Objects First With Java
A Practical Introduction Using BlueJ

Att förbättra strukturen med hjälp av arv

Nya begrepp

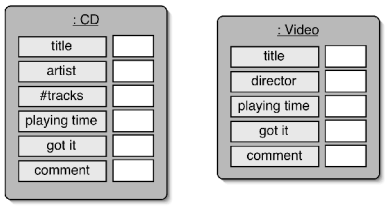
- Arv (eng. inheritance)
- Subtyper
- Substitution
- Polymorfiska variabler

DoME

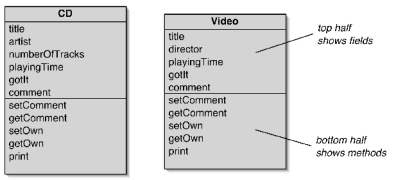
"Database of Multimedia Entertainment"

- Lagrar detaljer om CD-skivor och videoband
  - CD: title, artist, # tracks, playing time, got-it, comment
  - Video: title, director, playing time, got-it, comment
- Lagrade listor kan skrivas ut.

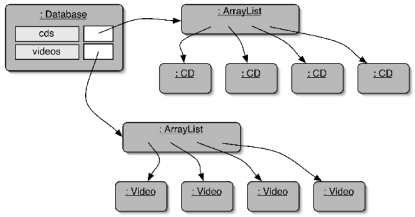
DoME-objekt



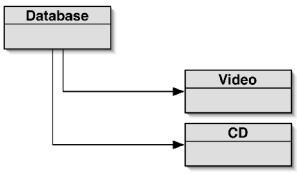
DoME-klasser



DoME objektdiagram



Klassdiagram



CD -källkod

```
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()
   { ... }
   ...
}
```

Video -källkod

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

   Video(String theTitle, String theDirect)
   {
       title = theTitle;
       director = theDirect;
       comment = " ";
   }

   void setComment(String newComment)
   { ... }

   String getComment()
   { ... }

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

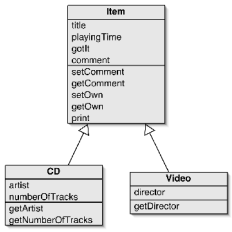

Databas - källkod

```
class Database {

   private ArrayList cds;
   private ArrayList videos;
   ...

   public void list()
   {
      for(Iterator iter = cds.iterator(); iter.hasNext(); ) {
          CD cd = (CD)iter.next();
          cd.print();
          System.out.println();   // empty line between items
       }

      for(Iterator iter = videos.iterator(); iter.hasNext(); ) {
          Video video = (Video)iter.next();
          video.print();
          System.out.println();   // empty line between items
       }
   }
}
```


Kritik av DoME

- koduppprepning
  - CD och Video klasserna är väldigt lika (stora delar är identiska)
  - försvårar underhåll - mer arbete
  - ökar sannolikheten för fel när koden uppdateras eller ändras
- även i Database-klassen finns det kod som upprepas

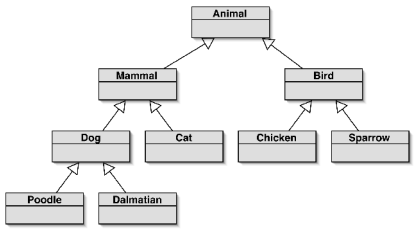
Användning av arv



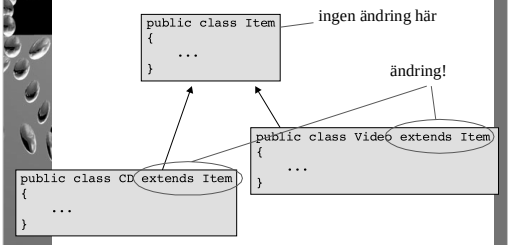
Arv

- definiera en superklass : Item
- definiera subklasser för Video och CD
- superklassen definierar gemensamma attributes
- subklasserna ärver superklassens attribut
- subklasserna har också ha egna attribut

Ärvnings hierarkier



Arv i Java



Superklass

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

    // constructors and methods omitted.
}
```

Subklasser

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

    // constructors and methods omitted.
}
```

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

    // constructors and methods omitted.
}
```


Arv och konstruktorer

```
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
}
```

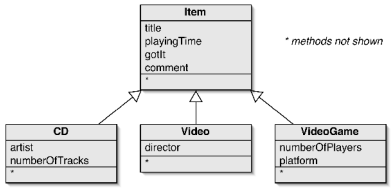

Arv och konstruktorer

```
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
}
```

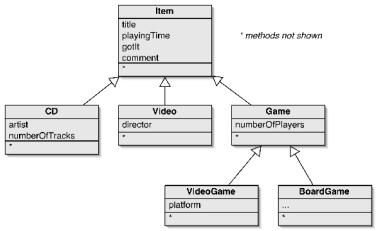

Anrop av superklassens konstruktor

- Konstruktorn i en subklass bör alltid innehålla ett 'super'-anrop.
- Om det inte finns något, kommer kompilatorn att sätta in ett (utan parameterar)
  - fungerar bara om superklassen har en konstruktor utan parameterar
  - Obs: super måste alltid vara den första satsen i subklassens konstruktor

Vi kan nu lägga till fler subklasser till Item



Djupare hierarkier



Summering (så långt)

Bra saker med arv:

- Man undviker koduppprepning
- Kod kan återanvändas lättare
- Enklare underhåll
- Enklare utvidgning

Ny källkod för Database

```
public class Database
{
    private ArrayList items;

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

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

undviker koduppprepning!

Ny källkod för Database

```
/**
 * Print a list of all currently stored CDs and
 * videos to the text terminal.
 */
public void list()
{
    for(Iterator iter=items.iterator(); iter.hasNext(); )
    {
        Item item = (Item)iter.next();
        item.print();
        System.out.println(); // empty line between items
    }
}
```


Subtyper

Tidigare hade vi:

```
public void addCD(CD theCD)
public void addVideo(Video theVideo)
```

Nu har vi:

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

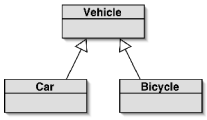
Anrop av den nya metoden:

```
Video myVideo = new Video(...);
database.addItem(myVideo);
```


Subklasser och subtyper

- Klasser definierar typer.
- Subklasser definierar subtyper.
- Objekts av en subklass kan användas i de fall objekt av superklassen kan användas.
  (Kallas substitution.)

Subtyper och tilldelning



objekt av en subklass kan tilldelas variabler av dess superklass.

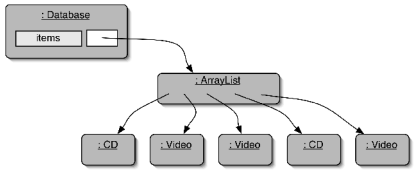
```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Subtyper och parameteröverföring

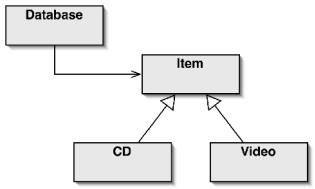
```
public class Database
{

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

Video video = new Video(...);
CD cd = new CD(...);

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

objekt av en subklass kan överföras till parametrar av dess superklass.

Objektdiagram



Klassdiagram



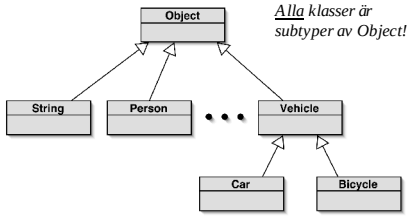
Polymorfiska variabler

- Variabler av typ Object i Java är polymorfiska.

  (Kan referera objekt av många typer.)

- De kan referera objekt av den deklarerade typen och av subtyper till den deklarerade typen.

Klassen Object



Alla klasser är subtyper av Object!

Polymorfiska kollektioner

- Alla kollektioner är polymorfiska.
- Alla elementen är av typ Object.

```
public void add(Object element)

public Object get(int index)
```


Typomformning (casting) igen

- Man kan tilldela en subtyp till en supertyp.
- Man kan inte tilldela en supertyp till subtyp!

```
String s1 = myList.get(1);
```
fel!

- Typomformning (type casting) fixar detta:

```
String s1 = (String) myList.get(1);
```

(men fungerar bara om elementet verkligen är av typ String!)

Omslagsklasser (Wrapper classes)

- Alla typer av objekt kan sättas in i en kollektion ...
- ... därför att kollektioner förutsätter element av typen Object...
- ... och alla klasser är subtyper av Object.
- Bra! Men om vi vill arbeta med primitiva typer?

Omslagsklasser

- Primitiva typer (int, char, etc) är inte objekt. De måste få ett omslag så att de blir ett objekt!
- Oslagsklasser finns för alla primitiva typer:

primitiva typer	omslagsklass
int	Integer
float	Float
char	Character
...	...

Omslagsklasser

Skapa ett omslag för int värdet

```
int i = 18;
Integer iwrap = new Integer(i);

myCollecton.add(iwrap);
...
```

sätt in omslaget i kollektionen

```
Integer element = (Integer) myCollection.get(0);
int value = element.intValue()
```

ta ut omslaget

ta ut värdet

Summering

- Nya klasser kan definieras som utvidgningar av gamla klasser vars attribut och egenskaper ärvs.
- Arv
  - gör att man undviker koduppprepning
  - underlättar återanvändning av kod
  - förenklar koden
  - förenklar underhåll och utbyggnad
- Vraiabler kan referera objekt som är subtyper till variabelns typ.
- Subtyper kan användas överallt där en supertyp kan användas (substitution).