

Improving structure with inheritance

Main concepts to be covered

- Inheritance
- Subtyping
- Substitution
- Polymorphic variables

4.1

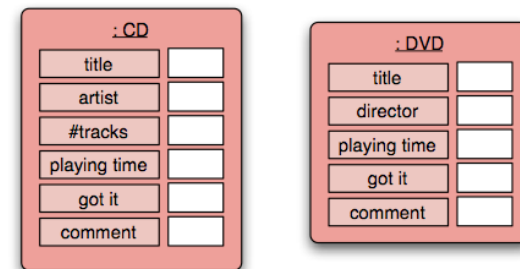
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

The DoME example

DoME objects

"Database of Multimedia Entertainment"

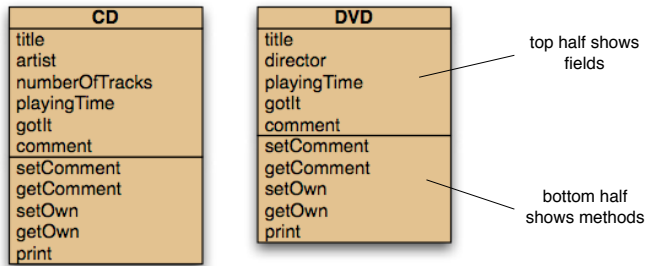
- stores details about CDs and DVDs
 - CD: title, artist, # tracks, playing time, got-it, comment
 - DVD: title, director, playing time, got-it, comment
- allows (later) to search for information or print lists



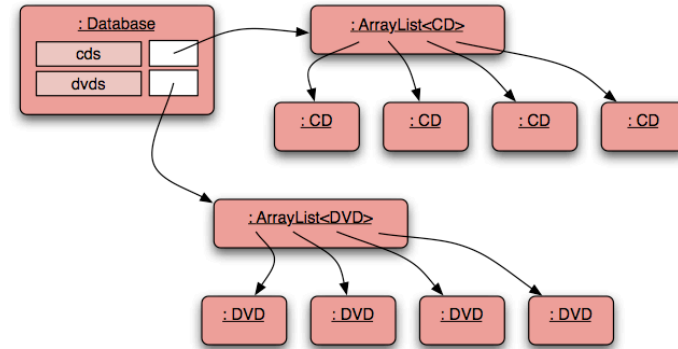
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

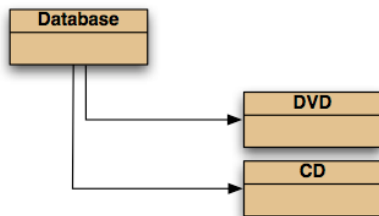
DoME classes



DoME object model



Class diagram



CD source code

**[incomplete
(comments!)]**

```

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

    public CD(String theTitle, String theArtist)
    {
        title = theTitle;
        artist = theArtist;
        comment = " ";
    }

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

    public String getComment()
    { ... }

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

DVD source code

[incomplete
(comments!)]

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

    public DVD(String theTitle, String theDirector)
    {
        title = theTitle;
        director = theDirector;
        comment = " ";
    }

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

    public String getComment()
    { ... }

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

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Database source code

```
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
        }
    }
}
```

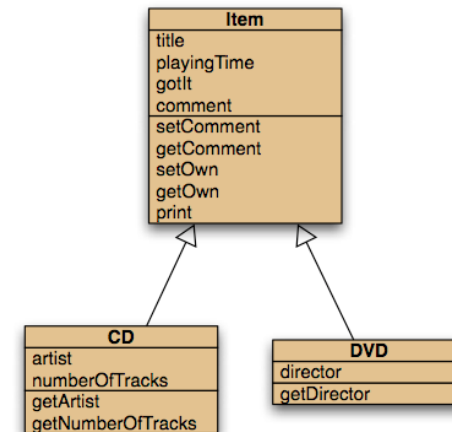
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Critique of DoME

- code duplication
 - CD and DVD classes very similar (large part are identical)
 - makes maintenance difficult/more work
 - introduces danger of bugs through incorrect maintenance
- code duplication also in Database class

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Using inheritance



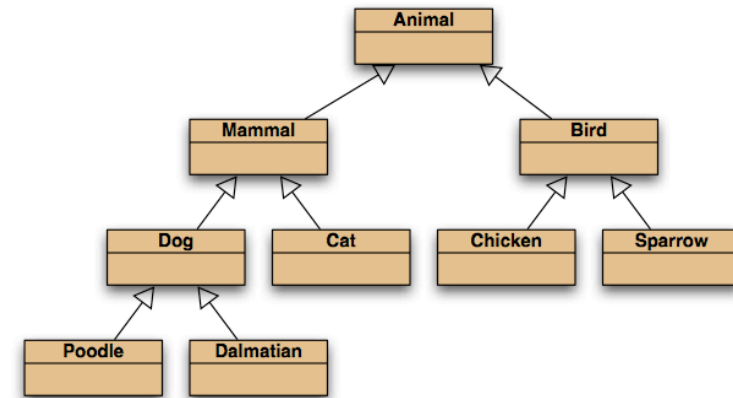
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Using inheritance

- define one **superclass** : Item
- define **subclasses** for DVD and CD
- the superclass defines common attributes
- the subclasses **inherit** the superclass attributes
- the subclasses add own attributes

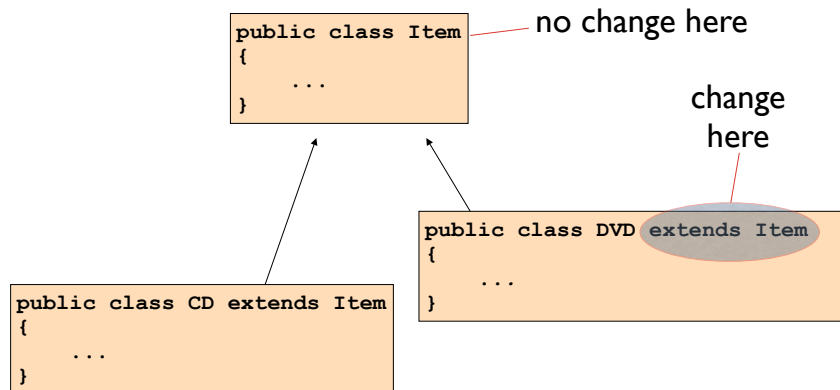
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Inheritance hierarchies



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Inheritance in Java



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Superclass

```
public class Item  
{  
    private String title;  
    private int playingTime;  
    private boolean gotIt;  
    private String comment;  
  
    // constructors and methods omitted.  
}
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Subclasses

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

    // constructors and methods omitted.
}

public class DVD extends Item
{
    private String director;

    // constructors and methods omitted.
}
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Inheritance and constructors

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

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

    // methods omitted
}
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Inheritance and constructors

```
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
}
```

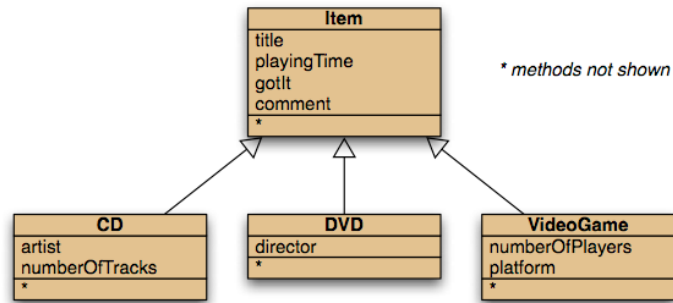
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Superclass constructor call

- Subclass constructors must always contain a 'super' call.
- If none is written, the compiler inserts one (without parameters)
 - works only, if the superclass has a constructor without parameters
- Must be the first statement in the subclass constructor.

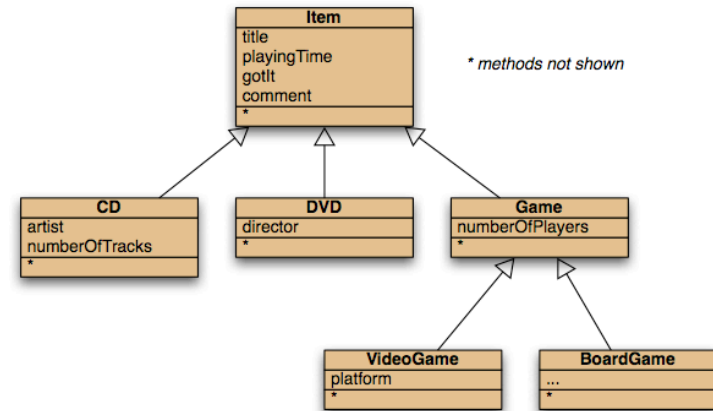
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Adding more item types



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Deeper hierarchies



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Review (so far)

Inheritance (so far) helps with:

- Avoiding code duplication
- Code reuse
- Easier maintenance
- Extendibility

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

```

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);
    }
    ...
}
    
```

New Database source code

avoids code duplication in client!

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

New Database source code

```
/**
 * 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();
    }
}
```

Subtyping

First, we had:

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

Now, we have:

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

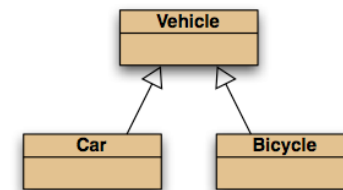
We call this method with:

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

Subclasses and subtyping

- Classes define types.
- Subclasses define subtypes.
- Objects of subclasses can be used where objects of supertypes are required. (This is called **substitution** .)

Subtyping and assignment



subclass objects may be assigned to superclass variables

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

Subtyping and parameter passing

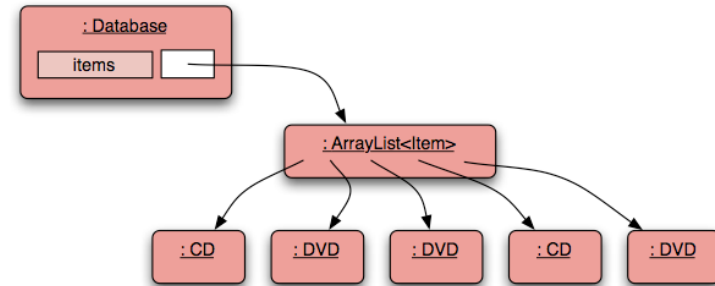
```
public class Database
{
    public void addItem(Item theItem)
    {
        ...
    }
}

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

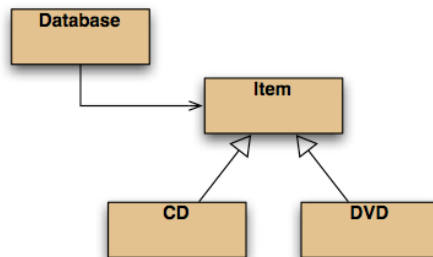
database.addItem(dvd);
database.addItem(cd);
```

*subclass
objects may be
passed to
superclass
parameters*

Object diagram



Class diagram



Polymorphic variables

- Object variables in Java are **polymorphic**.
(They can hold objects of more than one type.)
- They can hold objects of the declared type, or of subtypes of the declared type.

Casting

- Can assign subtype to supertype.
- Cannot assign supertype to subtype!

```
Vehicle v;  
Car c = new Car();  
v = c; // correct;  
c = v; compile-time error!
```

- Casting fixes this:

```
c = (Car) v;
```

(only ok if the vehicle really is a Car!)

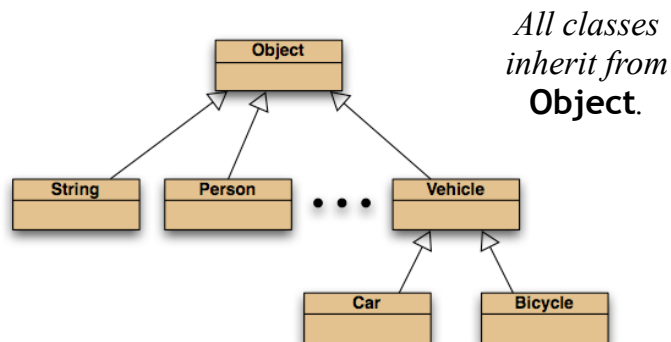
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Casting

- An object type in parentheses.
- Used to overcome 'type loss'.
- The object is not changed in any way.
- A runtime check is made to ensure the object really is of that type:
 - **ClassCastException** if it isn't!
- Use it sparingly.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

The Object class



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Polymorphic collections

- All collections are polymorphic.
- The elements are of type **Object**.

```
public void add(Object element)
```

```
public Object get(int index)
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Collections and primitive types

- All objects can be entered into collections ...
- ... because collections accept elements of type Object ...
- ... and all classes are subtypes of Object.
- Great! But what about simple types?

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Wrapper classes

- Primitive types (int, char, etc) are not objects. They must be wrapped into an object!
- Wrapper classes exist for all simple types:

<i>simple type</i>	<i>wrapper class</i>
int	Integer
float	Float
char	Character
...	...

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Wrapper classes

```
int i = 18;  
Integer iwrap = new Integer(i); ——— wrap the value  
...  
int value = iwrap.intValue(); ——— unwrap it
```

In practice, *autoboxing* and *unboxing* mean we don't often have to do this.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Autoboxing and unboxing

```
private ArrayList<Integer> markList;  
...  
public void storeMark(int mark)  
{  
    markList.add(mark); ——— autoboxing  
}
```

```
int firstMark = markList.remove(0); ——— unboxing
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Review

- Inheritance allows the definition of classes as extensions of other classes.
- Inheritance
 - avoids code duplication
 - allows code reuse
 - simplifies the code
 - simplifies maintenance and extending
- Variables can hold subtype objects.
- Subtypes can be used wherever supertype objects are expected (substitution).