



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01 Информатика и вычислительная техника

МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/07 Интеллектуальные системы анализа,
обработки и интерпретации больших
данных

ОТЧЕТ

по лабораторной работе № 6

Название: Коллекции

Дисциплина: Языки программирования для работы с большими
данными

Студент ИУ6-22М

Д. Р. Григорян

Преподаватель

П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Цель работы:

Ознакомиться с языком программирования Java и научиться работать с коллекциями (Collections), стеком (Stack), HashSet, TreeMap, HashMap.

Выполнение:

Задача 1.1:

Ввести строки из файла, записать в список ArrayList. Выполнить сортировку строк, используя метод sort() из класса Collections.

Листинг 1 программы main:

```
import java.util.*;
import java.io.*;
//7. Ввести строки из файла, записать в список ArrayList.
//    Выполнить сортировку строк, используя метод sort() из класса
Collections.
public class Main {
    public static void main(String[] args) throws Exception{
        var fr = new FileReader("src/test.txt");
        var reader = new BufferedReader(fr);
        var line = reader.readLine();
        var list1 = new ArrayList<String>();
        while (line!= null) {
            list1.add(line);
            line = reader.readLine();
        }
        System.out.println("Unsorted list");
        for (String next: list1) {
            System.out.println(next);
        }
        Collections.sort(list1);
        System.out.println("-----");
        System.out.println("Sorted list");
        for (String next: list1) {
            System.out.println(next);
        }
    }
}
```

Результат приведен на рисунках 1-2:

```

C:\Users\sonar\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:
Unsorted list
My mother told me
Someday I will buy
Galleys with good oars
Sail to distant shores
My mother told me
Someday I will buy (buy)
Galleys with good oars
Sail to distant shores
Stand up on the prow
Noble barque I steer
(Steady) steady course to the haven
Hew many foe-men
Hew many foe-men
My mother told me
Someday I will buy
Galleys with good oars
Sail to distant shores
My mother told me
Someday I will buy (buy)
Galleys with good oars
Sail to distant shores
-----
Sorted list

```

Рисунок 1 – Результат выполнения программы

```

-----
Sorted list
(Steady) steady course to the haven
Galleys with good oars
Galleys with good oars
Galleys with good oars
Galleys with good oars
Hew many foe-men
Hew many foe-men
My mother told me
My mother told me
My mother told me
My mother told me
Noble barque I steer
Sail to distant shores
Sail to distant shores
Sail to distant shores
Sail to distant shores
Someday I will buy
Someday I will buy
Someday I will buy (buy)
Someday I will buy (buy)
Stand up on the prow

```

Рисунок 2 – Результат выполнения программы

Задача 1.2:

Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'. Проверить правильность расстановки скобок. Использовать стек.

Листинг 2 программы main:

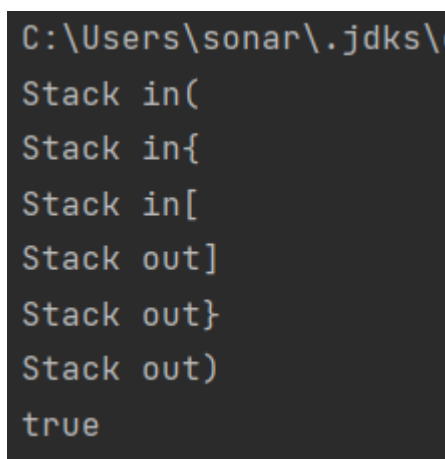
```
import java.util.*;
import java.util.Stack;

//8.  Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'.
//    Проверить правильность расстановки скобок. Использовать стек.
public class Main {

    public static boolean parsing(String st) {
        var symbols = st.toCharArray();
        HashMap<String, String> test = new HashMap<>();
        test.put("{","}");
        test.put("[","]");
        test.put("(" ,")");
        var list = new Stack<>();
        for (int i = 0; i < symbols.length; i++) {
            String symbol = st.substring(i, i + 1);
            if (test.containsKey(symbol)) {
                list.push(symbol);
                System.out.print("Stack in");
                System.out.println(symbol);
                continue;
            }
            else if (test.get(list.peek()).equals(symbol)) {
                System.out.print("Stack out");
                System.out.println(symbol);
                list.pop();
            }
        }
        if (!list.empty()) {
            System.out.println("Wrong");
            list.clear();
            return false;
        }
        else return true;
    }

    public static void main(String[] args) {
        var str = "({[]})";
        System.out.println(parsing(str));
    }
}
```

Результат приведен на рисунках 3-4:



```
C:\Users\sonar\.jdk\...
Stack in(
Stack in{
Stack in[
Stack out]
Stack out}
Stack out)
true
```

Рисунок 3 – Результат выполнения программы

```
Stack in(  
Stack in{  
Stack in[  
Stack out]  
Wrong  
false
```

Рисунок 4 – Результат выполнения программы

Задача 2.1:

На плоскости задано N отрезков. Найти точку пересечения двух отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.

Листинг 3 программы main:

```
import java.util.*;  
import java.util.TreeMap;  
//7. На плоскости задано N отрезков.  
// Найти точку пересечения двх отрезков, имеющую минимальную абсциссу.  
Использовать класс TreeMap  
public class Main{  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        System.out.println("Input the N lines");  
        int n = scanner.nextInt();  
        System.out.println("Input the coordinates of lines");  
        List<Line> lines = new ArrayList<>();  
        for (int i = 0; i < n; i++) {  
            double x1 = scanner.nextDouble();  
            double y1 = scanner.nextDouble();  
            double x2 = scanner.nextDouble();  
            double y2 = scanner.nextDouble();  
            lines.add(new Line(new Point(x1, y1), new Point(x2, y2)));  
        }  
        TreeMap<Point, Line[]> treeMap = new TreeMap<>();  
        for (int i = 0; i < n; i++) {  
            for (int j = i + 1; j < n; j++) {  
                Line l1 = lines.get(i);  
                Line l2 = lines.get(j);  
                Point p = l1.getIntersection(l2);  
                if (p != null) {  
                    Line[] arr = {l1, l2};  
                    treeMap.put(p, arr);  
                }  
            }  
        }  
        if (treeMap.isEmpty()) {  
            System.out.println("There is not intersection point");  
        }  
        else {  
            Point point = treeMap.firstKey();  
            Line[] linesArr = treeMap.get(point);  
            Line l1 = linesArr[0];  
            Line l2 = linesArr[1];  
            Point intersection = l1.getIntersection(l2);
```

```

        System.out.println(intersection.x + " " + intersection.y);
    }
}

```

Листинг 4 класса Line:

```

public class Line {
    Point p1, p2;
    Line(Point p1, Point p2) {
        this.p1 = p1;
        this.p2 = p2;
    }
    Point getIntersection(Line l) {
        double x1 = p1.x, y1 = p1.y;
        double x2 = p2.x, y2 = p2.y;
        double x3 = l.p1.x, y3 = l.p1.y;
        double x4 = l.p2.x, y4 = l.p2.y;
        double d = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
        if (d == 0) {
            return null;
        }
        double xi = ((x3 - x4) * (x1 * y2 - y1 * x2) - (x1 - x2) * (x3 * y4 -
y3 * x4)) / d;
        double yi = ((y3 - y4) * (x1 * y2 - y1 * x2) - (y1 - y2) * (x3 * y4 -
y3 * x4)) / d;
        if (xi < Math.min(x1, x2) || xi > Math.max(x1, x2)) {
            return null;
        }
        if (xi < Math.min(x3, x4) || xi > Math.max(x3, x4)) {
            return null;
        }
        return new Point(xi, yi);
    }
}

```

Листинг 5 класса Point:

```

public class Point implements Comparable<Point> {
    double x, y;
    Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
    @Override
    public int compareTo(Point o) {
        return Double.compare(x, o.x);
    }
}

```

Результат приведен на рисунке 5:

```
Input the N lines
2
Input the coordinates of lines
2
3
4
5
6
4
2
3
2.0 3.0
```

Рисунок 5 – Результат выполнения программы

Задача 2.2:

Выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Задание: Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов. Вывести: а) список автомобилей заданной марки; б) список автомобилей заданной модели, которые эксплуатируются больше n лет; с) список автомобилей заданного года выпуска, цена которых больше указанной.

Листинг 6 класса FigureFinder:

```
import java.util.HashSet;

public class FigureFinder {

    private int[][] grid;
    private int rows;
    private int cols;
    private HashSet<HashSet<String>> figures;

    public FigureFinder(int[][] grid) {
        this.grid = grid;
        this.rows = grid.length;
        this.cols = grid[0].length;
        this.figures = new HashSet<>();
    }

    public HashSet<HashSet<String>> findFigures() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
```

```

        if (grid[i][j] == 1) {
            HashSet<String> figure = new HashSet<>();
            dfs(i, j, figure);
            if (!isFigureDuplicate(figure)) {

                figures.add(figure);
            }
        }
    }
    return figures;
}

private void dfs(int i, int j, HashSet<String> figure) {
    if (i < 0 || i >= rows || j < 0 || j >= cols || grid[i][j] == 0 ||
figure.contains(i + "," + j)) {
        return;
    }
    figure.add(i + "," + j);
    dfs(i - 1, j, figure);
    dfs(i + 1, j, figure);
    dfs(i, j - 1, figure);
    dfs(i, j + 1, figure);
}

private boolean isFigureDuplicate(HashSet<String> figure) {
    for (HashSet<String> existingFigure : figures) {
        if (existingFigure.containsAll(figure)) {
            return true;
        }
    }
    return false;
}
}

```

Листинг 7 программы main:

```

import java.util.HashSet;

//8. На клетчатом листе бумаги закрашена часть клеток. Выделить все
различные фигуры, которые образовались при этом.
// Фигурой считается набор закрашенных клеток, достижимых друг из
друга при движении в четырёх направлениях.
// Две фигуры являются различными, если их нельзя совместить поворотом
на угол, кратный 90 градусам,
// и параллельным переносом. Используйте класс HashSet.

public class Main {
    public static void main(String[] args) {
        int[][] grid = {
            {0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0},
            {0,0,0,1,1,0,0,0,0,0},
            {0,0,0,1,1,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,1,1,0,0},
            {0,0,0,0,0,0,1,0,0,0},
            {0,0,0,0,0,0,0,0,0,0}
        };
    }
}

```



```
var figureFinder = new FigureFinder(grid);  
HashSet<HashSet<String>> figures = figureFinder.findFigures();  
System.out.println(figures.size());  
}  
}
```

Результат приведен на рисунке 6:

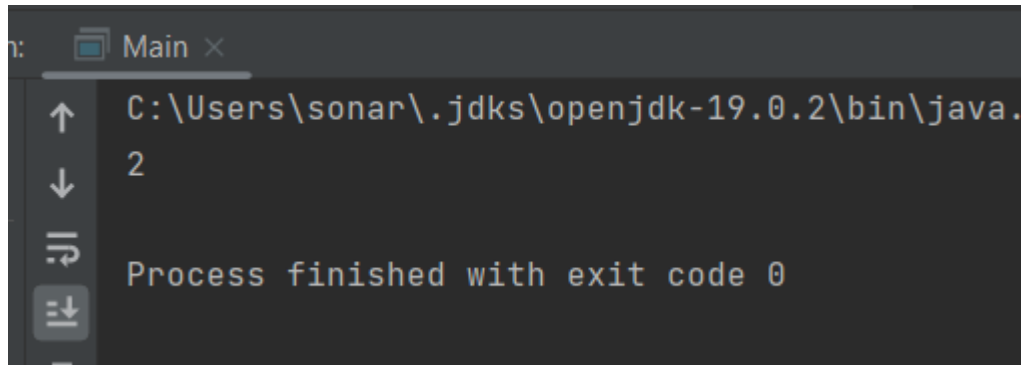


Рисунок 6 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы были написаны программы согласно выданному заданию, используя коллекции (Colletctions), стек (Stack), HashSet, TreeMap, HashMap, а также их методы и функции.