МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Специальность ПО3

Выполнила Р.И. Гаврилюк, студентка группы ПО3

Проверил А.А. Крощенко,
ст. преп. кафедры ИИТ,
«——» ———— 2020 г.

Брест 2020

Вариант 6

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

Задание 1.

Реализовать простой класс.

Требования к выполнению

• Реализовать пользовательский класс по варианту.

• 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()

6) Множество вещественных чисел ограниченной мощности – Предусмотреть возможность объединения двух множеств, вывода на печать элементов множества, а так же 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.

Код программы

Set.java

```
class Set{
    private Double[] set;

    public Set(Double[] set){
        this.set = new Double[set.length];
        this.set = set;
    }

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

    public Double[] sort(Double[] set){
        for (int i = 0; i < set.length; ++i){
```

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

        }
        return set;
    }

    public Set mergeSets(Set set2){
        Set mergedSet = new Set(this.set.length);

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

        for(int i = 0; i < set2.set.length; ++i){
            mergedSet.addElement(set2.set[i]);
        }

        mergedSet.set = sort(mergedSet.set);

        for(int i = 0; i < mergedSet.set.length; ++i){
            System.out.print(mergedSet.set[i]);
            System.out.print(" ");
        }
        System.out.println();

        return mergedSet;
    }

    public boolean isPartOfSet(Double x){
        for(int i = 0; i < this.set.length; ++i){
            if(this.set[i].equals(x)){
                return true;
            }
        }
        return false;
    }

    public void printSet(){
        System.out.println("Set:");
        for(int i = 0; i < this.set.length; ++i){
            System.out.print(this.set[i]);
            System.out.print(" ");
        }
        System.out.println();
    }

    public boolean addElement(Double el){
        if (!this.isPartOfSet(el)){
            Double[] newSet = new Double[this.set.length + 1];
```

```
            for(int i = 0; i < this.set.length; ++i){
                newSet[i] = this.set[i];
            }
            newSet[this.set.length] = el;
            this.set = sort(newSet);
            return true;
        }
        return false;
    }

    public boolean deleteElement(Double el){
        boolean isDeleted = this.isPartOfSet(el);
        Double[] newSet = new Double[this.set.length - 1];
        for(int i = 0, j = 0; i < this.set.length; ++i, ++j){
            if(this.set[i].equals(el)){
                ++i;
            }
            newSet[j] = this.set[i];
        }
        this.set = newSet;

        return isDeleted;
    }

    public boolean equals(Set set2){
        if(this.set.length != set2.set.length){
            return false;
        }
        for(int i = 0; i < this.set.length; ++i){
            if(!set2.set[i].equals(this.set[i])){
                return false;
            }
        }
        return true;
    }

    public String toString(){
        String result = new String();
        for(int i = 0; i < this.set.length; ++i){
            result += Double.toString(this.set[i]);
        }
        return result;
    }
}
```

Laba3.java

```
class Main{
    public static void main(String[] args) {
        //task 1
        Set set = new Set(new Double[]{4.0, 7.0, 8.0});
        set.printSet();
        set = set.mergeSets(new Set(new Double[] {3.0, 4.0, 5.0}));
        set.addElement(5.0);
```

```
        set.addElement(1.0);
        set.deleteElement(5.0);
        set.printSet();
        System.out.println(set.equals(new Set(new Double[] {1.0, 3.0, 4.0, 7.0, 8.0})));
        System.out.println(set.equals(new Set(new Double[] {9.0, 4.0, 7.0, 8.0})));
        System.out.println(set.toString());
    }
}
```

Рисунки с результатами работы программы

```
Set:
4.0 7.0 8.0
3.0 4.0 5.0 7.0 8.0
Set:
1.0 3.0 4.0 7.0 8.0
true
false
1.03.04.07.08.0
```

Задание 2.

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.

Код программы

Flat.java

```
class Flat{
    private int mFloor;
    private int mNumRooms;
    private Double mArea;
    private Double mPrice;
    private String mAddress;
    private boolean mIsForRent;

    public Flat(int floor, int numRooms, Double area, Double price, String address, bool
ean isForRent){
        this.setAddress(address);
        this.setArea(area);
        this.setFloor(floor);
        this.setIsRented(isForRent);
        this.setNumRooms(numRooms);
        this.setPrice(price);
    }

    public void setFloor(int floor){
        this.mFloor = floor;
    }

    public void setNumRooms(int numRooms){
        this.mNumRooms = numRooms;

    }

    public void setArea(Double area){
        this.mArea = area;
    }

    public void setPrice(Double price){
        this.mPrice = price;
    }

    public void setAddress(String address){
        this.mAddress = address;
    }

    public void setIsRented(Boolean isForRent){
        this.mIsForRent = isForRent;
    }

    public int getFloor(){
        return this.mFloor;
    }

    public int getNumRooms(){
        return this.mNumRooms;
    }

    public Double getArea(){
        return this.mArea ;
```

```
    }

    public Double getPrice(){
        return this.mPrice;
    }

    public String getAddress(){
        return this.mAddress;
    }

    public Boolean getIsForRent(){
        return this.mIsForRent;
    }

    public void printFlat(){
        System.out.println("---------------------");
        System.out.print("Address: ");
        System.out.println(mAddress);
        System.out.print("Number of rooms:");
        System.out.println(mNumRooms);
        System.out.print("Area: ");
        System.out.println(mArea);
        System.out.print("Floor: ");
        System.out.println(mFloor);
        System.out.print("Price: ");
        System.out.println(mPrice);
        System.out.print("IsForRent: ");
        System.out.println(mIsForRent);
        System.out.println();
    }

    public boolean equals(Flat fl){
        if(this.mFloor == fl.mFloor &&
        this.mNumRooms == fl.mNumRooms &&
        this.mArea.equals(fl.mArea) &&
        this.mPrice.equals(fl.mPrice) &&
        this.mAddress.equals(fl.mAddress) &&
        this.mIsForRent == fl.mIsForRent){
            return true;
        }
        return false;
    }
}
```

Db.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.function.DoubleBinaryOperator;

class db{
    private List<Flat> mListOfFlats;

    public db(){
        mListOfFlats = new ArrayList<>();
```

```
}

public boolean addFlat(Flat fl){
    return mListOfFlats.add(fl);
}

public boolean deleteFlat(Flat fl){
    return mListOfFlats.remove(fl);
}

public List<Flat> getListOfFlatsWithMoreArea(Double area){
    List<Flat> listOfFlatsWithMoreArea = new ArrayList<>();
    for(Flat flat : mListOfFlats){
        if(flat.getArea() > area){
            listOfFlatsWithMoreArea.add(flat);
        }
    }
    return listOfFlatsWithMoreArea;
}

public List<Flat> getListOfFlatsWithNumRooms(int numRooms){
    System.out.println(numRooms);

    List<Flat> listOfFlatsWithMoreArea = new ArrayList<>();
    for(Flat flat : mListOfFlats){
        if(flat.getNumRooms() == numRooms){
            listOfFlatsWithMoreArea.add(flat);
        }
    }
    return listOfFlatsWithMoreArea;
}

public List<Flat> getListOfFlatsWithNumRoomsAndFloor(int numRooms, int floor){
    System.out.print(numRooms);
    System.out.print(" and floor: ");
    System.out.println(floor);

    List<Flat> listOfFlatsWithMoreArea = new ArrayList<>();
    for(Flat flat : mListOfFlats){
        if(flat.getNumRooms() == numRooms
        && flat.getFloor() == floor){
            listOfFlatsWithMoreArea.add(flat);
        }
    }
    return listOfFlatsWithMoreArea;
}

public void print(){
    System.out.println("List of all flats: ");

    for(Flat flat : mListOfFlats){
        flat.printFlat();
    }
}
```

```
    public void moveFromRentedListToFreeList(Flat fl){
        for(Flat flat : mListOfFlats){
            if(flat.equals(fl)){
                flat.setIsRented(false);
            }
        }
    }

    public void moveFromFreeListToRentedList(Flat fl){
        for(Flat flat : mListOfFlats){
            if(flat.equals(fl)){
                flat.setIsRented(true);
            }
        }
    }

    public List<Flat> findSuitableOption(int floor, int numRooms, Double area){
        List<Flat> listOfSuitableFlats = new ArrayList<>();

        for(Flat flat : this.getListOfFreeFlats()){
            if(flat.getNumRooms() == numRooms &&
            flat.getFloor() == floor){
                if(flat.getArea() - area >= 0 && flat.getArea() - area <= 10
                || area - flat.getArea() >= 0 && area - flat.getArea() <= 10) {
                    listOfSuitableFlats.add(flat);
                    break;
                }
            }
        }

        return listOfSuitableFlats;
    }

    public List<Flat> getListOfFreeFlats(){
        List<Flat> listOfFreeFlats = new ArrayList<>();
        for(Flat flat : mListOfFlats){
            if(flat.getIsForRent()){
                listOfFreeFlats.add(flat);
            }
        }
        return listOfFreeFlats;
    }

    public List<Flat> getListOfRentedFlats(){
        List<Flat> listOfRentedFlats = new ArrayList<>();
        for(Flat flat : mListOfFlats){
            if(!flat.getIsForRent()){
                listOfRentedFlats.add(flat);
            }
        }
        return listOfRentedFlats;
    }

    public void printListOfFlats(List<Flat> listOflats){
        for(Flat flat : listOflats){
```

```
            flat.printFlat();
        }
    }
}
```

Laba3.java

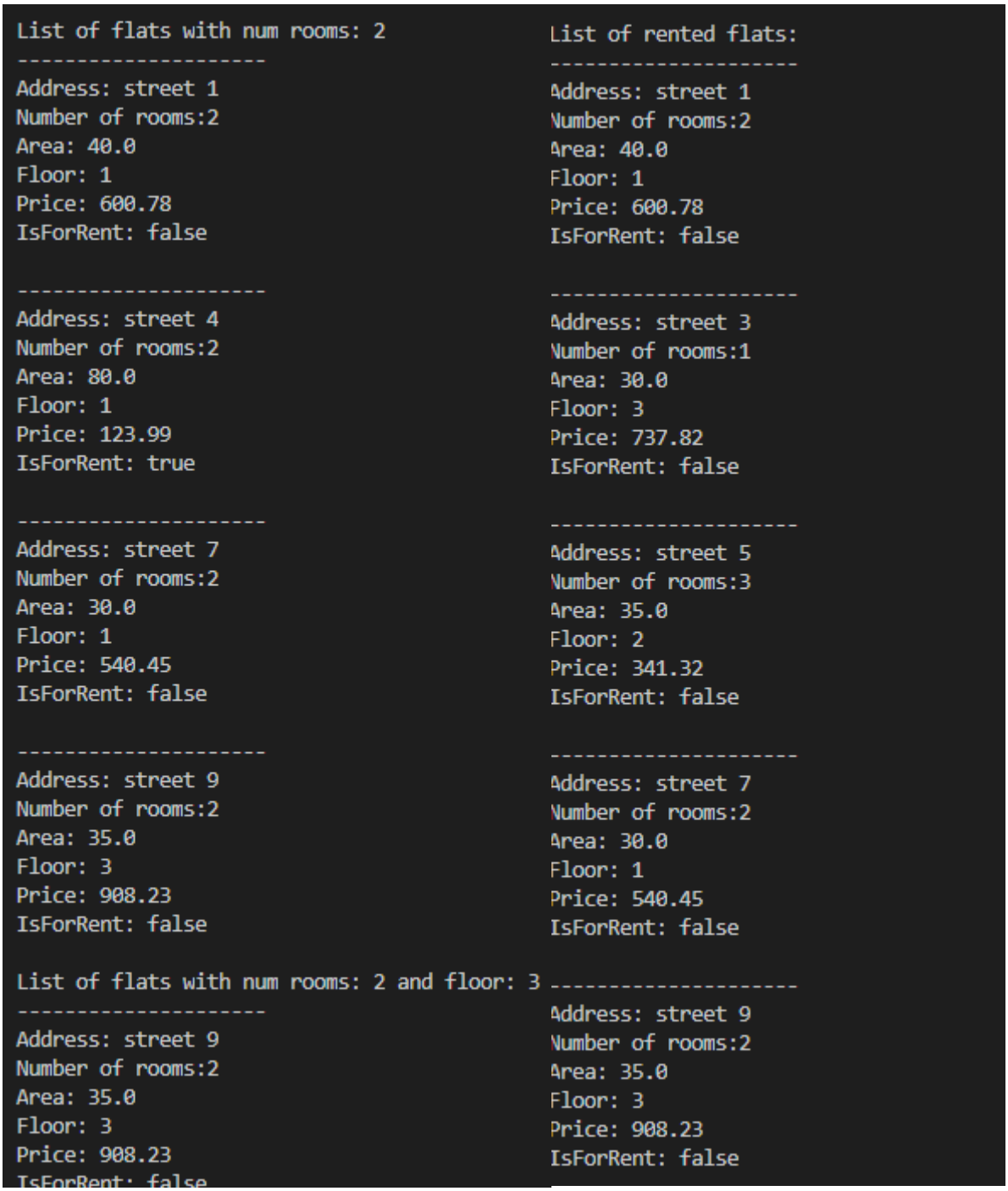
```
class Main{
    public static void main(String[] args) {
        //task 2
        Flat fl1 = new Flat(1, 2, 30.0, 890.09, "street Abracadabra", true);
        db flats = new db();
        flats.addFlat(new Flat(1, 2, 40.0, 600.78, "street 1", false));
        flats.addFlat(new Flat(2, 3, 95.0, 892.19, "street 2", true));
        flats.addFlat(new Flat(3, 1, 30.0, 737.82, "street 3", false));
        flats.addFlat(new Flat(1, 2, 80.0, 123.99, "street 4", true));
        flats.addFlat(new Flat(2, 3, 35.0, 341.32, "street 5", false));
        flats.addFlat(new Flat(3, 1, 20.0, 245.87, "street 6", true));
        flats.addFlat(new Flat(1, 2, 30.0, 540.45, "street 7", false));
        flats.addFlat(new Flat(2, 3, 40.0, 435.91, "street 8", true));
        flats.addFlat(new Flat(3, 2, 35.0, 908.23, "street 9", false));

        flats.print();

        flats.addFlat(fl1);
        flats.print();

        flats.deleteFlat(fl1);
        flats.print();

        System.out.println("List of suitable flats: ");
        flats.printListOfFlats(flats.findSuitableOption(2, 3, 90.0));

        System.out.println("List of flats with superior area:");
        flats.printListOfFlats(flats.getListOfFlatsWithMoreArea(40.0));

        System.out.print("List of flats with num rooms: ");
        flats.printListOfFlats(flats.getListOfFlatsWithNumRooms(2));

        System.out.print("List of flats with num rooms: ");
        flats.printListOfFlats(flats.getListOfFlatsWithNumRoomsAndFloor(2, 3));

        System.out.println("List of rented flats: ");
        flats.printListOfFlats(flats.getListOfRentedFlats());

        System.out.println("Moving flat from state empty to state rented.");
        flats.moveFromFreeListToRentedList(fl1);

        System.out.println("List of rented flats: ");
        flats.printListOfFlats(flats.getListOfRentedFlats());

        System.out.println("List of free flats: ");
        flats.printListOfFlats(flats.getListOfFreeFlats());
```

```
        System.out.println("Moving flat from state rented to state empty.");
        flats.moveFromFreeListToRentedList(fl1);

        System.out.println("List of free flats: ");
        flats.printListOfFlats(flats.getListOfFreeFlats());

        flats.deleteFlat(fl1);
        flats.print();
    }
}
```

Рисунки с результатами работы программы

```
List of all flats:
--------------------
Address: street 1
Number of rooms:2
Area: 40.0
Floor: 1
Price: 600.78
IsForRent: false

--------------------
Address: street 2
Number of rooms:3
Area: 95.0
Floor: 2
Price: 892.19
IsForRent: true

--------------------
Address: street 3
Number of rooms:1
Area: 30.0
Floor: 3
Price: 737.82
IsForRent: false

--------------------
Address: street 4
Number of rooms:2
Area: 80.0
Floor: 1
Price: 123.99
IsForRent: true

--------------------
Address: street 5
Number of rooms:3
Area: 35.0
Floor: 2
Price: 341.32
IsForRent: false

--------------------
Address: street 6
```

```
List of suitable flats:
--------------------
Address: street 2
Number of rooms:3
Area: 95.0
Floor: 2
Price: 892.19
IsForRent: true
```



Вывод: научилась создавать и использовать классы в программах на языке программирования Java