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 «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №5

по курсу «Языки и методы программирования»

«Монады в языке Java»

«Вариант 31»

Студент группы ИУ9-21Б: Пенкин А. Д.

Преподаватель: Посевин Д. П.

Москва 2023

1 Цель

Приобретение навыков использования монад Optional и Stream в программах на языке Java.

2 Условие

Во время выполнения лабораторной работы требуется разработать на языке Java Булевскую матрицу размером m × n, где 1 <= m, n <= 8, с операциями:

1. порождение потока сумм элементов строк по модулю 2 (т.е., исключающее ИЛИ);

2. поиск строки, в которой единиц больше, чем во всех остальных строках вместе взятых. Элементы матрицы должны быть представлены битами в числе типа long.

Проверить работу первой операции нужно путём подсчёта количеств вхождений каждого из присутствующих в последовательности чисел.

3 Код решения

1. Test.java

```
public class Test {
    public static void main(String[] args) {
        MatrixBool A = new MatrixBool(new int[][]{
                {1, 1, 1, 0},
                {1, 0, 1, 0},
                {0, 1, 1, 1},
                {0, 1, 1, 0},
                {0, 0, 0, 1}});
        MatrixBool B = new MatrixBool(new int[][]{
                {1, 1, 1, 1, 1, 1, 0, 1},
                {0, 0, 0, 1, 0, 0, 0, 1},
                {0, 0, 0, 0, 0, 0, 1, 0},
                {0, 0, 1, 0, 0, 0, 0, 0}});

        System.out.println("Создадим матрицу 1:");
        A.printMatrix();
        System.out.println("");
        System.out.println("Создадим матрицу 2:");
        B.printMatrix();
        System.out.println("");
```

```
        System.out.println("Выведем через поток исключающее или");
        System.out.println("для каждой из строк матрицы 1 и 2:");
        System.out.print("**");
        A.xorStream().forEach(x -> System.out.print(" " + x + " "));
        System.out.println("**");
        System.out.print("**");
        B.xorStream().forEach(x -> System.out.print(" " + x + " "));
        System.out.println("**");

        System.out.println("Посчитаем количество \"1\" и \"0\" в этих потоках:");
        System.out.print("\"1\" - " + A.xorStream().filter(x -> x == 1).count());
        System.out.println("; \"0\" - " + A.xorStream().filter(x -> x == 0).count());
        System.out.print("\"1\" - " + B.xorStream().filter(x -> x == 1).count());
        System.out.println("; \"0\" - " + B.xorStream().filter(x -> x == 0).count() + "\n");

        System.out.println("Найдём строку, матрицы 1, в которой \"1\"");
        System.out.println("больше чем в остальных строках вместе взятых:");
        if (A.getBestString().isPresent()) {
            System.out.println(A.getBestString().get().getString() + "\n");
        }
        else {
            System.out.println("такой строки не существует.\n");
        }

        System.out.println("Найдём строку, матрицы 2, в которой \"1\"");
        System.out.println("больше чем в остальных строках вместе взятых:");
        if (B.getBestString().isPresent()) {
            System.out.println(B.getBestString().get().getString() + "\n");
        }
        else {
            System.out.println("такой строки не существует.");
        }
    }
}
```

2. MatrixBool.java

```
import java.util.Arrays;
import java.util.*;
import java.util.stream.Stream;
```

```
public class MatrixBool {
    private ArrayList<StringMatrix> A;
    long n;
    public MatrixBool(int[][] A1) {
        A = new ArrayList<>(1);
        Arrays.stream(A1)
                .map(x -> new StringMatrix(x))
                .forEach(x -> A.add(x));
        n = A.stream()
                .map(x -> x.getSumBinary())
                .reduce((x, y) -> x + y).get();
    }

    public StringMatrix getStringMatrix(int i){
        return A.get(i);
    }

    public long getN() {
        return n;
    }

    public void printMatrix(){
        A.stream().map(x -> x.getString())
                .map(x -> x.split(""))
                .forEach(x -> {
                    Arrays.stream(x)
                            .forEach(y -> System.out.print(y + " "));
                    System.out.println("");
                });
    }

    public Stream<Long> xorStream() {
        ArrayList<Long> result = new ArrayList<>();
        A.stream()
                .map(x -> x.getSumBinary() % 2)
                .forEach(x -> result.add(x));
        return result.stream();
    }
```

```
    public Optional<StringMatrix> getBestString() {
        Optional<StringMatrix> result = Optional.empty();
        Optional<StringMatrix> tmp = A
                .stream()
                .filter(x -> x.getSumBinary() > n - x.getSumBinary())
                .findFirst();
        if (tmp.isPresent()) {
            result = Optional.ofNullable(tmp.get());
        }
        return result;
    }
}
```

3. StringMatrix.java

```
import java.util.Arrays;

public class StringMatrix {
    private long str, sumBinary;
    int count;
    public StringMatrix(int[] a) {
        str = 0;
        Arrays.stream(a).forEach(y-> {
            str = (str << 1) + y;
            sumBinary += y;});
        count = a.length;
    }

    public long getSumBinary() {
        return sumBinary;
    }

    public long getStr() {
        return str;
    }
    public String getString(){
        String a = Long.toBinaryString(str);
        if (a.length() < count){
            a = "0".repeat(count - a.length()) + a;
```

```
        }
        return a;
    }
}
```

4 Результаты работы программы

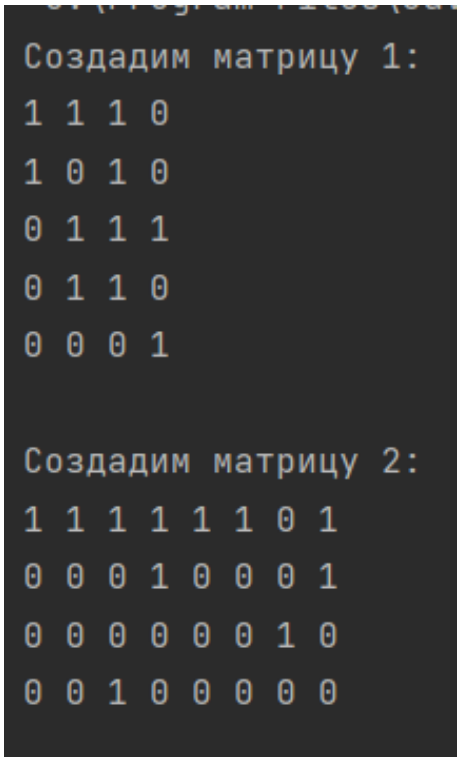

Рис. 1 — создание матриц и их вывод

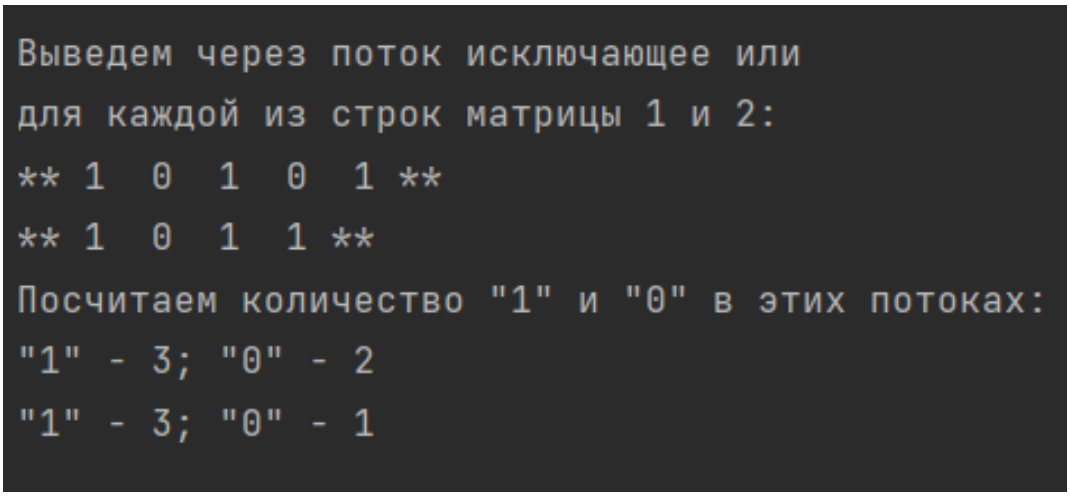

Рис. 2 — работа потока исключающего или

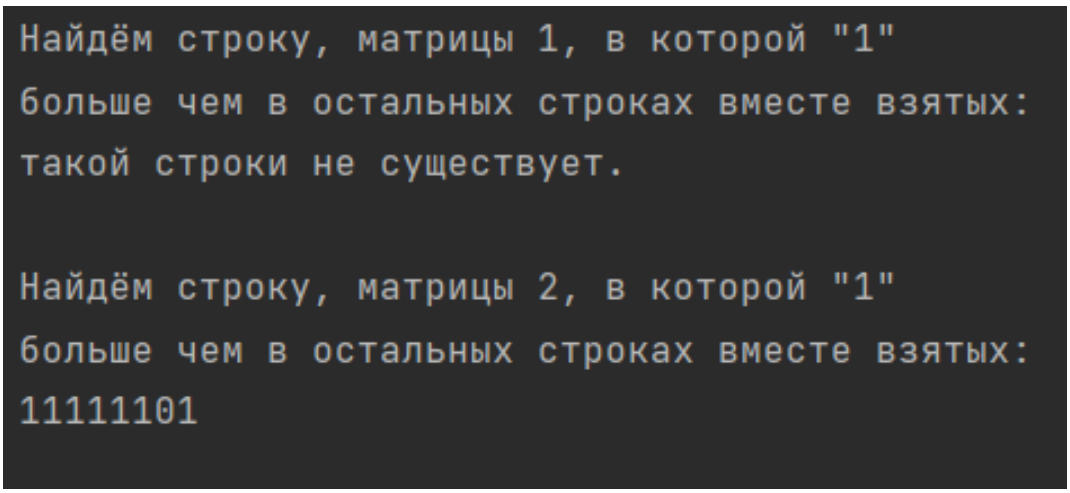

Рис. 3 — поиск строки с наибольшим количеством единиц