Université Paris Diderot **Programmation Orientée Objet**
L2, EIDD et M1 linguistiques **Année 2016-2017**

TD n° 4

1 Héritage

Exercice 1 On définit les classes A,B,C de la manière suivante :

```
1  public class A{
2      public void g(){
3        System.out.println(0);
4      }
5  }
6
7  public class B extends A{
8      public void g(){
9          System.out.println(1);
10     }
11 }
12
13 public class C extends A{}
```

et on écrit un programme de test qui contient le code suivant :

```
1   public static void main(String[] args){
2          A[] tab = new A[3];
3          tab[0] = new A();
4          tab[1] = (A)new B();
5          tab[2] = (A)new C();
6          for(int i=0; i<3;i++){tab[i].g();}
7      }
```

Est-il possible de supprimer le cast explicite ? Que se passe-t-il à l'exécution ?

2 Modélisation

Exercice 2 On définit une classe Personne de la manière suivante :

```
1  public class Personne{
2      private String name;
3      public Personne(String name){
4          this.name = name;
5      }
6
7      public String toString(){
8          return "Je m'appelle " + this.name+ ". ";
9      }
10 }
```

On veut ici illustrer la notion d'héritage en modélisant la structure de la société française au moyen-âge. Cette structure reposait sur une division en trois ordre : la noblesse, le clergé, et le tiers-état. Un membre de la noblesse est un noble, un membre du clergé est un prêtre, et un membre du tiers-état est un roturier.

1. Définir des classes Noble, Pretre, Roturier, qui hérite de Personne, de telle sorte que l'exécution du code suivant produise :
   Je m'appelle Louis. Je suis un noble.

```
1 Noble n = new Noble("Louis");
2 System.out.println(n);
```

2. On ajoute maintenant à la classe Roturier :

```
1   private int argent = 0;
2   public void addArgent(int i) {
3         this.argent = this.argent + i;
4   }
```

   Le code suivant compile-t-il ? S'exécute-t-il sans erreur ?

```
1 Personne n = new Noble("Louis");
2 ((Roturier)n).addArgent(1);
```

3. Le tiers-état est un ordre très hétérogène socialement, qui comprend à la fois des paysans, des artisans et des bourgeois. Quelles classes doit-on créer pour modéliser cela ? De quelle classe doivent-elle hériter ?
4. On considère maintenant une classe Société, qui a comme attribut un tableau de personne.
   — Écrire un constructeur de Société, qui prend en argument un entier n, et qui crée une société de n personnes, de rôle social choisi aléatoirement.
   — Écrire une méthode : public int nbPaysan(), qui renvoie le nombre de paysan dans la société.
   — Écrire une méthode : public int argentTotal(), qui renvoie la somme de l'argent que chaque roturier membre de la société possède.

Exercice 3 On crée maintenant une classe Percepteur, qui correspond à un collecteur d'impôt. Il a comme attribut une société (l'ensemble des gens qu'il peut taxer), et un attribut argent qui correspond à l'argent obtenu. Néanmoins, seuls les roturiers doivent payer des impôts (attention, historiquement c'est faux).

1. Écrire un constructeur Percepteur(int n), qui crée un percepteur avec une quantité d'argent initiale égale à 0, et qui a comme attribut une société de taille n initialisée aléatoirement.
2. Ajouter à la classe Percepteur une méthode :
   public void impot()
   , telle que : pour chaque personne du tableau, l'argent du Percepteur augmente de 1 si cette personne est un roturier, et n'augmente pas sinon.
3. Considérez les casts (implicite et explicite) que vous avez fait dans les questions précédentes. Quand peut-on parler de "downcast" ? de "upcast" ?
4. Les percepteurs ont une certaine autonomie dans la collecte des impôts. Écrire une classe PercepteurProportionnel, qui hérite de Percepteur, pour que l'imposition soit proportionnelle à l'argent que possède la personne.

5. Écrire une classe PercepteurInégalitaire, tel que le taux d'imposition dépende de sa position sociale : paysan, artisan ou bourgeois. De quelle classe doit hériter PercepteurInegalitaire ?

Exercice 4 Inspirez vous des classes définies précédemment pour avoir un modèle de société, tel que, à chaque tour :

— Les paysans et les artisans produisent des ressources
— Les percepteurs taxent ces ressources
— Les percepteurs rendent une partie de leur gain aux nobles (la gabelle), au clergé (la dîme).

3 Questions de cours

Exercice 5 Variables et méthodes statiques.
On définit une classe A par le code suivant.

```
1 public class A{
2     public static int a = 3;
3     public int b;
4
5     public A (int c){
6         this.b = c;
7     }
8 }
```

1. On définit une méthode statique h dans la classe A. Quels attributs peut-elle utiliser ? Même question pour une méthode non statique.
2. On définit dans la classe A une méthode g() par :

```
1 public void g(){
2         a = a+1;
3         b = b+1;
4     }
```

On utilise la classe A dans une classe Test auxiliaire, comme ci-dessous. Qu'obtient on à l'exécution ?

```
1  public class Test{
2      public static void main(String[] args){
3          A u = new A(0);
4          A v = new A(0);
5          u.g();
6          v.g();
7          System.out.println("u: a ="+ u.a +" ; b =" + u.b);
8          System.out.println("v: a ="+ v.a +" ; b =" + v.b);
9      }
10 }
```

3. Ajoutez à la classe A un attribut statique correspondant à un tableau d'entiers de longueur 10, initialisé de telle sorte que toutes les cases contiennent 2.

Exercice 6 Passage d'argument
Qu'obtient-on à l'exécution ...

1. ... du code suivant ?

```
1  public static int g(int i){
2          i = i+1;
3          return i;
4      }
5  int i = 0;
6  g(i);
7  System.out.println(i);
```

2. On suppose maintenant avoir la classe C suivante (elle encapsule un entier).

```
1  public class C{
2      private int a;
3      public C(int a){
4          this.a =a ;
5      }
6      public String toString(){
7          return Integer.toString(a);
8      }
9      public void setNumber(int j){
10         this.a = j;
11     }
12 }
```

   Qu'obtient on ...
   — ... si on exécute le code suivant ?

```
1  public static C h(C k){
2          k.setNumber(5);
3          return k;
4      }
5  C k = new C(0);
6  h(k);
7  System.out.println(k);
```

   — ... si on exécute le code suivant ?

```
1  public static C h(C k){
2          k= new C(5);
3          return k;
4      }
5  C k = new C(0);
6  h(k);
7  System.out.println(k);
```

3. Expliquez en quoi les exemples précédent illustre le passage par valeur utilisé par java.