**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, искусственный интеллект и системы управления**

КАФЕДРА **Компьютерные системы и сети**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 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 (Группа) _____ (Подпись, дата) М.Э.Хабаров (И.О. Фамилия)

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

*2024 г*

**Цель:** изучить работу с коллекциями на языке Java.

## Задание №1

Формулировка задания и код программы представлены в листинге 1:

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

/*
    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) {
        try {
            File inputFile = new File("src/input.txt");
            Scanner scanner = new Scanner(inputFile);

            ArrayList<String> lines = new ArrayList<>();
            while (scanner.hasNextLine()) {
                lines.add(scanner.nextLine());
            }

            scanner.close();

            Collections.sort(lines);

            for (String line : lines) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("Ошибка при обработке файла: " +
e.getMessage());
        }
    }
}
```

## Задание №2

Формулировка задания и код программы представлены в листинге 2:

```
import java.util.Stack;

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

public class Main {
    public static void main(String[] args) {
        String input = "([]{()})"; // Заданная строка
        if (checkBrackets(input)) {
            System.out.println("Скобки расставлены правильно.");
        } else {
            System.out.println("Скобки расставлены неправильно.");
        }
    }
```

```
    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.peek() == '(') {
                stack.pop();
            } else if (c == ']' && !stack.isEmpty() && stack.peek() == '[') {
                stack.pop();
            } else if (c == '}' && !stack.isEmpty() && stack.peek() == '{') {
                stack.pop();
            } else {
                return false; // Неправильное сочетание скобок
            }
        }
        return stack.isEmpty(); // Все скобки должны быть закрыты
    }
}
```

## Задание №3

Формулировка задания и код программы представлены в листинге 3:

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.TreeMap;

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

class Point {
    double x;
    double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}

class Line {
    Point start;
    Point end;

    public Line(Point start, Point end) {
        this.start = start;
        this.end = end;
    }

    public double slope() {
        return (end.y - start.y) / (end.x - start.x);
    }

    public double intercept() {
        return start.y - slope() * start.x;
    }

    public Point intersection(Line other) {
        double x = (other.intercept() - this.intercept()) / (this.slope() -
```

```
other.slope());
        double y = this.slope() * x + this.intercept();
        return new Point(x, y);
    }
}

public class Main {

    public static void main(String[] args) {
        Line[] lines = {
                new Line(new Point(1, 1), new Point(3, 2)),
                new Line(new Point(2, 3), new Point(4, 1)),
                new Line(new Point(0, 4), new Point(5, 0))
        };

        TreeMap<Double, Point> intersections = new TreeMap<>();

        for (int i = 0; i < lines.length; i++) {
            for (int j = i + 1; j < lines.length; j++) {
                Point intersection = lines[i].intersection(lines[j]);
                intersections.put(intersection.x, intersection);
            }
        }

        try (FileWriter writer = new FileWriter("output.txt")) {
            writer.write("Point of intersection with minimum abscissa: (" +
intersections.firstEntry().getValue().x + ", " +
intersections.firstEntry().getValue().y + ")\n");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the
file.");
            e.printStackTrace();
        }
    }
}
```

## Задание №4

Формулировка задания и код программы представлены в листинге 4:

```
import java.util.Stack;

/*
    Дана матрица из целых чисел. Найти в ней прямоугольную подматрицу,
состоящую из
    максимального количества одинаковых элементов. Использовать класс Stack.
 */

public class Main {

    public static void main(String[] args) {
        int[][] matrix = {
                {1, 1, 2, 2},
                {2, 2, 2, 2},
                {2, 2, 2, 2}
        };

        int maxCount = 0;

        for (int i = 0; i < matrix.length; i++) {
            int[] hist = new int[matrix[i].length];
            for (int j = i; j < matrix.length; j++) {
```

```
                for (int k = 0; k < matrix[j].length; k++) {
                    if (matrix[j][k] == matrix[i][k]) {
                        hist[k]++;
                    } else {
                        hist[k] = 0;
                    }
                }

                Stack<Integer> stack = new Stack<>();
                int count = 0;
                for (int h : hist) {
                    while (!stack.isEmpty() && hist[stack.peek()] >= h) {
                        int height = hist[stack.pop()];
                        int width = stack.isEmpty() ? count : count -
stack.peek() - 1;
                        maxCount = Math.max(maxCount, height * width);
                    }
                    stack.push(count++);
                }
                while (!stack.isEmpty()) {
                    int height = hist[stack.pop()];
                    int width = stack.isEmpty() ? count : count -
stack.peek() - 1;
                    maxCount = Math.max(maxCount, height * width);
                }
            }
        }

        System.out.println("Максимальное количество одинаковых элементов: " +
maxCount);
    }
}
```

**Вывод:** в результате выполнения лабораторной работы были освоены основные принципы коллекций на языке Java.