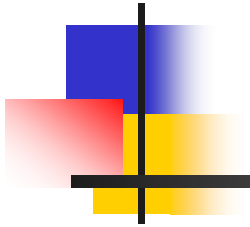
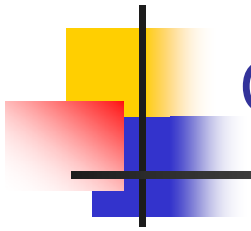


Uke 6 - Objekter, klasser og pekere

27. Sept. 2011,

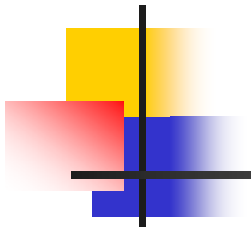
Arne Maus

Inst. for informatikk, UiO



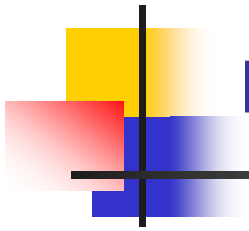
Objekter

- Hvorfor deler vi verden inn i ulike ting/gjenstander når vi snakker om den ?
  - En blomst, fjorten trær, ti mennesker, en bil, en vei, mange murstein, en bankkonti,....
- *Svar* : For bedre å kunne tenke om, snakke om og forstå verden.



Verden består av mange objekter, noen ganske like, noen ulike

- Ser vi rundt oss i auditoriet, ser vi
  - Studenter (mange), stoler, murstein, lysarmatur, blyanter,...
- Vi ser at når vi betrakter verden, deler vi den opp i et passe antall enheter/deler/'klumper' som vi har egne navn for.
- Slike enheter/'klumper' vi deler verden inn i, kaller vi ***objekter***
- Mange av objektene i verden er egentlig like, av samme type, men kan skille seg fra hverandre med feks. ulike navn
  - Studentene Kia og Espen
  - to bøker med ulik tittel/forfatter
- Objekter som er egentlig er like sier vi tilhører samme ***klasse***
  - De beskrives av samme sett med variable, men har *ulike* verdier på noen disse (to biler av samme bilmerke men med ulike reg.nummer)

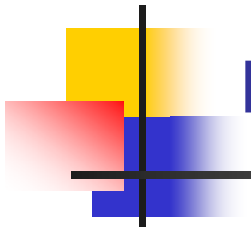


Klasser og objekter i verden

- To objekter kan også være helt vesensforskjellige (f.eks et tre og en lastebil)
  - De er da to objekter av hver sin klasse (klassen Tre og klassen Lastbil)
- Hva vi velger å betrakte som et objekt, og hvilke klasser vi bruker for å beskrive en problemstilling, er ikke bestemt på forhånd. Innenfor vide rammer bestemmer vi det selv.

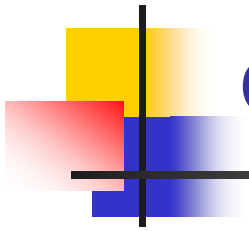
Omdebattert av filosofene, men som OO-programmerer, hevder jeg :

- Klasser er generelle begreper som egentlig ikke eksisterer i verden – det som eksisterer er objektene.
- Generelle begreper (klasser) er menneskenes måte å beskrive og strukturere verden, ikke gitt en gang for alle og vi kan godt lage oss nye begreper.



Hvor mange klasser (og objekter) er det ?

- Hvilke klasser vi bruker til å beskrive et problem, varierer ofte etter hvor detaljert vi betrakter en problemstilling og hvilke spørsmål vi ønsker å kunne gi svar på:
  - Beskriver vi problemet med veitrafikk og køer på veiene, er vi neppe interessert i mer enn å telle antall biler og kanskje skille mellom lastebiler, busser og personbiler.
  - Beskriver vi problemet til en bilfabrikk, trenger vi en meget detaljert og komplisert beskrivelse av hver bil (bestående av motor, hjul, karosseri, lys,..., hver beskrevet med sin klasse) og mange ulike typer av biler. Vi se da en rekke klasser som hver beskriver sin sine objekter.



Objektorientert Programmering - I

Når vi betrakter et problem vi skal lage et datasystem for, gjør vi to avgrensninger:

1. Vi ser bare på ***en del av verden*** (vårt problemområde)
2. Innenfor problemområdet betrakter og beskriver vi bare det som er der med ***en viss detaljeringsgrad*** - bare så mange detaljer vi trenger for å svare på de spørsmål datasystemet skal kunne gi svar på.

Eks: Hvordan beskrive en student ? Skal vi lage:

a) Et Studentregister, registrerer vi bare navn, personnummer, adresse, tidligere utdannelse og kurs (avlagte og kurs vedkommende tar nå)

b) Et legesystem for studenter, ville vi ta med svært mange opplysninger om hver student (medisiner, sykdommer, resultat fra blodprøver, vekt...) som vi ikke ville drømme om å ha i et vanlig studentregister

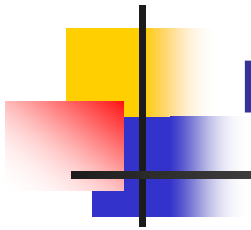
Objektorientert Programmering - II

- Når vi skal lage et programsystem, så skal det i størst mulig grad være en modell av vårt problemområde – en en-til-en kopi:

  - ***Ett objekt i problemområdet*** *skal medføre at det skal være* ***ett objekt i programmet*** *som representerer dette 'verdens-objektet'.*

  (i tillegg kommer mange klasser og objekter i programmet for å lese fra tastatur og fil, skrive til skjerm,..)

Eks.: Hver virkelig student skal ha sitt Student-objekt i et studentregister-system.

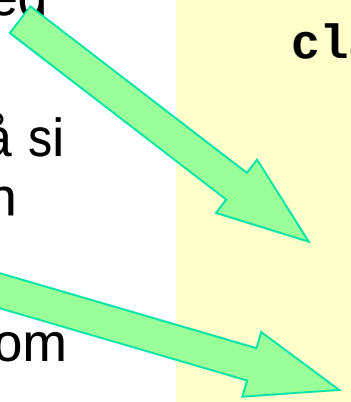


Forholdet mellom klasser og objekter i Java

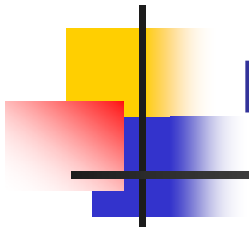
- Klasser er objektfabrikker. Vi sier **new** på en klasse og får et objekt tilbake.
- Det objektet er en kopi av klassen (deklarasjonene inne i klassen + metodene i klassen).
- Har klassen en start-metode (som heter det samme som klassen), har vi også greid å gi spesielle startverdier til noen av variablene i objektet.
- En **Klasse** er altså som en slags støpeform/ fabrikk som lar oss lage en eller flere objekter av den klassen.
- Det er kall på metoder i **objektene** som gjør at programmet vårt gjør noe.
  - Et lite unntak her for statiske metoder – undervises senere.

Hvordan lage klasser og objekter i et program.

- Klasser deklarerer vi med **`class`**
- Vi lager pekere til objekter ved å deklarere dem med klassenavnet
- Vi lager et objekt med å si **`new`** foran et klassenavn
- Forholdet mellom et objekt og en peker er som en array-peker og et array-objekt



```
class Student {
   String navn, adresse;
}

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

    Student s1, s2;

    s1 = new Student();
    s2 = new Student();

   }
}
```



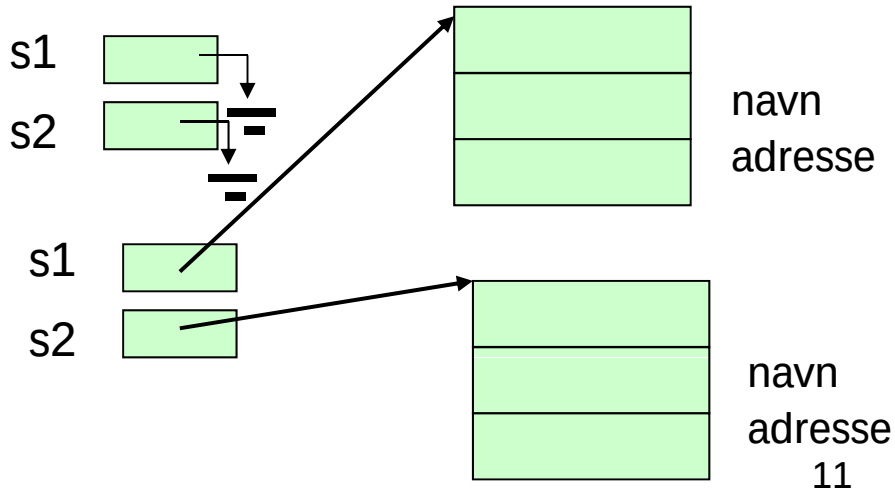
Hva er et objekt i programmet?

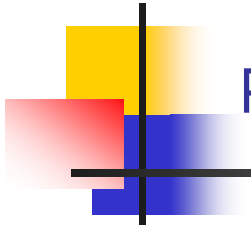
- Et objekt er et område i lageret som inneholder *en kopi* av alle de metodene og variable i en klasse det *ikke* står **`static`** foran
- Klassene er en slags mal/form/oppskrift/fabrikk som vi kan lage objekter med.
- Lager vi to, tre,.. objekter av klassen, får vi to, tre,.. slike kopier
- De variable og metode det *ikke* står **`static`** foran, kalles *objekt-variable og objekt-metoder*
- De variable og metoder det står **`static`** foran, kalles *klasse-metoder og klasse-variable*, og blir ikke med i objektene (men ligger lagret i bare ett eksemplar et annet sted)

Peker og objekter

```
class Student {
   String navn, adresse;
}

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

    Student s1,
            s2;

    s1 = new Student();
    s2 = new Student();

   }
}
```

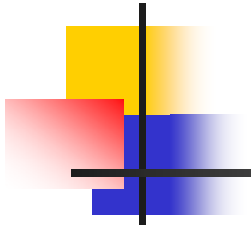
- En peker inneholder adressen til hvor et objekt ligger i lageret
- eller den inneholder 'null' (= ikke-objekt)
- Vi tegner adressen som en 'pil'





Programmer i Java består av en eller flere klasser

- Vi deler opp programmet vårt i flere klasser
  - Fordi hver programdel (klasse) skal være mulig å holde oversikt over – ikke for stor
  - Fordi en klasse skal være god modell av en del av problemet vårt vi lager program for.
    - Anta at vi hadde et datasystem som omhandlet kurs og studenter. Da ville vi ha en klasse Student og en klasse Kurs i programmet.
- En klasse inneholder
  - Deklarasjon av null eller flere variable
    - som beskriver *ett eksemplar* av det klassen er modell av
  - Null eller flere metoder
- En klasse representer et generelt begrep som:
  Student, Kurs, Person, Dokument, Eiendom, DVDRegister, DVD, Billett, Bil, Fly, Tre, Ku, Motorsykkel…



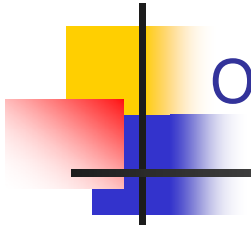
Objekter og pekere, og hvordan få adgang til innmaten av et objekt (.)

- Når vi har laget et objekt med **new**, har vi altså fått en kopi av objekt-variablene og –metodene, men hvordan får tak i dem ?
- Vi bruker operatoren . (punktum).
  - Foran punktumet har vi navnet på en peker til et objekt.
  - Etter punktum har vi navnet på en variabel eller metode inne i objektet –
  - Punktumet leses som 'sin' eller 'sitt'
    - eks: La s1 peke på et Student-objekt.

```
s1.navn = "Ola N";
```

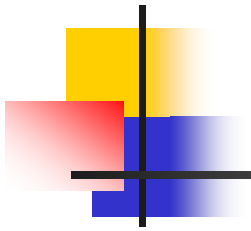
```
class Student {
   String navn, adresse;

   void skrivUt() {
     System.out.println("Student med navn:"
         + navn+ ", adr:" + adresse);
   }
}

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

    Student s1, s2;

    s1 = new Student();
    s1.navn = "Ola N";
    s1.adresse = "Storgt. 12, 1415 Nordby";
    s2 = new Student();
    s2.navn = "Åsne S";
    s2.adresse ="bokhandelen i Kabul";
    s1.skrivUt();
    s2.skrivUt();
   }
}
```

```
>java StudentRegister
Student med navn:Ola N, adr:Storgt. 12, 1415 Nordby
Student med navn:Åsne S, adr:bokhandelen i Kabul
```



Oppsummering om klasser, objekter, pekere og .

- Verden består av **objekter** av ulike typer (**klasser**).
  Ofte er det mange objekter av en bestemt type.
- Objekter som er av samme klasse, beskrives med de samme variablene, men vil ha forskjellige verdier på noen av disse.
  - Eks: To bankkonti med ulik eier og kontonummer, men kan f.eks ha samme beløp på saldo (tilfeldigvis)
- Vi lager OO-programmer ved å lage en modell av problemområdet i Javaprogrammet
  - ett objekt i verden gir ett tilsvarende Java-objekt i programmet
  - Objekter kan være av ulik type, og for hver slik type deklarerer vi en klasse i programmet



.. oppsummering forts.

- Et Javaprogram består av en eller flere klasser
- En klasse er en deklarasjon av data og metoder for ***ett objekt*** av klassen.
- Vi deklarerer pekere til objekter av en bestemt klasse – f.eks. class Kurs {..} slik:

```
Kurs kurs14, k2, k;
```

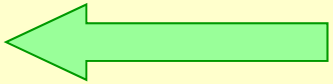
- Vi lager objekter fra klassen med **new**

```
k2 = new Kurs();
```

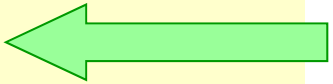
- Et objekt inneholder en kopi av alle ikke-statiske variable og ikke-statiske metoder i klasse
  - Disse kalles objekt-variable og objekt-metoder
- Vi får adgang (lese, skrive og kalle metoder) til det som er inni et objekt ved . operatoren :
  - **Vi må ha en peker til et objekt etterfulgt av punktum .**

```
s2.adresse ="bokhandelen i Kabul";
s1.skrivUt();
```

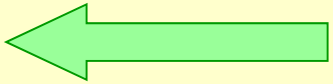
```
class Kurs {
   String kurskode;
   int studiepoeng;

   void skrivUt() {
     System.out.println("Kurs med kode:"
         + kurskode+ ", og stp:" + studiepoeng);
   }
}

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

    Kurs inf, mat;

    inf = new Kurs();
    inf.kurskode = "INF1000";
    inf.studiepoeng = 10;
    inf.skrivUt();

    mat = new Kurs();
    mat.kurskode = "MAT1010";
    mat.skrivUt();

   }
}
```



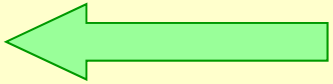
objektvariable



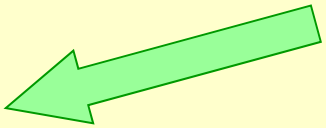
Objekt - metode



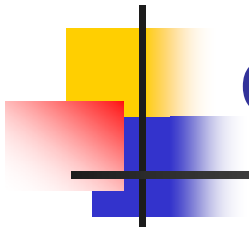
Klasse - metode



Lager to objekter av klassen Kurs

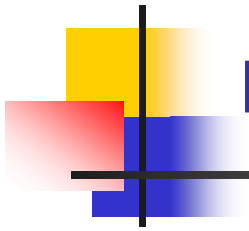


```
>java KursRegister
Kurs med kode:INF1000, og stp:10
Kurs med kode:MAT1010, og stp:0
```



Oversikt

- Mer om static
  - Klassevariable
  - Klasse-metoder
- Arrayer av pekere til objekter
- UML objektdiagram
  - Eks CD-samling
- Konstruktører
- eksempel med tre klasser: Student, Kurs og StudentRegister



Klasse-variabel (=statisk variabel)

- Setter vi static foran en variabel, er det er bare **én** felles variabel med det navnet for alle objektene.
- Setter vi **`static`** foran en metode, har den bare utsikt til :
  - sine egne lokale variable og parametere
  - andre statiske variable og metoder
  - klassenavnene
- Statiske metoder og variable kan man få adgang til både
  - via klassenavnet og punktum
  - via peker til et objekt av klassen og punktum

```
class B {
        static int i = 0;
        double x = 0.0;
}
class A
{       int k;

    public static void  main ( String[] args)  {
            B b1 = new B(), b2 = new B();
            // endre klassevariable (det er bare en felles)
            System.out.println("b1.i :"+ b1.i+", b2.i:" + b2.i);
            b1.i = 4;
            System.out.println("b1.i :"+ b1.i+", b2.i:" + b2.i);
            // endre objektvariabel (en kopi i hvert objekt)
            System.out.println("b1.x :"+ b1.x+", b2.x:" + b2.x);
            b1.x = 2;
            System.out.println("b1.x :"+ b1.x+", b2.x:" + b2.x);
       }
}
```

```
>java A
b1.i :0, b2.i:0
b1.i :4, b2.i:4
b1.x :0.0, b2.x:0.0
b1.x :2.0, b2.x:0.0
```

```
class A2
{        int k;  // objektvariabel 'k'
    public static void  main ( String[] args)  {
              k = 1;
 }}
```

```
>javac a2.java
a2.java:6: non-static variable k cannot be referenced from a static context
         k = 1;
         ^
1 error
```

```
class A2
{      int k;

     public static void  main ( String[] args)  {
               A2 aa = new A2();
               aa.k = 1;
       }
}
```

```
>javac A2.java


>
```

Arrayer av pekere til objekter

- Vi kan lage arrayer av pekere til objekter (men ikke av objektene direkte) omlag på samme måte som vi lager arrayer av int, double og String
- Har vi deklarert klassen **`Kurs {...}`** kan vi lage array som følger:

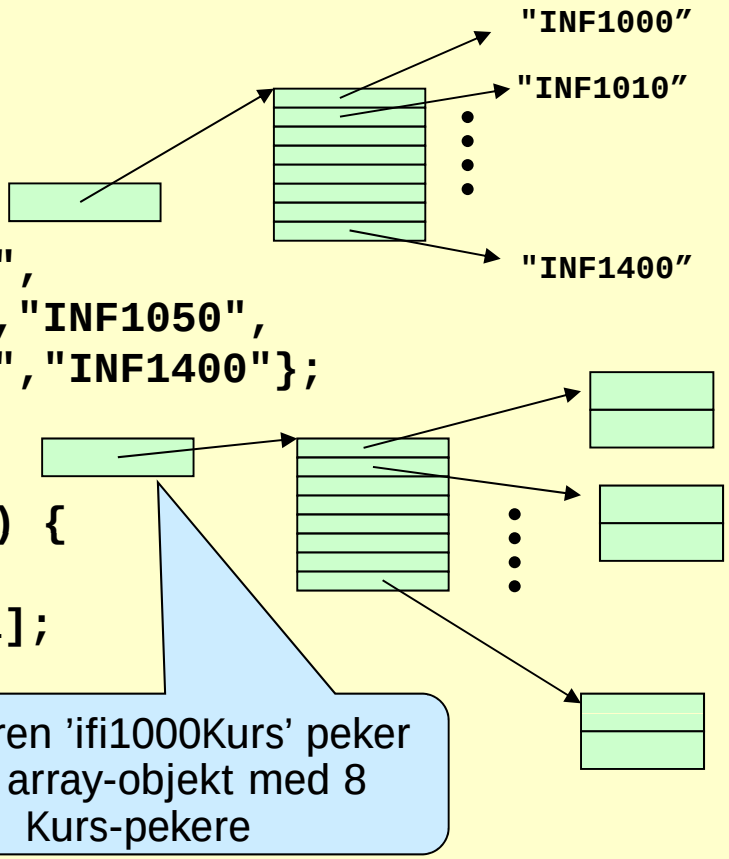
```
Kurs [] ifiKurs;

ifiKurs = new Kurs[120];

ifiKurs[0] = new Kurs();
```

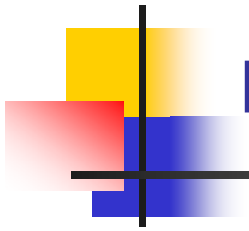
Her deklareres 'bare array-pekeren

Her lages array-objektet med 120 'tomme' pekere

Her settes det første pekeren i array-objektet til å peke på et nytt Kurs-objekt

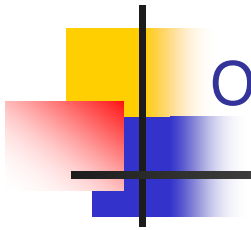
```
class Kurs {
   String kurskode;
   int studiepoeng=10;

   void skrivUt() {
     System.out.println("Kurs med kode:"
         + kurskode + ", og stp:"
          + studiepoeng);
   }
}

class KursRegister2 {
  public static void main(String args []) {

    String [] kursKoder= {"INF1000","INF1010",
                          "INF1020","INF1040","INF1050",
                           "INF1060","INF1070","INF1400"};
    Kurs [] ifi1000Kurs = new Kurs[8];

    for(int i = 0; i < kursKoder.length; i++) {
       ifi1000Kurs[i] = new Kurs();
       ifi1000Kurs[i].kurskode = kursKoder[i];
       ifi1000Kurs[i].skrivUt();
    }
 }
}
```





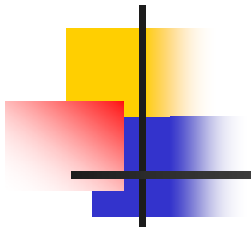
Eksekvering av KursRegister2

```
>java KursRegister2
Kurs med kode:INF1000, og stp:10
Kurs med kode:INF1010, og stp:10
Kurs med kode:INF1020, og stp:10
Kurs med kode:INF1040, og stp:10
Kurs med kode:INF1050, og stp:10
Kurs med kode:INF1060, og stp:10
Kurs med kode:INF1070, og stp:10
Kurs med kode:INF1400, og stp:10
```



Oppsummering om klasser, objekter, pekere og .

- Verden består av **objekter** av ulike typer (**klasser**).
  Ofte er det mange objekter av en bestemt type.
- Objekter som er av samme klasse, beskrives med de samme variablene, men vil ha forskjellige verdier på noen av disse.
  - Eks: To bankkonti med ulik eier og kontonummer, men kan f.eks ha samme beløp på saldo (tilfeldigvis)
- Vi lager OO-programmer ved å lage en modell av problemområdet i Javaprogrammet
  - ett objekt i verden gir ett tilsvarende Java-objekt i programmet
  - Objekter kan være av ulik type, og for hver slik type deklarerer vi en klasse i programmet



.. oppsummering forts.

- Et Javaprogram består av en eller flere klasser
- En klasse er en deklarasjon av data og metoder for ***ett objekt*** av klassen.
- Vi deklarerer pekere til objekter av en bestemt klasse – f.eks. class Kurs {..} slik:

```
Kurs kurs14, k2, k;
```

- Vi lager objekter fra klassen med **new**

```
k2 = new Kurs();
```

- Et objekt inneholder en kopi av alle ikke-statiske variable og ikke-statiske metoder i klasse
  - Disse kalles objekt-variable og objekt-metoder
- Vi får adgang (lese, skrive og kalle metoder) til det som er inni et objekt ved . Operatoren :
  - **Vi må ha en peker til et objekt etterfulgt av punktum .**

```
s2.adresse ="bokhandelen i Kabul";
s1.skrivUt();
```