ASSIGNMENT (SLOT 17)

Read the instructions below carefully before start coding.

Students are ONLY allowed to use:

- Materials on his/her computer (including JDK, NetBeans...).
- For distance learning: Google Meet, Hangout (for Exam Monitoring Purpose).

Follow the steps below to complete PE:

1. Create a folder to save given projects, e.g. CSD_given (1). Down load given materials to (1).
2. Steps to do question 1 (do the same for questions 2 and 3): Open NetBeans, open the given Q1 project, then edit the MyList.java file according to the requirements of the exam. (edit the file BSTree.java for Q2 and Graph.java for Q3).
3. Before submission: Run the function **"Clean and Build Project"** (Shift+F11), to ensure BUILD SUCCESSFUL (if not, the project will get 0 mark).
4. **Submission**: to submit the project Q1, at first you must select Question No = 1, browse and select the project folder (e.g. 1, Q1 or Q1X,...) then click the **Submit** button. Do the same for other questions. **Do not submit** the un-edited given project. If project is too big for submission, delete all f1.txt, f2.txt,....
5. **Do not use accented Vietnamese** when writing comments in programs.
6. **Do not add** new **import** statement(s) to given files.
7. Software tools must be used: **NetBeans IDE 8.x** and **Java JDK 1.8**.
   If at least one of the above requirements is not followed, the exam will get ZERO.

Trouble shooting:

If the given project (e.g. Q1) runs with error, you need to run "Clean and Build Project" (Shift+F11). If still error, try to rename or copy the project to other one, e.g. from Q1 to Q1X or Q1Y,...

==========

Write a Interface named **Sound** with the following information

Sound
+ **void playSound();**

Write a Interface named **Rotations** with the following information

Rotations
+ **void rotate();**

Write a abstract class named **Shape** implements Rotations, Sound, Comparable with the following information

Shape
+ **code:** int. Access modifier: private
+ **abstract float perimeter():**Calculating the perimeter of Shape. Access modifier: public

+ **abstract float area()**: Calculating the area of Shape. Access modifier: public
+ **getCode()**: int, return code, Access modifier: public
+ **setCode(int code)**: Access modifier: public
+ overriding playSound() and rotate()

Write a class named **Point** with the following information

Point

+ **x, y:** float. Access modifier: private.

+ **Point(){}**
+ **Point(px,py):** Constructor that assigns x=px, y=py
+ **getX()**: return x;
+ **getY()**: return y;
+ setX(float x);
+ setY(float y);
+ distanceTo(Point A): Access modifier: public. Return distance to A;
+ String toString():Access modifier: public. Return "("+ x + ", " + y + ')'

Write a class named **Triangle** that extends Shape

Triangle

+ **A,B,C:** Point

+ **Triangle(Point A, Point B, Point C)**: constructor triangle object with point A, point B and point C. Access modifier: public
+ **Point center()**: String, return center point of triangle
+ **public int compareTo(Object o)**{

```
    Triangle st = (Triangle)o;
    int no=getCode();
    if(no>st.getCode())
        return 1;
     else if(no == st.getCode())
        return 0;
     else
        return -1;
  }
```

+ Override abstract methods of the super class

Write a class named **Col** with the following member

Col

+ **listOfPoints:** List. Access modifier: default.
+ **setOfPAs:** Set. Access modifier: default.
+ **mapOfCenters:** Map. Access modifier: default.

+ **Col()**:initialize listOfPoints=new ArrayList(); setOfPAs=new TreeSet(); map Of Centers=new TreeMap();
+ **setListOfPoints(Object a[])**: listOfPoints includes Points taken from A.
+ **displayListOfPoints:** display list of Points

+ **setSetOfPAs (Point A, Point B)**: set of Triangles, if t is a triangle then element t in setOfPAs is "("+t.getCode()+": perimeter="+t.perimeter()+",area="+t.area() +")". Access modifier: public.
+ **displaySetOfPAs()**: display SetOfPAs. Access modifier: public.
+ **setMapOfCenters(Point A, Point B):** Map of triangle centers, if t is a triangle then element t in mapOfCenters is ((Integer)t.getCode()).toString(), t.center(). Access modifier: public.
+ **displayMapOfCenters()**: display MapOfCenters()

Program output might look something like:

Test case 1 (2 points):

1. Test list Of points:
2. Test set of triangle perimeter and area:
3. Test map of triangle centers:
Your selection (1 -> 3): 1
OUTPUT:
List of point:
[(0.0, 1.0), (1.0, 1.0), (2.0, 1.0), (3.0, 1.0), (4.0, 1.0), (5.0, 1.0), (6.0, 1.0), (7.0, 1.0), (8.0, 1.0), (9.0, 1.0)]

Test case 2 (2 points):

1. Test list Of points:
2. Test set of triangle perimeter and area:
3. Test map of triangle centers:
Your selection (1 -> 3): 2
OUTPUT:
Set of triangle perimeters and areas:
(0: perimeter=2.29,area=1.0) (1: perimeter=2.2,area=1.01) (2: perimeter=2.29,area=1.0) (3: perimeter=2.57,area=1.01) (4: perimeter=2.9,area=1.01) (5: perimeter=3.21,area=1.01) (6: perimeter=3.5,area=1.0) (7: perimeter=3.77,area=1.01) (8: perimeter=4.02,area=1.0) (9: perimeter=4.26,area=1.0)

Test case 3(2 points):

1. Test list Of points:
2. Test set of triangle perimeter and area:
3. Test map of triangle centers:
Your selection (1 -> 3): 3
OUTPUT:
Map of triangle centers:
{Triangle[0](0.67, 0.33)} {Triangle[1](1.0, 0.33)} {Triangle[2](1.33, 0.33)} {Triangle[3](1.67, 0.33)} {Triangle[4](2.0, 0.33)} {Triangle[5](2.33, 0.33)} {Triangle[6](2.67, 0.33)} {Triangle[7](3.0, 0.33)} {Triangle[8](3.33, 0.33)} {Triangle[9](3.67, 0.33)}