**ГБОУ “Президентский ФМЛ 239”**

**Нахождение правильных треугольников**

**Годовой проект по информатике**

**Павлов Михаил 10-1**

**1. Постановка задачи.**

Дано множество точек на плоскости. Определить среди них множество точек наибольшего размера такое, что каждая точка этого множества является вершиной равностороннего треугольника, вершины которого принадлежат этому множеству, вывести на экран это множество и все равносторонние треугольники.

**2. Уточнение исходных и выходных данных.**

Во входном файле points.txt хранятся координаты исходных точек. Количество точек не известно. Координаты точек по модулю не превышают 1. Необходимо вывести на экран все равносторонние треугольники, а также их вершины.

**3. Математическая модель.**

Ax, Ay, Bx, By, Cx, Cy – координаты вершин треугольника.

(Ax - Bx)2 + (Ay – By)2 ==(Ax - Cx)2 + (Ay – Cy)2 ==(Cx - Bx)2 + (Cy – By)2 – проверка треугольника на правильность.

**4. Анализ используемой структуры данных.**

Так как я планирую перебирать все тройки точек, каждую точку нужно хранить. Поскольку количество точек не известно, я буду хранить их в динамическом массиве. Количество точек и треугольников в ответе не известно, значит их тоже надо хранить в динамических массивах.

**5. Выбор метода решения.**

Создадим класс triangle. Переберем все тройки точек, по каждой тройке точек построим треугольник и проверим является ли он равносторонним. Если является, то занесём его и его вершины в ответы.

**6. Комментированный листинг.**

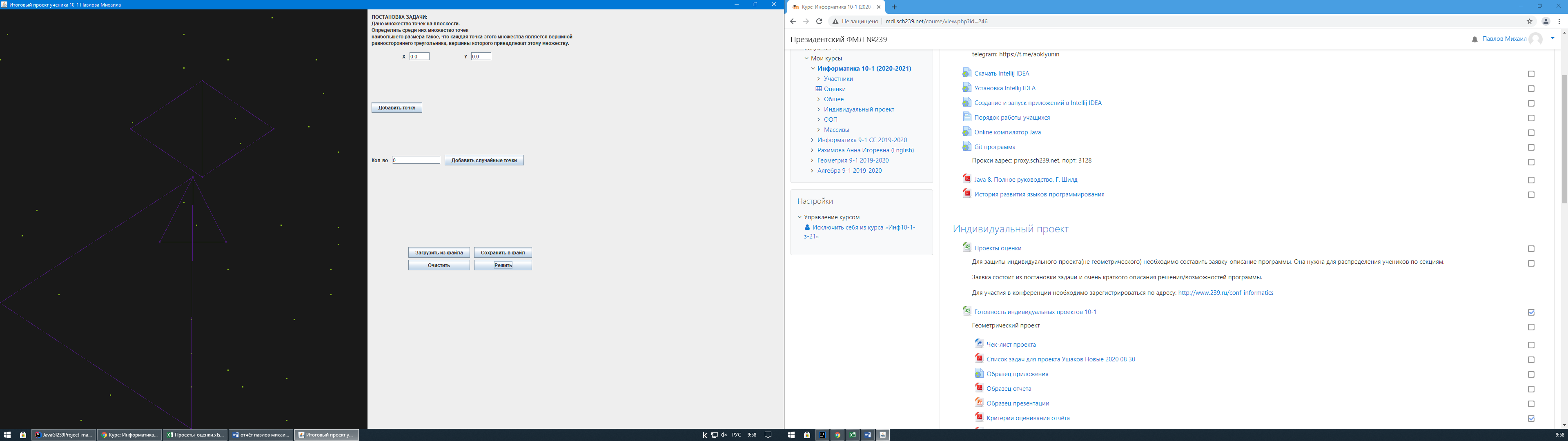
resultTriangles.clear();  
resultPoints.clear();  
// перебираем тройки точек  
for (int i = 0; i < points.size(); i++) {  
 for (int j = i + 1; j < points.size(); j++) {  
 for (int k = j + 1; k < points.size(); k++) {  
 Point p1 = points.get(i);  
 Point p2 = points.get(j);  
 Point p3 = points.get(k);  
 Triangle t = new Triangle(p1, p2, p3);  
 //для каждого треугольника проверяем является ли он правильным  
 if (t.regular()) {  
 //если треугольник правильный заносим его и его вершины в ответ  
 resultTriangles.add(t);  
 resultPoints.add(p1);  
 resultPoints.add(p2);  
 resultPoints.add(p3);  
 }  
 }  
 }  
}

**7.**  **Пример работы программы.**

Исходные данные:

-0,80 0,04  
0,04 -0,48  
0,56 -0,76  
-1,00 -0,40  
-0,56 -0,96  
-0,28 0,46  
-0,88 -0,08  
0,04 -0,64  
0,68 0,44  
0,76 0,60  
-0,68 -0,36  
0,00 0,08  
0,80 -0,24  
0,84 -0,04  
0,60 -0,48  
0,00 0,72  
0,24 -0,24  
0,00 0,88  
0,04 -0,80  
0,24 0,76  
0,04 -1,00  
0,80 -0,36  
-1,00 0,76  
0,72 0,76  
0,28 0,48  
0,56 -0,96  
0,32 -0,80  
0,84 0,32  
-0,40 -0,84  
-0,96 0,88  
0,48 0,96  
0,84 -0,12  
0,24 -0,72  
0,49 0,43  
0,10 0,20  
0,10 0,66  
-0,29 0,43  
0,31 0,36  
0,07 -0,03  
0,53 -0,03  
-0,13 -0,11  
0,23 -0,11  
0,05 0,20

Выходные данные:



**8. Анализ правильности решения.**

Переберём все тройки точек из примера вручную и убедимся в правильности работы программы