

## INF1000 - Forelesning 8: Objekter, klasser og pekere

2. mars 2009  
Christian Mahesh Hansen  
Institutt for informatikk, UiO

1

### Oblig 3 – to versjoner

- Normalvariant: Gulbrand Grås husleiesystem
  - Bra for å lære objektorientert programmering
  - Anbefales for de som ikke kan oo-programmering fra før
- Matematisk variant: Beregne desimaler av Pi
  - Artig for de som er glad i matte og kan oo-prog!
  - Kan ikke regne med like god hjelp på orakeltjenesten
- Kun én oppgave skal leveres!
- Les oppgaveteksten allerede nå!
- Omfattende orakeltjeneste
  - Info om tid/sted kommer om ca. én uke

2

### Objekter - motivasjon

- Vi deler ofte verden inn i enheter når vi snakker om den
  - En blomst
  - Fjorten trær
  - Ti mennesker
  - Én bil
  - Én vei
  - Mange murstein
  - En bankkonto
  - ....
- Hvorfor? For bedre å kunne tenke om, snakke om og forstå verden.

3

### Verden består av mange objekter: noen ganske like, andre ulike

- Rundt oss i auditoriet ser vi
  - Studenter, stoler, murstein, lysarmatur, blyanter, ...
- Når vi betrakter verden, deler vi den opp i et passe antall enheter/deler med egne navn
- Slike enheter kaller vi **objekter**
- Mange objekter i verden er ganske like, av samme type, men kan skille seg fra hverandre med f.eks. ulike navn
  - Studentene Kia og Espen
  - To bøker med ulik tittel/forfatter
- Objekter av samme type sier vi tilhører samme **klasse**
  - Beskrives av de samme variablene, men med *ulike* verdier på noen disse
  - Eks: to biler av samme bilmerke men med ulike reg.nummer

4

## 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.

5

## 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:
  - Beskrive problemet med veitrafikk og køer på veiene:
    - Telle antall biler og kanskje skille mellom personbiler, lastebiler og busser
    - Men neppe mer...
  - Beskrive problemet til en bilfabrikk:
    - Trenger en detaljert og komplisert beskrivelse av hver bil (bestående av motor, hjul, karosseri, lys, ..., hver beskrevet med sin klasse)
    - Mange ulike typer av biler
    - Har da en rekke klasser som hver beskriver sine objekter

6

## 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) Studentregister: bare registrere navn, personnummer, adresse, tidligere utdanning og kurs (avlagte og kurs vedkommende tar nå)
- b) Legesystem for studenter: 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

7

## Objektorientert Programmering - II

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

*Ett objekt i problemområdet (vår avgrensning av verden) skal medføre at det finnes ett objekt i programmet som representerer dette objektet*

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

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

8

## Hvordan lage klasser og objekter i et program.

- Klasser deklarerer vi med **class**
- Vi lager pekere til objekter ved å deklare 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();

    }
}
```

9

## 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 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)

10

## Peker og objekter

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

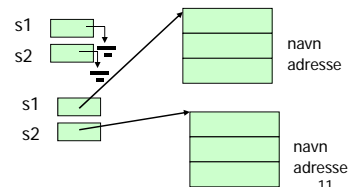
class StudentRegister {
    public static void
    main(String args []) {

        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'



11

## 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 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, ...

12

## 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";
```

13

```
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 ulike 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 ulike typer, og for hver slik type deklarerer vi en klasse i programmet

15

## .. 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();
```

16

```

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
- Eksempel: Banksystem
- Konstruktører
- eksempel med tre klasser: Student, Kurs og StudentRegister

18

## 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

19

```

class Desimaltall {
    static int i = 0;
    double x = 0.0;
}

class Start {
    public static void main(String[] args) {
        Desimaltall d1 = new Desimaltall();
        Desimaltall d2 = new Desimaltall();
        // endre klassevariable (det er bare en felles)
        System.out.println("d1.i :"+ d1.i+"", d2.i:" + d2.i);
        d1.i = 4;
        System.out.println("d1.i :"+ d1.i+"", d2.i:" + d2.i);
        // endre objektvariabel (en kopi i hvert objekt)
        System.out.println("d1.x :"+ d1.x+"", d2.x:" + d2.x);
        d1.x = 2;
        System.out.println("d1.x :"+ d1.x+"", d2.x:" + d2.x);
    }
}

```

```

>java Start
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 Start2 {
    int k; // objektvariabel 'k'
    public static void main(String[] args) {
        k = 1;
    }
}

>javac Start2.java
Start2.java:6: non-static variable k cannot be referenced from a static context
        k = 1;
        ^
1 error

class Start2 {
    int k;
    public static void main(String[] args) {
        Start2 st = new Start2();
        st.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 deklartert klassen **Kurs** {...} kan vi lage array som følger:
  - `Kurs [] ifiKurs;` Her deklarerer 'bare array-pekere'
  - `ifiKurs = new Kurs[120];` Her lages array-objektet med 120 'tomme' pekere
  - `ifiKurs[0] = new Kurs();` Her settes det første pekeren i array-objektet til å peke på et nytt Kurs-objekt

22

```
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();
        }
    }
}

arraypekeren 'ifi1000Kurs' peker til et array-objekt med 8 Kurs-pekere
```

## 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
```

24

## 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

25

## Data i Konto og Bank (en bank har mange konti)

```

class BankSystem{
    public static void main (String [] args) {
        Bank b = new Bank();
        b.navn = "BB-Bank";
        b.løkke();
    }
}

class Bank{
    Konto [] kontiene = new Konto[100000];
    int antallKonti = 0;
    String navn;

    // Metoder mangler
}

class Konto {
    String navn, adresse;
    int kontoNummer;
    double saldo =0.0;

    // Metoder mangler
}

```

Objekter av klassen Konto

Betyr at pekeren peker på ingenting: null

26

## Metoder i Bank og Konto (+ noen hjelpemetoder)

```

class Bank{
    Konto [] kontiene = new Konto[100000];
    static int kontoNummer = 500000;
    int antallKonti = 0;
    In tast = new In();
    String navn;

    double sumInnskudd() { }

    void nyKonto() { }

    int menyValg() { }

    public static void løkke (String [] args) {
        int valg =0;
        Konto k;
        double kr ;

        do {
            valg = menyValg();
            switch(valg) {
                .....
            } while (valg > 0);
        } // end main

        double spørSvar (String s){}

        Konto riktigKonto() {}
    } // end Bank

class Konto {
    String navn,adresse;
    int kontoNummer;
    double saldo =0.0;

    void settInn (double kr) {}

    boolean taUt(double kr) {}
} // end class Konto

```

```

int menyValg() {
    System.out.println(" \nVelg funksjon i "+ navn+":");
    System.out.println ("1 - ny konto:");
    System.out.println ("2 - innskudd:");
    System.out.println ("3 - uttak:");
    System.out.println ("4 - sum forvaltningskapital\n");
    return tast.inInt();
}

void løkke () {
    int valg =0;
    Konto k;
    double kr ;

    do {
        valg = menyValg();
        switch(valg) {
            case 1: nyKonto(); break;
            case 2 :k = riktigKonto();
                kr = spørSvar("Gi innskudd");
                k.settInn(kr);
                break;
            case 3 :k = riktigKonto();
                kr = spørSvar("Gi uttaksbeløp");
                if (! k.taUt(kr))
                    System.out.println("IKKE NOK PENGER");
                break;
            case 4: System.out.println(navn+
                " Sum innskudd:" + sumInnskudd());
                break;
        }
    } while (valg > 0);
    System.out.println("*** AVSLUTTER BANKEN ****");
} // end løkke

```

```

double spørSvar(String s){
    System.out.print(s+":");
    return tast.inDouble();
}

Konto riktigKonto() {
    System.out.print("Gi navn til eksisterende konto:");
    String s = tast.inWord();
    for ( int i = 0; i < antallKonti; i++)
        if (kontiene[i].navn.equals(s) )return kontiene[i];
    return null;
}

void nyKonto() {
    System.out.print("Gi navn til ny kontoinnehaver:");
    String navn = tast.inWord();
    System.out.print("Gi adresse:");
    String adr = tast.inWord();

    Konto k = new Konto();
    k.adresse = adr; k.navn = navn;
    k.kontoNummer= kontoNummer++;
    kontiene[antallKonti] = k;
    antallKonti++;
}

```

## Stringer er ordentlige objekter

- String er en klasse i Java-biblioteket, men har en egen spesiell syntaks (skrivemåte) så det ser ut som den er en av de basale typene (som int, double,...).
- Når vi har en string, har vi altså både en peker (den vi deklarerer navnet på) og et stringobjekt.
- String-objekter kan ikke endres (trenger du endrbare tekster, bruk klassen StringBuffer)
- Egen skrivemåte for stringkonstanter:
 

```
String s = "En fin dag i mai";
```

 Er det samme som:
 

```
String s = new String("En fin dag i mai");
```
- Klassen String inneholder mer enn 50 metoder for konvertering mellom ulike datatyper og tekst, samt tekstsøking.

30

## Null, && og søppeltømmingen

- 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 søppeltømmingen – et program som automatisk startes hvis det er lite plass i hukommelsen

31

## 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 parentesen 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

32

## 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++;
        }
    }
}
```

33

## 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++;
        }
    }
}
```

34

## 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 ( gjerne av en annen klasse). Da kan vi bruke this for å overføre en parameter til denne metoden om hvilket objekt kallet kom fra.

35

## Oppsummering

- Klasser er oppskrifter for hvordan vi lager objekter med **new**
- Vi deklarerer pekere til objekter og bruker punktum .
- Kan ha arrayer av pekere til objekter
- Klasse- og objektvariable og –metoder.
- Konstruktører er 'startmetoder' med samme navn som klassen, kalles hver gang vi sier **new**.

36