

8 Improving structure with inheritance

Main concepts to be covered

- Inheritance
- Subtyping
- Substitution
- Polymorphic variables

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 2

The DoME example

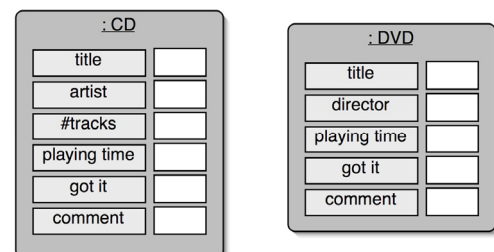
"Database of Multimedia Entertainment"

- 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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 3

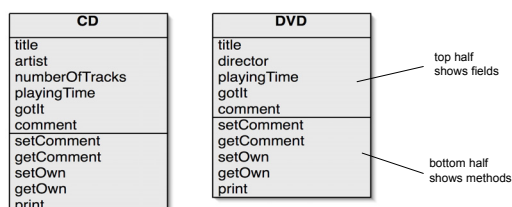
DoME objects



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 4

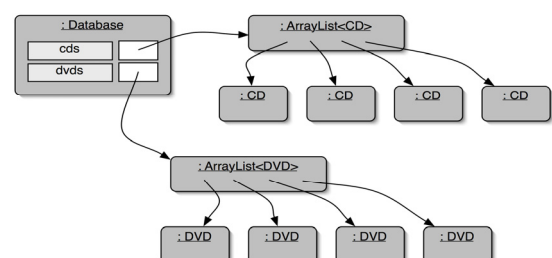
DoME classes



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 5

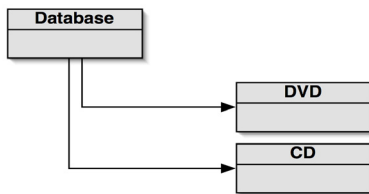
DoME object model



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 6

Class diagram



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 7

CD source code

[incomplete]

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

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

    void setComment(String newComment)
    { ... }

    String getComment()
    { ... }

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 8

DVD source code

[incomplete]

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

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

    void setComment(String newComment)
    { ... }

    String getComment()
    { ... }

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 9

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
        }
    }
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 10

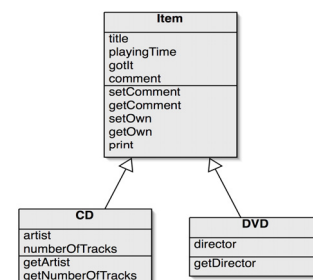
Critique of DoME

- 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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 11

Using inheritance



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 12

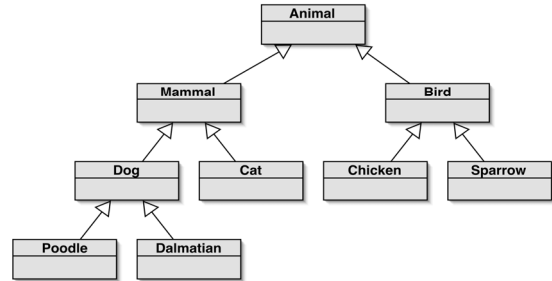
Using inheritance

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 13

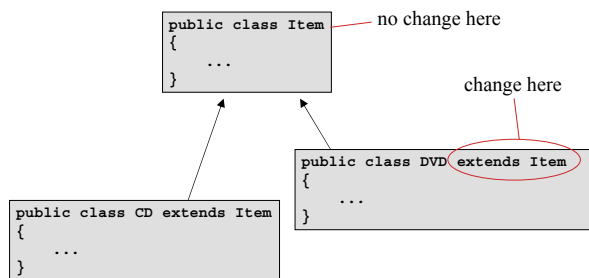
Inheritance hierarchies



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 14

Inheritance in Java



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 15

Superclass

```

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

    // constructors and methods omitted.
}
    
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 16

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.
}
    
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 17

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
}
    
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 18

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
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 19

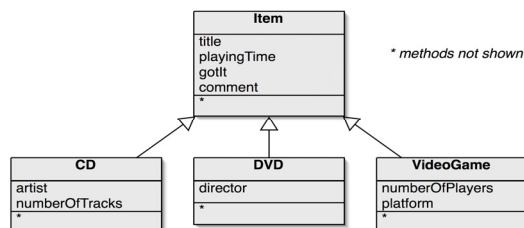
Superclass constructor call

- 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.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 20

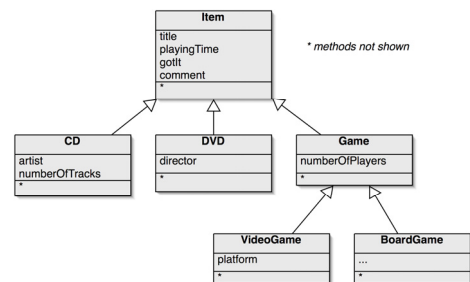
Adding more item types



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 21

Deeper hierarchies



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 22

Review (so far)

Inheritance (so far) helps with:

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 23

```
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!*

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 24

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();
    }
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 25

Subtyping

First, we had:

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

Now, we have:

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

We call this method with:

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 26

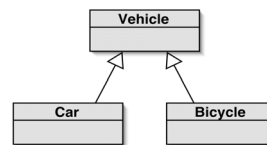
Subclasses and subtyping

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 27

Subtyping and assignment



*subclass objects
may be assigned
to superclass
variables*

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 28

Subtyping and parameter passing

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

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

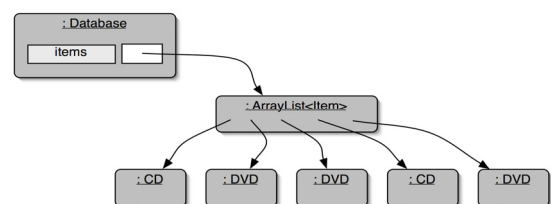
database.addItem(dvd);
database.addItem(cd);
```

*subclass objects
may be passed to
superclass
parameters*

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 29

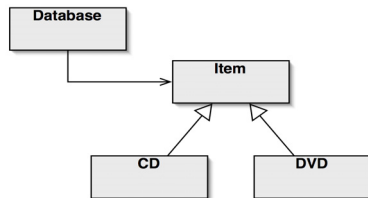
Object diagram



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 30

Class diagram



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 31

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.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 32

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!)

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 33

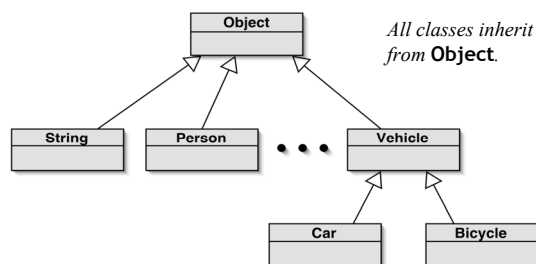
Casting

- 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.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 34

The Object class



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 35

Side Note: Polymorphic collections

- Previous Java versions used **polymorphic collections** instead of **generic collections**.
- The elements are of type **Object**.

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

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

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 36

Polymorphic collections (2)

- All objects can be entered into collections ...
- ... because collections accept elements of type `Object` ...
- ... and all classes are subtypes of `Object`.
- Great! So a list can contain elements of many types (*heterogeneous collection*).

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 37

Polymorphic collections (3)

Ex. Watch out for type cast errors!

```
ArrayList box = new ArrayList();  
Integer banana = new Integer(37);  
box.add("apple");  
box.add(banana);
```

Type cast `Object` to `String`



```
String fruit = (String)box.get(0);
```



```
fruit = (String)box.get(1);
```

Oops! The next element is an `Integer`

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 38

Polymorphic collections (4)

- Polymorphic collections means *weaker* type checking.
- Type checking that could be done at **compile time** is done at **runtime**.
- Use *generic collections* instead!

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 39

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 extension
- Variables can hold subtype objects.
- Subtypes can be used wherever supertype objects are expected (substitution).

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 8 40