Programmation en Java

Applications 03 : Héritage - Polymorphisme

Pour mener à bien cet atelier, chaque étudiant devra créer un projet par exercice en suivant la notation :

Appli01_ExoX_Prenom_Nom ou *Appli01_ExoX_Groupe_NumeroDuGroupe*

Le *X* doit être remplacé le numéro correspondant de l'exercice. *Prenom* et *Nom* doit être remplacé par votre prenom et nom sans les accents. *NumeroDuGroupe* doit être remplacé par le numéro du Groupe.

Exercice 1 :

On dispose de la classe suivante

```
class Point
{
  public void initialise (int x, int y) { this.x = x ; this.y = y ; }
  public void deplace (int dx, int dy)  { x += dx ; y += dy ; }
  public int getX() { return x ; }
  public int getY() { return y ; }
  private int x, y ;
}
```

Réaliser une classe PointA, dérivée de Point disposant d'une méthode affiche affichant les coordonnées d'un point. Ecrire un petit programme utilisant les deux classes Point et PointA.

Que se passerait-il si la classe Point ne disposait pas des méthodes getX et getY ?

Exercice 2 :

On dispose de la classe suivante :

```
class Point
{
  public Point (int x, int y)  { this.x = x ; this.y = y ; }
  public void affiche() {
    System.out.println ("Coordonnees : " + x + " " + y) ;
  }
  private int x, y ;
}
```

Réaliser une classe PointNom, dérivée de Point permettant de manipuler des points définis par leurs coordonnées et un nom (caractère). On y prévoira les méthodes suivantes :

- constructeur pour définir les coordonnées et le nom d'un objet de type PointNom,
- affiche pour afficher les coordonnées et le nom d'un objet de type PointNom.

Exercice 3 :

- Définir une classe **Voiture** avec les attributs suivants : *Id,* Marque*, Vitesse, Puissance*.
- Définir un **constructeur** permettant d'initialiser les attributs d'un objet voiture par des valeurs passées en paramètre. Sachant que *Id* doit être auto-incrément.
- Définir les accesseurs aux différents attributs de la classe.
- Définir la méthode **toString ( )** permettant d'afficher les informations d'une voiture.
- Écrire un programme testant la classe **Voiture**.

Exercice 4 :

Ecrire les classes nécessaires au fonctionnement du programme suivant (en ne fournissant que les méthodes nécessaires à ce fonctionnement). N'oubliez pas de commenter votre code :

```
public class TestEcole {
      public static void main(String[] argv) {
            Personne[] personnes = new Personne[3];
            personnes[0] = new Enseignant("Paul");
            personnes[1] = new Personnel("Jean");
            personnes[2] = new Etudiant("Adrien");

            for (int i = 0; i < personnes.length; i++)
                  personnes[i].affiche();
            }

      }
}
```

Exercice 5 :

1. Écrivez une classe `Batiment` avec deux variables `adresse` et `surfaceHabitable` (un entier) et son constructeur `Batiment(String adresse, double surface)`. Implémentez la méthode `String toString()`.
2. Écrivez une classe `Maison` héritant de `Batiment` avec les variables `nbPieces` et `surfaceJardin`. Écrivez le constructeur `Maison(String adresse, int surfaceH, int surfaceJ, int nbPieces)` en utilisant un appel à `super(...)`. Écrivez aussi la méthode `String toString()`.
3. Écrivez une classe `Immeuble` héritant de `Batiment` avec la variable `nbAppart`. Écrivez le constructeur correspondant et la méthode `String toString()`.
4. Écrivez une méthode `main` dans une classe `TestBatiment`. Ce programme doit instancier un bâtiment, une maison, un immeuble et les afficher.

   Ensuite, créez un tableau de 10 bâtiments. Est-ce que les bâtiments sont instanciés ?

   Remplacez 2 éléments du tableau par la maison et l'immeuble précédemment créés. Affichez tout le tableau.