

## Inf1000 (Uke 9)

### UML og enda mer om klasser

Grunnkurs i programmering  
Institutt for Informatikk  
Universitetet i Oslo

Anja Bråthen Kristoffersen og Are Magnus Bruaset

## Hva skal vi lære i dag?

- Repetisjon om filbehandling
- Ny type, oppramsingstypen enum
- UML
  - Eksempler med klasser og objekter
- Oppsummering/repetisjon av klasser, objekter og UML.

13.03.2006

2

## Filbehandling

- Importer pakken `easyIO.*`;
- Lesing til/fra fil er veldig likt det å skrive til/fra skjerm.
  - Åpn filen `fil.txt` for lesing ved å lage et objekt (her `lesFra`) av klassen `In`:  

```
In lesFra = new In("fil.txt");
```
  - Åpn filen `ut.txt` for skrivning ved å lage et objekt (her `skrivTil`) av klassen `Out`:  

```
Out skrivTil = new Out("ut.txt");
```
  - Husk alltid å lukke objektet av klassen `Out` når du ikke skal skrive mer til filen:  

```
skrivTil.close();
```

13.03.2006

3

## Hvilke lesemetoder finnes?

Datatype	Eksempel	Beskrivelse
Int	<code>lesFra.inInt();</code>	Leser inn et heltall
Double	<code>lesFra.inDouble();</code>	Leser inn et desimaltall
Char	<code>lesFra.inChar(c);</code>	Leser inn et tegn
String	<code>lesFra.inWord();</code> <code>lesFra.inWord("\n");</code> <code>lesFra.inLine();</code>	Leser inn et ord (fram til mellomrom) Leser neste ikke-tomme linje Leser inn neste linje
	<code>lesFra.endOfFile();</code>	Sjekker om slutten på filen er nådd
	<code>lesFra.lastItem();</code>	Sjekker om slutten på filen er nådd, leser forbi blanke tegn.

13.03.2006

4

## Hvilke skrivemetoder finnes?

Datatype	Eksempel	Beskrivelse
int	<code>skrivTil.out(x);</code> <code>skrivTil.out(x, 6);</code>	Skriv x Skriv x høyrejustert på 6 plasser
double	<code>skrivTil.out(x, 2);</code> <code>skrivTil.out(x, 2, 6);</code>	Skriv x med 2 desimaler Skriv x med 2 desimaler på 6 plasser
Char	<code>skrivTil.out(c);</code>	Skriv c
String	<code>skrivTil.out(s);</code> <code>skrivTil.out(s, 6);</code>	Skriv s Skriv s på 6 plasser (venstrejustert)
	<code>skrivTil.outln();</code>	Skriv en linjeskift
	<code>skrivTil.close();</code>	Lukk filen

Merk: dersom antall plasser spesifiseres og det ikke er plass til det som skal skrives ut, vil det som skrives ut avsluttes med tre punktumer: ...

5

## Eksempel

- Anta at det fins en fil med navn på alle studenter samt kjønn og alder. Først i filen er det opplysning om kursnavn og antall studenter.
- Beregn prosentandelen jenter blant studentene.
- Beregn gjennomsnittsalder til studentene
- Skriv til ny fil:
  - kursnavn
  - antall studenter
  - prosentandel jenter
  - gjennomsnittsalder

13.03.2006

6

## Eksempler på tekstfilene

Les fra:

```
Inf1000
5
M 30 Per Hansen
M 20 Ola Normann
K 23 Heidi Larsen
M 22 Inge Braa
K 35 Anna Enger
```

Utskriftsfilen:

```
Inf1000
5
40% jenter
Gjennomsnittsalder: 26 år
```

13.03.2006

7

## Hva må vi lese inn / hvilke variable skal vi ta vare på

- Kursnavn: kaller variabelen kurs, les med `inWord()`
- Antall studenter: kaller variabelen `antStud`, les med `inInt()` (les resten av linjen med `inLine()`)
- Nå vet vi hvor mange linjer det er i resten av filen, dermed kan vi bruke ei for-løkke.
- For hver linje les inn:
  - kjønn (`inChar()`), øk en tellevariabel med en, enten k eller m
  - alder (`inInt()`), øk samlet `totalAlder` med alder
  - navnet (`inLine()`), dette trenger vi ikke senere, vi tar heller ikke vare på det

13.03.2006

8

## Hva skal vi skrive ut

- Kursets navn: `.outln(kurs);`
- Antall studenter: `.outln(antStud);`
- Prosentandel jenter
  - Formel: `Math.round(100*k/antStud)`
- Gjennomsnittsalder
  - Formel: `Math.round(totalAlder/antStud)`

**NB! HUSK Å LUKKE FILEN.**  
`.close();`

13.03.2006

9

```
import easyIO.*;
class StudentInfo{
    public static void main (String args[]){
        String fraFilNavn = args[0];
        String tilFilNavn = args[1];
        In fraFil = new In(fraFilNavn);
        String kurs = fraFil.inWord();
        int antStud = fraFil.inInt();
        fraFil.inLine();
        int alder = 0;
        int m = 0;
        int k = 0;
        for (int i = 0; i < antStud; i++){
            char c = fraFil.inChar();
            switch (c){
                case 'M': m++; break;
                case 'K': k++; break;
            }
            alder = alder + fraFil.inInt();
            fraFil.inLine();
        }
        Out tilFil = new Out(tilFilNavn);
        tilFil.outln(kurs);
        tilFil.outln(antStud);
        tilFil.out(Math.round(k*100/antStud));
        tilFil.outln("% jenter");
        tilFil.out("Gjennomsnittsalder: ");
        tilFil.outln(Math.round(alder/antStud));
        tilFil.close();
    }
}
```



## enum: oppramsingstyper

- I java 1.5 er det kommet en ny type variabel som bare skiller mellom ulike navngitte verdier.
- Eksempel:

```
enum Hverdag {mandag, tirsdag, onsdag, torsdag, fredag}
Class EnumEks{
    public static void main(String[] args){
        Hverdag d = Hverdag.mandag
        System.out.println("I dag er det" + d);
    }
}
```

13.03.2006

11

## Eksempel: iterer over elementene i enum

```
enum Hverdag {mandag, tirsdag, onsdag, torsdag, fredag}
Class EnumEks2{
    public static void main(String[] args){
        for (Hverdag d: Hverdag.values()){
            System.out.println("dag: " + d +
                " ukedagnummer: " + d.ordinal());
        }
    }
}
```

Metodene `values()` og `ordinal()` er forhåndslagde metoder i Java 1.5. Metoden `values()` itererer over alle verdiene i oppramsningstypen. Plasseringen det elementet vi ser på har i oppramsingen finner vi med metoden `ordinal()`.

13.03.2006

12





## Eksempel: verdi på elementene i enum

```
enum Hverdag {
    mandag(4), tirsdag(3), onsdag(2), torsdag(1), fredag(0);
    final int dagerIgjenTilHelgen;
    Hverdag (int v){
        dagerIgjenTilHelgen = v;
    }
}

class EnumEks3{
    public static void main(String[] args){
        for (Hverdag d: Hverdag.values()){
            System.out.println("dag: " + d +
                ", antall dager igjen til helgen er: "
                + d.dagerIgjenTilHelgen);
        }
    }
}
```

13

## Hva er UML (Unified Modeling Language)

- Internasjonal standard for planlegging og dokumentering av programmer (uavhengig av programmeringsspråk).
- Det fins mange typer diagrammer innen UML, vi skal lære om:
  - Objektdiagrammer
  - Klassediagrammer
- UML brukes både som hjelp og støtte til utforming av programmet og til dokumentasjon av et ferdig program.

13.03.2006

14

## UML-diagrammer av programmene våre

- Diagrammer over programmene
  - gir oversikt
  - gjør samarbeid med andre programmerere / systemutviklere enklere
- Arkitekter, ingeniører tegner først, så bygger de!
  - Enklere å endre en tegning enn et program
  - Enklere å diskutere en tegning enn et program
- UML – diagrammene er litt annerledes enn det vi har tegnet hittil (men mye av det samme)

(i UML er det ca 10 andre diagramtyper vi ikke skal lære)

13.03.2006

15

## Objekt-diagrammer

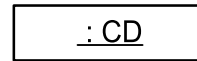
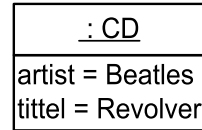
- Vi tegner en typisk situasjon av objekter i systemet vårt, når vi har fått datastrukturen på plass.
- Vi tegner og navngir bare de mest sentrale dataene som:
  - pekere
  - peker-arrayer
  - noen sentrale variable i objektene

13.03.2006

16

## 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

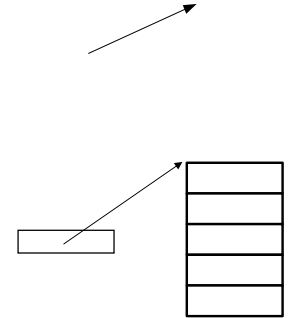


13.03.2006

17

## Andre elementer i et objektdiagram

- Pekere
- Peker-arrayer



13.03.2006

18

## Eksempel: En CD-samling

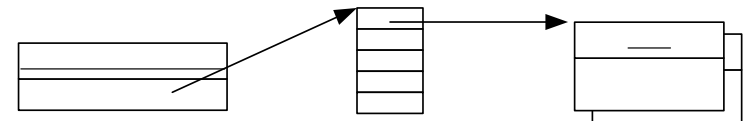
- Vi ønsker å lage et lite menystyrt program for å holde orden på CD-samlinga vår med mindre enn 1000 CDer.
- Vi skal ha funksjoner for å :
  - registrere ny CD
  - søke etter artist (dvs. skriv ut alle CD-ene til en artist)
  - skrive ut register over alle CD-ene
- Hvilke klasser har vi i dette problemet
  - Opplagt 'class CD'
  - Noen flere ?

13.03.2006

19

## Klassene: CD og CDsamling

- Vi tenker oss følgende datastruktur



- Er den tilstrekkelig?
- Vi har her forenklet programmet (klassen CDsamling forventes å inneholde main, meny-metode og sentral switch,... , mens CD inneholder utskriftsrutine) .

13.03.2006

20

minSamling : CDsamling

```

import easyIO.*;

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;

    public static void main(String args[]) {
        In tast = new In();
        Out skj = new Out();
        CDSamling e = new CDSamling();
        String a;
        CD c;
        int valg;
    }
}

```

13.03.2006

21

```

do{ skj.outln("Velg:");
    skj.outln(" 1 - les ny CD (skriv artist og CD-tittel);
    skj.outln(" 2 - skriv ut alle CD-er til gitt artist");
    skj.outln(" 3 - avslutt");
    valg = tast.inInt();

    switch(valg) {
        case 1: // les inn ny CD
            c = new CD();
            e.minSamling[e.antCDer++] = c;
            skj.out("Gi artistnavn:");
            c.artist = tast.inWord("\n");
            skj.out("Gi tittel:");
            c.tittel = tast.inWord("\n");
            break;
        case 2: // skriv data
            skj.out("Gi artistnavn:");
            a = tast.inWord("\n");
            for (int i = 0; i < e.antCDer; i++){
                if (e.minSamling[i].artist.equals(a)){
                    e.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);
}

```



## Eksempel: et helt studentregister med kurs og studenter

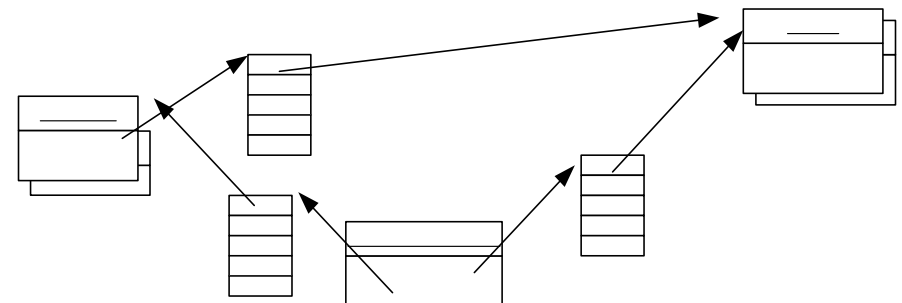
- Studenter ved Universitetet i Oslo tar som regel tre kurs hvert semester, men de kan ta flere eller færre (vi forenkler og antar at alle tar tre kurs). Vi har behov for å registrere hvilke kurs som eksisterer/går hvert semester og hvor mange studenter som tar hvert kurs.
- Vi tegner først en tenkt datastruktur
  - et UML objektdiagram
- Så skriver vi programmet

13.03.2006

23

## UML-diagram for Studentregister2

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



13.03.2006

24

```

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


    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;
    }
}

```



```

class StudentRegister2{
    public static void main(String args []) {
        String[] kurskode = {"INF1000","INF1050","MAT1030"};

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

        Student[] stud = new Student[2];
        stud[0] = new Student("Ola N", infKurs);
        stud[1] = new Student("Åsne S",infKurs);

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

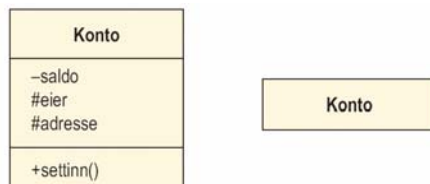
```

13.03.2006

26

## Klassediagrammer

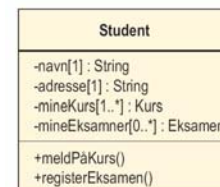
- En mer kompakt måte å tegne sammenhengen i programmet på enn objektdiagrammer
- Skiller seg fra objektdiagrammer ved at vi ikke direkte tegner datastrukturen (pekere og pekerarrayer), men bare forhold (assosiasjoner, forbindelser) mellom klassene.
- I klassediagrammer dokumenterer vi også sentrale 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().



13.03

27

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



Symboler for synlighet  
(fra resten av programmet)

+ public  
- private  
# package

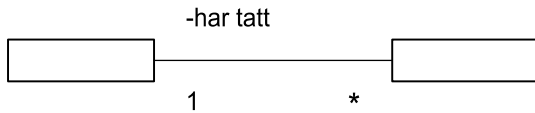
13.03.2006

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

28

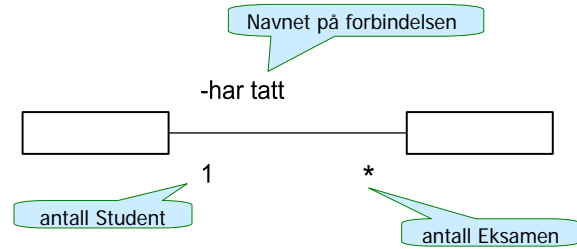
# Forhold mellom klasser

- 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



13.03.2006

29



□ En student har tatt null, en eller flere Eksamener"

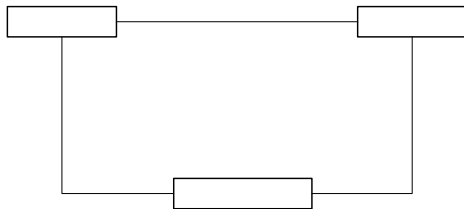
- Antallet objekter angis slik:

## Student

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

# Et litt mer komplisert studentregister

- Et studentregister holder orden på studentene og kursene. En student tar 3 kurs hvert semester
- I tillegg holder objektene i kurs rede på hvilke studenter som tar kurset

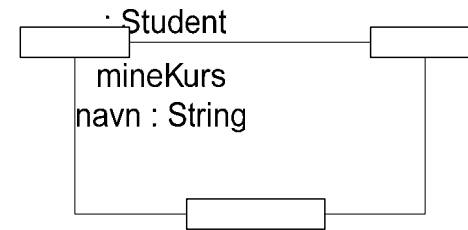
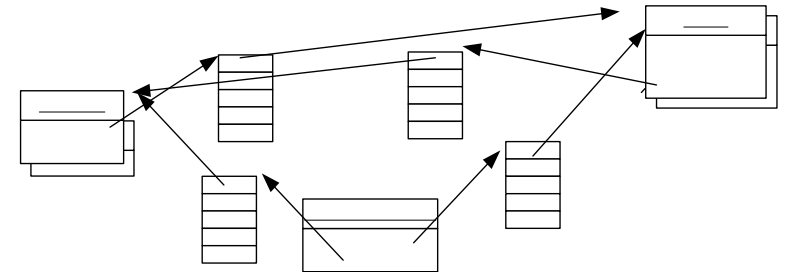


13.03.2006

31

-Tar

## Sammenligning: Objektdiagram og Klassediagram



: StudentReg

infKurs

stud



## 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 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 gjentar pkt. 1-3

13.03.2006

33

## Hvilke forhold skal vi ha med i klassediagrammet

- Hva ett objekt av den ene klassen:
  - inneholder
  - består av
  - eier,..en eller flere objekter av den andre klassen
- Der 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.

13.03.2006

34

## 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 samme beløp på saldo (tilfeldigvis)
- Vi lager objekt orienterte programmer ved å lage en modell av problemområdet i Java programmet
  - 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

13.03.2006

35

## .. 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();`

13.03.2006

36



## ... oppsummering forts.

---

- Klasser er oppskrifter på hvordan vi lager objekter med **new**
- Vi deklarerer pekere til objekter og bruker punktum .
- Kan ha arrayer av pekere til objekter
- Konstruktører er 'startmetoder' med samme navn som klassen. Konstruktørene kalles hver gang vi sier **new**.
- UML-diagrammer (Objekt- og Klasse-diagram)
  - gir oversikt og forenkling
  - som ingeniørene lager vi tegninger før vi lager systemet (programmerer)

13.03.2006

37



## Lenge til neste forelesning

---

- Uke 12: undervisningsfri / midtterm eksamener (Vi har ikke midttermeksamen i inf1000)
- 27. mars skal dere lære om HashMap som dere trenger for å løse oblig4.

13.03.2006

38



## Orakeltimer til oblig 3

---

13.03.2006

39