# Line Rendering

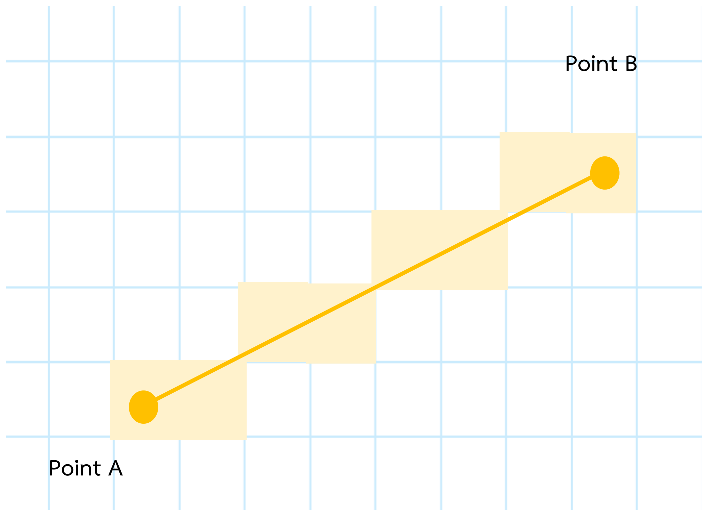
1. Thanaporn Yankomut 60070501018

2. Phuraefa Rattanatakun 60070501045

3.Wantanee Saetear 60070501053

4. Parattha Weerapong 60070501080

## Bresenham's Algorithm



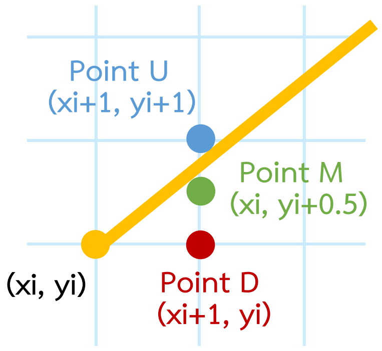
Bresenham is algorithm for find all intermediate points required for drawing line from point A to point B and every point has only integer coordinates.

1.declare the variable

```
public void plotLine(int x1,int y1,int x2,int y2)
{
    dx = x2 - x1;
    dy = y2 - y1;
    d = 2 * dy - dx;
    dU = 2 * dy;
    dD = 2*(dy-dx);
    x = x1;
    y = y1;
```

dx is difference between x of 2 point

dy is difference between y of 2 point



d is difference between point M and the line

if d > 0 is mean line is above point M

and if d < 0 is mean line is below point M

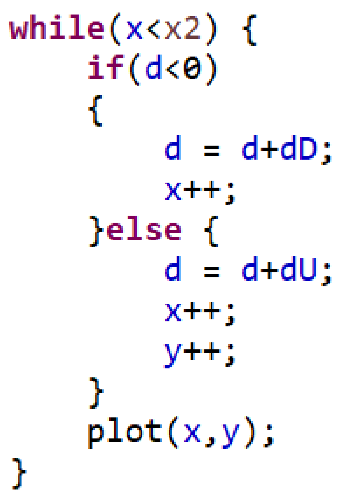
dU is difference between point M and point U

dD is difference between point M and point D

2.Plot the first point

```
plot(x,y);
```

**3.Plot the other points by choosing between point U and point D**



```
while(x<x2) {
    if(d<0)
    {
        d = d+dD;
        x++;
    }else {
        d = d+dU;
        x++;
        y++;
    }
    plot(x,y);
}
```

Choose the next point by d value.

If d is less than 0 is mean Point M is below the line, then you should choose point D.

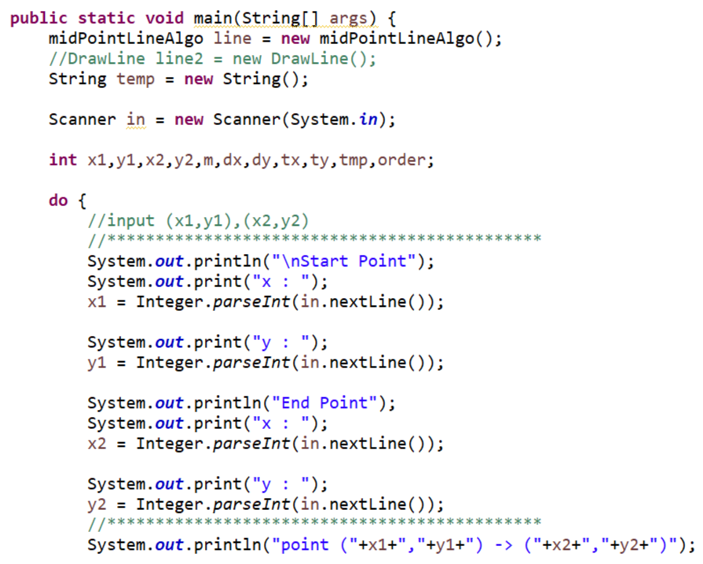
And change d value to d + dD.

Or If d is greater than 0 is mean Point M is above the line, then you should choose point U. And change d value to d + dU.

and do this repeatedly until reach the last point.

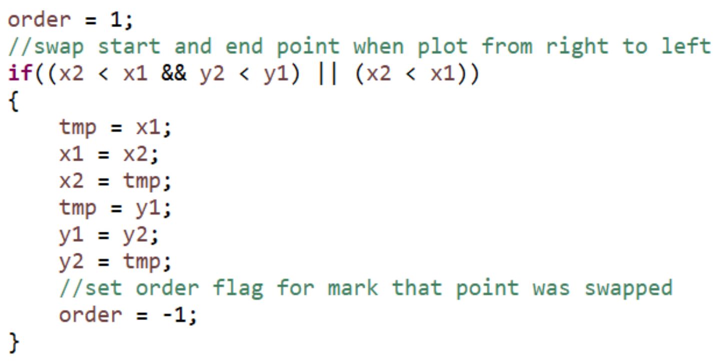
## Implementation

1. First get start point and end point from user input

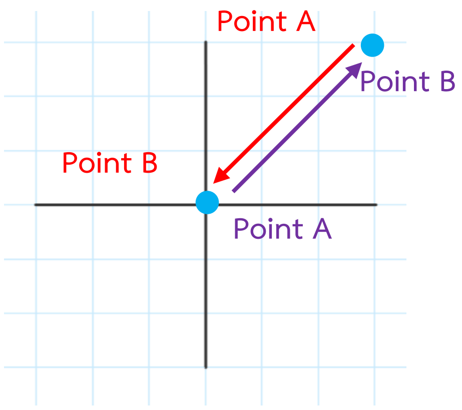


```
public static void main(String[] args) {
    midPointLineAlgo line = new midPointLineAlgo();
    //DrawLine line2 = new DrawLine();
    String temp = new String();

    Scanner in = new Scanner(System.in);

    int x1,y1,x2,y2,m,dx,dy,tx,ty,tmp,order;

    do {
        //input (x1,y1),(x2,y2)
        //**********************************************
        System.out.println("\nStart Point");
        System.out.print("x : ");
        x1 = Integer.parseInt(in.nextLine());

        System.out.print("y : ");
        y1 = Integer.parseInt(in.nextLine());

        System.out.println("End Point");
        System.out.print("x : ");
        x2 = Integer.parseInt(in.nextLine());

        System.out.print("y : ");
        y2 = Integer.parseInt(in.nextLine());
        //**********************************************
        System.out.println("point ("+x1+","+y1+") -> ("+x2+","+y2+")");
```

2. Check if the line is rendering from right to left (start point is greater than end point) swaps these 2 points to rendering from left to right.

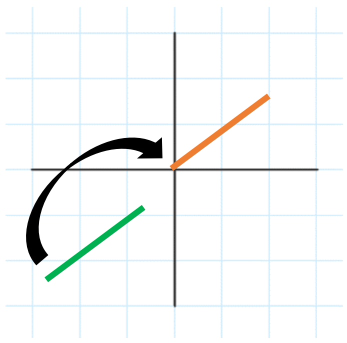


```
order = 1;
//swap start and end point when plot from right to left
if((x2 < x1 && y2 < y1) || (x2 < x1))
{
    tmp = x1;
    x1 = x2;
    x2 = tmp;
    tmp = y1;
    y1 = y2;
    y2 = tmp;
    //set order flag for mark that point was swapped
    order = -1;
}
```



3. Translate these 2 points to start at the origin

```
//translate (x1,y1) to origin
tx = -x1;
x1 = x1 + tx;
x2 = x2 + tx;
ty = -y1;
y1 = y1 + ty;
y2 = y2 + ty;
```



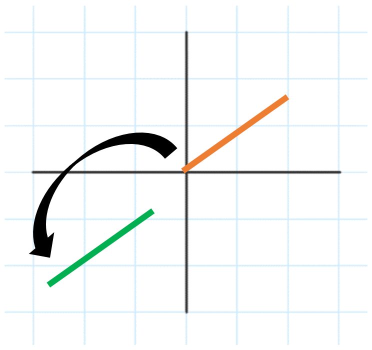
4. Plot the line by bresenham's algorithm.

```
line.plotLine(x1,y1,x2,y2);
```

*each case has some difference codes. Explain in next session.

5. Translate these 2 points to their old position and show all point that plotted.

```
//translate origin to (x1,y1)
int i = 0;
while(i < line.pointx.size())
{
    line.pointy.set(i, line.pointy.get(i) - ty);
    line.pointx.set(i, line.pointx.get(i) - tx);
    i++;
}

//show
show(line.pointx,line.pointy,order);
```



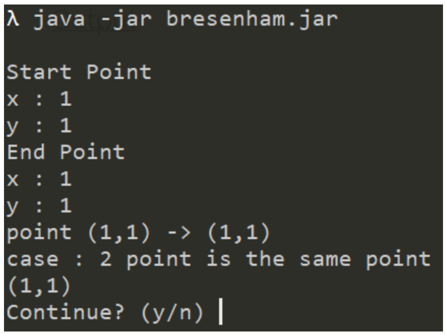
Case

1. 2 points are the same point.

```java
//2 point is the same point
if(x1 == x2 && y1 == y2)
{
    System.out.println("case : 2 point is the same point");
    System.out.println("("+x1+","+y1+")");
}
```

Just show the point that user input.

Testcase



```
λ java -jar bresenham.jar

Start Point
x : 1
y : 1
End Point
x : 1
y : 1
point (1,1) -> (1,1)
case : 2 point is the same point
(1,1)
Continue? (y/n)
```

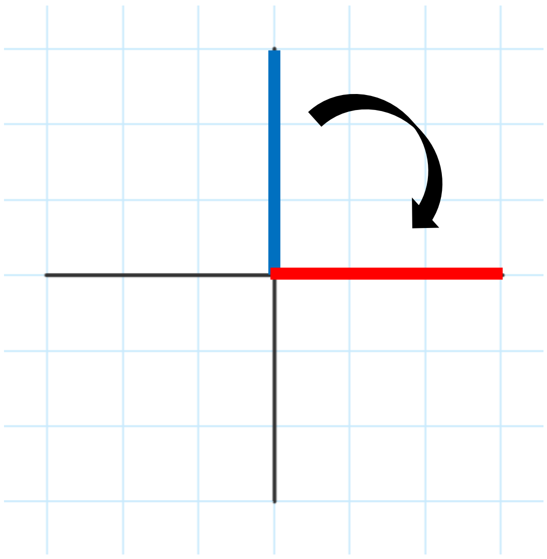
2. 2 points are part of vertical line.

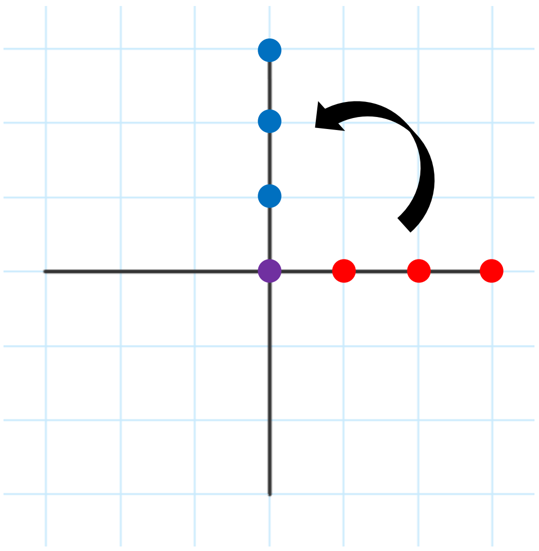
```
line.plotLine(x1,y1,y2,x2);
```

Swap x and y of end point to change vertical line to horizon line before plot.

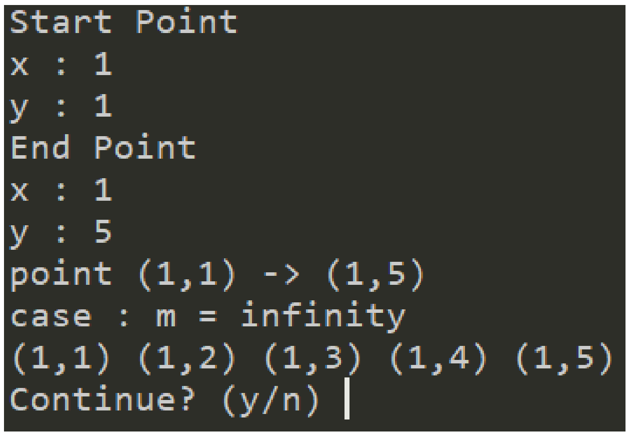
```
show(line.pointy,line.pointx,order);
```

After finish plotting the line swap x and y of all point to change horizon group of point. to vertical group of points.





Testcase



```
Start Point
x : 1
y : 1
End Point
x : 1
y : 5
point (1,1) -> (1,5)
case : m = infinity
(1,1) (1,2) (1,3) (1,4) (1,5)
Continue? (y/n)
```

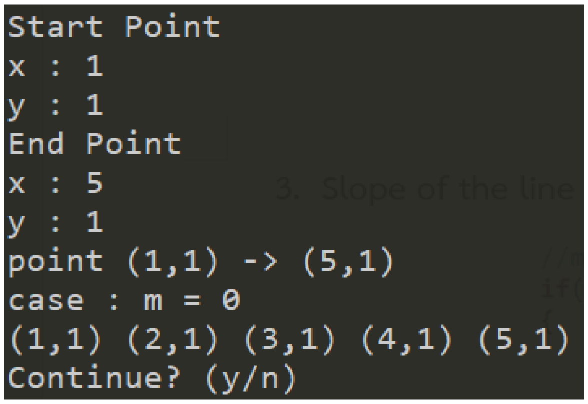
3. Slope of the line is greater than 0 degree and less than 45 degree

```java
//m = 0
if(y1 == y2)
{
    System.out.println("case : m = 0");
    line.plotLine(x1,y1,x2,y2);
}
```

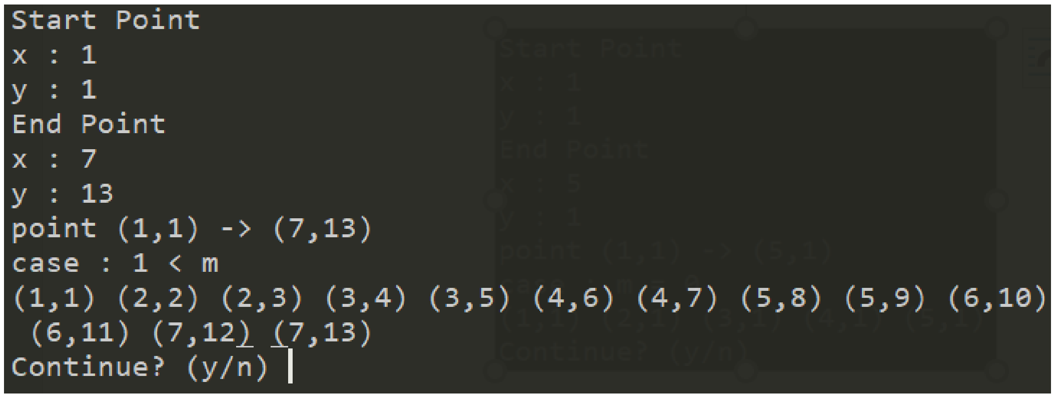
```java
//case 0 < m < 1
else if(m >= 0 && m < 1 && ((dx > 0 && dy > 0)||(dx < 0 && dy < 0)))
{
    System.out.println("case : 0 < m < 1");
    line.plotLine(x1,y1,x2,y2);
}
```

The line that has slope greater than 0 degree and less than 45 degree can render all point with bresenham's algorithm.

<u>Testcase</u>



```
Start Point
x : 1
y : 1
End Point
x : 5
y : 1
point (1,1) -> (5,1)
case : m = 0
(1,1) (2,1) (3,1) (4,1) (5,1)
Continue? (y/n)
```

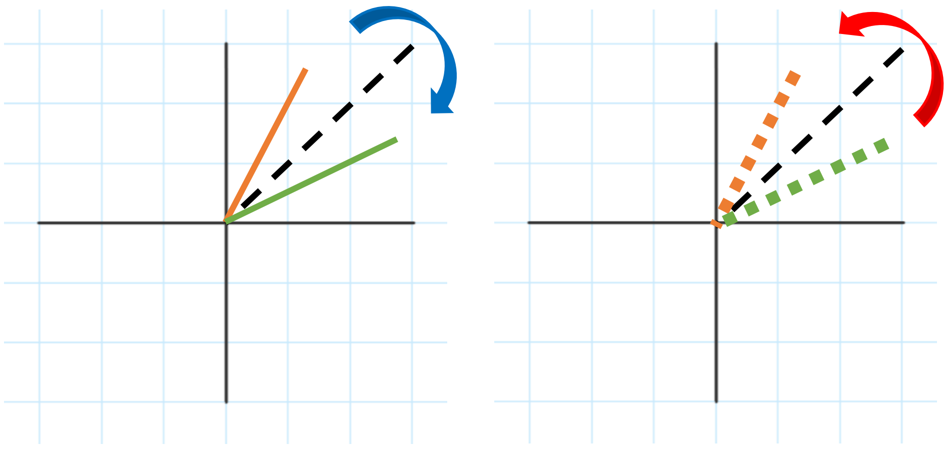


```
Start Point
x : 1
y : 1
End Point
x : 7
y : 13
point (1,1) -> (7,13)
case : 1 < m
(1,1) (2,2) (2,3) (3,4) (3,5) (4,6) (4,7) (5,8) (5,9) (6,10)
 (6,11) (7,12) (7,13)
Continue? (y/n)
```

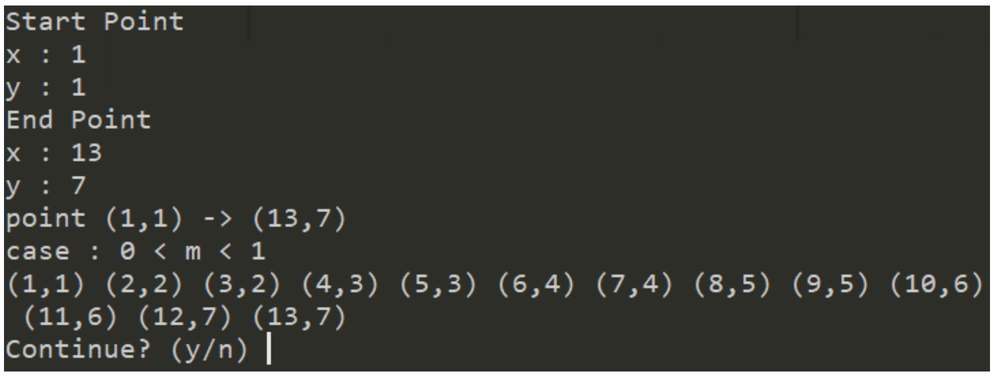
4. Slope of the line is greater than 45 degree and less than 90 degree

```
//case m > 1
else if(m >= 1)
{
    System.out.println("case : 1 < m");
    //mirror by 45 degree
    line.plotLine(x1,y1,y2,x2);
    //mirror back
    line.mirror45();
}
```

Mirror by 45 degree (swap x and y of end point) before plot to change the line into the area that can use bresenham's algorithm and when finish plotting the line mirror back.



Testcase

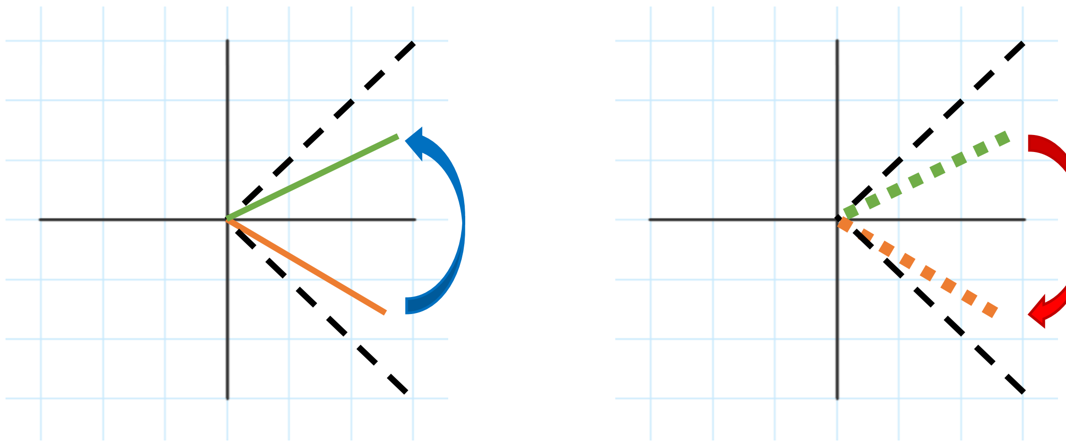


```
Start Point
x : 1
y : 1
End Point
x : 13
y : 7
point (1,1) -> (13,7)
case : 0 < m < 1
(1,1) (2,2) (3,2) (4,3) (5,3) (6,4) (7,4) (8,5) (9,5) (10,6)
 (11,6) (12,7) (13,7)
Continue? (y/n)
```

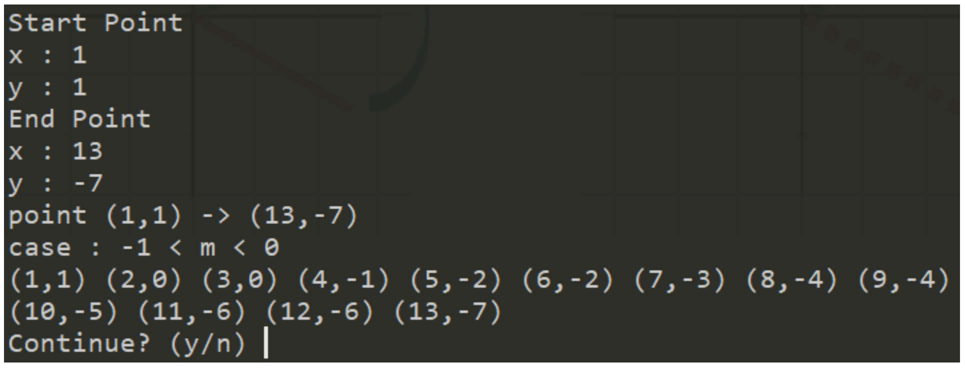
5. Slope is less than 0 degree and greater than -45 degree

```
//case m < 0
else if(m <= 0 && m > -1)
{
    System.out.println("case : -1 < m < 0");
    //mirror by x axis
    line.plotLine(x1,y1,x2,-y2);
    //mirror back
    line.mirrorY();
}
```

Mirror by y axis (negative y) before plot to change the line into the area that can use bresenham's algorithm and when finish plotting the line mirror back



<u>Testcase</u>

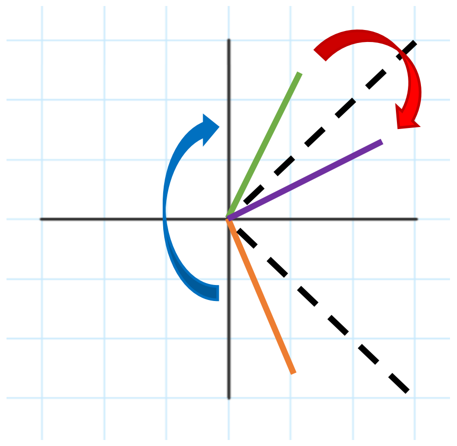


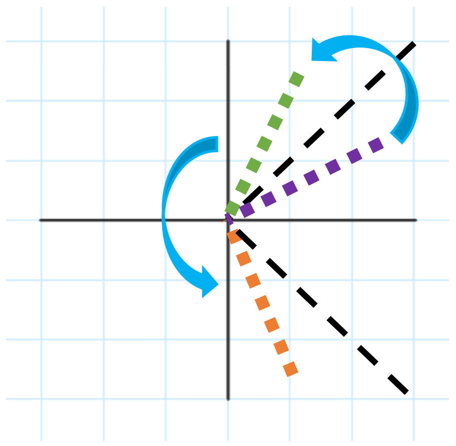
```
Start Point
x : 1
y : 1
End Point
x : 13
y : -7
point (1,1) -> (13,-7)
case : -1 < m < 0
(1,1) (2,0) (3,0) (4,-1) (5,-2) (6,-2) (7,-3) (8,-4) (9,-4)
(10,-5) (11,-6) (12,-6) (13,-7)
Continue? (y/n)
```

6. Slope is less than -45 degree

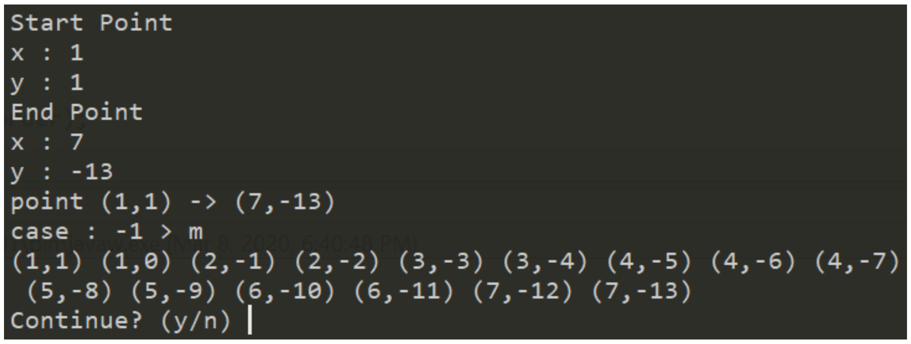
```
//case m < -1
else if(m <= -1)
{
    System.out.println("case : -1 > m");
    //mirror by y axis,then mirror by 45 degree
    line.plotLine(x1,y1,-y2,x2);
    //mirror by 45 degree back
    line.mirror45();
    //mirror by y axis back
    line.mirrorY();

}
```

Mirror by y axis (negative y) and then mirror by 45 degree before plot to change the line into the area that can use bresenham's algorithm and when finish plotting the line mirror back





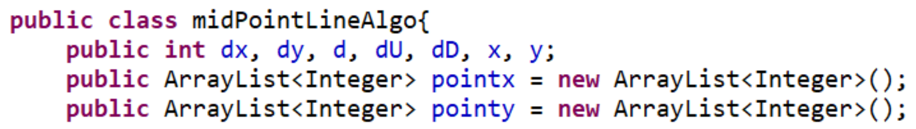
Testcase



```
Start Point
x : 1
y : 1
End Point
x : 7
y : -13
point (1,1) -> (7,-13)
case : -1 > m
(1,1) (1,0) (2,-1) (2,-2) (3,-3) (3,-4) (4,-5) (4,-6) (4,-7)
 (5,-8) (5,-9) (6,-10) (6,-11) (7,-12) (7,-13)
Continue? (y/n)
```

## Source Code

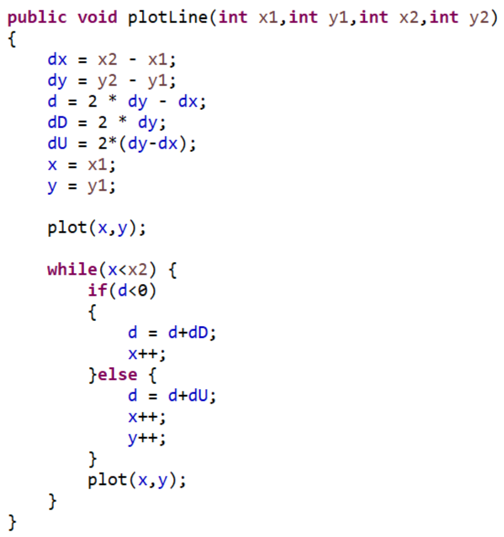
### Class midPointLineAlgo



```java
public class midPointLineAlgo{
    public int dx, dy, d, dU, dD, x, y;
    public ArrayList<Integer> pointx = new ArrayList<Integer>();
    public ArrayList<Integer> pointy = new ArrayList<Integer>();
```

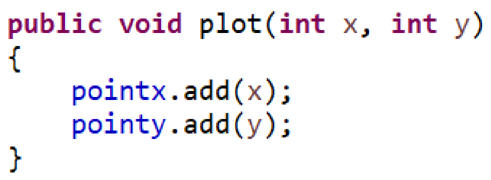
dx, dy, d, dU , dD, x, y are for bresenham's algorithm.

pointx, pointy are array List for keep rendered point.



```java
public void plotLine(int x1,int y1,int x2,int y2)
{
    dx = x2 - x1;
    dy = y2 - y1;
    d = 2 * dy - dx;
    dD = 2 * dy;
    dU = 2*(dy-dx);
    x = x1;
    y = y1;

    plot(x,y);

    while(x<x2) {
        if(d<0)
        {
            d = d+dD;
            x++;
        }else {
            d = d+dU;
            x++;
            y++;
        }
        plot(x,y);
    }
}
```

Bresenham's algorithm



```java
public void plot(int x, int y)
{
    pointx.add(x);
    pointy.add(y);
}
```

Method plot is for keep the rendered point in array list.

```
public void mirrorY ()
{
    int i = 0;
    while(i < pointy.size())
    {
        pointy.set(i,-pointy.get(i));
        i++;
    }
}
```

Method mirrorY is for change value of y to negative

```
public void mirror45 ()
{
    ArrayList<Integer> temp = new ArrayList<Integer>();
    temp.addAll(pointx);
    pointx.removeAll(pointx);
    pointx.addAll(pointy);
    pointy.removeAll(pointy);
    pointy.addAll(temp);
}
```

Method mirror45 is for swap value of x and y.

## Class Main

```
public static void main(String[] args) {
    midPointLineAlgo line = new midPointLineAlgo();
    String temp = new String();
    Scanner in = new Scanner(System.in);
    int x1,y1,x2,y2,m,dx,dy,tx,ty,tmp,order;
```

line is for call the midPointLineAlgo class.

temp is for input when asking to continue program.

in is for get user input.

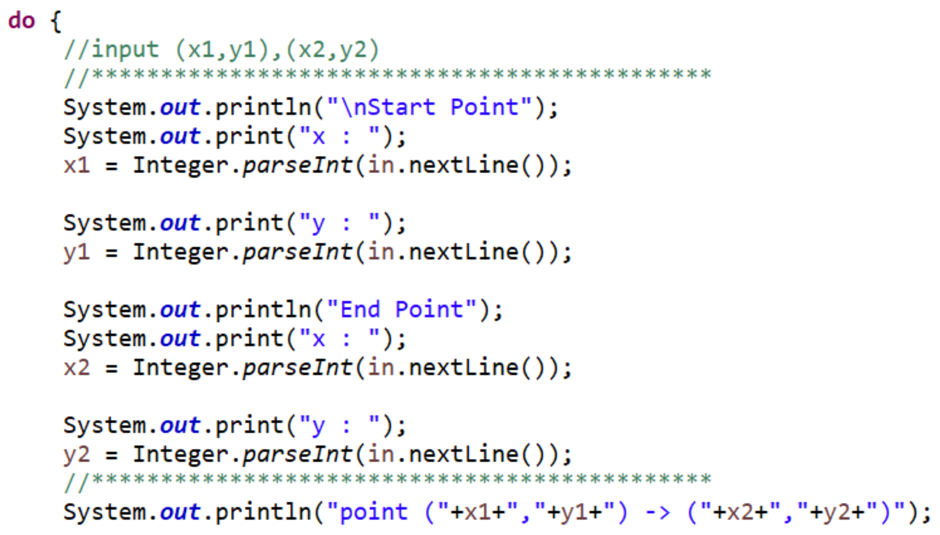
x1, y1 x2, y2 are for value of x,y for start and end point.

m, dx, dy are for check case of slope.

tx, ty are for translate points.

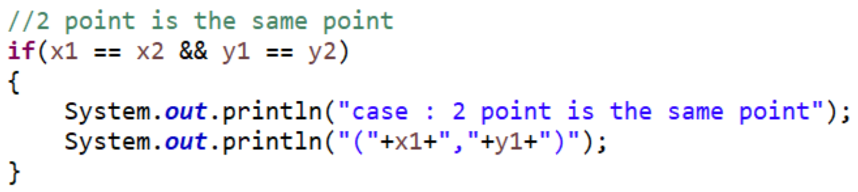
tmp is for swap points.

order is flag for ordering the point's result.



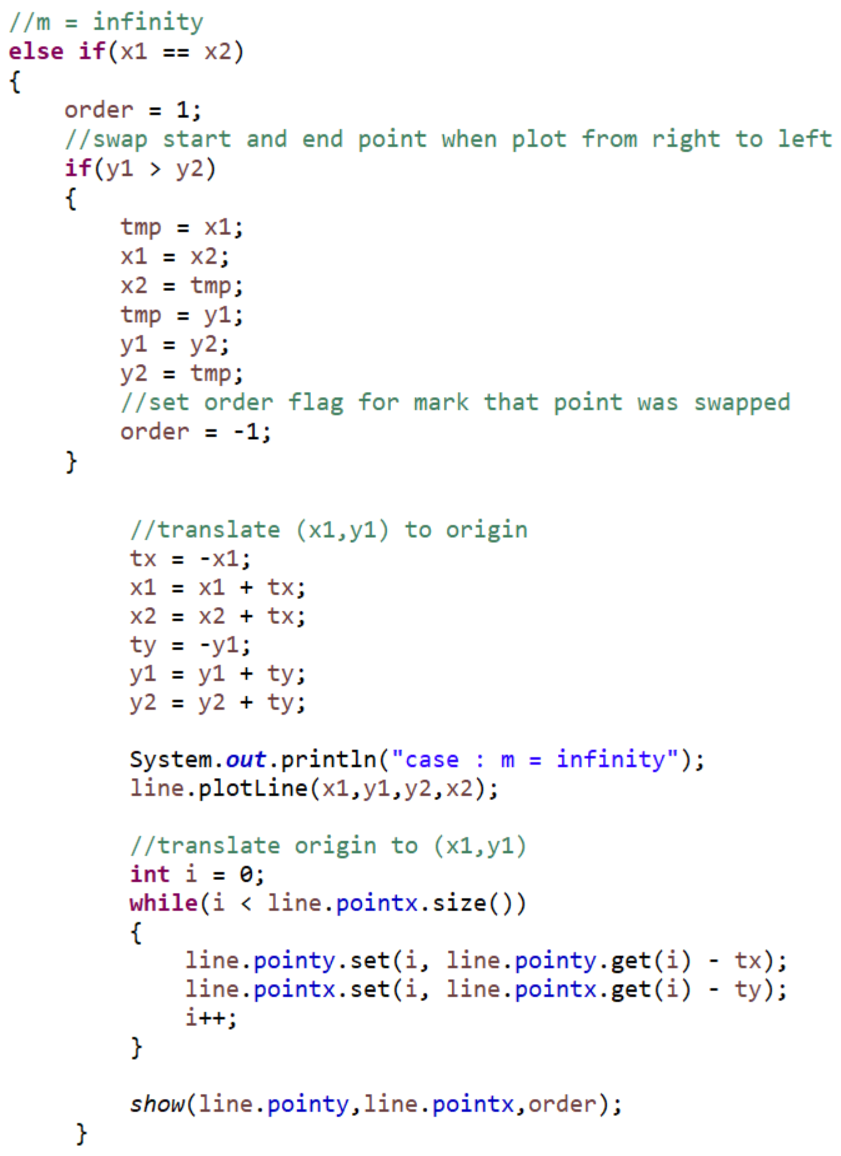
```java
do {
    //input (x1,y1),(x2,y2)
    //*********************************************
    System.out.println("\nStart Point");
    System.out.print("x : ");
    x1 = Integer.parseInt(in.nextLine());

    System.out.print("y : ");
    y1 = Integer.parseInt(in.nextLine());

    System.out.println("End Point");
    System.out.print("x : ");
    x2 = Integer.parseInt(in.nextLine());

    System.out.print("y : ");
    y2 = Integer.parseInt(in.nextLine());
    //*********************************************
    System.out.println("point ("+x1+","+y1+") -> ("+x2+","+y2+")");
```

Get x, y of start and end point from user.



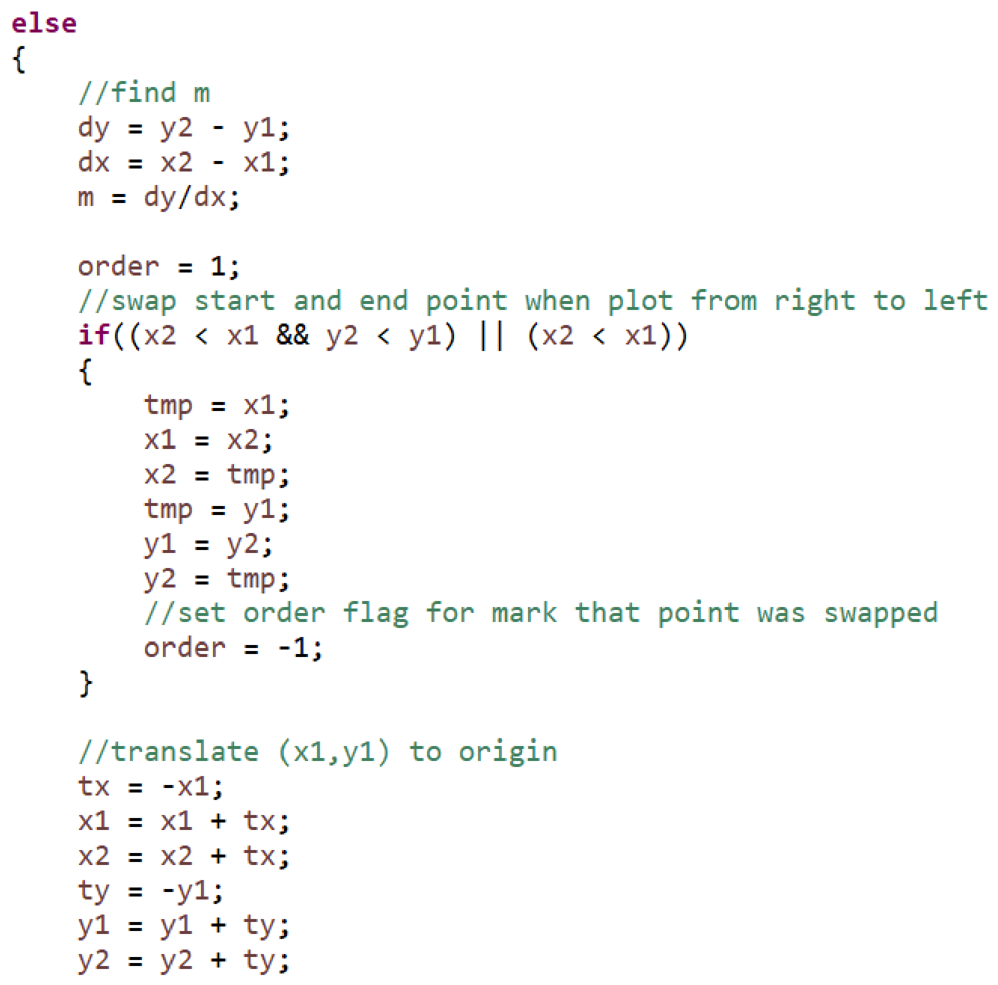
```java
//2 point is the same point
if(x1 == x2 && y1 == y2)
{
    System.out.println("case : 2 point is the same point");
    System.out.println("("+x1+","+y1+")");
}
```

Case 2 point is the same point



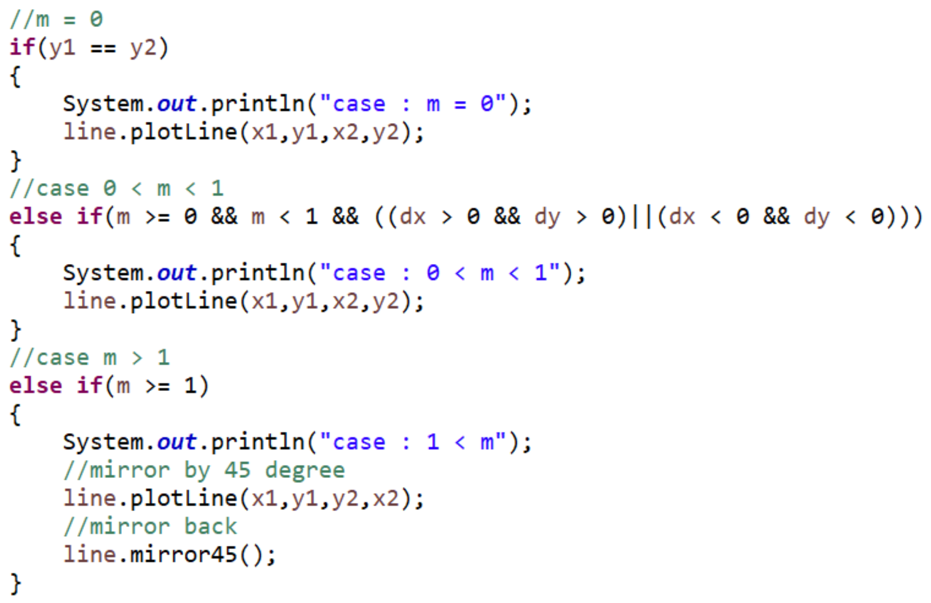
```java
//m = infinity
else if(x1 == x2)
{
    order = 1;
    //swap start and end point when plot from right to left
    if(y1 > y2)
    {
        tmp = x1;
        x1 = x2;
        x2 = tmp;
        tmp = y1;
        y1 = y2;
        y2 = tmp;
        //set order flag for mark that point was swapped
        order = -1;
    }

         //translate (x1,y1) to origin
         tx = -x1;
         x1 = x1 + tx;
         x2 = x2 + tx;
         ty = -y1;
         y1 = y1 + ty;
         y2 = y2 + ty;

         System.out.println("case : m = infinity");
         line.plotLine(x1,y1,y2,x2);

         //translate origin to (x1,y1)
         int i = 0;
         while(i < line.pointx.size())
         {
             line.pointy.set(i, line.pointy.get(i) - tx);
             line.pointx.set(i, line.pointx.get(i) - ty);
             i++;
         }

         show(line.pointy,line.pointx,order);
 }
```

Case vertical line.

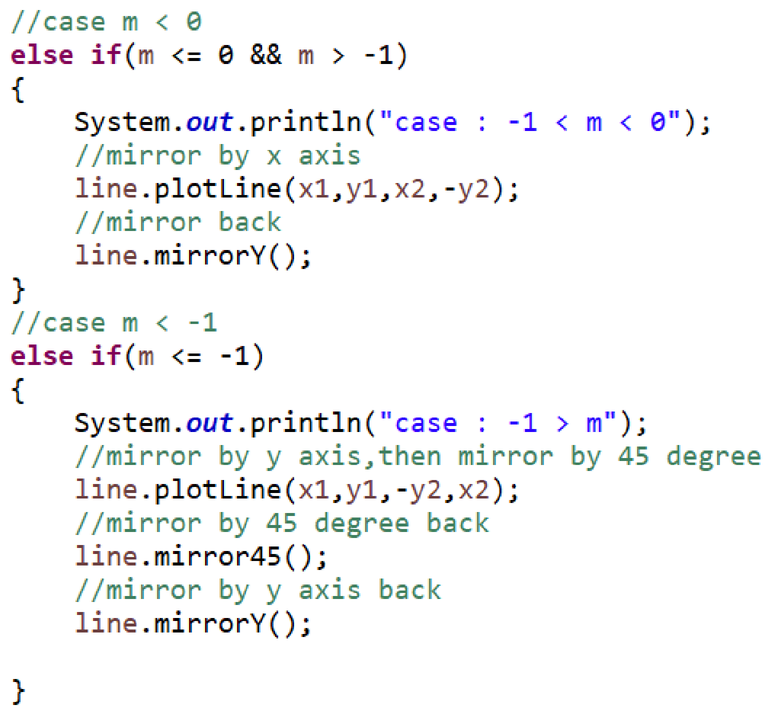


```
else
{
    //find m
    dy = y2 - y1;
    dx = x2 - x1;
    m = dy/dx;

    order = 1;
    //swap start and end point when plot from right to left
    if((x2 < x1 && y2 < y1) || (x2 < x1))
    {
        tmp = x1;
        x1 = x2;
        x2 = tmp;
        tmp = y1;
        y1 = y2;
        y2 = tmp;
        //set order flag for mark that point was swapped
        order = -1;
    }

    //translate (x1,y1) to origin
    tx = -x1;
    x1 = x1 + tx;
    x2 = x2 + tx;
    ty = -y1;
    y1 = y1 + ty;
    y2 = y2 + ty;
```

For the other cases find m for check slope, change the point to render from left to right and translate the point to the origin



```java
//m = 0
if(y1 == y2)
{
    System.out.println("case : m = 0");
    line.plotLine(x1,y1,x2,y2);
}
//case 0 < m < 1
else if(m >= 0 && m < 1 && ((dx > 0 && dy > 0)||(dx < 0 && dy < 0)))
{
    System.out.println("case : 0 < m < 1");
    line.plotLine(x1,y1,x2,y2);
}
//case m > 1
else if(m >= 1)
{
    System.out.println("case : 1 < m");
    //mirror by 45 degree
    line.plotLine(x1,y1,y2,x2);
    //mirror back
    line.mirror45();
}
```



```java
//case m < 0
else if(m <= 0 && m > -1)
{
    System.out.println("case : -1 < m < 0");
    //mirror by x axis
    line.plotLine(x1,y1,x2,-y2);
    //mirror back
    line.mirrorY();
}
//case m < -1
else if(m <= -1)
{
    System.out.println("case : -1 > m");
    //mirror by y axis,then mirror by 45 degree
    line.plotLine(x1,y1,-y2,x2);
    //mirror by 45 degree back
    line.mirror45();
    //mirror by y axis back
    line.mirrorY();

}
```

Plot the points.

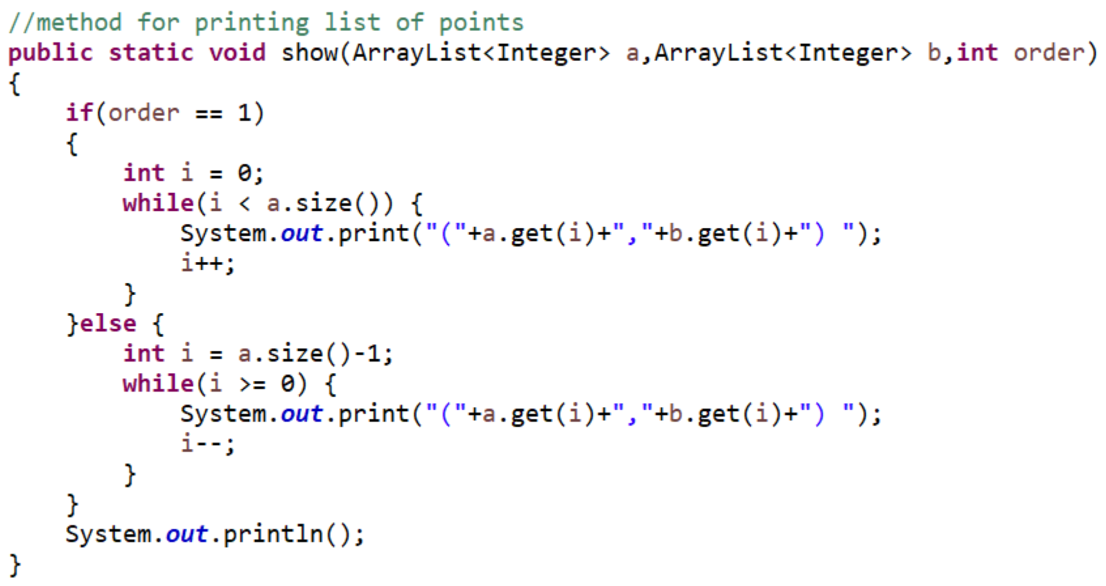
```
//translate origin to (x1,y1)
int i = 0;
while(i < line.pointx.size())
{
    line.pointy.set(i, line.pointy.get(i) - ty);
    line.pointx.set(i, line.pointx.get(i) - tx);
    i++;
}

//show
show(line.pointx,line.pointy,order);
```

Translate point to the old position and show the result.

```
            //clear list of all point
            line.pointx.removeAll(line.pointx);
            line.pointy.removeAll(line.pointy);

            //ask to continue
            System.out.print("Continue? (y/n) ");
            temp = in.nextLine();


        }while(!(temp.equalsIgnoreCase("n")));

        System.out.print("End Program");
    }
```

Clear array list after showing the result and then ask user whether continue?

If user press any key except “n” the program will loop again to render the new line but if user press key “n” the program will end.



```java
//method for printing list of points
public static void show(ArrayList<Integer> a,ArrayList<Integer> b,int order)
{
    if(order == 1)
    {
        int i = 0;
        while(i < a.size()) {
            System.out.print("("+a.get(i)+","+b.get(i)+") ");
            i++;
        }
    }else {
        int i = a.size()-1;
        while(i >= 0) {
            System.out.print("("+a.get(i)+","+b.get(i)+") ");
            i--;
        }
    }
    System.out.println();
}
```

Method show is for show 2 array lists in coordinate format. And if start and end point was swap the order flag will set to be -1 and the list will swap for show the real order.