МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчёт
по лабораторной работе №3

Выполнил:
студент группы ПО-9
Зеленков К. И.

Проверил:
Крощенко А. А.

Брест 2024

Вариант 6

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

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.

Код программы:

NumberSet.java:

```
import java.util.Arrays;

public class NumberSet {
    private double[] elements;
    private int capacity;
    private int factSize;

    public NumberSet(int capacity) {
        this.capacity = capacity;
        this.elements = new double[capacity];
        this.factSize = 0;
    }

    private int indexOf(double element) {
        for (int i = 0; i < factSize; i++) {
            if (elements[i] == element) {
                return i;
            }
        }
        return -1;
    }

    public boolean contains(double element) {
        return indexOf(element) != -1;
    }

    public void addElement(double element) {
        if(!contains(element)){
            if (factSize < capacity) {
                elements[factSize++] = element;
            }
            else {
                System.out.println("Множество полное, невозможно добавить
новый элемент.");
            }
        }
```

```
    }

    public void removeElement(double element) {
        int index = indexOf(element);
        if (index != -1) {
            System.arraycopy(elements, index + 1, elements, index, factSize
- index - 1);
            factSize--;
        } else {
            System.out.println("Элемент не найден в множестве.");
        }
    }

    public NumberSet union(NumberSet otherSet) {
        int newCapacity = this.capacity + otherSet.capacity;
        NumberSet newSet = new NumberSet(newCapacity);

        System.arraycopy(this.elements, 0, newSet.elements, 0,
this.factSize);
        newSet.factSize = this.factSize;

        for (int i = 0; i < otherSet.factSize; i++) {
            if (!newSet.contains(otherSet.elements[i])) {
                newSet.addElement(otherSet.elements[i]);
            }
        }

        return newSet;
    }

    @Override
    public String toString() {
        return Arrays.toString(Arrays.copyOf(elements, factSize));
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        NumberSet that = (NumberSet) obj;
        return Arrays.equals(elements, that.elements);
    }
}
```

Main1.java:

```
public class Main1 {
    public static void main(String[] args) {
        NumberSet set1 = new NumberSet(5);
        set1.addElement(1.0); //1
        set1.addElement(2.0); //2
        set1.addElement(3.1); //3
        set1.addElement(49.2); //4
        set1.removeElement(3.1); //3
        set1.addElement(3.0); //4
        set1.addElement(2.0); //5
        System.out.println("множество set1: " + set1);
        set1.removeElement(2.0);
        System.out.println("Множество set1 после удаления элемента 2.0: " +
set1);
        System.out.println("Принадлежит ли 49.2 множеству set1: " +
set1.contains(49.2));

        NumberSet set2 = new NumberSet(5);
        set2.addElement(2.0); //1
        set2.addElement(3.0); //2
```

```
        set2.addElement(4.0); //3
        set2.addElement(5.0); //4
        set2.addElement(6.0); //5
        set2.addElement(9.0); // проверка на полноту (6)
        System.out.println("множество set2: " + set2);

        NumberSet unionSet = set1.union(set2);
        System.out.println("Объединение множеств: " + unionSet);


        boolean areEqual = set1.equals(set2); //Проверка на равенство двух
множеств

        if (areEqual) {
            System.out.println("Множества set1 и set2 равны.");
        } else {
            System.out.println("Множества set1 и set2 не равны.");
        }
    }
}
```

Результат работы программы:

```
Множество set1: [1.0, 2.0, 24.4, 3.0]
Множество set1 после удаления элемента 2.0: [1.0, 24.4, 3.0]
Принадлежит ли 24.4 множеству set1: true
Множество полное, невозможно добавить новый элемент.
Множество set2: [2.0, 3.0, 4.0, 5.0, 6.0]
Объединение множеств: [1.0, 24.4, 3.0, 2.0, 4.0, 5.0, 6.0]
Множества set1 и set2 не равны.
```

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

Apartment.java:

```
public class Apartment {
    private int numberOfRooms; //количество комнат;
    private int floor; //этаж;
    private double totalArea; //общая площадь;
    private double rentPrice; //цена аренды.
    private String address; //адрес;
    private boolean isOccupied; // сдается ли квартира.

    public Apartment(int numberOfRooms, double totalArea, int floor, String
address, double rentPrice) {
        this.numberOfRooms = numberOfRooms;
        this.totalArea = totalArea;
        this.floor = floor;
        this.address = address;
        this.rentPrice = rentPrice;
        this.isOccupied = false;
    }

    public int getNumberOfRooms() {
        return numberOfRooms;
    }

    public double getTotalArea() {
        return totalArea;
    }

    public int getFloor() {
        return floor;
    }

    public boolean isOccupied() {
        return isOccupied;
    }

    public void setOccupied(boolean occupied) {
        isOccupied = occupied;
    }

    @Override
    public String toString() {
        return "Apartment{" +
                "Rooms=" + numberOfRooms +
                ", Area=" + totalArea +
                ", floor=" + floor +
                ", address='" + address + '\'' +
                ", Price=" + rentPrice +
                ", " + (isOccupied?"Сдана":"Сдаётся") +
                '}';
    }
}
```

ApartmentDAO.java:

```
import java.io.BufferedReader;
import java.io.FileReader;
```

```java
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class ApartmentDAO {
    private List<Apartment> availableApartments;
    private List<Apartment> occupiedApartments;

    public ApartmentDAO() {
        this.availableApartments = new ArrayList<>();
        this.occupiedApartments = new ArrayList<>();
    }

    public void addApartment(Apartment apartment) {
        availableApartments.add(apartment);
    }

    public List<Apartment> getAvailableApartments() {
        return availableApartments;
    }

    public void displayAllApartments() {
        displayAvailableApartments();
        displayOccupiedApartments();
    }

    public void displayAvailableApartments() {
        System.out.println("Доступные апартаменты:");
        for (Apartment apartment : availableApartments) {
            System.out.println(apartment);
        }
    }

    public void displayOccupiedApartments() {
        System.out.println("Занятые квартиры:");
        for (Apartment apartment : occupiedApartments) {
            System.out.println(apartment);
        }
    }

    public void findMatchingApartment(int numberOfRooms, int floor, double
minArea) {
        for (Apartment apartment : availableApartments) {
            if (apartment.isOccupied()) {
                continue;
            }

            if (apartment.getNumberOfRooms() == numberOfRooms &&
                    apartment.getFloor() == floor &&
                    Math.abs(apartment.getTotalArea() - minArea) <= 10) {
                System.out.println("Найдена подходящая квартира: " +
apartment);
                return;
            }
        }

        System.out.println("Подходящая квартира не найдена.");
    }

    public void rentApartment(Apartment apartment) {
        if (availableApartments.contains(apartment)) {
            availableApartments.remove(apartment);
            apartment.setOccupied(true);
            occupiedApartments.add(apartment);
            System.out.println("Квартира успешно сдана в аренду: " +
apartment);
```

```
        } else {
            System.out.println("Квартира не сдается в аренду.");
        }
    }

    public List<Apartment> getApartmentsByNumberOfRooms(int numberOfRooms) {
        List<Apartment> result = new ArrayList<>();
        for (Apartment apartment : availableApartments) {
            if (apartment.getNumberOfRooms() == numberOfRooms) {
                result.add(apartment);
            }
        }
        return result;
    }

    public List<Apartment> getApartmentsByRoomsAndFloor(int numberOfRooms,
int floorRangeStart, int floorRangeEnd) {
        List<Apartment> result = new ArrayList<>();
        for (Apartment apartment : availableApartments) {
            if (apartment.getNumberOfRooms() == numberOfRooms &&
                    apartment.getFloor() >= floorRangeStart &&
                    apartment.getFloor() <= floorRangeEnd) {
                result.add(apartment);
            }
        }
        return result;
    }

    public List<Apartment> getApartmentsByArea(double minArea) {
        List<Apartment> result = new ArrayList<>();
        for (Apartment apartment : availableApartments) {
            if (apartment.getTotalArea() > minArea) {
                result.add(apartment);
            }
        }
        return result;
    }

    public static void loadFromFile(String fileName, ApartmentDAO system) {
        try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(", ");
                if (parts.length == 5) {
                    int numberOfRooms = Integer.parseInt(parts[0]);
                    double totalArea = Double.parseDouble(parts[1]);
                    int floor = Integer.parseInt(parts[2]);
                    String address = parts[3];
                    double rentPrice = Double.parseDouble(parts[4]);

                    Apartment apartment = new Apartment(numberOfRooms,
totalArea, floor, address, rentPrice);
                    system.addApartment(apartment);
                }
            }
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Main2.java:

```
import java.util.List;

public class Main2 {
    public static void main(String[] args) {
        ApartmentDAO system = new ApartmentDAO();

        String path =
"C:\\Users\\kosty\\IdeaProjects\\lab3\\src\\apart.txt";

        // Загрузка данных из файла
        ApartmentDAO.loadFromFile(path, system);

        // Отображение всех квартир
        system.displayAllApartments();

        // Аренда квартиры
        List<Apartment> availableApartments =
system.getAvailableApartments();
        if (!availableApartments.isEmpty()) {
            Apartment apartmentToRent = availableApartments.get(0);
            system.rentApartment(apartmentToRent);
        } else {
            System.out.println("Свободных квартир для сдачи в аренду нет.");
        }

        // Отображение всех квартир после аренды
        system.displayAllApartments();

        // Поиск подходящей квартиры
        system.findMatchingApartment(2, 3, 80.0);

        system.findMatchingApartment(2, 4, 75.0);

        // Список квартир с заданным числом комнат
        List<Apartment> apartmentsByNumberOfRooms =
system.getApartmentsByNumberOfRooms(2);
        System.out.println("Апартаменты с 2 комнатами: " +
apartmentsByNumberOfRooms);

        // Список квартир с заданным числом комнат и этажем в заданном
промежутке
        List<Apartment> apartmentsByRoomsAndFloor =
system.getApartmentsByRoomsAndFloor(2, 1, 5);
        System.out.println("Апартаменты с 2 комнатами и этажом от 1 до 5: "
+ apartmentsByRoomsAndFloor);

        // Список квартир с площадью, превосходящей заданную
        List<Apartment> apartmentsByArea = system.getApartmentsByArea(80.0);
        System.out.println("Квартиры площадью более 80: " +
apartmentsByArea);
    }
}
```

apart.txt:

```
2, 75.5, 6, Sportivnaya 19, 1100.0
2, 90.0, 4, Gavrilova 223, 1530.0
2, 50.0, 2, Sovetkaya 13, 920.0
3, 80.0, 5, Shorsa 29, 1300.0
2, 80.0, 3, Shorsa 30, 1400.0
1, 80.0, 3, Shorsa 14, 1400.0
```

Результат работы программы:

Считываем из файла квартиры и выводим их на экран:

```
Доступные апартаменты:
Apartment{Rooms=2, Area=75.5, floor=6, address='Sportivnaya 19', Price=1100.0, Сдаётся}
Apartment{Rooms=2, Area=90.0, floor=4, address='Gavrilova 223', Price=1530.0, Сдаётся}
Apartment{Rooms=2, Area=50.0, floor=2, address='Sovetkaya 13', Price=920.0, Сдаётся}
Apartment{Rooms=3, Area=80.0, floor=5, address='Shorsa 29', Price=1300.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=3, address='Shorsa 30', Price=1400.0, Сдаётся}
Apartment{Rooms=1, Area=80.0, floor=3, address='Shorsa 14', Price=1400.0, Сдаётся}
```

Арендуем квартиру и выводим список арендованных квартир.

```
Занятые квартиры:
Квартира успешно сдана в аренду: Apartment{Rooms=2, Area=75.5, floor=6, address='Sportivnaya 19', Price=1100.0, Сдана}
```

Выводим все квартиры, проверяем что одна квартира сдана.

```
Доступные апартаменты:
Apartment{Rooms=2, Area=90.0, floor=4, address='Gavrilova 223', Price=1530.0, Сдаётся}
Apartment{Rooms=2, Area=50.0, floor=2, address='Sovetkaya 13', Price=920.0, Сдаётся}
Apartment{Rooms=3, Area=80.0, floor=5, address='Shorsa 29', Price=1300.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=3, address='Shorsa 30', Price=1400.0, Сдаётся}
Apartment{Rooms=1, Area=80.0, floor=3, address='Shorsa 14', Price=1400.0, Сдаётся}
Занятые квартиры:
Apartment{Rooms=2, Area=75.5, floor=6, address='Sportivnaya 19', Price=1100.0, Сдана}
```

Теперь найдем по данным критерием доступные квартиры:

1. Количество комнат: 2
2. Этаж: 3
3. Площадь приближена к 80

```
Найдена подходящая квартира: Apartment{Rooms=2, Area=80.0, floor=3, address='Shorsa 30', Price=1400.0, Сдаётся}
```

1. Количество комнат: 2
2. Этаж: 4
3. Площадь приближена к 85.0

```
Найдена подходящая квартира: Apartment{Rooms=2, Area=90.0, floor=4, address='Gavrilova 223', Price=1530.0, Сдаётся}
```

Найдем квартиры по последним пунктам лабораторной работы:

```
Апартаменты с 2 комнатами: [Apartment{Rooms=2, Area=90.0, floor=4,
address='Gavrilova 223', Price=1530.0, Сдаётся}, Apartment{Rooms=2,
Area=50.0, floor=2, address='Sovetkaya 13', Price=920.0, Сдаётся},
Apartment{Rooms=2, Area=80.0, floor=3, address='Shorsa 30', Price=1400.0,
Сдаётся}]
Апартаменты с 2 комнатами и этажом от 1 до 5: [Apartment{Rooms=2, Area=90.0,
floor=4, address='Gavrilova 223', Price=1530.0, Сдаётся}, Apartment{Rooms=2,
Area=50.0, floor=2, address='Sovetkaya 13', Price=920.0, Сдаётся},
Apartment{Rooms=2, Area=80.0, floor=3, address='Shorsa 30', Price=1400.0,
Сдаётся}]
Квартиры площадью более 80: [Apartment{Rooms=2, Area=90.0, floor=4,
address='Gavrilova 223', Price=1530.0, Сдаётся}]
```

Вывод: я научился создавать и использовать классы в программах на языке программирования Java.