

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

Отчёт
по лабораторной работе №3

Выполнил:
студент группы ПО-9
Солышко Дмитрий Андреевич

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:

DoubleNumberSet.java:

```
import java.util.Arrays;

public class DoubleNumberSet {
    private double[] elements;
    private int capacity;
    private int size;

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

    public void addElement(double element) {
        if (size < capacity) {
            elements[size++] = 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, size - index - 1);
            size--;
        } else {
            System.out.println("Элемент не найден в множестве.");
        }
    }

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

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

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

```

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

        return newSet;
    }

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

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

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

```

MainDoubleNumberSet.java:

```

public class MainDoubleNumberSet {
    public static void main(String[] args) {
        DoubleNumberSet set1 = new DoubleNumberSet(5);
        set1.addElement(1.0);
        set1.addElement(2.0);
        set1.addElement(3.1);
        set1.addElement(49.2);

        System.out.println("множество set1: " + set1);

        DoubleNumberSet set2 = new DoubleNumberSet(5);
        set2.addElement(2.0);
        set2.addElement(3.0);
        set2.addElement(4.0);
        set2.addElement(5.0);
        set2.addElement(6.0);
        set2.addElement(9.0); // проверка на полноту множества

        System.out.println("множество set2: " + set2);

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

        System.out.println("Принадлежит ли 4.0 множеству set1: " +
set1.contains(4.0));

        set1.removeElement(2.0);
        System.out.println("Множество set1 после удаления элемента 2.0: " +
set1);
    }
}

```

```

        boolean areEqual = set1.equals(set2);

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

        // Сделаем идентичные элементы у объекты set3 и set2, но с разными
        // мощностями, и проверим равенство
        DoubleNumberSet set3 = new DoubleNumberSet(6);
        set3.addElement(2.0);
        set3.removeElement(2.0);
        set3.addElement(2.0);
        set3.addElement(3.0);
        set3.addElement(4.0);
        set3.addElement(5.0);
        set3.addElement(6.0);

        areEqual = set2.equals(set3);

        if (areEqual) {
            System.out.println("Множества set2 и set3 равны.");
        } else {
            System.out.println("Множества set2 и set3 не равны.");
        }

        // Сделаем идентичные объекты set3 и set4 и проверим равенство
        DoubleNumberSet set4 = new DoubleNumberSet(6);
        set4.addElement(2.0);
        set4.addElement(3.0);
        set4.addElement(4.0);
        set4.addElement(5.0);
        set4.addElement(6.0);

        areEqual = set3.equals(set4);

        if (areEqual) {
            System.out.println("Множества set3 и set4 равны.");
        } else {
            System.out.println("Множества set3 и set4 не равны.");
        }
    }
}

```

Результат работы программы:

```

C:\Users\Disoland\.jdk\openjdk-21.0.2\bin\java.exe "-javaagent:D:\I
множество set1: [1.0, 2.0, 3.1, 49.2]
Множество полное, невозможно добавить новый элемент.
множество set2: [2.0, 3.0, 4.0, 5.0, 6.0]
Объединение множеств: [1.0, 2.0, 3.1, 49.2, 3.0, 4.0, 5.0, 6.0]
Принадлежит ли 4.0 множеству set1: false
Множество set1 после удаления элемента 2.0: [1.0, 3.1, 49.2]
Множества set1 и set2 не равны.
Множества set2 и set3 не равны.
Множества set3 и set4 равны.

Process finished with exit code 0

```

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 double totalArea; //общая площадь;
    private int floor; //этаж;
    private String address; //адрес;
    private double rentPrice; //цена аренды.
    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") +
            '}';
    }
}

```

ApartmentSystem.java:

```

import java.util.ArrayList;
import java.util.List;

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

    public ApartmentSystem() {
        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("Available Apartments:");
        for (Apartment apartment : availableApartments) {
            System.out.println(apartment);
        }
    }

    public void displayOccupiedApartments() {
        System.out.println("Occupied Apartments:");
        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("Matching Apartment found: " +
apartment);
                return;
            }
        }

        System.out.println("No matching apartment found.");
    }

    public void rentApartment(Apartment apartment) {
        if (availableApartments.contains(apartment)) {
            availableApartments.remove(apartment);
            apartment.setOccupied(true);
            occupiedApartments.add(apartment);
            System.out.println("Apartment rented successfully: " +
apartment);
        } else {
            System.out.println("Apartment not available for rent.");
        }
    }

    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;
    }
}

```

MainApartmentSystem.java:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.sql.SQLOutput;
import java.util.List;

```

```

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

        String path =
"D:\\Github_Repositories\\spp_po9\\reports\\Solyshko\\3\\src\\apartments.txt
";

        // Загрузка данных из файла
        loadApartmentsFromFile(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("No available apartments to rent.");
        }

        // 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("Apartments with 2 rooms: " +
apartmentsByNumberOfRooms);

        // Список квартир с заданным числом комнат и этажем в заданном
промежутке
        List<Apartment> apartmentsByRoomsAndFloor =
system.getApartmentsByRoomsAndFloor(2, 1, 5);
        System.out.println("Apartments with 2 rooms and floor between 1 and
5: " + apartmentsByRoomsAndFloor);

        // Список квартир с площадью, превосходящей заданную
        List<Apartment> apartmentsByArea = system.getApartmentsByArea(80.0);
        System.out.println("Apartments with area greater than 80: " +
apartmentsByArea);
    }

    private static void loadApartmentsFromFile(String fileName,
ApartmentSystem 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());
}
}
}

```

apartments.txt:

```

PS D:\Github_Repositories\spp_po9\reports\Solyshko\3\src> cat .\apartments.txt
2, 75.5, 3, First Street 123, 1200.0
3, 90.0, 5, Oak Avenue 456, 1500.0
1, 50.0, 2, Maple Lane 789, 900.0
2, 80.0, 4, Pine Road 101, 1300.0
2, 80.0, 3, Pine Road 102, 1400.0
2, 80.0, 6, Pine Road 102, 1400.0

```

Результат работы программы:

Считываем из файла квартиры и выводим их на экран:

```

Available Apartments:
Apartment{Rooms=2, Area=75.5, floor=3, address='First Street 123', Price=1200.0, Сдаётся}
Apartment{Rooms=3, Area=90.0, floor=5, address='Oak Avenue 456', Price=1500.0, Сдаётся}
Apartment{Rooms=1, Area=50.0, floor=2, address='Maple Lane 789', Price=900.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=4, address='Pine Road 101', Price=1300.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=3, address='Pine Road 102', Price=1400.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=6, address='Pine Road 102', Price=1400.0, Сдаётся}

```

Успешно арендуем квартиру, предварительно создав список, который через метод `getAvailableApartments`, заполняем сдающимися квартирами.

```

Occupied Apartments:
Apartment rented successfully: Apartment{Rooms=2, Area=75.5, floor=3, address='First Street 123', Price=1200.0, Сдана}

```

Проверяем объект системы аренды квартир, в котором после аренды должно быть на 1 квартиру меньше квартир в списке сдающихся, и 1 квартира в списке сданных.

```

Available Apartments:
Apartment{Rooms=3, Area=90.0, floor=5, address='Oak Avenue 456', Price=1500.0, Сдаётся}
Apartment{Rooms=1, Area=50.0, floor=2, address='Maple Lane 789', Price=900.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=4, address='Pine Road 101', Price=1300.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=3, address='Pine Road 102', Price=1400.0, Сдаётся}
Apartment{Rooms=2, Area=80.0, floor=6, address='Pine Road 102', Price=1400.0, Сдаётся}
Occupied Apartments:
Apartment{Rooms=2, Area=75.5, floor=3, address='First Street 123', Price=1200.0, Сдана}

```

Теперь найдем по данным критериям доступные квартиры:

1. Количество комнат: 2
2. Этаж: 3
3. Площадь приближена к 80 (с разницей в 10)

```

Matching Apartment found: Apartment{Rooms=2, Area=80.0, floor=3, address='Pine Road 102', Price=1400.0, Сдаётся}

```

1. Количество комнат: 2
2. Этаж: 4
3. Площадь приближена к 75.0 (с разницей в 10)

```
Matching Apartment found: Apartment{Rooms=2, Area=80.0, floor=4, address='Pine Road 101', Price=1300.0, Сдаётся}
```

И далее найдем квартиры по последним пунктам лабораторной работы

- 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.

```
Apartments with 2 rooms: [Apartment{Rooms=2, Area=80.0, floor=4, address='Pine Road 101', Price=1300.0, Сдаётся}, Apartment{Rooms=2, Area=80.0, floor=3, address='Pine Road 102', Price=1400.0, Сдаётся}, Apartment{Rooms=2, Area=80.0, floor=6, address='Pine Road 102', Price=1400.0, Сдаётся}]
Apartments with 2 rooms and floor between 1 and 5: [Apartment{Rooms=2, Area=80.0, floor=4, address='Pine Road 101', Price=1300.0, Сдаётся}, Apartment{Rooms=2, Area=80.0, floor=3, address='Pine Road 102', Price=1400.0, Сдаётся}]
Apartments with area greater than 80: [Apartment{Rooms=3, Area=90.0, floor=5, address='Oak Avenue 456', Price=1500.0, Сдаётся}]
```

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