Introduktion

- ***Kursens innehåll***

- ***Repetition***
  -Klasser, metoder, objekt och referensvariabler,
  -Arv och generics types
  -Arrayer och arrayer av objekt .
  -Collection ramverket
  -Iteratorer

Kap 1-4 och 6.1-6.3

Vad handlar kursen om?

Mål: **Att ge fördjupad kunskap om metoder och hjälpmedel för analys och utformning av lösningar av programmeringsproblem.**
Detta omfattar :

- Datastruktur- en samling av variabler av någon typ som hör ihop på något sätt och löser ett vist typ av problem på bästa sättet.
  Ex: arrayer, listor, stack, träd, hash tabel

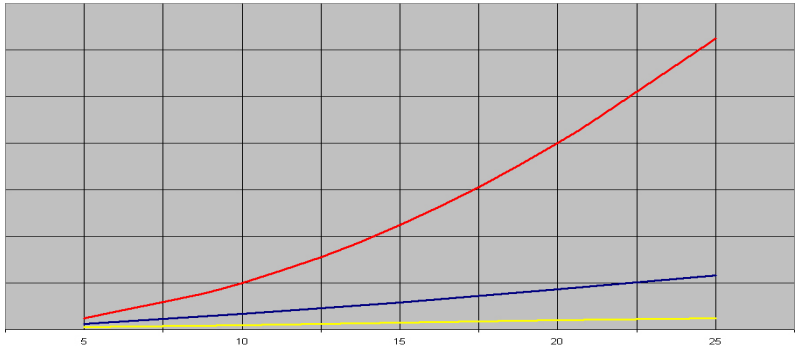
- Kända tillämpningar
  Ex: beräknemaskinner, kompilatorer, lexikon, simulatorer

Vad handlar kursen om?

- Algoritm - metod för att stegvis lösa ett problem

  Ex. Algoritm för effektiv sökning och sortering av data.
- Algoritm analys, tidskomplexitet- hur olika implementationer påverkar exekveringstiden för en algoritm

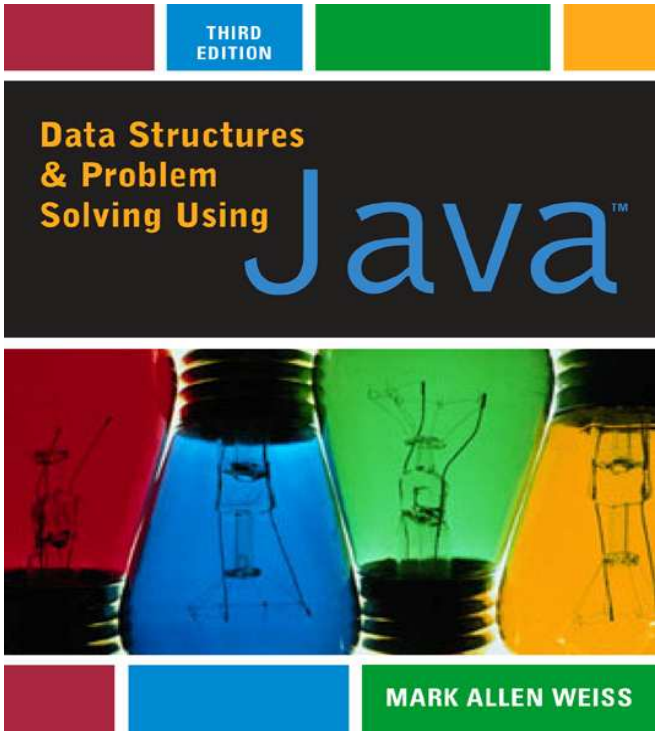
T(n)



n

Exekveringstiden representeras som funktioner

Kurslitteratur och webbsida



Webbsida : http://www.hh.se/db4002

Objekt-Orienterad Programmering

- Klasser är ”templates” som beskriver hur objekt ska designas i termer av egenskap, metoder och konstruktor.
- Objekt är *instance* av klassen, dvs konkret

- ***name***
- ***age***
- ***other***

-***kan_sing()***

-***kan_talk()***

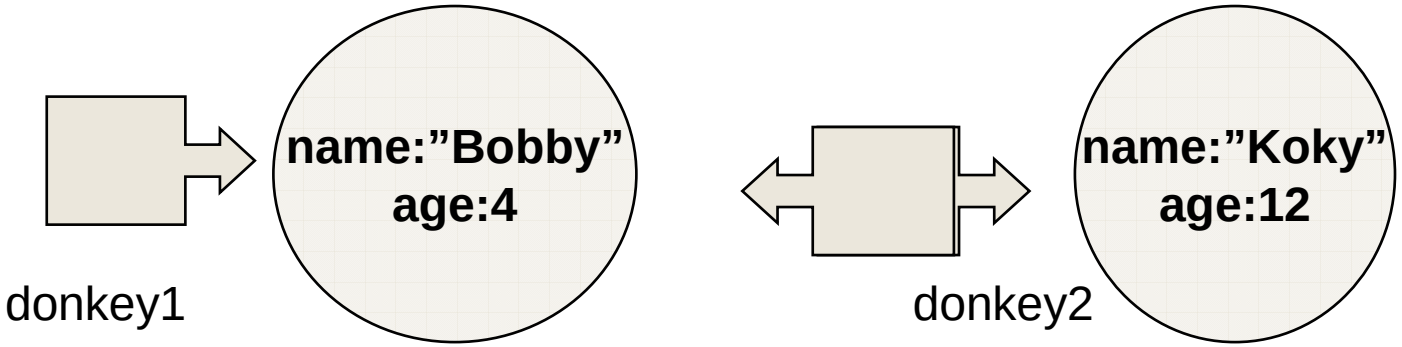
new

name: Bobby
age: 12
---
kan_sing()
kan_talk()

new

Name: Koky
age: 3
---
kan_sing()
kan_talk()

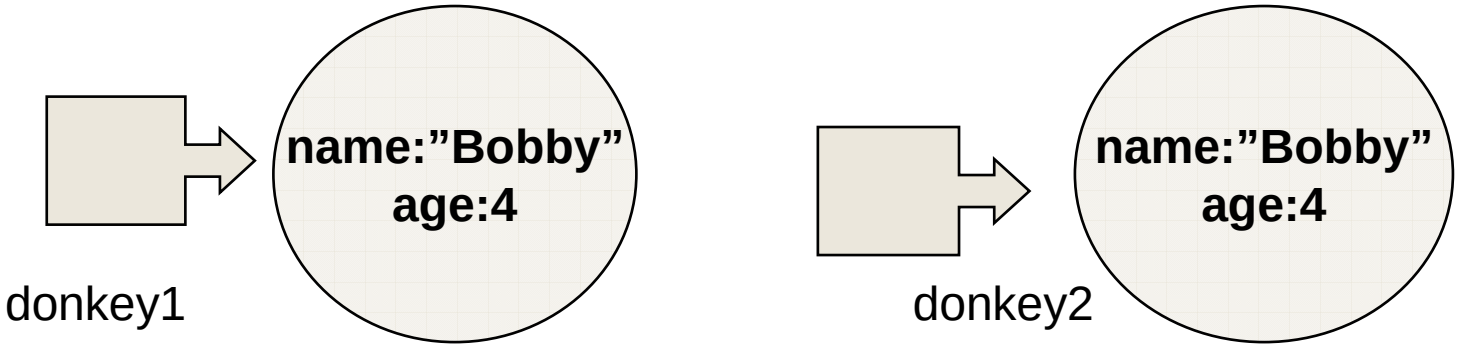
Objekt och referenser



adress	”innehåll”
1000	”Bobby” , 4
1024	”Koky”, 12
3200	donkey1->1000
3600	donkey2->1024 1000

Vad händer om man gör : donkey2=donkey1?

Objekt och referenser



adress	”innehåll”
1000	”Bobby” , 4
1024	”Koky”, 12
3200	donkey1->1000
3600	donkey2->1024

Vad blir if (donkey2==donkey1)?

Att jämföra objekt för ”equality”

Om två objekt ska kunna jämföras måste de ha en metod som ”gör detta”.

Klassen Object

```
class Program {
  public static void main(String[] args)
{
  String s1= ”HBK är bäst” ;
  String s2=”Java är bäst”;
  System.out.println(s1.equals(s2));
}
}
```

```
class Donkey{
// data och metoder
public boolean equals ( Donkey other)
{


  ??????????????????????

}
```

Objektorienterad mekanismen

- Skapar återanvänbara hierarkier genom följande :

  - Inheritance(Arv)
    En klass kan ärva från en annan och utvidga dennes funktionalitet
  - Interfaces
    En eller flera tomma metoder som måste inplementeras av klasserna som implementerar interfacen.

Skapa supertyper genom arv

```
class Person
{
    private String name;
    ..............................

    public Person( String n){
        name=n;
        }

    public String toString(){
    return ”The name is ”+ name;  }
```

Super klass och subklass, this och super

class Student extends Person{

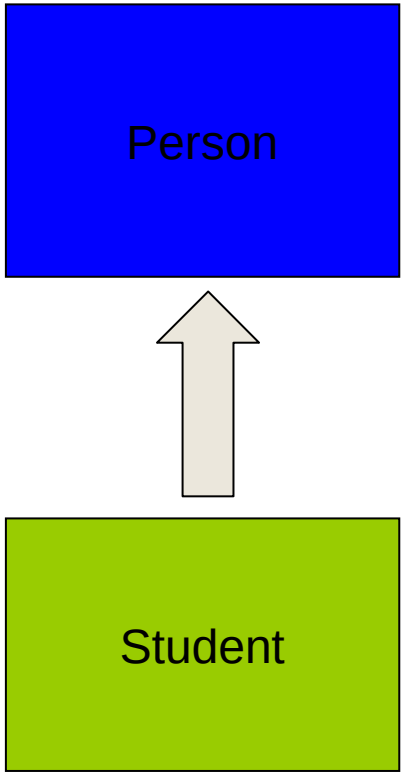
```
private int credits;
```

```
public Student (int c, String n){
        super(n);
        credits=c;
}
```

```
public String toString( ){
return super.toString()+ ” ” + credits
 }
```

```
public void addCredits(int newcredits)
    {   this.credits+=credits;}
```

Super typ/ Sub typ

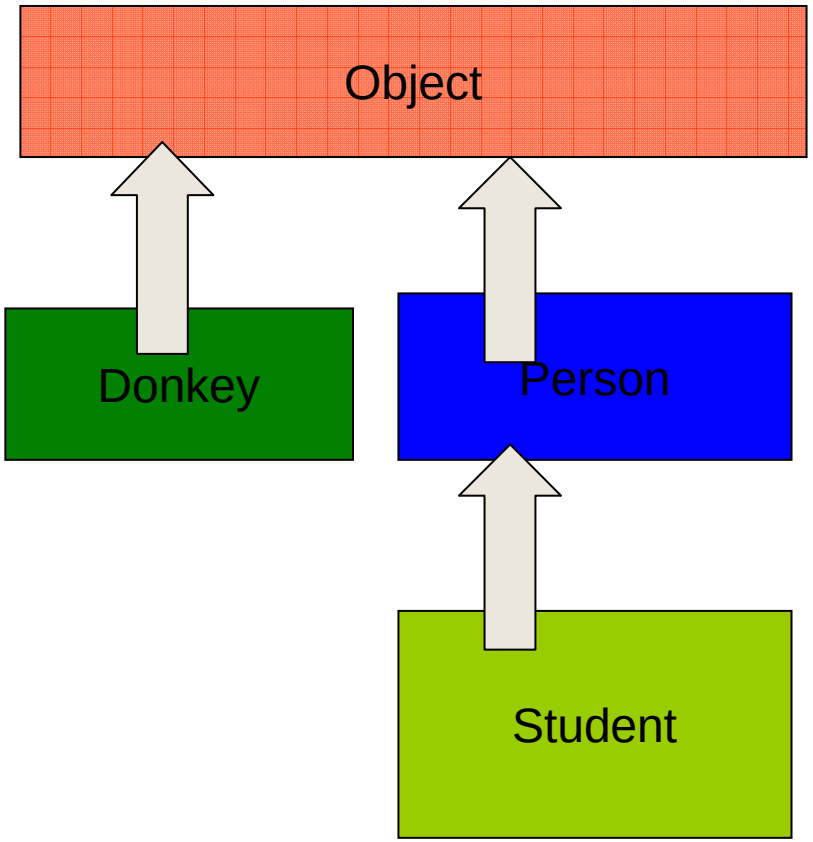


Person p1=new Person (” Kale Svensson”);

Student s1= new Student (0, ” Tina Kaos”);

Person p2= new Student (0, ” Tina Kaos”);

Datatypen Object är en ”generell typ”



Vad blir utskriften ?
System.out.println(o1.toString());
System.out.println(o2.toString());

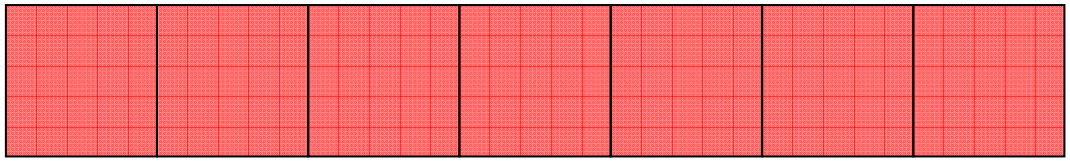
Object o1= new Person (”Kalle Svensson”);

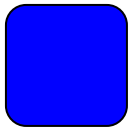
Object o2= new Student (0, ”Anders T”);

Object o3= new Donkeyt (” Bobby”, 23);

Array av datatypen Object

Object [ ] array=new Object [7]





array[1]=new Person(..)

array[2]=new Random(...)

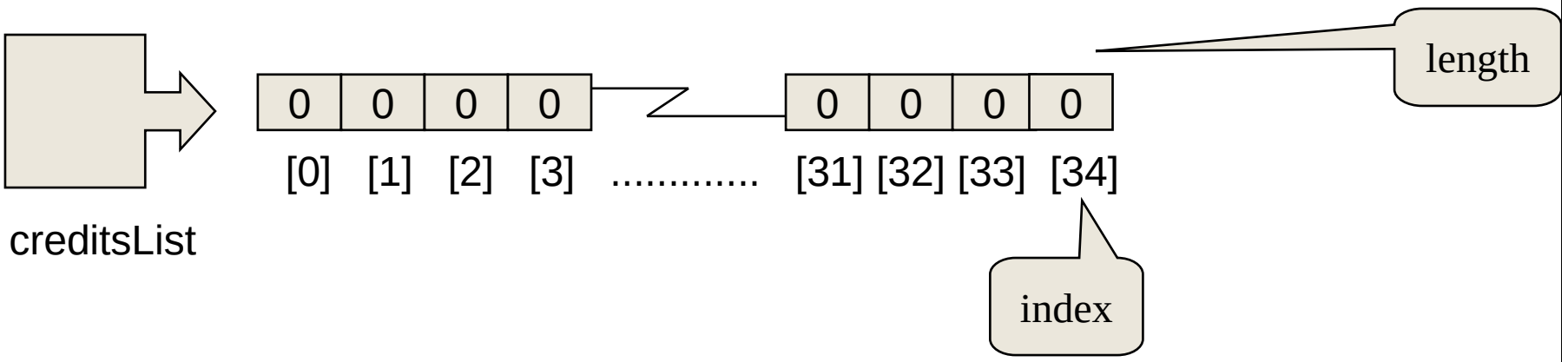
Datastrukturen array. Varför?

- Liknar den mest grundläggande databasen. Databas med en tabell.
- Data i arrayen skall vara av samma typ.
- När? Alltid mycket information skall lagras och manipuleras.
- Data i arrayen är lätt att accessa

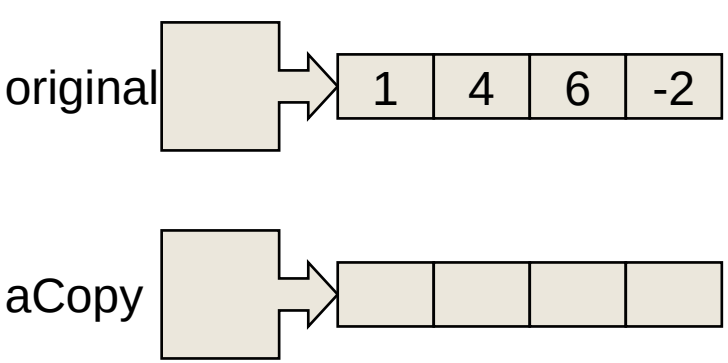
Datastrukturen array.

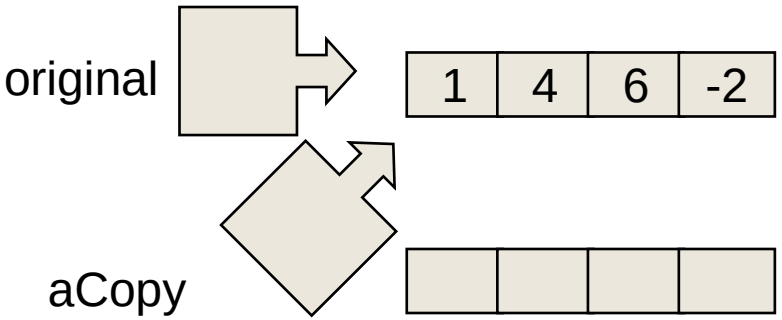
- Vi deklarerar och skapar array objekt

  int[ ] creditsList= new int[35];



Datastrukturen array






```
int[] original = {1, 4, 6, -2};
int[] aCopy = new int[original.lenght];
```

```
Efter tilldelning:
aCopy = original;
```

- Detta fungerar inte, vi får två referenser till samma aray

Arrayer som argument till metod

```
class  A
{
public static void  förvirra( int [ ] a)
{
  a[3]=7;
  a[4]=10;

for(int i=0;i<a.length;i++)

          System.out.println(a[i]);

}

}
```

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

  int [ ] array1={5,6,7,8,3,8,5};
  A.förvirra(array1);

for(int i=0;i<array1.length;i++)
        System.out.println(array[i]);

}
}
```

Interface Comparable

- I java biblioteket finns interfacen Comparable som används för att definiera objekt som kan jämföras och rangordnas

```
public interface Comparable
{
  public int compareTo( Object other);
}
```

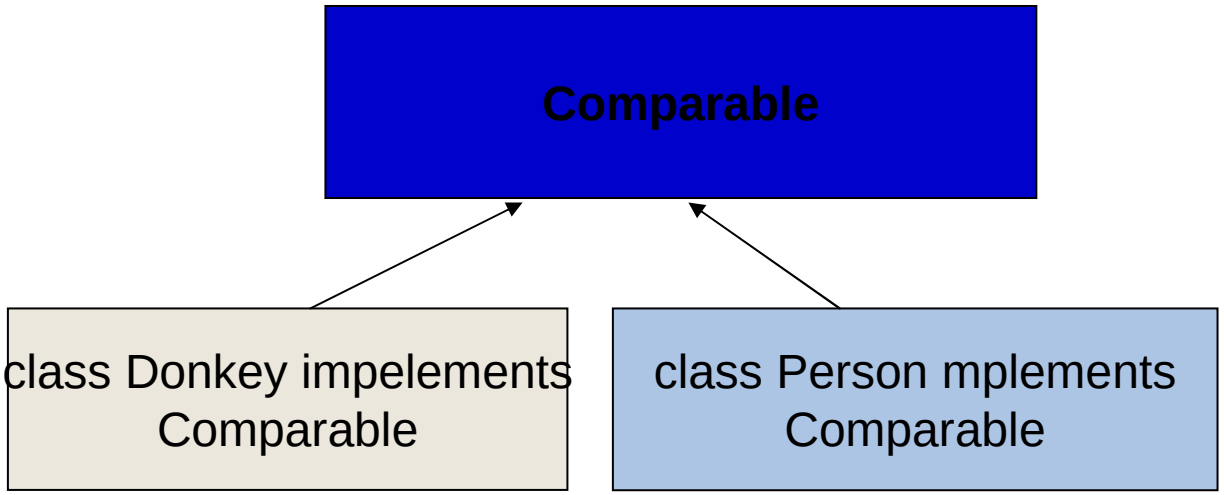
Interface och datatyp

```
import java.util.*;
class Person implements Comparable
{

 // andra metoder
public int compareTo(Object obj)
{
  Person p=(Person) obj;  // först casting från Object->Person


}
}
```

Interfacens namn blir datatyp, en superdatatyp




```
Comparable obj1=new Donkey(???);
Comparable obj2=new Person(????);
```

Klassen Object och Interface – är javas sätt att implementera ”generella” typer men....

```
class MyMethods
{

 public static void  förvirra1( Object  obj)
{
   System.out.println(obj);
}

public static void förvira2( Comparable obj1, Comparable obj2)
{
    System.out.println(obj1.compareTo(obj2));
}


}
```