OOP Objekt-orienterad programmering

Föreläsning 8

Mer om klasser och objekt
- Hantera många objekt
- ArrayList
- toString() – metoden
- this

Stefan Möller

Vi vill ofta hantera många objekt i ett program:

```
public static void main(String[]args){
   Myra[]allaMyror=new Myra[100];
   int antal=0;

   Myra m = new Myra("Myrre", 43);
   allaMyror[antal] = m;
   antal++;

   m = new Myra("Ante", 73);
   allaMyror[antal++] = m;

   allaMyror[antal++] = new Myra(32);
```

Nya myror läggs in sist i arrayen. Om vi vill placera dem på annat ställe?

Vad händer när antal blir 100?

Stefan Möller

Få plats med mer

Vi kan INTE utöka en array.
Däremot kan vi skapa en ny array som är större och kopiera över innehållet.

```
if (antal==allaMyror.length){
   Myra[] tmp = new Myra[antal*2];
   for (int x=0; x<allaMyror.length; x++)
      tmp[x]=allaMyror[x];

   allaMyror=tmp;
}
```


Om vi vill ta bort en Myra?

```
System.out.print("Vem skall bort?");
String namn=scan.nextLine();

for (int x=0; x<antal; x++)
   if (allaMyror[x].getNamn().equals(namn)){
      allaMyror[x]=null;
      break;
   }
```

Vi riskerar att få "hål" i arrayen, kan ge problem:

```
System.out.println("Dessa myror finns:");
for (int x=0; x<antal; x++)
   System.out.println(allaMyror[x].getNamn());
```

ArrayList<E>

En klass som används för att hantera objekt.
Har "byggt in" problematiken med storlek och eventuella hål så att programmeraren slipper de problemen.
Klass ArrayList ligger i java.util-delbiblioteket.

Vid skapande anges vilken sorts objekt som skall hanteras:

```
ArrayList<Myra> allaMyror = new ArrayList<Myra>();
```

Innehåller ett antal metoder som kan anropas:

```
Myra m = new Myra("Myrre", 43);
allaMyror.add(m);
```

Internt en array som automatiskt utökas vid behov, detta slipper man tänka på som programmerare.

Några av metoderna hos ArrayList<E>:

add(E nytt)	Objektet nytt adderas sist i listan
add(int index, E nytt)	Objektet hamnar på plats index
clear()	Ta bort alla objekt
get(int index)	Returnerar objektet på plats index
isEmpty()	Returnerar true om listan är tom
remove(int index)	Tar bort objektet på plats index
size()	Returnerar antal objekt i listan

E är vilken sorts objekt som hanteras:

```
ArrayList<Myra> allaMyror = new ArrayList<Myra>();
```

Här hanteras Myra-objekt.

```
ArrayList<String> namn = new ArrayList<String>();
```

Här hanterar man textsträngar.

```
public static void main(String[]args){
   ArrayList<Myra> allaMyror = new ArrayList<Myra>();

   Myra m = new Myra("Myrre", 37);
   allaMyror.add(m);

   allaMyror.add(new Myra(112));
   allaMyror.add(new Myra("Ante", 12));

   allaMyror.add(0, new Myra("Anna"));

   //Ta bort en myra:
   System.out.print("Vem skall bort?");
   String vem = scan.nextLine();

   for (int x=0; x<allaMyror.size(); x++)
      if (allaMyror.get(x).getNamn().equals(vem)){
         allaMyror.remove(x);
         break;
      }

   //OBS – inget "hål" i strukturen
```


Innehållet i en ArrayList är objekt.
Fungerar INTE för javas primitiva datatyper.

FEL: `ArrayList<int> talen = new ArrayList<int>();`

Till de olika primitiva datatyperna finns Wrappers:

int	Integer
double	Double
char	Character
boolean	Boolean

Wrapper-klasserna finns i java.lang. Innehåller även ett antal metoder, t.ex. parseInt

```
ArrayList<Integer> talen = new ArrayList<Integer>();
```

Sedan Java 5 automatisk konvertering

```
talen.add(17);
talen.add(43);
int x = talen.get(0);
```

Att skriva ut en Myra

```
class Myra{
   private String namn;
   private int barr;

   //Konstruktorer mm som tidigare

}
```

```
//Utskrift ”någonstans”, t.ex. i en main-metod

Myra m = new Myra(”Myrre”, 17);
System.out.print(m.getNamn()+” har ”+
                 m.getBarr()+” barr”);
```


Att skriva ut en Myra via en metod

```
class Myra{
   private String namn;
   private int barr;

   //Konstruktorer mm som tidigare

   public void skriv(){
      System.out.println(namn+" har "+barr+" barr");
   }
}
```

```
Myra m = new Myra(”Myrre”, 17);
m.skriv();
```

INTE utskrift inuti Myra-klassen utan en metod som returnerar en String med myrans "presentation"

```
class Myra{
   private String namn;
   private int barr;

   //Konstruktorer mm som tidigare

   public String getText(){
      return namn+" har "+barr+" barr";
   }
}
```

Vid anrop kan man sedan göra "vad man vill" med texten

```
Myra m = new Myra(137);
System.out.println(m.getText());
JOptionPane.showMessageDialog(null, m.getText());
```


Utskrift av en Myra-referens

```
Myra m = new Myra("Myrre", 25);
System.out.println(m);
```

Borde väl inte fungera???

Jo, utskriften blir (ungefär...): **Myra@fef3d**

Varför? Java-systemet anropar en metod som heter toString.
Varje klass har en sådan metod som default skriver som ovan.
Denna metod kan göras om:

I Myra:
```
public String toString(){
   return namn+" har "+barr+" barr";
}
```

Utskriften blir nu: **Myrre har 25 barr**

Metoden som returnerar en Myras presentation döps till toString

```
class Myra{
   private String namn;
   private int barr;

   //Konstruktorer mm som tidigare

   public String toString(){
      return namn+" har "+barr+" barr";
   }
}
```

toString blir automatiskt anropad

```
Myra m = new Myra(137);
System.out.println(m);
JOptionPane.showMessageDialog(null, m);
```


Utskrift av alla Myror i en ArrayList<Myra>

```
for (int x=0; x<allaMyror.size(); x++)
   System.out.println(allaMyror.get(x));
```

Med alternativ for-loop

```
for (Myra m : allaMyror)
   System.out.println(m);
```

ArrayList har en egen toString-metod

```
System.out.println(allaMyror);
```

this

Man kan få en referens ”till sig själv” med this.
Anta att två myror skall kunna gifta sig med varandra.
Vi lägger till attributet partner i class Myra samt en metod gifterSig:

```
class Myra{
   private Myra partner = null;

   public boolean gifterSig(Myra m){
      if (m!=null && partner==null && m.partner==null && m!=this){
         partner = m;
         m.partner = this;
         return true;
      }
      return false;
   }
}
```


Ett kommandostyrt program:

```
public static void main(String[]args){
   ArrayList<Myra> myrstack=new ArrayList<Myra>();
   Scanner sc=new Scanner(System.in);

   for(;;){
      System.out.print("1-Skapa myra\n2-Ändra barr\n"+
                       "3-Giftemål\n4-Skriv ut alla\n"+
                       "5-Avsluta\nAnge Kommando: ");
      int kom=Integer.parseInt(sc.nextLine());
      switch (kom){
         case 1: //Koden för att skapa en myra
                 break;
         case 2: //Koden för att ändra antal barr
                 break;
         case 3: break;
         case 4: break;
         case 5:
            System.exit(0);
         default:
            System.out.println("Felaktigt kommando");
      }//switch
   }//for
}//main
```

```
//Skapa en Myra:

System.out.print("Myrans namn: ");
String namn=sc.nextLine();
System.out.print("Antal barr: ");
int antBarr=sc.nextInt());
Myra ny = new Myra(namn, antBarr);
myrstack.add(ny);

//Ändra en Myra's antal barr:

System.out.print("Vilken Myra: ");
String namn=sc.nextLine();
Myra vem=null;
for (Myra m : myrstack)
   if (m.getNamn().equals(namn))
      vem=m;

if (vem!=null){
   System.out.print("Antal nya barr: ");
   int x=Integer.parseInt(sc.nextLine());
   vem.changeBarr(x);
}
else System.out.println("Den Myran fanns ej i stacken.");
```