



## Programmering för språkteknologer I

Markus Saers  
markus.saers@lingfil.uu.se  
9-2040  
stp.ling.uu.se/~markuss/vt09/pst1



## Innehåll

Repetition  
Samlingar  
Associativa samlingar  
`java.util.Scanner`  
Kapitel 12+

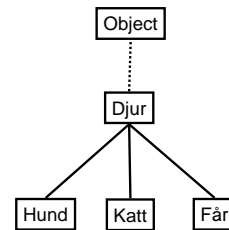


## Repetition

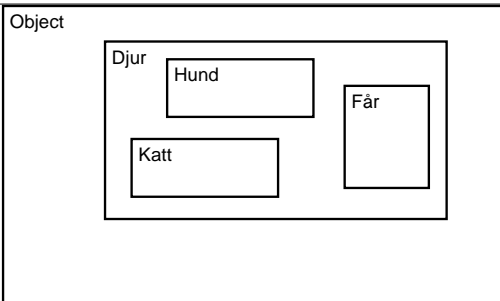
Arvshierarkier  
Förhållandet mellan klasser  
Instansiering  
Förhållandet mellan klasser och objekt  
Synlighet för variabler och metoder  
Synlighet för medlemmar



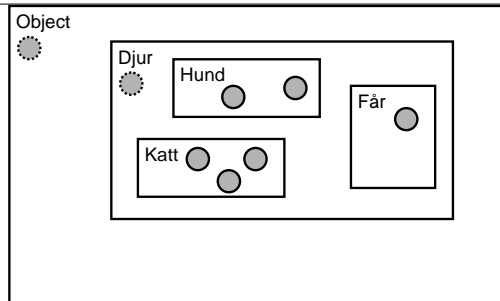
## Arvshierarki

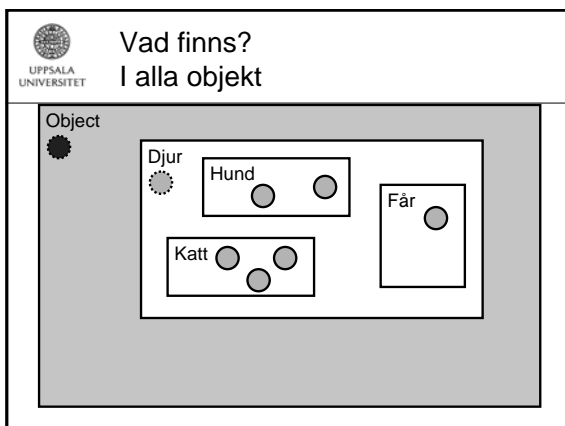
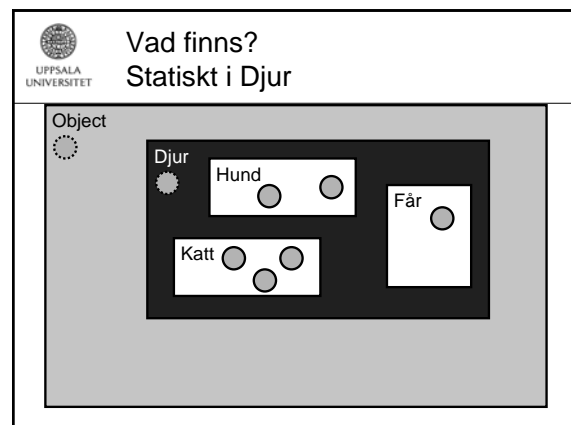
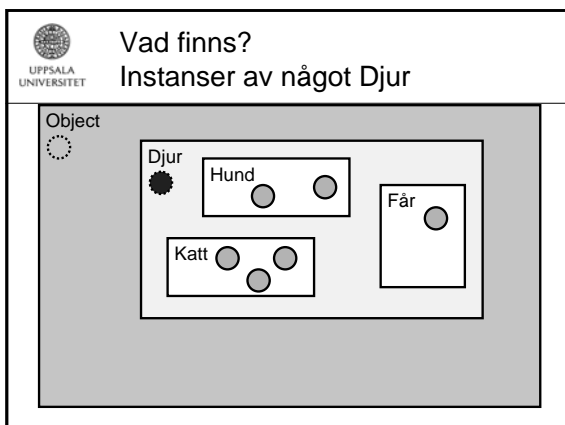
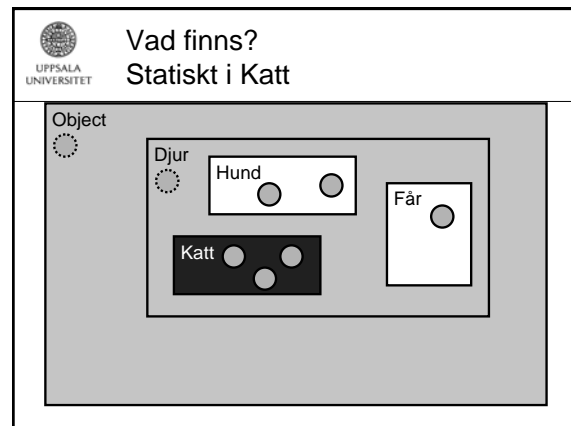
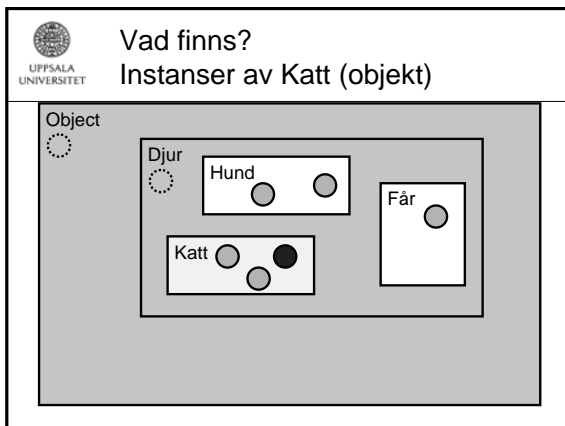


## Uppifrån



## Instansiering





UPPSALA UNIVERSITET

### Publika medlemmar

Nås via ett objekt  
`s.draw(w);`

Statiska nås via klassnamnet  
`Integer.parseInt("12");`  
`Math.PI;`  
`Color.BLACK;`

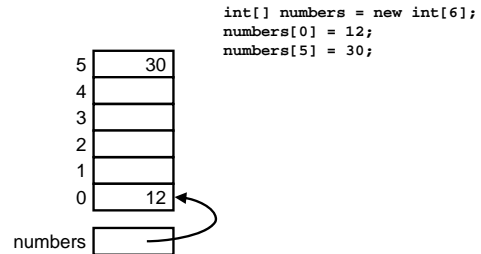


## Samlingar

Innehåller många objekt av en viss typ  
Vi har sett vektorer  
Inbyggd konstruktion i Javaspråket



## Vad är en vektor?



## För- / Nackdelar?

### Fördelar

- Inbyggt i språket
- Väldigt snabbt

### Nackdelar

- Förbestämd storlek
- Kan inte associera annat än index med innehållet
  - Hur bygger man t.ex. ett lexikon?
- Jobbigt att sätta in element mitt i



## Andra samlingar?

Mycket av datorvetenskapen har ägnats åt det här problemet

Ganska stort utbud av standardlösningar finns i `java.util`



## Collection

Gränssnitt i `java.util` som används av alla samlingar

Innehåller grundläggande funktionalitet

- `add`
- `clear`
- `contains`
- `isEmpty`
- `remove`
- `size`

Implementerar gränssnittet `Iterable`



## Samlingstyper

Flera olika typer av samlingar för olika ändamål

- List
- Set
- Queue
- Deque

Ett gränssnitt för varje



## Konkreta samlingar

**ArrayDeque**  
**ArrayList, LinkedList, Vector**  
**HashSet, TreeSet**  
**Stack**



## "Generics"

Används för att specificera vad en samling innehåller  
`Vector<Integer> list = new Vector<Integer>();`  
 Det specificerade måste vara en klass!  
 Fungerar inte med inbyggda typer  
 int, long, short, float, double, boolean  
 Använd respektive klass istället  
 Integer, Long, Short, Float, Double, Boolean



## Associativa samlingar

Låter oss associera två godtyckliga objekt, en nyckel och ett värde  
 Värdet kan hämtas ut med nyckeln  
 Bra för lexikon  
 Associera ett ord med uppslagsinformationen  
 Ni kommer att behöva ett frekvenslexikon för inlämningsuppgiften



## Associativa samlingar: Map

Låter oss "mappa" objekt av en typ till objekt av en annan typ  
 Specificeras med "generics"  
 Nycklar och värden  
 Två konkreta klasser  
**HashMap**  
**TreeMap**



## Gränssnittet **Iterator** och **Iterable**

**Iterator** låter oss löpa igenom en samling  
**Iterator** har två intressanta metoder  
 next  
 hasNext  
 Metoden **next** ger tillbaka ett objekt av den typ som anges med "generic"  
 Gränssnittet **Iterable** garanterar att objekten har en metod som heter **iterator** som get tillbaka en **Iterator**



## Exempel på skapande och genomlöpning av samling

```
import java.util.*;

public class CollectionTest {
    public static void main(String[] args) {
        List<Integer> l = new ArrayList<Integer>();

        for (int i = 0; i < 10; i++) {
            l.add(i);
        }

        Iterator<Integer> i = l.iterator();
        while (i.hasNext()) {
            Integer n = i.next();
            System.out.println(n);
        }
    }
}
```



## Enklare genomlöping i Java 1.5+

```
import java.util.*;

public class CollectionTest {
    public static void main(String[] args) {

        List<Integer> l = new ArrayList<Integer>();

        for (int i = 0; i < 10; i++) {
            l.add(i);
        }

        for (Integer n : l) {
            System.out.println(n);
        }
    }
}
```



## Grundläggande inläsning

Inläsning = att läsa information från någon källa

`java.util.Scanner` kan läsa in en serie med blankstegsseparatorerade **String**

Implementerar `Iterator<String>`

Har metoderna  
`boolean hasNext()`  
`String next()`

Låter oss läsa in strängar från en källa



## Vad är en "källa"?

Kan vara många saker

Standard input

Fil

Nätverkskontakt

Sträng

`Scanner` har ett flertal konstruktörer för olika källor



## Exempel

```
import java.util.Scanner;

public class EchoToken {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while(scanner.hasNext()) {
            String s = scanner.next();
            System.out.println(s);
        }
    }
}
```



## Exempel med samling

```
import java.util.*;

public class EchoToken {
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        Scanner scanner = new Scanner(System.in);
        while(scanner.hasNext()) {
            list.add(scanner.next());
        }
        for (String s : list) {
            System.out.println(s);
        }
    }
}
```



## Exempel med samling och metoder

```
import java.util.*;

public class EchoToken {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<String> list = read(scanner);
        write(list);
    }
    public static List<String> read(Scanner s) {
        List<String> list = new ArrayList<String>();
        while(s.hasNext()) {
            list.add(s.next());
        }
        return list;
    }
    public static void write(List<String> l) {
        for (String s : l) {
            System.out.println(s);
        }
    }
}
```



UPPSALA  
UNIVERSITET

## Exempel med samling och objekt

```
import java.util.*;

public class EchoToken {
    private Scanner scanner;
    private List<String> list;
    public EchoToken() {
        scanner = new Scanner(System.in);
        list = new ArrayList<String>();
    }
    public void read() {
        while(scanner.hasNext()) {
            list.add(scanner.next());
        }
    }
    public void write() {
        for (String s : list) {
            System.out.println(s);
        }
    }
    public void echo() {
        read();
        write();
    }
    public static void main(String[] args) {
        EchoToken e = new EchoToken();
        e.echo();
    }
}
```