Université Paris Diderot **Programmation Orientée Objet**
L2, EIDD et M1 linguistiques **Année 2015-2016**

TD n° 5

Un gros paquet d'erreurs

Dans ce TD 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{
  private int a;
}
class B extends A{
 public  int b;
  B(int a, int b) {this.a = a; this.b = b;}
}
```

6.
```
class A{
  int a;
  A(int a) {this.a = a;}
}
class B extends A{
  int b;
  B(int a,int b) {this.a = a; this.b = b;}
}
```

7.
```
class A{
  private int a;
  A() {this.a=0;}
}
class B extends A{
  private int b;
  B() {this.a=0; this.b=0;}
}
```

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

9.
```
class A{
  private int a;
  private A(){this.a=0;}
  public A(int a) {this.a=a;}
}
class B extends A{
  private int b;
  B() {super(); this.b=0;}
}
```

Exercice 2 *Redéfinition*

Les exemples suivants sont-ils corrects ? Justifiez.

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

2.
```
class A{
  public int f(int a) {return a++;}
}
```

```
class B extends A{
  public boolean f(int a) {return a==0;}
}
```

3.

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

4.

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

5.

```
class A{
  int a;
  public String toString() {return new String("a= "+a);}
}
class B extends A{
  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 *Point final*

Les exemples suivants sont-ils corrects ? Justifiez.

1.
```
public final class A{
  private int a;
  public A() {this.a = 0;}
}
public class B extends A{
  public int b;
  public B(){super(); this.b = 0;}
}
```

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

Exercice 4 *Liaison dynamique*

Qu'affiche le programme suivant ? Attention aux pièges, prenez bien le temps de réfléchir à la différence entre le type des objets la classe dont ils sont réellement l'instance...

```
1  class A {
2      public String f(B obj) { return ("A et B");}
3      public String f(A obj) { return ("A et A");}
4  }
5  class B extends A {
6      public String f(B obj) { return ("B et B");}
7      public String f(A obj) { return ("B et A");}
8  }
9  class test {
10     public static void main (String [] args){
11     A a1 = new A();
12     A a2 = new B();
13     B b = new B();
14     System.out.println(a1.f(a1));
15     System.out.println(a1.f(a2));
16     System.out.println(a2.f(a1));
17     System.out.println(a2.f(a2));
18     System.out.println(a2.f(b));
19     System.out.println(b.f(a2));
20     }
21 }
```

Exercice 5 *En vue du TP*

S'il reste du temps, nous allons commencer à préparer le terrain pour le TP. On ne s'intéressera ici qu'à la conception, et donc en particulier on ne cherchera pas à écrire le code en soi, mais uniquement à sa structuration.

Nous allons repartir de la modélisation d'une société médiévale afin de créer un jeu simple la modélisant. L'idée est de faire évoluer un royaume, constitué de villages, dans le but de le faire prospérer au maximum. Comme dans l'épisode précédent, un villages est composé de roturiers, sur lesquels un impôt (dont le montant correspond à la moitié de leur gain) est prélevable.

Le joueur sera en charge du royaume, est aura à chaque tour la possibilité d'effectuer des actions en fonction de l'argent qu'il a de disponible. Dans un premier temps, on considère qu'il peut :

- « acheter »un nouveau paysan, avec un certain coût ;
- construire un nouveau village, avec un certain coût aussi ;
- prélever l'impôt dans le royaume, ce qui correspond à le prélever dans chacun des villages ;
- finir le tour, qui a pour effet de faire passer une année, vieillir les personnages et effectuer une production.

1. De quelles classes va-t-on avoir besoin pour modéliser les différents éléments d'un royaume ? Quelles sont les relations les liant entre eux ?
2. Où va-t-on placer les différents méthodes (`production`,`impot`, etc...) pour faire évoluer le royaume ?
3. Au cours du jeu, on va donc avoir besoin de créer de nouveaux objets (personnes et villages) en cours d'exécution. Comment appelle-t-on une classe remplissant cette fonction ? Écrivez-en succintement le code.

Notre modélisation du royaume étant au point, nous allons donc chercher à concevoir un jeu. On veut donc que le jeu porte sur un seul royaume, donc l'évolution est dirigée par les choix du joueurs. De plus, on se restreindra pour le moment aux seules actions mentionnées précédemment.

1. Comment pouvez caractériser les actions ? En déduire la définition d'une interface `Action`, ainsi que l'implémentation des différents actions.
2. Comment allez-vous modéliser le joueur et le jeu ?
3. On aimerait, à terme, pouvoir remplacer le joueur par un programme implémentant une stratégie (et donc faisant les choix). Cela est-il compatible avec votre modélisation ?
4. Dessiner le diagramme de classes comprenant les différentes classes/interfaces/fabriques nécessaires à votre jeu.

Bonus Comme vous pouvez le constater, notre jeu est pour l'instant assez sommaire. Proposer des extensions possibles (ajout de personnages, d'actions, etc...) qui pourrait vous sembler intéressantes. Sont-elles compatibles avec la modélisation actuelle ?