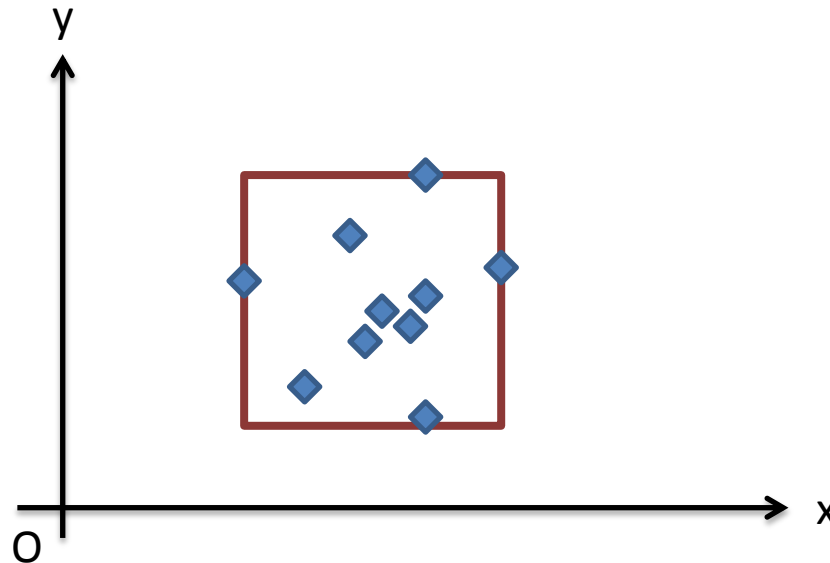


プログラミング基礎

第8.5回

課題

- 100個の点群(2次元)が与えられた際, それらを内包するXY軸に沿った矩形の大きさを求めよ



解き方

1: mainメソッド (RectangleFromPointsクラス)

```
public class RectangleFromPoints
{
    static public void main(String[] args)
    {
        int numps = 100;
        java.util.Random rand = new java.util.Random();

        Points points = new Points(numps);
        //numps個の点を生成
        for(int i = 0; i < numps; i++)
        {
            double x, y;
            Vector2D p;
            x = rand.nextGaussian() * 100.0;//ランダムな値を生成
            y = rand.nextGaussian() * 100.0;//ランダムな値を生成
            p = new Vector2D(x, y);
            System.out.println(x + ", " + y);
            points.setPoints(i, p);
        }
        //点群から最小, 最大のx, yを得る
        double minx, maxx, miny, maxy;
        minx = points.getMinX();
        maxx = points.getMaxX();
        miny = points.getMinY();
        maxy = points.getMaxY();

        System.out.println();
        System.out.println(minx + ", " + miny);
        System.out.println(minx + ", " + maxy);
        System.out.println(maxx + ", " + maxy);
        System.out.println(maxx + ", " + miny);
        System.out.println(minx + ", " + miny);
    }
}
```

1: 点を表すクラス

```
Vector2D
{
    double x;
    double y;

    Vector2D(double xx, double yy)
    {
        x = xx;
        y = yy;
    }

    double getX()
    {
        return x;
    }

    double getY()
    {
        return y;
    }
}
```

2.点群を表すクラス

*コメントが書かれている箇所を自分で作る

```
class Points
{
    int numPoints;
    Vector2D[] ps;

    Points(int numps)
    {
        numPoints = numps;
        ps = new Vector2D[numPoints];
    }

    void setPoints(int i, Vector2D p)
    {
        ps[i] = p;
    }

    double getMinX()
    {
        //配列psの中から最小のXを探し, 戻り値とする
    }

    double getMaxX()
    {
        //配列psの中から最大のXを探し, 戻り値とする
    }

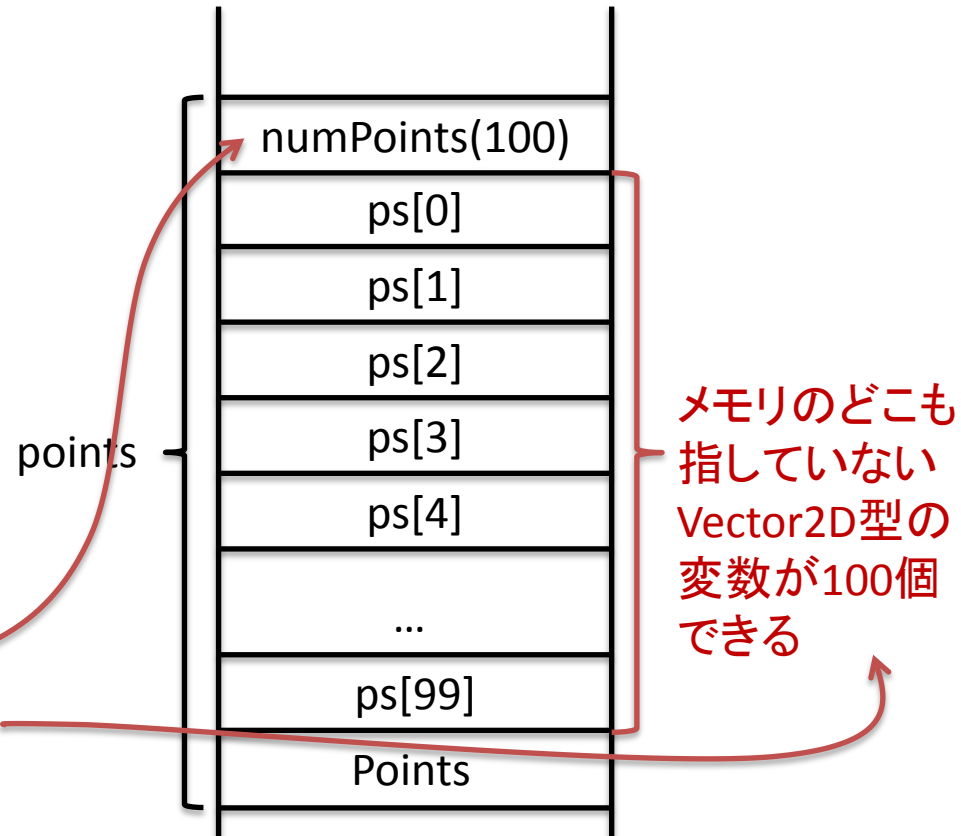
    double getMinY()
    {
        //配列psの中から最小のYを探し, 戻り値とする
    }

    double getMaxY()
    {
        //配列psの中から最大のYを探し, 戻り値とする
    }
}
```

補足

- どう動くのか？

```
int numps = 100;  
...  
Points points = new Points(numps);  
class Points  
{  
    int numPoints;  
    Vector2D[] ps;  
  
    Points(int numps)  
    {  
        numPoints = numps;  
        ps = new Vector2D[numPoints];  
    }  
}
```



補足

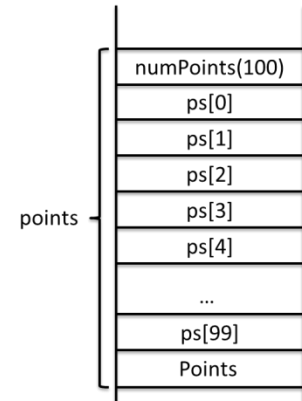
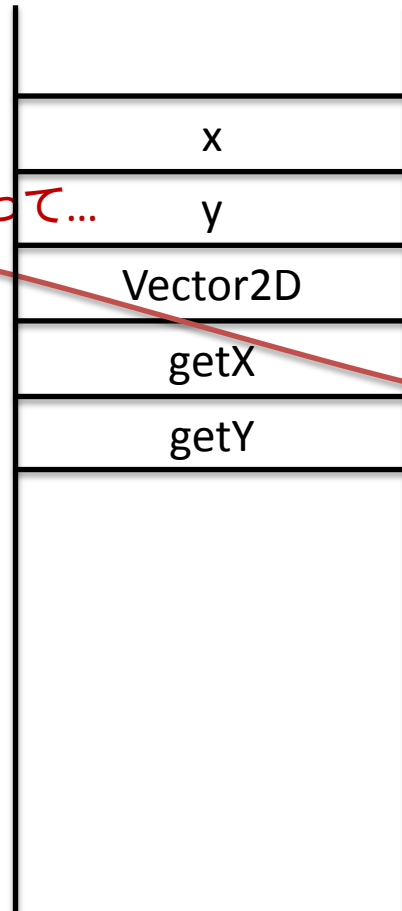
- どう動くのか？

例えば*i*が0の時...

```
for(int i = 0; i < numps; i++)  
{  
  ...  
  p = new Vector2D(x, y);  
  ...  
  points.setPoints(i, p);  
}
```

Vector2Dの
インスタンスを作って...

```
void setPoints(int i, Vector2D p)  
{  
  ps[i] = p;  
}
```



インスタンスがメモリの
どこにあるか (=p)
を配列ps[0]へ代入

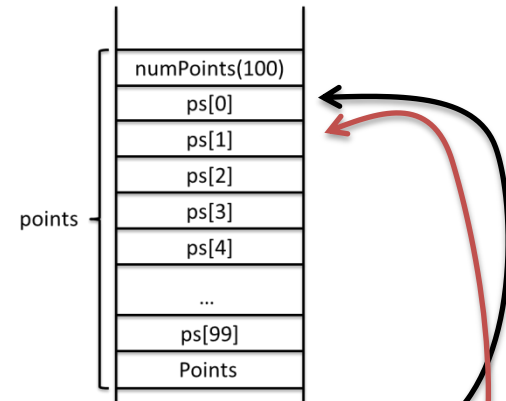
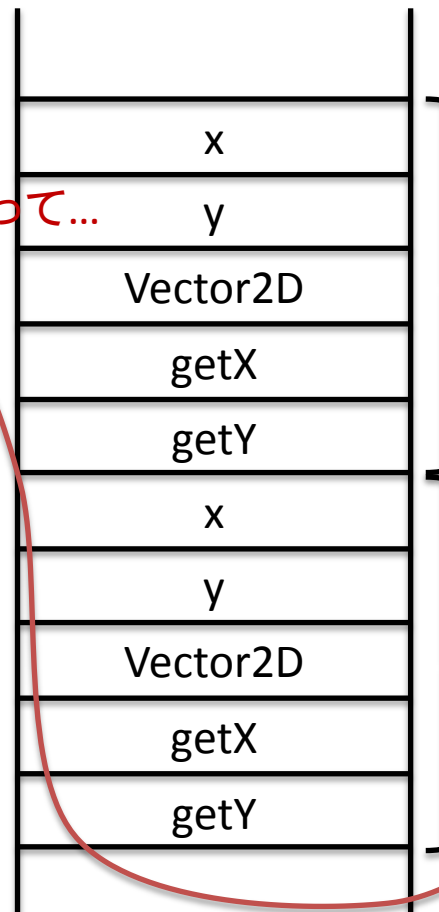
補足

- どう動くのか？

次に*i*が1の時は

```
for(int i = 0; i < numps; i++)  
{  
  ...  
  p = new Vector2D(x, y);  
  ...  
  points.setPoints(i, p);  
}  
  
void setPoints(int i, Vector2D p)  
{  
  ps[i] = p;  
}
```

さらにVector2Dの
インスタンスを作って...

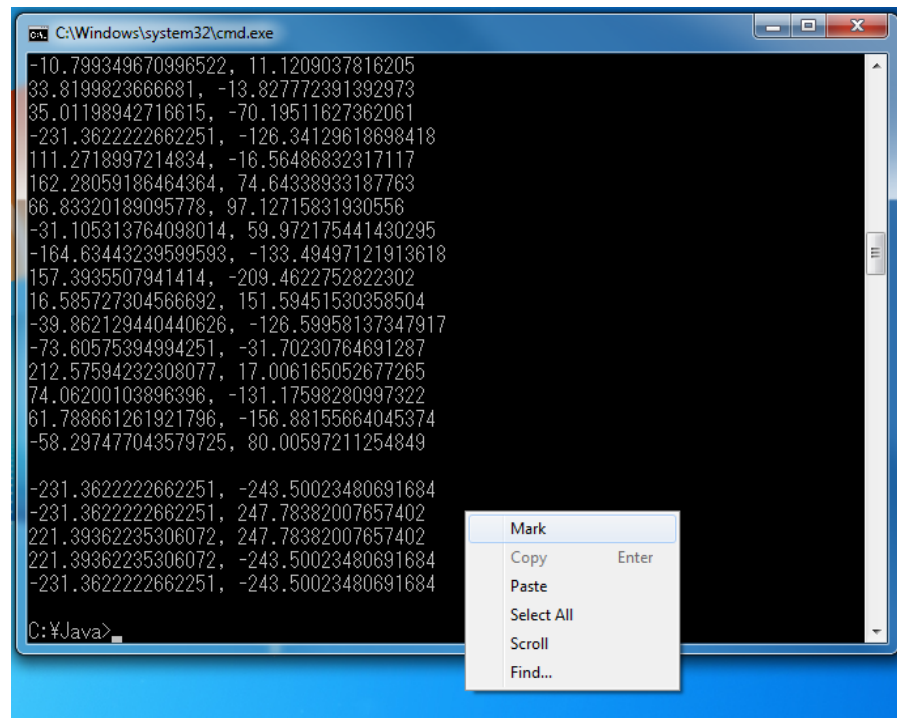


インスタンスがメモリの
どこにあるか (=p)
を配列ps[1]へ代入

以上が100回繰り返される

補足

- 結果の確認方法（Windowsの場合）
（UbuntuでもOpenOfficeで同様のことが可能？（未確認））



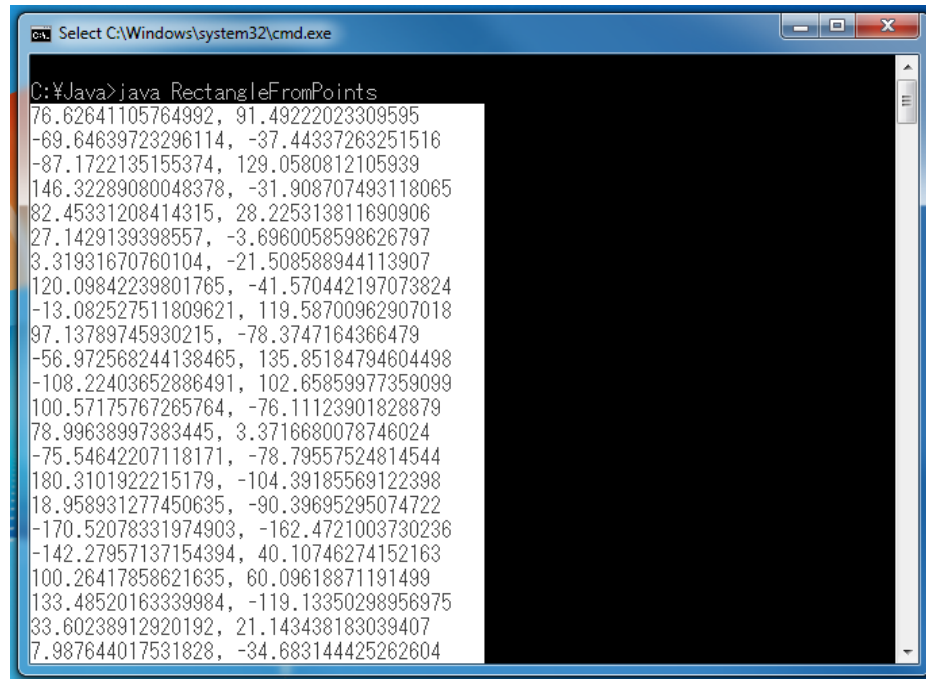
The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window contains a list of numbers, with a context menu open over the text. The context menu options are: Mark, Copy (with Enter next to it), Paste, Select All, Scroll, and Find... The numbers in the command prompt are:

```
-10.799349670996522, 11.1209037816205  
33.8199823666681, -13.827772391392973  
35.01198942716615, -70.19511627362061  
-231.3622222662251, -126.34129618698418  
111.2718997214834, -16.56486832317117  
162.28059186464364, 74.64338933187763  
66.83320189095778, 97.12715831930556  
-31.105313764098014, 59.972175441430295  
-164.63443239599593, -133.49497121913618  
157.3935507941414, -209.4622752822302  
16.585727304566692, 151.59451530358504  
-39.862129440440626, -126.59958137347917  
-73.60575394994251, -31.70230764691287  
212.57594232308077, 17.006165052677265  
74.06200103896396, -131.17598280997322  
61.788661261921796, -156.88155664045374  
-58.297477043579725, 80.00597211254849  
  
-231.3622222662251, -243.50023480691684  
-231.3622222662251, 247.78382007657402  
221.39362235306072, 247.78382007657402  
221.39362235306072, -243.50023480691684  
-231.3622222662251, -243.50023480691684
```

コマンドプロンプト上で右クリック, [選択]をクリック
（スクリーンショットは英語版Windowsのため英語表記）

補足

- 結果の確認方法 (Windowsの場合)

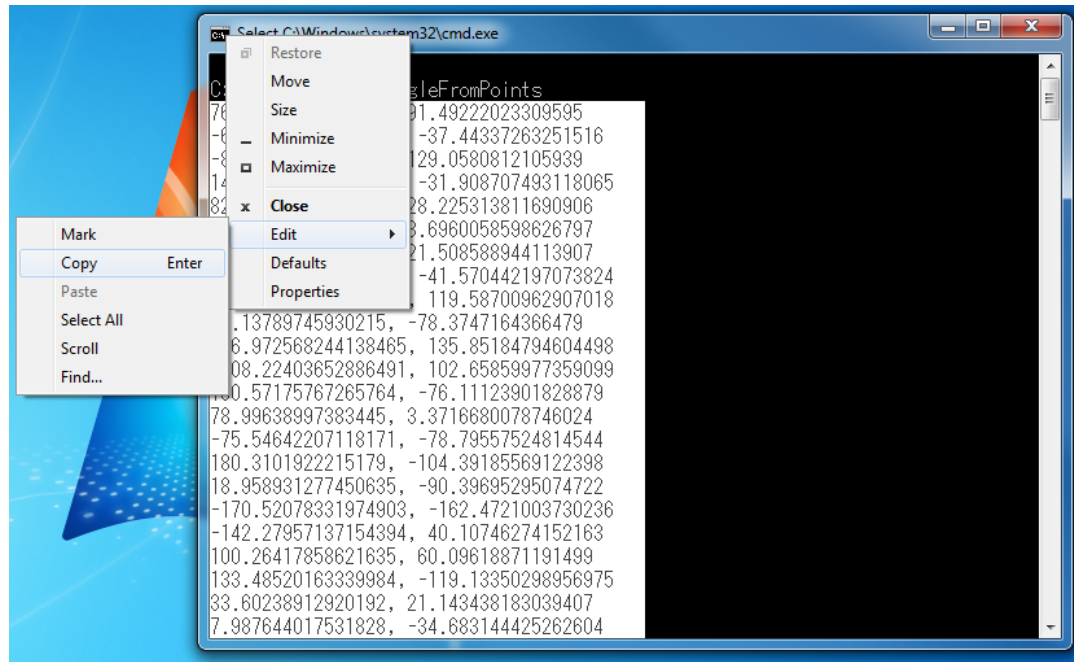


The image shows a Windows command prompt window titled "Select C:\Windows\system32\cmd.exe". The prompt is at "C:¥Java>". The user has entered the command "java RectangleFromPoints". The output consists of 20 lines of floating-point coordinates, each representing a point in a 2D space. The coordinates are: 76.62641105764992, 91.49222023309595; -69.64639723296114, -37.44337263251516; -87.1722135155374, 129.0580812105939; 146.32289080048378, -31.908707493118065; 82.45331208414315, 28.225313811690906; 27.1429139398557, -3.6960058598626797; 3.31931670760104, -21.508588944113907; 120.09842239801765, -41.570442197073824; -13.082527511809621, 119.58700962907018; 97.13789745930215, -78.3747164366479; -56.972568244138465, 135.85184794604498; -108.22403652886491, 102.65859977359099; 100.57175767265764, -76.11123901828879; 78.99638997383445, 3.3716680078746024; -75.54642207118171, -78.79557524814544; 180.3101922215179, -104.39185569122398; 18.958931277450635, -90.39695295074722; -170.52078331974903, -162.4721003730236; -142.27957137154394, 40.10746274152163; 100.26417858621635, 60.09618871191499; 133.48520163339984, -119.13350298956975; 33.60238912920192, 21.143438183039407; 7.987644017531828, -34.683144425262604.

出力されたデータをすべて選択

補足

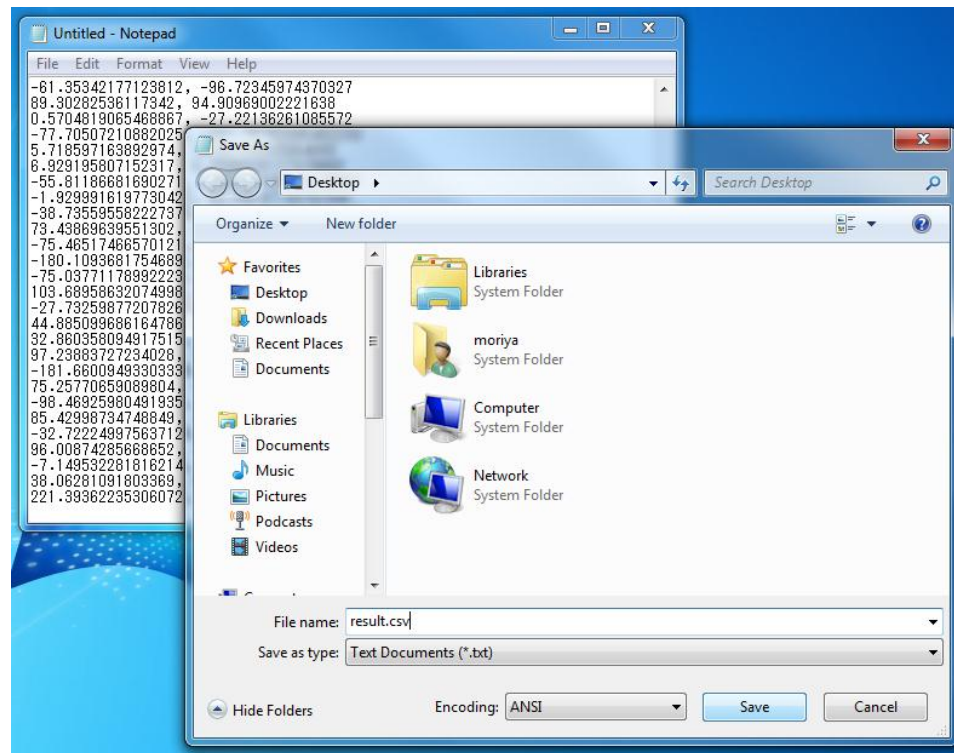
- 結果の確認方法 (Windowsの場合)



コマンドプロンプト左上のアイコンを右クリック
[編集]→[コピー]

補足

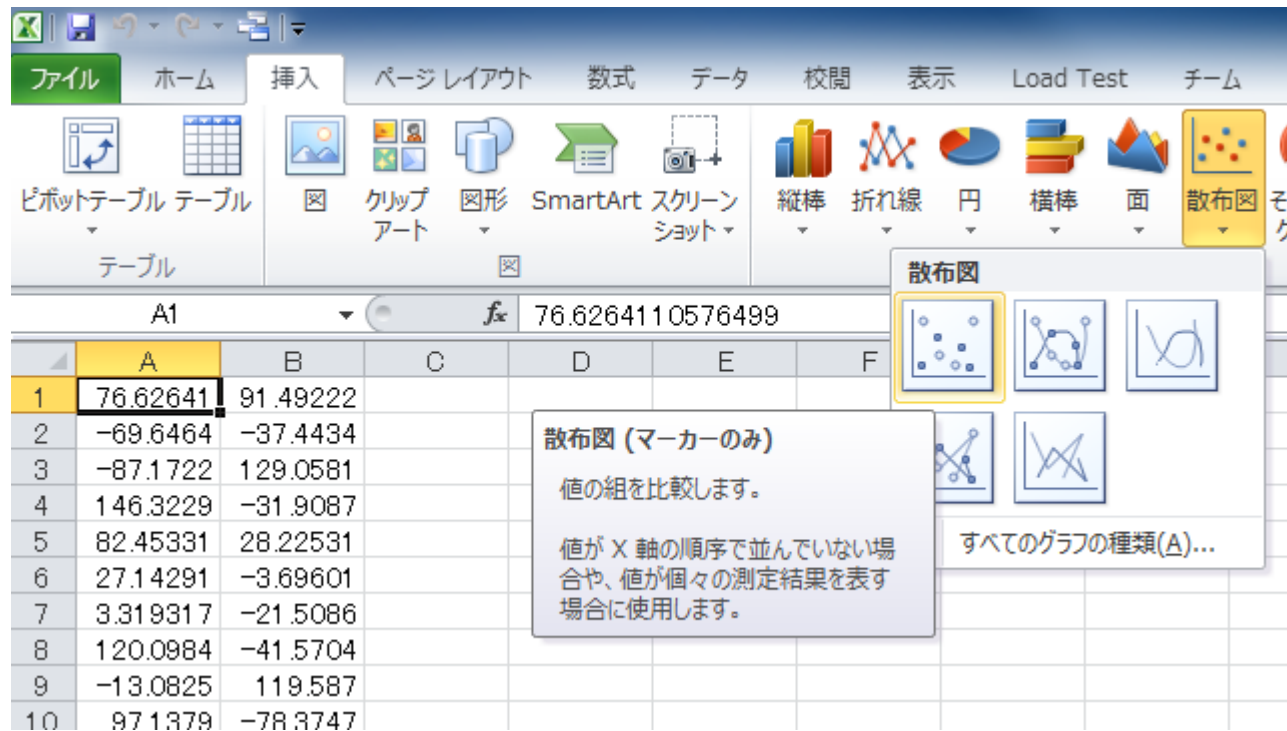
- 結果の確認方法 (Windowsの場合)



適当なエディタ(メモ帳でも構わない)へデータを貼り付け
拡張子をcsv(ファイル名.csv)にして保存

補足

- 結果の確認方法 (Windowsの場合)



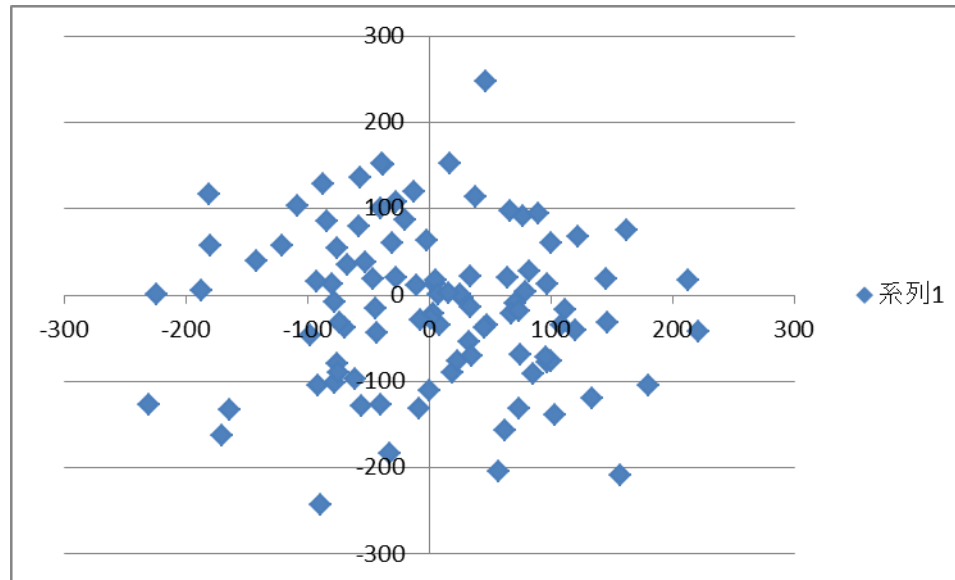
The screenshot shows the Microsoft Excel interface with the 'Insert' tab selected. The 'Charts' group is expanded to show 'Scatter' chart options. A tooltip for 'Scatter (Markers Only)' is displayed, explaining its use for comparing data groups or showing individual measurement results.

	A	B	C	D	E	F
1	76.62641	91.49222				
2	-69.6464	-37.4434				
3	-87.1722	129.0581				
4	146.3229	-31.9087				
5	82.45331	28.22531				
6	27.14291	-3.69601				
7	3.319317	-21.5086				
8	120.0984	-41.5704				
9	-13.0825	119.587				
10	97.1379	-78.3747				

保存したcsvファイルをExcelで開き,
[挿入]タブ→グラフ[散布図]→[散布図(マーカーのみ)]

補足

- 結果の確認方法 (Windowsの場合)

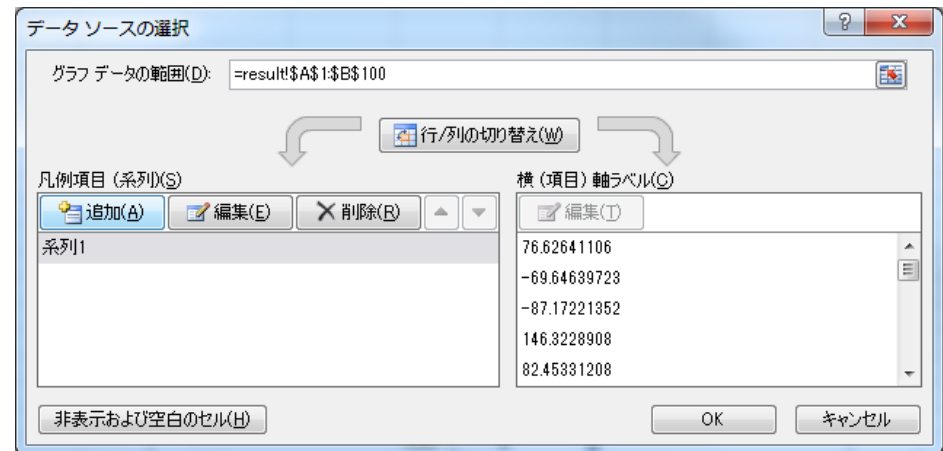
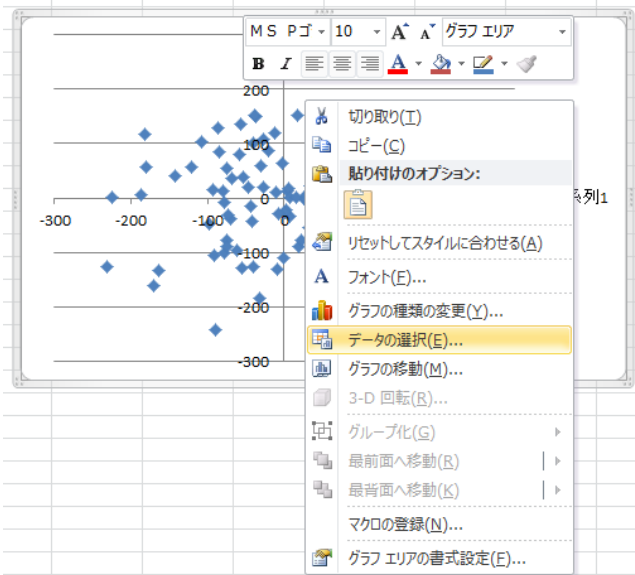


ここまででランダムに生成された100個の点がグラフ化される

補足

- 結果の確認方法 (Windowsの場合)

次にプログラムで求めた矩形をグラフ化する



グラフ上で右クリック, [データの選択]→[追加]

補足

- 結果の確認方法 (Windowsの場合)

95	-39.8621	-126.6
96	-73.6058	-31.7023
97	212.5759	17.00617
98	74.062	-131.176
99	61.78866	-156.882
100	-58.2975	80.00587
101		
102	-231.362	-243.5
103	-231.362	247.7838
104	221.3936	247.7838
105	221.3936	-243.5
106	-231.362	-243.5
107		
108		
109		

系列の編集

系列名(N):
Rectangle = Rectangle

系列 X の値(X):
=result!\$A\$102:\$A\$106 = -231.3622223, ...

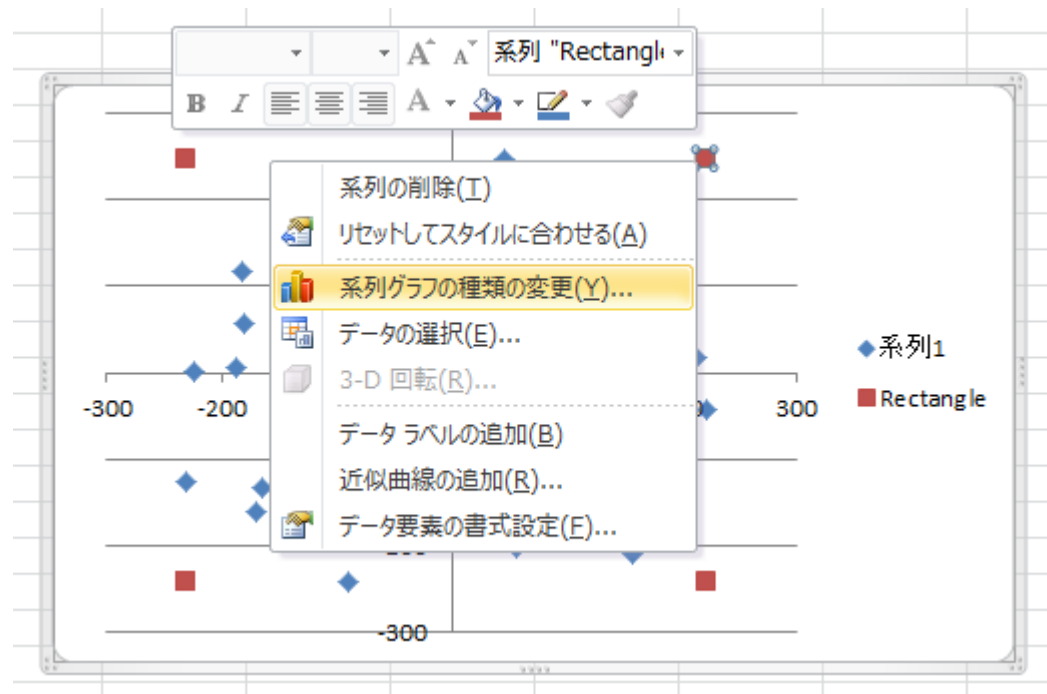
系列 Y の値(Y):
=result!\$B\$102:\$B\$106 = -243.5002348, ...

OK キャンセル

系列名をRectangle,
矩形の座標は、空白行以下にあるので系列X, Yをそれぞれ選択してOK

補足

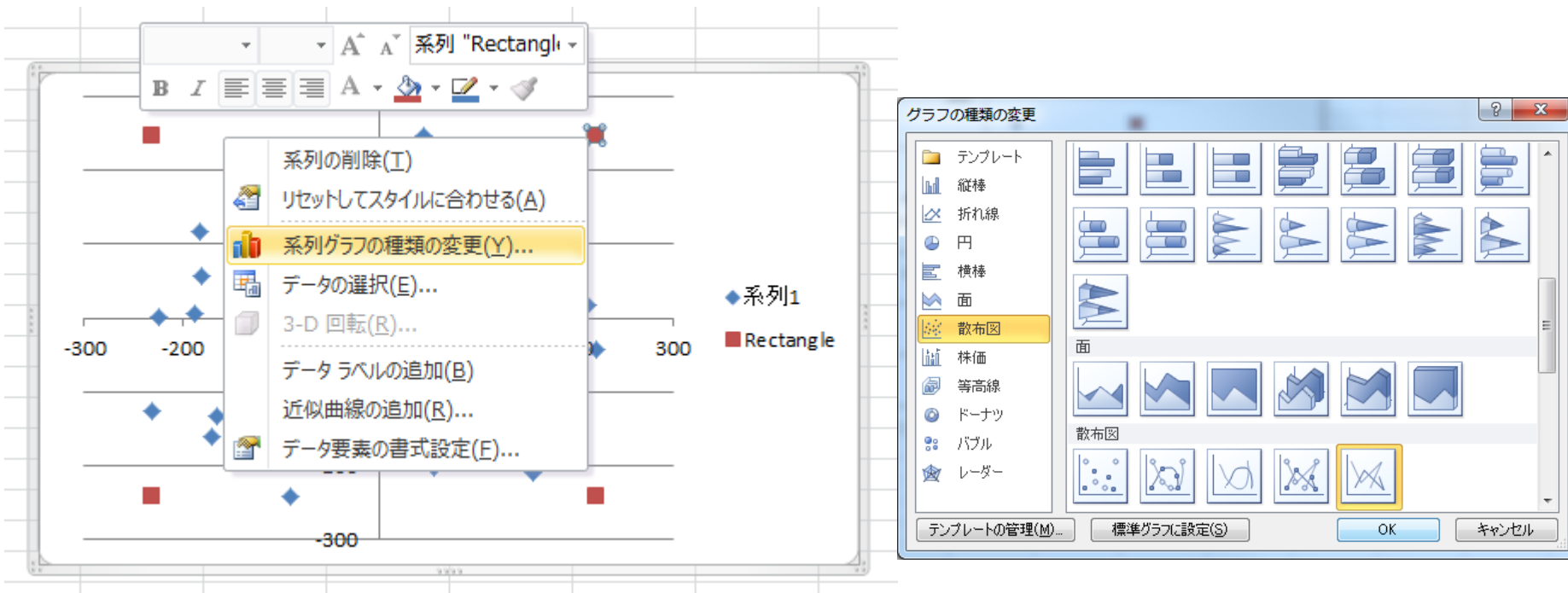
- 結果の確認方法 (Windowsの場合)



追加された点を右クリック→[系列グラフの種類の変更]

補足

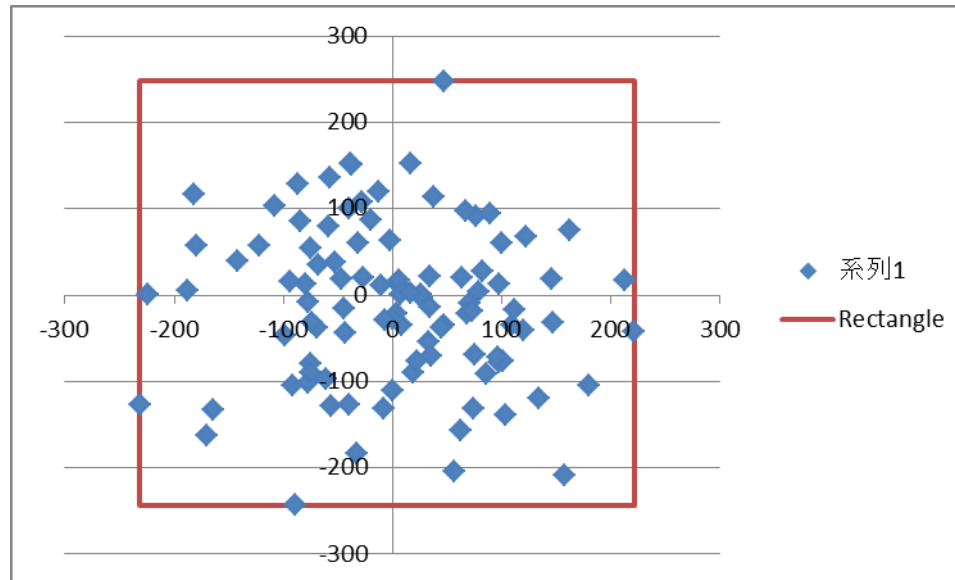
- 結果の確認方法 (Windowsの場合)



追加された点を右クリック→[系列グラフの種類の変更]
散布図(直線)を右クリックしてOK

補足

- 結果の確認方法 (Windowsの場合)



正しいプログラムが書けていれば, 点群を内包する
矩形が現れる