

## INF1000 - Forelesning 7

- Litt repetisjon fra forelesning 6
- Gjennomgang av BankSystem med full kode
- Mer om klasser og objekter
- UML

1. mars 2010  
Christian M. Hansen  
Institutt for informatikk, UiO

1

## 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 samme type variable, men har gjerne forskjellige verdier
  - 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

2

## .. 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();
```

3

## Et meget enkelt banksystem

- Vi har klassene:
  - Konto
  - Bank
  - BankSystem (med main)
 Hvilke opplysninger har vi i hver klasse/objekt ?
- Og operasjonene (dvs. metodene):
  - Ny konto
  - Sett inn
  - Ta ut
  - Vis summen av innskudd i banken
- Full kode og kjøreekspler...

4

## Null, && og søppeltømmeren

- Av og til har vi behov for å teste om en peker virkelig peker på et objekt eller ikke:

```
Student s = hyblene[i][j].leietager ;
if (s != null && s.navn.equals("Ola")) {
    // her kommer vi bare hvis s peker på et studentobjekt
    // og navnet i det studentobjektet er lik "Ola"
    .....
}
```

- Hvis vi vil fjerne et objekt fra en peker og fra hele systemet:

```
hyblene[i][j].leietager = null;
```

- Et objekt som ingen pekere peker på, blir tatt av søppeltømmeren – et program som automatisk startes hvis det er lite plass i hukommelsen

5

## Konstruktører – startmetoder i klasser

- Når vi lager et objekt av en klasse med **new**, kaller vi egentlig en metode som heter det samme som klassen (derfor parentes bak klassenavnet).
- Vi får automatisk med en slik konstruktør-metode fra oversetteren dersom vi ikke skriver en slik konstruktør selv. Den vi får automatisk er uten parametere og gjør ingen ting.
- Konstruktører nyttes i all hovedsak til å gi fornuftige startverdier for variable i objektet som dannes.
- De konstruktørene vi skriver kan ha parametere.
- Konstruktørene skal ikke ha noen type foran seg, heller ikke void.
- Vi kan ha flere konstruktører i en klasse, men da må parametrene være ulike i antall eller typen av parametrene

6

## Eksempel Student med én konstruktør

```
class Student {
    String navn;
    Kurs[] mineKurs;

    Student(String navn, Kurs[] k) {
        this.navn = navn;
        mineKurs = new Kurs[k.length];
        for (int i = 0; i<k.length; i++){
            mineKurs[i] = k[i];
            mineKurs[i].antStudenter++;
        }
    }
}
```

7

## Eksempel Student med to konstruktører

```
class Student {
    String navn;
    Kurs[] mineKurs;
    Student() {
        mineKurs = new Kurs[3];
    }
    Student(String navn, Kurs[] k) {
        this.navn = navn;
        mineKurs = new Kurs[k.length];
        for (int i = 0; i<k.length; i++) {
            mineKurs[i] = k[i];
            mineKurs[i].antStudenter++;
        }
    }
}
```

8

## this

- Av og til trenger vi en peker til det objektet metoden vi utfører er inne i. Java-ordet **this** gir oss alltid det.
- Brukes i to situasjoner:
  - Vi har en konstruktør, og parametrene til denne heter det samme som objekt-variable i objektet. Eks:

```
class A {
    int antall;
    A (int antall ){
        this.antall = antall;
    }
} // end A
.. A apek = new A(12);
```

- Vi skal kalle en metode i et annet objekt (gjør av en annen klasse). Da kan vi bruke this for å overføre en parameter til denne metoden om hvilket objekt kallet kom fra.

9

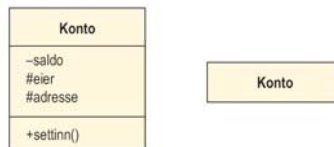
## UML-diagrammer av programmene våre

- Hvorfor tegne diagrammer over programmene
  - Oversikt
  - Samarbeid med andre programmerere / systemutviklere
  - Arkitekter, ingeniører tegner først, så bygger de!
  - Enklere å endre en tegning enn programmet
- Objektdiagrammer
- Klassediagrammer

10

## Klassediagrammer

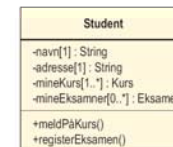
- En mer **kompakt** måte enn objektdiagrammer å tegne sammenhengen i programmet
- Skiller seg fra objektdiagrammer ved at vi ikke direkte tegner datastrukturen (pekere og pekerarrayer), men bare forhold (assosiasjoner, forbindelser) mellom klassene.
- I klassediagrammer dokumenterer vi også sentrale metoder.
- Forholdene er linjer med et logisk navn og antall objekter i hver ende
- Anta at vi har laget en class Konto med tre objektvariable: saldo, eier og adresse og en metode: settInn().



11

## Tre (fire) mulig felter i tegning av en klasse

- Navnefeltet (alltid)
  - klassenavnet
- Kan utelates:
- Variabelfeltet (attributtene)
  - variabelnavn evt. med type
- Metode-feltet
  - Evt med parametere og returverdi
- (Unntaks-feltet)



Symboler for synlighet  
(fra resten av programmet)

+ public  
- private  
# protected  
~ package

12

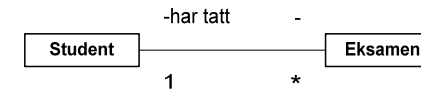
## UML Klassediagrammet kan nyttes til

- Modell av problemområdet (domenemodell)
- Modell av klassene i programmet  
(+ modell av databasen,...)
- Men siden vi skal modellere virkeligheten en-til-en i programmet vårt, så blir de like i utgangspunktet

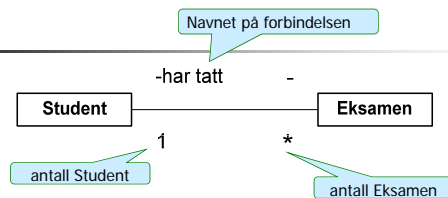
13

## Forhold mellom klasser

- **En student har null eller flere eksamener**
- Vi tegner et forhold mellom to klasser som har med hverandre å gjøre logisk sett, og:
- hvor vi i programmet vil kunne følge pekere for å få adgang til variable eller metoder
- Vi skriver hvor mange objekter det maksimalt på ett tidspunkt kan være på hver side av et slikt forhold
- Siden vi med: Eksamen mener en avlagt enkelt-eksamen vil en Eksamen bare være tilknyttet en bestemt student



14



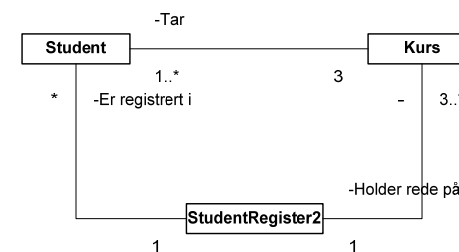
- Forbindelsen leses fra venstre: "En student har tatt null, en eller flere Eksamener"
- Antallet objekter angis slik:

Skrivemåte	Betydning
1	en
*	null, en eller flere
1..*	minst en
3..*	minst tre
3,4,5	tre, fire eller fem

15

## Studentregister2 – med tillegg: klassen Kurs vet hvilke Studenter som tar kurset

- Et studentregister holder orden på studentene og kursene, og en student tar 3 kurs hvert semester



16

## Regler for å plassere riktige antall på et forhold

1. Anta at du står i **ett** objekt av en klasse og ser over til (langs en forbindelse) til en annen klasse:
2. Hvor mange objekter ser du da maksimalt *på et gitt tidspunkt* av den andre klassen
3. Det antallet noteres (jfr. tabellen) på den andre siden
4. Du går så over forbindelsen til den andre klassen og antar at du nå står i **ett** objekt av denne klassen og gjenntar pkt. 1-3

17

## Hvilke forhold skal vi ha med i klassediagrammet ?

- Slike forhold hvor ett objekt av den ene klassen:
  - inneholder
  - består av
  - eier,...
 en eller flere objekter av den andre klassen
- Det vi i programmet vil følge en peker for å få tak i verdien på visse variable i den andre klassen eller kalle en metode.

Det er da ikke 'naturgitt' hvilke forhold vi har i et klassediagram, det avhenger av hvilke spørsmål vi vil være interessert i å svare på.

18

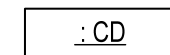
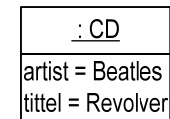
## Objekt-diagrammer

- Vi tegner en typisk situasjon av objekter i systemet vårt, når vi har fått datastrukturen på plass.
- Vi tegner og navngir bare de mest sentrale dataene som:
  - pekere
  - peker-arrayer
  - noen sentrale variable i objektene

19

## Tegning av et objekt (med mer eller mindre detaljer)

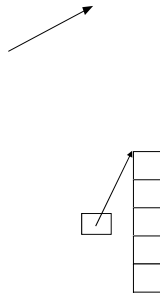
- To eller ett felt(er) i en boks
- Navnfeltet
  - objektnavn:klassenavn eller bare
  - :klassenavn
- Attributt-feltet (kan være tomt)
  - Navnet på sentrale objektvariable evt. også med verdier



20

## Andre elementer i et objektdiagram

- Pekere
- Peker-arrayer



21

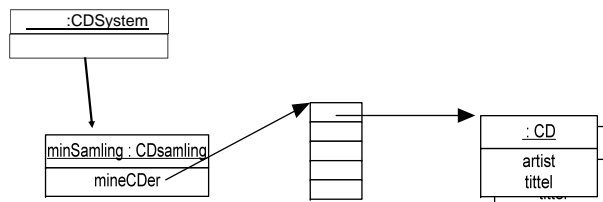
## Eksempel: Et CD-samling

- Vi ønsker å lage et lite menystyrt program for å holde orden på CD-samlinga vår med mindre enn 1000 CDer. Vi skal ha funksjoner for å :
  - registrere ny CD
  - Søke etter artist (skriv ut alle platene med denne)
  - Skrive ut hele registeret
- Hvilke klasser har vi i dette problemet
  - Opplagt 'class CD'
  - Noen fler ?

22

## Klassene: CD og CDsamling

- Vi tenker oss følgende datastruktur (er den tilstrekkelig?)
- Vi har her forenklet programmet (klassen CDSYSTEM inneholder main, CDSamling meny-metode og sentral switch i løkke-metoden,... , mens CD inneholder utskriftsrutine .



23

```

import easyIO.*;

class CDSYSTEM {
    public static void main(String args []) {
        new CDSamling().løkke();
    }
}

class CD {
    String artist, tittel;
    void skrivUt(Out u) {
        u.outln("Artist:" + artist + ", Tittel:" + tittel);
    }
}

class CDSamling {
    CD [] minSamling = new CD[1000];
    int antCDer = 0;

    void løkke() {
        In tast = new In();
        Out skj = new Out();
        String a;
        CD c;
        int valg;
    }
}
  
```

```

do{ skj.outln("Velg:");
  skj.outln(" 1 - les ny plate (skriv artist platetittel);
  skj.outln(" 2 - skriv artist");
  skj.outln(" 3 - avslutt");
  valg = tast.inInt();

  switch(valg) {
    case 1: // les data
      c = new CD();
      minSamling[antCDer++] = c;
      skj.out("Gi artistnavn:");
      c.artist = tast.inWord();
      skj.out("Gi tittel:");
      c.tittel = tast.inWord();
      break;
    case 2: // skriv data
      skj.out("Gi artistnavn:");
      a = tast.inWord();
      for(int i = 0; i < antCDer; i++)
        if (minSamling[i].artist.equals(a))
          minSamling[i].skrivUt(skj);
      break;
    case 3: // avslutt
      skj.out("Systemet avslutter");
      break;
    default: // feil
      skj.out("Bare gi verdier: 1 - 3");
  }
} while (valg != 3);
}

```

```

>java CDSystem
Velg:
1 - les ny plate (skriv artist platetittel)
2 - skriv artist
3 - avslutt
1
Gi artistnavn:PJHarvey
Gi tittel:RidOfMe
Velg:
1 - les ny plate (skriv artist platetittel)
2 - skriv artist
3 - avslutt
1
Gi artistnavn:TomWaits
Gi tittel:MuleVariations
Velg:
1 - les ny plate (skriv artist platetittel)
2 - skriv artist
3 - avslutt
2
Gi artistnavn:TomWaits
Artist:TomWaits, Tittel:MuleVariations
Velg:
1 - les ny plate (skriv artist platetittel)
2 - skriv artist
3 - avslutt

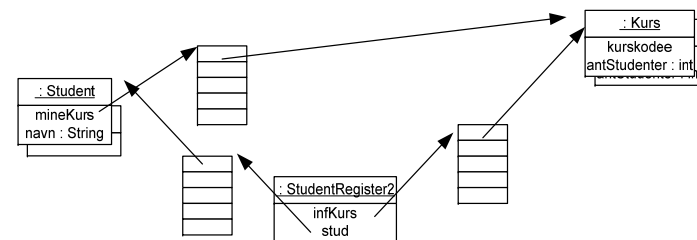
```

## Et helt studentregister med kurs, studenter og registeret

- Vi har Studenter på Ifi som første semester tar tre kurs, samtidig som vi har behov for å registrere kurs og hvor mange studenter som tar hvert kurs.
- Vi tegner først en tenkt datastruktur – et UML objektdiagram
- så skriver vi programmet

27

Objektdiagrammet er en forenkling av programmet. Det tar bare med den essensielle datastrukturen (mest pekere og peker-arrayer) som holder datastrukturen sammen



```

class StudentRegister2{
    public static void main(String args []) {
        String [] kurskode = {"INF1000","INF1040","MAT1030"};

        // lag kurs
        Kurs [] infKurs = new Kurs[3];
        for (int i = 0 ; i< infKurs.length; i++)
            infKurs[i] = new Kurs(kurskode[i]);

        //lag studenter på informatikk bachelor
        Student [] stud = new Student[3];
        stud[0] = new Student("Ola N", infKurs);
        stud[1] = new Student("Åsne S",infKurs);
        stud[3] = new Student();

        for (int i = 0 ; i< stud.length; i++)
            stud[i].skrivUt();
    }
}
    
```

```

class Student {
    String navn;
    Kurs [] mineKurs = new Kurs[3];

    Student(){mineKurs = new Kurs[0];}

    Student(String navn, Kurs [] k){
        this.navn = navn;
        for (int i = 0; i<k.length; i++)
            mineKurs[i] = k[i];
            mineKurs[i].antStudenter++;
    }

    void skrivUt() {
        System.out.println("Student med navn:"+ navn+ ",og kurs:");
        for (int i = 0; i < mineKurs.length; i++)
            System.out.println(mineKurs[i].kurskode);
    }
}

class Kurs {
    String kurskode ;
    int antStudenter = 0;

    Kurs(String k) {
        kurskode = k;
    }
}
    
```

```

>java StudentRegister2
Student med navn:Ola N, og kurs:
INF1000
INF1040
MAT1030
Student med navn:Åsne S, og kurs:
INF1000
INF1040
MAT1030
Student med navn:null, og kurs:
    
```

