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

TD n° 2

Domaines d'accès, associations de classes

NB : dans plusieurs questions, on parle de "spécifier" des méthodes au lieu de les écrire. Il s'agit alors de décrire leur comportement avec précision, afin de les programmer plus tard en TP.

Exercice 1 *Classes et accessibilité.*

On considère les deux classes suivantes :

```
1  public class Personne {
2
3      private String nom;
4      private String prenom;
5      public int age;
6
7      public Personne(String nom,
           String prenom, int age
       ){
8          this.nom = nom;
9          this.prenom = prenom;
10         this.age = age;
11     }
12
13     public void setPrenom(
           String p){
14         this.prenom = p;
15     }
16
17     public void anniversaire(){
18         this.age ++;
19     }
20
21     public String toString(){
22         return "Je m'appelle :
               " + this.prenom
23             + " " + this.nom +
               ". J'ai " +
               this.age + "
               ans.";
24     }
25
26 }
27
28 public class Test {
29
30     public static void main(
           String[] args){
31         Personne tony = new
               Personne("Parker","
               Tony",33);
32         System.out.println(tony
               );
33         Personne mickael = tony
               ;
34         mickael.setPrenom("
               Mickael");
35         System.out.println(
               mickael);
36         System.out.println(tony
               );
37     }
38
39 }
```

1. Peut-on placer ces deux classes publiques dans un seul fichier ?
2. Qu'obtient-on dans le terminal à l'exécution de `Test.class` ?
3. Peut-on exécuter les lignes suivantes dans le `main` pour changer le nom d'une Personne ?

```
1 mickael.nom = "Gelabale";
2 System.out.println(mickael);
```

Si non, proposez une façon de faire sans modifier les champs de la classe. Étant donné la méthode anniversaire, est-il utile que le champ age soit public ?

4. Si le main avait été écrit dans la classe `Personne` et non dans la classe `Test` aurait-on eu le droit d'écrire mickael.nom = "Gelabale"; ?

Exercice 2 *Petites manipulations*

1. Modifiez la classe `Personne` de l'exercice précédent, en ajoutant un champ représentant la quantité de monnaie que la personne possède.
2. Les deux méthodes suivantes (redondantes) doivent permettre à deux personnes de se transmettre une certaine somme. La méthode retourne un booléen. Celui-ci vaut **true** si le payement s'est bien déroulé.

```
1 public static boolean donne (Personne p1,
2     Personne p2, int montant) { ... }
3 public boolean donne(Personne p,  int montant){ ... }
```

   Écrivez ces méthodes. Donnez un exemple d'appel pour chacune.
3. Laquelle de ces deux méthodes vous semble la plus judicieuse ?
4. En commentaire, complétez la spécification en précisant qui paye à qui.

Exercice 3 *Encapsulation*
Dans cet exercice, nous allons définir un compte en banque. Un compte en banque se caractérise par un solde et par un titulaire (en l'occurrence une personne).

1. Écrivez les déclarations (complètes) des attributs de la classe Compte.
2. Spécifiez les méthodes suivantes (signature + description de leurs effets, notamment sur les attributs) :
   — getSolde qui renvoie le montant présent sur un compte.
   — credite qui crédite le compte d'un certain montant.
   — debite qui débite depuis le compte un certain montant.
   — une méthode qui permette au titulaire de retirer une certaine somme pour la mettre dans sa poche. (Pas de découvert)
   — une méthode qui permette au titulaire d'effectuer un dépôt, à partir de l'argent qu'il a en poche.
3. On se propose d'attribuer un numéro unique à chaque compte. Ainsi, le premier compte instancié aura pour numéro 0, le suivant 1 et ainsi de suite.
   Écrivez, dans la classe Compte, les déclarations des attributs suivants :
   — nbComptes, qui sert à se rappeler le nombre de comptes créés jusque là. Quelles méthodes devra/devront le modifier et comment ?
   — numero, qui stocke pour chaque compte son numéro unique. Comment et quand est-elle initialisée ?

Exercice 4 *Liens croisés*
Dans notre modélisation, un compte est lié à son titulaire, mais une personne ne l'est pas au compte qu'elle possède. Nous proposons ici une modification simple de ce modèle.

1. Modifiez la classe Personne, en ajoutant un champ de type Compte[] qui contiendra l'ensemble des comptes associés à une personne.
2. Modifiez le constructeur de Personne, en ajoutant un paramètre **int** n. Le constructeur se chargera de fabriquer n comptes différents de solde nul pour cette personne.
3. Écrivez dans votre main quelques manipulations de crédit sur ces différents comptes.
4. Quand on cherche à retirer une somme importante, il se peut que les fonds d'un seul compte ne suffisent pas. On peut alors vider chacun de nos comptes jusqu'à avoir atteint cette somme. Il se peut également que la somme de tous les fonds ne suffise pas. Écrivez la méthode retrait correspondante.

Exercice 5 *Mot de passe*
On souhaite écrire une classe qui corresponde à une entrée de base de données de clients. Une telle entrée est définie par un client (une personne), un login et un mot de passe. Supposons que cette classe aura pour nom Entree.

1. En utilisant la classe Personne définie précédemment, définir la classe Entree et le constructeur adapté.
2. Discuter l'accessibilité des différents champs. Lesquels peuvent-être publics ?
3. Écrire une méthode autorise qui prend en argument une chaîne de caractères et renvoies un booléen. Elle renvoie **true** si la chaîne fournie correspond au mot de passe. Afin de comparer deux chaînes de caractères, il faut utiliser la méthode equals :

```
1 String s1 = "haha";
2 String s2 = "haha";
3 String s3 = "hehe";
4 System.out.println(s1.equals(s2)); \\ va afficher true
5 System.out.println(s1.equals(s3)); \\ va afficher false
```

4. Modifier la classe Entree pour afin de stocker le nombre de verifications de mot de passe ratées.
5. Écrire une méthode changerMdp qui prend en argument deux chaînes de caractères. Si la première correspond au mot de passe, elle remplace le mot de passe actuel par la deuxième.
6. Discuter de l'accessibilité des deux méthodes précédentes.
7. Quels champs de cette classe peuvent être **final** ?

Exercice 6 *Un peu de modélisation*

On veut modéliser par un programme Java le fonctionnement d'un parking automobile. Attention : employez judicieusement les mots-clefs **private**, **public** et **static**.

1. Écrivez la classe Voiture vide et la classe Parking dont vous définirez les attributs.

   Un parking se caractérise notamment par :

— son nombre total de places ;
— son nombre de places libres ;
— la liste des voitures garées dans le parking ;
— son pourcentage de remplissage.

2. Écrire les méthodes **boolean** entrerParking(Parking p) et **void** sortirParking(Parking p) qui permettent à une voiture d'entrer (s'il reste des places) et de sortir d'un parking. Faites en sorte qu'elle ne puisse pas être dans plus d'un parking à la fois.
3. Améliorer la classe Parking afin que les voitures puissent demander une place précise. La classe Parking devra vérifier que cette place est libre. (Indice : pensez à utiliser un tableau de Voiture !)
4. Ajouter une méthode **int** premierePlaceLibre() dans Parking qui fournit le numéro de la première place libre.
5. Ajouter une méthode **int** voiturePerdue(Voiture v) pour retrouver une voiture dans le parking.
6. (**Bonus.**) En supposant qu'il n'y ait qu'un parking, pourrait-on se passer de la classe Parking (en mettant tout dans Voiture) ? [1]
7. (**Bonus.**) Comment pourriez-vous modifier votre code (et au besoin vos structures de données) pour optimiser :
   — la méthode de recherche d'une place libre ?
   — la méthode de recherche d'une voiture perdue ?

1. Indice : utilisez un champ **static**