实用Java教程

基于BlueJ的对象优先方法
（第3版）

[英] David J. Barnes　Michael Kölling 著

第八章 通过继承改进结构

Java语言程序设计
课程教案

本章将涉及到的主要概念

- 继承 (inheritance)
- 多态 (polymorphic)

本章的目标

如何使用面向对象程序设计语言所提供的内在机制来帮助改进应用程序总体结构?

案例—DoME

Database of Multimedia Entertainment：一个存储CD和DVD信息的应用程序

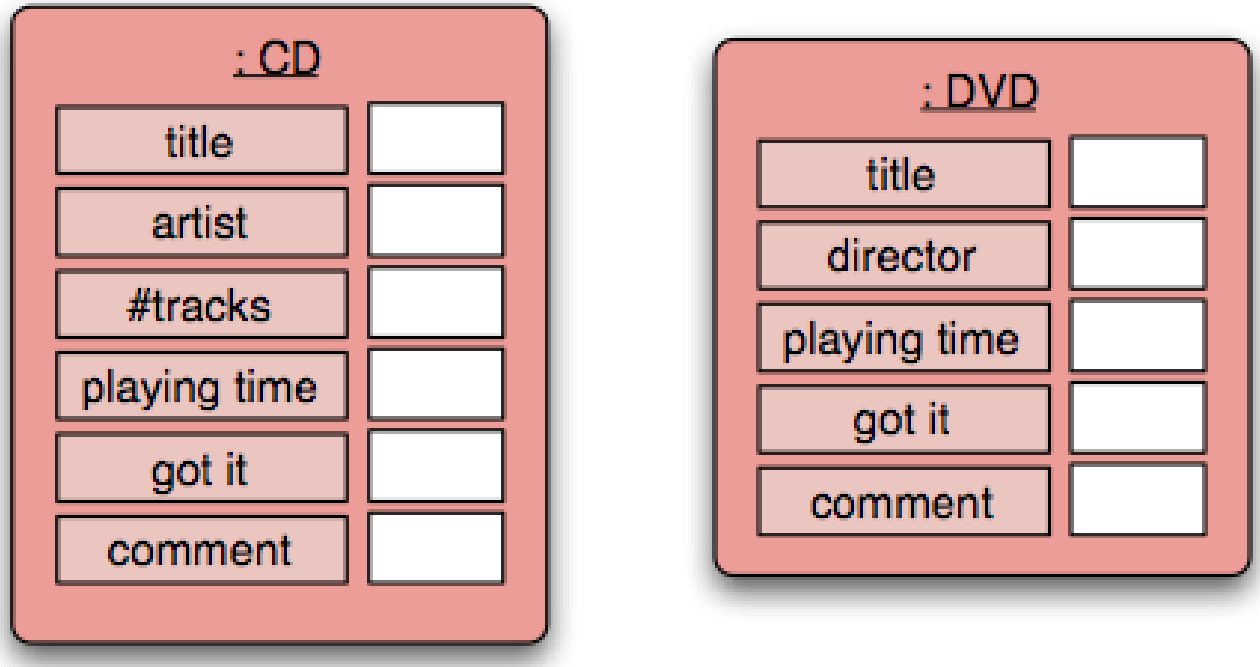
- 可以录入和存储所拥有的CD和DVD的信息供以后使用

  CD 专辑标题(title)、艺术家名称(artist)、曲目数量(numberOfTracks)、总播放时间(playing time)、拥有标记(got it)、专辑说明(comment)

  DVD DVD标题(title)、导演名称(director)、放映时间(playing time)、拥有标记(got it)、专辑说明(comment)
- 提供搜索功能，可以查找指定艺术家的CD或某位导演的DVD
- 提供打印功能，以便打印数据库中所有DVD或CD的列表

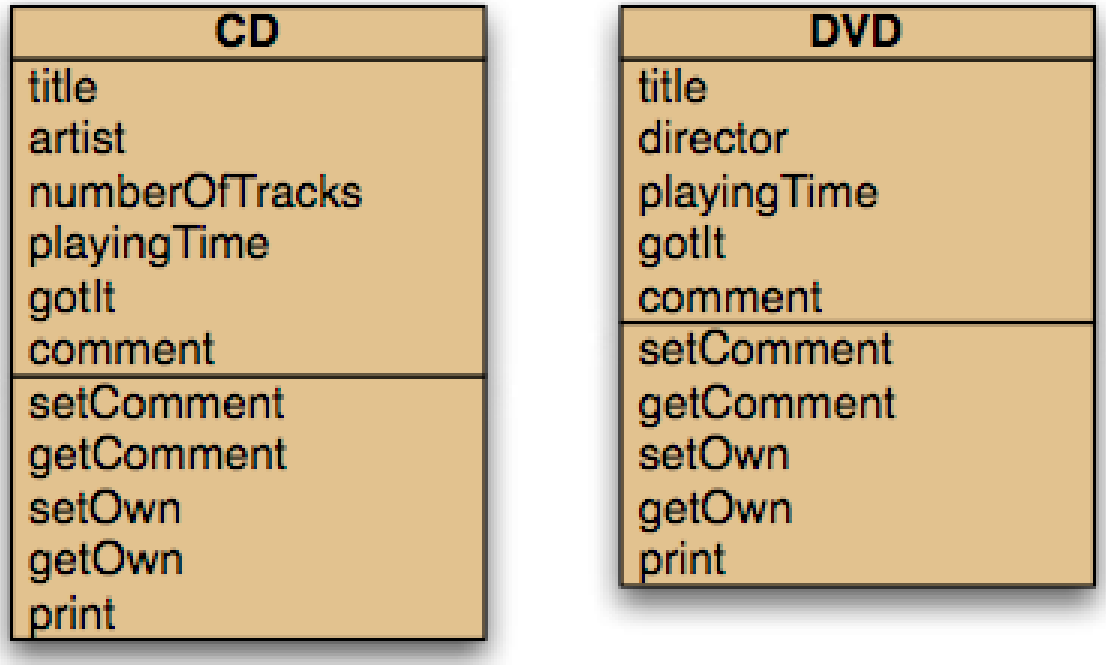
DoME中的类与对象

在CD和DVD对象中的字段



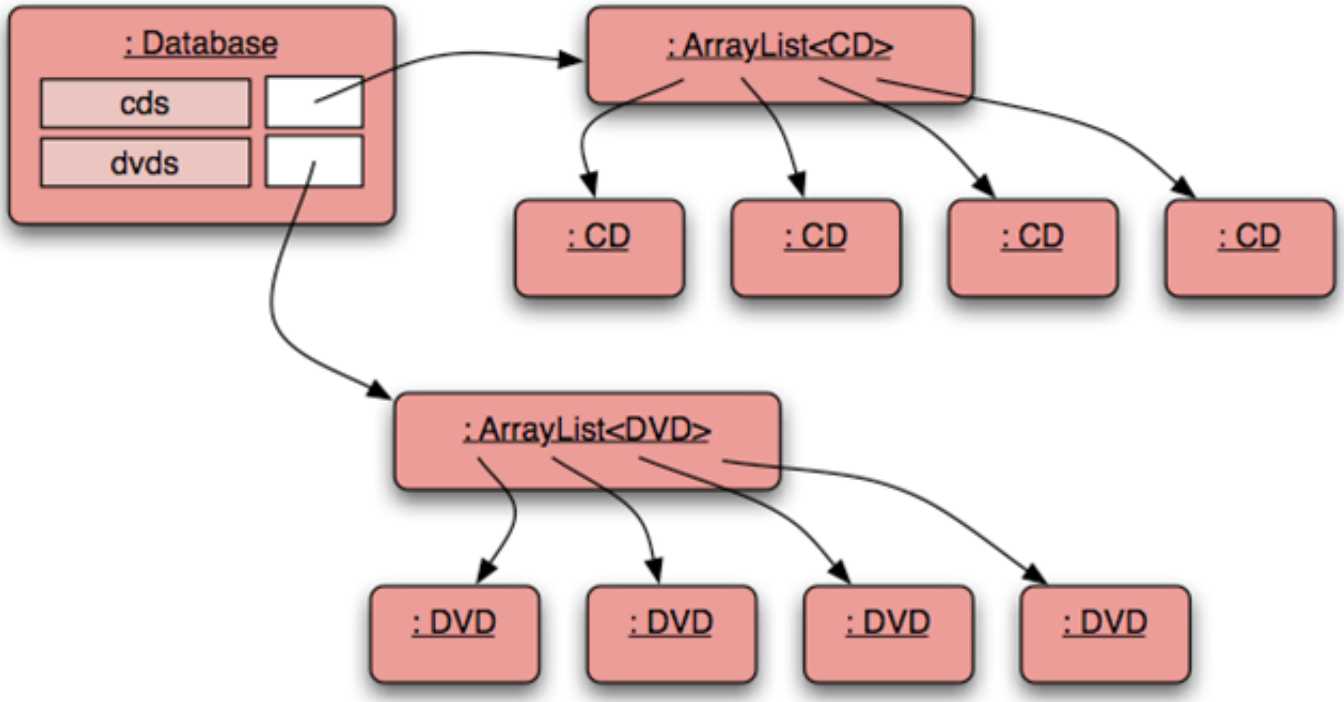
DoME中的类与对象

CD类和DVD类的方法



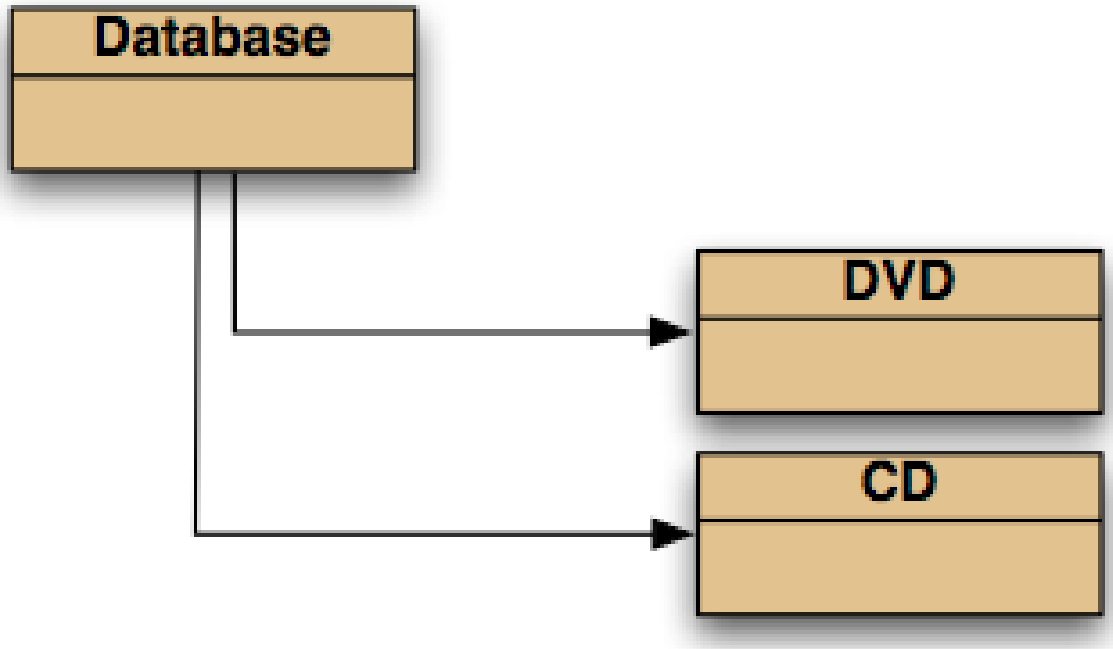
DoME中的类与对象

DoME程序中的对象



DoME中的类与对象

DoME的类图



CD类的源代码

CD.java (project: dome-v1)

```
public class CD
{
    private String title;
    private String artist;
    private int numberOfTracks;
    private int playingTime;
    private boolean gotIt;
    private String comment;

    public CD(String theTitle, String theArtist, int tracks, int time)
    {
        title = theTitle;           artist = theArtist;
        numberOfTracks = tracks;    playingTime = time;
        gotIt = false;              comment = " ";
    }
    public void setComment(String newComment) { ... }
    public String getComment() { ... }
    public void setOwn(boolean ownIt) { ... }
    public String getOwn() { ... }
    public void print() { ... }
}
```

DVD类的源代码

DVD.java (project: dome-v1)

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

    public DVD(String theTitle, String theDirector, int time)
    {
        title = theTitle;          director = theDirector;
        playingTime = time;
        gotIt = false;             comment = " ";
    }
    public void setComment(String newComment) { ... }
    public String getComment() { ... }
    public void setOwn(boolean ownIt) { ... }
    public String getOwn() { ... }
    public void print() { ... }
}
```

Database类的源代码

Database.java (project: dome-v1)

```
class Database
{
   private ArrayList<CD> cds;
   private ArrayList<DVD> dvds;

   public Database() { ... }
   public void addCD(CD theCD) { ... }
   public void addDVD(DVD theDVD) { ... }

   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
       }
   }
}
```

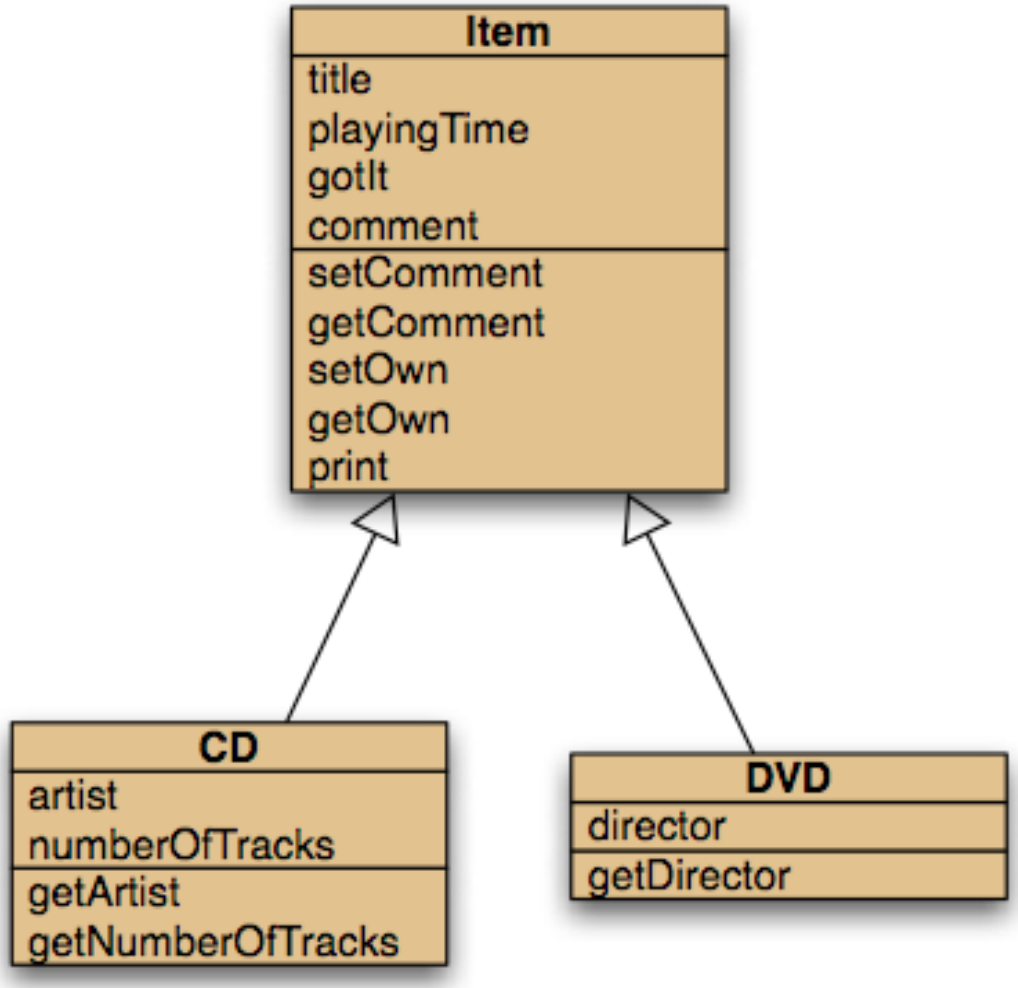
Demo

DoME (project: dome-v1)

对DoME程序的讨论

- CD类和DVD类中的代码重复
  - CD类和DVD类很相似，两个类的主要部分基本都是相同的
  - 在维护重复代码时会造成更多的工作量
  - 维护重复代码也会容易造成程序错误
- 在Database类中同样存在代码重复
  - 这个类里面几乎任何东西都是两份

使用继承


Item
title
playingTime
gotIt
comment
setComment
getComment
setOwn
getOwn
print
CD
artist
numberOfTracks
getArtist
getNumberOfTracks
DVD
director
getDirector

使用继承

继承

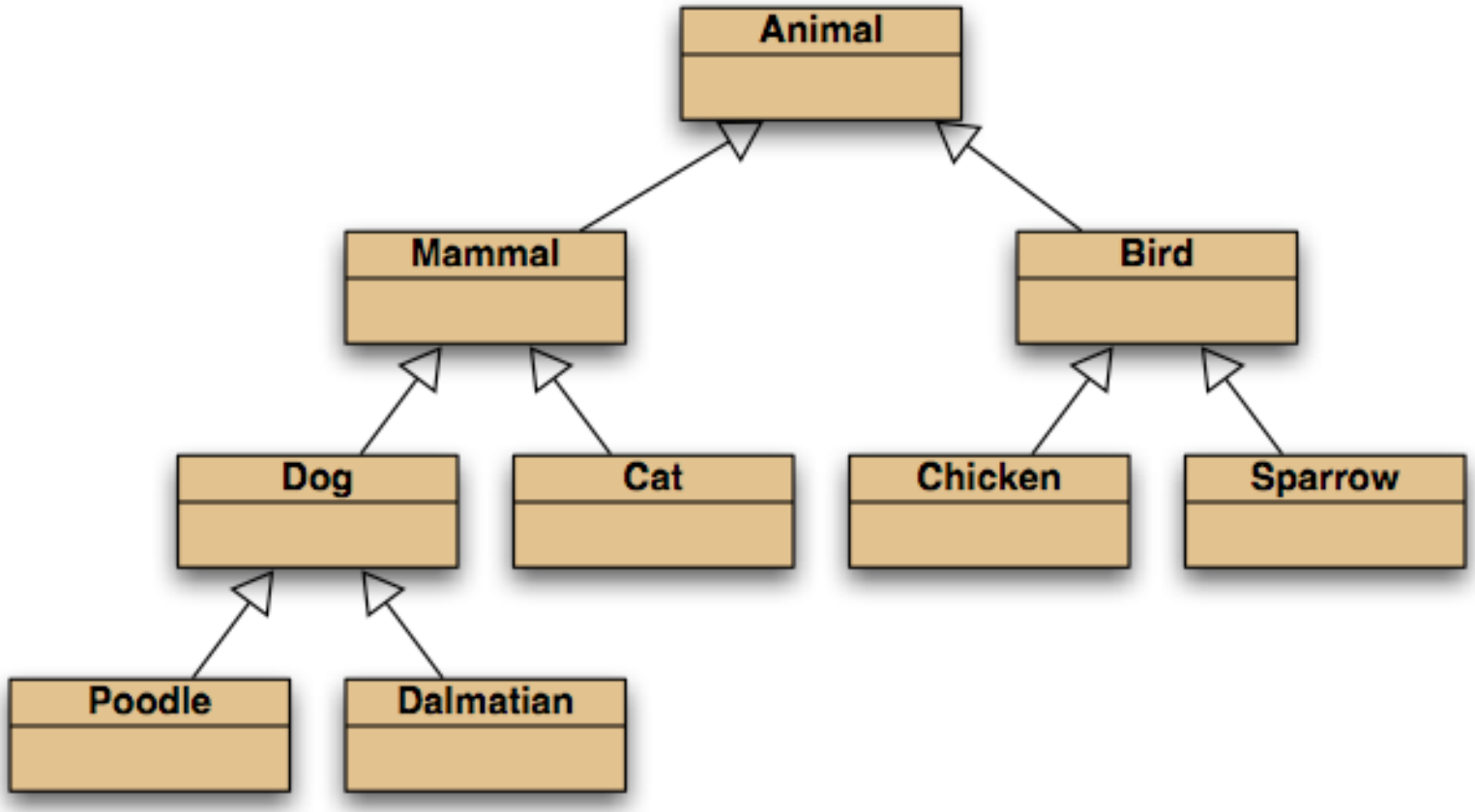
继承机制（inheritance）允许定义一个类作为另一个类的扩充版本。

- 定义一个父类（superclass）： Item
- 将CD和DVD分别定义为Item的子类（subclass）
- 超类用来定义共同的部分
- 子类继承（inherit）超类中的字段和方法
- 子类还可以添加自己的字段和方法

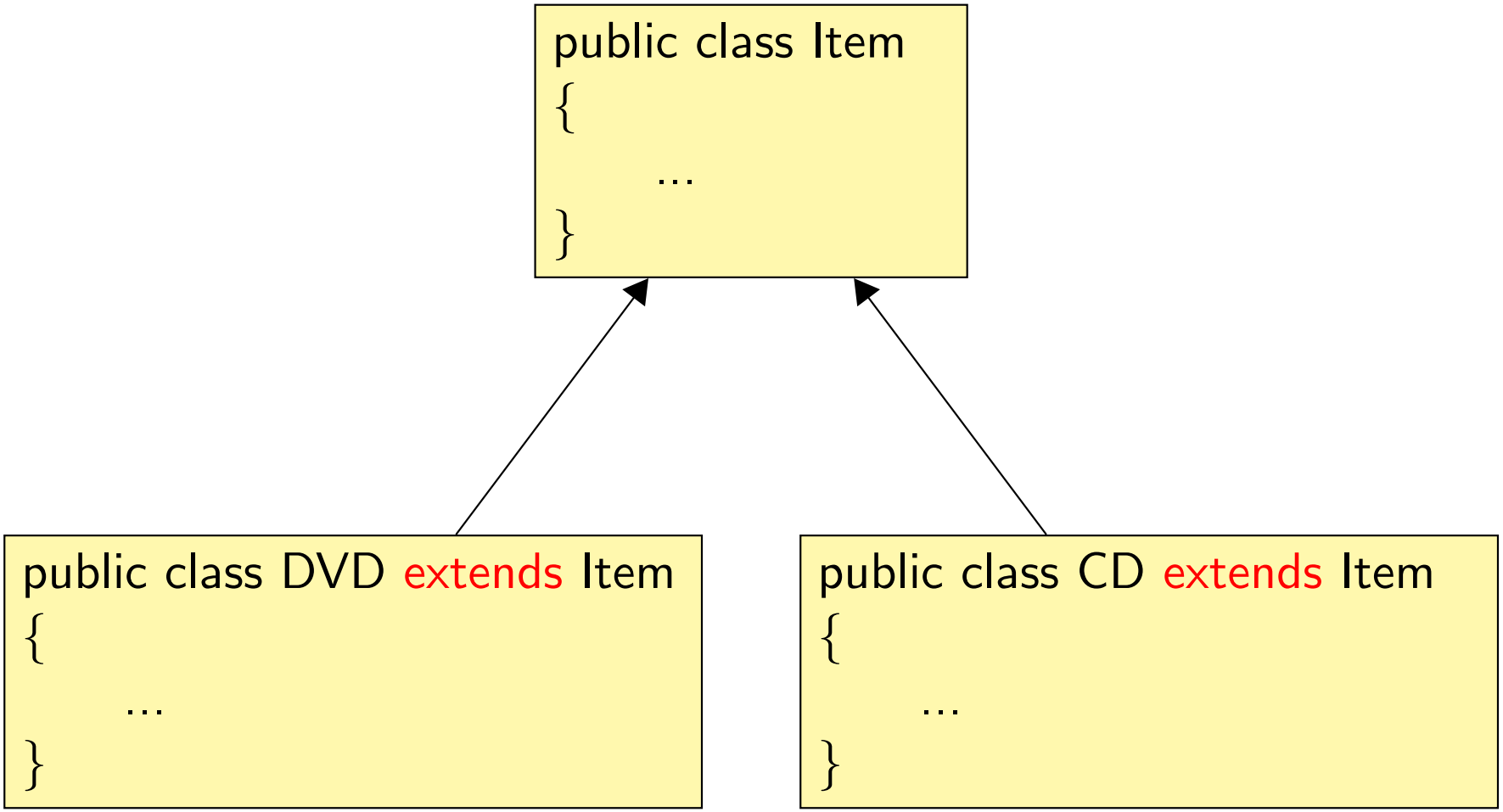
继承层次

继承层次

通过继承关系联系起来的一些类构成了一个继承层次（inheritance hierarchy）。



Java中的继承


public class Item
{
...
}
public class DVD extends Item
{
...
}
public class CD extends Item
{
...
}

继承与字段

Item.java (project: dome-v2)

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

    //constructors and methods omitted here.
    ...
}
```

CD.java (project: dome-v2)

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

    //constructors and methods omitted.
    ...
}
```

DVD.java (project: dome-v2)

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

    //constructors and methods omitted.
    ...
}
```

继承与初始化

Item.java (project: dome-v2)

```
...
public Item(String theTitle, int time)
{
    title = theTitle;    playingTime = time;
    gotIt = false;       comment = "";
}
...
```

CD.java (project: dome-v2)

```
...
public CD(String theTitle,
          String theArtist,
          int tracks, int time)
{
    super(theTitle, time);
    artist = theArtist;
    numberOfTracks = tracks;
}
...
```

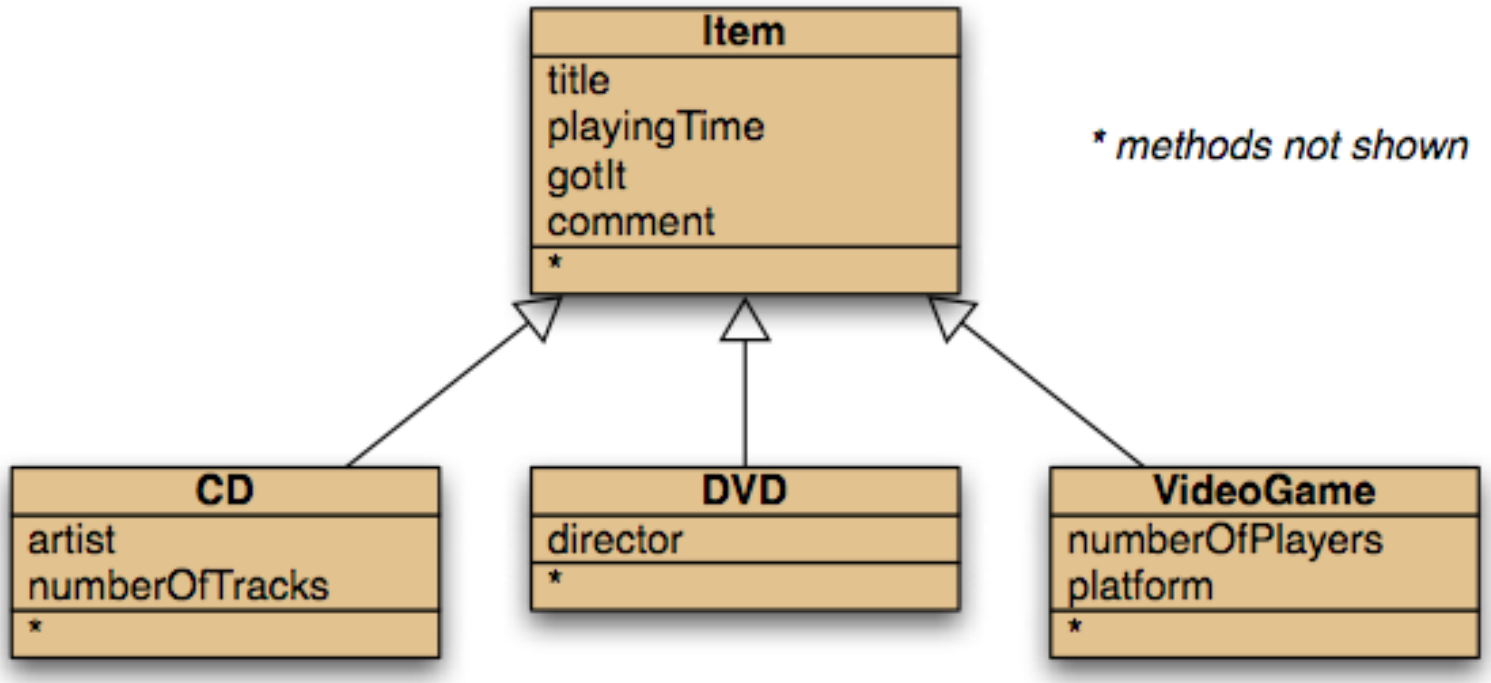
DVD.java (project: dome-v2)

```
...
public DVD(String theTitle,
           String theDirector,
           int time)
{
    super(theTitle, time);
    director = theDirector;
}
...
```

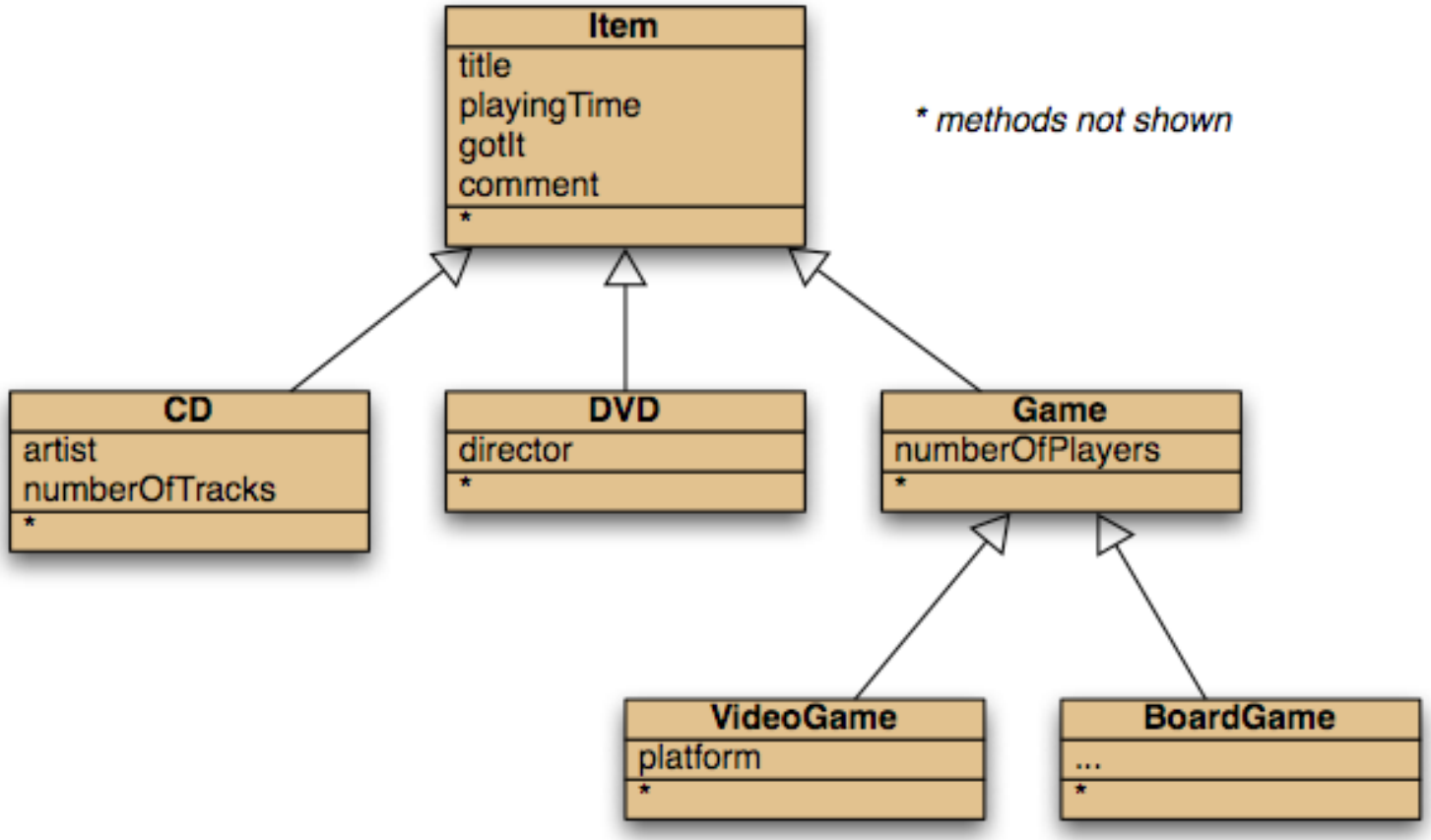
继承与初始化

- 子类的构造器必须调用其超类的构造器。
- 对超类构造器的调用必须是子类构造器的第一条语句。
- 如果子类源代码中没有这样的调用，Java编译器会自动插入一条对超类构造器的调用。
  - 这种自动调用能有效地唯一条件是：超类具有一个没有任何参数的构造器。
- 总的来说，即使编译器可以自动产生一个调用，但通常还是在子类构造器中包含一条显式地对超类构造器的调用，这样可以避免可能产生的错误解读和困惑。

利用继承增加另一种媒体类型



利用继承增加更多的媒体类型



（迄今为止）继承的优点

在继续研究继承的其它特性之前，先来总结一下至今已经看到的优点：

- 避免代码重复
- 代码重用
- 易于维护
- 可扩展性

下一个问题：对Database类的修改

应该如果修改Database类的代码，使其能够适应使用了继承的媒体类？

Database.java (project: dome-v2)

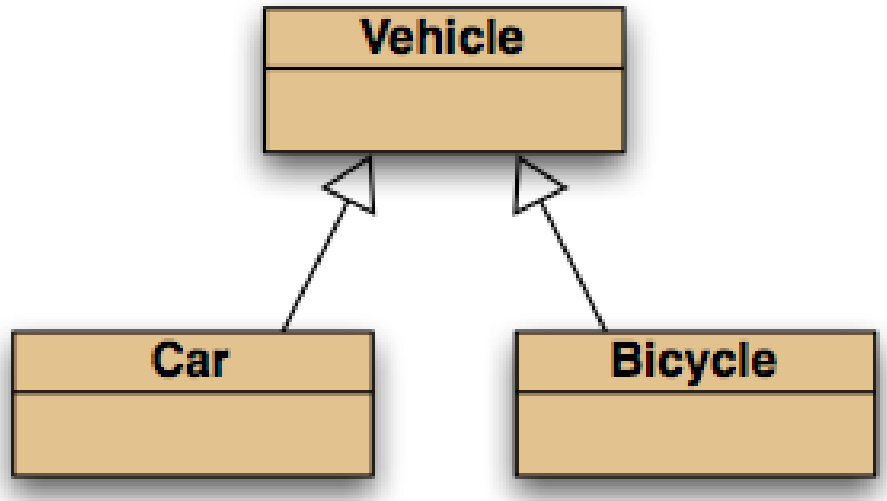
```
public class Database
{
    private ArrayList<Item> items;

    public Database()
    {
        items = new ArrayList<Item>();
    }

    public void addItem(Item theItem)
    {
        items.add(theItem);
    }

    public void list()
    {
        for(Item item : items) {
            item.print();
            System.out.println();
        }
    }

}
```

Database类两个版本之间的变化

- 之所以可以在新的版本中缩短代码，是因为Item类取代了过去CD和DVD两个类型。
  - 在第一个版本中，有如下两个方法向数据库中增加不同的条目。
    - public void addCD(CD theCD)
    - public void addDVD(DVD theDVD)
  - 在新版本中，只需要一个方法就可以实现相同的目的。
    - public void addItem(Item theItem)
- 我们可以采用如下形式调用新版本中的方法。
  - DVD myDVD = new DVD(...);
    database.addItem(myDVD);

子类与子类型

- 类（class）定义了类型（type）
- 子类（class）定义了子类型（subtype）
- 类似于类的层次，类型也构成了类型层次
- 子类型的对象可以被用在需要其父类型对象的场合，这被称为替换（substitution）
- 变量可以保存其声明的类型的对象，或该类型的任何子类型的对象



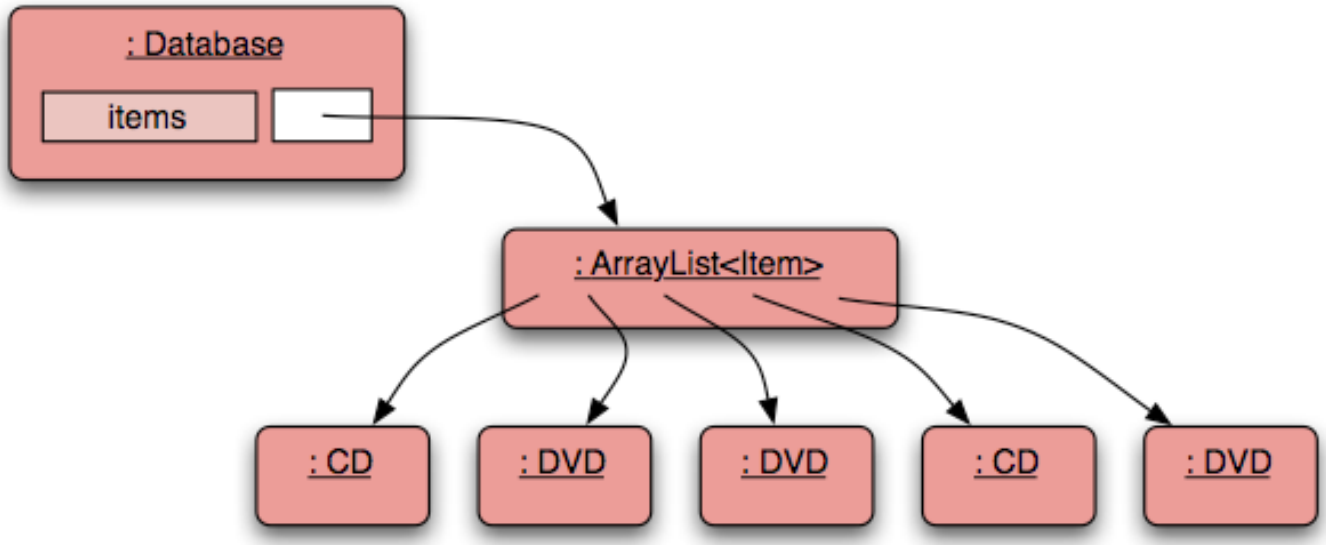

```
Vehicle v1 = new Vehicle(); //ok!
Vehicle v2 = new Car();     //ok!
Vehicle v3 = new Bicycle(); //ok!

Car c1 = new Vehicle();     //error!
Car c1 = new Bicycle();     //error!
```

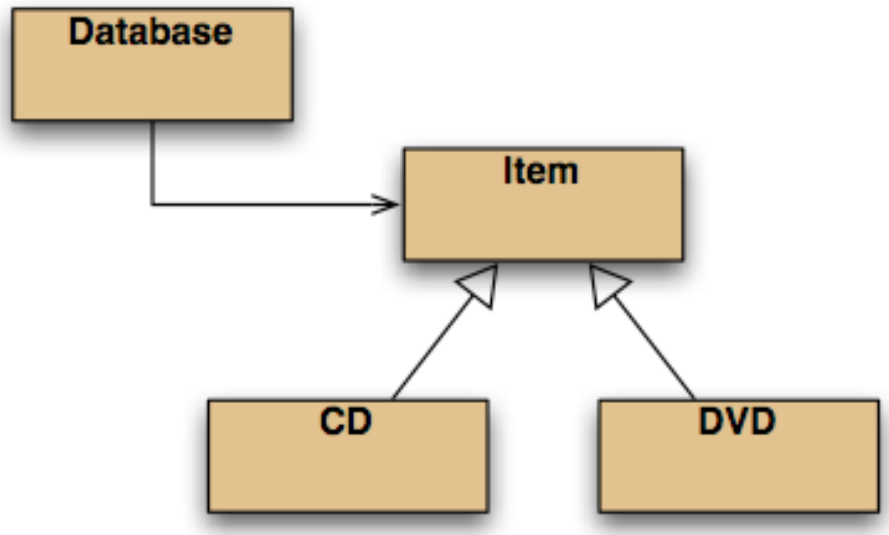
子类型与参数传递

- 传递参数（也就是将一个实际参数赋值给一个形式参数）的过程与赋值给变量的过程完全一致。这就是为什么可以把一个DVD的对象传给一个参数为Item的方法。

```
public class Database
{

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

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

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

子类型与参数传递


: Database
items
: ArrayList<Item>
: CD
: DVD
: DVD
: CD
: DVD


Database
Item
CD
DVD

多态变量

- Java中的对象类型的变量实际上是“多态（polymorphic）变量”。
- “多态变量”指的是一个变量可以保存不同类型（即其声明的类型或任何子类型）的对象。

```
public class Database
{
    ...
    public void list()
    {
       for(Item item : items) {
           item.print();
           System.out.println();
       }
    }
    ...
}
```

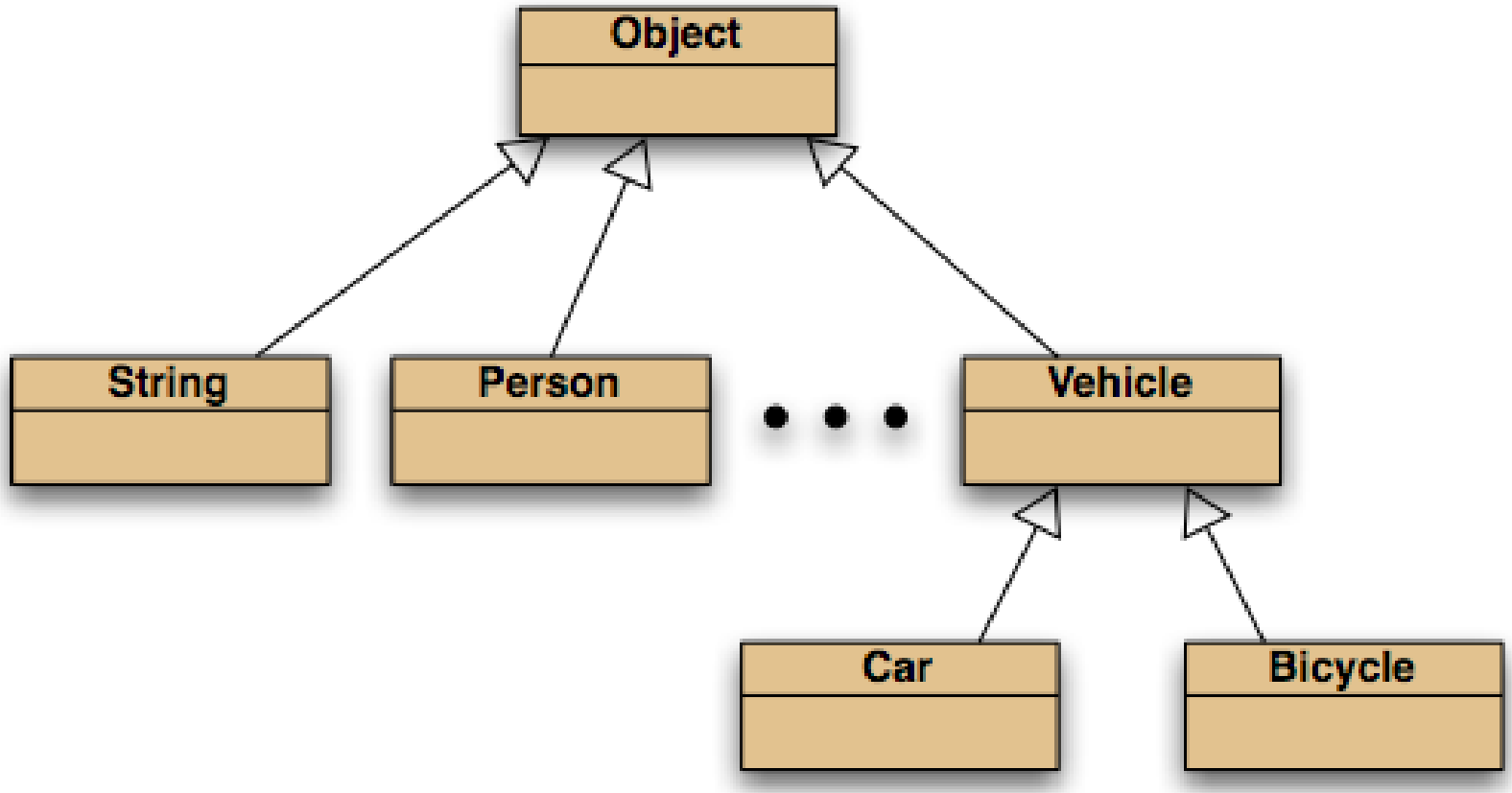
换型

- 在多态变量的机制下，可以把子类型的对象赋值给父类型的变量。
- 要想把父类型的对象赋值给子类型的变量则需要通过换型。
- 在程序设计过程中，换型应尽量避免。

```
Vehicle v;
Car c;
Bicycle b;

c = new Car();
v = c;   //ok!

c = v;   //compile time error! type loss!
c = (Car)v;   //ok!

b = (Bicycle)c;   //compile-time error!
b = (Bicycle)v;   //runtime error!
```

Object类

- Java中所有的类都（直接或间接）继承于Object类，它是Java类层次中的根类。
- 所有没有声明其父类的类都以Object类为其父类。



本章小结

- Java中的所有类都处于一个继承体系中，每个类可以显式地声明其父类，或隐含地从Object类继承。
- 子类通常代表父类的特殊情况，因此继承关系也被称为“is-a”关系。
- 子类从其父类继承得到所有的字段和方法。
- 继承可以使得类结构的设计更灵活。
  - 避免代码重复
  - 代码重用
  - 易于维护
  - 可扩展性
- 子类也形成了子类型，继而形成了多态变量。
  - 子类的对象可以被代换为超类的对象
  - 变量可以保存其声明类型的子类型的实例对象

本章涉及到的概念

- 继承 (inheritance)
- 超类/父类 (superclass/parent class)
- 子类 (subclass/child class)
- 子类型 (subtype)
- is-a关系 (is-a relationship)
- 继承层次 (inheritance hierarchy)
- 抽象类 (abstract class)
- 子类型 (subtype)
- 代换 (substitution)
- 多态变量 (polymorphic variables)
- 类型丢失 (type loss)
- 类型转换/换型 (cast)
- 自动包装 (autoboxing)
- 包裹类 (wrapper class)