMode.Open, FileAccess.Read);

            using (var stream = new MemoryStream())
            {
                fileStream.CopyTo(stream);
                stream.Position = 0;


    System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);

                    using (var reader = ExcelReaderFactory.CreateReader(stream))
                    {
                        reader.Read();
                        int lineNumber = 0;

                        while (reader.Read())
                        {
                            int columnCount = reader.FieldCount;

                            var roomCount = columnCount > 0 ? reader.GetValue(0) :
null;
                            var area = columnCount > 0 ? reader.GetValue(1) : null;
                            var floor = columnCount > 0 ? reader.GetValue(2) : null;
                            var address = columnCount > 0 ? reader.GetValue(3) : null;
                            var price = columnCount > 0 ? reader.GetValue(4) : null;
                            var isRent = columnCount > 0 ? reader.GetValue(5) : null;

                            data.rooms.Add(new Room()
```

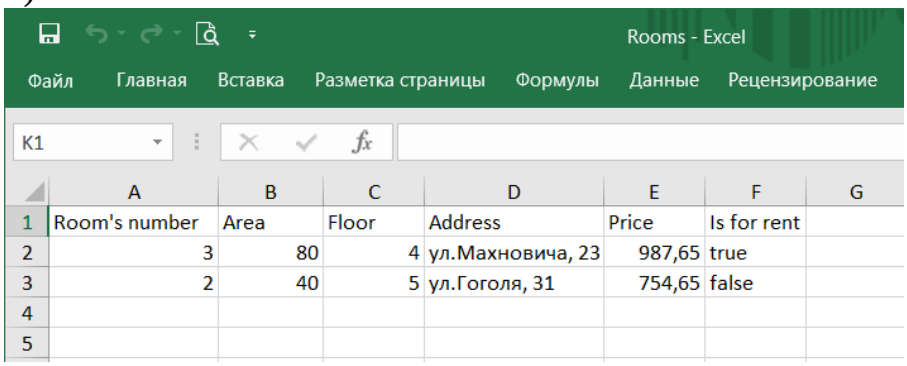
```
                                        {
                                            Id = ++lineNumber,
                                            RoomCount = roomCount is double ?
Convert.ToInt32(roomCount) : 0,
                                            Area = area is double ? Convert.ToInt32(area) : 0,
                                            Floor = floor is double ? Convert.ToInt32(floor) :
0,
                                            Address = address is string ? (string)address :
null,
                                            RentalPrice = price is double ? (double)price : 0,
                                            IsRent = (string)isRent == "true" ? true : false
                                        });
                                }
                            }
                        }

                        Console.WriteLine("Данные из xlsx файла: ");
                        data.Print();

                        Console.WriteLine("Свободные квартиры: ");
                        var fr = data.FreeRoomsList();
                        fr.Print();

                        Console.WriteLine("Занятые квартиры: ");
                        var or = data.OccupiedRoomsList();
                        or.Print();

                        Console.WriteLine("Поиск подходящего варианта: ");
                        var so = data.SuitableOption(3, 4, 85);
                        so.Print();

                        Console.WriteLine("Удаление квартиры из списка свободных квартир: ");
                        data.FromFreeToOccupiedList(data.rooms.Find(r => r.Id == 1));
                        data.Print();
                        Console.WriteLine("Свободные квартиры: ");
                        var frAfterDelete = data.FreeRoomsList();
                        frAfterDelete.Print();

                        Console.WriteLine("Удаление квартиры из списка сдаваемых квартир: ");
                        data.FromOccupiedToFreeList(data.rooms.Find(r => r.Id == 2));
                        data.Print();
                        Console.WriteLine("Занятые квартиры: ");
                        var orAfterDelete = data.OccupiedRoomsList();
                        orAfterDelete.Print();

                        Console.WriteLine("Список квартир, имеющих заданное число комнат: ");
                        var roomWithCertainRoomsNumber = data.SuitableOptionWithCertainRoomsNumber(2);
                        roomWithCertainRoomsNumber.Print();

                        Console.WriteLine("Список квартир, имеющих заданное число комнат и
расположенных на этаже, который находится в заданном промежутке: ");
                        var roomWithCertainRoomsNumberAndFloor =
data.SuitableOptionWithCertainRoomsNumberAndFloor(3, 2, 5);
                        roomWithCertainRoomsNumberAndFloor.Print();

                        Console.WriteLine("Список квартир, имеющих площадь, превосходящую заданную:
");
                        var roomWithCertainArea = data.SuitableOptionWithLargerRoomsArea(60);
                        roomWithCertainArea.Print();
                    }
                }
            }
```

Результаты работы программы:

1)

```
Консоль отладки Microsoft Vis
Set: -2, 3,5, 4, 7
Set: -2, 3,5, 4, 7
Set: -2, 3,5, 4, 7, 10
Set: -2, 4, 7
False
Set: -2, 3,5, 4, 7, 10
-2 3,5 4 7 10
```

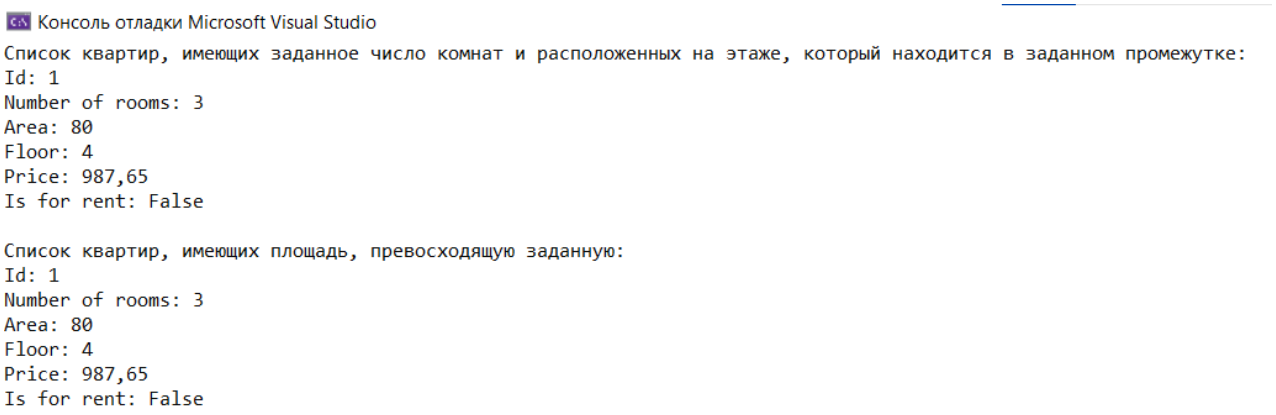
2)



	A	B	C	D	E	F	G
1	Room's number	Area	Floor	Address	Price	Is for rent	
2	3	80	4	ул.Махновича, 23	987,65	true	
3	2	40	5	ул.Гоголя, 31	754,65	false	
4							
5							

```
Консоль отладки Microsoft Visual Studio
Данные из xlsx файла:
Id: 1
Number of rooms: 3
Area: 80
Floor: 4
Price: 987,65
Is for rent: True

Id: 2
Number of rooms: 2
Area: 40
Floor: 5
Price: 754,65
Is for rent: False

Свободные квартиры:
Id: 1
Number of rooms: 3
Area: 80
Floor: 4
Price: 987,65
Is for rent: True

Занятые квартиры:
Id: 2
Number of rooms: 2
Area: 40
Floor: 5
Price: 754,65
Is for rent: False

Поиск подходящего варианта:
Id: 1
Number of rooms: 3
Area: 80
Floor: 4
Price: 987,65
Is for rent: True
```

```
Консоль отладки Microsoft Visual Studio
Удаление квартиры из списка свободных квартир:
Id: 1
Number of rooms: 3
Area: 80
Floor: 4
Price: 987,65
Is for rent: False

Id: 2
Number of rooms: 2
Area: 40
Floor: 5
Price: 754,65
Is for rent: False

Свободные квартиры:
Удаление квартиры из списка сдаваемых квартир:
Id: 1
Number of rooms: 3
Area: 80
Floor: 4
Price: 987,65
Is for rent: False

Id: 2
Number of rooms: 2
Area: 40
Floor: 5
Price: 754,65
Is for rent: True

Занятые квартиры:
Id: 1
Number of rooms: 3
Area: 80
Floor: 4
Price: 987,65
Is for rent: False

Список квартир, имеющих заданное число комнат:
Id: 2
Number of rooms: 2
Area: 40
Floor: 5
Price: 754,65
Is for rent: True
```



Вывод: научилась создавать и использовать классы в программах на языке программирования C#.