Université Paris Diderot **Programmation Orientée Objet**
L2 **Année 2016-2017**

TD - Séance n°5

Héritage, interfaces et classes abstraites

1 Un paquet d'erreurs

Dans cette première partie nous allons explorer un certain nombre d'erreurs possibles dans les définitions de classe, de constructeurs, etc...

Exercice 1 *Une histoire de construction*

Les exemples suivants sont-ils corrects ? Justifiez.

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

2.
```
class A{
  private int a;
  private int b;
  public A() {System.out.println(a);}
  public A(int a){ this.a = a;this.b = 0;  }
  public A(int a, int b) { this(a); this.b = b; }
}
```

3.
```
class A{
  public int a;
}
class B extends A{
 public  int b;
  B(int a, int b) {this.a = a; this.b = b;}
}
```

4.
```
class A{
 int a;
}
class B extends A{
  int b;
  B(int a, int b) {this.a = a; this.b = b;}
}
```

5.

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

6.

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

7.

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

8.

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

9.

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

Exercice 2 *Redéfinition*

Les exemples suivants sont-ils corrects ? Justifiez.

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

2.
```
class A{
  public int f(int a) {return a++;}                         2
}
class B extends A{                                          4
  public boolean f(int a) {return a==0;}
}                                                           6
```

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

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

5.
```
class A{                                                    1
  int a;
  public String toString() {return new String("a= "+a);}   3
}
class B extends A{                                          5
  int b;
```

```
     public StringBuffer toString() {
        return new StringBuffer("a= "+a +" b="+b);}
  }
```

6.

```
  class A{
    int a;
    public StringBuffer toString() {
        return new StringBuffer("a= "+a);
    }
  }
  class B extends A{
    int b;
     public StringBuffer toString() {
        return new StringBuffer("a= "+a +" b="+b);}
  }
```

7.

```

public class Val2 {
        int val1;
        int val2;
        public Val2 decrese(int i){
                return new Val2();
        }
}
public class Val3 extends Val2 {
        int val3;
        public Val3 decrese(int i){
                return new Val3();
        }
}
```

Exercice 3 *Liaison dynamique*

Qu'affiche le programme suivant ? Attention aux pièges, prenez bien le temps de réfléchir à la différence entre le type d'une variable d'objet et la classe dont l'objet référencé est réellement l'instance...

```
  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();
```

```
12          A a2 = new B();
            B b = new B();
14          System.out.println(a1.f(a1));
            System.out.println(a1.f(a2));
16          System.out.println(a2.f(a1));
            System.out.println(a2.f(a2));
18          System.out.println(a2.f(b));
            System.out.println(b.f(a2));
20          }
    }
```

2 Interfaces et classes abstraites

Exercice 4 *Interfaces vs classes abstraites*

1. Peut-on instancier une interface ? une classe abstraite ?
2. Peut-on y mettre un constructeur ? un constructeur avec un corps ?
3. Peut-on écrire le code suivant : A a = **new** B() ;
   - si A est une classe abstraite, dérivée par la classe B ?
   - si A est une interface, implémentée par une classe B ?
4. Une interface/classe abstraite peut-elle contenir des méthodes abstraites ? non-abstraites ? statique et abstraite ?
5. Une interface/classe abstraite peut-elle contenir des attributs ? avec quels modificateurs ? doivent-il être instanciés ?
6. Une interface peut-elle hériter d'une autre interface ? d'une classe abstraite ?
7. Une classe abstraite peut-elle hériter d'une autre classe abstraite ? d'une interface ?

Exercice 5 *Instruments de musique*

Dans cet exercice, on va tenter modéliser une structure de classes pour des instruments de musique.

Il existe plusieurs façons de classifier les instruments. Une première consiste a différencier selon le procédé qui permet de produire le son. Certains sont dits mécaniques, dans le sens où le son provient d'une vibration mécanique d'une pièce ou d'une masse d'air (tous les instruments traditionnels, mais aussi la guitare électrique ou les pianos électriques type orgue Hammond, Rhodes, etc...) et d'autres, dits électroniques, dont le son est produit par un générateur électronique oscillant (les synthétiseurs). Ensuite le son peut être amplifié par une caisse de résonnance ou bien a l'aide d'un microphone. Une guitare électrique, par exemple, n'est pas un synthétiseur, le son provient bien de la vibration d'une corde, mais il est amplifié a l'aide de microphones qui transforment cette vibration en signal électrique.

Les instruments mécaniques sont séparés en trois grandes familles,

– Les Cordes, qui peuvent être pincées (guitare), frappées (piano) ou frottées (violon).
– Les Vents divisés en Bois (le son vient de la vibration de l'air sur une pièce mécanique), et Cuivres (le son vient de la vibration des lèvres à l'embouchure).
– Les Percussions (on frappe une peau ou une pièce de bois ou de métal).

1. Fournir une architecture de classes et/ou interfaces pour représenter ceci. Tous les types descendront du type Instrument. Tous les instruments ont en commun de pouvoir être joués. On munira donc Instrument d'une méthode abstraite **abstract public void** play(), qui devra être implémentée par chaque instrument. Ils ont aussi en commun d'avoir un nom : Instrument disposera donc d'un champ String name. Les instruments à cordes disposent d'un champ indiquant le nombre de cordes (comment peut-on le définir ?). Les instruments électriques disposent d'un champ indiquant le type de prise. Écrire les classes et/ou interfaces Instrument, Cordes, Amplifie. Donner la déclaration (l'en-tête) de la classe Orgue, de la classe Saxophone, et de la classe GuitareElectrique.
2. On aimerait bien aussi pouvoir dire qu'un piano à queue, un orgue d'église, un synthétiseur électronique, appartiennent tous a la famille des claviers. Comment faire cela ?