Université du Maine L3-INF338B

Programmation Orienté Objet.

TD n° 8 : Ceci est le dernier TD

PARTIE 1 : Suis-je fort en Java ?

Question 1 : Qu'affiche ce programme ? Justifier.

```
class C {
     private int i = 0;
     public void inc() { i++; }
     public void println() { System.out.println(i); } }
class Run {
     public static void main (String args[]) {
          C c1=new C() ;
          C c2=new C() ;
          c1.println() ;
          c2.println() ;
          c2=c1 ;
          c1.inc() ;
          c1.println() ;
          c2.println() ;
     }
}
```

Question 3 : Le programme suivant compile-t-il ? Justifier.

```
// fichier Run.java
public class C {
     public void f() { System.out.println("C.f"); }}
public class Run {
     public static void main(String args[]) { C c = new C(); }
}
```

Question 5 : Ce programme compile-t-il ? Justifier et corriger si nécessaire. Qu'affiche-t-il ?

```
class C {
     private char c = 'a';
     private int i;
     private String s;
     private Double d;
     public C() {
          System.out.println("C : c = " + c + " - " +
               "i = " + i + " - " + "s = " + s + " - " + "d= " + d); }

     {    d = new Double(3.14);  System.out.println("d = " + d); }
     public void inc() { int k; k++; }
}
class Run {
     public static void main(String args[]) { C c = new C(); }
}
```

Question 7 : Qu'affiche ce programme ? Est-ce correct ? Justifier.

```
class C {
     private int i = 0;
     public C() { i = 1; f(); }
     public void f() { System.out.print("i = " + i); }
}
class D extends C {
     private int j = 0;
     public D() { j = 1; }
     public void f() { super.f(); System.out.println(" - j = " + j); }
}

class Run {
     public static void main(String args[]) {
          C c = new D();
          c.f();
     }
}
```

Question 8 : Ce programme compile-t-il ? Justifier.

```
class C {
  private int i = 0;
}

class D {
     private int i = 0;
     public D(int aInt) { i = aInt; }
}

class Run {
     public static void main(String args[]) {
          C c = new C();
          D d = new D();
     }
}
```

Question 17 : Ce programme ne compile pas. Pourquoi ? Proposer 2 corrections possibles.

```
class C
   public C(String s) {}
}
class D extends C {}

class Run
     public static void main(String args[]) {
          D d = new D();
     }
}
```

Question 18 : Qu'affiche ce programme ? Justifier.

```
class C {
   private void f() { System.out.println("C.f()"); }
   void g() { f() ;}
}

class D extends C {
   private void f() { System.out.println("D.f()"); }
}
class Run {
     public static void main(String args[])
          C c=new D();
          c.g();
     }
}
```

Question 19 : Ce programme compile-t-il ? Justifier.

```
interface C {
     String s = "My string";
     int i;
     void f();
}

class D implements C {
     void f() { }
}

class Run {
     public static void main(String args[]) {
          System.out.println(D.s + " - " + D.i);
     }
}
```

Question 22 : Qu'affiche le programme suivant ? Justifier.

```
abstract class C {
     abstract public void f();
     public C() { f(); }
}

class D extends C {
     private int id = 1;
     public D(int i) { id = i; }
     public void f() { System.out.println("D.f() : " + id); }
}
class Run {
     public static void main(String args[]) {
          D d = new D(10);
          d.f();
     }
}
```

Question 23 : Ce programme compile-t-il ? Si oui, qu'affiche-t-il ?

```
class C {
  public void f(double d) {System.out.println("void C.f(double)");}
  public void f(String s) {System.out.println("void C.f(String)");}
}

class D extends C {
  public void f(double d) {System.out.println("void D.f(double)");}
}
class E extends C {
  public int f() {System.out.println("int E.f(void)");
  return 0;
}
class Run {
  public static void main(String args[]) {
     C c = new C() ;
     c.f(3.14);
     c.f(new String());
     D d = new D();
     d.f(3.14);
     d.f(new String());
     E e = new E();
     e.f();
     e.f(3.14);
  }
}
```

Question 25 : Ce programme compile-t-il ? Justifier et corriger si nécessaire.

```
class Run {
     public static void main(String args[]){
          String[10] ts1;
          String ts2[10][20];
          for(int i = 0; i < ts1.length(); i++) { ts1[i] = "" + i; }
     }
}
```

Question 26 : Que fait ce programme ? Justifier et corriger si nécessaire.

```
import java.util.*;
class C (
     public int key = 0;
     public C(int i) { key = i; }
}
class Run {
     public static void main(String args[]) (
          C x = new C(3);
          C[] Tab = { new C(0), new C(1), new C(2), x };
          int index = Arrays.binarySearch(Tab, x);
     }
}
```

Question 27 : Hormis les tableaux, donner les 3 containers majeurs, ainsi que leurs principales lasses dérivées.

Décrire rapidement leur caractéristique principale.

PARTIE 2 : Maintenant je suis fort en Java

Exercice 4 :

Supposons que nous ayons défini une exception appelée `NotANumberException()`. Supposons aussi que `NotAPositiveNumberException()` dérive de `NotANumberException()`. Le code suivant capture-t-il une exception `NotAPositiveNumberException` ?

```
catch(NotANumberException) {
 System.out.println("capture d'une exception NotANumberException\n");
}
```

Maintenant supposons que nous ayons la clause suivante et qu'une exception `NotAPositiveNumberException` soit lancée, que se passe-t-il ?

```
catch(NotANumberException) {
 System.out.println("capture d'une exception NotANumberException\n");
}
catch(NotAPositiveNumberException) {
 System.out.println("capture d'une exception NotAPositiveNumberException\n");
}
```

Exercice 5 :

Vous trouverez ci-dessous le code de la classe `Point` :

```
public class Point {
    private double x,y;

    public Point(double x, double y) {
      this.x=x;      this.y=y;
    }

    public double calculerDistance(Point p) {
      double dx=x - p.x;
      double dy=y - p.y;
      return Math.sqrt(dx*dx+dy*dy);
    }
}
```

NB : `Math.sqrt` retourne un double représentant la racine carrée du double passé en paramètre.

Question 1 : Redéfinissez la méthode `public boolean equals(Point p)` de la classe `Point`, pour qu'elle indique si 2 Points sont égaux.

Question 2 : Ecrivez la classe `Droite` qui permet de définir des droites du plan, selon les spécifications suivantes:

- attributs
  - deux `double a` et `b` (définissant l'équation de la droite $y = ax + b$),
- méthodes
  - un constructeur qui prend deux `double` en paramètres,
  - un deuxième constructeur qui prend 2 `Point` (ce constructeur devra lancer l'exception `PointsConfondusException` si les deux points passés en paramètres sont égaux. Le code de la classe `PointsConfondusException` vous est donnée ci-dessous :

```
public class PointsConfondusException extends Exception {
    Point A,B;
    public PointsConfondusException(Point p1,Point p2) {
      A=p1; B=p2;
    }
    public String toString() {
      return "les points "+A+"et"+B+"sont confondus";
    }
}
```

Question 3 : Ajoutez une méthode `public boolean estSur(Point p)` à la classe `Droite`, qui retourne `true` si le `Point p` trouve sur la droite.

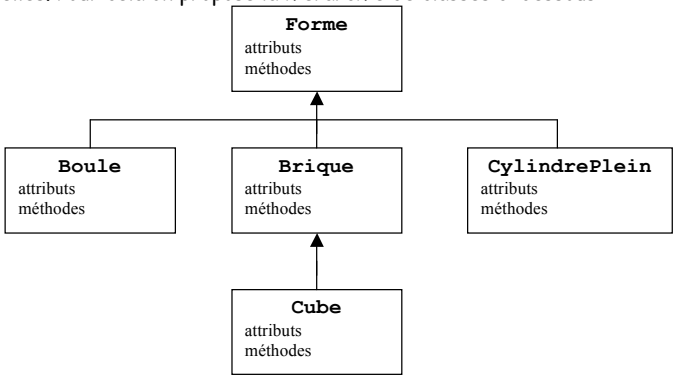
Question 4 : Ajoutez une méthode `public boolean paralleles(Droite d)` à la classe `Droite`, qui retourne `true` si la droite d est parallèle à la droite sur laquelle la méthode est invoquée.

Question 5 : Ajoutez une méthode `public Point intersection(Droite d)` à la classe `Droite`, qui calcule les coordonnées et retourne le point d'intersection des deux droites ou `null` si les droites sont parallèles.

Question 6 : Soit la classe `Triangle` dont les attributs sont trois `Point` `p1`, `p2` et `p3` représentant les sommets du triangle.
Ecrire le constructeur de la classe `Triangle` qui prend trois `Droite` en paramètre.

Exercice 6 :

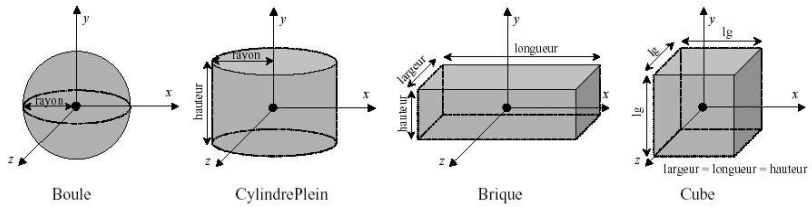
On souhaite disposer d'un ensemble de classes permettant de manipuler des formes tridimensionnelles. Pour cela on propose la hiérarchie de classes ci dessous



Les contraintes sur ces classes sont les suivantes :

- La classe `Forme` ne pourra pas être instanciée.
- Chaque forme possède un attribut de type `Point3D` (cf. fin de ce sujet) qui représente son centre de gravité et un attribut réel représentant sa densité.
- Un objet de type `Boule` est caractérisé par son centre de gravité, sa densité et son rayon.

- Un objet de type `CylindrePlein` est caractérisé par son centre de gravité, sa densité, une hauteur et un rayon.
- Un objet de type `Brique` est caractérisé par son centre de gravité, sa densité, une largeur, une longueur et une hauteur.
- Un objet de type `Cube` est une brique pour laquelle largeur = longueur = hauteur.



Boule CylindrePlein Brique Cube

Les formes disposent des méthodes `calculerSurface`, `calculerVolume` et `calculerPoids` calculant respectivement la surface, le volume et le poids (volume x densité) de la forme.
De plus, toute forme est capable de donner sa représentation sous la forme d'une chaîne de caractères contenant le nom de sa classe et la description textuelle de chacun de ses attributs.

Exemple : la chaîne de caractères produite pour un objet de classe Brique :

```
[Brique
 centre de gravité : [Point3D x :10.0 , y : 4.0, z : 3.0]
 densité : 1.2
 largeur : 10.5
 longueur : 14.3
 hauteur : 4.6
]
```

Question 1 : Ecrivez le code Java des classes `Forme` et `Brique`.
Rappel : surface d'une "brique" : 2x(largeur x longueur + largeur x hauteur + longueur x hauteur).

Question 2 : On suppose que la classe `Boule` dispose du constructeur suivant :
`public Boule(double r)` qui crée une boule de rayon r centrée en l'origine et de densité 1.
Etant données les déclarations et initialisations suivantes :

```
Boule b1 = new Boule(100.0) ; // crée une boule de rayon 100
             //de centre (0,0,0) et de densité 1.
Boule b2 = new Boule(100.0) ; // crée une seconde boule de rayon 100
              //de centre (0,0,0) et de densité 1.
```

Quelle est la valeur de l'expression booléenne `b1 == b2` ?

Que faudrait-il faire pour tester l'égalité de deux `Formes`, sachant que l'on considère que deux formes sont égales si elles sont de **même classe** et si elles représentent le même objet géométrique (par exemple deux sphères sont égales si elles ont même centre de gravité, même rayon et même densité) ? Répondez à la question sans écrire de code !

Question 3 : On souhaite maintenant que les formes soient mobiles. Ce comportement de mobilité n'étant pas propre aux formes nous décidons de créer une interface `Mobile` qui contient la méthode `bouger` prenant comme paramètres trois réels représentant les delta à ajouter aux composantes x, y et z. Faîtes-le nécessaire, et en bon fainéants n'en faîtes pas plus qu'il ne faut !

Question 4 : On ajoute à la classe `Forme` un attribut `listeDeFormes` qui contiendra la liste des références de toutes les formes instanciées
Précisez la déclaration de cet attribut. Que faut-il modifier dans le code de la question 1 pour que cet attribut contienne effectivement la liste des références à toutes les formes instanciées ?
Ecrivez le code de la méthode `poidsTotal` qui permet de calculer et d'afficher sur la sortie standard (`System.out`) le poids total des formes contenues dans la liste `listeDeFormes`.

Classe Point3D

Constructeur : création d'un Point3D par la donnée de ses coordonnées
`Point3D(double x, double y double z)`

Méthodes :

`void bouger(double(vx, double vy, double vz);`
translate le point d'un vecteur donné
`boolean equals(Point3D p);`
renvoie vrai si l'objet passé en paramètre est un Point3D identique à ce point
`double getX();`
renvoie la valeur de la coordonnée x du point
`double getY();`
renvoie la valeur de la coordonnée Y du point
`double getZ();`
renvoie la valeur de la coordonnée Z du point
`void setX(double v);`
fixe la valeur de la coordonnée x du point
`void setY(double v);`
fixe la valeur de la coordonnée y du point
`void setZ(double v);`
fixe la valeur de la coordonnée z du point
`void setXYZ(double v1, double v2, double v3);`
fixe la valeur des coordonnées x, y et z du point
`java.lang.String toString();`
retourne la repréentation textuelle du point sous la forme (si x=10.0, y=4.0 et z=3.0)
`[Point3D x :10.0 , y : 4.0, z : 3.0]`