```
package gobalkrishnan_v_18_06_1995.graphics;

import java.util.ArrayList;
public class gki2Polygon {

public ArrayList<gki2Point> points=new ArrayList<>();
public ArrayList<gki2DotLine>   lines=new ArrayList<>();
public ArrayList<gki2Point> linepoint=new ArrayList<>();
public gkiColor p1c=new gkiColor(255,150,0,255),p2c=new
gkiColor(255,150,0,255),p3c=new gkiColor(255,150,0,255),p4c=new
gkiColor(255,150,0,255),
single=new gkiColor(255,150,0,255);
gki2QuickHull gg=new gki2QuickHull();

public void p1c(gkiColor c1){
   p1c=c1;
}
public void p2c(gkiColor c1){
   p2c=c1;
}
public void p3c(gkiColor c1){
   p3c=c1;
}
public void p4c(gkiColor c1){
   p4c=c1;
}
public void  gkiColor(gkiColor c){
   p1c(c);
   p2c(c);
   p3c(c);
   p4c(c);
}
```

```
public void gki2Point(gki2Point p){
        points.add(p);

}
    public void gki2Point(double x,double y){
        points.add(new gki2Point(x, y));

    }
    public void gki2Point(gki2Point[] p){
      for(int i=0;i<p.length;i++){
         points.add(p[i]);
      }
    }

    public void gki2Point(ArrayList<gki2Point> p){
        for(int i=0;i<p.size();i++){
            points.add(p.get(i));
        }

    }




    public double minX,maxX,minY,maxY;

    ArrayList<gki2Point> colorlinepoint=new ArrayList<>();
    ArrayList<gki2Point> colorpoint=new ArrayList<>();

    private void colorprocess(){
        try{
```

```
 colorlinepoint.removeAll(colorlinepoint);
 colorpoint.removeAll(colorpoint);
 gki2Point p1=new gki2Point(minX,minY);
 gki2Point p2=new gki2Point(minX,maxY);
 gki2Point p3=new gki2Point(maxX,maxY);
 gki2Point p4=new gki2Point(maxX,minY);
 colorpoint.add(p1);
 colorpoint.add(p2);
 colorpoint.add(p3);
 colorpoint.add(p4);
 if(c!=null){
 for(int i=0;i<colorpoint.size();i++){
    colorpoint.get(i).rotate(c, r);

 }}
 //colorpoint= gg.quickhull(colorpoint);


gki2DotLine l1=new gki2DotLine();
l1.sp(colorpoint.get(0));
l1.sc(p1c);
l1.ep(colorpoint.get(1));
l1.ec(p2c);
l1.processX();
colorpoint(l1);
gki2DotLine l2=new gki2DotLine();
l2.sp(colorpoint.get(1));
l2.sc(p2c);
l2.ep(colorpoint.get(2));
l2.ec(p3c);
l2.processX();
colorpoint(l2);
```

```
  gki2DotLine l3=new gki2DotLine();
  l3.sp(colorpoint.get(2));
  l3.sc(p3c);
  l3.ep(colorpoint.get(3));
  l3.ec(p4c);
  l3.processX();
  colorpoint(l3);

  gki2DotLine l4=new gki2DotLine();
  l4.sp(colorpoint.get(3));
  l4.sc(p4c);
  l4.ep(colorpoint.get(0));
  l4.ec(p1c);
  l4.processX();
  colorpoint(l4);


  gki2Point[] arr=new gki2Point[colorlinepoint.size()];

   for(int i=0;i<arr.length;i++){
      arr[i]=colorlinepoint.get(i);
   }
   gkiMergeSort gki=new gkiMergeSort();
   gki.sortAscendingY(arr);
   colorlinepoint.removeAll(colorlinepoint);

   for(int i=0;i<arr.length;i++){
      colorlinepoint.add(arr[i]);
   }
   }catch(NullPointerException|IndexOutOfBoundsException e){}
}
```

```
private void colorpoint(gki2DotLine g){
   for(int i=0;i<g.point.size();i++){
      gki2Point point=new gki2Point(g.point.get(i),g.color.get(i));
      colorlinepoint.add(point);


   }


}

public void processX(){
   lines.removeAll(lines);
   linepoint.removeAll(linepoint);
   if(!points.isEmpty()){
      for(int i=0,j=1;i<points.size();i++,j++){

         if(j==points.size()){
            j=0;
         }

          gki2DotLine l=new gki2DotLine();
          l.sp(points.get(i));
          l.ep(points.get(j));
          l.processX();
          lines.add(l);

      }

      for(int i=0;i<lines.size();i++){
         for(int j=0;j<lines.get(i).point.size();j++){
               linepoint.add(lines.get(i).point.get(j));
```

```
    }
}

gki2Point[] arr=new gki2Point[linepoint.size()];
double[] x_=new double[linepoint.size()];
double[] y_=new double[linepoint.size()];

for(int i=0;i<arr.length;i++){
    arr[i]=linepoint.get(i);
    x_[i]=linepoint.get(i).x;
    y_[i]=linepoint.get(i).y;
}
gkiMergeSort gki=new gkiMergeSort();
gki.sortAscendingY(arr);
gki.sortAscending(x_);
gki.sortAscending(y_);
minX=x_[0];
minY=y_[0];
maxX=x_[x_.length-1];
maxY=y_[x_.length-1];

colorprocess();



try{
    int count=0;
for(int i=0,j=1;i<arr.length-1;i++,j++){

      int a=(int)arr[i].y;
      int b=(int)arr[j].y;
      gki2Line l=new gki2Line();
```

```
if(a==b  ){


   gki2Point u=arr[i];
   gki2Point v=arr[j];

   if(u.x>v.x){
      gki2Point t=u;
      u=v;
      v=t;
    //  System.out.println(count);
   }
   count++;
   l.sp(u);
   l.ep(v);
  // System.out.println(l);

   gki2Point x1=colorlinepoint.get(i);
   gki2Point y1=colorlinepoint.get(j);

   if(x1.x>y1.x){
      gki2Point t=x1;
      x1=y1;
      y1=t;
    //  System.out.println(count);
   }
  //  System.out.println(x1+":"+y1);
   //if(j<colorlinepoint.size()){
      l.sc(x1.color);
      l.ec(y1.color);
  //}
```

```
                for(int k=0;k<l.point.size();k++){
                  linepoint.add(new gki2Point(l.point.get(k),l.color.get(k)));

                   }

              }
          }
       }catch(IndexOutOfBoundsException|NullPointerException e){

       }

     }

  }

  public void translateX(double x){
     for(int i=0;i<points.size();i++){
        points.get(i).translateX(x);
     }
  }
  public void translateY(double y){
     for(int i=0;i<points.size();i++){
        points.get(i).translateY(y);
     }
  }


  public void translate(double x,double y){
     translateX(x);
     translateY(y);
  }
  public void rotate(gki2Point c,double r){
    for(int i=0;i<points.size();i++){
        points.get(i).rotate(c, r);

    }
    points= gg.quickhull(points);
//System.out.println(points);
    this.c=c;
    this.r=r;
}

public void colorrotate(gki2Point c,double r){

//System.out.println(points);
    this.c=c;
    this.r=r;
}
gki2Point c;
double r;

public void shearX(double shx){
    for(int i=0;i<points.size();i++){
        points.get(i).shearX(shx);;
    }

}

public void shearY(double shy){
    for(int i=0;i<points.size();i++){
        points.get(i).shearY(shy);;
    }

}
```

```
public void scaleX(double shx){
   for(int i=0;i<points.size();i++){
      points.get(i).scaleX(shx);;
   }
}

public void scaleY(double shy){
   for(int i=0;i<points.size();i++){
      points.get(i).scaleY(shy);;
   }
}

public void shear(double x,double y){
   shearX(x);
   shearY(y);
}
public void scale(double x,double y){
   scaleX(x);
   scaleY(y);
}



private double fabs(double d) {
   // TODO Auto-generated method stub
   if(d<0){
      d*=-1;
   }
   return d;
}
```

```
    private void alongXaxis(){
        for(int i=0;i<linepoint.size();i++){

            gki2Point po=linepoint.get(i);
        }
    }

}
```