TP 5 – Héritage et polymorphisme

1 L'héritage appliqué aux employés d'une entreprise, polymorphisme

Vous allez programmer le calcul des salaires hebdomadaires des employés d'une entreprise. Cette entreprise comporte plusieurs types d'employés :

- Des employés qui sont payés suivant le nombre d'heures qu'ils ont travaillé dans la semaine. Ils sont payés à un certain tarif horaire et leurs heures supplémentaires (au-delà de 39 heures) sont payées 30 % de plus que les heures normales.
- D'autres employés, payés de la même façon, mais leurs heures suplémentaires sont payées 50 % de plus que les heures normales.
- Les commerciaux sont payés avec une somme fixe à laquelle on ajoute 1 % du chiffre d'affaires qu'ils ont fait dans la semaine.

Modélisez cette situation à l'aide de classes.

- Vous donnerez un nom à chacun des employés. On ne pourra modifier le nom d'un employé.
- Vous commencerez par écrire une classe `Employe` dont hériteront les autres classes.
- Le temps est compté pour faire ce TP. Pour ne pas avoir trop de modificateurs vous simplifierez en ne donnant qu'un seul modificateur `setInfosSalaire` pour entrer ou modifier les informations brutes nécessaires au calcul des salaires (nombre d'heures de travail, chiffre d'affaire,...). N'essayez pas de faire du polymorphisme avec cette méthode.
- Les classes comporteront au moins 2 constructeurs : un qui ne prend en paramètre que le nom de l'employé et l'autre qui prend en paramètres le nom et toutes les informations pour le calcul du salaire.
- Le calcul des salaires se fera dans la méthode `getSalaire()` qui sera utilisée pour faire du polymorphisme.

Un conseil : écrivez votre hiérarchie d'héritage sur papier avant de commencer à coder.

Une classe `Paie` comportera une unique méthode `main()` qui entrera les informations sur des employés des différents types (au moins 3 commerciaux). Les employés seront enregistrés dans un tableau `employes`. Au moins un des employés sera créé avec le constructeur qui n'a que le nom en paramètre, et vous entrerez ensuite les informations pour son salaire avec la méthode `setInfosSalaire`. Pour au moins un autre employé, vous utiliserez le constructeur pour entrer les informations sur le salaire. La méthode `main` affichera le salaire hebdomadaire de chacun des employés dans une boucle "`for`" qui parcourra le tableau des employés. Vous utiliserez le polymorphisme avec un accesseur pour le salaire. L'affichage aura exactement la forme : "Dupond gagne 2500 euros". Vérifiez les calculs des salaires !

2 L'héritage chez les animaux (super !)

Des animaux, des mamifères et des poissons. Des chiens et des hommes...

Modélisez avec des classes. Ajoutez à chaque classe fille au moins une variable et une méthode nouvelles. Arrangez-vous pour que les classes filles puissent manipuler directement les nouvelles variables.

Voici ci-dessous une classe de test et l'affichage que vos classes devront fournir. La méthode `getType` de vos classes devra fournir une description de l'instance, en incluant les descriptions de toutes les classes mères. Pour cela, il vous est imposé d'utiliser le mot-clé `super`.

La classe de test :

```
public class TestAnimal {
  public static void main(String[] args) {
    Animal[] animaux = new Animal[5];
    animaux[0] = new Animal("Truc");
    animaux[1] = new Animal();
    animaux[2] = new Chien("Médor");
```

```
    animaux[3] = new Homme();
    animaux[4] = new Homme("Robert");
    for (int i = 0; i < animaux.length; i++) {
      System.out.println(animaux[i].getType());
    }
  }
}
```

L'affichage :

```
Je suis un animal de nom Truc.
Je suis un animal.
Je suis un animal de nom Médor. Je suis un mamifère. Je suis un chien.
Je suis un animal. Je suis un mamifère. Je suis un homme.
Je suis un animal de nom Robert. Je suis un mamifère. Je suis un homme.
```