ГБОУ "Президентский ФМЛ №239"

Нахождение треугольника наименьшей площади

Отчёт по годовому проекту

Ученик: Берхман Евгений Юрьевич
Преподаватель: Клюнин Алексей Олегович

Класс: 10-3

Санкт-Петербург
2017

Содержание

1 Постановка задачи

На плоскости заданно множество точек. Выбрать из них такие три точки, не лежащие на одной прямой, которые составляют треугольник наименьшей площади.

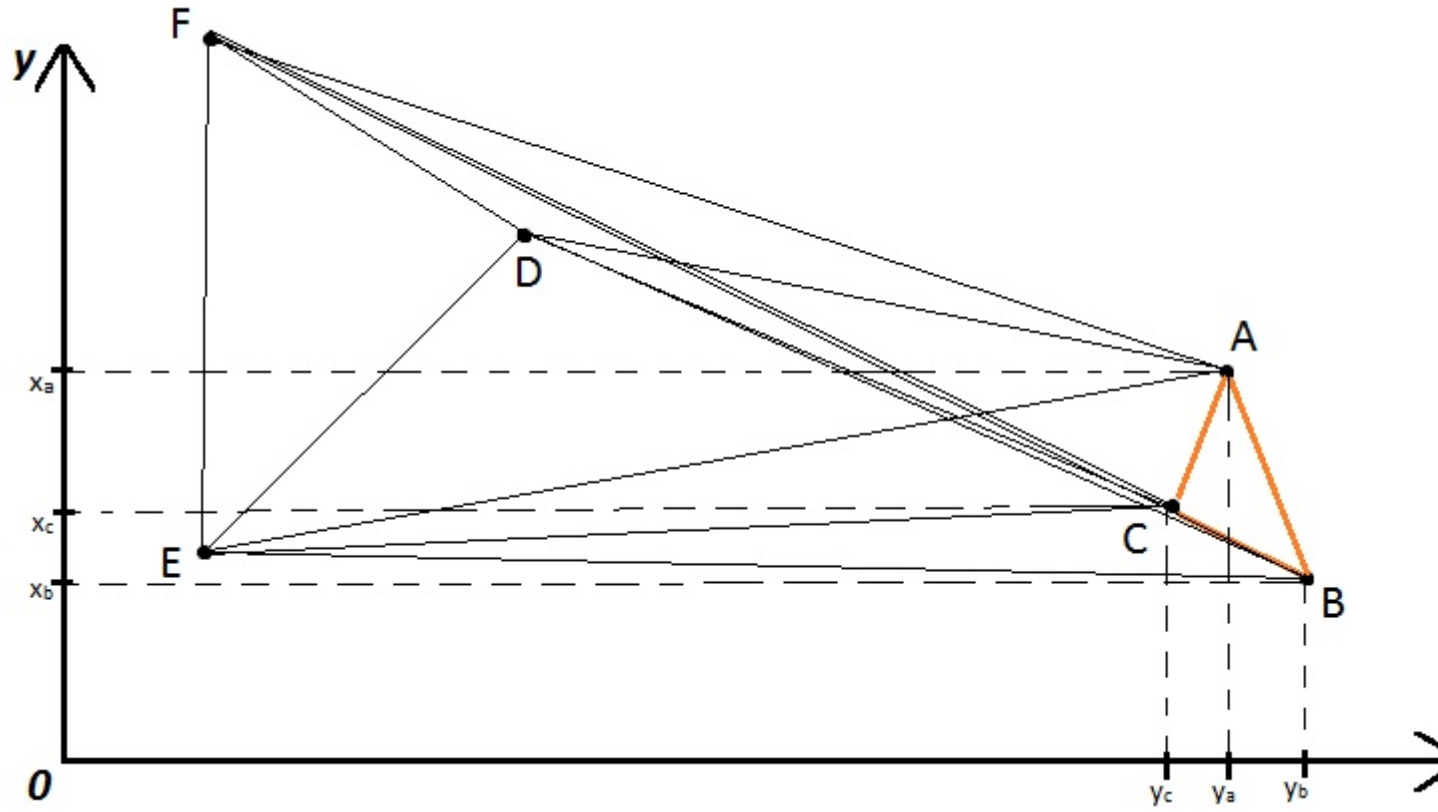


Рис. 1: **Множество из шести точек, где точки A, B и C-искомые, образуют треугольник наименьшей площади.**

2 Уточнение исходных и выходных данных и ограничений на них

2.1 Исходные данные

Будем решать задачу в системе координат. С клавиатуры на вход подаётся число $\mathbf{n}$ типа int, количество данных точек($\mathbf{n \geqslant 3}$). Также введем переменную **min**, которой будет присваиваться наименьшее значение площади. Для каждого из $\mathbf{n}$ экземпляров класса **Point** с клавиатуры считываются (или случайным образом. пользователь может выбрать вариант: вводить координаты самому с клавиатуры или предоставить компьютеру выбрать их случайным образом, для этого будет создана переменная типа $boolean$) значения переменных $\mathbf{x_n}$ и $\mathbf{y_n}$, координаты точек на плоскости.

2.2 Выходные данные

Необходимо вывести на экран значение площади для треугольника, имеющего её минимальную. Помимо этого на рисунке будут выделены те три точки, которые образуют этот треугольник. При желании пользователь может вывести координаты этих точек в файл (output.txt, он будет лежать в одной папке с программой) нажатием кнопки.

3 Выбор метода решения

3.1 Базовые структуры данных

Класс **Point** описывает точку, состоит из двух полей $\mathbf{x_n}$ и $\mathbf{y_n}$ типа $double$, задающих координаты точки на плоскости.

Класс **Set** описывает множество точек, состоит из двух полей: **k** типа int(в данной задаче **k** всегда равна 3, так как три точки образуют треугольник) и массив состоящий из **k** экземпляров класса **Point**.

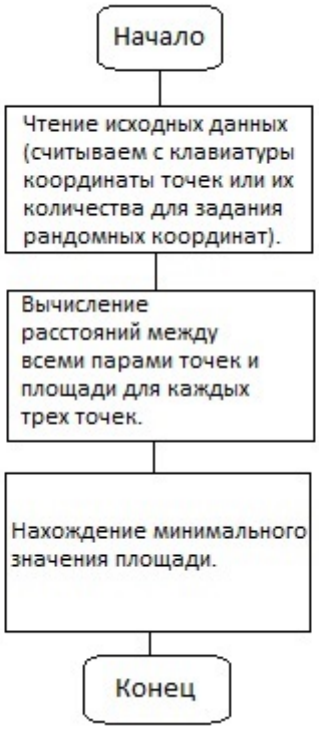
3.2 Описание алгоритма

Будем решать задачу в системе координат. Создадим $\mathbf{C_n^3}$ ($\mathbf{C_n^3} = \frac{\mathbf{n!}}{\mathbf{3!*(n-3)!}}$) экземпляров класса **Set**, состоящих из 3-х точек(**Point**). С помощью метода $square$ получим значение площади для каждого из цэ треугольников, т.е. для каждого из **Set**'ов. Опишем метод: будем считать 3 расстояния для каждого экземпляра: от точки $\mathbf{P_a}$ до точки $\mathbf{P_b}$, от точки $\mathbf{P_b}$ до точки $\mathbf{P_c}$ и от точки $\mathbf{P_c}$ до точки $\mathbf{P_a}$. Получив данные значения длины трех сторон по формуле $L_a = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$-расстояние от $\mathbf{P_a}$ до $\mathbf{P_b}$(далее аналогично), посчитаем значение площади треугольника по формуле Герона: $S = \sqrt{p(p - L_a)(p - L_b)(p - L_)}$, где $p = (L_a + L_b + L_c)/2$. Будем каждый раз сравнивать значение площади **Set.square** со значением переменной **min** (с самого начала присвоим **min** значение площади первого **Set**'а), и если новое значение меньше, то будем присваивать его переменной **min**. Проверив все $\mathbf{C_n^3}$ вариантов получим конечное значение **min**. 3 точки, образующие треугольник, соответствующий данному значению переменной **min**, и будут искомыми. Ответом на данную задачу является значение минимальной площади. При этом на плоскости будут отображены те три точки, которые образуют треугольник наименьшей площади.

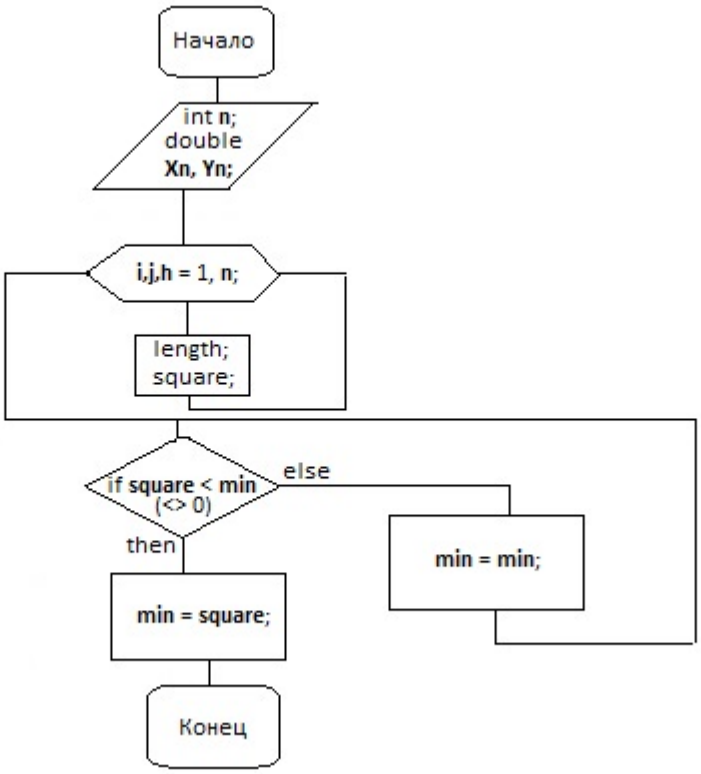
Примечания: Для каждого **Set**'а площадь не должна быть нулевой, иначе данный **Set** противоречит условию(3 токчи лежат на одной пряой). То есть при создании **Set**'а, если его площадь равно 0, то значение переменной **min** не меняется, а именно не становится нулевым.

4 Построение алгоритма

4.1 Обобщенная блок-схема алгоритма



4.2 Блок-схема алгоритма



5 Листинг программы

```
double min = -1; // значение минимальной площади
MySet answertriangle = new MySet(); //создадим треугольник минимальной площади
Point[] points2 = new Point[points.size()];
int n = points.size();
for (int i = 0; i < n; i++) points2[i] = points.get(i);
for (int i = 0; i < n; i++)
for (int j = 0; j < n; j++)
for (int h = 0; h < n; h++)
if (i == j || i == h || j == h) continue; // проверим, что мы выбрали три различные точки
MySet triangle = new MySet(3, points2[i], points2[j], points2[h]); // создадим треугольник
с данными тремя вершинами
double square = triangle.Square(); // найдем его площадь
if (square > 0 'and' (min == -1 || min > square))
-ный(площадь больше нуля) и его площадь меньше, чем та что была
min = square; // обновим значение минимальной площади
answertriangle = triangle.myCopy();
```

6 Пример работы программы

6.1 Исходные данные

```
123 123
123 456
56 74
156 90
125 76
65 13
145 200
231 345
461 398
423 358
10 53
```

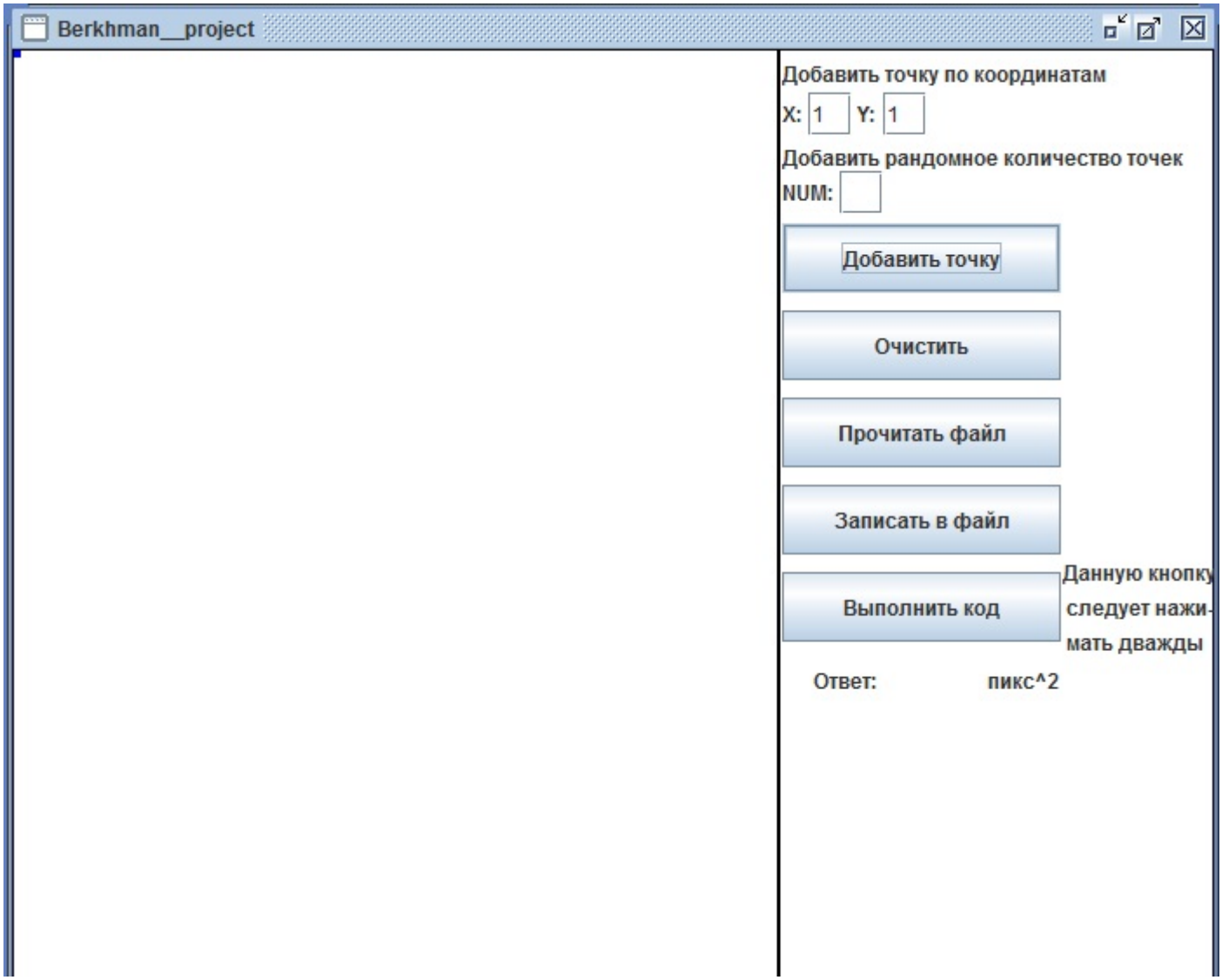
6.2 Выходные данные

```
198.0
123 456
156 90
145 200
```

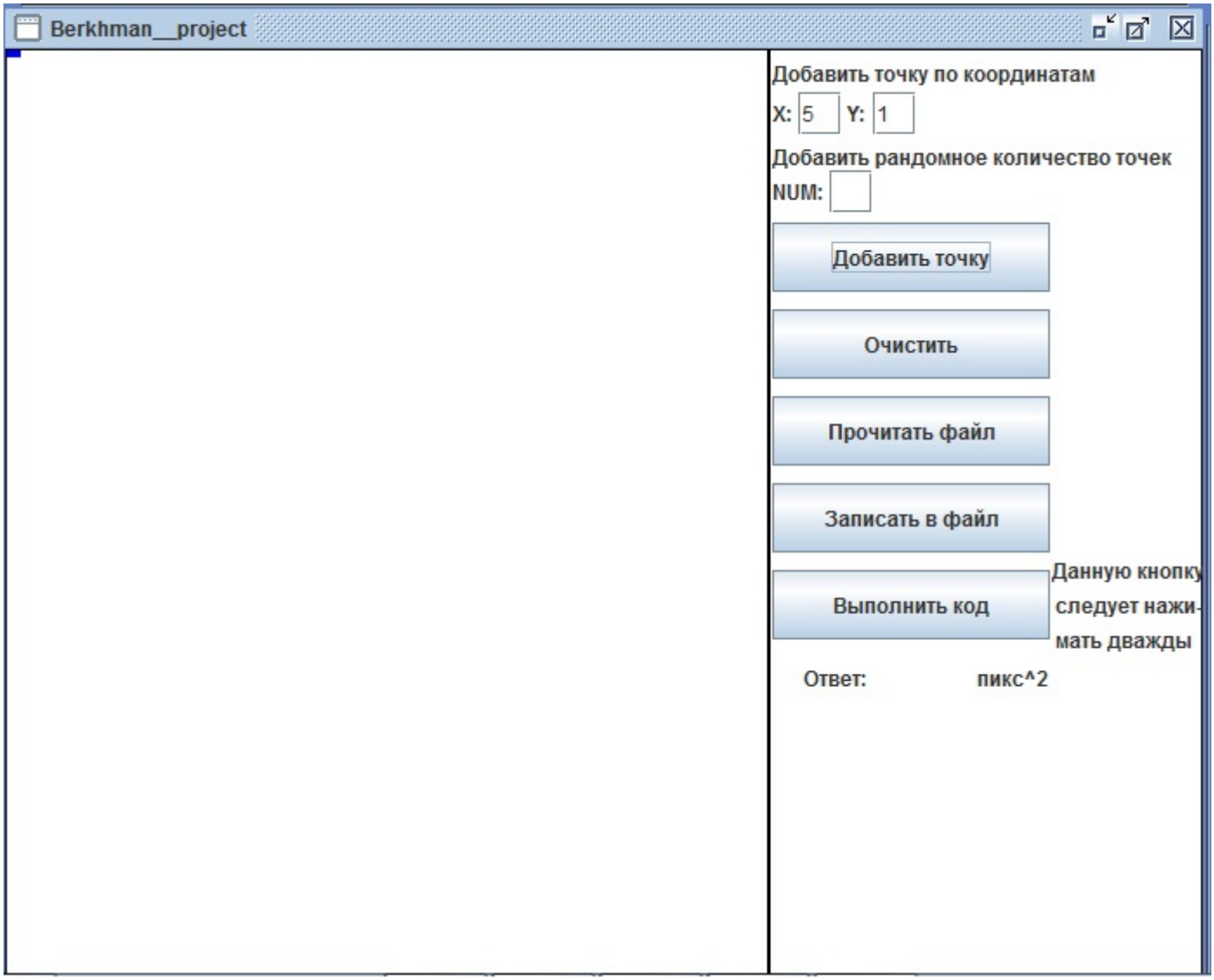
7 Анализ правильности решения

Чтобы убедиться в праильности решения выберем четыре точки, для которых будет очевидно нахождение треугольника наименьшей площади, и посчитаем это значение. Затем убедимся в том, что программа выдает те же значения.

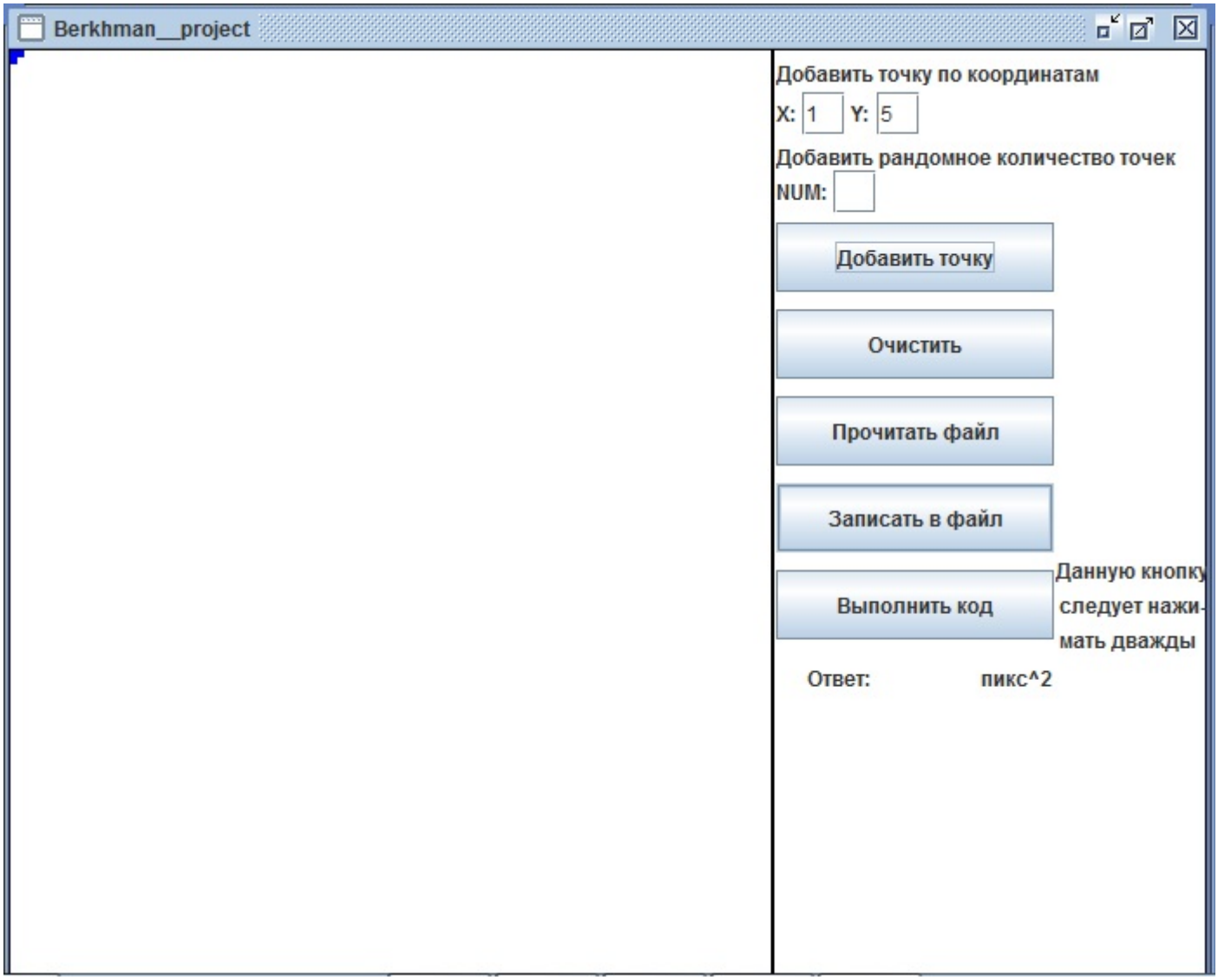
Создадим вручную три точки:



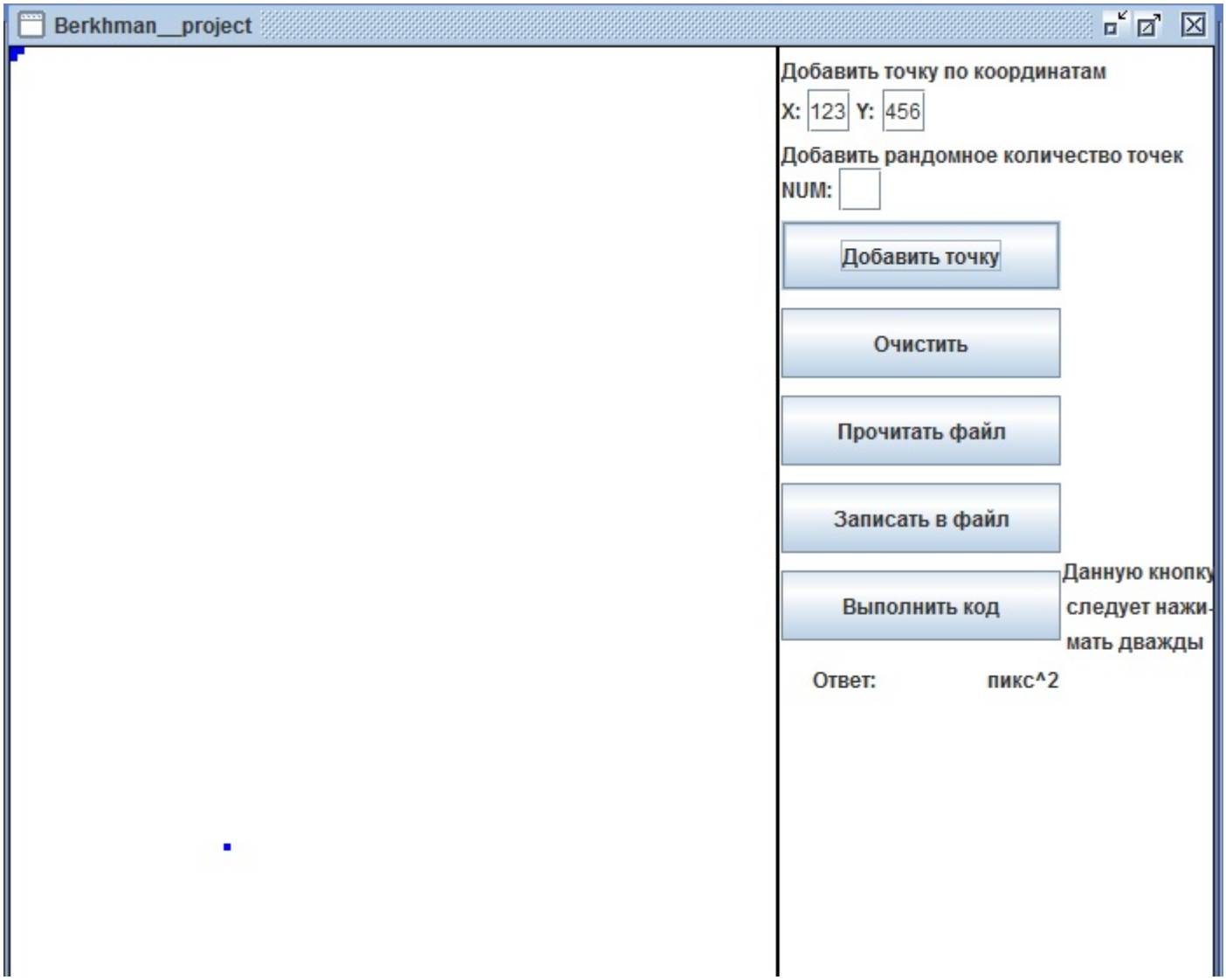
первая точка с координатами $(\mathbf{1};\mathbf{1})$,



вторая точка с координатами $(\mathbf{5}; \mathbf{1})$,

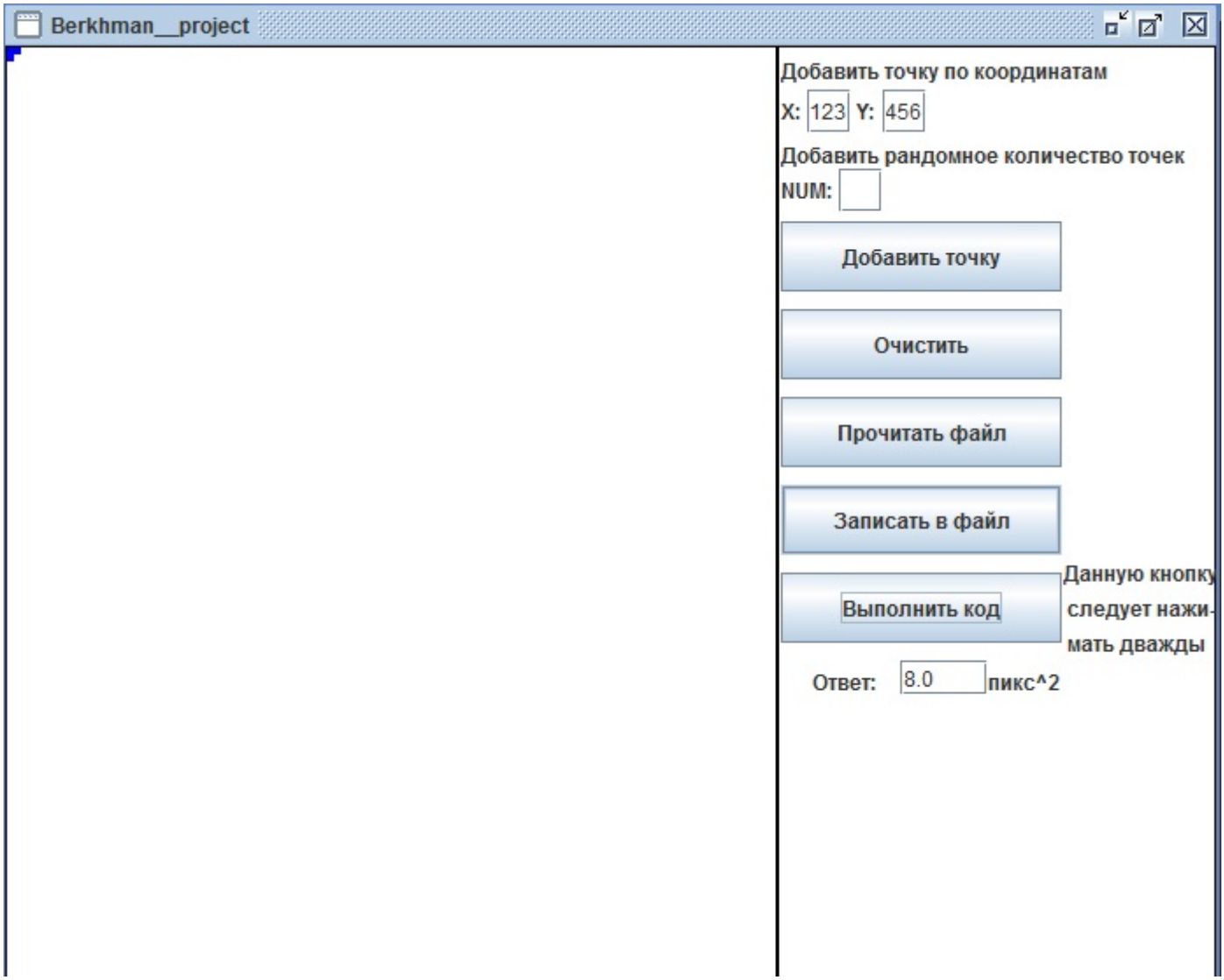


третья точка с координатами $(\mathbf{1}; \mathbf{5})$

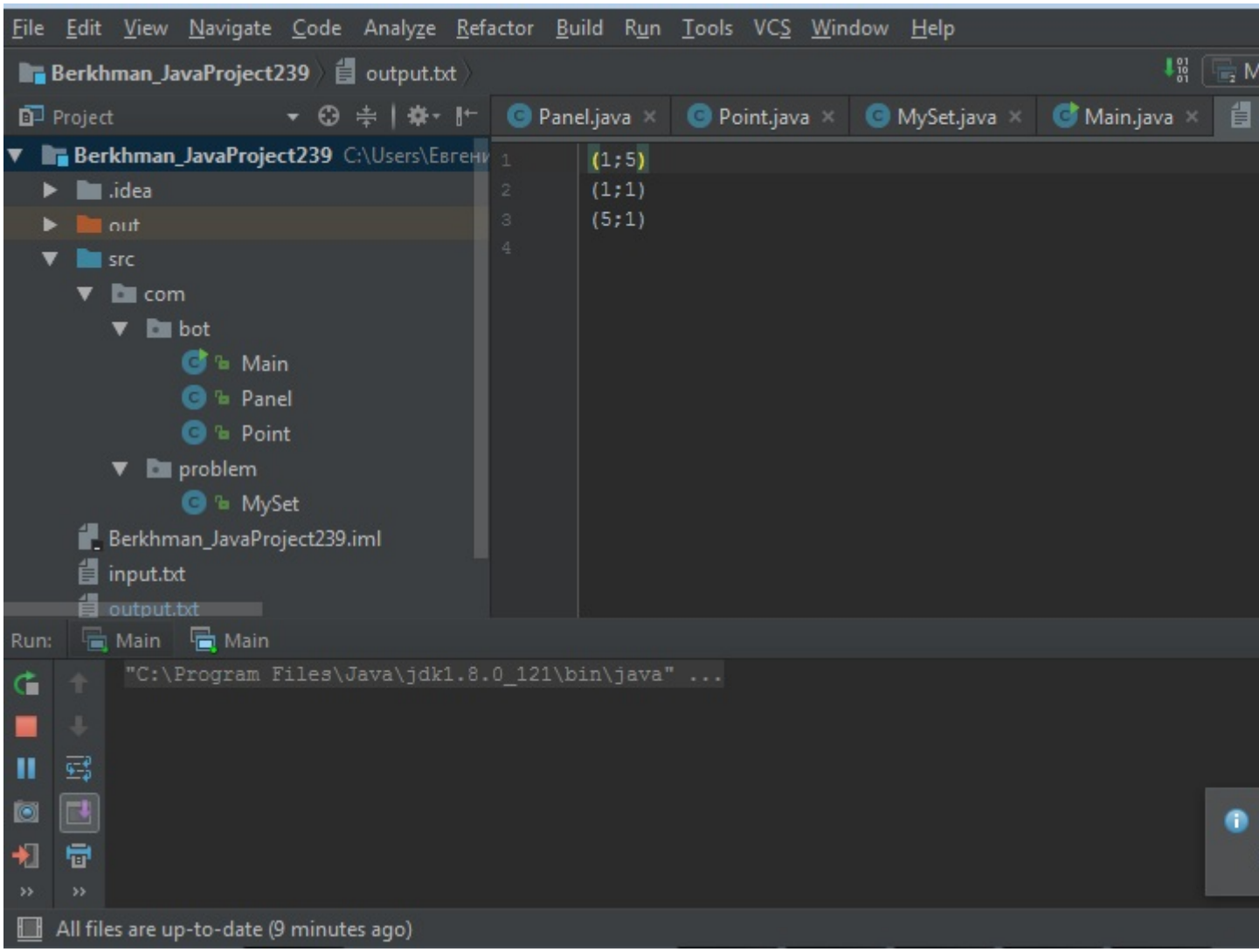


и четвёртая точка с координатами $(\mathbf{123}; \mathbf{456})$.

Очевидно, что наименьшую площадь образует треугольник, заданный первыми тремя точками. Посчитаем его площадь. Первая сторона равна: $\sqrt{(5-1)^2+(1-1)^2} = 4$, вторая сторона: $\sqrt{(1-5)^2+(5-1)^2} = 4\sqrt{2}$ и третья сторона: $\sqrt{(1-1)^2+(5-1)^2} = 4$. Теперь посчитаем площадь(облегчили вычисления тем, что треугольник прямоугольный). Площадь равна: $\frac{(4*4)}{2} = 8$.



Запустив код, убедимся в том, что треугольник выбран правильно и площадь посчитана верно.



Теперь проверим правильность выбранных точек, а именно, проверим, что будет при выводе в файл(там толжны быть отображены координаты тех трех точек, которые образуют треуггольник наименьшей площади).

Таким образом на простом примере мы убедились в корректной работе описываемой выше программы.