Leçon 5: Les syntaxes de base en java : classe abstraite, interface, héritage

MIT University

M1 GL/RT

Sept 2015

L'héritage

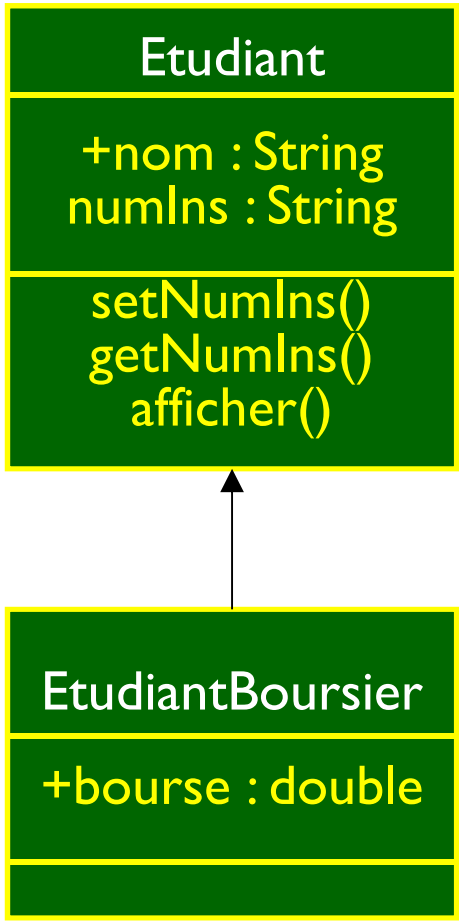
✺L'héritage permet de définir une nouvelle classe à partir d'une (ou plusieurs) classe(s) existante(s).

✺Les classes sont souvent organisées en hiérarchies; toutes les classes Java héritent de la classe **java.lang.Object**.

✺Java n'autorise que l'héritage simple,

✺L'héritage multiple est "remplacé" par la notion d'**interface**.

L'héritage simple : exemple



L'héritage simple : exemple

```
public class EtudiantBoursier extends Etudiant {

        private double bourse;

   public EtudiantBoursier(String nom, double bourse) {
                super(nom); //appel au constructeur
Etudiant(String)
                this.bourse = bourse;
   }
   public EtudiantBoursier(String nom) {
                this(nom, null); //appel au constructeur précédant
   }

        public void afficher() { // redefinition
                System.out.println("etudiant :"+ this.nom);
                System.out.println("bourse :"+ this.bourse);
   }
}
```

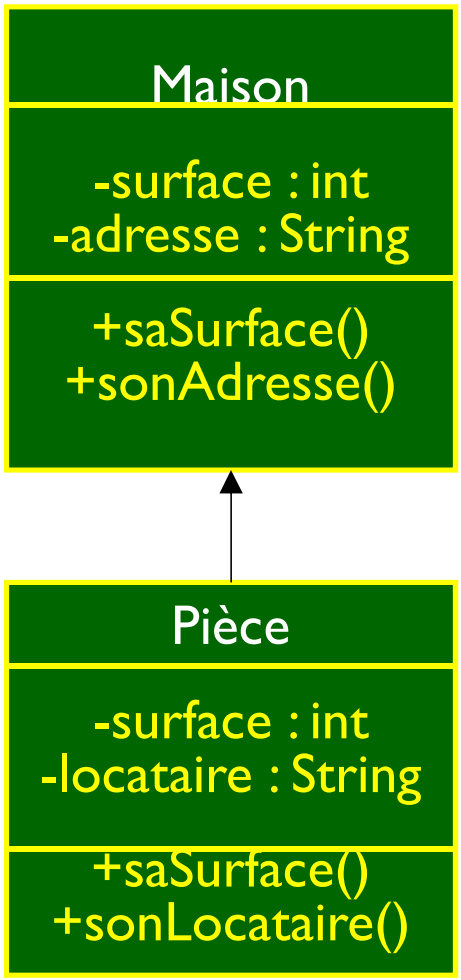
La pseudo-variable super

✺En faisant hériter une classe d'une autre, on peut définir une méthode dont l'identificateur est le même que celui de sa classe mère.

➔masquage de la méthode de la classe mère.

✺La pseudo-variable super permet d'accéder à la méthode masquée de la classe mère.

La pseudo-variable super : exemple



Les classes abstraites

- Elles permettent de définir l'interface des méthodes.
- Elles contiennent au moins une méthode abstraite.
- Une méthode abstraite est précédée du mot-clé abstract et ne possède pas de corps.
- Elles ne peuvent pas être instanciées.
- Elles doivent être dérivées en sous-classes fournissant une implémentation à toutes les méthodes abstraites.
- Elles sont définies par l'introduction du mot-clé abstract.

```
abstract class FormeGeometrique {
abstract double perimetre();
abstract double surface();
}
```

Les interfaces (1/4)

- Ce sont des classes abstraites dont l'instanciation serait sans intérêt. Elles ne peuvent donc pas être instanciées.
- Elles peuvent hériter d'autres interfaces (héritage simple).
- Elles ne peuvent contenir que des attributs statiques constants (*static final*) et des méthodes publiques non implémentées.
- Elles sont implémentées par une ou plusieurs classes.
- Une classe peut implémenter une ou plusieurs interfaces.
- Lorsqu'une classe implémente une interface, elle doit implémenter toutes les méthodes définies dans l'interface.

Utilité de l'interface

- Permet de savoir qu'une classe contient les implantations de certaines méthodes
- On peut utiliser ces méthodes sans connaître les détails de leur implantation
- Souvent utilisée pour des types abstraits de données (e.g. pile, queue, …)

Les interfaces (2/4)

```
public interface FormeGeometrique {
        public double perimetre();
        public double surface();
}
```

Utiliser une classe interface :

```
public class Rectangle implements FormeGeometrique {
        public double larg, long;
        public Rectangle(double long, double larg) {
                this.long=long;
                this.larg=larg;
        }
        public double perimetre() {return 2*(larg+long);}
        public double surface() {return larg*long;}
}
```

Les interfaces (3/4)

```
public class Carre implements FormeGeometrique {
      public  double cote;
      public Carre(double cote) {this.cote = cote;}
      public double perimetre() {return 4*cote;}
      public double surface() {return cote*cote;}
}
public class Cercle implements FormeGeometrique {
                  public final static double PI = 3.1416;
                  public  double rayon;
      public Cercle (double rayon) {this.rayon = rayon;}
                  public double perimetre() {return 2*PI*rayon;}
                  public double surface() {return PI*rayon*rayon;}
                  public void afficher() {
                        System.out.println("cercle de rayon"+rayon); }
}
```

Les interfaces (4/4)

```
public class Figures {
        public static void main (String[] args) {
FormeGeometrique[] formes;
formes = new FormeGeometrique[3];
formes[0] = new Cercle(10);
formes[1] = new Carre(4);
formes[2] = new Rectangle(5,8);
for (int i=0;i<formes.length;i++){
        System.out.print("surface de la forme"+i+" : ");
        System.out.println(formes[i].surface());
}
        }
}
```

La classe Object

✺Toutes les classes héritent de la classe Object qui contient certaines propriétés et méthodes intéressantes permettant, par exemple :

- ✍Connaître la classe d'un objet : getClass()
- ✍Comparer des objets : equals()
- ✍Afficher des objets : toString()

Exemple:

```
public class Fraction {
    public int num, den ;
    public Fraction (int n, int d) { this.num=n; this.den=d;}
            public String toString() {
                return this.num + "/" + this.den;}
}
```

Exemple d'héritage : compte et compte d'épargne
(1/3)

```
class Compte {

  private String id;
        private long solde;

        public Compte (String id, long depot){
              this.id = id;
              this.solde = depot;
        }
        public String getId (){
              return this.id;
        }
        public float getSolde (){
              return this.solde;
        }
}
```

Exemple d'héritage : compte et compte d'épargne
(2/3)

```
class CompteEpargne extends Compte {
    private float taux;
            private int annees;
            public CompteEpargne (String id, float depot, float
taux){
                    super (id, depot);
                    this.taux = taux;
            }
            public void setAnnees (int annees){
                  if (annees >= 0)  this.annees = annees;
            }
            public int getAnnees(){         return this.annees; }
            public float getTaux(){         return this.taux; }
            public float getSolde (){
                  float solde = super.getSolde();
                  for (int i=0; i<this.annees; i++) solde *= 1 + this.taux;
                  return solde;
            }
}
```

Exemple d'héritage : compte et compte d'épargne
(3/3)

```
class CalculInterets{
   public static void main(String arg[]){
                    Compte compte1 = new Compte("A01", 100000f);
                    CompteEpargne compte2 = new CompteEpargne("E99",
100000f, 0.1f);
                    compte2.setAnnees(5);
                    Compte c;
                    String s = "L'argent qui dort ne rapporte rien:";
                    c = compte1;
                    s += "\n solde du compte n° " + c.getId() + ":" +
c.getSolde();
                    c = compte2;
                    s += "\n solde du compte n° " + c.getId() + ":" +
c.getSolde();
          javax.swing.JOptionPane.showMessageDialog(null, s);
                  }
}
```