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»

Студент группы ИУ9-22Б Тараканов В. Д.

Преподаватель Посевин Д. П.

Москва 2024

1 Задание

Последовательность простых дробей с операциями: 1. порождение потока квадратных уравнений с целыми коэффициентами, решениями которых являются пары соседних дробей последовательности (вспомнить теорему Виета); 2. поиск максимального дискриминанта квадратного уравнения, решениями которого является пара соседних дробей последовательности. Проверить работу первой операции нужно путём ранжирования квадратных уравнений по сумме их коэффициентов.

2 Результаты

Листинг 1 — Класс Fraction

```
1     public class Fraction {
2       int chisl;
3       int znam;
4       public Fraction(int a, int b){
5         this.chisl = a;
6         this.znam = b;
7       }
8     }
9
10
11
```

Листинг 2 — Класс SumComparator

```
import java.util.*;

class SumComparator implements Comparator<ArrayList<Integer>> {
  public int compare(ArrayList<Integer> a, ArrayList<Integer> b) {
    int a0 = 0, b0 = 0;
    for (int elem: a){
      a0+=elem;
    }
    for (int elem: b){
      b0+=elem;
    }
    if (a0 > b0) {
      return 1;
    }
    if (a0 == b0) {
      return 0;
    }
    return -1;
  }
}

```

Результат запуска представлен на рисунке 1.

Листинг 3 — Класс QuadraticEquation, в котором реализована программа по заданию

```
1     import java.util.ArrayList;
2     import java.util.Arrays;
3     import java.util.HashMap;
4     import java.util.Optional;
5     import java.util.stream.Stream;
6
7     public class QuadraticEquation {
8
9       private HashMap<Integer, ArrayList<Fraction>> table;
10
11      public QuadraticEquation(ArrayList<Fraction> fractions) {
12        this.table = new HashMap<>();
13        for (int i = 0; i < fractions.size() - 1; i++) {
14          ArrayList<Fraction> pair = new ArrayList<>();
15          pair.add(fractions.get(i));
16          pair.add(fractions.get(i + 1));
17          this.table.put(i, pair);
18        }
19      }
20
21      int getDisc(ArrayList<Integer> koefs){
22        int disc = koefs.get(1)*koefs.get(1) - 4* koefs.get(0)*koefs.get
    (2);
23        return disc;
24      }
25
26      public int gcd(int a, int b) {
27        if (a<0){
28          a = -a;
29        }
30        if (b<0){
31          b = -b;
32        }
33        while (b != 0) {
34          int tmp = a % b;
35          a = b;
36          b = tmp;
37        }
38        return a;
39      }
40
41      public ArrayList<Integer> makeKoefs(ArrayList<Fraction> pair) {
42        ArrayList<Integer> res = new ArrayList<>();
43        int a = pair.get(0).znam * pair.get(1).znam;
44        int b = -(pair.get(0).chisl*pair.get(1).znam + pair.get(1).chisl
    *pair.get(0).znam);
45        int c = pair.get(0).chisl * pair.get(1).chisl;
46        int gcd = gcd(a,gcd(b,c));
47        a/=gcd;
48        b/=gcd;
49        c/=gcd;
50        res.add(a);
51        res.add(b);
52        res.add(c);
53        return res;
54      }
55
```

Листинг 4 — Класс QuadraticEquation (продолжение)

```
1       public Stream<ArrayList<Integer>> generateKoefs(){
2         ArrayList<ArrayList<Integer>> result = new ArrayList<>();
3         this.table.entrySet().stream().forEach(x->result.add(makeKoefs(x
    .getValue())));
4         return result.stream();
5       }
6       public Stream<Integer> generateDics(){
7         Stream<ArrayList<Integer>> koefs = generateKoefs();
8         ArrayList<Integer> res = new ArrayList<>();
9         koefs.forEach(x-> res.add(getDisc(x)));
10        return res.stream();
11      }
12
13      public Optional<Integer> getMaxDics(){
14        Stream<Integer> disc = generateDics();
15        return disc.max(Integer::compareTo);
16      }
17    }
18
19
```

Листинг 5 — Класс Main, в котором реализована проверка работы класса QuadraticEquation

```
1       import java.util.ArrayList;
2       import java.util.Optional;
3
4
5       public class Main {
6         public static void main(String[] args) {
7           Fraction a = new Fraction(1, 2);
8           Fraction b = new Fraction(7, 3);
9           Fraction c = new Fraction(5, 4);
10          ArrayList<Fraction> posl = new ArrayList<>();
11          posl.add(c);
12          posl.add(b);
13          posl.add(a);
14          QuadraticEquation equation = new QuadraticEquation(posl);
15          equation.generateKoefs().sorted(new SumComparator()).forEach(
    System.out::println);
16          Optional<Integer> maxNumber = equation.getMaxDics();
17          maxNumber.ifPresentOrElse(
18          num->System.out.println("Max discriminant among all: " + num),
19          () -> System.out.println("Number doesn't exist!!!")
20          );
21        }
22      }
23
24      }
25
```

```
[6, -17, 7]
[12, -43, 35]
Max discriminant among all: 169
```

Рис. 1 — Результат