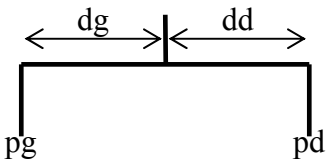
programmation

examen

Un mobile est composé de boules et de tiges. Une boule a un poids. Une tige est composée de deux mobiles accrochés à ses deux extrémités, et d'un point d' « accrochage ». Une tige n'a pas de poids. Nous nous proposons de définir trois classes permettant de représenter de tels mobiles.
La classe Mobile est une classe abstraite définie par :
La méthode *getPoids* permet d'obtenire le poids d'un mobile. Le poids d'une *Tige*, est la somme des poids des mobiles supspendus à ses deux extrémités.
La méthode *estEquilibre* retourne *true* si le mobile est équilibré, et *false* sinon. Une *Boule* est toujours equilibrée, une *Tige* est equilibrée si pg*dg=pd*dd. *pg* est le poids du mobile de gauche, et *pd* le poids du mobile de droite.



La méthode *equilibrer* permet d'equilibrer un mobile. Pour équilibrer un mbile on déplace le point d'accrochage.
La méthode *affiche* dessine une représentation graphique du mobile en utilisant le *Graphics g*.
Les méthodes *largeur* et *hauteur* permettent de calculer la largeur et la hauteur d'un mobile.

Question 1 (3 points)

Définir les constructeurs des classes *Boule* et *Tige*. Le constructeur de la classe Tige lève une exception quand le point d'accrochage est supérieur à la longueur de la tige.

Question 2 (3 points)

Définir les méthodes *getPoids* des classes *Boule* et *Tige*. Le poids d'un mobile de type *Tige* est la somme des poids des mobiles accrochés aux deux extrémités.

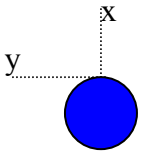
Question 3 (3 points)

Définir les méthodes *estEquilibre* des classes *Boule* et *Tige*.

Question 4 (3 points)

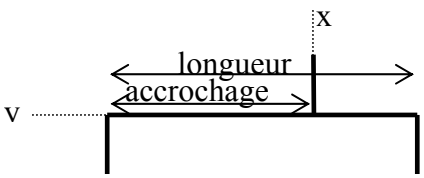
Définir les méthodes *equilibrer* des classes *Boule* et *Tige*.

Question 5 (3 points)



Définir les méthodes *affiche* des classes *Boule* et *Tige*. L'affichage d'une boule dessine un cercle dont la surface est proportionelle au poids de la boule.L'affichage d'une tige est un trait horizontal de longueur la longueur de la tige, et aux deux extrémités des traits verticaux de longueur 30.
La classe *Graphics* est munie des méthodes :

- *drawOval(int x, int y, int l, int h)* qui dessine une ellipse inscrite dans le rectangle de coin haut gauche *(x, y)* de largeur *l*, et de hauteur *h*.
- *drawLine(int x1, int y1, int x2, int y2)* qui dessine une ligne du point *(x1, y1)* au point *(x2, y2)*.



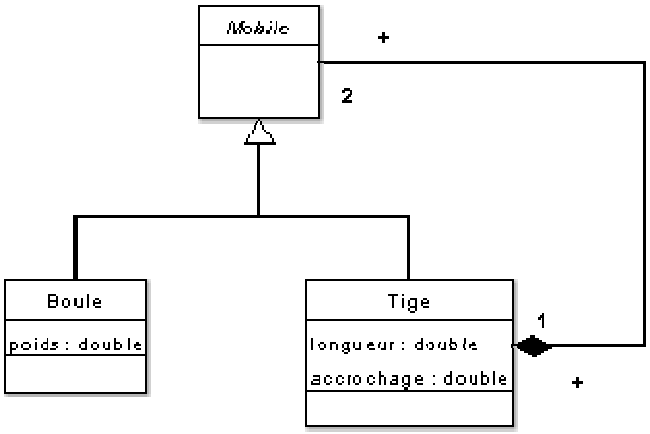
Question 6 (3 points)

Définir les méthodes *hauteur* des classes *Boule* et *Tige*. La hauteur d'un mobile est définie en nombre de pixels.

Question 7 (3 points)

Définir les méthodes *largeur* des classes *Boule* et *Tige*. La largeur d'un mobile est définie en nombre de pixels. Ne pas oublier qu'un mobile est mobile, et que donc, il peut tourner autour de son point d'accrochage. La largeur est l'espace que peut occuper le mobile pendant une rotation complète.

Annexes :




```
abstract public class Mobile {
 public Mobile() ;
 public abstract double getPoids();
 public abstract boolean estEquilibre();
 public abstract void equilibrer();
 public abstract void affiche(Graphics g,
                              int x, int y);
 public abstract double hauteur();
 public abstract double largeur();
}
```

```
public class Boule
              extends Mobile {
 private double poids;
 public Boule (double poids) ;
 public double getPoids();
 public boolean estEquilibre();
 public void affiche(Graphics g,
                     int x, int y);
 public void equilibrer();
 public double largeur();
 public double hauteur();
}
```

```
public class Tige
               extends Mobile {
 private double longueur, accrochage ;
 private Mobile gauche, droit ;
 public Tige throws Exception (
              double longueur,
              double accrochage ,
              Mobile gauche,
              Mobile droit) ;
 public double getPoids();
 public boolean estEquilibre();
 public void affiche(Graphics g,
                              int x, int y);
 public void equilibrer();
 public double largeur();
 public double hauteur();
}
```