ГБОУ "Президентский ФМЛ № 239"

Нахождение и визуализация расстояний между точками на плоскости

Годовой проект по информатике

Работу выполнила ученица 10-2 класса
Меркулова Маргарита

Цель проекта

- Нахождение пары точек с максимальным расстоянием между ними во множестве точек на плоскости;
- Нахождение расстояния между этими точками;
- Визуализация решения.

Задачи проекта

- Решение поставленной геометрической задачи;
- Работа с графикой;
- Изучение языка программирования Java SE 9;
- Изучение среды разработки IntelliJ IDEA.

Используемые технологии

- Язык программирования Java SE 9;
- Среда разработки IntelliJ IDEA.

Исходные данные

С клавиатуры вводится целое число n (int n) равное заданному количеству точек на плоскости, затем в созданные массивы координат x, y (double x[n], double y [n]) с клавиатуры вводятся значения координат по осям x, y соответственно.

Выходные данные

На экран выводятся два целых числа (0<=(int ii, jj)<n), являющиеся номерами в массивах (x, y) точек с максимальным расстоянием между ними; расстояние между этими точками- число типа double (double max); график-окно с прямоугольной системой координат, на котором изображены все возможные отрезки между заданными точками, отрезок максимальной длины выделен другим цветом.

Алгоритм решения

Создание пустого окна, изображение координатных осей, задание масштаба.

Создание переменной типа double- максимального расстояния, которое надо найти, ii, jj типа int- номеров в массивах точек, между которыми расстояние максимально.

Изначально ii=0; jj=0; max=0.

Прохождение по всем парам точек, сравнивание расстояний между каждой парой (k, m) точек со значением переменной max, если max меньше этого расстояния, присваивание переменной max значение расстояния между этими двумя точками (k, m), присваивание переменным ii,jj значения, равные k, m – номерам точек (в массивах координат x,y).

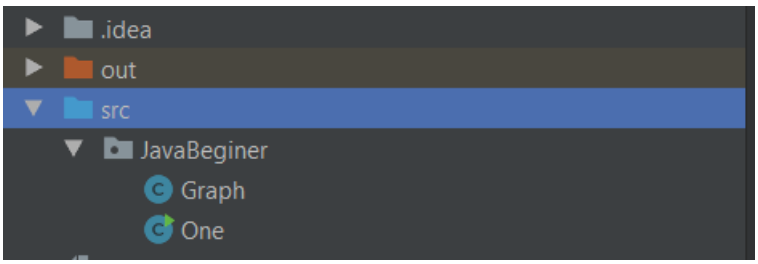
При рассмотрении каждой пары точек (k, m) также происходит изображение отрезка, соответствующего этой паре (x[k], y[k], x[m], y[m]).

После нахождения максимального расстояния и соответствующих точек (ii, jj) другим цветом изображение отрезка (x[ii], y[ii], x[jj], y[jj]).

Структура проекта

Использование объектно-ориентированного программирования:

- в классе One создание окна, задача его размеров, прозрачности;
- в классе Graph все вычисления, все методы для рисования в созданном окне.



Математическая модель

Расстояние между точками с координатами x[i], y[i], x[j], y[j] в прямоугольной системе координат по теореме Пифагора равно $r=\sqrt{(x[i]-x[j])^2+(y[i]-y[j])^2}$.

Листинг

Класс Graph:

```
package JavaBeginer; //пакет

import java.awt.*; //импорт библиотеки для создания графического интерфейса
import java.util.Scanner; //импорт сканера
import javax.swing.*; //импорт библиотеки для создания графического интерфейса

public class Graph extends JPanel { //создаем класс, наследуемый от класса JPanel
   Scanner sc = new Scanner(System.in); //создание объекта класса Scannerдля ввода с клавиатуры

   public void paintComponent(Graphics g)//в параметрах объект класса Graphics {
      super.paintComponent(g); //вызов конструктора суперкласса
      this.setBackground(Color.WHITE);//задаем белый цвет окна
      int n = sc.nextInt();//считываем с клавиатуры значение переменной, хранящей количество
точек

      double x[] = new double[n];//создаем массив, хранящий координаты точек по оси x
      double y[] = new double[n];  //создаем массив, хранящий координаты точек по оси y
      for (int a = 0; a < n; a++) {
         x[a] = sc.nextDouble();
      } //считываем массив x с клавиатуры
      for (int a = 0; a < n; a++) {
         y[a] = sc.nextDouble();
      } //считываем массив y с клавиатуры
      double max = 0; //задаем переменную, ранящую максимальное расстояние
      int ii = 0;
      int jj = 0;//задаем индексы (номера) точек, между которыми расстояние максимально

       g.setColor(Color.GREEN); //задаем цвет отрезков

//перебираем все варианты пар точек i,j:
      for (int i = 0; i < n; i++) {
         for (int j = 1; j < n; j++) {
            g.drawLine((int) (x[i] + this.getWidth() / 2), (int) (-y[i] + this.getHeight() / 2), (int) (x[j] +
this.getWidth() / 2), (int) (-y[j]) + this.getHeight() / 2); //рисуем отрезок с заданными координатами
            if (max < Math.sqrt(Math.pow(x[i] - x[j], 2) + Math.pow(y[i] - y[j], 2))) {
               max = Math.sqrt(Math.pow(x[i] - x[j], 2) + Math.pow(y[i] - y[j], 2));
               ii = i;
               jj = j;
            } // сравниваем расстояние между каждой парой точек с int max, если max меньше этого
расстояния, присваиваем его значение переменной max, и присваиваем ii=i, jj=j
         }
      }
      g.setColor(Color.MAGENTA);//задаем новый цвет для отрезка максимальной длины
      g.drawLine((int) (x[ii] + this.getWidth() / 2), (int) (-y[ii] + this.getHeight() / 2), (int) (x[jj] + this.getWidth()
/ 2), (int) (-y[jj]) + this.getHeight() / 2); //рисуме отрезок максимальной длины
      System.out.println(ii + " " + jj); //вывод номеров точек, расстояние между которыми
максимально
      System.out.println(max);//вывод максимального расстояния
```

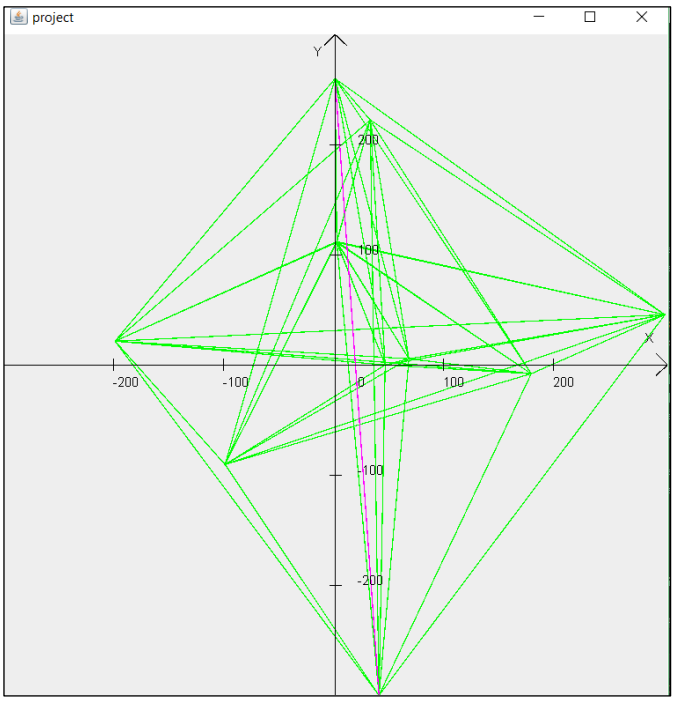
```
Далее везде используем методы класса Graphics:
        g.setColor(Color.BLACK);//задаем цвет координатных осей
        g.drawLine(0, this.getHeight() / 2, this.getWidth(), this.getHeight() / 2);//рисуем ось x
        g.drawLine(this.getWidth() / 2, 0, this.getWidth() / 2, this.getHeight());//рисуем ось y
        int dx = 5;//задаем дополнительную переменную для рисования масштаба
        g.drawLine(this.getWidth() / 2, 0, 2 * dx + this.getWidth() / 2, 2 * dx);
        g.drawLine(this.getWidth() / 2, 0, this.getWidth() / 2 - 2 * dx, 2 * dx);
        g.drawLine(this.getWidth(), this.getHeight() / 2, this.getWidth() - 2 * dx, this.getHeight() / 2 - 2 * dx);
        g.drawLine(this.getWidth(), this.getHeight() / 2, this.getWidth() - 2 * dx, this.getHeight() / 2 + 2 * dx);
        g.drawLine(100, this.getHeight() / 2 - dx, 100, this.getHeight() / 2 + dx);
        g.drawLine(200, this.getHeight() / 2 - dx, 200, this.getHeight() / 2 + dx);
        g.drawLine(400, this.getHeight() / 2 - dx, 400, this.getHeight() / 2 + dx);
        g.drawLine(500, this.getHeight() / 2 - dx, 500, this.getHeight() / 2 + dx);
        g.drawLine(this.getWidth() / 2 - dx, 100, this.getWidth() / 2 + dx, 100);
        g.drawLine(this.getWidth() / 2 - dx, 200, this.getWidth() / 2 + dx, 200);
        g.drawLine(this.getWidth() / 2 - dx, 400, this.getWidth() / 2 + dx, 400);
        g.drawLine(this.getWidth() / 2 - dx, 500, this.getWidth() / 2 + dx, 500);

//рисуем направления осей x, y; рисуем масштабные риски на осях
        g.drawString("Y", this.getWidth() / 2 - 4 * dx, 4 * dx);
        g.drawString("X", this.getWidth() - 4 * dx, this.getHeight() / 2 - 4 * dx);//подписываем оси

        g.drawString("0", this.getWidth() / 2 + 4 * dx, this.getHeight() / 2 + 4 * dx);
        g.drawString("100", 400, this.getHeight() / 2 + 4 * dx);
        g.drawString("200", 500, this.getHeight() / 2 + 4 * dx);
        g.drawString("-100", 200, this.getHeight() / 2 + 4 * dx);
        g.drawString("-200", 100, this.getHeight() / 2 + 4 * dx);
        g.drawString("100", this.getWidth() / 2 + 4 * dx, 200);
        g.drawString("200", this.getWidth() / 2 + 4 * dx, 100);
        g.drawString("-100", this.getWidth() / 2 + 4 * dx, 400);
        g.drawString("-200", this.getWidth() / 2 + 4 * dx, 500);
//задаем масштаб

    }
}
```

Класс One:

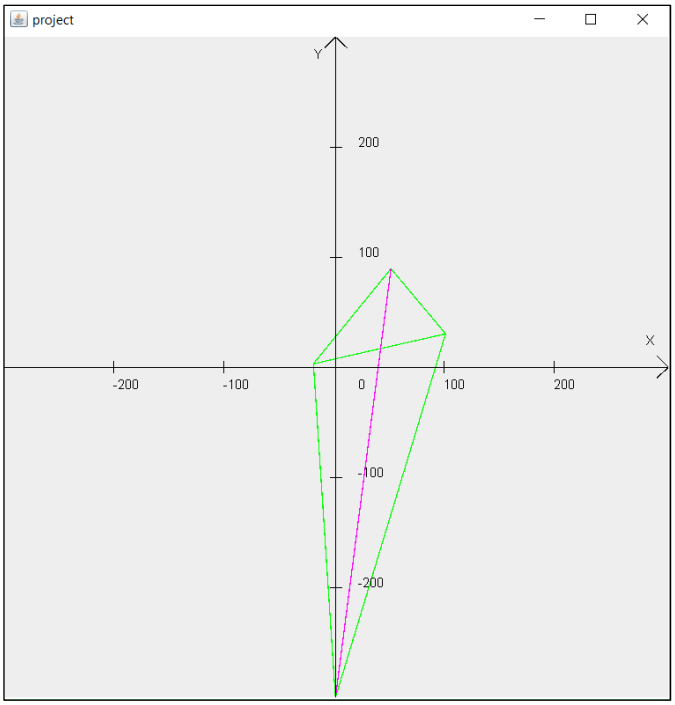
```
package JavaBeginer;//пакет

import javax.swing.*;// импорт библиотеки для создания графического интерфейса


public class One {
   public static void main(String[] args) {
      JFrame frame = new JFrame("project");//создаем объект класса JFrame-пустое окно
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//добавляем функцию закрытия окна при
нажатии “x”
      JavaBeginer.Graph a = new JavaBeginer.Graph();
      frame.add(a);//добавляем окно
      frame.setSize(600+20, 600+40);//задаем размер окна
      frame.setVisible(true);//задаем прозрачность окна
   }
}
```

Пример работы программы

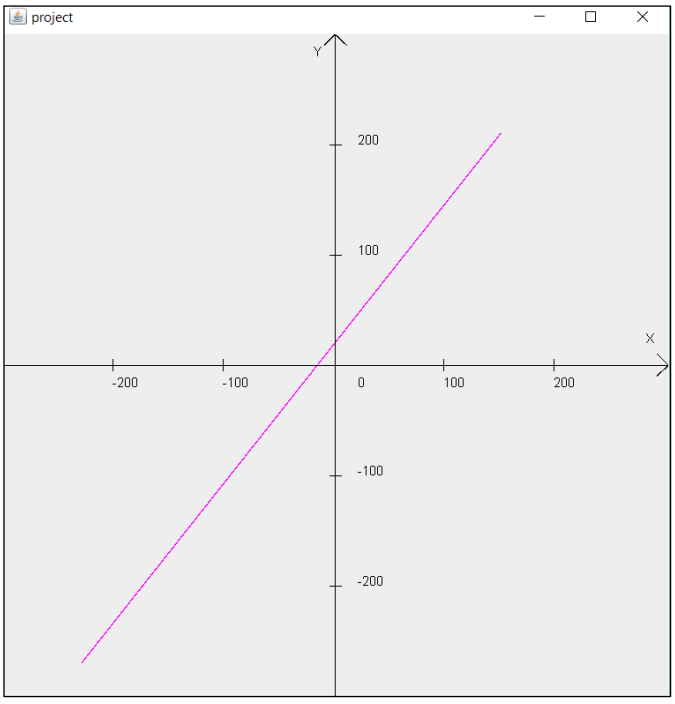
Входные данные:
int n=10;
double x[]={40; 300; -100; 67,1; 45; 0; -200; 178; 32; 1};
double y[]={-300; 45; -91; 5; 0; 260; 23; -8; 222; 111,1};
Выходные данные:
int ii=0; int jj=5;
double max=561,426753904728



Входные данные:
int n=4;
double x[]={100,5; 0; 50;-20};
double y[]={30; -300; 89; 3};
Выходные данные:
int ii=1; int jj=2;
double max=392,2002039775094



Входные данные:
int n=2;
double x[]={150; -230};
double y[]={210; -270};
Выходные данные:
int ii=0; int jj=1;
double max=612,209114600558



Возникшие трудности

- выбор графических библиотек, выбрала java.swing и java.awt;
- в размеры созданного пустого окна входит рамка, задавая его размеры, как (a; b) получалось поле для рисования меньшего размера, исправила, добавив нужное количество пикселей в ширину (20) и в высоту (40).

Анализ правильности решения

На полученном графике изображены все рассматриваемые точки плоскости и всевозможные отрезки, соединяющие их, отрезок максимальной длины получен, что наглядно видно на графике, также получены длина этого отрезка и номера точек, расстояние между которыми максимально.