Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Лабораторная работа №3
по дисциплине «ССП»

Выполнил:
Студент 4 курса
Факультета ЭИС
Группы АС – 50
Мендель Д.А.
Проверил:
Войцехович Г.Ю.

Брест 2020

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

Вариант 6

Задание 1

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.

Задание 2

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:

```
package com.company;

import com.company.task1.SetOfDoubleNumbers;
import com.company.task2.FlatService;

import java.io.IOException;

public class Main {

   public static void main(String[] args) {
```

```
        // TASK 1
        System.out.println("Task 1");
        SetOfDoubleNumbers set = new SetOfDoubleNumbers(new double[] {1,2,3,4,5});
         System.out.println(set.isBelongToSet(2));
        set.addElement(8);
        set.addElement(4);
        set.deleteElement(5);
        set.deleteElement(5);
        SetOfDoubleNumbers set1 = new SetOfDoubleNumbers(new double[] {1,2,3,7,9});
        set.mergeSets(set1);
         System.out.println(set);

         // TASK 2
        FlatService flatService = new FlatService();
        try {
          System.out.println("\nTask 2\nList of free flats:");
          flatService.getList(true).forEach(System.out::println);

          System.out.println("\nList of suitable options:");
          flatService.findSuitableOption(6,4,150.0).forEach(System.out::println);

          // make flat is busy
          System.out.println("\nMaking flat with 6 rooms not free...");
          flatService.changeIsFree(flatService.findSuitableOption(6).get(0));

          System.out.println("\nCheck if flat is busy now");
          System.out.println(flatService.findSuitableOption(6).get(0));

          System.out.println("\nList of all flats:");
          flatService.getList().forEach(System.out::println);

          System.out.println("\nList of flats with 2 rooms:");
          flatService.findSuitableOption(2).forEach(System.out::println);

          System.out.println("\nList of flats with 2 rooms on floor 1-3:");
          flatService.findSuitableOption(2,1,3).forEach(System.out::println);

          System.out.println("\nList of flats with square more than 105:");
          flatService.findSuitableOption(105.0).forEach(System.out::println);

          // restore data to initial state
          flatService.changeIsFree(flatService.findSuitableOption(6).get(0));
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
    }

    package com.company.task1;
```

```
import java.util.Arrays;
import java.util.Objects;

public class SetOfDoubleNumbers {

   private double[] set;
   private int power;

   SetOfDoubleNumbers(int power) {
      this.power = power;
      set = new double[power];
   }

   public double[] getSet() {
      return getSet();
   }

   public SetOfDoubleNumbers(double[] initialSet) {
      if (null == initialSet) {
         set = new double[0];

      } else set = initialSet;
      power = set.length;
   }

   public boolean isBelongToSet(double element) {
      return Arrays.stream(set).anyMatch(thisElement -> thisElement == element);
   }

   public void addElement(double element) {
      if (!isBelongToSet(element)) {
         double[] tempSet = Arrays.copyOf(set, power + 1);
         tempSet[tempSet.length - 1] = element;
         this.set = tempSet;
         power = tempSet.length;
      } else System.out.println("This element is already in the set: " + element);
   }


   public void deleteElement(double element) {
      double[] resultArray;
      if (isBelongToSet(element)) {
         resultArray = new double[set.length - 1];
      } else
         resultArray = new double[set.length];

      for (int i = 0, j = 0; i < set.length; i++, j++) {
         if (set[i] == element) {
            power -= 1;
            j -= 1;
```

```
            } else resultArray[j] = set[i];
        }
        set = resultArray;
    }

    public SetOfDoubleNumbers mergeSets(SetOfDoubleNumbers newSet) {
        for (double element :
                newSet.set) {
            addElement(element);
        }
        return null;
    }

    public int getPower() {
        return power;
    }


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SetOfDoubleNumbers that = (SetOfDoubleNumbers) o;
        return power == that.power &&
                Arrays.equals(set, that.set);
    }

    @Override
    public int hashCode() {
        int result = Objects.hash(power);
        result = 31 * result + Arrays.hashCode(set);
        return result;
    }

    @Override
    public String toString() {
        return "set = " + Arrays.toString(set) +
                "\npower = " + power;
    }
}
```

```
package com.company.task2;

import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class FlatService {
    private final String PATH = System.getProperty("user.dir") + "\\test.txt";
```

```
private BufferedReader readFile(String path) {
    try {
        File file = new File(path);
        //создаем объект FileReader для объекта File
        FileReader fr = new FileReader(file);
        //создаем BufferedReader с существующего FileReader для построчного считывания
        return new BufferedReader(fr);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}

public List<Flat> getList() throws IOException {
    List<Flat> listOfFlat = new ArrayList<>();
    BufferedReader reader = readFile(PATH);
    assert reader != null;
    String line = reader.readLine();
    while (line != null) {
        listOfFlat.add(parseLineToFlat(line));
        line = reader.readLine();
    }
    return listOfFlat;
}

public List<Flat> getList(boolean isFree) throws IOException {
    List<Flat> listOfFlat = new ArrayList<>();
    BufferedReader reader = readFile(PATH);
    assert reader != null;
    String line = reader.readLine();
    while (line != null) {
        Flat flat = parseLineToFlat(line);
        if (flat.isFree() == isFree)
            listOfFlat.add(parseLineToFlat(line));
        line = reader.readLine();
    }
    return listOfFlat;
}

private Flat parseLineToFlat(String line) {
    Flat flat = new Flat();
    String[] array = line.split("--");
    flat.setNumberOfRooms(Integer.parseInt(array[0]));
    flat.setSquare(Double.parseDouble(array[1]));
    flat.setFloor(Integer.parseInt(array[2]));
    flat.setAddress(array[3]);
    flat.setPrice(Integer.parseInt(array[4]));
    flat.setFree(Boolean.parseBoolean(array[5]));
    return flat;
```

```
    }

    public List<Flat> findSuitableOption(int numberOfRooms, int floor, double square) throws IOException
{

        return getList(true).stream().filter(flat ->
        {
            boolean n = (flat.getNumberOfRooms() == numberOfRooms);
            boolean f = (flat.getFloor() == floor);
            boolean sq = (flat.getSquare() <= square + 10 && flat.getSquare() >= square - 10);
            return (n && f && sq);
        }).collect(Collectors.toList());
    }

    public List<Flat> findSuitableOption(int numberOfRooms) throws IOException {
        return getList().stream().filter(flat -> flat.getNumberOfRooms() ==
numberOfRooms).collect(Collectors.toList());
    }

    public List<Flat> findSuitableOption(int numberOfRooms, int minFloor, int maxFloor) throws
IOException {
        return getList().stream().filter(
                flat -> flat.getNumberOfRooms() == numberOfRooms
                        && flat.getFloor() >= minFloor
                        && flat.getFloor() <= maxFloor)
                .collect(Collectors.toList());
    }

    public List<Flat> findSuitableOption(double square) throws IOException {
        return getList().stream().filter(
                flat -> flat.getSquare() > square)
                .collect(Collectors.toList());

    }

    public void changeIsFree(Flat flat) throws IOException {
        rewriteFile(PATH, flat);
    }

    private void rewriteFile(String path, Flat flat) throws IOException {
        BufferedReader reader = readFile(path);
        StringBuilder buffer = new StringBuilder();
        assert reader != null;
        String line = reader.readLine();
        while (line != null) {
            buffer.append(line).append(System.lineSeparator());
            line = reader.readLine();
        }
        reader.close();
        String fileContents = buffer.toString();
```

```
        String oldLine = flat.toTempString();
        flat.setFree(!flat.isFree());
        String newLine = flat.toTempString();
        //Replacing the old line with new line
        fileContents = fileContents.replaceAll(oldLine, newLine);
        //instantiating the FileWriter class
        FileWriter writer = new FileWriter(path);
        writer.append(fileContents);
        writer.flush();
    }
}
```

```
package com.company.task2;

public class Flat {
    private int numberOfRooms;
    private double square;
    private int floor;
    private String address;
    private int price;
    private boolean isFree;

    @Override
    public String toString() {
        return "Flat{" +
                "numberOfRooms=" + numberOfRooms +
                ", square=" + square +
                ", floor=" + floor +
                ", address='" + address + '\'' +
                ", price=" + price +
                ", isFree=" + isFree +
                '}';
    }

    public String toTempString() {
        return ""
                + numberOfRooms + "--"
                + square + "--"
                + floor + "--"
                + address + "--"
                + price + "--"
                + isFree;
    }

    public int getNumberOfRooms() {
        return numberOfRooms;
    }

    public void setNumberOfRooms(int numberOfRooms) {
        this.numberOfRooms = numberOfRooms;
```

```
    }

    public double getSquare() {
        return square;
    }

    public void setSquare(double square) {
        this.square = square;
    }

    public int getFloor() {
        return floor;
    }

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

    public String getAddress() {
        return address;
    }

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

    public int getPrice() {
        return price;
    }

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

    public boolean isFree() {
        return isFree;
    }

    public void setFree(boolean free) {
        isFree = free;
    }
}
```

Тестовые данные:

```
1--100.0--1--г.Брест,Ул.Советская,44,33--12--true
2--105.0--2--г.Брест,Ул.Советская,44,34--132--true
2--115.0--2--г.Брест,Ул.Советская,44,35--132--true
2--115.0--3--г.Брест,Ул.Советская,44,36--132--true
2--115.0--3--г.Брест,Ул.Советская,44,37--132--true
3--120.0--4--г.Брест,Ул Советская,44,38--122--false
6--155.0--4--г.Брест,Ул.Советская,44,39--122--true
```

Результат :

```
Task 1
true
This element is already in the set: 4.0
This element is already in the set: 1.0
This element is already in the set: 2.0
This element is already in the set: 3.0
set = [1.0, 2.0, 3.0, 4.0, 8.0, 7.0, 9.0]
power = 7
```

```
Task 2
List of free flats:
Flat{numberOfRooms=1, square=100.0, floor=1, address='г.Брест,Ул.Советская,44,33', price=12, isFree=true}
Flat{numberOfRooms=2, square=105.0, floor=2, address='г.Брест,Ул.Советская,44,34', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=2, address='г.Брест,Ул.Советская,44,35', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,36', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,37', price=132, isFree=true}
Flat{numberOfRooms=6, square=155.0, floor=4, address='г.Брест,Ул.Советская,44,39', price=122, isFree=true}

List of suitable options:
Flat{numberOfRooms=6, square=155.0, floor=4, address='г.Брест,Ул.Советская,44,39', price=122, isFree=true}

Making flat with 6 rooms not free...

Check if flat is busy now
Flat{numberOfRooms=6, square=155.0, floor=4, address='г.Брест,Ул.Советская,44,39', price=122, isFree=false}

List of all flats:
Flat{numberOfRooms=1, square=100.0, floor=1, address='г.Брест,Ул.Советская,44,33', price=12, isFree=true}
Flat{numberOfRooms=2, square=105.0, floor=2, address='г.Брест,Ул.Советская,44,34', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=2, address='г.Брест,Ул.Советская,44,35', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,36', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,37', price=132, isFree=true}
Flat{numberOfRooms=3, square=120.0, floor=4, address='г.Брест,Ул Советская,44,38', price=122, isFree=false}
Flat{numberOfRooms=6, square=155.0, floor=4, address='г.Брест,Ул.Советская,44,39', price=122, isFree=false}
```

```
List of flats with 2 rooms:
Flat{numberOfRooms=2, square=105.0, floor=2, address='г.Брест,Ул.Советская,44,34', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=2, address='г.Брест,Ул.Советская,44,35', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,36', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,37', price=132, isFree=true}

List of flats with 2 rooms on floor 1-3:
Flat{numberOfRooms=2, square=105.0, floor=2, address='г.Брест,Ул.Советская,44,34', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=2, address='г.Брест,Ул.Советская,44,35', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,36', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,37', price=132, isFree=true}

List of flats with square more than 105:
Flat{numberOfRooms=2, square=115.0, floor=2, address='г.Брест,Ул.Советская,44,35', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,36', price=132, isFree=true}
Flat{numberOfRooms=2, square=115.0, floor=3, address='г.Брест,Ул.Советская,44,37', price=132, isFree=true}
Flat{numberOfRooms=3, square=120.0, floor=4, address='г.Брест,Ул Советская,44,38', price=122, isFree=false}
Flat{numberOfRooms=6, square=155.0, floor=4, address='г.Брест,Ул.Советская,44,39', price=122, isFree=false}
```

Вывод: в ходе выполнения лабораторной работы приобрел практические навыки языка программирования Java.