VIII. L'héritage

- Principe : récupérer du code existant pour le modifier en partie avant de le réutiliser ;
- Pourquoi ? Pour raccourcir les temps d'écriture et de mise-au-point d'une application ;

- L'héritage permet de définir une classe dérivée à partir d'une classe existante ;
- La classe dérivée hérite des champs et méthodes de la classe de base ;
- Pas d'héritage multiple en Java : une classe peut hériter d'une seule autre ;

1. Généralisation - spécialisation

- Le mot-clé extends indique la classe-mère :

```
class Voiture extends Vehicule { ... }
class Velo extends Vehicule { ... }
class VTT extends Velo { ... }
```

- On dit qu'une classe-fille (ou sous-classe) spécialise sa classe-mère (ou super-classe) ;
- Tandis qu'une classe-mère généralise sa classe-fille ;

Que peut faire une sous-classe ?

- La classe qui hérite peut :
  - Ajouter des méthodes, des variables et des constructeurs ;
  - Surcharger des méthodes ;
  - Redéfinir des méthodes ;

Pas tout a fait correcte

Exemple : la classe Point

```
class Point
{
  private int x, y ;
  public void initialise (int a, int b)
    x = a ; y = b ;
  public void deplace (int dx, int dy)
    x += dx ; y += dy ;
  public void affiche ()
    System.out.println ("Je suis en "
    + x + " " + y) ;
}
```


Exemple : la classe Point

```
class Point
{
  private int x, y ;
  public void initialise (int a, int b)
    x = a ; y = b ;
  public void deplace (int dx, int dy)
    x += dx ; y += dy ;
  public void affiche ()
    System.out.println ("Je suis en "
    + x + " " + y) ;
}
```

- Imaginons que nous ayons besoin d'une classe Pointcol, destinée à manipuler des points colorés d'un plan ;
- Elle disposera des mêmes fonctionnalités que Point et une méthode "colore" chargée de définir la couleur ;
- Elle héritera donc de Point ;

Exemple : la classe Point

```
class Point
{
  private int x, y ;
  public void initialise (int a, int b)
    x = a ; y = b ;
  public void deplace (int dx, int dy)
    x += dx ; y += dy ;
  public void affiche ()
    System.out.println ("Je suis en "
    + x + " " + y) ;
}
```

```
class Pointcol extends Point
{
    private byte couleur ;
    public void colore (byte
    couleur)
    {
        this.couleur = couleur ;
    }
}
```


Nous pouvons maintenant déclarer des variables de type Pointcol :

```
public class TestPointcol
{
    public static void main (String args[])
    {
        Pointcol pc = new Pointcol() ;
        pc.initialise (3, 5) ;
        pc.colore ((byte)3) ;
        pc.deplace (2, -1) ;
        pc.affiche() ;
    }
}
```

Remarque

Si on veut ajouter à Pointcol une méthode affichec fournissant à la fois les coordonnées du point et sa couleur, ceci ne sera pas permis :

```
public void affichec ()
{
        System.out.println ("Je suis en " + x + " " + y) ;
        System.out.println ("Ma couleur est " + couleur) ;
}
```


Remarque

Pour écrire la méthode affichec, on peut s'appuyer sur la méthode affiche de Point :

```
public void affichec ()
{
        affiche() ;
        System.out.println ("Ma couleur est " + couleur) ;
}
```

Exercices

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 ;
}
```

Ecrire la classe PointA, dérivée de Point disposant d'une méthode affiche pour afficher les coordonnées d'un point.

2) On dispose de la classe suivante :

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

Ecrire la classe PointNom, dérivée de Point permettant de manipuler des points définis par 2 coordonnées et un nom (char). On y prévoira les méthodes suivantes :

- setPointNom pour définir les coordonnées et le nom d'un objet de type PointNom ;
- affCoordNom pour afficher les coordonnées et le nom d'un objet de type PointNom ;