

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №3-2
по курсу: «Языки и методы программирования»
«Полиморфизм на основе интерфейсов в языке Java»

Выполнил:
Студент группы ИУ9-21Б

Проверил:
Посевин Д. П.

Москва, 2024

1. Цель

Приобретение навыков реализации интерфейсов для обеспечения возможности полиморфной обработки объектов класса.

2. Персональный вариант

Класс, представляющий множество арифметических прогрессий, с порядком на основе количества чисел из интервала (0; 100), принадлежащих прогрессиям множества.

3. Решение

3.1. Код

```
import java.util.*;  
  
public class Test {  
    static ArithmeticProgression  
        ↪ genNewArithemticProgression(Random rand) {  
            int length = rand.nextInt(1, 10);  
            double first = rand.nextInt(-10, 100),  
                  step = rand.nextInt(1, 20);  
            return new ArithmeticProgression(first, step, length);  
    }  
  
    public static void main(String[] args) {  
        Random rand = new Random();  
        ArrayList<ArithmeticProgression> a = new  
            ↪ ArrayList<ArithmeticProgression>();  
        int m = rand.nextInt(3, 10);  
        for (int i = 0; i < m; i++)  
            a.add(genNewArithemticProgression(rand));  
  
        Collections.sort(a);  
  
        for (ArithmeticProgression ap : a)  
            ap.print();  
    }  
}
```

Код 1: Test.java

```

public class ArithmeticProgression implements Comparable<ArithmeticProgression> {
    private double first, step;
    private int length;

    public ArithmeticProgression(double first, double step, int length) {
        this.first = first;
        this.length = length;
        this.step = step;
    }

    public void print() {
        System.out.printf("First: %f; Step: %f; Length: %d\n",
                           first, step, length);
        System.out.printf("Points in (0, 100): %d\n",
                           this.countPoints());
    }

    public int countPoints() {
        int count = 0;
        double value = this.first;

        for (int k = 0; k < this.length && value < 100; k++,
             value += step)
            if (0 < value)
                count++;

        return count;
    }

    public int compareTo(ArithmeticProgression other) {
        return this.countPoints() - other.countPoints();
    }
}

```

Код 2: ArithmeticProgression.java

3.2. Скриншоты

```
[thecakeisfalse@desktop-sc src]$ java Test
First: 86.000000; Step: 12.000000; Length: 6
Points in (0, 100): 2
First: 90.000000; Step: 8.000000; Length: 4
Points in (0, 100): 2
First: 84.000000; Step: 8.000000; Length: 3
Points in (0, 100): 2
First: -7.000000; Step: 12.000000; Length: 4
Points in (0, 100): 3
First: 38.000000; Step: 10.000000; Length: 3
Points in (0, 100): 3
First: 55.000000; Step: 2.000000; Length: 6
Points in (0, 100): 6
First: 44.000000; Step: 7.000000; Length: 8
Points in (0, 100): 8
[thecakeisfalse@desktop-sc src]$
```

Рис. 1: Пример работы

The image shows two side-by-side Java code editors within an IDE interface. Both editors have a dark theme.

Top Editor (Test.java):

```

1 import java.util.*;
2
3 public class Test {
4     @
5         static ArithmeticProgression genNewArithemticProgression(Random rand) {
6             int length = rand.nextInt( origin: 1, bound: 10 );
7             double first = rand.nextInt( origin: -10, bound: 100 ),
8                 step = rand.nextInt( origin: 1, bound: 20 );
9             return new ArithmeticProgression(first, step, length);
10
11 }
12
13     public static void main(String[] args) {
14         Random rand = new Random();
15         ArrayList<ArithmeticProgression> a = new ArrayList<~>();
16         int m = rand.nextInt( origin: 3, bound: 10 );
17         for (int i = 0; i < m; i++)
18             a.add(genNewArithemticProgression(rand));
19
20         Collections.sort(a);
21
22         for (ArithmeticProgression ap : a)
23             ap.print();
24     }
25 }
```

Bottom Editor (ArithmeticProgression.java):

```

1 usages
2 public class ArithmeticProgression implements Comparable<ArithmeticProgression> {
3     private double first, step;
4     private int length;
5
6     public ArithmeticProgression(double first, double step, int length) {
7         this.first = first;
8         this.length = length;
9         this.step = step;
10
11     }
12
13     public void print() {
14         System.out.printf("First: %f; Step: %f; Length: %d\n", first, step, length);
15         System.out.printf("Points in (0, 100): %d\n", this.countPoints());
16     }
17
18     public int countPoints() {
19         int count = 0;
20         double value = this.first;
21
22         for (int k = 0; k < this.length && value < 100; k++, value += step)
23             if (0 < value)
24                 count++;
25
26         return count;
27     }
28
29     @Override
30     public int compareTo(ArithmeticProgression other) { return this.countPoints() - other.countPoints(); }
31 }
```

Рис. 2: Исходный код