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 Федуков А. А.

Преподаватель Посевин Д. П.

10 апреля 2024 г.

Цель работы

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

Задание

Во время выполнения лабораторной работы требуется разработать на языке Java один из классов, перечисленных в таблице, которая приведена ниже. В каждом классе нужно реализовать по крайней мере два метода: первый метод должен возвращать Stream, а второй – Optional. Операции, выполняемые каждым методом, указаны в вариантах задания. В методе main вспомогательного класса Test нужно продемонстрировать работоспособность разработанного класса, осуществив группировку содержимого потока, возвращаемого первым методом, с помощью группирующего коллектора. В исходном коде (включая класс Test) запрещено использовать циклы и рекурсию.

Задание 1

Последовательность точек в трёхмерном пространстве с операциями: 1. порождение потока отрезков, соединяющих соседние точки и пересекающих плоскость Oxy. 2. вычисление отношения количества точек, попадающих в шар с центром в начале координат и радиусом r, к количеству точек, которые в этот шар не попадают. Проверить работу первой операции нужно путём подсчёта количества отрезков, пересекающих плоскость Oxy в каждой из её четвертей.

Реализация

Я описал классы Point, Line, Space, конструирующиеся друг из друга последовательно, и реализовал необходимые операции, исходя из условия задания, в файле Space.java
Сгенерировал их экземпляры и проверил работоспособность уже в Test.java

Код

Листинг 1: Файл Space.java

```
1  package task1;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Optional;
6  import java.util.stream.Collectors;
7  import java.util.stream.Stream;
8
9  import static java.lang.Math.*;
10
11 class Point implements Comparable<Point>{
12     public double id, x, y, z;
13     Point(double id, double x, double y, double z){
14         this.id = id;
15         this.x = x;
16         this.y = y;
17         this.z = z;
18     }
19
20     @Override
21     public int compareTo(Point o) {
22         return (int) (o.x - this.x + o.y - this.y + o.z - this.z + o.id
       - this.id);
23     }
24
25     @Override
26     public String toString() {
27         return "task1.Point{" +
28                 "id=" + id +
29                 ", x=" + x +
30                 ", y=" + y +
31                 ", z=" + z +
32                 '}';
33     }
34 }
35 class Line implements Comparable<Line>{
36     public Point start, end, crossOXY;
37     public double length;
38     Line(Point start, Point end){
39         this.start = start;
40         this.end = end;
41         this.length = sqrt(pow(end.x - start.x, 2) + pow(end.y - start.y
       , 2) + pow(end.z - start.z, 2));
```

```
42          this.crossOXY = getOXY();
43      }
44      private Point getOXY(){
45          double t = (double) start.z /(start.z - end.z);
46          double x = (end.x - start.x)*t + start.x;
47          double y = (end.y - start.y)*t + start.y;
48          return new Point(-1, x, y, 0);
49      }
50      public boolean getQuarter(int t){
51          if (crossOXY.x >= 0 & crossOXY.y >= 0 & t == 1){
52              return true;
53          } else if (crossOXY.x < 0 & crossOXY.y >= 0 & t == 2) {
54              return true;
55          } else if (crossOXY.x < 0 & crossOXY.y < 0 & t == 3) {
56              return true;
57          }
58          else if (crossOXY.x >= 0 & crossOXY.y < 0 & t == 4) {
59              return true;
60          }
61          else return false;
62      }
63
64      @Override
65      public int compareTo(Line o) {
66          if (this.start == o.start & this.end == o.end){
67              return 0;
68          }
69          if (this.start.x > o.start.x){
70              return 1;
71          }
72          else {
73              return -1;
74          }
75      }
76
77      @Override
78      public String toString() {
79          return "task1.Line{" +
80                  "start=" + start +
81                  ", end=" + end +
82                  ", crossOXY=" + crossOXY +
83                  ", length=" + length +
84                  '}';
85      }
86  }
87
```

```
88
89 public class Space {
90     ArrayList<Point> Points;
91     int count = 0;
92     Space(){
93         Points = new ArrayList<>();
94     }
95     public void add(int x, int y, int z){
96         Points.add(new Point(count, x, y, z));
97         count += 1;
98     }
99     public Point getPoint(int id){
100        return Points.stream().filter(p -> p.id == id).findFirst().get()
       ;
101    }
102
103    public boolean isNeighbour(Line l, int neighbour_distance){
104        return l.start.id < l.end.id & l.length <= neighbour_distance;
105    }
106    public boolean crossOXY(Line l){
107        return l.start.z >= 0 & l.end.z <= 0 || l.end.z >= 0 & l.start.z
       <= 0;
108    }
109
110    public Stream<Line> linesStream(int neighbour_distance) {
111        ArrayList<Line> result = new ArrayList<>();
112        Points.stream()
113                .forEach(p1 -> Points.stream()
114                        .forEach(p2 -> result.add(new Line(p1, p2))));
115        return result.stream().filter(this::crossOXY).filter(line ->
       isNeighbour(line, neighbour_distance));
116    }
117    public boolean isInside(Point p, int r){
118        return abs(p.x) <= r & abs(p.y) <= r & abs(p.z) <= r;
119    }
120    public Optional<List<Point>> getPointsInside(int r) {
121        Optional<List<Point>> result = Optional.of(Points.stream().
       filter(point -> isInside(point, r)).collect(Collectors.toList()));
122        return result;
123    }
124    public Optional<List<Point>> getPointsOutside(int r) {
125        Optional<List<Point>> result = Optional.of(Points.stream().
       filter(point -> !isInside(point, r)).collect(Collectors.toList()));
126        return result;
127    }
128 }
```

Листинг 2: Файл Test.java

```
1  package task1;
2
3  public class Test {
4      public static void main(String[] args) {
5          Space space = new Space();
6          int neighbour_distance = 10;
7          int r = 1;
8
9          space.add(1, 1, 1);
10 //         space.add(2, 10, 10);
11         space.add(-1, -1, 0);
12         space.add(0, -2, 1);
13         space.add(-3, 0, 1);
14         space.add(0, 1, -1);
15
16         System.out.println("    All the lines:");
17         space.linesStream(neighbour_distance).forEach(System.out::
       println);
18
19         System.out.println("    All the quarters of crossOXY:");
20         System.out.println("I: " + space.linesStream(neighbour_distance)
       .filter(line -> line.getQuarter(1)).count());
21         System.out.println("II: " + space.linesStream(neighbour_distance
       ).filter(line -> line.getQuarter(2)).count());
22         System.out.println("III: " + space.linesStream(
       neighbour_distance).filter(line -> line.getQuarter(3)).count());
23         System.out.println("IV: " + space.linesStream(neighbour_distance
       ).filter(line -> line.getQuarter(4)).count());
24
25         System.out.println("    All the points inside/outside sphere:");
26         space.getPointsInside(r).ifPresent(System.out::println);
27         space.getPointsOutside(r).ifPresent(System.out::println);
28         if (space.getPointsInside(r).isPresent() & space.
       getPointsOutside(r).isPresent()
29                 & !space.getPointsOutside(r).get().isEmpty()){
30             System.out.println((double) space.getPointsInside(r).get().
       size() / space.getPointsOutside(r).get().size());
31         }
32
33     }
34
35 }
```

Вывод программы

Программа создала пространтво и поместило туда точки, затем по этим точкам были созданы линии, с помощью которых я посчитал и вывел необходимую информацию.

Листинг 3: Вывод программы

```
    All the lines:
task1.Line{start=task1.Point{id=0.0, x=1.0, y=1.0, z=1.0}, end=task1.Point{id=1.0, x=-1.0, y=-1.0, z=0.0}, crossOXY=task1.Point{id=-1.0, x=-1.0, y=-1.0, z=0.0}, length=3.0}
task1.Line{start=task1.Point{id=0.0, x=1.0, y=1.0, z=1.0}, end=task1.Point{id=4.0, x=0.0, y=1.0, z=-1.0}, crossOXY=task1.Point{id=-1.0, x=0.5, y=1.0, z=0.0}, length=2.23606797749979}
task1.Line{start=task1.Point{id=1.0, x=-1.0, y=-1.0, z=0.0}, end=task1.Point{id=2.0, x=0.0, y=-2.0, z=1.0}, crossOXY=task1.Point{id=-1.0, x=-1.0, y=-1.0, z=0.0}, length=1.7320508075688772}
task1.Line{start=task1.Point{id=1.0, x=-1.0, y=-1.0, z=0.0}, end=task1.Point{id=3.0, x=-3.0, y=0.0, z=1.0}, crossOXY=task1.Point{id=-1.0, x=-1.0, y=-1.0, z=0.0}, length=2.449489742783178}
task1.Line{start=task1.Point{id=1.0, x=-1.0, y=-1.0, z=0.0}, end=task1.Point{id=4.0, x=0.0, y=1.0, z=-1.0}, crossOXY=task1.Point{id=-1.0, x=-1.0, y=-1.0, z=0.0}, length=2.449489742783178}
task1.Line{start=task1.Point{id=2.0, x=0.0, y=-2.0, z=1.0}, end=task1.Point{id=4.0, x=0.0, y=1.0, z=-1.0}, crossOXY=task1.Point{id=-1.0, x=0.0, y=-0.5, z=0.0}, length=3.605551275463989}
task1.Line{start=task1.Point{id=3.0, x=-3.0, y=0.0, z=1.0}, end=task1.Point{id=4.0, x=0.0, y=1.0, z=-1.0}, crossOXY=task1.Point{id=-1.0, x=-1.5, y=0.5, z=0.0}, length=3.7416573867739413}
    All the quarters of crossOXY:
I: 1
II: 1
III: 4
IV: 1
    All the points inside/outside sphere:
[task1.Point{id=0.0, x=1.0, y=1.0, z=1.0}, task1.Point{id=1.0, x=-1.0, y=-1.0, z=0.0}, task1.Point{id=4.0, x=0.0, y=1.0, z=-1.0}]
[task1.Point{id=2.0, x=0.0, y=-2.0, z=1.0}, task1.Point{id=3.0, x=-3.0, y=0.0, z=1.0}]
1.5

Process finished with exit code 0
```

Задание 2

Множество целых чисел с операциями: 1. порождение потока таких чисел из множества, что каждое из них равно сумме двух других чисел множества; 2. поиск такого числа x в множестве, что все другие числа множества, большие x, не равны сумме двух других чисел множества; 3. добавление числа в множество.

Проверить работу второй операции нужно путём ранжирования чисел на три группы: отрицательные, нулевые и положительные.

Реализация

Я создал классы myInt и SummableNumbers в файле SummableNumbers, и на основе myInt порождал поток чисел через объект SummableNumbers
Сгенерировал экземпляр SummableNumbers и проверил работоспособность операций уже в Test.java

Код

Листинг 4: Файл SummableNumbers.java

```
1  package task2;
2
3  import java.util.*;
4  import java.util.stream.Stream;
5
6  class myInt implements Comparable<myInt>{
7      int x;
8      int id;
9      int type;
10     myInt(int x, int id){
11         this.x = x;
12         this.id = id;
13         this.type = (int) Math.signum(x);
14     }
15
16     @Override
17     public boolean equals(Object o) {
18         if (this == o) return true;
19         if (o == null || getClass() != o.getClass()) return false;
20         myInt myInt = (myInt) o;
21         return x == myInt.x && id == myInt.id;
```

```
22      }
23
24      @Override
25      public int hashCode() {
26          return Objects.hash(x, id);
27      }
28
29
30      @Override
31      public int compareTo(myInt o) {
32          return Integer.compare(this.x, o.x);
33      }
34
35      @Override
36      public String toString() {
37  //          return "myInt{" +
38  //                  "x=" + x +
39  //                  ", id=" + id +
40  //                  ", type=" + type +
41  //                  '}';
42          return Integer.toString(x);
43      }
44  }
45
46  public class SummableNumbers {
47      ArrayList<myInt> numbers = new ArrayList<>();
48
49      SummableNumbers(int[] x){
50          Arrays.stream(x).forEach(q -> numbers.add(new myInt(q, numbers.
        size())));
51      }
52      SummableNumbers(List<myInt> x){
53          x.stream().forEach(q -> numbers.add(new myInt(q.x, numbers.size
        ())));
54      }
55      public void sort(){
56          numbers.sort(myInt::compareTo);
57      }
58
59      public List<myInt> getNumbersType(int type) {
60          return numbers.stream().filter(x -> x.type == type).toList();
61      }
62
63      public void add(int x){
64          numbers.add(new myInt(x, numbers.size()));
65      }
```

```
66      public Stream<Integer> sumsStream() {
67          ArrayList<Integer> result = new ArrayList<>();
68          numbers.forEach(x1 -> numbers.stream().filter(x2 -> x1.id < x2.
        id).forEach(x2 -> result.add(x1.x + x2.x)));
69          return result.stream();
70      }
71      public Stream<Integer> sumsStreamWithout(myInt w){
72          ArrayList<Integer> result = new ArrayList<>();
73          numbers.forEach(x1 -> numbers.stream().filter(x2 -> (x1.id < x2.
        id) & (x1 != w & x2 != w)).forEach(x2 -> result.add(x1.x + x2.x)));
74          return result.stream();
75      }
76
77      private boolean myFilter(myInt x){
78          Stream<myInt> more_than_x = numbers.stream().filter(x2 -> x2.x >
         x.x);
79          return more_than_x.noneMatch(x2 -> sumsStreamWithout(x2).
        noneMatch(x3 -> x3 == x2.x));
80      }
81      public Optional<Integer> findSpecialX() {
82          Optional<Integer> result = numbers.stream().filter(this::
        myFilter).map(x -> x.x).findFirst();
83          return result;
84      }
85  }
```

Листинг 5: Файл Test.java

```
1  package task2;
2
3  /*
4  Множество целых чисел с операциями:
5  1. порождение потока таких чисел из множества,
6  что каждое из них равно сумме двух других чисел
7  множества;
8  2. поиск такого числа x в множестве, что все
9  другие числа множества, большие x, не равны
10 сумме двух других чисел множества;
11 3. добавление числа в множество.
12
13 Проверить работу второй операции нужно путём
14 ранжирования чисел на три группы:
15 отрицательные, нулевые и положительные.
16 */
17
18 public class Test {
19     private static String num2(SummableNumbers numbers){
```

```
20          if (numbers.findSpecialX().isEmpty()){
21              return "Нет такого числа";
22          } else {
23              return Integer.toString(numbers.findSpecialX().get());
24          }
25      }
26      public static void main(String[] args) {
27          SummableNumbers numbers = new SummableNumbers(new int[]{1, 2, 3,
        -15, -1, -2});
28
29          System.out.println("1) Суммы последовательности:");
30          System.out.println(numbers.sumsStream().toList());
31  //          numbers.sort(); // Не работает сортировка почему?
32  //          System.out.println(numbers.sumsStream().toList());
33
34          System.out.println("2) Число в соотвествии с условием: ");
35          System.out.println(num2(numbers));
36
37          System.out.println("Added 0 btw");
38          numbers.add(0);
39
40          SummableNumbers negative = new SummableNumbers(numbers.
        getNumbersType(-1));
41          SummableNumbers nulls = new SummableNumbers(numbers.
        getNumbersType(0));
42          SummableNumbers positive = new SummableNumbers(numbers.
        getNumbersType(1));
43
44          System.out.println("Число по группам: ");
45          System.out.println("Числа: " + negative.numbers + " Суммы: " +
        negative.sumsStream().toList() + " Число: " + num2(negative));
46          System.out.println("Числа: " + nulls.numbers + " Суммы: " +
        nulls.sumsStream().toList() + " Число: " + num2(nulls));
47          System.out.println("Числа: " + positive.numbers + " Суммы: " +
        positive.sumsStream().toList() + " Число: " + num2(positive));
48
49
50      }
51  }
```

Вывод программы

Программа посчитала все возможные суммы и числа, заданные в условии

Листинг 6: Вывод программы

```
1) Суммы последовательности:
[3, 4, -14, 0, -1, 5, -13, 1, 0, -12, 2, 1, -16, -17, -3]
2) Число в соотвествии с условием:
1
Added 0 btw
Число по группам:
Числа: [-15, -1, -2] Суммы: [-16, -17, -3] Число: -1
Числа: [0] Суммы: [] Число: 0
Числа: [1, 2, 3] Суммы: [3, 4, 5] Число: 2

Process finished with exit code 0
```

Вывод

На этот раз я научился использовать монады в языке Java, попробовал полностью заменить ими циклы, мне не понравилось, однако я понял, что в некоторых случаях они способны упрощать код.