2024/02/29 112-1 期末檢討

1. 撰寫程式完成以下要求，
(a) 設計一類別 DataSet 可以記錄多個二維資料點，每個二維資料點以類別 TwoDimPoint 的物件表示
(b) 對一個 DataSet 物件，找出一個最小矩形(以Rectangle 類別的物件表示)，滿足以下條件：
  (i) 矩形邊線與 X, Y 軸平行或垂直，
  (ii) DataSet 內所有資料點位於矩形內(落在矩形的邊線上也算)
(c) 對於 DataSet 物件 A 和 B, 個別找出最小矩形 R_A 和 R_B符合上面(b)之條件，將所有落於 R_A 與 R_B 交集區域的資料點找出來，記錄在新的 DataSet 物件 ibAB
(d) 對一個 DataSet 物件，找出一個最小圓形(以Circle 類別的物件表示)，滿足以下條件：
  (i) 圓心為 DataSet 物件所包含的所有資料點之中心，
  (ii) DataSet 內所有資料點位於圓內(落在圓的邊線上也算)
(e) 對於 DataSet 物件 A 和 B, 找出其最小圓形 C_A 和 C_B符合(d)之條件，將所有落於 C_A 與 C_B 交集區域的資料點找出來，記錄在新的 DataSet 物件 icAB

依類別分開敘述：

1. main

- 由此範例可測程式正確性

```
/* 2024/02/29 楊育哲
 * 複習上學期期末( 112_1 )
*/
public class reviewFinal {
    public static void main(String args[]) {
        final int sizeOfSetA=4, sizeOfSetB=7;
        DataSet dataSetA, dataSetB;

        //initialize dataSetA
        int ax[]={5, 8, 7, 10}, ay[]={9, 12, 6, 7};
        // int ax[]={5, 8, 7, 10, 11}, ay[]={9, 12, 6, 7, 7};
        TwoDimPoint TwoDimPoints[] = new TwoDimPoint[sizeOfSe
        for(int i=0; i<sizeOfSetA; i++){
            TwoDimPoints[i]=new TwoDimPoint((double)ax[i], (d
        }
        dataSetA = new DataSet(TwoDimPoints);
        //initialize dataSetB
        int bx[]={10, 9, 12, 8, 10, 6, 13}, by[]={8, 6, 6, 5,
        // int bx[]={10, 9, 12, 8, 10, 6}, by[]={8, 6, 6, 5,
```

```
        dataSetB = new DataSet();
        for(int i=0; i<sizeOfSetB; i++){
            dataSetB.addTwoDimPoint(new TwoDimPoint(bx[i],by[
        }

        //print data set A and B (20%)
        System.out.println("setA: "+dataSetA);
        System.out.println("setB: "+dataSetB);

        //print bounding box of set A and B (20%)
        System.out.println("setA BB: "+dataSetA.findBoundingB
        System.out.println("setB BB: "+dataSetB.findBoundingB

        //find data points localed within the intersection of
        DataSet ibAB = dataSetA.boundingBoxInterset(dataSetB)
        System.out.println("data points within BB intersectio

        //print bounding CIRCLE of set A and B (20%)
        System.out.println("setA BC: "+dataSetA.findBoundingC
        System.out.println("setB BC: "+dataSetB.findBoundingC

        // find data points localed within the intersection o
        DataSet icAB = dataSetB.boundingCircleInterset(dataSe
        System.out.println("data points within BB intersectio
    }
}
```

程式輸出：<注意，並非輸出結果正確就可以得滿分，評分時對於：類別設計的邏輯、程式寫作風格等因素會綜合考量>

```
setA: (5, 9), (8, 12), (7, 6), (10, 7)                          // (a) 輸出結果
setB: (10, 8), (9, 6), (12, 6), (8, 5), (10, 5), (6, 2), (13, 1)  // (a) 輸出結果
setA BB: bottom Left=(5, 6), width=5, height=6                  // (b) 輸出結果
setB BB: bottom Left=(6, 1), width=7, height=7                  // (b) 輸出結果
data points within BB intersection: (7, 6), (10, 7), (10, 8), (9, 6)  // (c) 輸出結果
setA BC: center=(7.5, 8.5), radius=3.5355                       // (d) 輸出結果
setB BC: center=(9.7, 4.7), radius=4.959                        // (d) 輸出結果
data points within BC intersection: (10, 8), (9, 6), (8, 5), (7, 6), (10, 7)  // (e) 輸出結果
```

輸出比對:

```
setA:  (5.0,9.0),(8.0,12.0),(7.0,6.0),(10.0,7.0)
setB:  (10.0,8.0),(9.0,6.0),(12.0,6.0),(8.0,5.0),(10.0,5.0),(6.0,2.0),(13.0,1.0)
setA BB: bottom left=(5.0,6.0), width=5.0, height=6.0
setB BB: bottom left=(6.0,1.0), width=7.0, height=7.0
data points within BB intersection:  (7.0,6.0),(10.0,7.0),(10.0,8.0),(9.0,6.0)
setA BC: center=(7.5,8.5), radius=3.5355
setB BC: center=(9.7,4.7), radius=4.959
data points within BB intersection:  (10.0,8.0),(9.0,6.0),(8.0,5.0),(7.0,6.0),(10.0,7.0)
```

2. Rectangle

- 矩形類別 包跨:

  data member

  - double x // botton left coord x
  - double y // botton left coord y
  - double width // rectangle width
  - double height // rectangle height

  function member (此忽略建構子與toString)

  - boolean contains(TwoDimPoint pt) //回傳點是否在矩形內

```
class Rectangle{
    private double x, y, width, height;
    Rectangle(double bottomLeftX, double bottomLeftY, double
        this.x = bottomLeftX;
        this.y = bottomLeftY;
        this.width = init_w;
        this.height = init_h;
    }
    boolean contains(TwoDimPoint pt){
        final double ptX = pt.getX();
        final double ptY = pt.getY();
        if(ptX>=x&&ptX<=x+width&&ptY>=y&&ptY<=y+height){
            return true;
        }else return false;
    }
    public String toString(){
        return "bottom left=("+this.x+","+this.y+"), width="+
```

```
    }
}
```

3. Circle

- 圓形類別 包跨:

  data member

  - TwoDimPoint center   // 圓心座標點
  - double radius   // 半徑

  function number

  - boolean contains(TwoDimPoint pt) // 回傳點是否在圓內

```
class Circle{
    private TwoDimPoint center;
    private double radius;
    Circle(TwoDimPoint init_center, double init_radius){
        this.center = new TwoDimPoint(init_center.getX(), ini
        this.radius = init_radius;
    }
    boolean contains(TwoDimPoint pt){
        return (center.getDistanceTo(pt)<=radius);
    }
    public String toString(){
        return "center=("+(int)(this.center.getX()*10)/10.0+"
    }
}
```

4. TwoDimPoint

- 點類別 包跨:

  data member

  - double x   // coord x
  - double y   // coord y

  function member

- setX ; setY　// 設定x或y
- getX ; getY　// 取得x或y
- getDistanceTo(TwoDimPoint pt) // 計算自身點位置與pt位置之距離

```
class TwoDimPoint{
    private double x, y;
    public TwoDimPoint(double init_x, double init_y){
        this.x = init_x;
        this.y = init_y;
    }
    public void setX(double new_x){
        this.x = new_x;
    }
    public void setY(double new_y){
        this.y = new_y;
    }
    public double getX(){
        return this.x;
    }
    public double getY(){
        return this.y;
    }
    public double getDistanceTo(TwoDimPoint pt){
        return (Math.sqrt(Math.pow(pt.getX()-x, 2)+Math.pow(p
    }
    public String toString(){
        return "("+(int)(this.x*10)/10.0+","+(int)(this.y*10)
    }
}
```

5. DataSet

- 多型建構子: 可使用TwoDimPoint[]建立
- 資料集類別 存取一些座標點 包跨:

  data member

  - int capacity　　// 容量

- int top   // 座標點量
- TwoDimPoint[] points // 座標點陣列

function member

- int size()   // get top (取得座標點量)
- TwoDimPoint getPt(int index) // 取得相應位置的座標點物件(new)
- void addTwoDimPoint(TwoDimPoint pt) // 新增座標點進points
- Rectangle findBoundingBox() // 取得最小含括所有座標點的矩形
- Circle findBoundingCircle()   // 取得最小含括所有座標點的圓形
- DataSet boundingBoxInterset(DataSet B) // 取得兩資料集之最小矩形重疊處的座標點集合
- DataSet boundingCircleInterset(DataSet B) // 取得兩資料集之最小圓形重疊處的座標點集合

```
class DataSet{
    final int defaultCapacity=10;
    private int capacity, top;
    private TwoDimPoint[] points;
    public DataSet(){
        this.capacity = defaultCapacity;
        this.top = 0;
        this.points = new TwoDimPoint[capacity];
    }
    public DataSet(TwoDimPoint[] init){
        this.capacity = init.length*2;
        this.top = init.length;
        this.points = new TwoDimPoint[capacity];
        for(int i=0; i<init.length; i++){
            points[i] = new TwoDimPoint(init[i].getX(), init[
        }
    }
    public int size(){
        return top;
    }
    public TwoDimPoint getPt(int index){
```

```
        if(index>=0&&index<top){
            return new TwoDimPoint(points[index].getX(), poin
        }
        return null;
    }
    public void addTwoDimPoint(TwoDimPoint newPoint){
        if(top==capacity-1){
            TwoDimPoint[] tmp = new TwoDimPoint[capacity*2];
            for(int i=0; i<top; i++){
                tmp[i] = new TwoDimPoint(points[i].getX(), po
            }
            capacity*=2;
            points = new TwoDimPoint[capacity];
            points = tmp;
        }
        points[top++] = new TwoDimPoint(newPoint.getX(), newP
    }
    public Rectangle findBoundingBox(){
        double rx=1000, ry=1000, rw=0, rh=0;
        for(int i=0; i<top; i++){
            rx = Math.min(rx, points[i].getX());
            ry = Math.min(ry, points[i].getY());
            rw = Math.max(rw, points[i].getX());
            rh = Math.max(rh, points[i].getY());
        }
        return new Rectangle(rx, ry, Math.abs(rw-rx), Math.ab
    }
    public Circle findBoundingCircle(){
        double cx=0, cy=0, cr=0;
        for(int i=0; i<top; i++){
            cx+=points[i].getX();
            cy+=points[i].getY();
        }
        TwoDimPoint cp = new TwoDimPoint(cx/top, cy/top);
        for(int i=0; i<top; i++){
            double distance = cp.getDistanceTo(points[i]);
            cr = Math.max(cr, distance);
        }
```

```
        return new Circle(cp, cr);
    }
    public DataSet boundingBoxInterset(DataSet B){
        DataSet result = new DataSet();
        Rectangle rA = findBoundingBox();
        Rectangle rB = B.findBoundingBox();
        for(int i=0; i<size(); i++){
            if(rB.contains(points[i]))
                result.addTwoDimPoint(points[i]);
        }
        for(int i=0; i<B.size(); i++){
            if(rA.contains(B.getPt(i)))
                result.addTwoDimPoint(B.getPt(i));
        }
        return result;
    }
    public DataSet boundingCircleInterset(DataSet B){
        DataSet result = new DataSet();
        Circle cA = findBoundingCircle();
        Circle cB = B.findBoundingCircle();
        for(int i=0; i<size(); i++){
            if(cB.contains(points[i]))
                result.addTwoDimPoint(points[i]);
        }
        for(int i=0; i<B.size(); i++){
            if(cA.contains(B.getPt(i)))
                result.addTwoDimPoint(B.getPt(i));
        }
        return result;
    }
    public String toString(){
        String s="";
        for(int i=0; i<top; i++){
            s += ((i>0)?",":" ")+points[i];
        }
        return s;
    }
}
```