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

TP n° 4

Héritage et interfaces 2

Exercice 1 Implémenter ce qu'on à fait en TD. Pour l'implémentation de la méthode `attaque(Personne p)` on reprendra ce qu'on a fait dans la question 1 de l'exercice 3. Tester les différentes méthodes.

Exercice 2 On va ici modéliser les gestions du salaire et des primes des employés d'une entreprise. On va pour cela utiliser la notion de classe abstraite. Une classe abstraite est une classe ne contenant aucune instance et que l'on déclare de la façon suivante : **`abstract class`** `MaClasse`. Bien qu'il n'y ait aucune instance dans une classe abstraite, elle peut contenir des attributs ainsi qu'un constructeur avec un corps. Une classe abstraite peut également contenir des méthodes avec un corps et des méthodes sans corps (dites *abstraites*) que l'on déclare de la façon suivante : **`abstract`** `typeDeRetour maMethode(Parametres)`.

1. Créer une classe abstraite `Employe` contenant les attributs privés `String nom` et `String prenom`. Munir la classe d'un constructeur passant les attributs en argument ainsi que d'une méthode abstraite **`double`** `salaire()`.
2. Créer les classes filles de `Employe` suivantes : `Vendeur`, `Representant`, `Technicien` et `Manutentionnaire`. Les vendeurs et représentants on un attribut **`int`** `chiffreDAffaire`. Les techniciens ont un attribut **`int`** `unitesProduites`. Les manutentionnaires ont un attribut **`int`** `nombreDHeure`. Créer des constructeurs dans chacune de ces classes (les constructeurs passant les attributs en argument).
3. Redéfinir la méthode `salaire()` dans chacune des classes filles sachant que :
   (a) le salaire d'un vendeur est égal à 20 pour cent de son chiffre d'affaire plus 400 euros,
   (b) le salaire d'un représentant est égal à 20 pour cent de son chiffre d'affaire plus 800 euros,
   (c) le salaire d'un technicien est égal à 5 fois le nombre d'unités qu'il produit,
   (d) un manutentionnaire gagne 12 euros par heure de travail.
4. Parmi les techniciens et les manutentionnaires, certains manipulent des produits dangereux. Créer deux nouvelles classes (que vous devez insérer correctement dans la hiérarchie de classes) correspondant à ces employés. Chacune de ces deux classes possède un attribut **`int`** `prime`. Les constructeurs de ces deux classes ont la même signature que celui de leur classe mère.
5. Les primes sont calculées de la façon suivante :

(a) la prime d'un technicien est égale au nombre d'unités qu'il produit,

(b) la prime d'un manutentionnaire est de deux euros par heure de travail.

On utilisera des méthodes **int** `prime()` permettant d'initialiser cet attribut.

6. Créer une classe `Entreprise` ayant un attribut `ArrayList<Employe> staff`. Cette classe contient également les méthodes suivantes :

   (a) **void** `ajouterEmploye(Employe e)` qui ajoute `e` dans l'entreprise,

   (b) **void** `affiche()` qui donne un description (i.e quel type de métier et quelles sont les valeurs des attributs) de chacun des employés de l'entreprise,

   (c) **double** `salaireMoyen()` qui retourne le salaire moyen des employés de l'entreprise.

   Afin d'alléger le code de la méthode d'affichage, vous êtes encouragés à écrire des méthodes `toString()` pour chacune des sous-classes de `Employe`. Plutôt que de définir directement cette méthode dans chacune des classes, penser à utiliser l'héritage.