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

TD n° 3

Héritage

Ce TD vise à manipuler la notion d'héritage, en définissant des relations hiérarchiques d'appartenance.

Exercice 1 *Arborescence de Classe*

On définit une classe Personne de la manière suivante :

```
1  public class Personne {
2
3      private String nom;
4      private int ressources;
5      private int argent;
6
7      public Personne(String nom, int ressources, int argent){
8          this.ressources = ressources;
9          this.nom = nom;
10         this.argent = argent;
11     }
12
13     public String toString(){
14         return "Je m'appelle :" + this.nom + ". J'ai " + this.
               ressources + " ressources, et " + this.argent " unités
               monétaires ".;
15     }
16 }
```

1. Ajoutez à la classe `Personne` les méthodes publiques `augmenteRessources`, `diminueRessources`, `augmenteArgent`,`diminueArgent`.
2. Spécifiez la classe `TiersEtat`, héritant de la classe `Personne`. Plus précisément, on veut :
   - un attribut `capacitéDeProductionParAn`, et une méthode `production()` qui modélise l'augmentation de la quantité de ressources au bout d'un an ;
   - un constructeur (dans sa définition, servez-vous du constructeur hérité de la classe Personne) ;
   - une méthode `toString()` qui donne les informations suivantes : nom, ressources, argent, capacité de production. On utilisera pour ce faire la méthode `toString` héritée de la classe Personne.
3. Considérez la classe Test suivante. Peut-elle être compilée sans erreur ? Exécutée sans erreur ?

```
1  public class Test {
2    public static void main(String[] args){
3      Personne dupont = new Personne("Dupont",0,10);
4      int ressources = ((TiersEtat)(dupont)).capacité
           DeProductionParAn;
5    }
6  }
```

4. Parmis le TiersEtat, on peut distinguer les paysans, qui ne disposent pas d'argent (initialement). Spécifiez une classe Paysan, qui hérite de la classe TiersEtat, et écrire le constructeur associé.

Exercice 2 *Méthodes discriminant selon la (sous)-classe de leur argument*

1. Spécifiez la classe Noblesse, héritant de la classe Personne. Un noble dispose d'une liste de roturiers qui lui sont asservis.
2. Le noble peut soumettre à l'impôt ses roturiers (c'est-à-dire leur confisquer des ressources ou de l'argent). Écrivez une méthode `impot()`
3. En fait, le noble prélève un impôt différent sur les paysans : il confisque la moitié de leur ressources, contre seulement un quart pour les autres roturiers. Modifiez la méthode `impot()` en conséquence.
4. Spécifiez la classe Roi, héritant de la classe Noblesse. Un roi a également des vassaux (qui sont nobles), qu'il peut soumettre à l'impôt (mais à un taux beaucoup plus bas...). Écrivez la méthode `impot()` associée.

Exercice 3 *Interface*

On modélise en fait ici une structure sociale féodale, semblable à la situation en France au moyen-âge. Pour compléter l'analogie, on va modéliser l'existence du clergé. Le clergé a deux techniques pour produire des ressources : l'agriculture et la prière. La dernière dépend de l'efficacité (ou de la bonne volonté) du dieu considéré.

1. On modélise les dieux par une interface. Écrivez l'interface `Dieu` correspondante (qui a une méthode `int exauce(Personne p)`).
2. Écrire la classe `Clergé`, héritant de la classe `Personne` : chaque prêtre peut vénérer plusieurs dieux.
3. Écrire une classe réalisant la classe `Dieu`, telle que la quantité de ressources obtenue par la prière est constante.
4. Écrire une autre classe réalisant la classe `Dieu`, qui modélise une réponse aléatoire à la prière, une autre qui modélise une augmentation du rendement lorsque le nombre de prières déjà effectuées (par la même personne) augmente...

Exercice 4 *Modélisation globale*

On souhaite en fait créer une classe Société, qui :
— a comme attribut une liste de personne,
— a une méthode `anniversaire()`, lors de laquelle des ressources sont produites, de l'argent est échangé (et éventuellement crée), des ressources sont consommées...,
— gère la mortalité dûe au fait de ne plus avoir de ressources.

1. Proposez une modélisation, et sa spécification (en uml), à partir des éléments donnés dans les exercices précédents et en mettant l'accent sur les relations entre les classes.