Université Paris Diderot **Programmation Orientée Objet**
L2/L3 **Année 2014-2015**

TD n° 4

Héritage, composition, conception

Exercice 1 (Classification)
On prend les catégories suivantes : `Etudiant`, `Personne`, `EtudiantTravailleur`, `Enseignant`, `EtudiantSportif` et `Travailleur`.

1. Dessinez une arborescence cohérente pour ces classes en la justifiant, et en veillant à ce qu'elle soit implémentable en Java.
2. Où se situeront les champs suivants : `salaire`, `anneDEtude`, `nom`, `age` et `sportPratique` ?
3. Écrivez une méthode qui prend un tableau de personnes en argument et retourne la somme des salaires de ces personnes.

Exercice 2 Les exemples suivants sont-ils corrects ? Justifiez.

```
1  public final class A{
2    private int a;
3    public A() {this.a = 0;}
4  }
5  public class B extends A{
6    public int b;
7    public B(){super(); this.b = 0;}
8  }
```

```
1  public class Suite extends String{
2    String s;
3    public Suite(String w) {this.s = w;}
4  }
```

Exercice 3 (Constructeurs)
Les exemples suivants sont-ils corrects ? Justifiez.

1.
```
1  class A{
2    private int a;
3    public A() {System.out.println(a);}
4    public A(int a) {this.a = a; this();}
5  }
```

2.
```
1  class A{
2    private int a;
3    private int b;
4    public A() {System.out.println(a);}
5    public A(int a){ this.a = a;this.b = 0;  }
6    public A(int a, int b) { this(a); this.b = b; }
7  }
```

3.

```
1  class A{
2    public int a;
3  }
4  class B extends A{
5   public  int b;
6    B(int a, int b) {this.a = a; this.b = b;}
7  }
```

4.

```
1  class A{
2   int a;
3  }
4  class B extends A{
5    int b;
6    B(int a, int b) {this.a = a; this.b = b;}
7  }
```

5.

```
1  class A{
2    private int a;
3  }
4  class B extends A{
5   public  int b;
6    B(int a, int b) {this.a = a; this.b = b;}
7  }
```

6.

```
1  class A{
2    int a;
3    A(int a) {this.a = a;}
4  }
5  class B extends A{
6    int b;
7    B(int a,int b) {this.a = a; this.b = b;}
8  }
```

7.

```
1  class A{
2    private int a;
3    A() {this.a=0;}
4  }
5  class B extends A{
6    private int b;
7    B() {this.a=0; this.b=0;}
8  }
```

8.

```
1  class A{
2    private int a;
3    private String s;
4    public A(String s, int a) {this.a=a; this.s = s;}
5  }
6  class B extends A{
7    private int m;
8    public B(String s, int a, int m) {
9        super(s,a); this.m=m;}
10 }
```

9.

```
1  class A{
2    private int a;
3    private A(){this.a=0;}
4    public A(int a) {this.a=a;}
5  }
6  class B extends A{
7    private int b;
8    B() {super(); this.b=0;}
9  }
```

Exercice 4 (Liaison dynamique)
Qu'affiche le programme suivant ?

```
class A {
    public String f(B obj) { return ("A et B");}
    public String f(A obj) { return ("A et A");}
}
class B extends A {
    public String f(B obj) { return ("B et B");}
    public String f(A obj) { return ("B et A");}
}
class test {
    public static void main (String [] args){
    A a1 = new A();
    A a2 = new B();
    B b = new B();
    System.out.println(a1.f(a1));
    System.out.println(a1.f(a2));
    System.out.println(a2.f(a1));
    System.out.println(a2.f(a2));
    System.out.println(a2.f(b));
    System.out.println(b.f(a2));
    }
}
```

Exercice 5 (Redéfinition)
Les exemples suivants sont-ils corrects ? Justifiez.

1.

```
1  class A{
2    public void f(){System.out.println("Hello.");}
3  }
4  class B extends A{
5    private void f(){System.out.println("Hello world.");}
6  }
```

2.

```
1  class A{
2    public int f(int a) {return a++;}
3  }
4  class B extends A{
```

```
5       public boolean f(int a) {return a==0;}
6     }
```

3.

```
1     class A{
2       public int f(int a) {return a++;}
3     }
4     class B extends A{
5       public int f(int a, int b) {return a+b;}
6     }
7     class Test{
8       B obj = new B();
9       int x = obj.f(3);
10      int y = obj.f(3,3);
11    }
```

4.

```
1     class A{
2       public int f(int a) {return a++;}
3     }
4     class B extends A{
5       public int f(int a, int b) {return a+b;}
6     }
7     class Test{
8       A obj = new B();
9       int x = obj.f(3);
10      int y = obj.f(3,3);
11    }
```

5.

```
1     class A{
2       int a;
3       public String toString() {return new String("a= "+a);}
4     }
5     class B extends A{
6       int b;
7        public StringBuffer toString() {
8           return new StringBuffer("a= "+a +" b="+b);}
9     }
```

6.

```
1     class A{
2       int a;
3       public StringBuffer toString() {
4           return new StringBuffer("a= "+a);
5       }
6     }
7     class B extends A{
8       int b;
9        public StringBuffer toString() {
10          return new StringBuffer("a= "+a +" b="+b);}
11    }
```

Exercice 6 (Champs de classe)
Qu'affiche le programme suivant ?

```
1  class A{
2    int i;
3    int f() {return i;}
4    static String g() {return "A";}
5    String h() {return g();}
6  }
7  class B extends A{
8    int i=2;
9    int f() {return -i;}
10   static String g() {return "B";}
11   String h() {return g();}
12 }
13 class Test{
14   public static void main(String[] args) {
15     B b = new B();
16     System.out.println(b.i);
17     System.out.println(b.f());
18     System.out.println(b.g());
19     System.out.println(b.h());
20     A a = b;
21     System.out.println(a.i);
22     System.out.println(a.f());
23     System.out.println(a.g());
24     System.out.println(a.h());
25   }
26 }
```

Exercice 7 (Variance)

1. On considère le code suivant. Quelles sont les lignes qui vont poser problème, et à quel moment (compilation, exécution) ?

```
1  class A {
2  }
3
4  class B extends A {
5  }
6
7  class Main {
8      public static void main(String[] args) {
9          B b = new B();
10         A a = new A();
11         Object o = new Object();
12         ArrayList<A> as = new ArrayList<A>();
13
14         as.add(a);
15         as.add(b);
16         as.add(o);
17
18         Object o2 = as.get(0);
19         A a2 = as.get(0);
20         B b2 = as.get(0);
21     }
```

```
22 }
```

2. Même question :

```
1  class Main {
2      public static void main(String[] _) {
3          A a = new A();
4          ArrayList<B> bs = new ArrayList<B>();
5          ArrayList<A> as = (ArrayList<A>) bs;
6          as.add(a);
7      }
8  }
```

3. Même question :

```
1  class Main {
2      public static void main(String[] _) {
3          String[] t1 = new String[1];
4          Object[] t2 = t1;
5          t2[0] = new Integer(42);
6      }
7  }
```

Exercice 8 (Composition, substitution) On cherche à représenter le système de fichiers au sein d'une application Java. L'abstraction la plus simple est une classe `Fichier` définie avec trois champs `chemin`, `nom` et `extension` de type `String`, un constructeur prenant ces trois arguments, et des accesseurs. Cette classe possède également une méthode `deplacer(String)` (prenant le nouveau chemin en argument) qui renvoie un nouvel objet `Fichier` si le chemin est accessible, et `null` sinon.

On possède également une classe `Media`, indépendante de la classe `Fichier`, possédant des champs privés `auteur` et `titre` de type `String`.

1. Un objet `Fichier` reste valide si l'on remplace son nom ou son extension. Définissez une méthode `changeExtension` qui prend en argument un objet `Fichier` et un `String` et qui modifie l'objet `Fichier` pour lui donner la nouvelle extension (comme le ferait la commande Unix `mv`).
2. Il est tout à fait possible qu'un fichier ne possède pas d'extension (auquel cas il n'y a pas de point dans son nom). Dans ce cas (et dans ce cas seulement), le champ `extension` doit être laissé à `null`. Donnez le code du constructeur correspondant.
3. Notre application doit pouvoir manipuler différents types de médias (nommés `Image`, `Film`, etc). représentant des fichiers contenant les médias correspondants. Ces fichiers doivent cependant posséder la bonne extension. Par exemple, les images ne peuvent être contenues que dans des fichiers dont l'extension est `".jpg"` ou `".png"`. Comment faut-il les définir (de quelles classes peut-on hériter) ?
4. Notre application doit également gérer des fichiers particuliers, appartenant à certains utilisateurs. On représente ces fichiers par une classe `FichierPersonnel`, qui doit représenter un fichier tout en portant le nom du propriétaire. Comment la définir ? Le morceau de code suivant est-il correct ?

```java
public class FichierPersonnel extends Fichier {
    public FichierPersonnel(String nom, String chemin,
                            String extension,
                            String proprietaire) {
        this.nom = nom;
        this.chemin = chemin;
        this.extension = extension;
        this.proprietaire = proprietaire;
    }
    ...
}
```

5. Le *principe de substitution de Liskov* est un principe de bonne conception dans les langages orientés objet. Il peut être compris ainsi : *toute "bonne propriété" vraie pour les instances d'une classe A doit être vraie pour les instances de toute sous-classe de A*. Dans cet exercice, quelles "bonnes propriétés" a-t-on mises en avant pour concevoir les relations entre nos classes ?