



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 09.04.01/12 Интеллектуальный анализ больших
данных в системах поддержки принятия решений.

О Т Ч Е Т
по лабораторной работе № 6
Вариант № 7

Название: коллекции

Дисциплина: языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

Ф.А. Лучкин

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

Цель: освоить принципы работы с коллекциями на языке программирования Java.

Задание 1: 7. Ввести строки из файла, записать в список ArrayList. Выполнить сортировку строк, используя метод sort() из класса Collections.

Main:

```
package lab6_var1_7;

import java.io.*;
import java.util.ArrayList;
import java.util.Collections;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> lines = new ArrayList<>();

        try {
            BufferedReader reader = new BufferedReader(new
            FileReader("src/lab6_var1_7/input.txt"));
            String line;

            while ((line = reader.readLine()) != null) {
                lines.add(line);
            }

            reader.close();
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }

        Collections.sort(lines);

        for (String sortedLine : lines) {
            System.out.println(sortedLine);
        }
    }
}
```

input.txt:

```
cherry
banana
grape
date
apple
```

Работа программы показана на рисунке 1.

```
apple
banana
cherry
date
grape
```

Рисунок 1 – Работа программы

Задание 2: Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'. Проверить правильность расстановки скобок. Использовать стек.

Код класса Main:

```
package lab6_var1_8;

import java.util.Stack;

public class Main {
    public static boolean checkBrackets(String input) {
        Stack<Character> stack = new Stack<>();

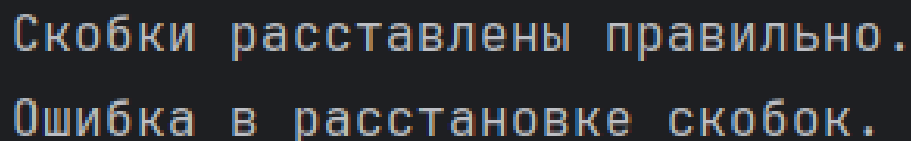
        for (char c : input.toCharArray()) {
            if (c == '(' || c == '[' || c == '{') {
                stack.push(c);
            } else if (c == ')' && (stack.isEmpty() || stack.pop() != '(')) {
                return false;
            } else if (c == ']' && (stack.isEmpty() || stack.pop() != '[')) {
                return false;
            } else if (c == '}' && (stack.isEmpty() || stack.pop() != '{')) {
                return false;
            }
        }

        return stack.isEmpty();
    }

    public static void main(String[] args) {
        String input = "([{}()])";
        if (checkBrackets(input)) {
            System.out.println("Скобки расставлены правильно.");
        } else {
            System.out.println("Ошибка в расстановке скобок.");
        }

        input = "{([{}()])}";
        if (checkBrackets(input)) {
            System.out.println("Скобки расставлены правильно.");
        } else {
            System.out.println("Ошибка в расстановке скобок.");
        }
    }
}
```

Работа программы показана на рисунке 2.



Скобки расставлены правильно.
Ошибка в расстановке скобок.

Рисунок 2 – Работа программы

Задание 3: на плоскости задано N отрезков. Найти точку пересечения двух отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.

Код класса Segment:

```
package lab6_var2_7;

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

class Segment {
    double x1, x2, y1, y2;

    public Segment(double x1, double x2, double y1, double y2) {
        this.x1 = x1;
        this.x2 = x2;
        this.y1 = y1;
        this.y2 = y2;
    }

    public List<double[]> getIntersectionPoints(Segment other) {
        List<double[]> intersections = new ArrayList<>();

        double x1 = this.x1, y1 = this.y1; // Координаты первой точки первого
отрезка
        double x2 = this.x2, y2 = this.y2; // Координаты второй точки первого
отрезка
        double x3 = other.x1, y3 = other.y1; // Координаты первой точки второго
отрезка
        double x4 = other.x2, y4 = other.y2; // Координаты второй точки второго
отрезка

        double den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);

        if (den != 0) {
            double t = ((x1 - x3) * (y3 - y4) - (y1 - y3) * (x3 - x4)) / den;
            double u = -((x1 - x2) * (y1 - y3) - (y1 - y2) * (x1 - x3)) / den;

            if (t >= 0 && t <= 1 && u >= 0 && u <= 1) {
                double intersectX = x1 + t * (x2 - x1);
                double intersectY = y1 + t * (y2 - y1);
                intersections.add(new double[]{intersectX, intersectY});
            } else {
                System.out.println("Отрезки не пересекаются");
            }
        } else {
            System.out.println("Отрезки параллельны");
        }

        return intersections;
    }
}
```

Код класса Main:

```
package lab6_var2_7;

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

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

```

List<Segment> segments = new ArrayList<>();
segments.add(new Segment(2, 4, 3, 5));
segments.add(new Segment(1, 4, 4, 1));
segments.add(new Segment(4, 4, 1, 5));

TreeMap<Double, double[]> intersectionPointsWithMinAbscissa = new
TreeMap<>();

for (int i = 0; i < segments.size(); i++) {
    Segment s1 = segments.get(i);
    for (int j = i + 1; j < segments.size(); j++) {
        Segment s2 = segments.get(j);
        List<double[]> intersectionPoints = s1.getIntersectionPoints(s2);
        for (double[] point : intersectionPoints) {
            double x = point[0];
            double y = point[1];
            intersectionPointsWithMinAbscissa.put(x, new double[]{x, y});
            System.out.println("Точка пересечения: (" + x + ", " + y +
")");
        }
    }
}

if (!intersectionPointsWithMinAbscissa.isEmpty()) {
    double[] minPoint =
intersectionPointsWithMinAbscissa.firstEntry().getValue();
    System.out.println("Точка пересечения с минимальной абсциссой: (" +
minPoint[0] + ", " + minPoint[1] + ")");
} else {
    System.out.println("Нет точек пересечения");
}
}
}

```

Работа программы показана на рисунке 3.

```

Точка пересечения: (2.0, 3.0)
Точка пересечения: (4.0, 5.0)
Точка пересечения: (4.0, 1.0)
Точка пересечения с минимальной абсциссой: (2.0, 3.0)

```

Рисунок 3 – Работа программы

Задание 4: 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. Код класса Main:

```

package lab6_var2_8;

import java.util.HashSet;

```

```

public class Main {
    public static void main(String[] args) {
        char[][] grid = {
            {'*', '*', '.', '.', '.'},
            {'*', '.', '.', '*', '.'},
            {'.', '.', '*', '.', '.'},
            {'.', '*', '*', '*', '*'}
        };

        HashSet<String> figures = new HashSet<>();

        int rows = grid.length;
        int cols = grid[0].length;

        boolean[][] visited = new boolean[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (grid[i][j] == '*' && !visited[i][j]) {
                    StringBuilder figure = new StringBuilder();
                    explore(grid, visited, i, j, figure);
                    figures.add(figure.toString());
                }
            }
        }

        System.out.println("Различные фигуры на клетчатом листе бумаги:");
        for (String figure : figures) {
            System.out.println(figure);
        }

        private static void explore(char[][] grid, boolean[][] visited, int row, int
col, StringBuilder figure) {
            if (row < 0 || col < 0 || row >= grid.length || col >= grid[0].length ||
grid[row][col] == '.' || visited[row][col]) {
                return;
            }

            visited[row][col] = true;
            figure.append(row).append(",").append(col).append(";");

            explore(grid, visited, row + 1, col, figure);
            explore(grid, visited, row - 1, col, figure);
            explore(grid, visited, row, col + 1, figure);
            explore(grid, visited, row, col - 1, figure);
        }
    }
}

```

Работа программы показана на рисунке 4.

```

Различные фигуры на клетчатом листе бумаги:
0,0;1,0;0,1;
2,2;3,2;3,3;3,4;3,1;
1,3;

```

Рисунок 4 – Работа программы

Ссылка на git-репозиторий: https://github.com/FedorLuchkin/Java_bmstu

Вывод: были освоены принципы работы с коллекциями на языке программирования Java.