**Федеральное агентство связи**

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования   
«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторная работа №2: Основы объектно-ориентированного программирования

по дисциплине «Кроссплатформенные технологии программирования»

Java позволяет использовать объекты. В данной лабораторной работе необходимо использовать классы по одному на файл, чтобы описать, как эти объекты работают.

Выполнил: студент

группы БСТ1803

Корнеев Александр

Вариант 12

Москва 2020

**Содержание**

[1 Задание на разработку программы 3](#_Toc41583172)

[2 Разработанные функции и классы 4](#_Toc41583173)

[2.1 Класс Point3d 4](#_Toc41583174)

[2.1.1 Метод double getX 4](#_Toc41583175)

[2.1.2 Метод double getY 5](#_Toc41583176)

[2.1.3 Метод double getZ 5](#_Toc41583177)

[2.1.4 Метод void setX 5](#_Toc41583178)

[2.1.5 Метод void setY 5](#_Toc41583179)

[2.1.6 Метод void setZ 5](#_Toc41583180)

[2.1.7 Метод boolean equals 5](#_Toc41583181)

[2.1.8 Метод double distanceTo 5](#_Toc41583182)

[2.2 Класс Lab1 6](#_Toc41583183)

[2.2.1 Метод double computeArea 6](#_Toc41583184)

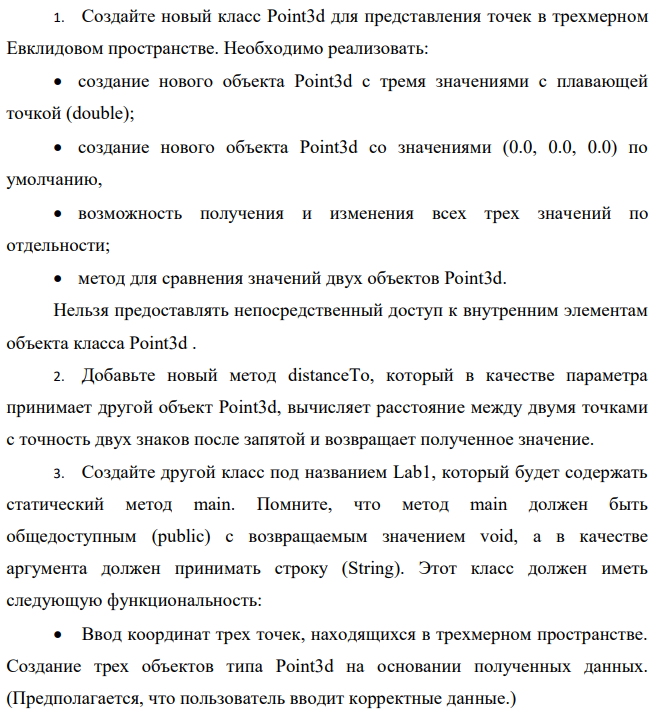
[2.2.2 Метод boolean testEqual 6](#_Toc41583185)

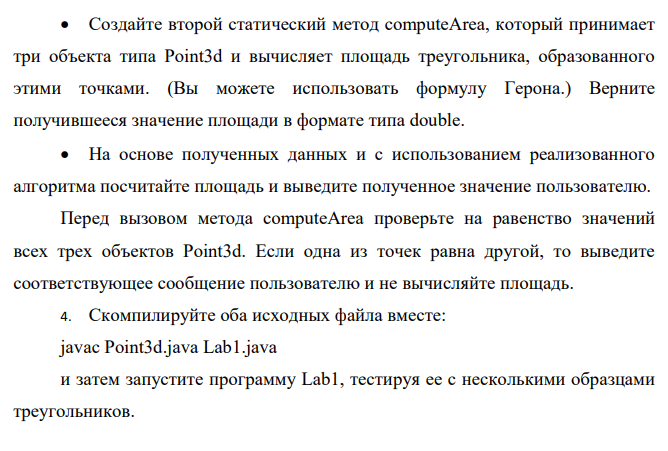
[2.2.3 Метод void main 6](#_Toc41583186)

[3 Код программы 6](#_Toc41583187)

[4 Результат работы 9](#_Toc41583188)

# Задание на разработку программы





# Разработанные функции и классы

## Класс Point3d

Класс описывающий трехмерную точку

* + 1. Метод double getX

Возвращает координату X

* + 1. Метод double getY

Возвращает координату Y

* + 1. Метод double getZ

Возвращает координату Z

* + 1. Метод void setX

Установка значения координаты X

* + 1. Метод void setY

Установка значения координаты Y

* + 1. Метод void setZ

Установка значения координаты Z

* + 1. Метод boolean equals

Метод сравнения классов Point3d, возвращает **true** если точки классов совпадают, иначе **false**

* + 1. Метод double distanceTo

Метод возвращающий расстояние между двумя точками

## Класс Lab1

Класс возвращающий площадь треугольника, образованного точками в трехмерном пространстве. Входными параметрами являются строки, передающиеся через командную строку в формате x1 y1 z1 x2 y2 z2 x3 y3 z3, где x,y,z - соответствующие координаты. 1,2,3 - соответствуют номеру точки, к которой относится координата.

* + 1. Метод double computeArea

Метод вычисляющий площадь треугольника по формуле Герона

* + 1. Метод boolean testEqual

Метод возвращающий **true,** если все точки разные, иначе **false**

* + 1. Метод void main

Осуществляет парсинг точек из входных данных, проверку того, что входные точки не совпадают и выводит площадь введённого треугольника.

# Код программы

/\*\*

\* Класс возвращающий площадь треугольника, образованного

\* точками в трехмерном пространстве. Входными параметрами являются

\* Строки передающиеся через командную строку в формате

\* x1 y1 z1 x2 y2 z2 x3 y3 z3, где x,y,z - соответствующие координаты

\* 1,2,3 - соответсвуют номеру точки к которой относится координата

\*/

**public** **class** Lab1 {

**public** **static** **void** main(String[] args) {

/\*\* координата X первой точки \*\*/

**double** xCoord1 = Double.*parseDouble*(args[0]);

/\*\* координата Y первой точки \*\*/

**double** yCoord1 = Double.*parseDouble*(args[1]);

/\*\* координата Z первой точки \*\*/

**double** zCoord1 = Double.*parseDouble*(args[2]);

/\*\* координата X второй точки \*\*/

**double** xCoord2 = Double.*parseDouble*(args[3]);

/\*\* координата Y второй точки \*\*/

**double** yCoord2 = Double.*parseDouble*(args[4]);

/\*\* координата Z второй точки \*\*/

**double** zCoord2 = Double.*parseDouble*(args[5]);

/\*\* координата X третей точки \*\*/

**double** xCoord3 = Double.*parseDouble*(args[6]);

/\*\* координата Y третей точки \*\*/

**double** yCoord3 = Double.*parseDouble*(args[7]);

/\*\* координата Z третей точки \*\*/

**double** zCoord3 = Double.*parseDouble*(args[8]);

/\*\* Инициализация первой точки \*\*/

Point3d point1 = **new** Point3d(xCoord1,yCoord1,zCoord1);

/\*\* Инициализация второй точки \*\*/

Point3d point2 = **new** Point3d(xCoord2,yCoord2,zCoord2);

/\*\* Инициализация третей точки \*\*/

Point3d point3 = **new** Point3d(xCoord3,yCoord3,zCoord3);

// Проверка не совпадают ли точки

**if** (*testEqual*(point1,point2,point3)) System.***out***.println("Площадь треугольника " + *computeArea*(point1,point2,point3));

**else** System.***out***.println("Данные введены некорректно, некоторые точки совпадают");

}

/\*\* Метод вычисляющий площадь треугольника по формуле Герона \*\*/

**public** **static** **double** computeArea(Point3d point1,Point3d point2,Point3d point3)

{

// Вычисление длин сторон треугольника

**double** a = point1.distanceTo(point2);

**double** b = point1.distanceTo(point3);

**double** c = point2.distanceTo(point3);

// Вычисление полупериметра

**double** p = (a+b+c)/2;

// Вычисление площади треугольника

**return** Math.*sqrt*(p\*(p-a)\*(p-b)\*(p-c));

}

/\*\* Метод возвращающий true, если все точки разные, иначе false \*\*/

**public** **static** **boolean** testEqual(Point3d point1, Point3d point2, Point3d point3)

{

**if** (point1.equals(point2)) **return** **false**;

**if** (point1.equals(point3)) **return** **false**;

**if** (point2.equals(point3)) **return** **false**;

**return** **true**;

}

}

/\*\*

\* класс трехменрной точки.

\*\*/

**public** **class** Point3d

{

/\*\* координата X \*\*/

**private** **double** xCoord;

/\*\* координата Y \*\*/

**private** **double** yCoord;

/\*\* координата Z \*\*/

**private** **double** zCoord;

/\*\* Конструктор инициализации \*\*/

**public** Point3d ( **double** x, **double** y, **double** z)

{

xCoord = x;

yCoord = y;

zCoord = y;

}

/\*\* Конструктор по умолчанию. \*\*/

**public** Point3d ()

{

//Вызовите конструктор с тремя параметрами и определите источник.

**this**(0, 0, 0);

}

/\*\* Возвращение координаты X \*\*/

**public** **double** getX()

{

**return** xCoord;

}

/\*\* Возвращение координаты Y \*\*/

**public** **double** getY()

{

**return** yCoord;

}

/\*\* Возвращение координаты Z \*\*/

**public** **double** getZ()

{

**return** zCoord;

}

/\*\* Установка значения координаты X. \*\*/

**public** **void** setX ( **double** val)

{

xCoord = val;

}

/\*\* Установка значения координаты Y. \*\*/

**public** **void** setY (**double** val)

{

yCoord = val;

}

/\*\* Установка значения координаты Z. \*\*/

**public** **void** setZ (**double** val)

{

zCoord = val;

}

**public** **boolean** equals(Point3d val)

{

**if** (**this**.xCoord!= val.getX()) **return** **false**;

**if** (**this**.yCoord!= val.getY()) **return** **false**;

**if** (**this**.zCoord!= val.getZ()) **return** **false**;

**return** **true**;

}

**public** **double** distanceTo(Point3d val)

{

**return** Math.*sqrt*(Math.*pow*(**this**.xCoord-val.getX(),2)+Math.*pow*(**this**.yCoord-val.getY(),2)+Math.*pow*(**this**.zCoord-val.getZ(),2));

}

}

# Результат работы

Входные аргументы



Результат работы

