L3 Informatique
EXAMEN de POO/JAVA : PARTIE 1

Introduction

Licence Informatique, 20 April 2009 (created 6 April 2009)

no tags

- Cette première partie comprend quatre questions.
- Vous devez répondre directement sur cette feuille d'énoncé

Question 01

Licence Informatique, 20 April 2009 (created 30 January 2009)

no tags

A. Définir en JAVA une classe `Geometrie2D` pour représenter en toute généralité de tels objets. Une instance de cette classe possèdera une position dans l'espace 2D représentée par ses coordonnées `xcor` et `ycor` de type `double`. Compléter le code suivant :

```
public class Geometrie2D {
    _________________ double xcor ;
    _________________ double ycor ;
    public Geometrie2D(double x, double y) {_____________________________}
    public _________________ getXcor() {_____________________________}
    public _________________ getYcor() {_____________________________}
    public _________________ setXcor(_____________________________) {_____________________________
    public _________________ setYcor(_____________________________) {_____________________________
}
```

B. Un **cercle** est une sorte d'objet `Geometrie2D` qui possède en plus un `rayon` de type `double`. Compléter le code suivant :

```
public class Cercle extends _____________________________ {
    _________________ double rayon;
    public Cercle(){_____________________________}
    public Cercle(double r) {_____________________________}
    public Cercle(double x, double y, double r){_____________________________}
    public _________________ getRayon() {_____________________________}
    public _________________ setRayon(_____________________________) {_____________________________
    public String toString() {
       return "Cercle de rayon:"+rayon+" positionné en:("+getXcor()+","+getYcor()+")";
    }
}
```

Par exemple, en exécutant la procédure `main` suivante :

```
public static void main(String [] args){
      List<Cercle> a = new ArrayList<Cercle>() ;
      a.add(new Cercle()) ; a.add(new Cercle(10)) ; a.add(new Cercle(20,30,12)) ;
      for(Cercle c : a){System.out.println(c);}
}
```

vous devez obtenir le résultat :

```
Cercle de rayon:0.0 positionné en:(0.0,0.0)
Cercle de rayon:10.0 positionné en:(0.0,0.0)
Cercle de rayon:12.0 positionné en:(20.0,30.0)
```

Question 02

Licence Informatique, 20 April 2009 (created 30 January 2009)

no tags

A. On suppose maintenant qu'un cercle est "**zoomable**" au sens où l'on peut modifier son rayon en appliquant un coefficient multiplicatif positif. Le rayon diminue si le coefficient du zoom est strictement inférieur à 1 et augmente s'il est strictement supérieur à 1. Compléter le code suivant qui définit l'interface des objets "Zoomables" :

```
public interface Zoomable {void zoomer(double coef) ____________________ }
```

et modifier en conséquence la classe `Cercle`.

Par exemple, en exécutant la procédure `main` suivante :

```
public static void main(String [] args){
      List<Cercle> a = new ArrayList<Cercle>() ;
      a.add(new Cercle()) ; a.add(new Cercle(10)) ; a.add(new Cercle(20, 30,12)) ;
      for(Cercle c : a){System.out.println(c);} ; System.out.println("-----------");
      a.get(1).zoomer(1.1) ; a.get(2).zoomer(0.9) ; for(Cercle c : a){System.out.printl
}
```

Vous devez obtenir le résultat :

```
Cercle de rayon:0.0 positionné en:(0.0,0.0)
Cercle de rayon:10.0 positionné en:(0.0,0.0)
Cercle de rayon:12.0 positionné en:(20.0,30.0)
-----------
Cercle de rayon:0.0 positionné en:(0.0,0.0)
Cercle de rayon:11.0 positionné en:(0.0,0.0)
Cercle de rayon:10.8 positionné en:(20.0,30.0)
```

B. Afin de trier des cercles, on souhaite pouvoir **comparer** deux cercles selon leur rayon.
Modifier en conséquence la classe `Cercle`.

Par exemple, en exécutant la procédure `main` suivante :

```
public static void main(String [] args){
      List<Cercle> a = new ArrayList<Cercle>() ;
      a.add(new Cercle());a.add(new Cercle(10));a.add(new Cercle(20, 30,12));
      for(Cercle c : a){System.out.println(c);}
      System.out.println("-----------");
      a.get(1).zoomer(1.1) ; a.get(2).zoomer(0.9);for(Cercle c : a){System.out.println(
      System.out.println("-----------");
      Collections.sort(a) ; for(Cercle c : a){System.out.println(c);}
    }
```

Vous devez obtenir le résultat :

```
Cercle de rayon:0.0 positionné en:(0.0,0.0)
Cercle de rayon:10.0 positionné en:(0.0,0.0)
Cercle de rayon:12.0 positionné en:(20.0,30.0)
-----------
Cercle de rayon:0.0 positionné en:(0.0,0.0)
Cercle de rayon:11.0 positionné en:(0.0,0.0)
Cercle de rayon:10.8 positionné en:(20.0,30.0)
-----------
Cercle de rayon:0.0 positionné en:(0.0,0.0)
Cercle de rayon:10.8 positionné en:(20.0,30.0)
Cercle de rayon:11.0 positionné en:(0.0,0.0)
```

Question 03

Licence Informatique, 20 April 2009 (created 30 January 2009)

no tags

A. Dans cette question tous les objets `Geometrie2D` sont **dessinables**, c'est à dire qu'ils doivent pouvoir répondre au message `dessiner(Graphics g)` en dessinant leur représentation graphique dans le contexte graphique `g` fourni en paramètre.
Modifier `Geometrie2D` en conséquence.

B. Modifier la classe `Cercle` afin de pouvoir dessiner des cercles.

On pourra utiliser la méthode `drawOval` de la classe `Graphics`

```
public abstract void drawOval(int x, int y, int width, int height)
    The result is a circle or ellipse that fits within the rectangle specified by the x
    Parameters:
        x - the x coordinate of the upper left corner of the oval to be drawn.
        y - the y coordinate of the upper left corner of the oval to be drawn.
        width - the width of the oval to be drawn.
        height - the height of the oval to be drawn.
```

Pour tester cette application, on pourrait, par exemple, utiliser la classe `Test` suivante :

```
public class Test extends JPanel {
       public static void main(String[] args) {
               Test panel=new Test();
               panel.setPreferredSize(new Dimension(400,400));
               JFrame frame=new JFrame("Test") ; frame.add(panel);
               frame.pack() ; frame.setVisible(true);
       }
       public void paint(Graphics g) {
           Cercle c=new  Cercle(20, 100, 100) ; c.dessiner(g);
           c.zoomer(2) ; c.dessiner(g);
       }
```

C. Un **cadre** est une sorte d'objet `Geometrie2D` qui possède en plus une `largeur` et une `hauteur`. Un Cadre peut indiquer si, oui ou non, un point de coordonnées (x,y) est sur un de ses quatre bords ; la méthode `public boolean bord(double x, double y){...}` permettra d'exprimer ce comportement.
Définir la classe `Cadre` :

```
public class Cadre









}
```

On pourra utiliser la méthode `drawRect` de la classe `Graphics`

```
public void drawRect(int x, int y, int width, int height)
The left and right edges of the rectangle are at x and x + width.
The top and bottom edges are at y and y + height.
    Parameters:
        x - the x coordinate of the rectangle to be drawn.
        y - the y coordinate of the rectangle to be drawn.
        width - the width of the rectangle to be drawn.
        height - the height of the rectangle to be drawn.
```

Pour tester cette application, on pourrait par exemple, utiliser la classe `Test` suivante :

```
public class Test extends JPanel {
       private static int Xmax=500 ; private static int Ymax=500 ;
       public static void main(String[] args) {
              Test panel=new Test();
              panel.setPreferredSize(new Dimension(Xmax,Ymax));
              JFrame frame=new JFrame("Test");frame.add(panel);
              frame.pack();frame.setVisible(true);
       }
       public void paint(Graphics g) {Cadre c=new Cadre(10,10,Xmax-20,Ymax-20) ; c.dess
}
```

Question 04

Licence Informatique, 20 April 2009 (created 6 April 2009)

no tags

A. Une **particule** est une sorte d'objet `Geometrie2D` qui possède en plus un `cap` dans la direction duquel elle peut avancer.
Le `cap` est exprimé en degré ; la valeur **0** correspond à la direction **Est**, une particule est dirigée vers le **Sud** si son cap est de **90°**, vers l'**Ouest** pour **180°** et vers le **Nord** pour un cap de **270°**. Une particule peut `tourner` d'un certain angle et ainsi modifier son `cap`. Une particule peut `avancer` d'une certaine longueur dans la direction de son cap.
Compléter la classe suivante :

```
public class Particule ____________________________ {
    ____________________ double cap ;
    public Particule(double x, double y, double c) { ____________________________ }
    public void dessiner(Graphics g) {
         // dessiner un cercle pour matérialiser la position de la particule
         g.drawOval( ________________________________ , ________________________________ ,
         // dessiner un segment de droite pour matérialiser le cap de la particule
         double capR=this.cap*Math.PI/180 ; // conversion degre2radian
         g.drawLine((int)Math.round(this.getXcor()) ,
                    (int)Math.round(this.getYcor()) ,
                    (int)(this.getXcor()+Math.round(10*Math.cos(capR))) ,
                    (int)(this.getYcor()+Math.round(10*Math.sin(capR))));
    }
    public void tourner(double angle){ ________________________________ }
    public void avancer(double lo){
        double capEnRadian=cap*Math.PI/180;
        setXcor(getXcor()+lo*Math.cos(capEnRadian));
        setYcor(getYcor()+lo*Math.sin(capEnRadian));
    }
}
```

B. Un **cyclope** est une sorte de `Particule` qui possède en plus un angle de `vision` et un `domaine` dans lequel il peut se déplacer. Lors d'un déplacement, un cyclope avance d'une unité dans une direction qui dévie aléatoirement par rapport à son `cap` sans toutefois aller au delà de la zone couverte par son angle de `vision`. Si un cyclope touche un des quatre bords de son domaine alors il fait demi-tour.
Compléter la classe suivante :

```
public class Cyclope ____________________  {
    _________________ double vision ; _________________ Cadre domaine ;
    public Cyclope(double x, double y, double c, double v, Cadre d) {
       ___________________________________ }
    public ___________________  getVision() { _____________________________ }
    public ___________________  setVision(double v) { _____________________________ }

    public void deplacer(){
       if (domaine.bord( _____________________________ ) { _____________________________ ;}
       else {tourner(Math.random()*2*vision - vision);}
       avancer(1);
    }
```