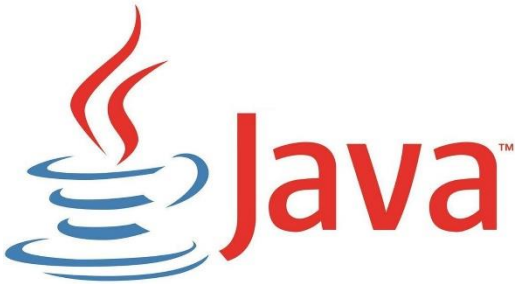
Java
- Chapitre : Héritage -

Ichrak MEHREZ



Présentation

- *Héritage* est un des concepts les plus importants de la programmation orientée objet.
- L'héritage est un mécanisme permettant de créer une nouvelle classe à partir d'une classe existante en lui proférant ses propriétés et ses méthodes.
- Une classe Java dérive toujours d'un autre classe: ***Object***
- Il représente la relation: **EST-UN**

Un chat **est un** animal

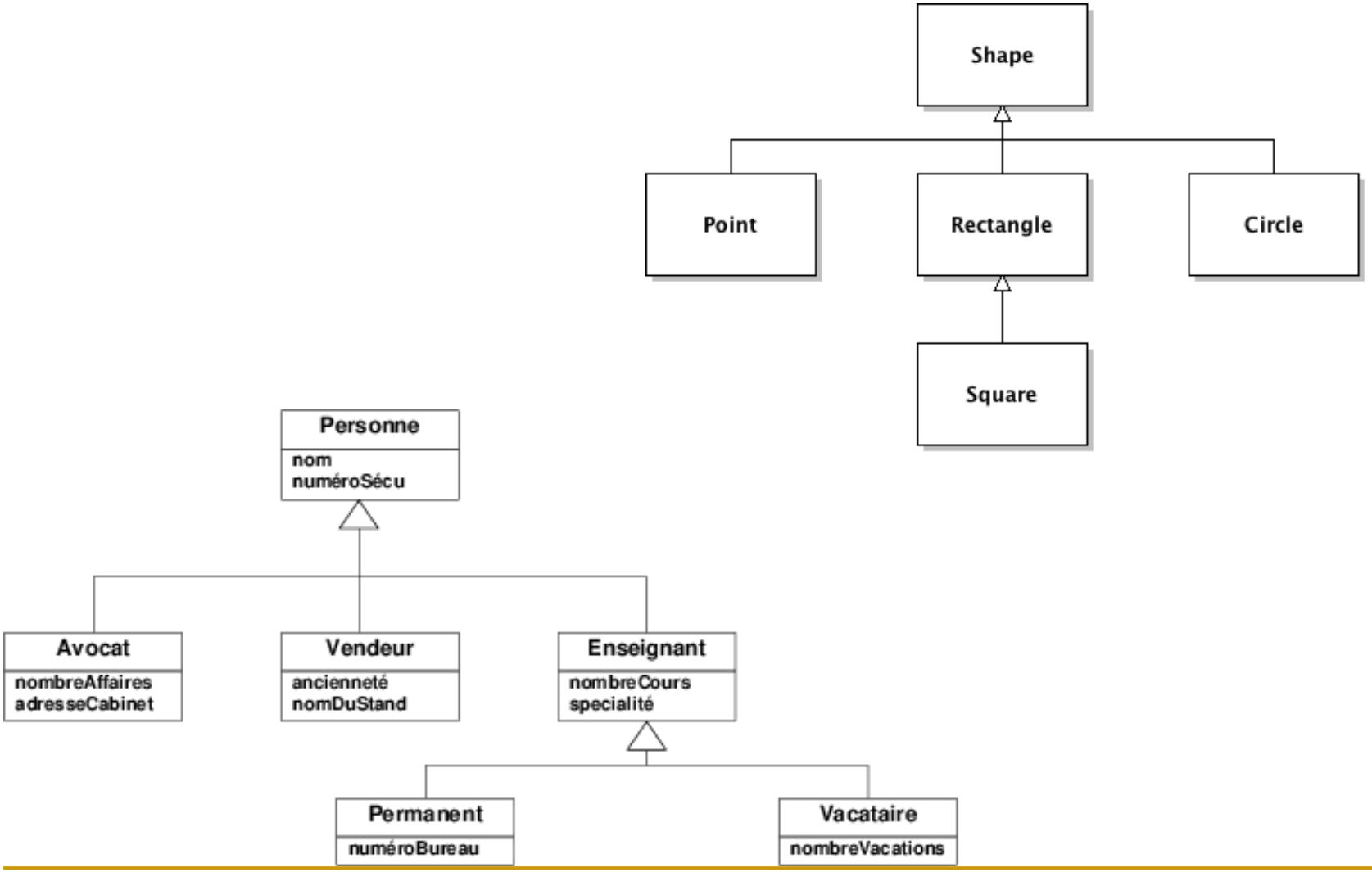
Un étudiant **est une** Personne

Un carrée **est une** forme

Présentation

- L'héritage est mis en œuvre par la construction de classes dérivées.
- classe héritée == sous-classe == classe fille == classe dérivée
- classe de base == super-classe == classe mère
- Lors de l'instanciation d'une classe, celle-ci hérite de (cela signifie qu'elle reçoit) l'ensemble des propriétés et des méthodes de sa superclasse.

Exemples


Shape
Point
Rectangle
Circle
Square
Personne
nom
numéroSécu
Avocat
nombreAffaires
adresseCabinet
Vendeur
ancienneté
nomDuStand
Enseignant
nombreCours
specialité
Permanent
numéroBureau
Vacataire
nombreVacations

« Extends »

- En Java, l'héritage est mis en œuvre au moyen du mot-clé ***extends***

```
public class Mere {

    //Attributs

    //Méthodes

}
```

```
public class Fille extends Mere {

    //Attributs

    //Méthodes

}
```

« super »

- Si un constructeur de la classe dérivée appelle explicitement un constructeur de la classe de base, cet appel doit être obligatoirement la première instruction de constructeur.
- Il doit utiliser pour cela, le mot clé **super**

```
public class Personne {
    private int id;
    private String nom;
    private String prenom;
    public Personne(int i,String n,String p){
    this.id=i; this.nom=n; this.prenom=p;
    }
}
```

```
public class Etudiant extends Personne {
    private int numInsc;
    Etudiant(int i,String n,String p, int num)
    {
        super(i,n,p);
        numInsc=num;
    }
}
```

Droits d'accès

- Si les membres de la classe mère sont :
  - ***public*** : la classe dérivée aura accès à ces membres (champs et méthodes),
  - ***private*** : la classe dérivée n'aura pas accès aux membres privés de la classe de mère.
  - ***protected*** : Un membre de la classe mère déclaré protected est accessible à ses classes dérivées ainsi qu'aux classes du même paquetage.

« instanceof »

- L'opérateur ***instanceof*** permet de savoir à quelle classe appartient une instance

```
public class Test {

    public static void main(String[] args) {
        Personne p=new Personne(1,"Med","Ali");
        Etudiant e=new Etudiant(2,"Sami","Ali",123);
        if(e instanceof Etudiant)
            System.out.println("Etudiant");
        else if(e instanceof Personne)
            System.out.println("Personne");

    }
}
```

```
Etudiant
-------------------
BUILD SUCCESS
-------------------
```

Redéfinition des méthodes

- Une méthode définie dans une classe peut être **redéfinie** dans une classe dérivée.
  - ***Signature*** identique :même type de retour et mêmes paramètres.
  - La redéfinition ne doit pas diminuer les *droits d'accès* à une méthode

Redéfinition des méthodes

```java
public class Test {
    public static void main(String[] args) {
        Personne p=new Personne(1,"Med","Ali");
        Etudiant e=new Etudiant(2,"Sami","Ali",123);
        p.afficher();
        e.afficher();
    }
}
```

```java
public class Personne {
    private int id;
    private String nom;
    private String prenom;
    public Personne(int i,String n,String p){
    this.id=i; this.nom=n; this.prenom=p;
    }
    public void afficher()
    {System.out.println("Une Personne");}
}
```

```java
public class Etudiant extends Personne {
    private int numInsc;
    Etudiant(int i,String n,String p, int num)
    {
        super(i,n,p);
        numInsc=num;
    }
    @Override
     public void afficher()
    {System.out.println("Un Etudiant");}
}
```

Redéfinition des méthodes

```
public class Etudiant extends Personne {
    private int numInsc;
    Etudiant(int i,String n,String p, int num)
    {
        super(i,n,p);
        numInsc=num;
    }
    @Override
     public void afficher()
    {System.out.println("Un Etudiant");}
}
```

Annotation ***@override***: un « *commentaire* » utilisé pour définir une méthode qui est héritée de la classe mère.

Cette annotation permet au compilateur de vérifier que la méthode est correctement utilisée (mêmes arguments, même type de valeur de retour) et affichera un message d'avertissement si ce n'est pas le cas

Surcharge des méthodes

- La surcharge permet à une classe de définir la même méthode plusieurs fois avec des signatures différentes.

- En Java, la surcharge de méthode ne peut pas être effectuée en modifiant uniquement le type de retour de la méthode.

  ➔ Le type de retour peut être identique ou différent dans la surcharge de méthode. Mais vous devez changer le paramètre.