Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В. Ф. Уткина»

Кафедра ЭВМ

Отчёт о лабораторной работе №3

**«Классы и методы»**

По дисциплине

«Промышленное программирование»

**Выполнили:**

ст. гр. 245 бригада № 6

Евдокимов Александр

Комарова Ксения

**Проверил:**

ст. пр. каф. ЭВМ Бастрычкин А.С.

Рязань 2024

# Цель работы

Изучение принципов ООП, приобретение навыков работы с классами и методами в Java-программах.

Практическая часть

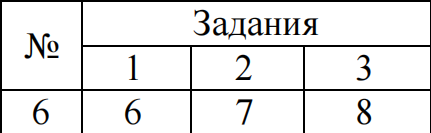


Рисунок 1 - Вариант задания

# Задание 1

Создали классы, спецификации которых приведены ниже. Определили конструкторы и методы **set***Тип***()**, **get***Тип***()**, **toString()**. Определили дополнительно методы в классе, создающем массив объектов. Задали критерий выбора данных и вывели эти данные на консоль. В каждом классе, обладающем информацией, объявили несколько конструкторов.

**House**: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации.

Создали массив объектов. Вывели:

a) список квартир, имеющих заданное число комнат;

b) список квартир, имеющих заданное число комнат и

расположенных на этаже, который находится в заданном промежутке;

c) список квартир, имеющих площадь, превосходящую заданную.

Результат работы программы представлен на рисунке 2.

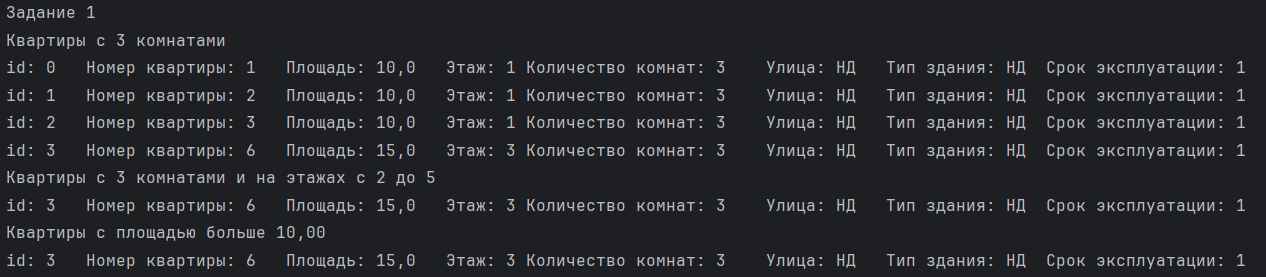


Рисунок 2 – Результат работы программы 1

# Задание 2

Определили класс **Треугольник** на плоскости. Определили площадь и периметр треугольника. Создали массив/список/множество объектов и подсчитали количество треугольников разного типа (равносторонний, равнобедренный, прямоугольный, произвольный).

Определили для каждой группы наибольший и наименьший по площади (периметру) объект.

Результат работы программы представлен на рисунке 3.

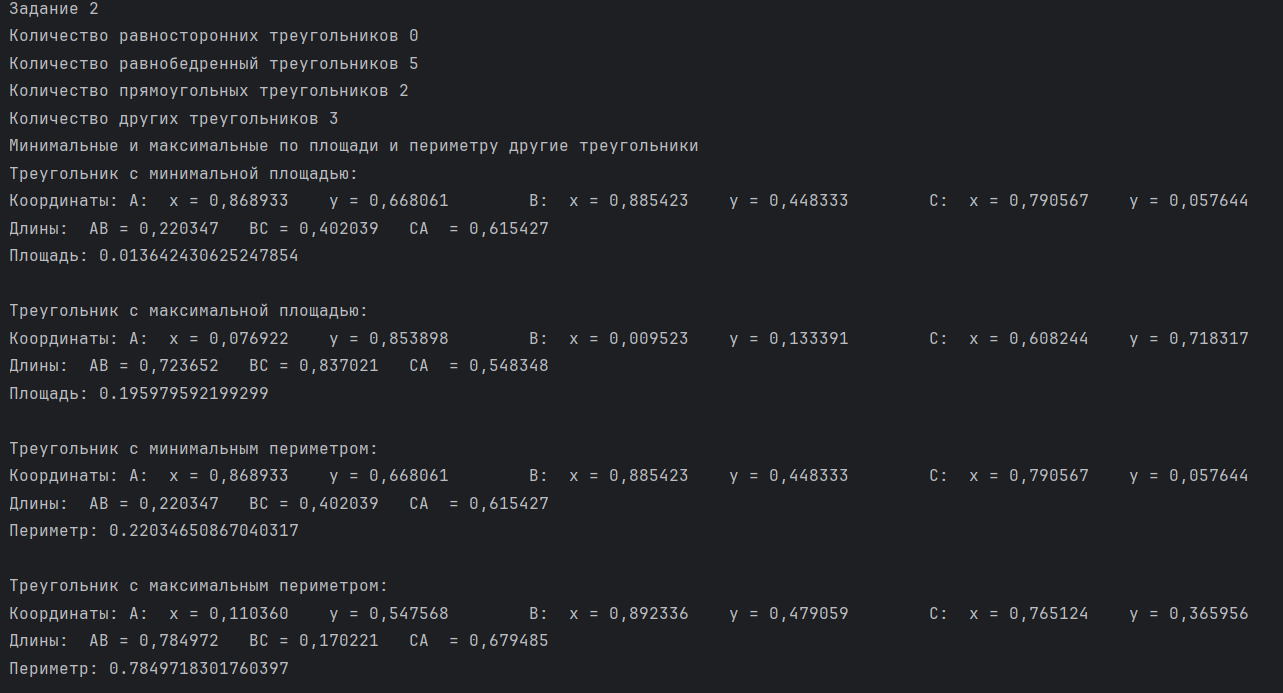


Рисунок 3 – Результат работы программы 2

# Задание 3

Определили класс **Точка** в пространстве, координаты которой задаются с помощью Комплексного числа. Создали методы по определению расстояния между точками и расстояния до начала координат.

Результат работы программы представлен на рисунке 4.

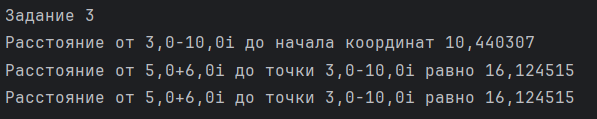


Рисунок 4 – Результат работы программы 3

Код программы представлен в приложении A/B/C/D/E/F/G.

Заключение

В ходе выполнения лабораторной работы были изучены принципы ООП, приобретены навыков работы с классами и методами в Java-программах.

Приложение А. Код класса Main

public class Main {  
  
 public static void main(String[] args) {  
 System.*out*.println("Задание 1");  
 *Task1*();  
 System.*out*.println("Задание 2");  
 *Task2*();  
 System.*out*.println("Задание 3");  
 *Task3*();  
 }  
  
 public static void Task1(){  
 HouseGenerator houseGen = new HouseGenerator(3);  
 houseGen.addNew(new House(0,6,15.f,3,3,1));  
 houseGen.printHousesWithRooms(3);  
 houseGen.printHousesWithRoomsAndFloorRange(3,2,5);  
 houseGen.printHousesWithSquareMoreThan(10.f);  
 System.*out*.printf("\nСписок домов\n%s", houseGen);  
 }  
  
 public static void Task2(){  
 TriangleGenerator trGen = new TriangleGenerator(10);  
 trGen.printCountOfSorted();  
 trGen.recountTriangles();  
 System.*out*.println("Минимальные и максимальные по площади и периметру другие треугольники");  
 TriangleGenerator.*printMinNMaxForGroup*(trGen.getOtherTriangles());  
 }  
  
 public static void Task3(){  
 PointComplex complexPoint1 = new PointComplex(5,6);  
 PointComplex complexPoint2 = new PointComplex(3,-10);  
 System.*out*.printf("Расстояние от %s до начала координат %f\n", complexPoint2, complexPoint2.getDistFromThisToCordStart());  
 System.*out*.printf("Расстояние от %s до точки %s равно %f\n", complexPoint1, complexPoint2, complexPoint2.getDistFromThisToPointComplex(complexPoint1));  
 System.*out*.printf("Расстояние от %s до точки %s равно %f\n", complexPoint1, complexPoint2, PointComplex.*getDistFromOneToAnotherPoint*(complexPoint1, complexPoint2));  
 }  
}

Приложение B. Код класса House

public class House{  
 private int id, number;  
 private float square;  
 private int floor, rooms;  
 private String street;  
 private String buildingType;  
 private int lifetime;  
  
 House(int id, int number, float sq, int floor, int rooms, String street, String bType, int lifetime){  
 this.id = id;  
 this.number = number;  
 this.square = sq;  
 this.floor = floor;  
 this.rooms = rooms;  
 this.street = street;  
 this.buildingType = bType;  
 this.lifetime = lifetime;  
 }  
  
 House(int id, int number, float sq, int floor, int rooms, int lifetime){ this(id,number,sq,floor,rooms,"НД", "НД", lifetime);  
 }  
  
 House(){ this(-1,-1,-1,-1,-1, "НД", "НД", -1);  
 }  
  
 public void setId(int newId){  
 this.id=newId;  
 }  
  
 public void setNumber(int newNumber){  
 this.number=newNumber;  
 }  
  
 public void setSquare(int newSq){  
 this.square=newSq;  
 }  
  
 public void setFloor(int newFloor){  
 this.floor=newFloor;  
 }  
  
 public void setRooms(int newRooms){  
 this.rooms=newRooms;  
 }  
  
 public void setStreet(String newStreet){  
 this.street=newStreet;  
 }  
  
 public void setBuildingType(String newBType){  
 this.buildingType=newBType;  
 }  
  
 public void setLifetime(int newLifetime){  
 this.lifetime=newLifetime;  
 }  
  
 public int getId(){  
 return this.id;  
 }  
  
 public int getNumber(){  
 return this.number;  
 }  
  
 public float getSquare(){  
 return this.square;  
 }  
  
 public int getFloor(){  
 return this.floor;  
 }  
  
 public int getRooms(){  
 return this.rooms;  
 }  
  
 public String getStreet(){  
 return this.street;  
 }  
  
 public String getBuildingType(){  
 return this.buildingType;  
 }  
  
 public int getLifetime(){  
 return this.lifetime;  
 }  
  
 public String toString() {  
 return String.*format*("id: %d\tНомер квартиры: %d\tПлощадь: %.1f\tЭтаж: %d\t" + "Количество комнат: %d\tУлица: %s\tТип здания: %s\tСрок эксплуатации: %d", id, number, square, floor, rooms, street, buildingType, lifetime);  
 }  
  
 public void setData(int id, int number, float sq, int floor, int rooms, String street, String bType, int lifetime){  
 this.id = id;  
 this.number = number;  
 this.square = sq;  
 this.floor = floor;  
 this.rooms = rooms;  
 this.street = street;  
 this.buildingType = bType;  
 this.lifetime = lifetime;  
 }  
}

Приложение C. Код класса HouseGenerator

import java.util.Arrays;  
  
public class HouseGenerator {  
 private House[] houses;  
 HouseGenerator(int count){  
 houses = new House[count];  
 for(int i =0 ;i<count; i++)  
 houses[i] = new House(i, i+1, 10.f, 1+i/4, 3, 1);  
 }  
  
 public House getHouseById(int id){  
 if (id >= 0 && id < houses.length) return houses[id];  
 System.*out*.print("Ошибка: Выход за границы массива\n");  
 return null;  
 }  
  
 public House[] getHouses() {  
 return houses;  
 }  
  
 public String toString(){  
 String output = "";  
 for(House each : houses)  
 output = output.concat(each+"\n");  
 return output;  
 }  
  
 public void addNew(House newHouse){  
 houses = Arrays.*copyOf*(houses, houses.length+1);  
 houses[houses.length-1]=newHouse;  
 houses[houses.length-1].setId(houses.length-1);  
 }  
  
 public void eraseById(int id){  
 if(id<0 || id>= houses.length) {  
 System.*out*.print("Ошибка: Выход за границы массива\tНичего не изменено\n");  
 return;  
 }  
 for(int i = 0, k = 0; i < houses.length; i++){  
 if (i == id) {  
 continue;  
 }  
 houses[k++] = houses[i];  
 }  
 houses = Arrays.*copyOf*(houses, houses.length-1);  
 }  
  
 public void printHousesWithRooms(int rooms){  
 System.*out*.printf("Квартиры с %d комнатами\n", rooms);  
 for(House one : houses){  
 if(one.getRooms()==rooms)  
 System.*out*.println(one);  
 }  
 }  
  
 public void printHousesWithRoomsAndFloorRange(int rooms, int minFloor, int maxFloor){  
 System.*out*.printf("Квартиры с %d комнатами и на этажах с %d до %d\n", rooms, minFloor, maxFloor);  
 for(House one : houses){  
 if(one.getRooms()==rooms && one.getFloor() >= minFloor && one.getFloor()<=maxFloor)  
 System.*out*.println(one);  
 }  
 }  
  
 public void printHousesWithSquareMoreThan(float sq){  
 System.*out*.printf("Квартиры с площадью больше %.2f\n", sq);  
 for(House one : houses)  
 if(one.getSquare()>sq)  
 System.*out*.println(one);  
 }  
}

Приложение D. Код класса Point2d

public class Point2d {  
 private double x,y;  
  
 public double getX(){  
 return x;  
 }  
  
 public double getY() {  
 return y;  
 }  
  
 public void setX(double x){  
 this.x = x;  
 }  
  
 public void setY(double y) {  
 this.y = y;  
 }  
  
 Point2d(double x, double y){  
 this.x = x;  
 this.y = y;  
 }  
  
 Point2d(){  
 this(0,0);  
 }  
  
 public static double getDistance(Point2d A, Point2d B){  
 return Math.*sqrt*(Math.*pow*(B.x-A.x,2)+ Math.*pow*(B.y-A.y,2));  
 }  
  
 public String toString(){  
 return String.*format*("x = %f\ty = %f", this.x, this.y);  
 }  
}

Приложение E. Код класса Triangle

public class Triangle {  
  
 private Point2d A;  
 private Point2d B;  
 private Point2d C;  
  
 public static final double *accuracy* = 0.0001;  
  
 Triangle(Point2d A, Point2d B, Point2d C){  
 this.A = A;  
 this.B = B;  
 this.C = C;  
 }  
  
 Triangle(){  
 this.A = new Point2d();  
 this.B = new Point2d();  
 this.C = new Point2d();  
 }  
  
 public String toString(){  
 return String.*format*("Координаты:\tA:\t%s\t\t", A) +  
 String.*format*("B:\t%s\t\t", B) +  
 String.*format*("C:\t%s\n", C) +  
 String.*format*("Длины:\tAB = %f\tBC = %f\tCA = %f\n",  
 Point2d.*getDistance*(A,B),  
 Point2d.*getDistance*(B,C),  
 Point2d.*getDistance*(C,A));  
 }  
  
 public double findSq(){  
 return 0.5\*Math.*abs*((B.getX()-A.getX())\*(C.getY()-A.getY())-(C.getX()-A.getX())\*(B.getY()-A.getY()));  
 }  
  
 public double findP(){  
 return Math.*sqrt*(Math.*pow*(B.getX() - A.getX(),2) + Math.*pow*(B.getY() - A.getY(),2));  
 }  
  
 public void setPoints(Point2d newPointA,Point2d newPointB,Point2d newPointC){  
 this.A=newPointA;  
 this.B=newPointB;  
 this.C=newPointC;  
 }  
public boolean isEquilateralTriangles(){ //равносторониий  
 return Math.*abs*(Point2d.*getDistance*(A,B)-Point2d.*getDistance*(B,C))<*accuracy* && Math.*abs*(Point2d.*getDistance*(B,C)-Point2d.*getDistance*(A,C))<*accuracy*;  
 }public boolean isIsoscelesTriangle(){ //равнобедренный  
 return (  
 Math.*abs*(Point2d.*getDistance*(A,B)-Point2d.*getDistance*(B,C))<*accuracy* && Math.*abs*(Point2d.*getDistance*(A,B)-Point2d.*getDistance*(A,C))>=*accuracy* ) || (  
 Math.*abs*(Point2d.*getDistance*(A,C)-Point2d.*getDistance*(A,B))<*accuracy* && Math.*abs*(Point2d.*getDistance*(A,B)-Point2d.*getDistance*(C,B))>=*accuracy* ) || (  
 Math.*abs*(Point2d.*getDistance*(A,C)-Point2d.*getDistance*(C,B))<*accuracy* && Math.*abs*(Point2d.*getDistance*(A,C)-Point2d.*getDistance*(A,B))>=*accuracy* );  
 }  
public boolean isRightTriangle(){ //прямоугольный  
 return (  
 Math.*abs*((Math.*pow*(Point2d.*getDistance*(A,B),2)  
 + Math.*pow*(Point2d.*getDistance*(B,C),2)) - Math.*pow*(Point2d.*getDistance*(A,C),2))<*accuracy* ) || (  
 Math.*abs*((Math.*pow*(Point2d.*getDistance*(A,B),2)  
 + Math.*pow*(Point2d.*getDistance*(A,C),2)) - Math.*pow*(Point2d.*getDistance*(B,C),2))<*accuracy* ) || (  
 Math.*abs*((Math.*pow*(Point2d.*getDistance*(B,C),2)  
 + Math.*pow*(Point2d.*getDistance*(A,C),2)) - Math.*pow*(Point2d.*getDistance*(A,B),2))<*accuracy* );  
 }  
}

Приложение F. Код класса TriangleGenerator

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Random;  
  
public class TriangleGenerator {  
 private int count;  
 private Triangle[] triangles;  
  
 private ArrayList<Triangle> equilateralTriangles;  
 private ArrayList<Triangle> isoscelesTriangle;  
 private ArrayList<Triangle> rightTriangle;  
 private ArrayList<Triangle> other;  
  
 public ArrayList<Triangle> getEquilateralTriangles(){  
 return equilateralTriangles;  
 }  
  
 public ArrayList<Triangle> getIsoscelesTriangle(){  
 return isoscelesTriangle;  
 }  
  
 public ArrayList<Triangle> getRightTriangle(){  
 return rightTriangle;  
 }  
  
 public ArrayList<Triangle> getOtherTriangles(){  
 return other;  
 }  
  
 TriangleGenerator(int count, double origin, double bounds){  
 Random rd = new Random();  
 this.count = count;  
 triangles = new Triangle[count];  
  
 equilateralTriangles = new ArrayList<>();  
 isoscelesTriangle = new ArrayList<>();  
 rightTriangle = new ArrayList<>();  
 other = new ArrayList<>();  
  
 for(int i =0; i<count; i++){  
 Point2d A = new Point2d(rd.nextDouble(origin,bounds), rd.nextDouble(origin,bounds));  
 Point2d B = new Point2d(rd.nextDouble(origin,bounds), rd.nextDouble(origin,bounds));  
 Point2d C = new Point2d(rd.nextDouble(origin,bounds), rd.nextDouble(origin,bounds));  
 triangles[i]=new Triangle(A,B,C);  
 }  
 recountTriangles();  
 }  
  
 TriangleGenerator(int count){  
 this(count, 0,1);  
 }  
  
 TriangleGenerator(){  
 this(0, 0,1);  
 }  
  
 TriangleGenerator(Triangle[] triangles){  
 this.count = triangles.length;  
 this.triangles = triangles;  
  
 equilateralTriangles = new ArrayList<>();  
 isoscelesTriangle = new ArrayList<>();  
 rightTriangle = new ArrayList<>();  
 other = new ArrayList<>();  
  
 recountTriangles();  
 }  
  
 public void setTriangles(Triangle[] triangles){  
 this.count = triangles.length;  
 this.triangles = triangles;  
 }  
  
 public void recountTriangles(){  
 equilateralTriangles = new ArrayList<>();  
 isoscelesTriangle = new ArrayList<>();  
 rightTriangle = new ArrayList<>();  
 other = new ArrayList<>();  
 for(int i =0; i<count; i++){  
 if(triangles[i].isEquilateralTriangles()) equilateralTriangles.add(triangles[i]);  
 else if(triangles[i].isIsoscelesTriangle()) isoscelesTriangle.add(triangles[i]);  
 else if(triangles[i].isRightTriangle()) rightTriangle.add(triangles[i]);  
 else other.add(triangles[i]);  
 }  
 }  
  
 public Triangle getTriangleById(int id){  
 if(id < 0 || id>triangles.length) {  
 System.*out*.print("Ошибка: Выход за границы массива\n");  
 return null;  
 }  
 return triangles[id];  
 }  
  
 public void addNew(Triangle newTriangle){  
 triangles = Arrays.*copyOf*(triangles, triangles.length+1);  
 triangles[triangles.length-1]=newTriangle;  
 }  
  
 public void eraseById(int id){  
 if(id<0 || id>= triangles.length) {  
 System.*out*.print("Ошибка: Выход за границы массива\tНичего не изменено\n");  
 return;  
 }  
 for(int i = 0, k = 0; i < triangles.length; i++){  
 if (i == id) {  
 continue;  
 }  
 triangles[k++] = triangles[i];  
 }  
 triangles = Arrays.*copyOf*(triangles, triangles.length-1);  
 }  
  
 public static void printList(ArrayList<Triangle> list){  
 if(list.isEmpty()) return;  
 for(Triangle one : list) {  
 System.*out*.printf("%s\n", one);  
 }  
 }  
  
 public void printCountOfSorted(){  
 recountTriangles();  
 System.*out*.printf("Количество равносторонних треугольников %d\n", equilateralTriangles.size());  
 System.*out*.printf("Количество равнобедренный треугольников %d\n", isoscelesTriangle.size());  
 System.*out*.printf("Количество прямоугольных треугольников %d\n", rightTriangle.size());  
 System.*out*.printf("Количество других треугольников %d\n", other.size());  
 }  
  
 public static void printMinNMaxForGroup(ArrayList<Triangle> list){  
 if(list.isEmpty()) {  
 System.*out*.print("Ошибка: Массив пуст\n");  
 return;  
 }  
 Triangle minSq = list.getFirst();  
 Triangle maxSq = list.getFirst();  
 Triangle minP = list.getFirst();  
 Triangle maxP = list.getFirst();  
 for(Triangle one : list){  
 if(one.findSq() > maxSq.findSq()) maxSq=one;  
 if(one.findSq() < minSq.findSq()) minSq=one;  
 if(one.findP() > maxP.findP()) maxP=one;  
 if(one.findP() < minSq.findP()) minP=one;  
 }  
 System.*out*.println("Треугольник с минимальной площадью:\n" + minSq + "Площадь: " + minSq.findSq() + "\n");  
 System.*out*.println("Треугольник с максимальной площадью:\n" + maxSq + "Площадь: " + maxSq.findSq() + "\n");  
 System.*out*.println("Треугольник с минимальным периметром:\n" + minP + "Периметр: " + minP.findP() + "\n");  
 System.*out*.println("Треугольник с максимальным периметром:\n" + maxP + "Периметр: " + maxP.findP() + "\n");  
 }  
}

Приложение G. Код класса PointComplex

public class PointComplex {  
 private double real, imaginary;  
  
 PointComplex(double real, double imaginary){  
 this.real = real;  
 this.imaginary = imaginary;  
 }  
   
 PointComplex(){  
 this(0,0);  
 }  
  
 public double getReal(){  
 return real;  
 }  
  
 public double getImaginary(){  
 return imaginary;  
 }  
  
 public void setReal(double real) {  
 this.real = real;  
 }  
  
 public void setImaginary(double imaginary) {  
 this.imaginary = imaginary;  
 }  
  
 public PointComplex sum(PointComplex anotherPoint){  
 return new PointComplex((this.real+anotherPoint.getReal()), (this.imaginary+anotherPoint.getImaginary()));  
 }  
  
 public static PointComplex sum(PointComplex point, PointComplex anotherPoint){  
 return new PointComplex((point.getReal()+anotherPoint.getReal()), (point.getImaginary()+anotherPoint.getImaginary()));  
 }  
  
 public PointComplex sub(PointComplex anotherPoint){  
 return new PointComplex((this.real-anotherPoint.getReal()), (this.imaginary-anotherPoint.getImaginary()));  
 }  
  
 public static PointComplex sub(PointComplex point, PointComplex anotherPoint){  
 return new PointComplex((point.getReal()-anotherPoint.getReal()), (point.getImaginary()-anotherPoint.getImaginary()));  
 }  
  
 public String toString(){  
 if(imaginary==0) return String.*format*("%.1fi", real);  
 if(imaginary<0) return String.*format*("%.1f%.1fi", real, imaginary);  
 return String.*format*("%.1f+%.1fi", real, imaginary);  
 }  
  
 public double getDistFromThisToPointComplex(PointComplex anotherPoint){  
  
 return Math.*sqrt*(Math.*pow*(anotherPoint.getReal()-this.real, 2)+Math.*pow*(anotherPoint.getImaginary()-this.imaginary, 2));  
 }  
  
 public double getDistFromThisToCordStart(){  
 return Math.*sqrt*(Math.*pow*(this.real, 2)+Math.*pow*(this.imaginary, 2));  
 }  
  
 public static double getDistFromOneToAnotherPoint(PointComplex onePoint, PointComplex secondPoint){  
 return Math.*sqrt*(Math.*pow*(onePoint.getReal()-secondPoint.getReal(), 2)+Math.*pow*(onePoint.getImaginary()-secondPoint.getImaginary(), 2));  
 }  
  
}