**Физико-математический лицей №239**

**Нахождение площади многоугольника**

**(не обязательно выпуклого)**

Годовой проект по информатике.

Работу выполнил

Ученик 10- класса

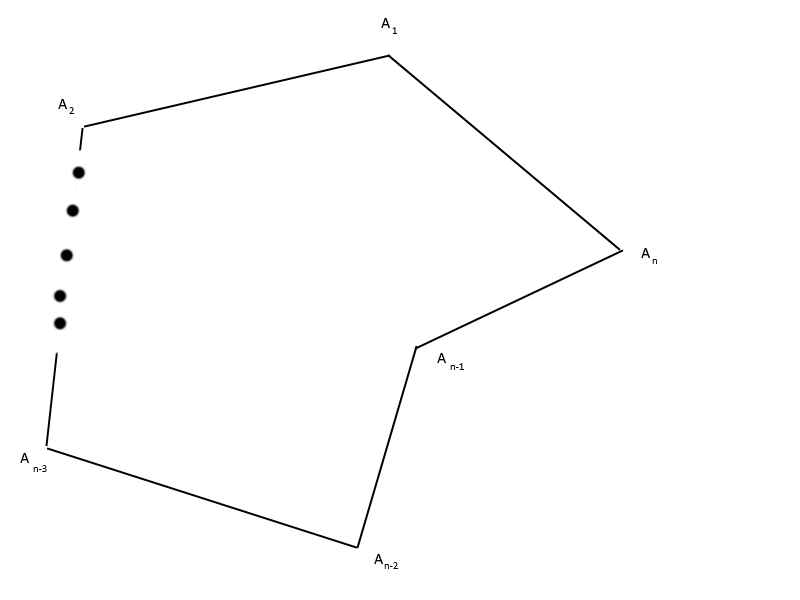
Бей Святослав

Санкт-Петербург

2017

**1.Постановка задачи**

Многоугольник (не обязательно выпуклый) задан на плоскости перечислением координат вершин в порядке обхода его границы. Определить площадь многоугольника.



**2.Уточнение исходных и входных данных и ограничений на них**

**2.1Исходные данные**

Во входном файле out.txt содержатся строки с информацией о колличестве вершин многоугольника, также там содержатся координаты каждой из вершин.

Формат записи следующий: колличество вершин многоугольника на первой строчке, и на каждой последующей два целых числа – координаты вершины: x и y. Колличество вершин – целое положительное число. Координаты вершин также являются целыми положительными числами, так как начало координат находится в левом верхнем углу панели, и стороны панели совпадают с положительно направленными осями координат.

x в пределах от 0 до 400, y от 0 до 700, из за размеров панели.

**2.2Выходные данные**

На выход подается вещественное положительное число типа double - площадь многоугольника.

**3.Выбор метода решения**

**3.1Анализ исходных данных и выбор используемой структуры данных**

Для решения задачи:

Пакет Problem:

1) Класс Point с полями x и y(координаты)

Конструктор Point(double x, double y).

2)Класс Triangle c полями Point a и Point b(вершины треугольника). Третья точка - начало координат.(смотри построение алгоритма)

Метод getOrientArea, считающий площадь треугольника как произведение 0.5\*(x1\*y2-x2\*y1).

3)Класс Polygon c полем Triangle tArr[]- массив треугольников.

Конструктор Polygon(int n,ArrayList<PanelPoint> panelPoints), который считывает соответствующие координаты вершин с текстовых полей на панели и добавляет в массив tArr[] треугольники в порядке обхода.

Метод getAreaP, считающий площадь многоугольника как сумму ориентированных площадей треугольников.

Для визуализации решения:

Пакет com.visual:

1) Класс Panel, наследуемый от JPanel.Здесь будут рисоваться кнопки, окошки для ввода координат вершин и вывода ответа, а также будет рисоваться сам многоугольник.

2) Класс PanelPoint, наследуемый от JPanel. Он содержит в себе поля

int n;(номер вершины)

JLabel X;

JLabel Y;

JTextField x;(поле для ввода координаты)

JTextField y;(поле для ввода координаты)

2 конструтора: PanelPoint(int n) и PanelPoint(**int** n, **int** x1, **int** y1), которые создают на ButPanel

Окошки для ввода координат и подписи к ним(x1, y1, x2 , y2 и т.д.).

2 метода Xget() и Yget(), которые возвращают содержащиеся в полях JTextField числа.

3) Класс PolygonD( D – значит Drawable). Это класс, методы которого рисуют многоугольник.

Он также наследуется от JPanel. У него есть поле int n -колличество вершин, а также ArrayList<PanelPoint> **panelPoints** = **new** ArrayList<>() – здесь содержатся координаты вершин.

Конструтор PolygonD(**int** n, ArrayList<PanelPoint> panelPoints): здесь в массив добавляются вершины из пакета Problem

Метод **public void** paint(Graphics g). Он при помощи метода drawPolygon рисует многоугольник.

4) Класс Vector, который проводит радиус-вектор в каждую вершину многоугольника. Этот класс нужен для визуализации решения. Он содержит в себе конструктор Vector и метод Paint.

5) Класс Main.

В данном классе есть метод CreateGUI, который и создает интерфейс программы. Он создает Panel, где рисуется многоугольник и ButPanel, где рисуются кнопки и окошки ввода и вывода. Всего кнопок 9: “добавить колличество вершин”, «задать случайные вершиныЭ», «задать», «чтение из файла», «вывод в файл», « посчитать площадь», «очистить».

**3.2Выбор метода решения**

Для решения этой задачи нам понадобится понятие ориентированной площади треугольника, взятое из геометрии.

Ориентированной площадью треугольника ABC называется величина (ABC), равная его площади, взятой со знаком плюс, если обход треугольника в порядке A–B–C–A совершается x

Если координаты точек A и B равны xa, ya, xb, yb, то ориентированная площадь треугольника OAB(O - начало координат) равна:

(находится через векторное произведение)

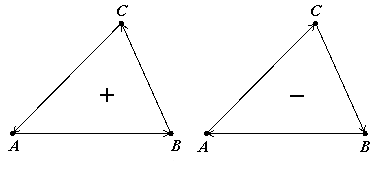
Причем площадь будет положительна, так как вершины ориентированы положительно.

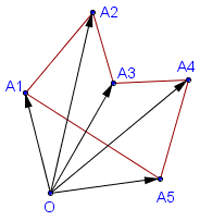
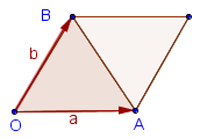
Аналогично мы можем найти площадь многоугольника, разбив его на треугольники. Итоговая формула будет иметь вид:

Где xi, yi  - координаты точки Ai

Модуль этой величины равен "обычной" площади многоугольника.

**Ориентированнная площадь треугольника**



 **Векторное произведение и многоугольник с радиус векторамию**

1) Ввод координат точек.

2)Создание массива точек, где каждой точке присваиваются ее координаты.

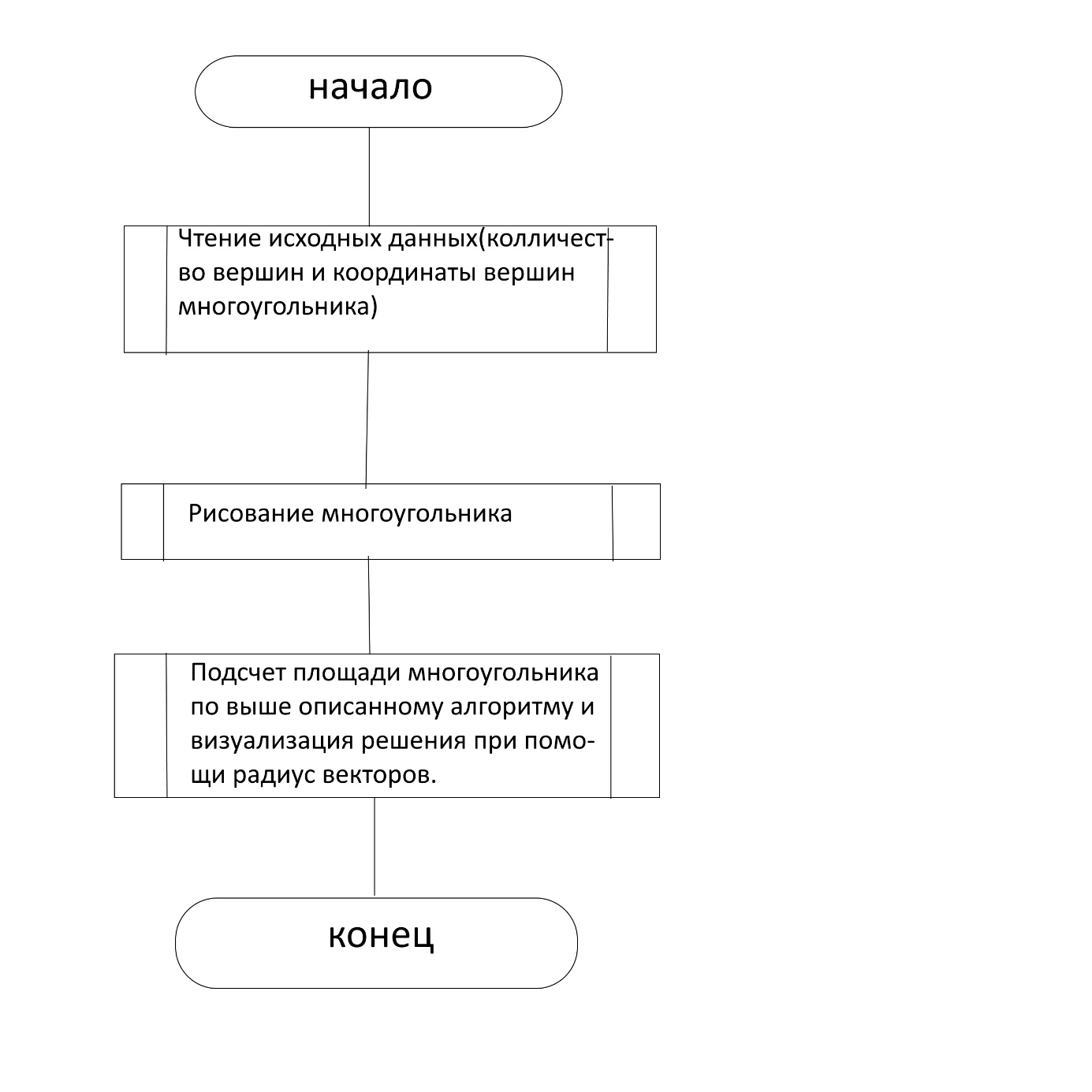
3) Создание массива треугольников, где задаются вершины каждого треугольника.

4) Создание метода подсчета площади треугольника.

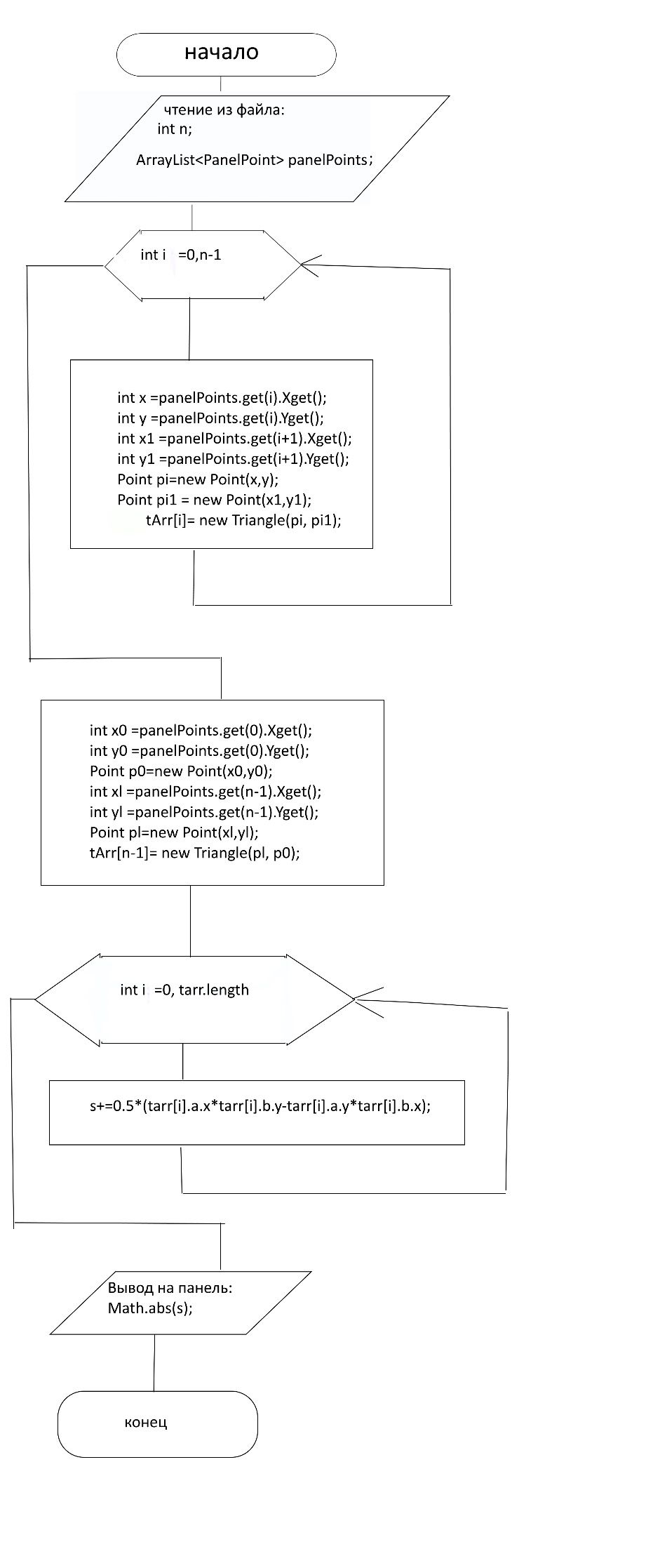
5)Подсчет площади по указанной выше формуле - как сумма площадей треугольников.

**4.Составление алгоритма**

**4.1 Обобщенная блок-схема алгоритма**



**4.2 Блок-схема алгоритма**



**5.Листинг программы**

**Классы для решения задачи:**

**1)Класс Point**

**package** problem;  
  
**public class** Point {  
 **public double x**;// координаты точки  
 **public double y**;  
  
 **public** Point(**double** x, **double** y){ // конструктор для присваивания координат  
 **this**.**x** = x;  
 **this**.**y** = y;  
 }  
  
}

**2)Класс Triangle**

**package** problem;  
  
**public class** Triangle {  
 **public** Point **a**; // вершины треугольника  
 **public** Point **b**;  
  
 **public** Triangle(Point a, Point b) { // конструктор для присвоения вершин  
 **this**.**a** = a;  
 **this**.**b** = b;  
 }  
  
 **public double** getOrientArea(){ // метод для подсчета ориентированной площади   
  
 **return** 0.5\*(**this**.**a**.**x**\***this**.**b**.**y**-**this**.**b**.**x**\***this**.**a**.**y**); // треугольника  
 }  
}

**3) Класс Polygon**

**package** problem;  
  
**import** com.visual.PanelPoint;  
  
**import** java.util.ArrayList;  
  
**public class** Polygon {  
 Triangle **tArr**[];*// массив треугольников, образованных радиус векторами из начала координат* **public** Polygon(**int** n,ArrayList<PanelPoint> panelPoints){*// конструктор, который преобразует содержимое полей JTextField элементов массива panelPoints типа String в Int. И заполняет массив tArr.* **tArr** = **new** Triangle[n];  
 **for**(**int** i =0;i<n-1;i++){  
 **int** x =panelPoints.get(i).Xget();  
 **int** y =panelPoints.get(i).Yget();  
 **int** x1 =panelPoints.get(i+1).Xget();  
 **int** y1 =panelPoints.get(i+1).Yget();  
 Point pi=**new** Point(x,y);  
 Point pi1 = **new** Point(x1,y1);  
 **this**.**tArr**[i]= **new** Triangle(pi, pi1);  
 }  
 **int** x0 =panelPoints.get(0).Xget();  
 **int** y0 =panelPoints.get(0).Yget();  
 Point p0=**new** Point(x0,y0);  
 **int** xl =panelPoints.get(n-1).Xget();  
 **int** yl =panelPoints.get(n-1).Yget();  
 Point pl=**new** Point(xl,yl);  
 **this**.**tArr**[n-1]= **new** Triangle(pl, p0);  
 }  
  
 **public double** getArea(Polygon p){*// Метод, считающий площадь многоугольника как сумму ориентированных площадей треугольников(элементов массива tArr[])* **double** s = 0;  
 **for**(**int** i = 0;i<p.**tArr**.**length**;i++){  
 s+=p.**tArr**[i].getOrientArea();  
 }  
 **return** Math.*abs*(s);  
 }  
  
}

**Классы для визуализации задачи:**

1. **Класс Panel**
2. **package** com.visual;  
     
   **import** javax.swing.\*;  
   **import** java.awt.\*;  
     
   **public class** Panel **extends** JPanel {  
    **public void** paintComponent(Graphics g){*// Метод, задающий цвет и размеры панели* g.setColor(Color.***white***);  
    g.fillRect(0,0,getWidth(),getHeight());  
    g.setColor(Color.***black***);  
    g.drawRect(0,0,getWidth()-1,getHeight()-1);  
    }  
   }

**2)Класс PanelPoint**

**package** com.visual;  
  
**import** javax.swing.\*;  
  
**public class** PanelPoint **extends** JPanel{  
 **int n**;*// номер точки* JLabel **X**;*// подписи к окошкам ввода* JLabel **Y**;  
 JTextField **x**;*// окошки ввода координат* JTextField **y**;  
 PanelPoint(**int** n){*// конструктор для отрисовки на панели окошек с подписями* **this**.**n** =n;  
 **X** = **new** JLabel(**"X"**+n+**":"**);  
 **Y** =**new** JLabel(**"Y"**+n+**":"**);  
 **x** = **new** JTextField();  
 **y** = **new** JTextField();  
 **X**.setBounds(0,0, 20,30);  
 **Y**.setBounds(60,0,20,30);  
 add(**X**);  
 add(**Y**);  
 **x**.setBounds(20,0,20,30);  
 **y**.setBounds(80,0,20,30);  
 setLayout(**null**);  
 add(**x**);  
 add(**y**);  
 repaint();  
 revalidate();  
 }  
 PanelPoint(**int** n, **int** x1, **int** y1){*//аналогичный предыдущему конструктор, но теперь окошки заполняются введенными координатами* **this**.**n** =n;  
 **X** = **new** JLabel(**"X"**+n+**":"**);  
 **Y** =**new** JLabel(**"Y"**+n+**":"**);  
 **x** = **new** JTextField(**""**+x1);  
 **y** = **new** JTextField(**""**+y1);  
 **X**.setBounds(0,0, 20,30);  
 **Y**.setBounds(60,0,20,30);  
 add(**X**);  
 add(**Y**);  
 **x**.setBounds(20,0,40,30);  
 **y**.setBounds(80,0,40,30);  
 setLayout(**null**);  
 add(**x**);  
 add(**y**);  
 repaint();  
 revalidate();  
 }  
  
 **public int** Xget(){*// конструктор для перевода содержимого JTextField x из String в Int* **int** x1=(!**x**.getText().equals(**""**)?Integer.*parseInt*(**x**.getText()):0);  
 **return** x1;  
 }  
 **public int** Yget(){*// конструктор для перевода содержимого JTextField y из String в Int* **int** y1=(!**y**.getText().equals(**""**)?Integer.*parseInt*(**y**.getText()):0);  
 **return** y1;  
 }  
}

**3)Класс PolygonD**

**package** com.visual;  
  
  
**import** javax.swing.\*;  
**import** java.awt.BasicStroke;  
**import** java.awt.Color;  
**import** java.awt.Graphics;  
**import** java.awt.Graphics2D;  
**import** java.awt.Polygon;  
**import** java.util.ArrayList;  
  
**public class** PolygonD **extends** JPanel {  
 **private int n**;*// количество вершин многоугольника* **private** ArrayList<PanelPoint> **panelPoints** = **new** ArrayList<>();*// динамический массив, где содержатся объекты класса PanelPoint* **public** PolygonD(**int** n, ArrayList<PanelPoint> panelPoints) {*// конструктор для заполнения массива panelPoints* **this**.**n** = n;  
 **for** (PanelPoint p: panelPoints){  
 **this**.**panelPoints**.add(p);  
 }  
 }  
  
 @Override  
 **public void** paint(Graphics g) {*// метод, рисующий многоугольник* Graphics2D gr2d = (Graphics2D) g;*//применение пакета Graphics2D* BasicStroke с = **new** BasicStroke(3);*//задание толщины линии* gr2d.setStroke(с);*// добавление линии* gr2d.setPaint(Color.***MAGENTA***);*//задание цвета линии* Polygon j = **new** Polygon();*//создание многоугольника* **for**(**int** i =0; i<**n**;i++){*// добавление вершин многоугольника* j.addPoint(Main.*panelPoints*.get(i).Xget(),Main.*panelPoints*.get(i).Yget());  
 }  
 g.drawPolygon(j);*// отрисовка многоуголника* }  
  
  
}

**4) Класс Vector**

**package** com.visual;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
  
**public class** Vector **extends** JPanel {  
 **private int x1**;*//координаты начала вектора* **private int y1**;  
 **private int x2**;*//координаты конца вектора* **private int y2**;  
 **private int xr**;*//координаты правой части стрелки* **private int yr**;  
 **private int xl**;*//координаты левой части стрелки* **private int yl**;  
 **private double c** = Math.***PI***/6;*// угол поворота стрелки* **public** Vector(**int** x1, **int** y1, **int** x2, **int** y2){*//конструктор присвоения координат вектора* **this**.**x1** = x1;  
 **this**.**y1** = y1;  
 **this**.**x2** = x2;  
 **this**.**y2** = y2;  
 }  
 @Override  
 **public void** paint(Graphics g) {  
 Graphics2D gr2d = (Graphics2D) g;*// применение пакета Graphics2D* BasicStroke с = **new** BasicStroke(2);*// задание толщины линии* gr2d.setStroke(с);*// добавление линии* gr2d.setPaint(Color.***green***);*// задание цвета линии* **double** beta = Math.*atan2*(**y1** - **y2**, **x2** - **x1**);*// угол поворота стрелки* **int** r1 = 10;*// длина 'острия' стрелки* **int** x3 = (**int**) Math.*round*(**x2** - r1 \* Math.*cos*(beta + **c**));*//получение координат 'острия' стрелки* **int** y3 = (**int**) Math.*round*(**y2** + r1 \* Math.*sin*(beta + **c**));  
 **int** x4 = (**int**) Math.*round*(**x2** - r1 \* Math.*cos*(beta - **c**));  
 **int** y4 = (**int**) Math.*round*(**y2** + r1 \* Math.*sin*(beta - **c**));  
 g.drawLine(**x1**, **y1**, **x2**, **y2**);*// рисование вектора* g.drawLine(**x2**, **y2**, x3, y3);  
 g.drawLine(**x2**, **y2**, x4, y4);  
 }  
}

**Класс Main объединяет пакеты com.visual и problem.**

**package** com.visual;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.File;  
**import** java.io.FileNotFoundException;  
**import** java.io.PrintStream;  
**import** java.util.ArrayList;  
**import** java.util.Random;  
**import** java.util.Scanner;  
  
**public class** Main {  
 **public static** ArrayList<PanelPoint> *panelPoints* = **new** ArrayList<PanelPoint>();  
 **public static** ArrayList<Vector> *vectors* = **new** ArrayList<>();  
 **public static** ArrayList<JTextArea> *jTextAreas* = **new** ArrayList<>();  
 **public static** PolygonD *pol*;  
 **public static** Random *random* = **new** Random();  
 Scanner **fin** = **null**;  
 PrintStream **fout** = **null**;  
  
  
 **public static void** addPanelPoints(**int** k, JPanel p){*//метод добавляющий окошки для ввода координат* **if** (k==0){  
 p.removeAll();  
 p.repaint();  
 p.revalidate();  
 **return**;  
 }  
 **for** (**int** i = 0;i<k;i++) {  
 PanelPoint pp = **new** PanelPoint(i+1);  
 pp.setBounds(0,i\*50,300,50);  
 *panelPoints*.add(pp);  
 p.add(pp);  
 p.repaint();  
 }  
 }  
 **public static void** addPanelPoints(**int** k, JPanel p, Scanner fin){*// такой же как предыдущий метод, но с чтением из файла* **if** (k==0){  
 p.removeAll();  
 p.repaint();  
 p.revalidate();  
 **return**;  
 }  
 **for** (**int** i = 0;i<k;i++) {  
 PanelPoint pp = **new** PanelPoint(i+1, fin.nextInt(), fin.nextInt());  
 pp.setBounds(0,i\*50,300,50);  
 *panelPoints*.add(pp);  
 p.add(pp);  
 p.repaint();  
 }  
 }  
 **public static void** addPanelPoints(**int** k, JPanel p, **int** x0, **int** y0){*//метод с добавлением окошек и заполнением случайными координатами* **double** cor = 0.0;  
 **double** dcor = 360/k;  
 **if** (k==0){  
 p.removeAll();  
 p.repaint();  
 p.revalidate();  
 **return**;  
 }  
 **for**(**int** i =0;i<k;i++) {  
 System.***out***.println(cor);  
 **int** r = *random*.nextInt(100);  
 **int** x = x0 + (**int**)(r \* Math.*cos*(cor/180\*Math.***PI***));  
 **int** y =y0+(**int**)(r\*Math.*sin*(cor/180\*Math.***PI***));  
 PanelPoint pp = **new** PanelPoint(i+1,x,y);  
 pp.setBounds(0,i\*50,300,50);  
 *panelPoints*.add(pp);  
 p.add(pp);  
 p.repaint();  
 cor+=dcor;  
 }  
 }  
  
 **public static void** createGUI() {  
 **final** JFrame frame = **new** JFrame(**"Polygon Square"**);*// cоздание различных панелей, окошек и подписей* frame.setPreferredSize(**new** Dimension(700,700));  
  
 JPanel panel = **new** JPanel(**new** BorderLayout());  
  
 **final** Panel butPanel = **new** Panel();  
 butPanel.setLayout(**null**);  
 butPanel.setPreferredSize(**new** Dimension(300,700));  
  
 **final** Panel panelpointpane = **new** Panel();  
 panelpointpane.setLayout(**null**);  
 panelpointpane.setPreferredSize(**new** Dimension(350,700));  
  
 **final** JLabel N = **new** JLabel(**"N:"**);  
 N.setBounds(2,90,15,25);  
 butPanel.add(N);  
  
 **final** JTextField n = **new** JTextField();  
 n.setBounds(20,90, 25,25);  
 butPanel.add(n);  
  
 **final** JLabel A = **new** JLabel(**"Ответ(площадь мн-ка):"**);  
 A.setBounds(2, 320, 150, 25);  
 butPanel.add(A);  
  
 **final** JTextField a = **new** JTextField(**""**);  
 a.setBounds(140,320, 100,30);  
 butPanel.add(a);  
  
 **final** JPanel p = **new** JPanel();  
 p.setBounds(0,400,300,400);  
 butPanel.add(p);  
  
 JButton button2 = **new** JButton(**"очистить"**);*// кнопка очитски* button2.setBounds(2,360,160,30);  
 butPanel.add(button2);  
  
  
 button2.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// действия, совершаемые при нажатии на кнопку* **for** (**int** i=0;i<*panelPoints*.size();i++){  
 **while**(*panelPoints*.size() > 0) {  
 **int** index = *panelPoints*.size() - 1;  
 PanelPoint ppoint = *panelPoints*.remove(index);  
 panelpointpane.remove(ppoint);  
 panelpointpane.repaint();  
 panelpointpane.revalidate();  
 }  
 **while**(*vectors*.size() > 0) {  
 **int** index = *vectors*.size() - 1;  
 Vector vector = *vectors*.remove(index);  
 panelpointpane.remove(vector);  
 panelpointpane.repaint();  
 panelpointpane.revalidate();  
 }  
 **while**(*jTextAreas*.size() > 0) {  
 **int** index = *jTextAreas*.size() - 1;  
 JTextArea j = *jTextAreas*.remove(index);  
 panelpointpane.remove(j);  
 panelpointpane.repaint();  
 panelpointpane.revalidate();  
 }  
 }  
 panelpointpane.remove(*pol*);  
 a.setText(**""**);  
 n.setText(**""**);  
  
 }  
 });  
  
 panel.add(panelpointpane,BorderLayout.***CENTER***);  
 panel.add(butPanel,BorderLayout.***EAST***);  
 frame.getContentPane().add(panel);  
 frame.pack();  
 frame.setLocationRelativeTo(**null**);  
 frame.setVisible(**true**);  
  
 **final** JButton button3 = **new** JButton( **"Добавить колличество вершин"**);  
 butPanel.add(button3);  
 button3.setBounds(2,2,250,30);  
  
 button3.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// создаются окошки для ввода координат* **int** K = (!n.getText().equals(**""**)?Integer.*parseInt*(n.getText()):0);  
 *addPanelPoints*(K, p);  
 }  
 });  
  
 JButton button5 = **new** JButton(**"посчитать площадь"**);  
 button5.setBounds(2,270,250,30);  
 butPanel.add(button5);  
  
 button5.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// подсчет площади многоугольника по алгоритму, рисование радиус векторов и подписывание координат* **int** l = (!n.getText().equals(**""**)?Integer.*parseInt*(n.getText()):0);  
 problem.Polygon polygon = **new** problem.Polygon(l,Main.*panelPoints*);  
 **double** s = polygon.getArea(polygon);  
 a.setText(**""**+s);  
 **for**(**int** i = 0;i<l; i++) {  
 **int** x = Main.*panelPoints*.get(i).Xget();  
 **int** y = Main.*panelPoints*.get(i).Yget();  
 Vector v = **new** Vector(0,0, x, y);  
 v.setBounds(0, 0, 350, 700);  
 *vectors*.add(v);  
 panelpointpane.add(v);  
 panelpointpane.repaint();  
 **int** b =i+1;  
 JTextArea j = **new** JTextArea(**"A"**+b+**"("**+x+**","**+y+**")"**);  
 j.setBounds(x,y,80,20);  
 *jTextAreas*.add(j);  
 panelpointpane.add(j);  
 panelpointpane.repaint();  
 }  
 }  
 });  
  
 JButton button6 = **new** JButton(**"задать случайные вершины"**);  
 button6.setBounds(2,50,250,30);  
 butPanel.add(button6);  
  
 button6.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// заполнение окошек случайными координатами, рисование многоугольника* **int** k = (!n.getText().equals(**""**)?Integer.*parseInt*(n.getText()):0);  
 **int** x0 = *random*.nextInt(300);  
 **int** y0 = *random*.nextInt(300);  
 *addPanelPoints*(k, p ,x0,y0);  
 *pol* = **new** PolygonD(k,Main.*panelPoints*);  
 *pol*.setBounds(0,0,350,700);  
 panelpointpane.add(*pol*);  
 panelpointpane.repaint();  
 }  
 });  
  
 JButton button7 = **new** JButton(**"чтение из файла"**);  
 button7.setBounds(2,160,160,30);  
 butPanel.add(button7);  
  
 button7.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// чтение координат вершин из файла* Scanner fin = **null**;  
 PrintStream fout = **null**;  
 **try** {  
 fin = **new** Scanner(**new** File(**"in.txt"**));  
 fout = **new** PrintStream(**new** File(**"out.txt"**));  
 **int** k = fin.nextInt();  
 n.setText(**""**+k);  
 *panelPoints*.clear();  
 *addPanelPoints*(k, p, fin);  
 }**catch**(FileNotFoundException e1){  
 System.***out***.println(**"не найден файл "** + e);  
 }**finally** {  
 **if** (fin != **null**) {  
 fin.close(); *//!* }  
 **if** (fout != **null**) {  
 fout.close(); *//!* }  
 }  
 }  
 });  
  
 JButton button8 = **new** JButton(**"вывод в файл"**);  
 button8.setBounds(2,200,160,30);  
 butPanel.add(button8);  
  
 button8.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// запись координат вершин в файл* Scanner fin = **null**;  
 PrintStream fout = **null**;  
 **try** {  
 fin = **new** Scanner(**new** File(**"in.txt"**));  
 fout = **new** PrintStream(**new** File(**"out.txt"**));  
 System.***out***.println(*panelPoints*.size());  
 fout.println(*panelPoints*.size());  
 **for** (PanelPoint p : *panelPoints*) {  
 System.***out***.println(p.Xget() + **" "** + p.Yget());  
 fout.println(p.Xget() + **" "** + p.Yget());  
 }  
 } **catch** (FileNotFoundException e1) {  
 System.***out***.println(**"не найден файл "** + e);  
 }**finally** {  
 **if** (fin != **null**) {  
 fin.close(); *//!* }  
 **if** (fout != **null**) {  
 fout.close(); *//!* }  
 }  
 }  
 });  
 JButton button9 = **new** JButton(**"задать мн-к"**);  
 button9.setBounds(170, 360,120,30);  
 butPanel.add(button9);  
  
 button9.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {*// рисование многоугольника, если вершины заданы не случайным образом, а вручную* **int** k = (!n.getText().equals(**""**)?Integer.*parseInt*(n.getText()):0);  
 *pol* = **new** PolygonD(k,Main.*panelPoints*);  
 *pol*.setBounds(0,0,350,700);  
 panelpointpane.add(*pol*);  
 panelpointpane.repaint();  
 }  
 });  
  
 }  
  
  
  
 **public static void** main(String[] args) {  
 javax.swing.SwingUtilities.*invokeLater*(**new** Runnable() {  
 **public void** run() {  
 JFrame.*setDefaultLookAndFeelDecorated*(**true**);  
 *createGUI*();  
 }  
 });  
 }  
}

**6.Пример работы программы**

**6.1** Исходные данные **6.2** Выходные данные

5 3319.5

199 82

166 149

125 96

138 77

151 63