III. Conception de classes en Java

III.4 Héritage : principe et utilisation

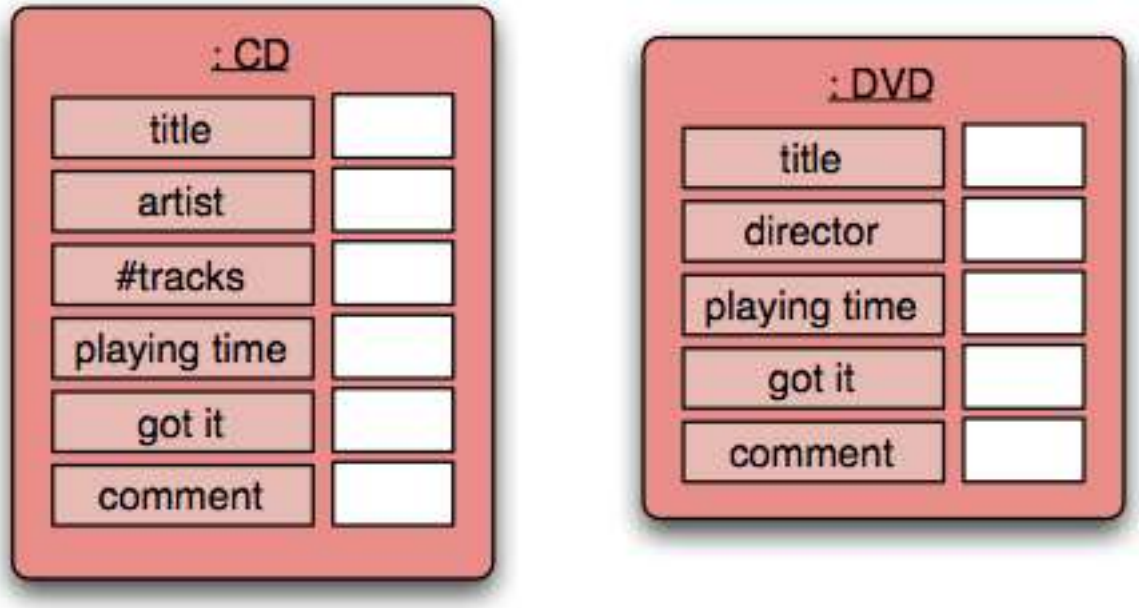
Héritage

- Exemple: la base de données multimédia DoME ("Database of Multimedia Entertainment“)

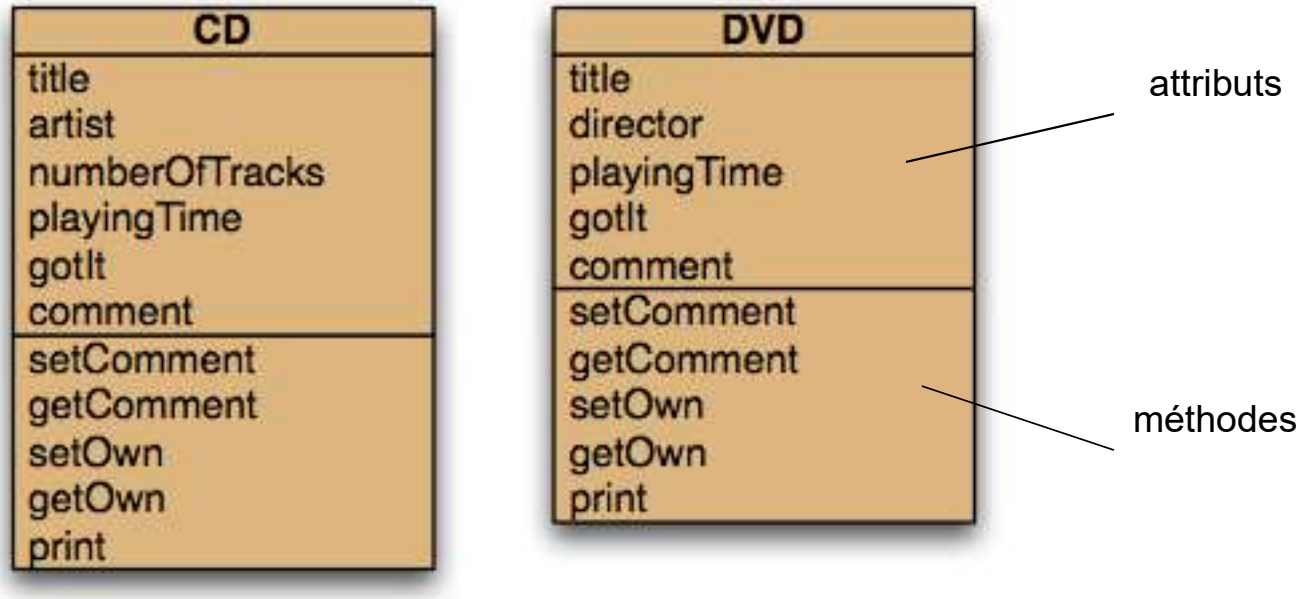
- DoME gère des informations sur des CD et des DVD
  - CD: title, artist, # tracks, playing time, got-it, comment
  - DVD: title, director, playing time, got-it, comment

- Permet de chercher des informations et d’imprimer des listes d’articles

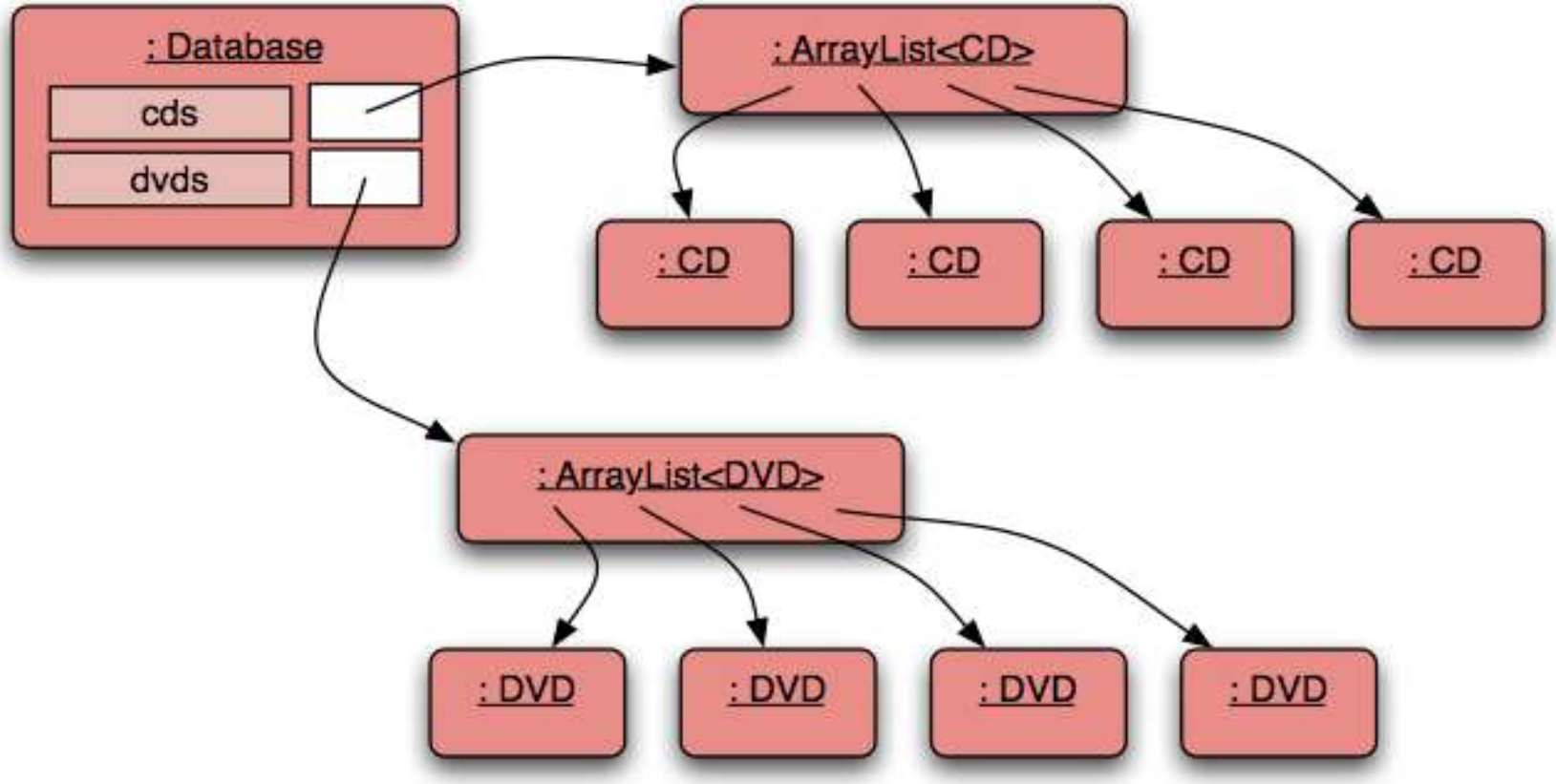
Objets de l'application DoME



Classes de DoME



Un exemple d'objets dans DoME



```java
public class CD
{
   private String title;
   private String artist;
   private String comment;

   public CD(String t, String a)
   {
         title = t;
         artist = a;
         comment = " ";
   }

   public void setComment(String c)
   { ... }

   public String getComment()
   { ... }

   public void print()
   { ... }
   ...
}
```

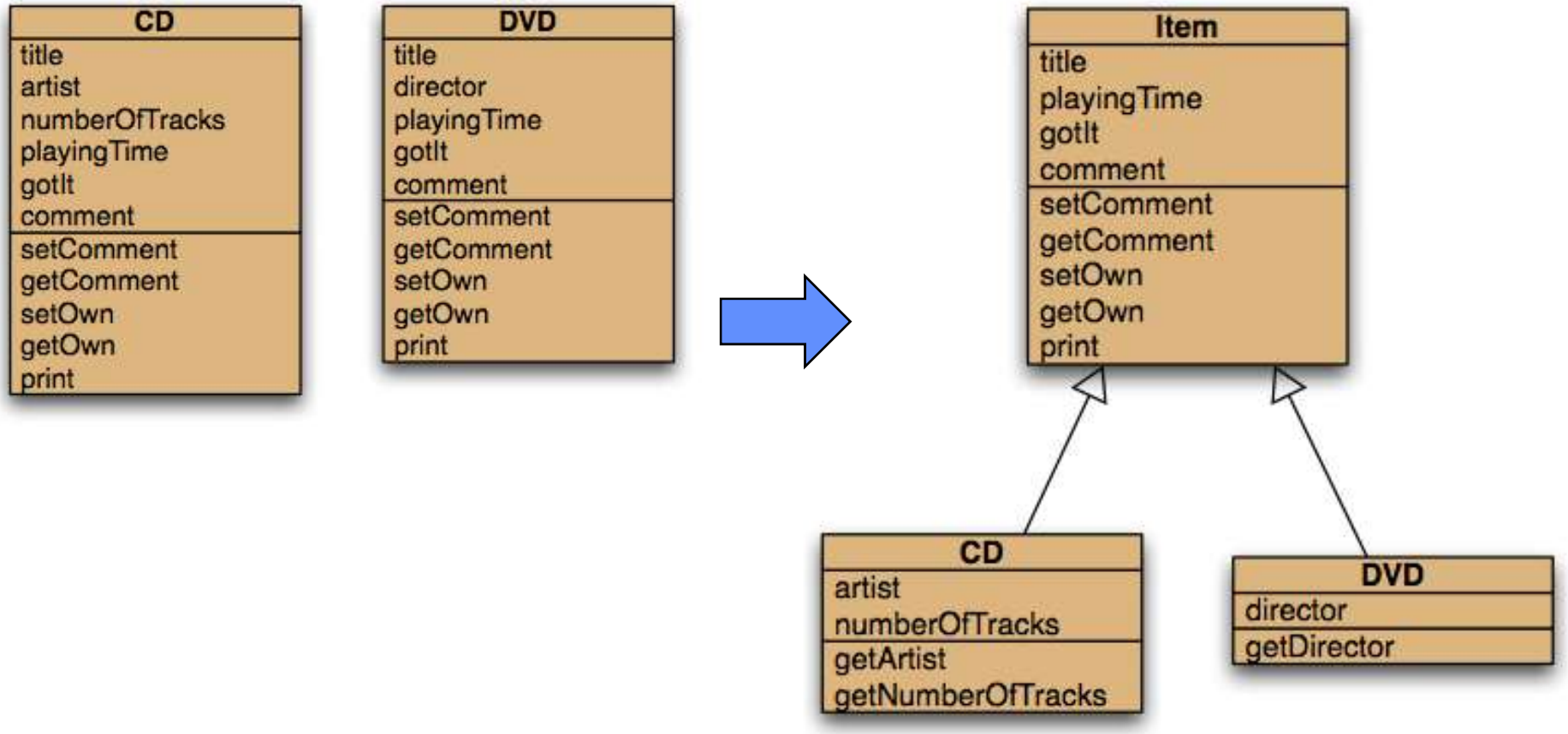
```java
public class DVD
{
   private String title;
   private String director;
   private String comment;

   public DVD(String t, String d)
   {
         title = t;
         director = d;
         comment = " ";
   }

   public void setComment(String c)
   { ... }

   public String getComment()
   { ... }

   public void print()
   { ... }
   ...
}
```

```
public class Database {

   private ArrayList<CD> cds;
   private ArrayList<DVD> dvds;
   ...
   public void list()
   {
       for(CD cd : cds) {
          cd.print();
          System.out.println();  // empty line between items
       }

       for(DVD dvd : dvds) {
          dvd.print();
          System.out.println();  // empty line between items
       }
   }
}
```

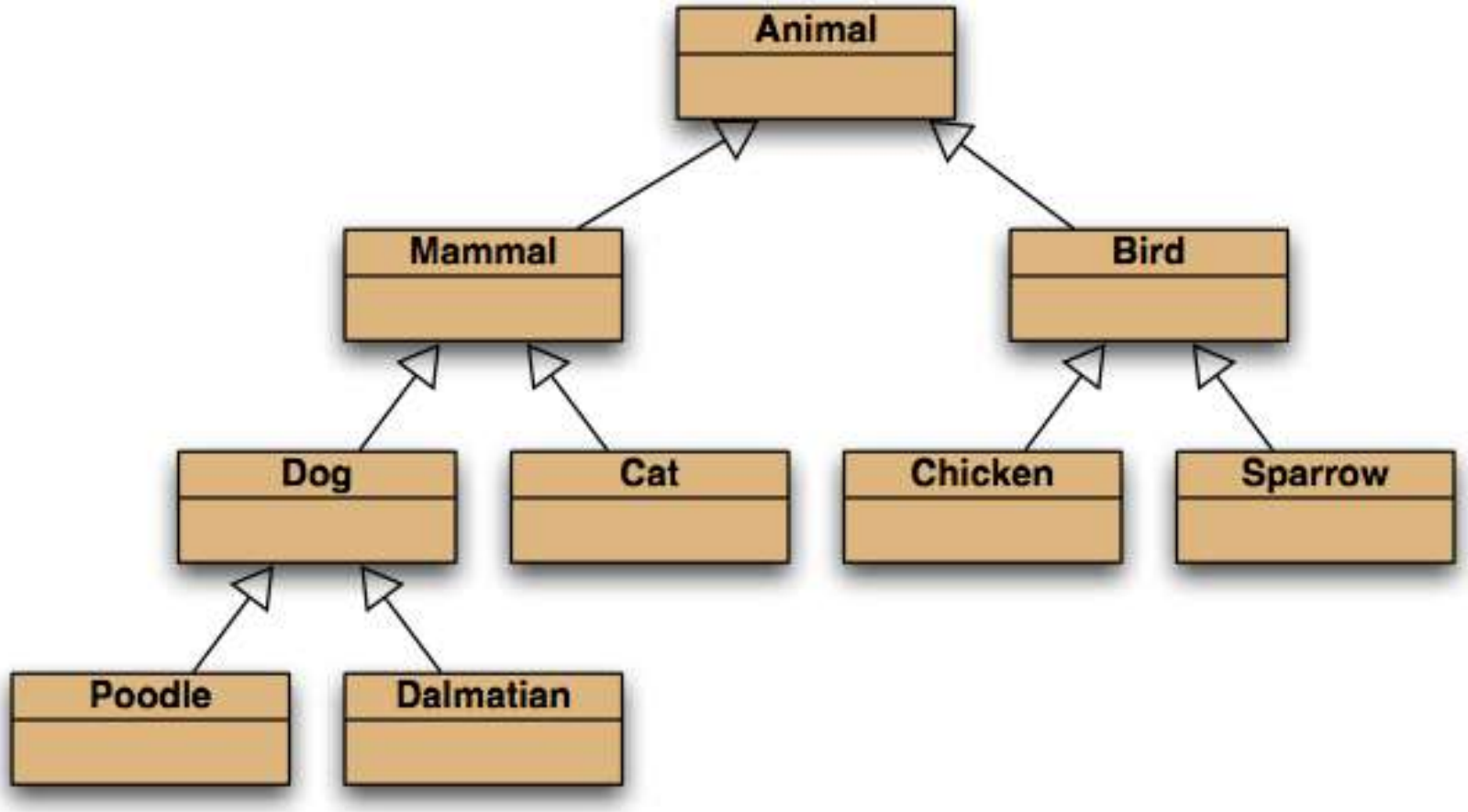
Critiques sur DoME

- Duplication de code
  - Les classes CD et DVD sont **très similaires**
    - Beaucoup de fonctions dont le code est identique ou presque (copier-coller)
      - *très mauvais pour la maintenance*
      - *peu efficace…*
  - Idem pour la classe Database
  - C'est un problème de *conception*
  - Peut-on *factoriser?*
    - CD et DVD partagent plusieurs caractéristiques

Héritage : principe



Héritage : plus que de la factorisation, une notion naturelle



Héritage – mode d'emploi

- On définit une **superclasse**
- On définit des **sous-classes**
- La superclasse definit les *attributs et méthodes communs*
- Les sous-classes **héritent** des attributs et des méthodes de la superclasse
- Les sous-classes **ajoutent** leurs propres attributs et méthodes ou **redéfinissent** des méthodes

Super-classe

```
public class Item
{
    private String title;
    private int playingTime;
    private boolean gotIt;
    private String comment;

    // suivent les constructeurs et méthodes.
}
```

Sous-classes

```
public class CD extends Item
{
    private String artist;
    private int numberOfTracks;

 // suivent les constructeurs et méthodes

}
```

```
public class DVD extends Item
{
    private String director;

     // suivent les constructeurs et méthodes
}
```

Héritage et constructeurs

```
public class Item
{
    private String title;
    private int playingTime;
    private boolean gotIt;
    private String comment;

    /**
     * Initialize the fields of the item.
     */
    public Item(String theTitle, int time)
    {
        title = theTitle;
        playingTime = time;
        gotIt = false;
        comment = "";
    }

    // methods …
}
```

Héritage et constructeurs

```
public class CD extends Item
{
    private String artist;
    private int numberOfTracks;

    /**
     * Constructor for objects of class CD
     */
    public CD(String theTitle, String theArtist,
              int tracks, int time)
    {
        super(theTitle, time);
        artist = theArtist;
        numberOfTracks = tracks;
    }

    // methods omitted
}
```

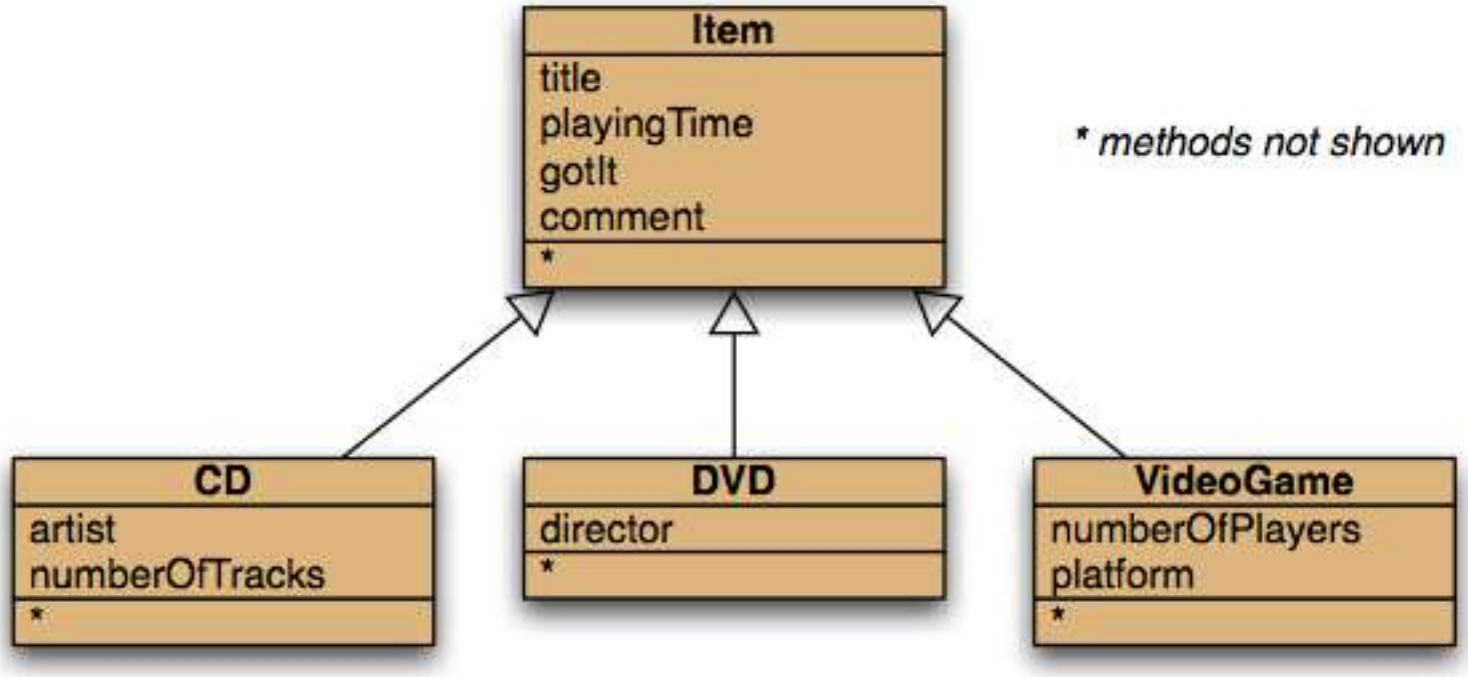
Constructeur de sous-classe

- Doit toujours contenir un appel à `super`
- Cet appel doit être la première instruction du constructeur
- Si absent, l'appel `super`() est inséré par défaut
  - … et ça ne marche que si la super-classe possède un constructeur sans paramètres

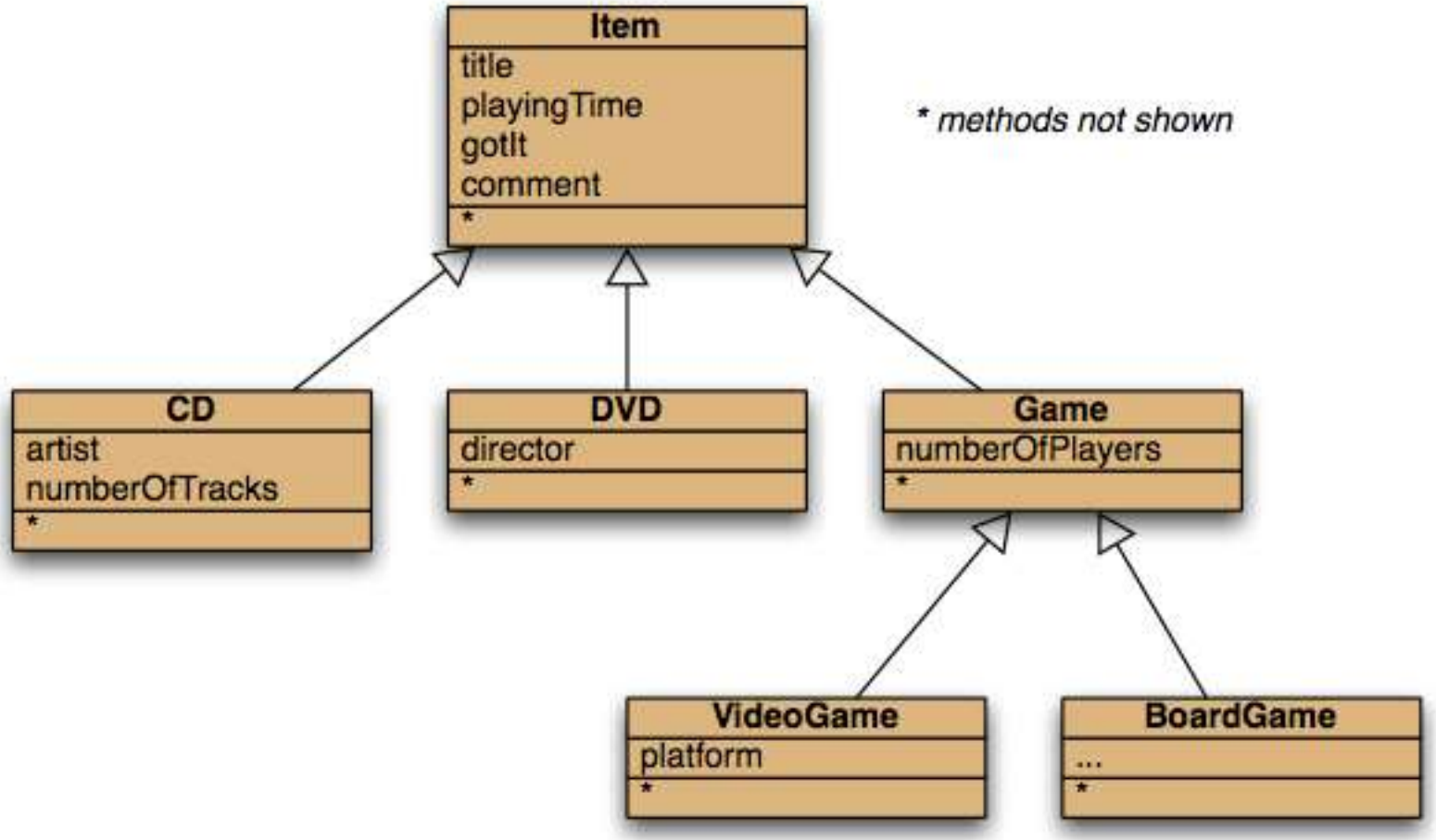
```
public class Database
{
    private ArrayList<Item> items;

    /**
     * Construct an empty Database.
     */
    public Database()
    {
        items = new ArrayList<Item>();
    }

    /**
     * Add an item to the database.
     */
    public void addItem(Item theItem)
    {
        items.add(theItem);
    }
    ...
}
```

```
/**
 * Print a list of all currently stored CDs and
 * DVDs to the text terminal.
 */
public void list()
{
    for(Item item : items) {
        item.print();
        // Print an empty line between items
        System.out.println();
    }
}
```

Ajout d'une autre sous-classe…



… et d'autres héritages



Sous-classes, sous-typage

- Une classe définit un type
- Une sous-classe définit un sous-type
- Les objets des sous-classes peuvent être utilisés là où des objets du super-type sont attendus (substitution)

Sous-typage

Avant:

```
public void addCD(CD theCD)
public void addVideo(DVD theDVD)
```

Après:

```
public void addItem(Item theItem)
```

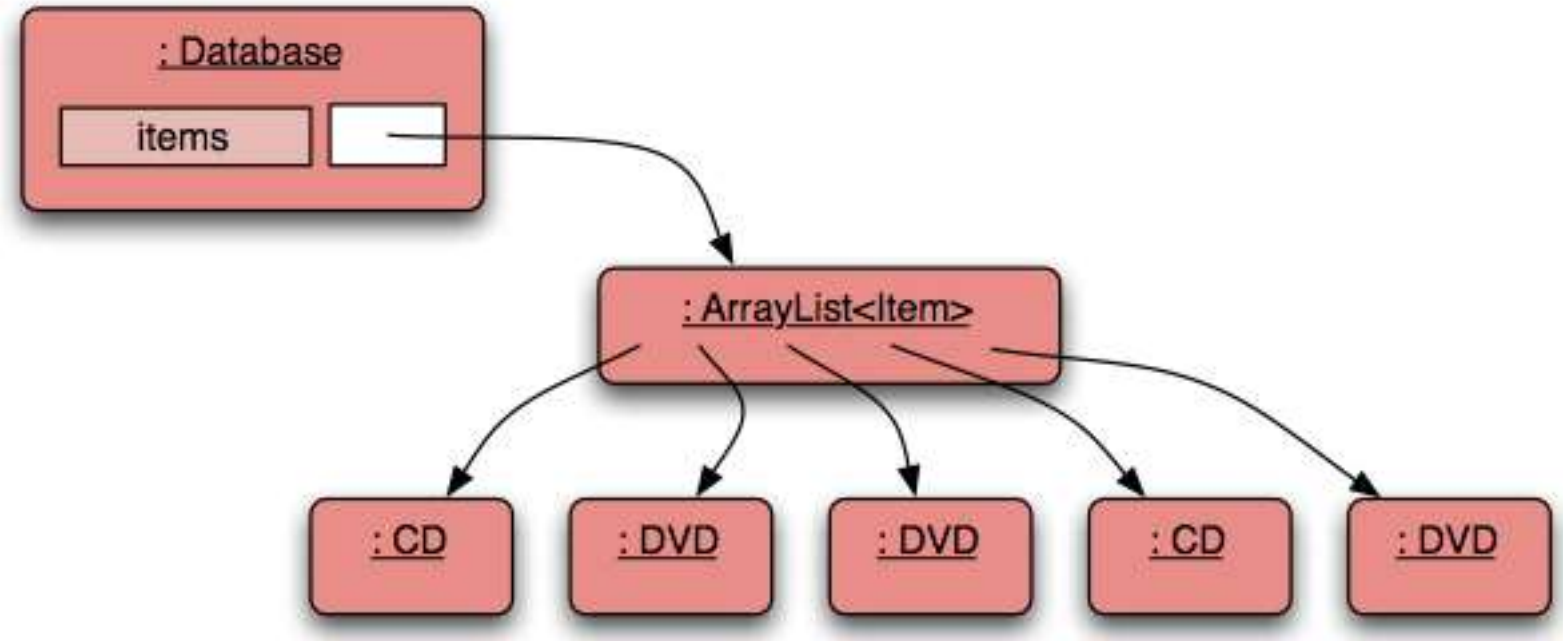
Appel de la méthode `addItem`:

```
DVD myDVD = new DVD(...);
database.addItem(myDVD);
```

Sous-typage et passage de paramètres

```
public class Database
{

    public void addItem(Item theItem)
    {
        ...
    }
}

DVD dvd = new DVD(...);
CD cd = new CD(...);

database.addItem(dvd);
database.addItem(cd);
```

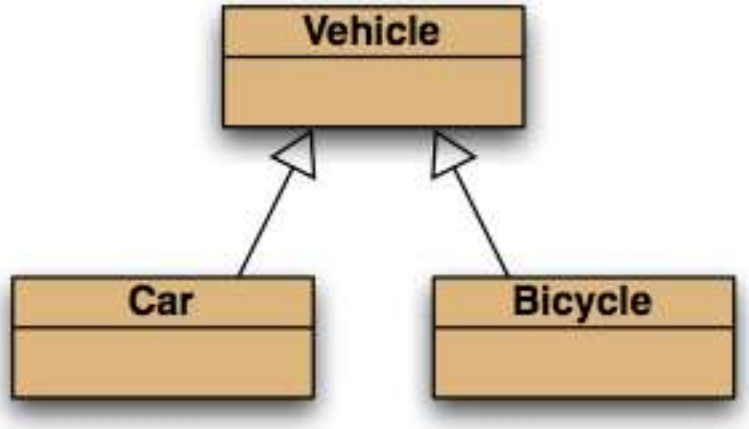
Diagramme d'objets



Variables polymorphes

- Les variables contenant différents types d'objets dans Java sont dites **polymorphes**
  - Elles peuvent contenir des objets de types différents (type déclaré mais aussi de tout sous-type de celui-ci)

Variables polymorphes

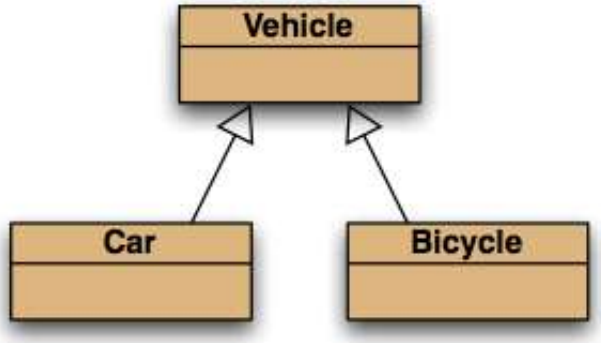



```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Conversions

- On ne peut pas affecter à une variable de type T un objet d'un super-type de T

```
Vehicle v;
Car c = new Car();
v = c; // correct;
c = v;  erreur!
```



- Conversion

```
c = (Car) v;  // ça ne marche que si v est un Car!
```

- Attention : la conversion est utilisée que pour éviter la "perte de type"
  - v est un véhicule de type Car sauf qu'à cet endroit du programme il est traité comme `Vehicle`
- La conversion ne modifie pas l'objet
- La correction de la conversion est vérifiée à l'exécution
  - **`ClassCastException`** si erreur
- A utiliser avec modération ...

Bilan des concepts introduits

- Héritage
- Super classe, sous classe, sous-type
- Super
- Variables polymorphes