

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ  
КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«Московский технический университет связи и информатики»**

Математическая кибернетика и информационные технологии

**Отчёт по лабораторной работе №4**

по дисциплине «ИТиП» на тему:

«Основы объектно-ориентированного программирования»

Выполнили: студенты группы

БВТ2104

Старостин Игорь Дмитриевич

Руководитель:

Мосева Марина Сергеевна

Москва 2022

**1. Цель работы:** изучение основ объектно-ориентированного программирования на Java.

## 2. Задание:

- a) Создать файл Point3d.java
  - i. Описать класс Point3d.
  - ii. Описать конструкторы класса Point3d.
  - iii. Описать геттеры/сеттеры класса Point3d.
  - iv. Написать метод compare класса Point3d.
  - v. Написать метод distanceTo класса Point3d.
- b) Создать файл Lab1.java
  - i. Описать класс Lab1.
  - ii. Считать координаты и создать по ним объекты трёх точек Point3d.
  - iii. Написать статический метод computeArea класса Point3d.
  - iv. Вывести рассчитанную площадь ранее сохранённых точек.
  - v. Запустить программу.

## 3. Ход работы:

- a)i. Опишем класс Point3d(См. Рис. 1)

```
//Класс трёхмерной точки в пространстве.  
6 usages  
public class Point3d {  
    //Объявляем переменные координат приватно, чтобы к ним не было доступа вне класса  
    private double x, y, z;  
}
```

**Рисунок 1 - Класс Point3d**

- ii. Описываем конструкторы класса(См. Рис. 2)

```

//Конструктор класса
2 usages
Point3d(double x, double y, double z) {
    this.x = x;
    this.y = y;
    this.z = z;
}

//Базовый конструктор класса
Point3d(){
    this( x: 0.0, y: 0.0, z: 0.0);
}

```

**Рисунок 2 - Конструкторы класса Point3d**

iii. Описываем геттеры/сеттеры класса Point3d(См. Рис. 3)

```

//Геттеры
public double getX() {
    return x;
}
public double getY() {
    return y;
}
public double getZ() {
    return z;
}

//Сеттеры
public void setX(double x) {
    this.x = x;
}
public void setY(double y) {
    this.y = y;
}
public void setZ(double z) {
    this.z = z;
}

```

**Рисунок 3 - геттеры/сеттеры класса Point3d.**

iv. Напишем метод compare класса Point3d(См. Рис. 4)

```
//Сравнивает объект с другим объектом, подаваемым через аргумент, по значению.  
3 usages  
public boolean compare(Point3d point){  
    return (this.x == point.x && this.y == point.y && this.z == point.z);  
}
```

**Рисунок 4 - Метод compare класса Point3d.**

Метод сравнивает два объекта класса Point3d по значениям.

v. Напишем метод distanceTo класса Point3d(См. Рис. 5)

```
//Возвращает геометрическое расстояние между двумя точками.  
3 usages  
public double distanceTo(Point3d point){  
    return Math.sqrt(Math.pow(this.x - point.x, 2) + Math.pow(this.y - point.y, 2) + Math.pow(this.z - point.z, 2));  
}
```

**Рисунок 5 - Метод distanceTo класса Point3d**

Метод считает геометрическое расстояние в трёхмерном пространстве.

b)i. Опишем класс Lab1(См. рис. 6)

```
import java.util.Scanner;  
//Класс Lab1 описан для работы с классом Point3d  
public class Lab1 {  
    public static void main(String[] args){  
    }  
}
```

**Рисунок 6 - Класс Lab1.**

Импортирован Scanner для реализации считывания дробных чисел с консоли.

ii. Считаем координаты и создадим объекты точек(См. Рис. 7)

```

public static void main(String[] args){
    //Инициализация массива из трёх точек и трёх координат
    Point3d[] points = new Point3d[3];
    double[] coords = new double[3];
    //Создание объекта класса Scanner для считывания координат точек
    Scanner scanner = new Scanner(System.in);
    for(int i = 0; i < 3; i++){
        //Итерация в цикле для считывания координат три раза и записи их в массив сначала координат, потом точек
        for(int j = 0; j < 3; j++){
            char coord;
            if(j == 0) coord = 'x';
            else if (j == 1) coord = 'y';
            else coord = 'z';
            System.out.print("Введите координату " + coord + " точки " + (i + 1) + ": ");
            coords[j] = scanner.nextDouble();
        }
        points[i] = new Point3d(coords[0], coords[1], coords[2]);
    }
}

```

**Рисунок 7 - Считывание координат трёх точек и создание объектов Point3d.**

Координаты считываются в массив, по которому затем создаётся объект Point3d, который помещается в массив точек.

iii. Напишем статический метод computeArea(См. Рис. 8)

```

//Функция для вычисления площади треугольника по трем точкам с применением формулы Герона.
1 usage
public static double computeArea(Point3d point1, Point3d point2, Point3d point3){
    double a = point1.distanceTo(point2);
    double b = point1.distanceTo(point3);
    double c = point2.distanceTo(point3);
    double p = (a + b + c) / 2;
    return Math.sqrt(p * (p - a) * (p - b) * (p - c));
}

```

**Рисунок 8 - Статический метод подсчёта площади.**

iv. В методе main выводим рассчитанную площадь(См. Рис. 9)

```

public static void main(String[] args){
    //Инициализация массива из трёх точек и трёх координат
    Point3d[] points = new Point3d[3];
    double[] coords = new double[3];
    //Создание объекта класса Scanner для считывания координат точек
    Scanner scanner = new Scanner(System.in);
    for(int i = 0; i < 3; i++){
        //Итерация в цикле для считывания координат три раза и записи их в массив сначала координат, потом точек
        for(int j = 0; j < 3; j++){
            char coord;
            if(j == 0)coord = 'x';
            else if (j == 1)coord = 'y';
            else coord = 'z';
            System.out.print("Введите координату " + coord + " точки " + (i + 1) + ": ");
            coords[j] = scanner.nextDouble();
        }
        points[i] = new Point3d(coords[0], coords[1], coords[2]);
    }
    //Проверка на совпадение координат точек
    if(points[0].compare(points[1]) || points[0].compare(points[2]) || points[1].compare(points[2])) {
        System.out.println("Одна из точек совпадает с другой!");
        return;
    }
    System.out.println("Площадь треугольника: " + computeArea(points[0], points[1], points[2]));
}

```

**Рисунок 9 - метод main класса Lab1.**

v. Запускаем программу(См. Рис. 10, Рис. 11)

```

Введите координату x точки 1: 1
Введите координату y точки 1: 2
Введите координату z точки 1: 1
Введите координату x точки 2: 1
Введите координату y точки 2: 4
Введите координату z точки 2: 2
Введите координату x точки 3: 1
Введите координату y точки 3: 3
Введите координату z точки 3: 2
Площадь треугольника: 0.4999999999999996

```

**Рисунок 10 - Результат работы программы для точек**

(1;2;1) (1;1;4) (1;3;2) -> 0.5

```
Введите координату x точки 1: 2
Введите координату y точки 1: 1
Введите координату z точки 1: 3
Введите координату x точки 2: 2
Введите координату y точки 2: 3
Введите координату z точки 2: 4
Введите координату x точки 3: 2
Введите координату y точки 3: 1
Введите координату z точки 3: 5
Площадь треугольника: 2.0
```

**Рисунок 11 - результат работы программы для точек:**

$(2;1;3) (2;3;4) (2;1;5) \rightarrow 2$

**Программа рассчитывает верно.**

**4. Вывод:** Мы изучили основы объектно-ориентированного программирования на Java.

**4. Литература:**

- a) !Методичка ТП.pdf
- b) лаб2.pdf

**Репозиторий с написанным кодом находится по адресу:**

**<https://github.com/Barsux/ITnP-lab-java2>**