Kursus i OOP og Java

Sidst opdateret 24. november 2001 − kun afpudset t.o.m lektion 12

Indholdsfortegnelse

Til underviseren

Om transparenterne

Dette dokument indeholder transparenter til et aftenkursus i OOP og Java på IT−Diplomuddannelsen. Det er inddelt i 15 lektioner og følger den foreslåede lektionsplan.

Jeg har valgt at udgive dem som et samlet dokument (til Word og StarOffice Writer) fordi det gør dem nemmere at tilrette til andre undervisningsforløb. Alt i dette dokument må således frit mangfoldiggøres og tilrettes efter behov (det kunne f.eks. være at der blev brugt et andet udviklingsmiljø end JBuilder).

Visse dele er direkte udklip fra bogen. Er der andre dele eller figurer fra bogen du kunne tænke dig at få med i dine præsentationer så skriv til mig: nordfalk@mobilixnet.dk .

Transparenterne bærer naturligvis præg af min undervisningsstil − som blandt andet forudsætter at eleverne selv læser stoffet, så det er kun nødvendigt at gennemgå centrale pointer i timerne. Har du lavet nogle gode præsentationer og undervisningsmateriale som du tror andre kunne få glæde af så modtager jeg dem gerne og lægger dem på bogens hjemmeside http://javabog.dk (dette gælder også præsentationer på dansk der gør brug af andre bøger).

Visse steder kan der være problemer med Word−dokumentet. Det skyldes at det hele er skrevet i StarOffice, da Microsoft sjovt nok ikke har udgivet Word til Linux (og jeg sjovt nok aldrig har syntes en Windows−licens var pengene værd :−). I så fald kan dokumentet redigeres med StarOffice, der kan hentes gratis på www.staroffice.com.

Om ugesedlerne

Der er flere øvelser til hver uge end eleverne kan nå at lave til timerne. Underviseren kan vælge om eleverne så skal forsøge at lave dem hjemme, om de er valgfrie eller om nogle af dem skal stryges.

Jacob Nordfalk.

Lektion 1

17.00	Praktiske forhold
	Forelæsning: Hvad er et program ?
17.45	Pause
17.55	Forelæsning: Brug af JBuilder
18.20	Øvelser
20.00	Opsamling m.m.
20.15	Farvel og tak

Mål med lektion 1

basal forståelse for hvad et program er

lære at bruge JBuilder, så I kan komme i gang med at programmere

Praktisk

Har alle e−post?

Litteratur på kurset:

- J. Nordfalk: "Objektorienteret programmering i Java" (dele af den kan læses på http://javabog.dk)
- Udleverede transparenter

Hvor mange har læst?

Navnerunde

Bare spørg løs

Hvad kommer til at foregå på skolen

- Gentager ikke hvad der står i bogen
- Emner behandles på opfordring

Målet med kurset

- Lære at programmere
- Lære at programmere objektorienteret
- Lære programmeringssproget Java
- Optakt til senere fag, f.eks OOAD

Hvad er et program?

Pandekager

800 g mel
4 tsk sukker
1 tsk salt
12 æg
16 dl mælk
12 spsk vand

1. Hæld mælk i en skål
2. Hæld vand i skålen
3. Tilsæt salt
4. Tilsæt sukker
5. Så længe der er æg tilbage:
    1. Tag et æg
    2. Slå ægget i stykker
    3. Put ægget i skålen
6. Tilsæt mel
7. Rør dejen

Instruktionerne udføres en ad gangen, oppefra og ned.

Et Javaprogram: Pandekagedej vha. en robot

```
...
    double kiloMel = 0.8;
    int tskSukker = 4;
    int tskSalt = 1;
    int antalÆg = 12;
    int dlMælk = 16;
    int spskVand = 12;

    KokkeRobot.hældMælkISkål(dlMælk);
    KokkeRobot.hældVandISkål(spskVand);
    KokkeRobot.putSaltISkål(tskSalt);
    KokkeRobot.putSukkerISkål(tskSukker);
...
```

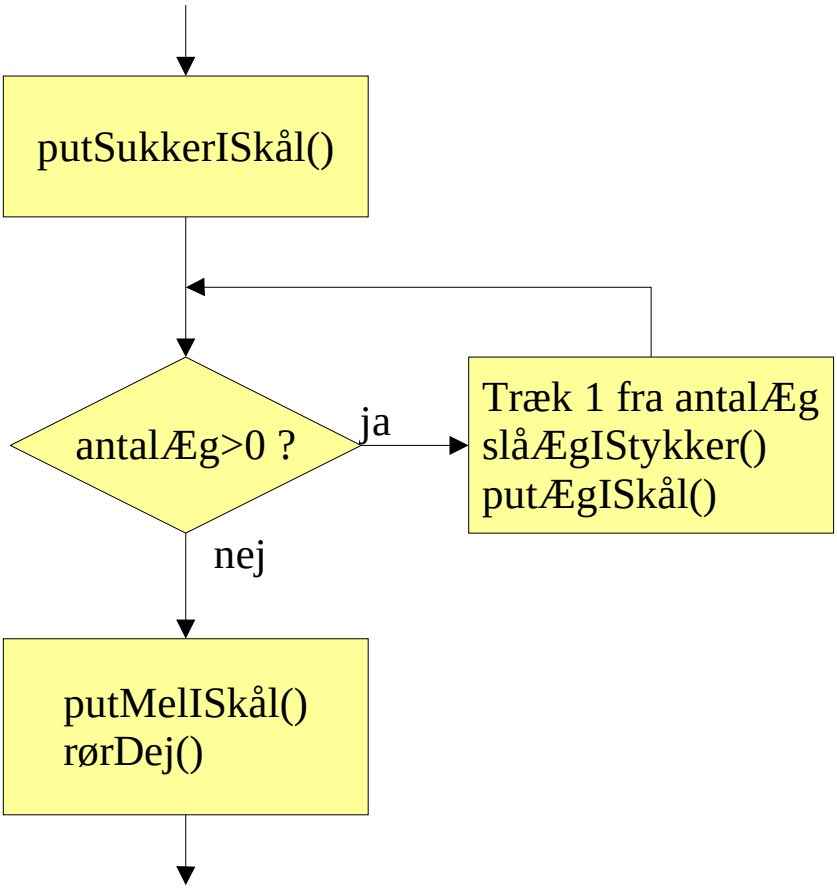
Variabler skal erklæres før de må bruges

Programmet udføres sekventielt (oppe fra og ned)

Når computeren skal gøre komplicerede ting

```
...
    KokkeRobot.putSukkerISkål(tskSukker);

    while (antalÆg > 0)
    {
      antalÆg = antalÆg – 1;
      KokkeRobot.slåÆggetIstykker();
      KokkeRobot.put1ÆgISkål();
    }

    KokkeRobot.putMelISkål(kiloMel);
    KokkeRobot.rørDejen();
...
```

Foretager valg ud fra simple logiske udsagn
Simple byggeklodser til at vælge handlinger



while−sætningen udfører en blok af kommandoer indtil udsagnet er falskt. Derefter fortsættes programmet

if−sætning udfører kun kommandoerne, hvis udsagnet er sandt

Hvad er et Javaprogram

Fil med endelsen ’.java’

kildekoden har en strukturdel:

```
public class Pandekager
{
  public static void main(String[] arg)
  {
    ...
  }
}
```

og noget indhold − en kogebogsopskrift − kommandoer der skal udføres:

```
...
    KokkeRobot.hældMælkISkål(dlMælk);
    KokkeRobot.hældVandISkål(spskVand);
    KokkeRobot.putSaltISkål(tskSalt);
    KokkeRobot.putSukkerISkål(tskSukker);
...
```

Til sammen en hel java−fil:

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

    double kiloMel = 0.8;
    int tskSukker = 4;
    int tskSalt = 1;
    int antalÆg = 12;
    int dlMælk = 16;
    int spskVand = 12;

    KokkeRobot.hældMælkISkål(dlMælk);
    KokkeRobot.hældVandISkål(spskVand);
    KokkeRobot.putSaltISkål(tskSalt);
    KokkeRobot.putSukkerISkål(tskSukker);

    while (antalÆg > 0)
    {
      antalÆg = antalÆg − 1;
      KokkeRobot.slåÆggetIstykker();
      KokkeRobot.put1ÆgISkål();
    }

    KokkeRobot.putMelISkål(kiloMel);
    KokkeRobot.rørDejen();
  }
}
```

Sproget Javas regler

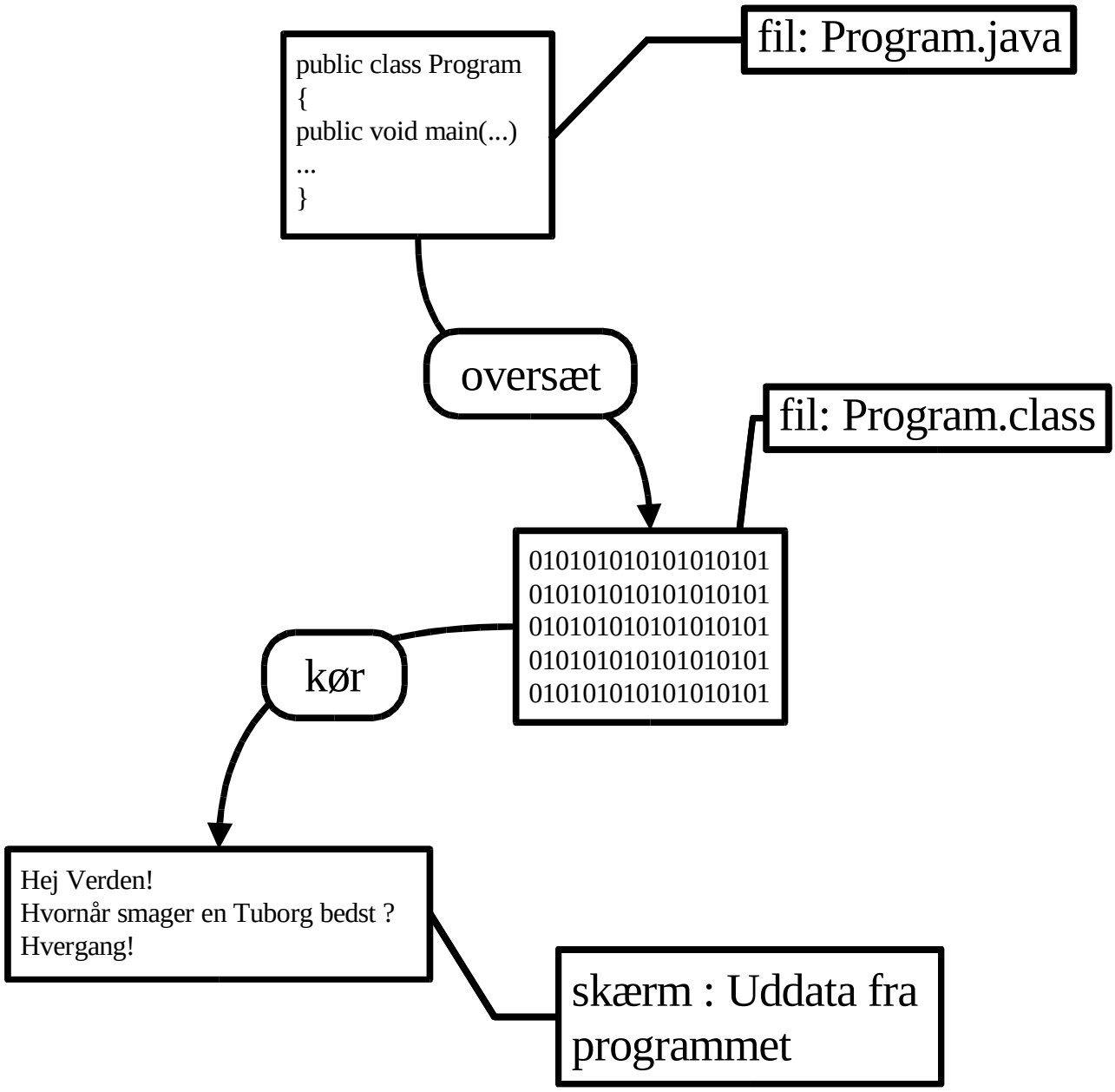
Struktur: Noget kan indeholde noget andet

- en klasse kan indeholde en main−metode
- en metode kan indeholde ’sætninger’ (kommandoer)

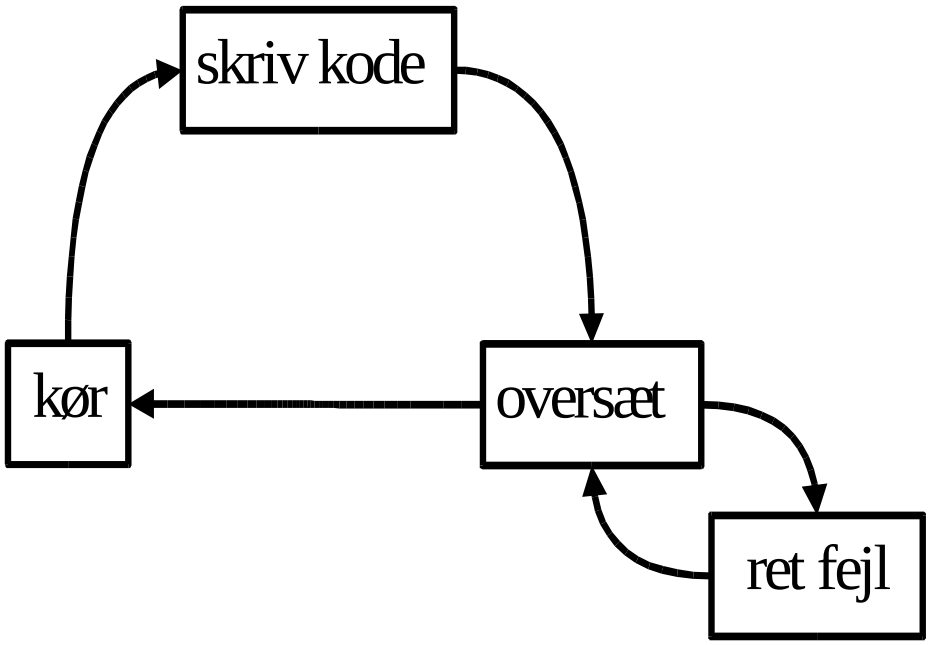
Hver ’sætning’ skal ses isoleret

```
...
    KokkeRobot.hældMælkISkål(dlMælk);
    KokkeRobot.hældVandISkål(spskVand);
...
```

Hvordan køres et javaprogram



Programmørens arbejdsgang

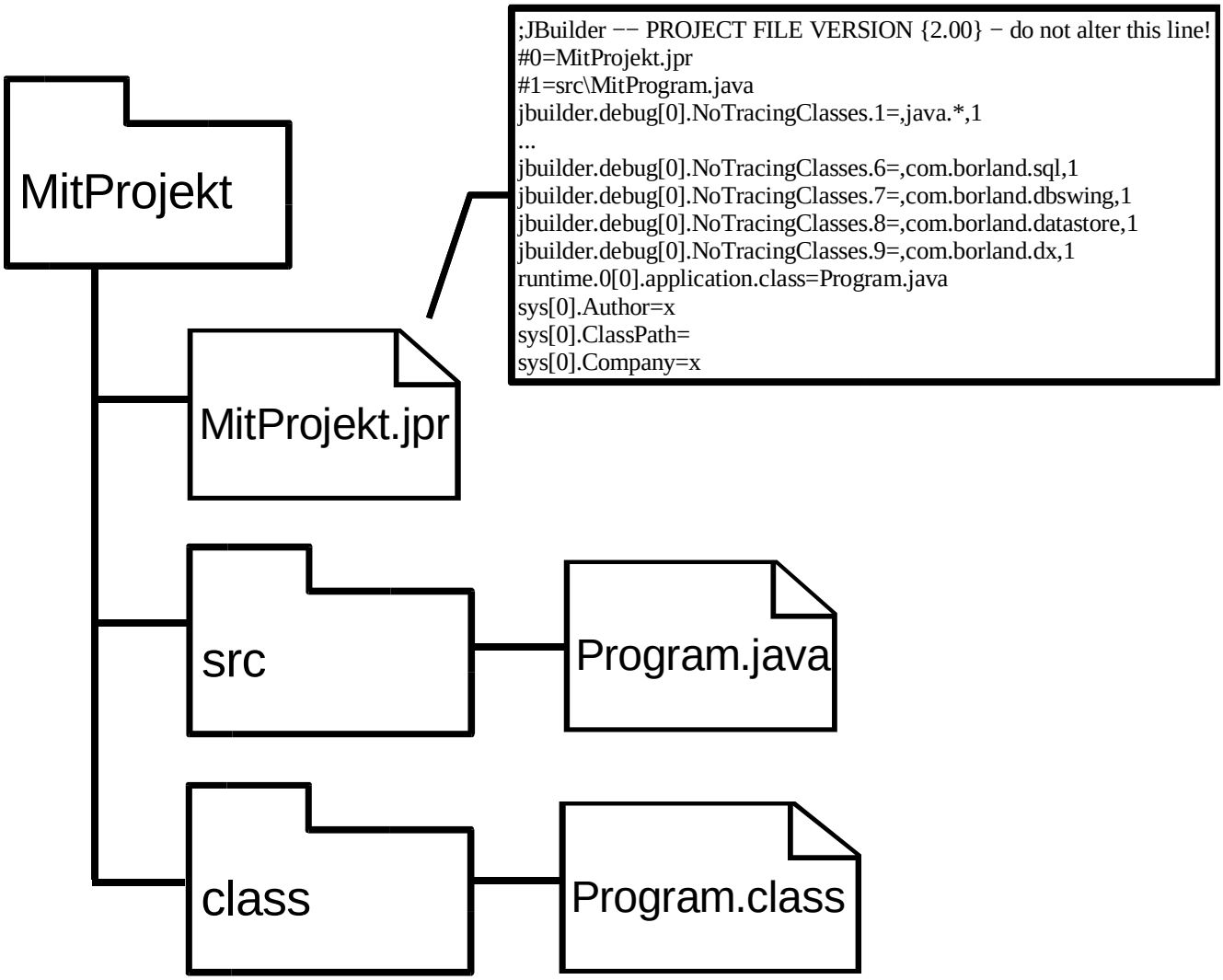


Udviklingsværktøjet (JBuilder)

- rette i programmer
- oversætte java−filer
- køre javaprogrammer
- finde fejl (bl.a. med trinvis gennemgang af programmer)

Projekter

- et projekt organiserer filer der hører sammen
- hvert program i sit projekt selvom programmerne er små
- holder rede på egenskaber for programmet
- ikke det samme som et katalog
- projekter ligger ofte i C:\Jbproject



Øvelserne

JBuilder skal registreres: Skriv jeres navn og tryk add.. og indtast serienummeret.

Sæt jer 2 og 2 ved maskinerne

Senere 1 og 1

Snak gerne med naboen
... han har måske lige løst det problem du sidder med

Opsamling

læs hjemme: resten af kapitel 2, undtagen ’Avanceret’

noter afsnit, der er svære at forstå

noter tidsforbrug

brug ’Test dig selv’

installer JBuilder

prøv eksemplerne fra kapitel 2

problemer: skriv per e−post

Lektion 2

17.00	Resumé af stoffet
17.15	Praktiske oplysninger og spørgsmål
17.30	Model for beregninger
17.40	Pause
17.55	Hvordan bliver et program til ?
	Introduktion til for−projekt
18.15	Individuelle øvelser ved maskinerne
19.00	Opsamling og evaluering af løsninger
	Lidt om det rigtige projekt

Praktisk

Spørgsmål til kapitel 1 og 2

Hvem har læst ?

Hvem har installeret JBuilder ?

Hvem har programmeret derhjemme ?

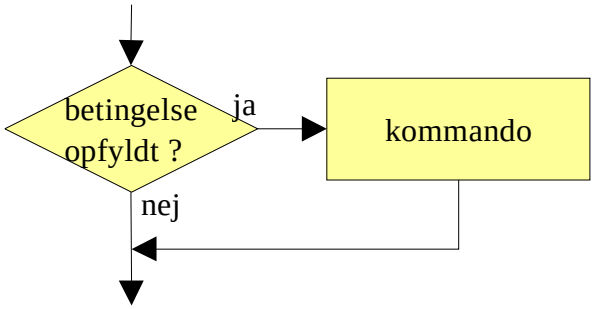
Mål med lektion 2

- Fortrolighed med løkker
- Programmere lidt selv
- I gang med for−projektet

Resumé

Betinget udførelse

```
if (betingelse) kommando;
```

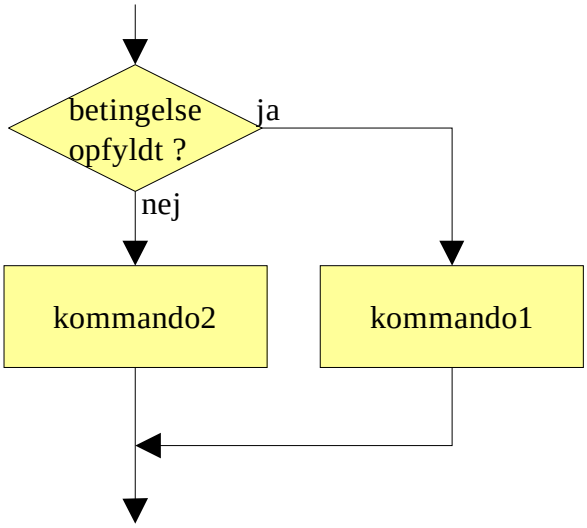


eksempler:

```
if (alder >= 18) System.out.println("Du er myndig");

if (alder == 18) System.out.println("Du er præcis atten");

if (alder != 18) System.out.println("Du er ikke atten.");
```

if−else:

```
if (betingelse) kommando1;
else kommando2;
```

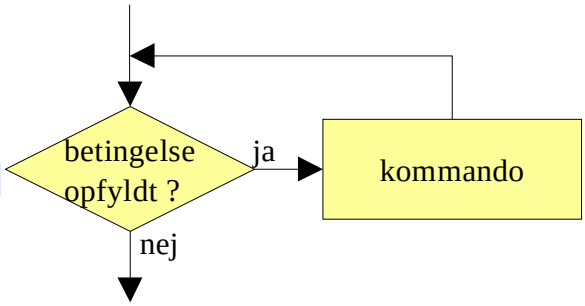


eksempel, hvor vi også bruger en blok:

```
if (alder >= 18) System.out.println("Du er myndig");
else
{
  System.out.println("Du er kun " + alder + " år.");
  System.out.println("Du er ikke myndig");
}
```

while−løkken

```
while (betingelse) kommando;
```

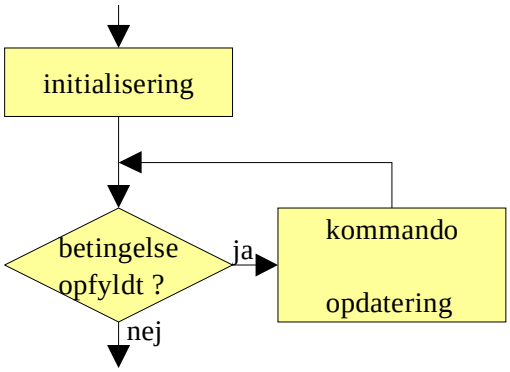



```
public class Syvtabel
{
  public static void main(String[] args)
  {
    int n;
    n = 1;

    while (n <= 10)
    {
      System.out.println(n+" : "+ 7*n);
      n = n + 1;
    }
  }
}
```

for−løkken

```
for (initialisering; betingelse; opdatering) kommando;
```

```
public class Syvtabel2
{
  public static void main(String[] args)
  {
    for (int n=1; n<=10; n=n+1)
      System.out.println(n+" : "+ 7*n);
  }
}
```



De simple typer

Type	*Art*	*Antal bit*	*Mulige værdier*	*Standard værdi*
byte	heltal	8	−128 til 127	0
short	heltal	16	−32768 til 32767	0
int	heltal	32	−2147483648 til 2147483647	0
long	heltal	64	−9223372036854775808 til 9223372036854775807	0
float	kommatal	32	±1.40239846E−45 til ±3.40282347E+38	0.0
double	kommatal	64	±4.94065645841246544E−324 til ±1.79769313486231570E+308	0.0
char	unicode	16	\u0000 til \uffff (0 til 65535)	\u0000
boolean	logisk	1	true og false	false

Konvertering

- automatisk hvor intervallet af de mulige værdier udvides
- eksplicit hvis intervallet af de mulige værdier indskrænkes

```
int x;
double y;

x = 15;       // OK
x = 15.1;     // Fejl
x = 15.0;     // Fejl

y = x;        // OK – implicit konvertering

x = y;        // Fejl

x = (int) y;  // OK – eksplicit konvertering
```

Model for udregninger, før kørsel

```
int x;
double y;

x=3;
y=2.9;

x = (int) ( x + y + x / 4 );
```

Regnestykket deles op i delberegninger

```
delresultat1 = x + y;

delresultat2 = x / 4;

delresultat3 = delresultat1 + delresultat2;

delresultat4 = (int) delresultat3;

x = delresultat4;
```

Der sættes type på delresultaterne

```
double delresultat1;

int delresultat2;

double delresultat3;

int delresultat4;
```

Typer på mellemresultater, værdier og variable ligger fast, når programmet er oversat

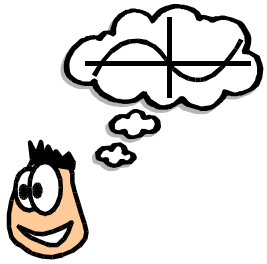
Model for udregninger – under kørsel

Udregningerne foretages og gemmes undervejs i delresultater

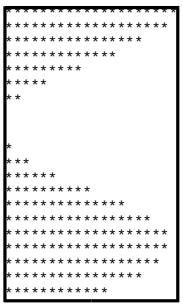
Delresultaterne smides væk

Hvordan bliver et program til ?

Et program til at tegne kurver



Idé



Konkret beskrivelse

```
x sættes til intervalstart
sålænge x < intervalslut
    beregn funktionens værdi for x
    skaler værdien til hvor mange '*'
    gennemløb så mange gange:
        udskriv '*'
    udskriv linieskift
    tæl x op
```

Pseudokode

Java program

Program

For−projekt

Formål:

- At komme i gang med en programmeringsopgave
- Få noget øvelse
- Motivere til at programmere derhjemme
- Evt. idé til projekt

På baggrund af erfaringer:

- Folk har ikke tid
- Folk kommer for sent i gang
- Folk mangler programmeringserfaring når de begynder projektet

Oplæg til forprojekt:

> **Du skal i løbet af de næste par øvelsesgange lave et program til at vise kurver og lave beregninger med**

Løsningerne skal præsenteres for en anden gruppe

Som lektionerne skrider frem putter vi mere på opgaven

Brug ’*’ til visning. Næste gang laver vi det grafisk

Afleveres til lektion 6

Tag projektet med hver gang på diskette. Hvis du ikke har en diskette med i dag, så skriv programmet ud, og tast det ind igen derhjemme.

Opsamling

Find en gruppe og evaluér sammen med. Præsentér jer for hinanden.

Læs til og med 3.5.1 til næste gang (også gerne resten)

Tænk på nogle ideer til OOP−projektet

Lektion 3

17:00	Praktisk
17:15	Et eksempel på et Kurveprogram
17:40	Kort om appletter
17:45	Pause
18:00	Objekter, klasser og Vector
18:45	Individuelle øvelser
20:15	Direkte hjem

En løsning på kurveprogrammet

```
public class VisGraf
{
  public static void main(String[] args)
  {
    double istart, islut;
    double skalering, forskydning;
    int antalLinier;

    istart=0;
    islut=10;
    skalering=10;
    forskydning=10;
    antalLinier=20;

    double trin; // hvor meget x skal tælles op med
    trin=(islut - istart)/antalLinier;

    int linienr;
    linienr=0;
    while (linienr < antalLinier)
    {
      double xVærdi;
      double funktionsVærdi;

      xVærdi=linienr*trin + istart;
      funktionsVærdi=Math.cos(xVærdi);

      int antalstjerner;
      antalstjerner=(int) (funktionsVærdi*skalering +forskydning);

      int kolonnenr;
      kolonnenr=0;
      while (kolonnenr < antalstjerner)
      {
        System.out.print("*");
        kolonnenr=kolonnenr+1;
      }
      System.out.println();
      linienr=linienr+1;
    }
  }
}
```

Udvidelser:

- max på kurven udregnes
- kurven tegnes u−udfyldt

```
double max;
max=-200000000000.0;

int linienr;
linienr=0;
while (linienr < antalLinier)
{
  double xVærdi;
  double funktionsVærdi;

  xVærdi=linienr*trin;
  funktionsVærdi=Math.cos(xVærdi);
  if (funktionsVærdi>max) max=funktionsVærdi;

  int antalstjerner;
  antalstjerner=(int)
     (funktionsVærdi*skalering + forskydning);
  int kolonnenr;
  kolonnenr=0;
  while (kolonnenr < antalstjerner-1)
  {
    System.out.print(" ");
    kolonnenr=kolonnenr+1;
  }
  if (antalstjerner>0) System.out.print("*");
  System.out.println();
  linienr=linienr+1;
}
System.out.println("max: "+max);
```

Resumé af objekter og klasser

En klasse ligger som en del af programmet i en fil
Findes inden programudførslen

Et objekt eksisterer kun når programmet kører
Når programudførslen kommer til et 'new'

En klasse er en beskrivelse af, hvordan de objekter der bliver skabt med denne klasse opfører sig.

F.eks. er klassen Point en beskrivelse af hvordan objekter skabt med 'new Point' er.

En klasse svarer til en idé, en forklaring eller en definition af hvad noget er. Et objekt er så en konkret forekomst af det 'noget'.

Quiz: Punkter

Hvad tror du en klasse er ?

Hvad tror du et objekt er ?

klassenavn: **Point**
variabler: x y
metoder: move(x,y) translate(x,y) Point() Point(x,y)

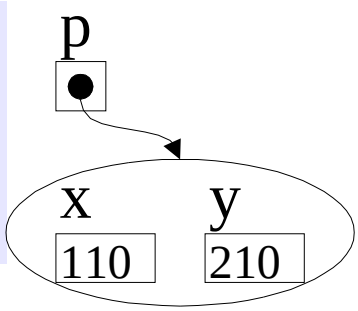
Løsning

Klasse:
"Et punkt er to heltal samlet i én komponent"

Objekter:
(1,0) (2,6) (13,2) (110,210) ...

For at oprette objekter i hukommelsenbruger man ’new’:

```
Point p;

p = new Point();

p.x = 110;
p.y = 210;
```

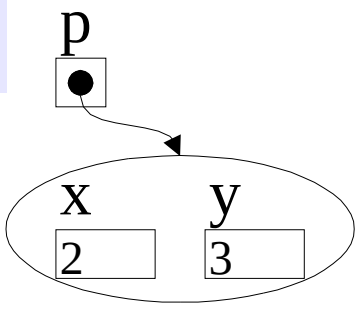


Der findes flere måder at oprette objekter på − defineret ved deres konstruktører

```
Point p2;

p2 = new Point(3,0);
```

Objekter har metoder der kan kaldes for at arbejde med dem

```
p.move(2,3);
```



Klassen String

```
public class Strengeleg
{
  public static void main(String[] args)
  {
    String s;
    s = "Ude godt";
    System.out.println("Strengen s indeholder: "+s);
    System.out.println("s er "+s.length()+" tegn lang");
    System.out.println("s med store bogstaver: "+s.toUpperCase());
    System.out.println("Tegn på plads nummer 2 er: "+s.charAt(2));
    System.out.println("Første g er på plads: "+s.indexOf("g"));
  }
}
```

```
Strengen s indeholder: Ude godt
s er 8 tegn lang
s med store bogstaver: UDE GODT
Tegn på plads nummer 2 er: e
Første g er på plads nummer: 4
```

Streng−objekterne ændrer sig aldrig, men gir nye strenge retur:

```
    s1 = "Ude godt, men hjemme bedst.";
    s2 = s1.toUpperCase();
    s3 = s2.replace('E', 'X');
    s4 = s3.substring(4, 20);
```

men vi kan naturligvis godt ændre variablerne:

```
    s1 = s1.toUpperCase();
```

s1 refererer nu til "UDE GODT, MEN HJEMME BEDST."

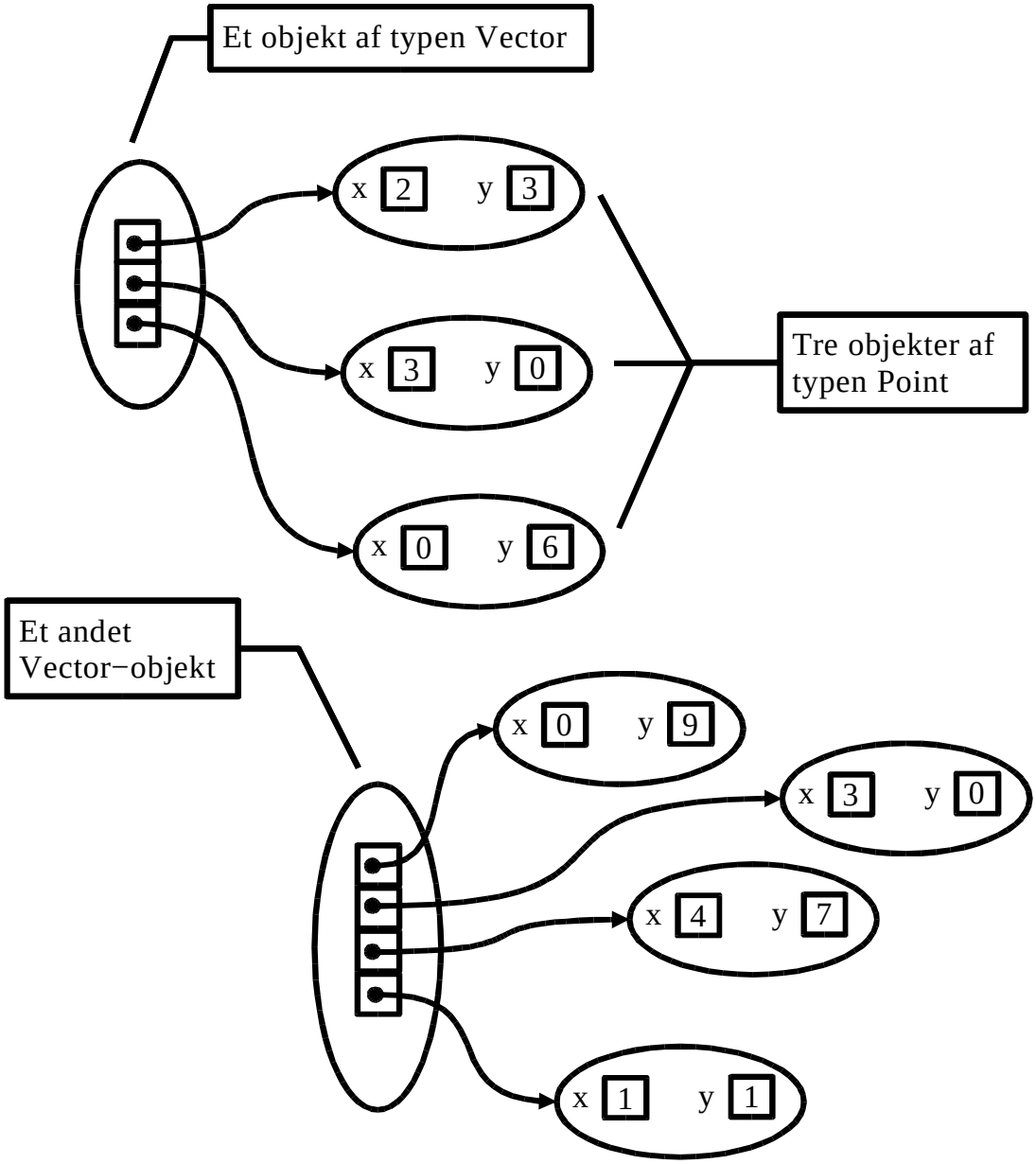
Klassen Vector

Når man har opretter et objekt af typen Vector, kan man bruge det som en variabel, der indeholder andre objekter.

F.eks. kan man oprette et Vector−objekt og lægge nogle objekter af type Point ind i det.

```
Vector v = new Vector();

v.addElement(p1);
v.addElement(p2);
v.addElement(new Point(0,6));
```

Illustration af computerens hukommelse mens et program der bruger Vector og Point kører:



Objekterne inde i et Vector−objekt kaldes Vector−objektets elementer.

Et Vector−objekt indeholder nummererede variable, der refererer til elementerne.

```
Point p;
p = (Point) v.elementAt(2);
```

Det er nødvendigt at lave en typekonvertering af elementet, når det skal gemmes i en variabel.

Med size() kan man se hvor mange elementer et Vector−objekt indeholder. Det kan bruges til at gennemløbe elementerne i en vektor:

```
for (int n=0; n<v.size(); n=n+1)
{
  Point p;
  p = (Point) v.elementAt(n);
  ...
}
```

Appletter

Småprogrammer, der skal ligge i en hjemmeside.

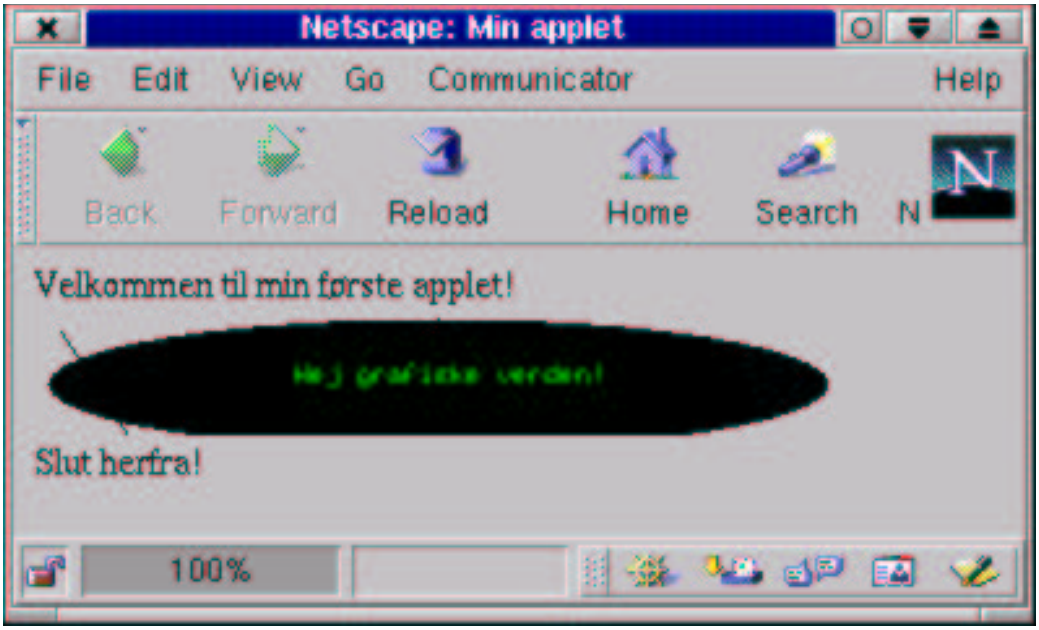
Ikke ligesom almindelige Java−programmer:
De har ingen main()
Anden struktur:

```
public class MinApplet extends Applet
{
   public void paint(Graphics g)
   {
     // g referer til et Graphics−objekt man kan tegne med.
     ...

   }
}
```

paint() udføres ikke kun når apletten starter, men hver gang appletten skal gentegnes (f.eks hvis vinduet har været skjult)

I paint() kan man sende kommandoer til Graphics−objektet:

```
   g.drawLine(10,10,50,70);

   g.fillOval(5,5,300,50);

   g.setColor(Color.green);

   g.drawString("Hej grafiske verden!",100,30);
```



Lektion 4

Praktisk

Fra UML−klassediagram til Java

Pause

Inspiration til for−projektet

Øvelser

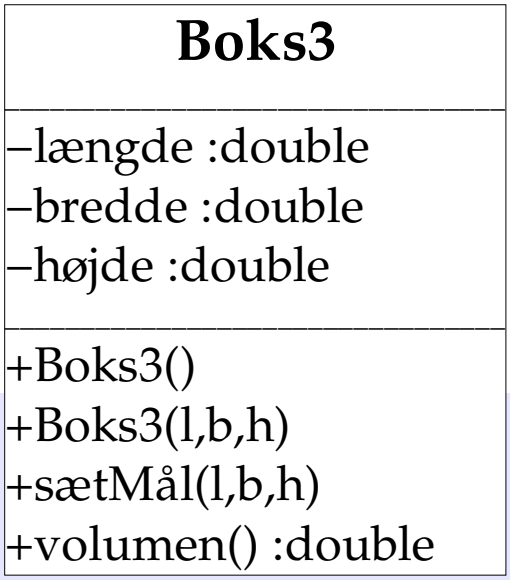
Praktisk

Hvem har arbejdet med for−projektet siden sidste gang ?

Hvem har fået lavet en grafisk kurvetegner−applet ?

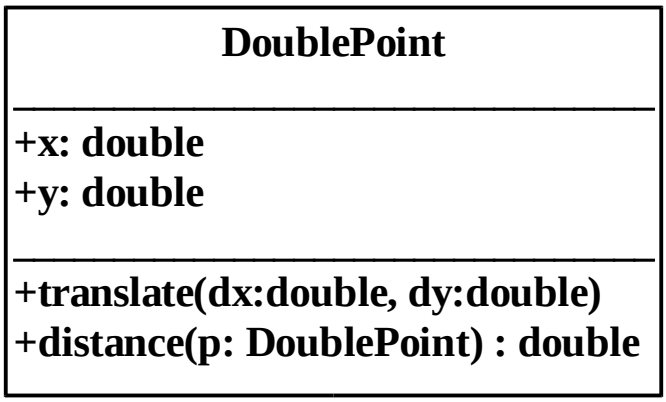
Spørgsmål til det læste

Fra UML−klassediagram til Java




```
public class Boks3
{
  private double længde;
  private double bredde;
  private double højde;

  public Boks3()
  {
    System.out.println("Standardboks oprettes");
    sætMål(10, 10, 10);
  }

  // En anden konstruktør der tager bredde, højde og længde
  public Boks3(double l, double b, double h)
  {
    System.out.println("Boks oprettes med l="+l+" b="+b+" h="+h);
    sætMål(l,b,h);
  }

  public void sætMål(double b, double h, double l)
  {
    if (l<=0 || b<=0 || h<=0)
    {
      System.out.println("Ugyldige mål. Bruger standardmål.");
      længde = 10.0;
      bredde = 10.0;
      højde  = 10.0;
    } else {
      længde = l;
      bredde = b;
      højde  = h;
    }
  }

  public double volumen()
  {
    return længde*bredde*højde;
  }
}
```

Prøv at lave din egen klasse



DoublePoint er en klasse, der bruges fra main()

```
public class DoublePoint
{
   public double x;
   public double y;

   public void translate(double dx, double dy)
   {
     ...
   }

   public double distance(DoublePoint p)
   {
      ...
   }

}
```

\+ bliver til public

− bliver til private

Typer står efter i UML − før i Java

Metodekroppen af translate()

Hvad skal den gøre ?

Ændre koordinaterne for objektet selv ?

Lave nyt DoublePoint−objekt ?

Tænk over hvordan den skal bruges

```
DoublePoint p;
P = new DoublePoint(2.5,3.4);

p.translate(1.0,1.0);
```

Metodekroppen af distance()

Hvad skal den gøre ?

Ændre koordinaterne for objektet selv ?

Lave nyt DoublePoint−objekt ?

Hvordan gives informationen tilbage til kalderen ?

Tænk over hvordan den skal bruges

```
DoublePoint p1, p2;
P1 = new DoublePoint(2.5,3.4);
p2 = new DoublePoint(20,10);

double afstand;
afstand = p1.distance(p2);
```

Man kan se udfra, hvordan den tænkes brugt, hvordan konstruktøren skal se ud

DoublePoints Konstruktører

Skal sætte alle objektets variabler passende

```
...
public DoublePoint(double initX, double initY)
{
   x=initX;
   y=initY;
}
...
```

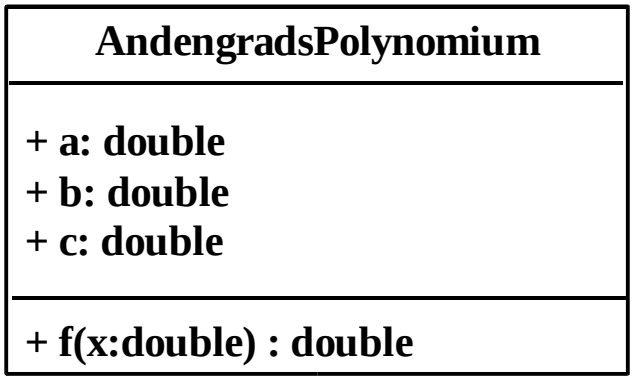
DoublePoints konstruktører skal naturligvis ligge i klassen DoublePoint

Godt råd: kald IKKE paramteren i konstruktøren det samme som objektvariablen

Inspiration til for−projekt: Polynomier

F.eks

f(x) = a*x*x + b*x + c



Eksempel på brug:

```
Andengradspolynomium p;
p = new Andengradspolynomium(1,−0.5,1); // 1*x*x−0.5*x+1
double y;
y = p.f(3);
```

Næste trin

Generelt polynomium: f.eks. $f(x) = 8x^3 + 10x^2 + 3x^1 + 12$

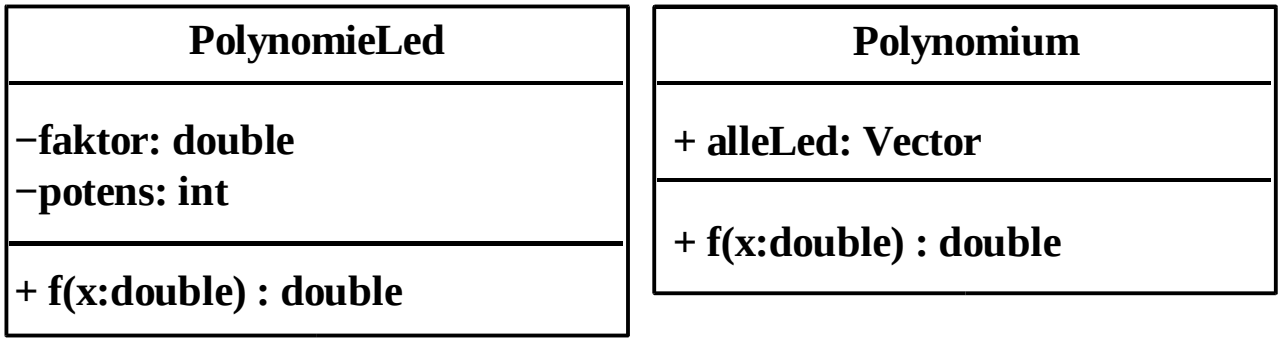
Polynomieled: f.eks. " $10x^2$ "

Polynomieled: y = faktor*xpotens

```
y = faktor*Math.pow(x,potens);
```

Polynomie består af en liste af Polynomieled

Klasser



Eksempel på brug:

```
double y;
PolynomieLed led;
led = new PolynomieLed(2.0,2);
y = led.f(3);
```

```
Polynomium pol;
pol = new Polynomium();

pol.alleLed.addElement(led);

led = new PolynomieLed(10.0,0);
pol.alleLed.addElement(led);

y =  pol.f(3);
```

pol er nu polynomiet $f(x) = 2x^2 + 10x^0 = 2x^2 + 10$

Husk at vektoren ’alleLed’ i Polynomium skal oprettes med
’new Vector()’ i konstruktøren

Lektion 5

17:00	Praktisk
17:10	Opsamling: Egne klasser, objekter og relationer
17:45	Test dig selv af kapitel 3 og 4
18:00	Pause
18:10	Øvelser fra sidste gang
18:15	Projektet
18:25	Vi går til i øvelseslokalet
20:15	Slut på undervisning

Praktisk

Hvem har arbejdet med for−projektet siden sidste gang ?

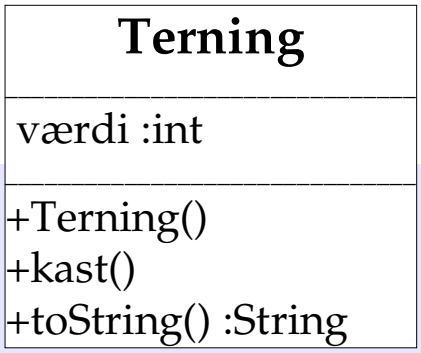
Hvem har lavet polynomiumsled ?

Hvem har styr på Vector ?

Spørgsmål

Egne klasser og objekter

(fra bogen)




```
public class BenytTerning
{
  public static void main(String args[])
  {
    Terning t;
    t = new Terning();  // opret terning

    // Slå nu med terningen indtil vi får en sekser
    boolean sekser = false;
    int antalKast = 0;

    while (sekser==false)
    {
      t.kast();
      antalKast = antalKast + 1;
      System.out.println("kast "+antalKast+": "+t.værdi);
      if (t.værdi == 6) sekser = true;
    }

    System.out.println("Vi slog en 6'er efter "
      +antalKast+" slag.");
  }
}
```

```
kast 1: 4
kast 2: 2
kast 3: 6
Vi slog en 6'er efter 3 slag.
```

Klassen Terning

(fra bogen)

```
// En klasse der beskriver 6-sidede terninger
public class Terning
{
  // den side der vender opad lige nu
  int værdi;

  // konstruktør
  public Terning()
  {
    kast(); // kald kast() der sætter værdi til noget fornuftigt
  }

  // metode til at kaste terningen
  public void kast()
  {
    // find en tilfældig side
    double tilfældigtTal = Math.random();
    værdi = (int) (tilfældigtTal * 6 + 1);
  }

  // giver en beskrivelse af terningen som en streng
  public String toString()
  {
    String svar = ""+værdi;  // værdi som streng, f.eks. "4"
    return svar;
  }
}
```

Formen af klasser

(fra bogen)

```
// Klassenavn skal være i filen Klassenavn.java

import klasser;   // f.eks. import java.util.*;
...

public class Klassenavn
{
  // mellem { og } skal definitionen af klassen stå:

  // erklæring af variabler (og evt. samtidig initialisering)
  synlighed type   navnPåObjektvariabel;
  public    int    n;
  private   String s;
  private   String s2 = "goddag"; // samtidig initialisering


  // erklæring af konstruktører, evt. med parametre
  synlighed Klassenavn(type1 parameter1, type2 parameter2, ...)
  {
    // kode der sætter objektvariablerne til startværdier
    ...
  }

  // eksempler på konstruktører:
  public Klassenavn()
  {
    n = 5;
    s = "hej";
  }

  public Klassenavn(int nn, String ss)
  {
    n = nn;
    s = ss;
  }


  // erklæring af metoder, evt. med parametre
  synlighed returtype metodenavn(type1 param1, type1 param2, ...)
  {
    ...
  }

  // eksempel:
  public int metode1()
  {
    ...
    return 15; // noget af type int
  }
}
```

Formen af en metode

Metoder består af et hoved:

```
public int metode1()
```

og en krop:

```
{
  type1 lokalVariabel1;
  type2 lokalVariabel2;
  ...
  // programkode her
}
```

Metodehovedet

Metodehovedet har formen:

```
synlighed returtype metodenavn(type1 param1, type1 param2, ...)
```

Eksempler:

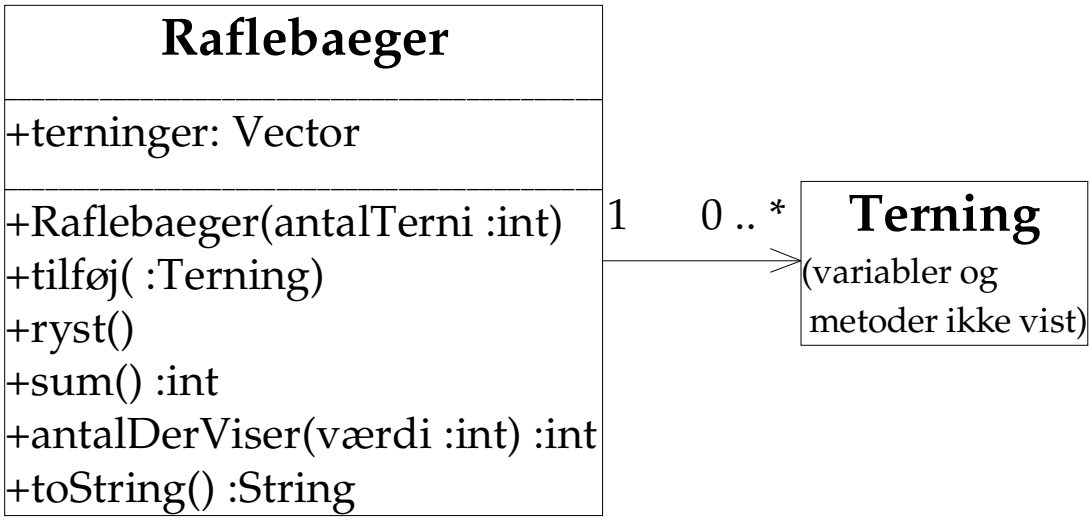
```
double volumen()

public void sætMål(double b, double h, double l)

public void tilføj(Terning t)

public void kast()

public int antalDerViser(int værdi)
public void præsentation()
public Konto(Person ejer)
public void overførsel(int kroner)
public String toString()
```

Metodekroppen

```
public int metode3()
{
  ...
  return 42;
}
```

Relationer mellem egne klasser

Et objekt kan have andre objekter i sig (en **har**−relation)




```
public class ToSeksere
{
  public static void main(String[] args)
  {
    Raflebaeger bæger;
    boolean toSeksere;
    int antalForsøg;

    bæger = new Raflebaeger(3);   // opret et bæger med 3 terninger
    toSeksere=false;
    antalForsøg = 0;
    while (toSeksere==false)
    {
      bæger.ryst();               // kast alle terningerne
      System.out.print("Bæger: " + bæger + " sum: " + bæger.sum());
      System.out.println(" Antal 6'ere: "+bæger.antalDerViser(6)
                         + " antal 5'ere: "+bæger.antalDerViser(5));
      if (bæger.antalDerViser(6) == 2)
      {
        toSeksere = true;
      }
      antalForsøg++;
    }
    System.out.println("Du fik to seksere efter "
      + antalForsøg+" forsøg.");
  }
}
```

```
Bæger: [4, 4, 4] sum: 12 Antal 6'ere: 0 antal 5'ere: 0
Bæger: [5, 5, 6] sum: 16 Antal 6'ere: 1 antal 5'ere: 2
Bæger: [6, 4, 1] sum: 11 Antal 6'ere: 1 antal 5'ere: 0
Bæger: [6, 6, 4] sum: 16 Antal 6'ere: 2 antal 5'ere: 0
Du fik to seksere efter 4 forsøg.
```

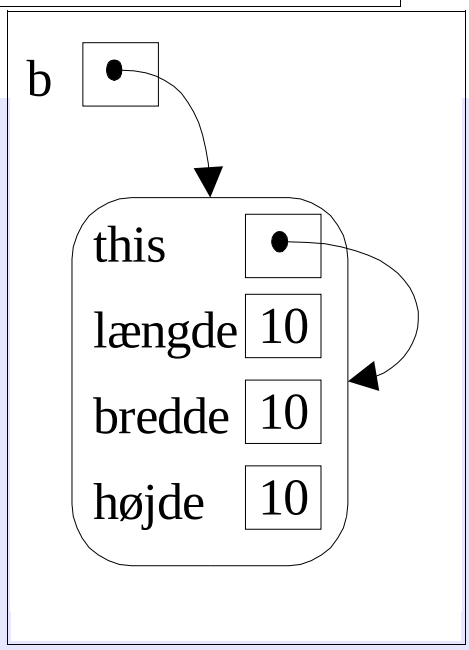
```
import java.util.*;
public class Raflebaeger
{
  public Vector terninger;  // Rafleb. har en vektor af terninger

  public Raflebaeger(int antalTerninger)
  {
    terninger = new Vector();
    for (int i=0;i<antalTerninger;i++)
    {
      Terning t = new Terning();
      tilføj(t);
    }
  }

  public void tilføj(Terning t) // Læg en terning i bægeret
  {
    terninger.addElement(t);
  }

  public void ryst()          // Kast alle terningerne
  {
    for (int i=0;i<terninger.size();i++)
    {
      Terning t = (Terning) terninger.elementAt(i);
      t.kast();
    }
  }

  public int sum()            // Summen af alle terningers værdier
  {
    int resultat = 0;
    for (int i=0;i<terninger.size();i++)
    {
      Terning t = (Terning) terninger.elementAt(i);
      resultat = resultat + t.værdi;
    }
    return resultat;
  }

  public int antalDerViser(int værdi) // Antal med denne værdi
  {
    int resultat = 0;
    for (int i=0;i<terninger.size();i++)
    {
      Terning t = (Terning) terninger.elementAt(i);
      if (t.værdi==værdi) resultat = resultat + 1;
    }
    return resultat;
  }
}
```

Variablen terninger refeterer til nogle Terning−objekter

Derfor relationen: Raflebaeger **har−en** Terning

Nøgleordet this

this refererer til det objekt man er i




```
public class Boks2medThis
{
  private double længde;
  private double bredde;
  private double højde;

  // næsten som om der her stod
  // private Boks2medThis this;

  public void sætMål(double længde,
             double bredde, double højde)
  {
    this.længde = længde;
    this.bredde = bredde;
    this.højde  = højde;
  }

  public void tilføjTilVektor(Vector v)
  {
    v.addElement(this);
  }

  ...
}
```

this virker som en variabel der refererer til objektet selv.

```
  Vector v = new Vector()
  Boks2medThis b = new Boks2medThis();
  ...
```

Normalt ville vi tilføje en boks til en vektor med:

```
  v.addElement(b);
```

Med metoden tilføjTilVektor() kan vi i stedet for bede b om at tilføje sig selv til en vektor:

```
  b.tilføjTilVektor(v);
```

Test dig selv kapitel 3 og 4

For−projektet

I opgaven skal I lave objekter der har relationer til hinanden.

For eksempel et polynomium−objekt der har nogle led−objekter:

Polynomium er f.eks. $f(x) = 8x^3 + 10x^2 + 3x^1 + 12$

Polynomieled er f.eks. $10x^2$

(men senere mere generelle funktioner!)

Husk når I laver jeres program:

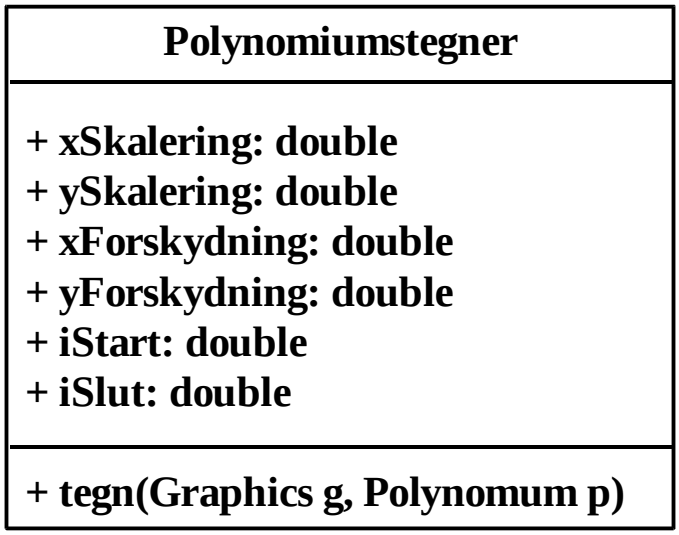
Programmets kildetekst skal være let at forstå (for en anden person på kurset)

Du skal definere tre klasser

Rapporten kan være et diagram over klasserne og en kort forklaring.

Hvem skal tegne polynomiet ?

For at kunne tegne et polynomie kræves ikke kun polynomiet selv, men også intervalstart, −slut o. a.



Overføre Graphics−objektet fra applettens paint() til PolynomiumsTegner−objektets tegn() :

```
public void paint(Graphics g)
{
  ...
  PolynomiumsTegner tegner = new PolynomiumsTegner();

  ...
  tegner.tegn(g,pol);
}
```

Nu er tegningen af grafen *uddelegeret* til et andet objekt

Polynomier skal ikke kun kunne tegnes i appletter, men også i andre grafiske programmer. De kan alle bruge PolynomiumsTegner.

Tænk også over:

Skal programmet kunne
- tegne flere funktioner samtidig ?
- skrive funktionsudtrykke(ne) ?
- tegne akserne, måske med et akse−objekt ?
- finde max og min ?
- integrere ?
- andet ?

Lektion 6

Praktisk

Øvelser fra sidste gang

Opsamling: Egne klasser, objekter og relationer

Pause

Nedarving

Vi går til i øvelseslokalet

Introduktion til nedarvning

Situation:

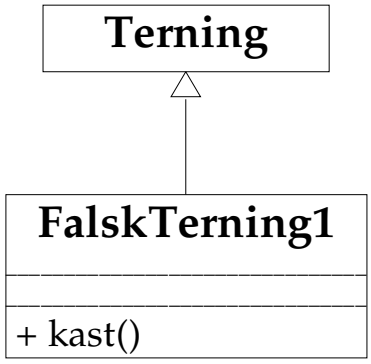
Man har en eksisterende klasse som virker, men man vil gerne tilføje ekstra funktionalitet, uden at ændre den oprindelige klasse.

To løsninger:

1)Man kan kopiere klassen, give den et nyt navn og tilføje den ønskede funktionalitet.
Men hvad hvis det er en klasse man ikke har kildekoden til ? Senere fejlrettelser i den oprindelige klasse skal laves flere steder.

2)Man kan lave en ny klasse og **arve** fra den oprindelige klasse.

Eksempel på nedarvning: FalskTerning




```
// En Terning-klasse for falske terninger.

public class FalskTerning1 extends Terning
{
  // tilsidesæt kast med en "bedre" udgave

  public void kast()
  {
    // udskriv så vi kan se at metoden bliver kaldt
    System.out.print("[kast() på FalskTerning1] ");

    værdi = (int) (6*Math.random() + 1);

    // er det 1 eller 2? Så lav det om til 6!
    if ( værdi <= 2 ) værdi = 6;
  }

}
```

FalskTerning1 arver variabler og metoder fra Terning, men metoden kast() er anderledes.

Er−en−relation: FalskTerning1 **er en** Terning.

```
public class Snydespil1
{
  public static void main(String[] args)
  {
    Terning t1 = new Terning();
    FalskTerning1 t2 = new FalskTerning1();

    System.out.println("t1: "+t1); // ku' også kalde t1.toString()
    System.out.println("t2: "+t2);

    for (int i=0; i<5; i++)
    {
      t1.kast();
      t2.kast();
      System.out.println("t1=" + t1 + "  t2=" + t2);
      if (t1.værdi == t2.værdi) System.out.println("To ens!");
    }
  }
}
```

```
[kast() på FalskTerning1] t1: 1
t2: 3
[kast() på FalskTerning1] t1=1  t2=5
[kast() på FalskTerning1] t1=1  t2=3
[kast() på FalskTerning1] t1=4  t2=3
[kast() på FalskTerning1] t1=6  t2=6
To ens!
[kast() på FalskTerning1] t1=2  t2=6
```

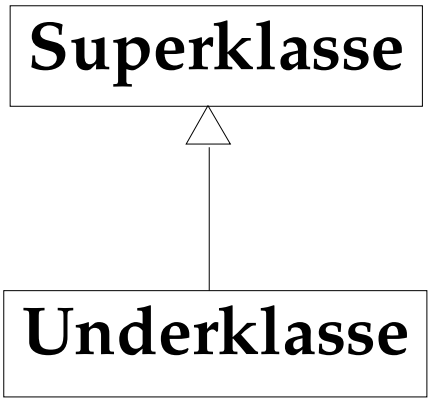
En klasse kan arve variabler og metoder fra en anden

Klassen der nedarves fra kaldes superklassen

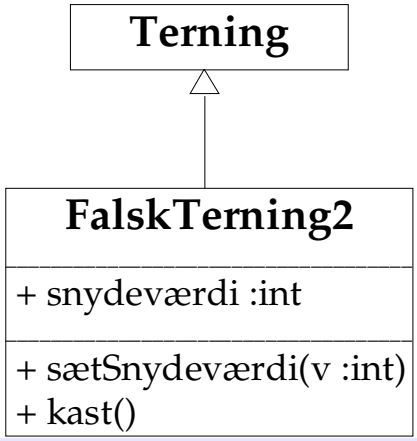
Den nye klasse (der arver fra superklassen) kaldes underklassen

Underklassen kan tilsidesætte (omdefinere) metoder arvet fra superklassen ved at definere dem igen

Det tegnes således:



At udbygge med flere metoder og variabler




```
public class FalskTerning2 extends Terning
{
  public int snydeværdi;

  public void sætSnydeværdi(int nySnydeværdi)
  {
    snydeværdi = nySnydeværdi;
  }

  public void kast()
  {
    System.out.print("[kast() på FalskTerning2] ");

    værdi = (int) (6*Math.random() + 1);

    // 1 eller 2? Så lav det om til snydeværdi!
    if ( værdi <= 2 ) værdi = snydeværdi;
  }
}
```

```
public class Snydespil2
{
  public static void main(String[] args)
  {
    FalskTerning2 t1 = new FalskTerning2();
    t1.sætSnydeværdi(4);

    for (int i=0; i<5; i++)
    {
      t1.kast();
      System.out.println("t1=" + t1);
    }
  }
}
```

```
[kast() på FalskTerning2] [kast() på FalskTerning2] t1=4
[kast() på FalskTerning2] t1=4
[kast() på FalskTerning2] t1=6
[kast() på FalskTerning2] t1=6
[kast() på FalskTerning2] t1=4
```

Introduktion til polymorfi

```
public class Snydespil2polymorfi
{
  public void main(String[] args)
  {
    FalskTerning2 falsktern = new FalskTerning2();
    falsktern.sætSnydeværdi(4);

    Terning
    tern = falsktern;  // tilladt

    for (int i=0; i<5; i++)
    {
      tern.kast();
      System.out.println("t1=" + t1);
    }

    // ikke tilladt: falsktern.sætSnydeværdi(4);
  }
}
```

```
[kast() på FalskTerning2] [kast() på FalskTerning2] t1=øjne:4
[kast() på FalskTerning2] t1=øjne:6
[kast() på FalskTerning2] t1=øjne:3
[kast() på FalskTerning2] t1=øjne:6
[kast() på FalskTerning2] t1=øjne:4
```

Tildele en **variabel af superklassen** et **objekt af underklassen**.

FalskTerning **er−en** Terning, og det er tilladt at behandle FalskTerning-objektet "som om" det er et Terning−objekt.

Men det er stadig FalskTernings kast()−metode der kaldes.

Generelt:

De metoder der **kan kaldes** er dem fra variabeltypen (Terning).

De metoder der **bliver kaldt** er dem fra objektet (FalskTerning).

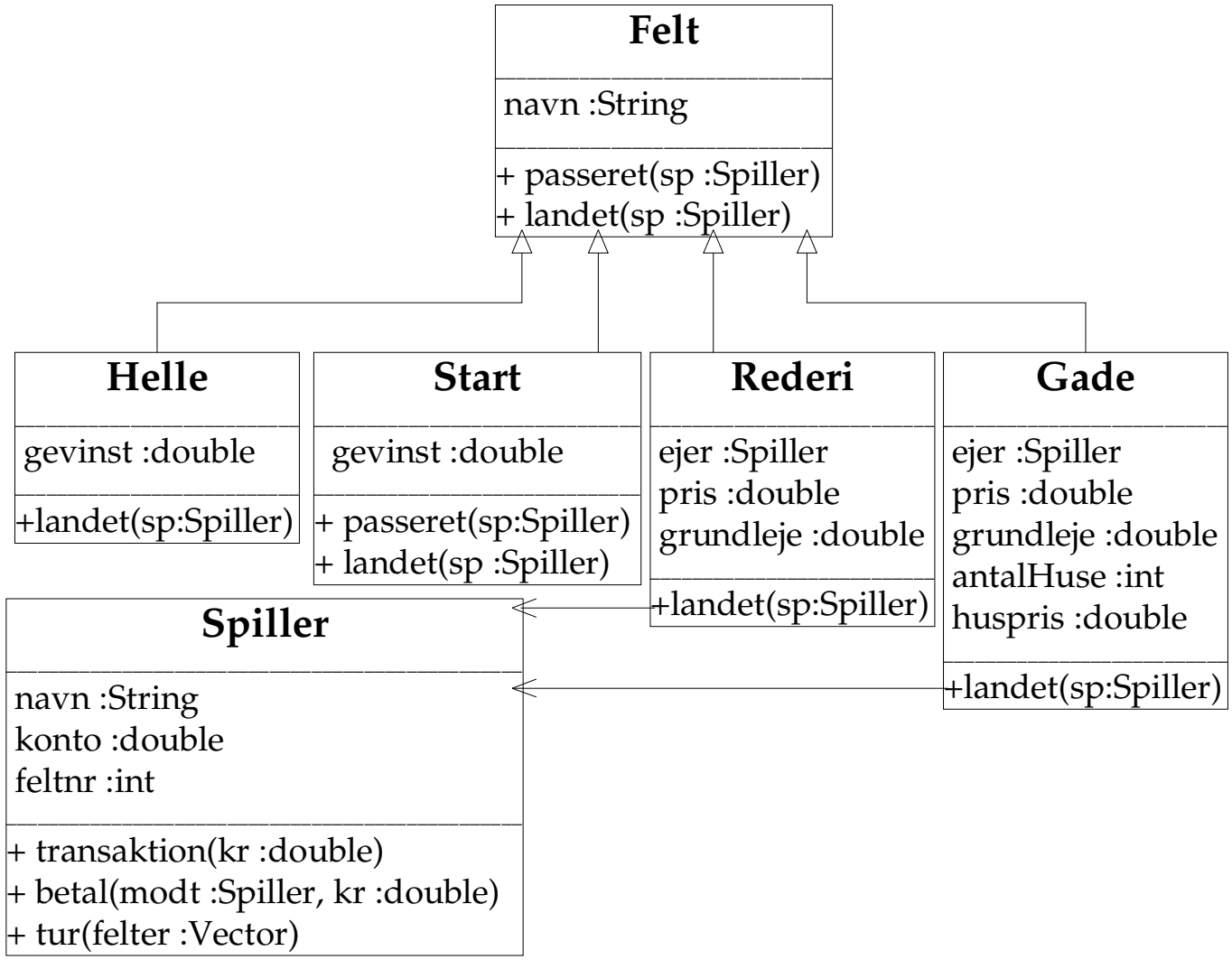
Polymorfi betyder "mange former".

En Terning−variabel kan referere til mange slags objekter: Et Terning−objekt, et FalskTerning−objekt eller et andet objekt, hvis klasse arver fra Terning.

Nedarvnings−hierakier

Med arv kan man skabe et hierarki af klasser der ligner hinanden og samtidig kan opføre sig forskelligt.

Her er vist en skitse til klassediagrammet fra et matadorspil.



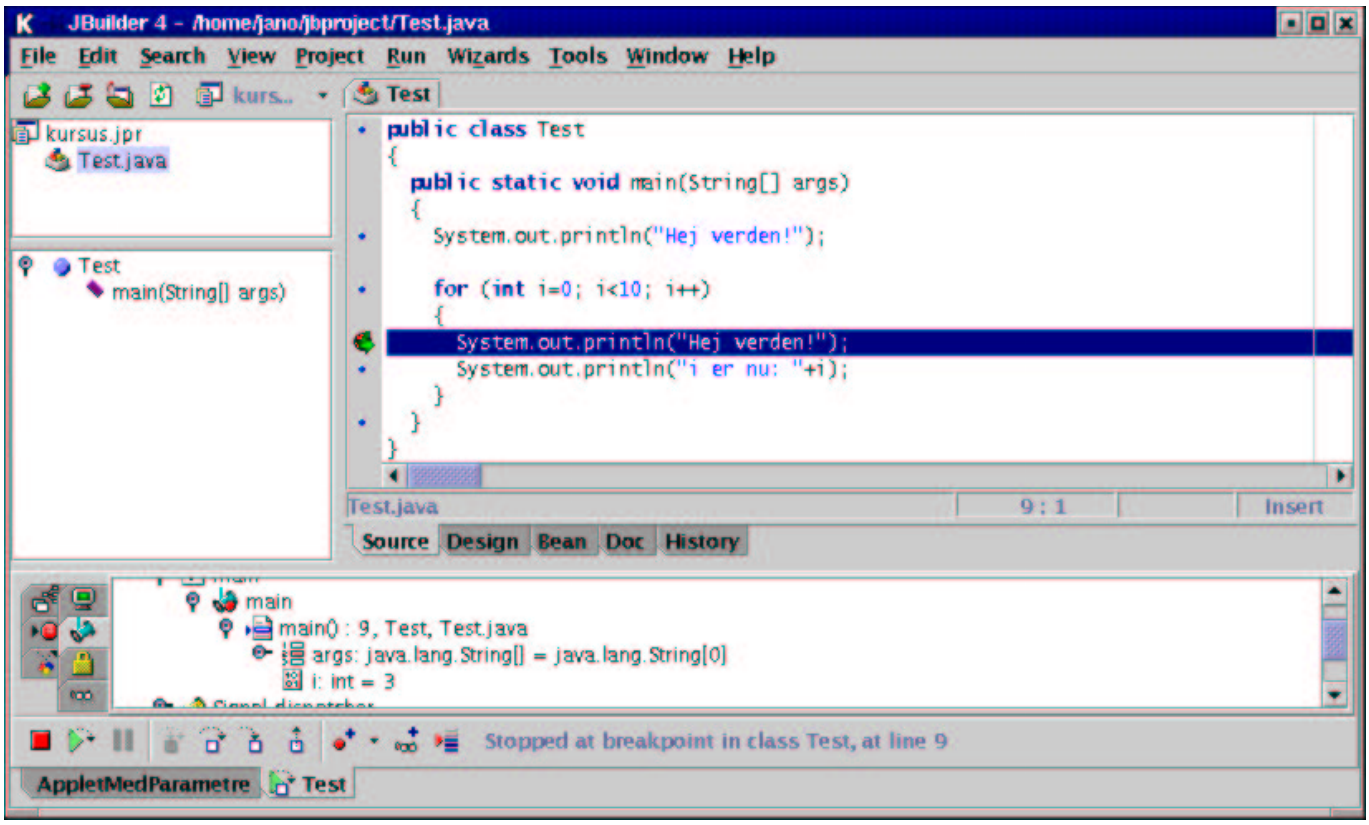
Ide til forprojekt:

Lav klassen Funktion med en metode til at beregne en funktionsværdi.
Lav nedarvinger Sinus, Kvadratrod, Polynomium, ...

Eet sted i programmet oprettes et funktions−objekt:
Funktion f = new Kvadratrod();

Resten af programmet bruger objektet som om det var en Funktion.

Trinvis gennemgang med JBuilder



Klik ude i venstre margen (den grå kant) på en af de første linier i dit program, der vil nu komme en rød prik. Dette markerer et stoppunkt (eng: breakpoint).

Når du trykker skift−F9, udfører JBuilder dit program, men i "debug" mode. Dvs. den stopper ved dit stoppunkt. Den grønne pil viser, hvor langt den er kommet. Du kan nu bruge F8 til at hoppe til næste linie.

Mulighederne findes i "run"−menuen:

- step over (F8): hopper til næste linie.
- step into (F7): Hvis du står ved et metodekald hopper du ind og kan se metoden blive udført (gør det kun ved de metoder du selv har defineret).
- step out: Hvis du er inde i et metodekald udføres programmet indtil det vender tilbage til kalderen.
- resume program (F9): fortsætter programudførslen og stopper ved næste stoppunkt den kommer til.
- reset program: stopper den trinvise gennemgang.

Det kan være lidt svært at få det til at virke. Læg mærke til at man kan komme til at køre sit program flere gange på samme tid. Der vises nederst flere faneblade med samme programnavn (session på figuren).

Bemærk iøvrigt at i denne version af JBuilder (den gratis version) viser den også system−klassernes udførsel. Det er du næppe interesseret i. Kommer du således ind i noget kode, som du ikke selv har skrevet, så brug "step out" for at komme tilbage til dine egne klasser. Det er derfor det er smart at sætte et stoppunkt ved første linie i programmet. Hvis man bruger F8 først, så hopper JBuilder ind i nogle skumle system−klasser.

Du skal også være opmærksom på at det ikke er alle steder i programmet, hvor man kan sætte stoppunkter: Hvis den røde prik får et kryds henover (), så betyder det at stoppunktet ikke virker. En tommelfingerregel er at man skal sætte stoppunkter på de linier, der "gør" noget. dvs. tildelinger og metodekald. Stoppunkter på erklæringer, blok−paranteser og metodehoveder virker ikke.

Du kan også undersøge variables værdi under programudførslen. I dialogen "run/add watch" kan du skrive navnet på en variabel. I nederste del af JBuilder (der hvor uddata for programmet normalt vises) er der en række faneblade. I fanebladet med et par briller () vises de variable du har tilføjet med "add watch". Du kan også bruge fanebladet med en lille rulle tråd som ikon (), men det er lidt sværere at overskue.

Lektion 7

Resume: arv, super

Alting arver fra Object

Forprojekterne

Resume: polymorfi

Konstruktører og arv

Nedarvning i graf−tegningsprogrammet

Øvelser / vejledning i obligatorisk opgave

Praktisk

Afkrydsning

Midtvejsevaluering

Hvem har lavet et forprojekt og har en udskrift med ?

Fordel for alle, hvis de der kender stoffet hjælper dem, der har svært ved det

Avanceret−afsnit

Spørgsmål til det læste

I dag: Udveksling af forprojekterne

Dan grupper på to eller tre.

Du skal mindst se ét program med Polynomium og Polynomieled

Sæt dig sammen med en du ikke før har været sammen med

Læreren og hver gruppemedlem får en kopi af hver rapport. Den læses til næste gang.

Forklar kort hvordan jeres programmer virker:

− hvad hedder klasserne

− hvilke metoder og variable har de

− hvor mange objekter oprettes der

− hvad kalder hvad

Næste gang: Gennemsyn af forprojekter

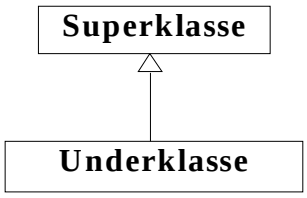
Læs rapporterne og forstå hvordan programmet virker

Foreslå forbedringer af de andres programmer på en positiv facon.
Prøv at se om du kan finde logiske fejl

Tag pænt imod forslag og kritik

Resume − arv

En klasse kan arve variabler og metoder fra en anden klasse (med "extends")



Klassen der nedarves fra kaldes superklassen

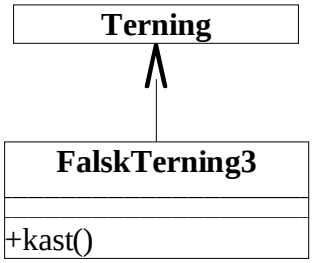
Klassen der arver fra superklassen kaldes underklassen

Underklassen kan tilsidesætte (omdefinere) metoder arvet fra superklassen ved at definere dem igen

For at tilsidesætte en metode, skal man i underklassen lave en eksakt kopi af metode−hovedet fra superklassen

super

Med super får man adgang til metoder som de er kendt i superklassen.



Det kan være nyttigt til at genbruge programkode

```
public class FalskTerning3 extends Terning
{
  public void kast()
  {
    super.kast(); // kald den oprindelige kast()−metode

    // blev det 1 eller 2? Så lav det om til 6!
    if ( værdi <= 2 ) værdi = 6;
  }
}
```

Polymorfi

En variabel kan godt referere til objekter af en underklasse af variablens type

f.eks:

```
Terning t;
t = new FalskTerning3();
```

Variablens type bestemmer, hvilke metoder man kan kalde på objektet, og hvilke objektvariable man kan læse og ændre

(f.eks er t af type Terning, ligemeget hvad den refererer til)

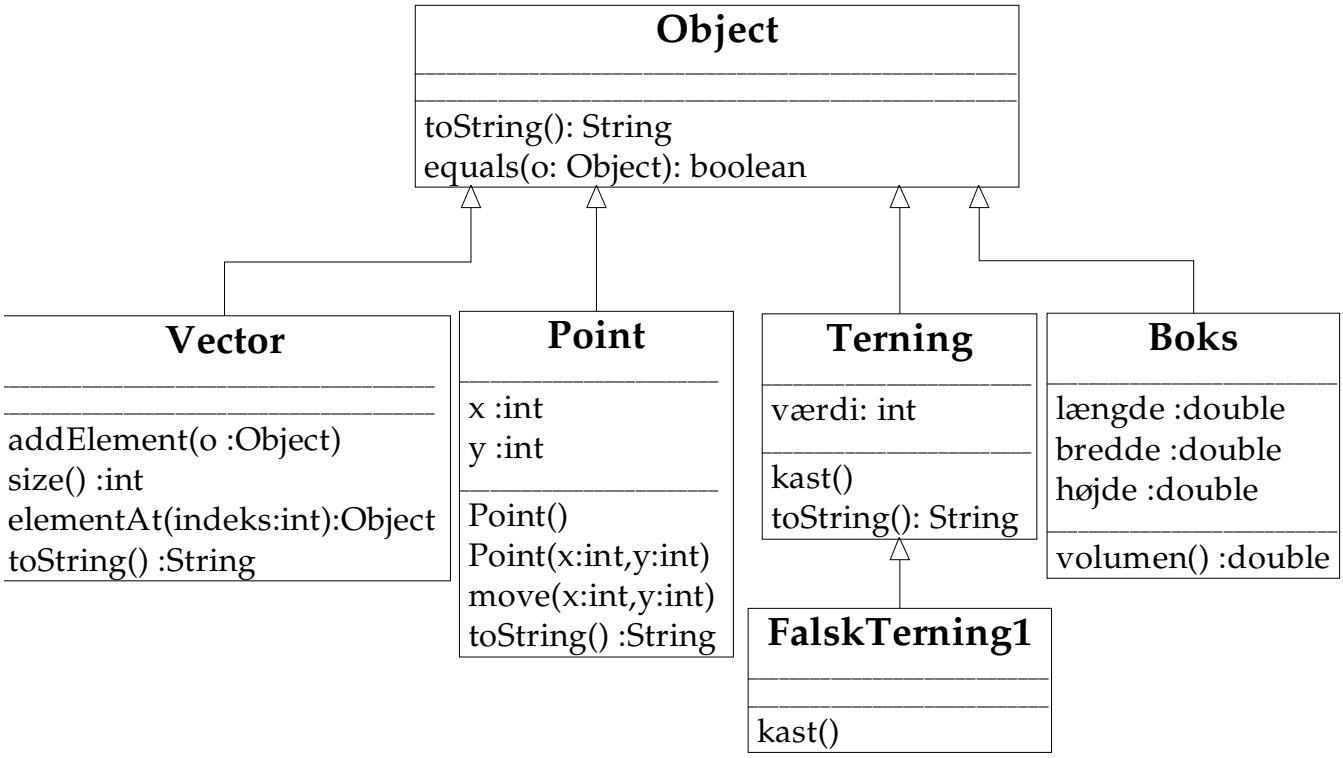
Objektets type bestemmer hvilken metode−definition (krop) der bliver udført

f.eks:

```
t.kast();    // her bliver FalskTerning3's kast() kaldt
```

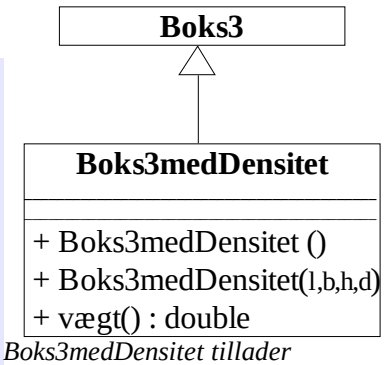
dette kan bruges til at skabe hierakier af objekter der ligner hinanden, men opfører sig forskelligt

Alting arver fra Object



derfor kendes bl.a. toString()−metoden af alle objekter

Konstruktører skal defineres på ny i en nedarving



Boks3medDensitet tillader oprettelse på to måder

```
public class Boks3medDensitet extends Boks3
{
  private double densitet;

  public Boks3medDensitet()
  {
    // overflødig, den kaldes implicit:
    // super();
    massefylde = 10.0;
  }


  public Boks3medDensitet(double l, double b,
        double h, double densitet)
  {
    // kald superklassens konstruktør med parametre
    super(l,b,h);
    this.densitet = densitet;
  }

  public double vægt()
  {
    // superklassen udregner volumen for os
    return volumen() * densitet;
  }
}
```

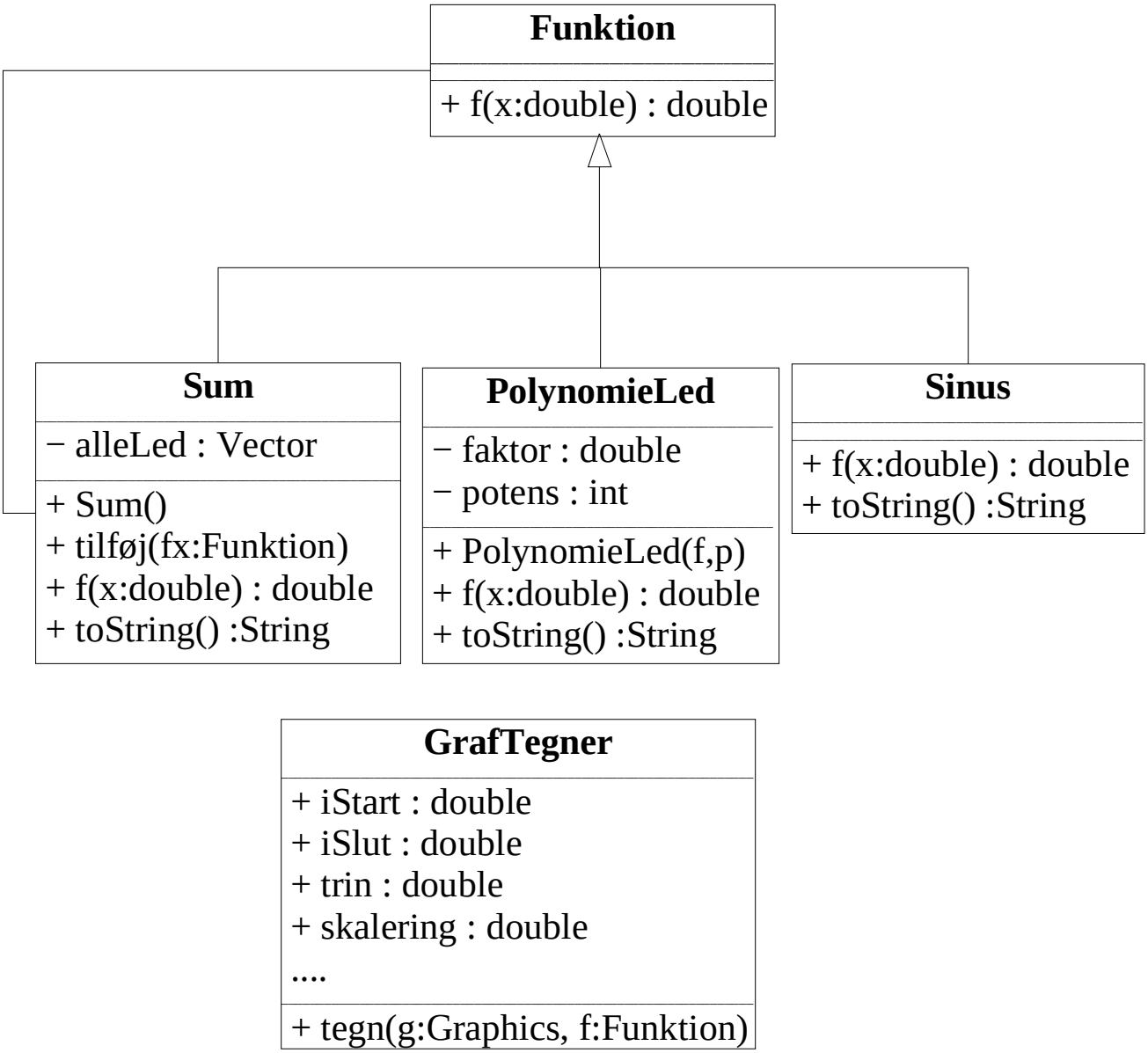
Konstruktører skal defineres på ny i en underklasse

En konstruktør i en underklasse kalder først en af superklassens konstruktører

Superklassens konstruktør kan kaldes med: super(parametre)

Hvis programmøren ikke kalder en af superklassens konstruktører, indsætter Java automatisk et kald af superklassens konstruktør uden parametre

Nedarvning i graf−tegningsprogrammet



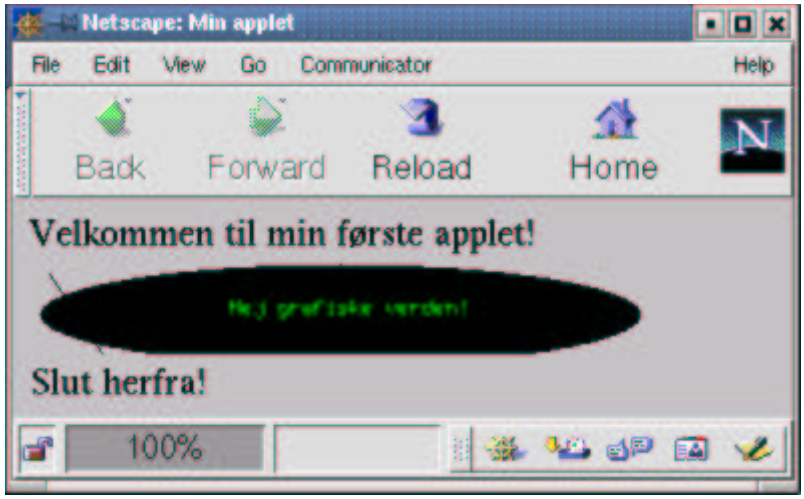

```
public class GrafTegner {
...
  void tegn(Graphics g, Funktion f)
  {
     double gx=iStart; // gammel x−værdi
     for (double x=iStart+trin;x<iSlut;x=x+trin)
     {
       //uden skalering:
       g.drawLine(  (int)gx,(int)f.f(gx),
                    (int)x, (int)f.f(x));
       gx=x;
     }
  }
}
```

Lektion 8

Test dig selv

Gå igennem afsnit 5.7 to og to og tjek at I kan svare på spørgsmålene

Appletter



HTML−dokumentet

```
<HTML>
<HEAD>
        <TITLE>Min applet</TITLE>
</HEAD>

<BODY>
        Velkommen til min første applet!<BR>

        <APPLET
          CODEBASE = "."
          CODE     = "MinApplet.class"
          WIDTH    = 400
          HEIGHT   = 300>
        </APPLET>

        Slut herfra!
</BODY>
</HTML>
```

Java−koden

```
import java.awt.*;
import java.applet.*;

public class MinApplet extends Applet
{
  public void paint(Graphics g)
  {
    // Herunder referer g til et Graphics−objekt man kan tegne med.
    g.drawLine(10,10,50,70);

    g.fillOval(5,5,300,50);

    g.setColor(Color.green);

    g.drawString("Hej grafiske verden!",100,30);
  }
}
```

Metoder som du kan kalde

Nogle af Applet−klassens metoder

repaint(int millisekunder)

int **getWidth**() (fra JDK 1.2)

int **getHeight**() (fra JDK 1.2)

Dimension **getSize**()

AudioClip **getAudioClip**(URL url, String filnavn)
returnerer et lydklip−objekt, typisk fra en .wav−fil.

Image **getImage**(URL url, String filnavn)
returnerer et billede−objekt, typisk fra en .jpg eller .gif−fil.

Metoder som fremviseren kalder

```
public class Fremvisertest extends Applet
{
  public Fremvisertest()
  { System.out.println("Konstruktøren blev kaldt"); }

  public void init()
  { System.out.println("Appletten er initaliseret"); }

  public void start()
  { System.out.println("Appletten er synlig"); }

  public void paint(Graphics g)
  { System.out.println("Appletten bliver tegnet på skærmen"); }

  public void stop()
  { System.out.println("Appletten er ikke mere synlig"); }

  public void destroy()
  { System.out.println("Appletten smides væk"); }
}
```

Generering af grafiske brugergrænseflader

```
import java.applet.*;
import java.awt.*;

public class MinApplet extends Applet
{
 TextField textFieldNavn = new TextField();

 public void paint(Graphics g)
 {
  ...
 }

 public MinApplet()
 {
  textFieldNavn.setText("Jacob");
  textFieldNavn.setBounds(new Rectangle(141, 61, 112, 29));
  this.setLayout(null);
  this.add(textFieldNavn, null);
 }
}
```

Udviklingsværktøjer

Bemærk at værktøjer som JBuilder vil lægge initialiseringen i en separat metode, f.eks.

```
 private void jbInit() throws Exception {
  textFieldNavn.setText("Jacob");
  textFieldNavn.setBounds(new Rectangle(141, 61, 112, 29));
  this.setLayout(null);
  this.add(textFieldNavn, null);
 }
```

som så kaldes fra konstruktøren:

```
 public MinApplet()
 {
  try {
   jbInit();
  }
  catch(Exception e) {
   e.printStackTrace();
  }
 }
```

Applikationer

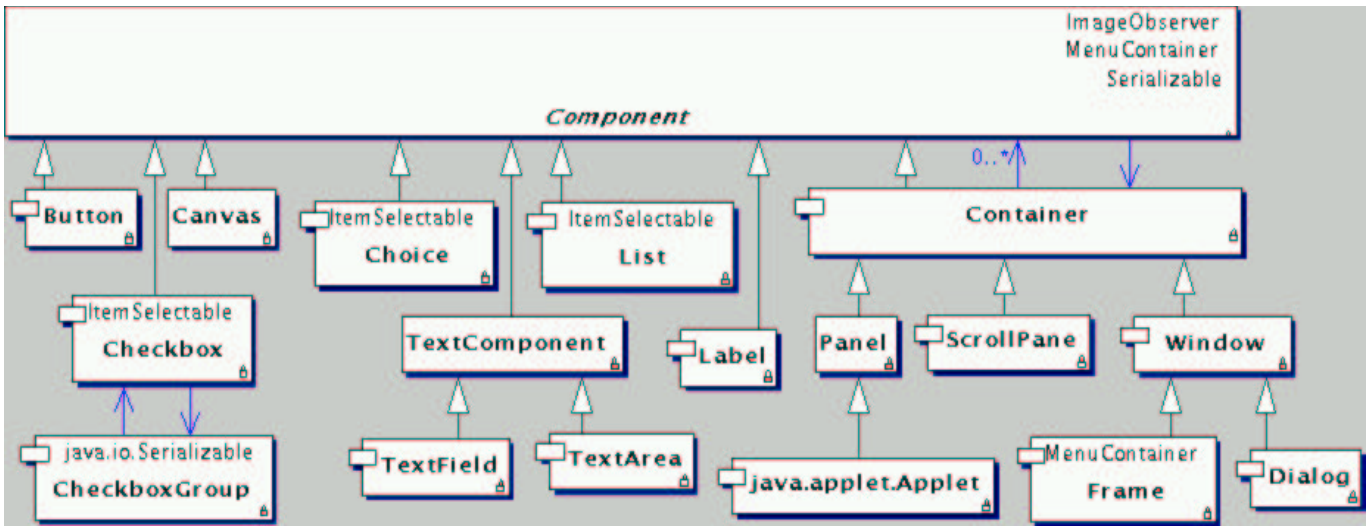
Programkoden i det grafiske vindue minder meget om koden i en applet, blot skal der arves fra Frame i stedet for Applet:

```
import java.awt.*;

public class GrafiskVindue extends Frame
{
  // paint()−metode og intialisering af grafiske komponenter:

  TextField textFieldNavn = new TextField();

  public void paint(Graphics g)
  {
    ...
  }

  public GrafiskVindue()
  {
    textFieldNavn.setText("Jacob");
    textFieldNavn.setBounds(new Rectangle(141, 61, 112, 29));
    this.setLayout(null);
    this.add(textFieldNavn, null);
  }
}
```

Beder man JBuilder oprette en ny applikation vil den generere to filer:
Frame1.java (der repræsenterer det grafiske vindue).
Application1.java (den der indeholder main()−metoden) og

```
public class GrafiskProgram
{
  public static void main(String args[])
  {
    GrafiskVindue gv = new GrafiskVindue();
    gv.pack();
    gv.show();
  }
}
```

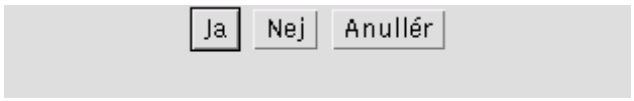
Grafiske komponenter



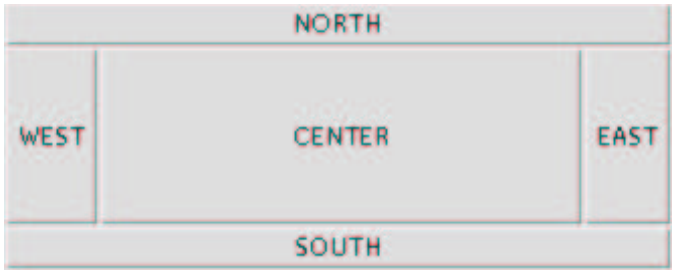
Komponenter: Label, TextField, Button, ...

Containere: Frame, Dialog (e.v.t. modal), Menubar, Menu, Toolbar

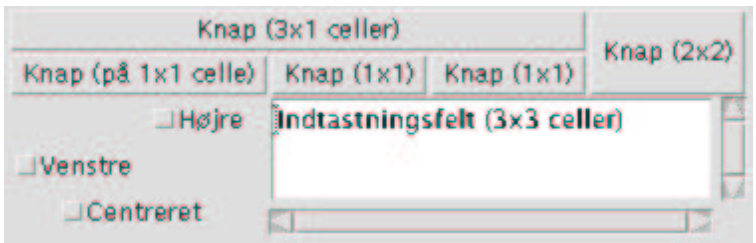
FlowLayout



BorderLayout



GridBagLayout



Java Examples in a Nutshell, kapitel 10

GUI = Graphical User Interface = Grafisk brugergrænseflade

Hændelsesbaseret programmering

AWT − de gamle klasser (i pakken java.awt)

Swing − de nye klasser (i pakken javax.swing)

Interface: Tænk på det som "klasse"

Alle et interface’s metoder skal tilsidesættes
Det hedder "implements" i stedet for "extends"

Hændelser

Forskellige typer:

ActionEvent, ItemEvent, KeyEvent, MouseEvent, MouseMotionEvent

Listener − hændelseslytter

ActionListener, ItemListener, KeyListener, MouseListener, MouseMotionListener

Adapter − Klasse ligesom den tilsvarende hændelseslytter

KeyAdapter, MouseMotionAdapter, MouseAdapter

Brug en adapter, istedet for en listener, hvis I kan

Lektion 9

Praktisk
Gennemgang af Matador?
Undtagelser og stakspor
JDBC − databaseadgang
Vejledning i obligatorisk opgave

Praktisk

Spørgsmål til det læste
Hvor mange har læst i bogen ?
Hvor mange har læst i Java Examples ?

Obligatorisk opgave:

- Er der nogen der ikke er i en gruppe?
- Er der nogen der ikke ved hvad de vil lave?

Næste gang: Obligatorisk opgave

Skitsér de skærmbilleder der endnu mangler

Find relevante navneord, udsagnsord og tillægsord for programmet

Løsning til Matador−opgave

Felt.java

```
public class Felt
{
  ...

  Point pos = new Point();
  public void tegn(Graphics g)
  {
    g.setColor(Color.black);
    g.drawString(navn,pos.x,pos.y);
  }

  public void tegnBil(Graphics g, Spiller s)
  {
    g.setColor(s.farve);
    g.drawString(s.navn,pos.x,pos.y−10);
    g.drawRoundRect(pos.x,pos.y−11,40,12,5,5);
  }
}
```

Grund2.java

```
public class Grund2 extends Felt
{
  ...

  public void tegn(Graphics g)
  {
    super.tegn(g);
    if (ejer != null) {
      g.setColor(Color.blue);
      g.drawString(ejer.navn,pos.x,pos.y+10);
    }
  }
}
```

Gade2.java

```
public class Gade2 extends Grund2
{
  ...

  public void tegn(Graphics g)
  {
    super.tegn(g);
    if (antalHuse > 0)
      g.drawString(antalHuse + " huse",pos.x,pos.y+20);
  }
}
```

MatadorApplet.java

```
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import java.applet.*;
import javax.swing.*;

public class MatadorApplet extends Applet {

 Spiller sp1=new Spiller("Søren",50000,Color.green);
 Spiller sp2=new Spiller("Gitte",50000,Color.yellow);
 int tur = 0;

 Vector felter=new Vector();
 JButton jButtonSpilEnTur = new JButton();


 public MatadorApplet() {
   felter.addElement(new Start(5000));
   felter.addElement(new Gade2("Gade 1",10000, 400,1000));
   felter.addElement(new Gade2("Gade 2",10000, 400,1000));
   felter.addElement(new Gade2("Gade 3",12000, 500,1200));
   felter.addElement(new Chancen(10000));
   felter.addElement(new Gade2("Gade 5",15000, 700,1500));
   felter.addElement(new Rederi2("Maersk",17000,4200));
   felter.addElement(new Helle(15000));
   felter.addElement(new Gade2("Gade 7",20000,1100,2000));
   felter.addElement(new Gade2("Gade 8",20000,1100,2000));
   felter.addElement(new Gade2("Gade 9",30000,1500,2200));

   for (int i=0; i<felter.size(); i++)
   {
     Felt f = (Felt) felter.elementAt(i);
     double v = Math.PI*2*i/felter.size();
     f.pos.move(100 + (int) (100*Math.cos(v)),
                110 + (int) (100*Math.sin(v)));
   }

   try {
     jbInit();
   }
   catch(Exception e) {
     e.printStackTrace();
   }
 }

 private void jbInit() throws Exception {
    jButtonSpilEnTur.setText("Spil en tur");
    jButtonSpilEnTur.setBounds(new Rectangle(280, 272, 117, 25));
    jButtonSpilEnTur.addActionListener(
     new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButtonSpilEnTur_actionPerformed(e);
      }
     }
    );
    this.setLayout(null);
    this.add(jButtonSpilEnTur, null);
 }
```

```
 public void paint(Graphics g)
 {
  for (int i=0; i<felter.size(); i++)
  {
   Felt f = (Felt) felter.elementAt(i);
   f.tegn(g);
  }


  Felt f = (Felt) felter.elementAt(sp1.feltnr);
  f.tegnBil(g,sp1);
  g.drawString(sp1.navn,300,10);
  g.drawString(sp1.konto+" kr",300,20);

  f = (Felt) felter.elementAt(sp2.feltnr);
  f.tegnBil(g,sp2);
  g.drawString(sp2.navn,300,110);
  g.drawString(sp2.konto+" kr",300,120);
  spilEnTur();
 }

 void spilEnTur() {
  tur = tur + 1;
  if (tur % 2 == 0) sp1.tur(felter);
  else sp2.tur(felter);
  repaint(500);
 }

 void jButtonSpilEnTur_actionPerformed(ActionEvent e) {
  spilEnTur();
 }

}
```

Undtagelser og stakspor

```
import java.util.*;

public class SimpelUndtagelse
{
  public static void main(String[] args)
  {
    System.out.println("Punkt A");       // punkt A
    Vector v = new Vector();
    System.out.println("Punkt B");       // punkt B
    v.elementAt(5);
    System.out.println("Punkt C");       // punkt C
  }
}
```

```
Punkt A
Punkt B
java.lang.ArrayIndexOutOfBoundsException: 5 >= 0
  at java.util.Vector.elementAt(Vector.java:417)
  at SimpelUndtagelse.main(SimpelUndtagelse.java:10)
Exception in thread "main"
```

En metode kan "protestere" med en undtagelse, hvis det er uforsvarligt at fortsætte.

Der findes forskellige typer undtagelser

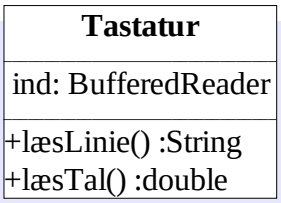
- ArrayIndexOutOfBoundsException
- NumberFormatException
- NullPointerException
- FileNotFoundException *
- IOException *

* = tvungen håndtering

De er allesammen klasser, og de arver fra klassen Exception

Stak−sporet viser hvor fejlen opstod

Undtagelser kan fanges og behandles




```
import java.io.*;

public class Tastatur1
{
  BufferedReader ind;

  public Tastatur1()
  {
    ind = new BufferedReader(
      new InputStreamReader(System.in));
  }

  public String læsLinie()
  {
    try {
      String linie = ind.readLine();
      return linie;
    } catch (IOException u)
    {
      u.printStackTrace();
    }
    return null;
  }

  public double læsTal()
  {
    String linie = læsLinie();
    return Double.parseDouble(linie);
  }
}
```

Syntaks:

```
  try
  {
    ...                    // programkode hvor der er en risiko
    ...                    // for at en undtagelse opstår
  }
  catch (Undtagelsestype u)// Undtagelsestype er f.eks. Exception
  {
    ...                    // kode som håndterer fejl af
    ...                    // typen Undtagelsestype
  }
  ...           // dette udføres både hvis ingen undtagelse opstod
  ...           // og hvis der opstod fejl af typen Undtagelsestype
```

```
public class SumMedTastatur {
  public static void main(String arg[]) {

    Tastatur1 t = new Tastatur1();
    boolean stop = false;

    try
    {
      while (!stop)
      {
        System.out.print("Hvormange tal lægge sammen? ");
        double antalTal = t.læsTal();
        double sum = 0;

        for (int i=0; i<antalTal; i=i+1)
        {
          System.out.print("Indtast tal: ");
          sum = sum + t.læsTal();
        }

        System.out.println("Summen er: "+sum);
        System.out.print("Vil du prøve igen? (j/n)");

        if ("n".equals(t.læsLinie()) stop = true;
      }
    }  catch (Exception u) {
       System.out.println("Der opstod en undtagelse!");
      u.printStackTrace();
    }
  }
}
```

```
Hvor mange tal vil du lægge sammen? 2
Indtast tal: 1
Indtast tal: 2
Summen er: 3.0
Vil du prøve igen? j
Hvor mange...
```

Brugeren taster og taster ... men hvad sker der hvis han taster forkert?

```
Hvor mange tal vil du lægge sammen? 3
Indtast tal: 1
Indtast tal: 17xxøføf
Der opstod en undtagelse!
java.lang.NumberFormatException: 17xxøføf
   at ..readJavaFormatString(FloatingDecimal.java:1182)
   at java.lang.Double.parseDouble(Double.java:190)
   at Tastatur1.læsTal(Tastatur1.java:27)
   at SumMedTastatur.main(SumMedTastatur.java:18)
```

.. og programmet stopper.

Præcis håndtering af undtagelser med try−catch inde i while−løkken:

```
public class SumMedTastatur2 {
  public static void main(String arg[]) {

    Tastatur1 t = new Tastatur1();
    boolean stop = false;

    while (!stop)
    {
      System.out.print("Hvor mange tal lægge sammen? ");
      try
      {
        double antalTal = t.læsTal();
        double sum = 0;

        for (int i=0; i<antalTal; i=i+1)
        {
          System.out.print("Indtast tal: ");
          sum = sum + t.læsTal();
        }
        System.out.println("Summen er: "+sum);
      }  catch (Exception u) {
        System.out.println("Indtastningsfejl − " + u);
      }
      System.out.print("Vil du prøve igen? (j/n)");
      if ("n".equals(t.læsLinie()) stop = true;
    }
  }
}
```

```
Hvor mange tal vil du lægge sammen? 5
Indtast tal: 1
Indtast tal: x2z
Indtastningsfejl − java.lang.NumberFormatException: x2z
Vil du prøve igen? j
Hvor mange tal vil du lægge sammen? 3
Indtast tal: 1200
Indtast tal: 1
Indtast tal: 1.9
Summen er: 1202.9
Vil du prøve igen? n
```

JDBC – databaseadgang

Indlæse driver – Oracle–database:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Oprette forbindelse – Oracle–database:

```
Connection forb = DriverManager.getConnection(
  "jdbc:oracle:thin:@oracle.cv.ihk.dk:1521:student",
  "stuk1001", "stuk1001");
```

Med Java under Windows følger en standard JDBC–ODBC–bro:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection forb = DriverManager.getConnection (
  "jdbc:odbc:datakilde1","","");
```

Derefter oprettes en ’prompt’

```
Statement stmt = forb.createStatement();
```

... og man er klar til at sende SQL til databasen

Kommandoer

```
stmt.executeUpdate(
  "create table KUNDER (NAVN varchar(32), KREDIT float)" );
stmt.executeUpdate("insert into KUNDER values(’Jacob’, –1799)");
stmt.executeUpdate("insert into KUNDER values(’Brian’, 0)");
```

Forespørgsler

```
ResultSet rs = stmt.executeQuery(
   "select NAVN, KREDIT from KUNDER");
```

ResultSet–objektet repræsenterer svaret på forespørglsen:

```
while (rs.next())
{
  String navn = rs.getString("NAVN");
  float kredit = rs.getFloat("KREDIT");
  System.out.println(navn+" "+kredit);
}
```

Indkapsling og abstaktion − pak det ind i en klasse for sig

Definér Databaseforbindelse−objekt og pak data ind i en specialiceret klasse, f.eks:

```
public class Kunde
{
  String navn;
  double kredit;

  public Kunde(String n, double k)
  {
    navn = n;
    kredit = k;
  }

  public String toString()
  {
    return navn+": "+kredit+" kr.";
  }
}
```

Så kan programmet abstrahere fra hvordan data er lagret

```
import java.util.*;

public class BenytDatabaseforbindelse
{
  public static void main(String arg[])
  {
    try {
      Databaseforbindelse dbf = new Databaseforbindelse();

      // dbf.opretTestdata(); // sæt ind hvis tabellen ikke findes
      Vector  v = dbf.hentAlle();
      System.out.println("Alle data: "+v);
      dbf.sletAlleData();

      dbf.indsæt( new Kunde("Kurt",1000) );

    } catch(Exception e) {
      System.out.println("Problem med database: "+e);
      e.printStackTrace();
    }
  }
}
```

```
Alle data: [Kurt: 1000.0 kr.]
```

```
import java.sql.*;
import java.util.*;

public class Databaseforbindelse
{
 private Connection forb;
 private Statement stmt;

 public Databaseforbindelse() throws Exception
 {
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection forb = DriverManager.getConnection(
   "jdbc:oracle:thin:@oracle.cv.ihk.dk:1521:student",
   "stuk1001","hemli'");
  stmt = forb.createStatement();
 }

 public void sletAlleData() throws SQLException
 {
  stmt.execute("truncate table KUNDER");
 }

 public void opretTestdata() throws SQLException
 {
  try { // hvis tabellen allerede eksisterer opstår der en
        // SQL-udtagelse
   stmt.executeUpdate(
    "create table KUNDER (NAVN varchar(32), KREDIT float)" );
  } catch (SQLException e) {
   System.out.println("Kunne ikke oprette tabel: "+e);
  }
  stmt.executeUpdate(
   "insert into KUNDER values('Jacob', -1799)");
  stmt.executeUpdate("insert into KUNDER values('Brian', 0)");
 }

 public void indsæt(Kunde k) throws SQLException
 {
  stmt.executeUpdate(
   "insert into KUNDER values(
    '" + k.navn + "', " + k.kredit + ")");
 }
```

```
  public Vector hentAlle() throws SQLException
  {
    Vector alle = new Vector();
    ResultSet rs = stmt.executeQuery(
      "select NAVN, KREDIT from KUNDER");
    while (rs.next())
    {
      Kunde k = new Kunde( rs.getString("NAVN"),
        rs.getDouble("KREDIT"));
      alle.addElement(k);
    }
    return alle;
  }
}
```

Lektion 10

Virkefelt
Intro til objektorienteret analyse og design (OOAD)
Demonstration: Grafiske programmer med JBuilder
Vejledning i obligatorisk opgave

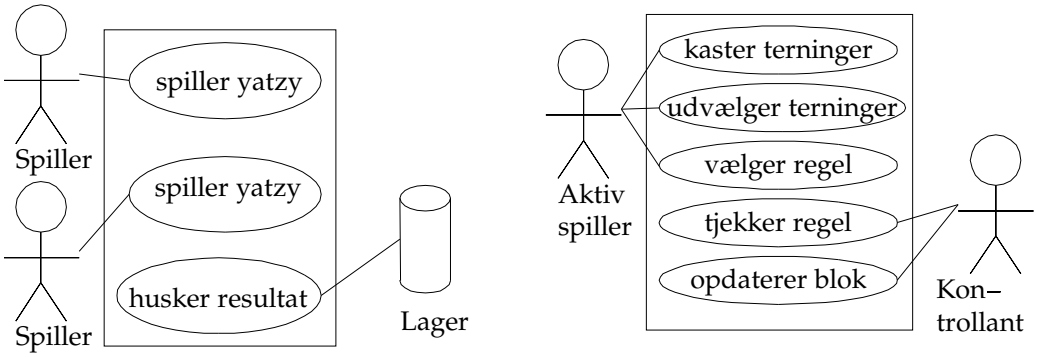
Objektorienteret analyse

Analysens formål er at afspejle virkeligheden mest muligt.

Skrive vigtige ord op

Yatzyspil
Terning − værdi, kaste, holde
Raflebæger − kombination, ryste, holde
Blok − skrive spillernavn på, skrive point på
Spiller − navn
Computerspiller
Strategi
Menneske
Regel (kunne også kaldes en mulighed eller et kriterium) − opfyldt, pointgivning
Lager

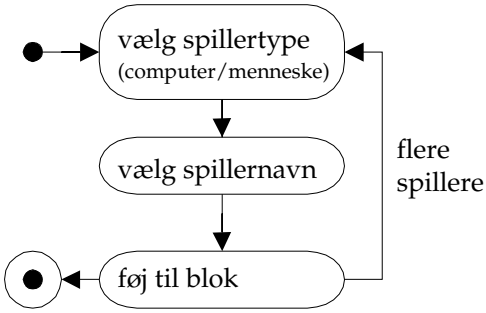
Brugsmønstre



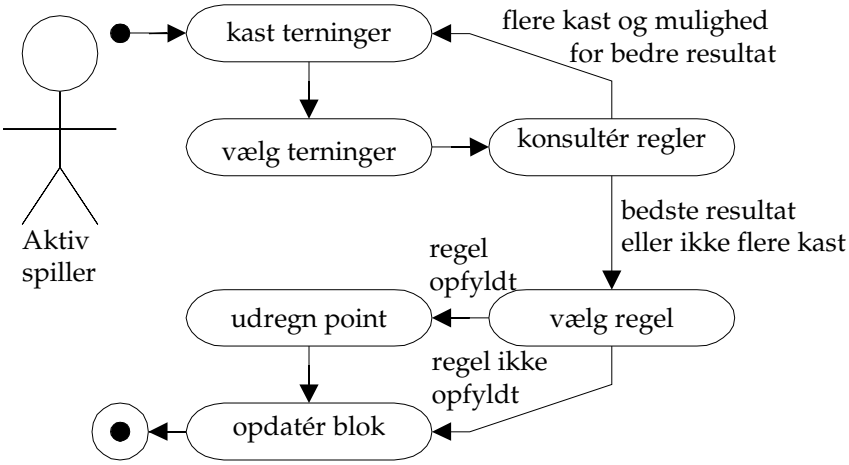
Aktivitetsdiagrammer

Aktivitetsdiagrammer beskriver den rækkefølge, som adfærdsmønstre og aktiviteter foregår i.

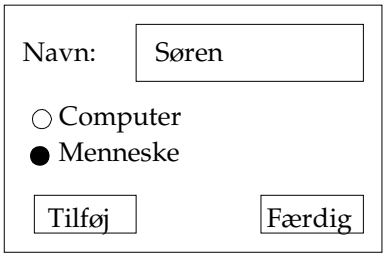
Eksempel: Aktiviteten "definere deltagere i spillet":



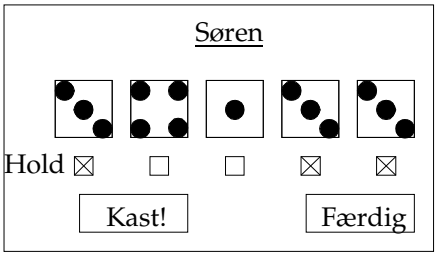
Herunder ses et diagram for spillets gang, "en tur":



Skærmbilleder



TilfoejSpillervindue



Turvindue

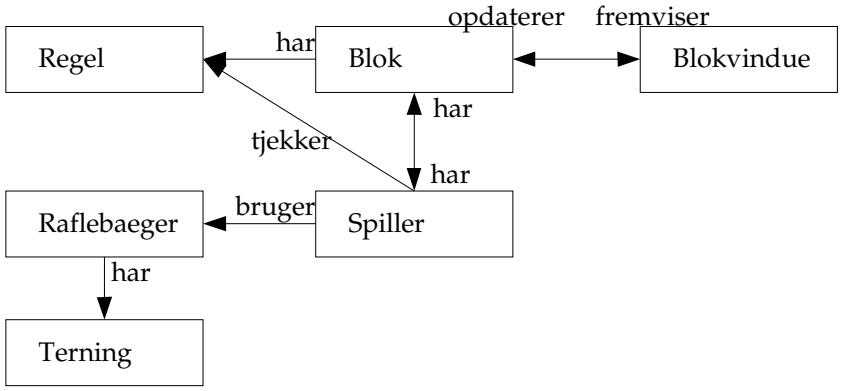
	Jacob	**Søren**
Ettere	4	
Toere		
Treere		9
etc...		
Sum		
Bonus		
Et par		
etc...		
Sum		

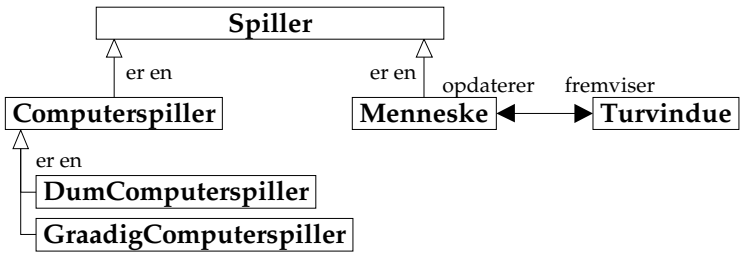
Blokvindue

Objektorienteret design

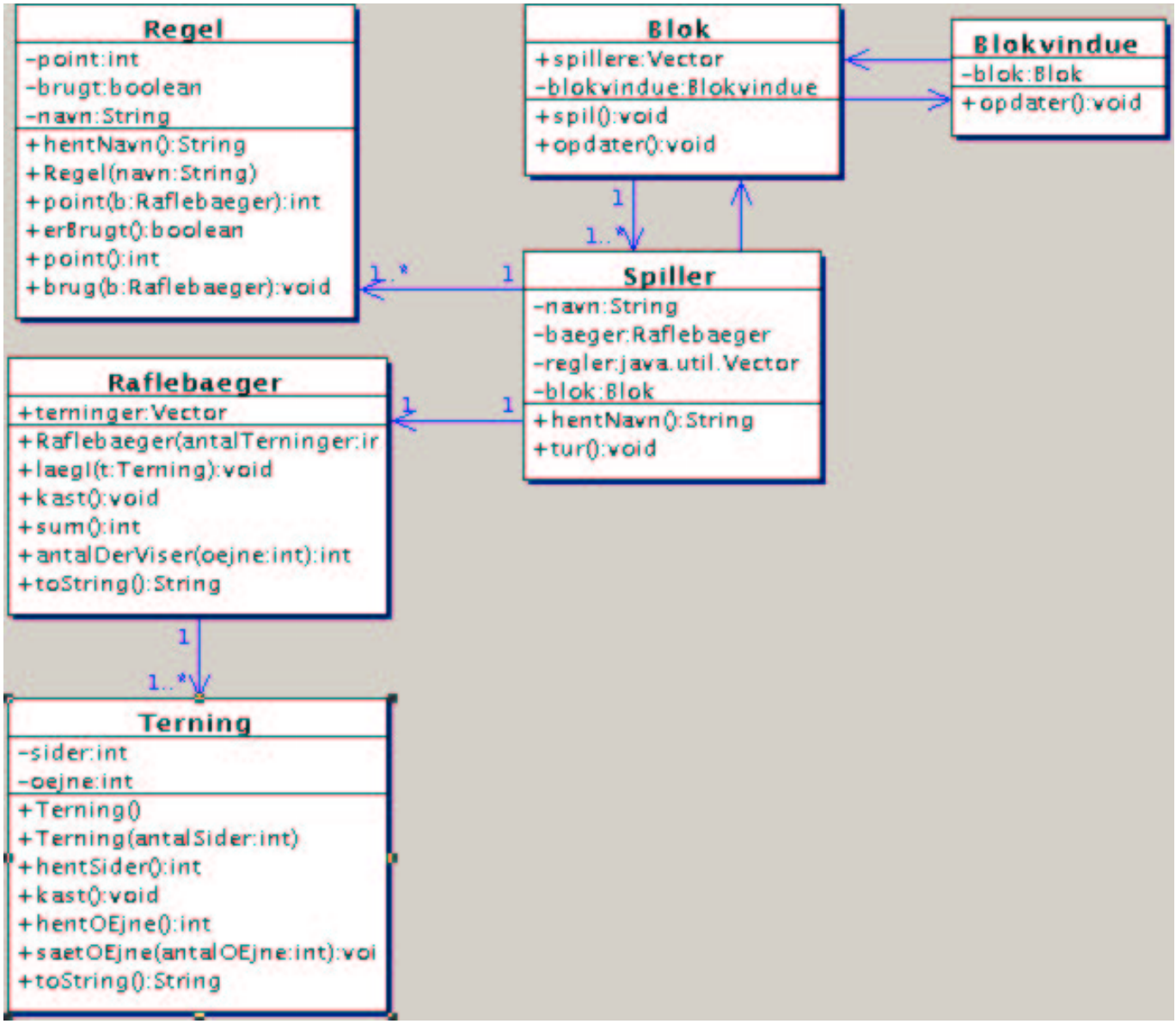
Designets formål er at beskrive, hvordan programmet skal implementeres.

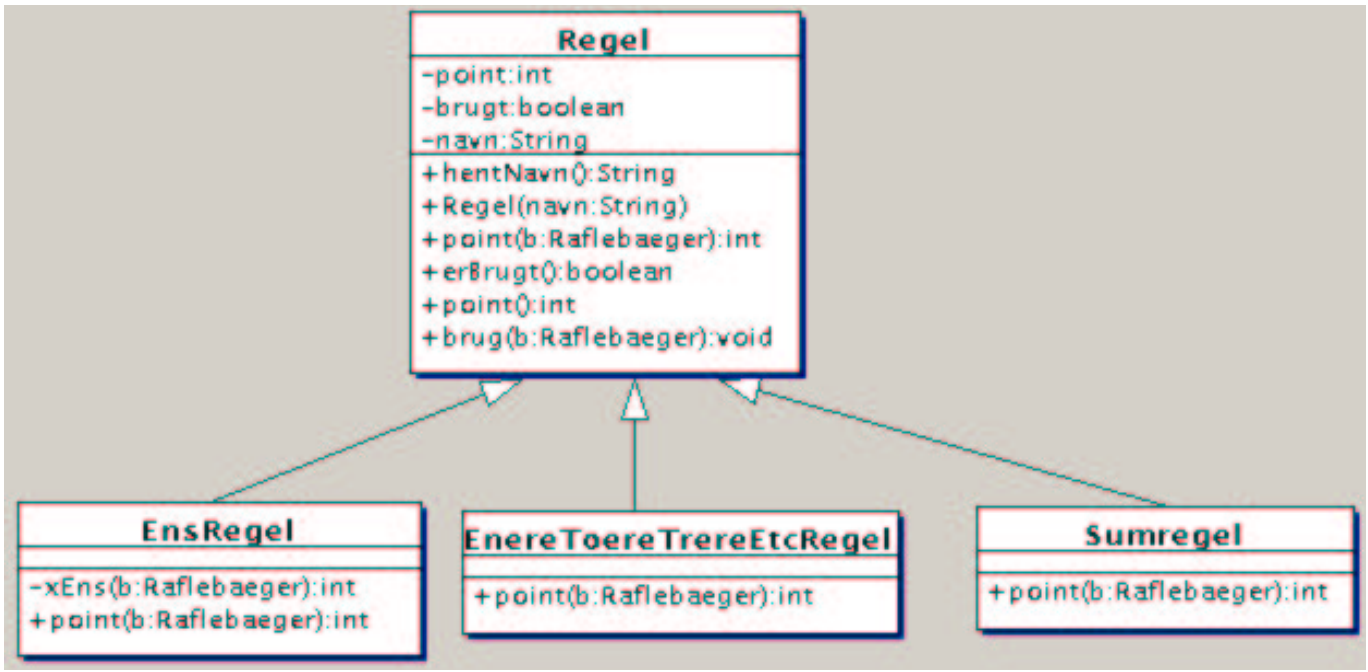
Kollaborationsdiagrammer





Klassediagrammer kan tegnes med et UML−værktøj





Virkefelt

En variabels virkefelt er de steder i programmet man kan "se" variabel−navnet.

```
main(...)
{
   int i;
   ...

   if (i>0)
   {
      int res=i;

   }
   else
   {
      int res=−i;
   }

   System.out.println("Den absolutte værdi af "+
                                          i+" er :");

   System.out.println(res);

}
```

En lokal variabels virkefelt er fra dens erklæring ned til afslutningen af den blok, den blev erklæret i.

En objekt−variabels virkefelt er hele klassen.

Demo:Grafiske programmer med JBuilder

åbn JBuilder

File/new project
File/new.../Applet: extends java.lang.Applet. Kald apletten StregApplet. Med standalone.

indsæt objektvariabel:

```
  Vector punkter = new Vector();
```

... og importer java.util.*

Design:

layout: BorderLayout
Swing Containers/JPanel placeres i bunden. Bemærk constraints=South. name ændres til knapPanel
Swing Containers/JPanel placeres i centrum. Bemærk constraints=Center. (den hedder nu JPanel2)

Klik på knapPanel. I midten er der en sort firkant. Swing/JButton placeres på denne firkant.
Swing/JToggleButton placeres på denne firkant.

Ændr text i JButton1 til "slet punkt" og JToggleButton1 til "flyt punkter"

Klik på JPanel2. klik på Events. klik til højre for MouseClicked, og tryk enter.

Den hopper nu over i en metode, der hedder jPanel2_mouseClicked(). udfyld den med:

```
if (!jToggleButton1.isSelected())
punkter.addElement(e.getPoint());
jPanel2.repaint();
```

Design: vælg jPanel2, og ændr name til tegnePlade

Source: File/new class. extends java.awt.Canvas. Kald klassen

TegnePlade.
lav en paint()−metode i TegnePlade (husk at importere alt som skal importeres − Du kan kopiere fra StregApplet):

Hvordan kommer TegnePlade til at kende Punkter ?

Lav en objektvariabel af typen StregApplet i TegnePlade−klassen, og lav en konstruktør der tager et StregApplet−objekt som parameter, og som sætter objektvariablen til parameteren.
Gå tilbage til StregApplet og vælg Source. find objektvariablen tegneplade, og ændre dens type til TegnePlade:

```
TegnePlade tegnePlade = new TegnePlade(this);
```

Nu kender TegnePladen altid sin applet, og kan dermed få fat i punkter−vektoren.

StregApplet

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.util.*;
import javax.swing.*;

public class StregApplet extends Applet {
//////////////////////////////////////
  Vector punkter=new Vector();
  Canvas tegnePlade;
  JPanel nedreKnapPanel = new JPanel();
  JButton sletPunktKnap = new JButton();
  JToggleButton flytPunkterKnap = new JToggleButton();
//////////////////////////////////////

  boolean isStandalone = false;
  BorderLayout borderLayout1 = new BorderLayout();

  /**Initialize the applet*/
  public void init() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  /**Component initialization*/
  private void jbInit() throws Exception {
    this.setLayout(borderLayout1);
    sletPunktKnap.setText("slet punkt");
    sletPunktKnap.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        sletPunktKnap_actionPerformed(e);
      }
    });
    tegnePlade = new TegnePlade(this);
    tegnePlade.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mouseClicked(MouseEvent e) {
        tegnePlade_mouseClicked(e);
      }
      public void mouseReleased(MouseEvent e) {
        tegnePlade_mouseReleased(e);
      }
    });
    flytPunkterKnap.setText("flyt punkter");
    flytPunkterKnap.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        flytPunkterKnap_actionPerformed(e);
      }
    });
    tegnePlade.addMouseMotionListener(new
java.awt.event.MouseMotionAdapter() {
      public void mouseDragged(MouseEvent e) {
        tegnePlade_mouseDragged(e);
      }
```

```
    });
    this.add(tegnePlade, BorderLayout.CENTER);
    this.add(nedreKnapPanel, BorderLayout.SOUTH);
    nedreKnapPanel.add(flytPunkterKnap, null);
    nedreKnapPanel.add(sletPunktKnap, null);
  }

  /**Main method*/
  public static void main(String[] args) {
    StregApplet applet = new StregApplet();
    applet.isStandalone = true;
    Frame frame;
    frame = new Frame() {
      protected void processWindowEvent(WindowEvent e) {
        super.processWindowEvent(e);
        if (e.getID() == WindowEvent.WINDOW_CLOSING) {
          System.exit(0);
        }
      }
      public synchronized void setTitle(String title) {
        super.setTitle(title);
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
      }
    };
    frame.setTitle("Applet Frame");
    frame.add(applet, BorderLayout.CENTER);
    applet.init();
    applet.start();
    frame.setSize(400,320);
    Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
    frame.setLocation((d.width − frame.getSize().width) / 2,
(d.height − frame.getSize().height) / 2);
    frame.setVisible(true);
  }

//////////////////////////////////////////////
  void tegnePlade_mouseClicked(MouseEvent e) {
    if (!flytPunkterKnap.isSelected()) {
      punkter.addElement(e.getPoint());
      tegnePlade.repaint();
    }
  }

  void sletPunktKnap_actionPerformed(ActionEvent e) {
    if (punkter.size()>0) punkter.removeElementAt(punkter.size()−
1);
    tegnePlade.repaint();
  }

  int punktSize=5;
  Point flytPunkt=null;

  void tegnePlade_mouseDragged(MouseEvent e) {
    if (flytPunkterKnap.isSelected()) {
      if(flytPunkt==null) {
        for (int i=0;i<punkter.size();i++) {
          Point p=(Point)punkter.elementAt(i);
          Rectangle r=new Rectangle(p.x−punktSize,p.y−
punktSize,punktSize*2,punktSize*2);
          if (r.contains(e.getPoint())) flytPunkt=p;
        }
      }
```

```
    if(flytPunkt!=null) {
      flytPunkt.x=e.getPoint().x;
      flytPunkt.y=e.getPoint().y;
      tegnePlade.repaint();
    }
   }
  }

  void tegnePlade_mouseReleased(MouseEvent e) {
   flytPunkt=null;
  }

  void flytPunkterKnap_actionPerformed(ActionEvent e) {
   tegnePlade.repaint();
  }
}
```

TegnePlade

```
import java.awt.Canvas;
import java.awt.*;
import java.applet.*;
import java.util.*;
import javax.swing.*;
public class TegnePlade extends Canvas {

  public void paint(Graphics g)
  {
   for (int i=1;i<applet.punkter.size();i++) {
    Point p1=(Point) applet.punkter.elementAt(i-1);
    Point p2=(Point) applet.punkter.elementAt(i);
    g.drawLine(p1.x,p1.y,p2.x,p2.y);
   }
   if (applet.flytPunkterKnap.isSelected()) {
    for (int i=0;i<applet.punkter.size();i++) {
     Point p=(Point) applet.punkter.elementAt(i);
     g.drawRect(p.x-applet.punktSize,p.y-applet.punktSize,
                applet.punktSize*2,applet.punktSize*2);
    }
   }
  }

  StregApplet applet;

  public TegnePlade(StregApplet ap) {
   applet=ap;
  }

  public TegnePlade() {
  }
}
```

Lektion 11

Interfaces
Eventuelt: Serialisering af objekter
Eventuelt: Netværkskommunikation
Vejledning i obligatorisk opgave

Interface

En klasse kan begrebsmæssigt opdeles i:
1)*Grænsefladen* − hvordan objekterne kan bruges udefra.
Dette udgøres af navnene på metoderne, der kan ses udefra.
2)*Implementationen* − hvordan objekterne virker indeni.
Dette udgøres af variabler og programkoden i metodekroppene.

Et ’interface’ svarer til punkt 1): En definition af, hvordan objekter bruges udefra. Man kan sige, at et interface er en "halv" klasse.

Et interface er en samling navne på metoder (uden krop)

```
import java.awt.*;

public interface Tegnbar
{
  public void sætPosition(int x, int y);

  public void tegn(Graphics g);
}
```

Implementere et interface

Lad os nu definere en klasse, der implementerer Tegnbar−interfacet.

> **En klasse kan erklære, at den *implementerer et interface*, og så skal den definere alle metoderne i interfacet og give dem en metodekrop**

Vi skal altså definere alle interfacets metoder sammen med programkoden, der beskriver hvad der skal ske, når metoderne kaldes.

```
import java.awt.*;

public class Stjerne implements Tegnbar
{
  private int posX, posY;

  public void sætPosition(int x, int y)   // kræves af Tegnbar
  {
    posX = x;
    posY = y;
  }

  public void tegn(Graphics g)            // kræves af Tegnbar
  {
    g.drawString("*",posX,posY);
  }
}
```

Variabler

```
   Tegnbar t;

   t = new Stjerne();   // Lovligt, Stjerne implementerer Tegnbar

   t = new Tegnbar();   // FEJL! Tegnbar er ikke en klasse

   t.sætPosition(5,5);  // OK, da Tegnbare objekt har metoden
```

Flere eksempler med Tegnbar−interfacet

```
public class GrafiskTerning extends Terning implements Tegnbar
{
```

```
public class GrafiskRaflebaeger
   extends Raflebaeger implements Tegnbar
{
```

En hvilken som helst klasse kan gøres til at være Tegnbar. Her er et tegnbart rektangel:

```
import java.awt.*;

public class Rektangel extends Rectangle implements Tegnbar
{
  public Rektangel(int x1, int y1, int width1, int height1)
  {
    super(y1,x1,width1,height1);
  }

  public void sætPosition(int x1, int y1)
  {
    x = x1;
    y = y1;
  }

  public void tegn(Graphics g)
  {
    g.drawRect(x,y,width,height);
  }
}
```

Polymorfi

```
   Tegnbar t;
   t = new Stjerne();
   t = new GrafiskTerning();
   t = new GrafiskRaflebaeger();
   t = new Rektangel();

   ...

   t.sætPosition(5,5);
```

En applet af tegnbare objekter

```
import java.applet.*;
import java.awt.*;
import java.util.*;

public class TegnbareObjekter extends Applet
{
 Vector tegnbare = new Vector();
 GrafiskRaflebaeger bæger = new GrafiskRaflebaeger();

 public void paint(Graphics g)
 {
  super.paint(g);
  for (int n=0; n<tegnbare.size(); n++)
  {
   Tegnbar t = (Tegnbar) tegnbare.elementAt(n);
   t.tegn(g);                                    // polymorfi!
  }
 }

 public void sætPositioner()
 {
  for (int n=0; n<tegnbare.size(); n++) {
   Tegnbar t = (Tegnbar) tegnbare.elementAt(n);
   int x = (int) (Math.random()*200);
   int y = (int) (Math.random()*200);
   t.sætPosition(x,y);                           // polymorfi!
  }
 }

 public void init() {
  Stjerne s = new Stjerne();
  tegnbare.addElement(s);

  tegnbare.addElement( new Rektangel(10,10,30,30) );

  tegnbare.addElement( new Rektangel(15,15,20,20) );

  GrafiskTerning t;
  t = new GrafiskTerning();
  bæger.tilføj(t);
  tegnbare.addElement(t);

  t = new GrafiskTerning();
  bæger.tilføj(t);
  tegnbare.addElement(t);

  tegnbare.addElement(bæger);

  sætPositioner();

  // mere kode her
  // ...
 }

 // flere metoder her
 // ...
}
```

Interfaces i standardbibliotekerne.

Sortering med Comparable

```
public class Element implements Comparable
{
  int x;

  public Element(int x1)
  {
    x = x1;
  }

  public String toString()
  {
    return "element"+x;
  }

  public int compareTo(Object obj)    // kræves af Comparable
  {
    Element andetElement = (Element) obj;

    if (x == andetElement.x) return 0;
    if (x  > andetElement.x) return 1;
    else return -1;
  }
}
```

Collections.sort() kan sortere lister af Comparable−objekter

```
import java.util.*;
public class BrugElementer
{
  public static void main(String args[])
  {
    Vector liste = new Vector();
    liste.addElement( new Element(5));
    liste.addElement( new Element(3));
    liste.addElement( new Element(13));
    liste.addElement( new Element(1));

    System.out.println("før: "+liste);
    Collections.sort(v);
    System.out.println("efter: "+liste);
  }
}
```

```
før: [element5, element3, element13, element1]
efter: [element1, element3, element5, element13]
```

Serialisering

```
import java.io.*;
public class Data implements Serializable
{
  public int a;
  public transient int tmp;

  public String toString()
  {
    return "Data: a="+a+" tmp="+tmp;
  }
}
```

```
import java.util.*;
public class HentOgGemData
{
  public static void main(String arg[]) throws Exception
  {
    Vector v;
    try {
      v = (Vector) Serialisering.hent("data.ser");
      System.out.println("Indlæst: "+v);
    } catch (Exception e) {
      v = new Vector();
      System.out.println("Oprettet: "+v);
    }

    Data d = new Data();
    d.a   = (int) (Math.random()*100);
    d.tmp = (int) (Math.random()*100);
    v.addElement(d);

    System.out.println("Gemt: "+v);
    Serialisering.gem(v,"data.ser");
  }
}
```

```
Oprettet: []
Gemt: [Data: a=88 tmp=2]
```

Køres programmet igen fås:

```
Læst: [Data: a=88 tmp=0]
Gemt: [Data: a=88 tmp=0, Data: a=10 tmp=10]
```

```
Læst: [Data: a=88 tmp=0, Data: a=10 tmp=0]
Gemt: [Data: a=88 tmp=0, Data: a=10 tmp=0, Data: a=52 tmp=96]
```

```
Læst: [Data: a=88 tmp=0, Data: a=10 tmp=0, Data: a=52 tmp=0]
Gemt: [Data: a=88 tmp=0, Data: a=10 tmp=0, Data: a=52 tmp=0, Data:
a=78 tmp=88]
```

Læg mærke til, at den transiente variabel tmp ikke bliver husket fra gang til gang.

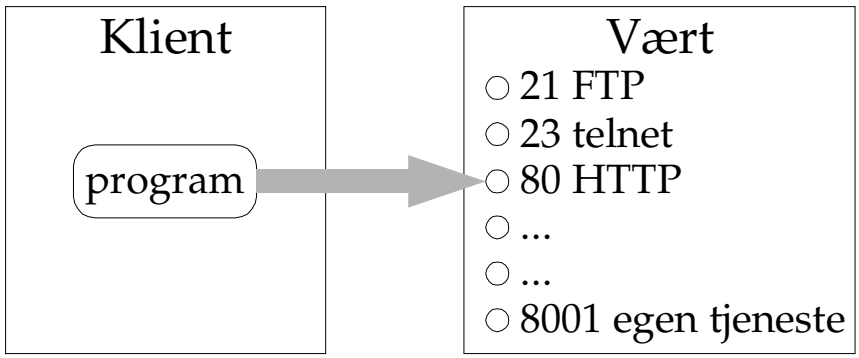
Kassen Serialisering er en hjælpeklasse med to metoder, der hhv. henter og gemmer objekter i en fil.

Du kan benytte klassen fra dine egne programmer.

```
import java.io.*;
public class Serialisering
{
  public static void gem(Object obj, String filnavn) throws IOException
  {
    FileOutputStream datastrøm = new FileOutputStream(filnavn);

    ObjectOutputStream p = new ObjectOutputStream(datastrøm);

    p.writeObject(obj);

    p.close();
  }


  public static Object hent(String filnavn) throws Exception
  {
    FileInputStream datastrøm = new FileInputStream(filnavn);

    ObjectInputStream p = new ObjectInputStream(datastrøm);

    Object obj = p.readObject();

    p.close();

    return obj;
  }
}
```

Netværkskommunikation

Kommunikation mellem to maskiner sker ved, at værtsmaskinen gør en tjeneste tilgængelig på en bestemt port, hvorefter klienter kan oprette en forbindelser til tjenesten.



Hjemmesider (HTTP−tjenesten) er tilgængelige på port 80.

En klient (der henter en hjemmeside)

```
import java.io.*;
import java.net.*;
public class HentHjemmeside
{
  public static void main(String arg[])
  {
    try {
      Socket forbindelse = new Socket("www.esperanto.dk",80);
      OutputStream fraOs = forbindelse.getOutputStream();
      InputStream  tilOs = forbindelse.getInputStream();
      PrintWriter    ud  = new PrintWriter(fraOs);
      BufferedReader ind = new BufferedReader(new InputStreamReader(tilOs));
      ud.println("GET / HTTP/0.9");
      ud.println("Host: www.esperanto.dk");
      ud.println();
      ud.flush();              // send anmodning afsted til værten
      String s = ind.readLine();
      while (s != null)        // null når datastrømmen lukkes
      {
        System.out.println("svar: "+s);
        s = ind.readLine();
      }
      forbindelse.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
svar: HTTP/1.1 200 OK
svar: Date: Tue, 17 Apr 2001 13:06:06 GMT
svar: Server: Apache/1.3.12 (Unix)  (Red Hat/Linux) PHP/4.0.2
mod_perl/1.24
svar: Content-Length: 896
svar: Content-Type: text/html
svar:
svar: <HTML><HEAD><TITLE>Esperanto.dk</TITLE>
svar: <META name="keywords" content="Esperanto, Danmark,
Esperanto-nyt, plansprog, kunstsprog, lingvistik, videnskab,
debat, Zamenhof, Danio, libroservo, planlingvo, teknika lingvo,
lingvistiko, debato">
svar: </HEAD>
svar: <h1>Velkommen til Esperanto.dk!</h1>
svar: Gå til <a href="da/velkomst.htm">velkomst-siden</a>
svar: eller til <a href="da/menu.htm">indholdsfortegnelsen</a>,
svar: </html>
```

Bemærk: Netop hjemmesider hentes nemmest med URL-klassen:

```
   URL url = new URL("http://www.esperanto.dk/");

   url.getInputStream();
```

En vært, der "serverer" hjemmesider

```
import java.io.*;
import java.net.*;
public class Hjemmesidevaert
{
  public static void main(String arg[])
  {
    try {
      ServerSocket værtssokkel = new ServerSocket(8001);
      while (true)
      {
        Socket forb = værtssokkel.accept();
        PrintWriter ud = new PrintWriter(forb.getOutputStream());

        BufferedReader ind = new BufferedReader(
          new InputStreamReader(forb.getInputStream()));

        String anmodning = ind.readLine();
        System.out.println("Anmodning: "+anmodning);

        ud.println("HTTP/0.9 200 OK");
        ud.println();
        ud.println("<html><head><title>Svar</title></head>");
        ud.println("<body><h1>Kære bruger</h1>");
        ud.println("Du har spurgt om "+anmodning+".");
        ud.println("</body></html>");
        ud.flush();
        forb.close();
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
Anmodning: GET / HTTP/0.9
Anmodning: GET / HTTP/1.1
Anmodning: GET /xxx.html HTTP/1.1
```

Flertrådet programmering

```
public class SnakkesagligPerson implements Runnable
{
  private String navn;
  private int ventetid;

  public SnakkesagligPerson(String n, int t)
  {
    navn = n;
    ventetid = t;
  }

  public void run()
  {
    for (int i=0; i<5; i++)
    {
      System.out.println(navn+": bla bla bla "+i);
      try { Thread.sleep(ventetid); } catch (Exception e) {}
    }
  }
}
```

```
public class BenytSnakkesagligePersoner
{
  public static void main(String arg[])
  {
    SnakkesagligPerson p = new SnakkesagligPerson("Jacob",150);
    Thread t = new Thread(p); // Ny tråd, klar til at køre
    t.start(); // .. Nu starter personen med at snakke...

    p = new SnakkesagligPerson("Troels",400);
    t = new Thread(p);
    t.start();

    // Det kan også gøres mere kompakt:
    new Thread(new SnakkesagligPerson("Henrik",200)).start();
  }
}
```

```
Jacob: bla bla bla 0
Troels: bla bla bla 0
Henrik: bla bla bla 0
Jacob: bla bla bla 1
Henrik: bla bla bla 1
Jacob: bla bla bla 2
Troels: bla bla bla 1
Henrik: bla bla bla 2
Jacob: bla bla bla 3
Henrik: bla bla bla 3
Jacob: bla bla bla 4
Troels: bla bla bla 2
Henrik: bla bla bla 4
Troels: bla bla bla 3
Troels: bla bla bla 4
```

En flertrådet webserver

```
import java.io.*;
```

```
import java.net.*;
import java.util.*;

public class Anmodning implements Runnable
{
  private Socket forbindelse;

  Anmodning(Socket forbindelse)
  {
    this.forbindelse = forbindelse;
  }

  public void run()
  {
    try {
      PrintWriter ud = new
PrintWriter(forbindelse.getOutputStream());
      BufferedReader ind = new BufferedReader(
        new InputStreamReader(forbindelse.getInputStream()));

      String anmodning = ind.readLine();
      System.out.println("start "+new Date()+" "+anmodning);

      ud.println("HTTP/0.9 200 OK");
      ud.println();
      ud.println("<html><head><title>Svar</title></head>");
      ud.println("<body><h1>Svar</h1>");
      ud.println("Tænker over "+anmodning+"<br>");
      for (int i=0; i<100; i++)
      {
        ud.print(".<br>");
        ud.flush();
        Thread.sleep(100);
      }
      ud.println("Nu har jeg tænkt færdig!</body></html>");
      ud.flush();
      forbindelse.close();
      System.out.println("slut  "+new Date()+" "+anmodning);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
import java.io.*;
import java.net.*;

public class FlertraadetHjemmesidevaert
{
  public static void main(String arg[])
  {
    try {
      ServerSocket værtssokkel = new ServerSocket(8001);

      while (true)
      {
        Socket forbindelse = værtssokkel.accept();
        Anmodning a = new Anmodning(forbindelse);
        new Thread(a).start();
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
start Tue Nov 28 15:37:31 GMT+01:00 2000 GET /xxx.html HTTP/1.0
start Tue Nov 28 15:37:38 GMT+01:00 2000 GET /yyy.html HTTP/1.0
start Tue Nov 28 15:37:42 GMT+01:00 2000 GET /zzz.html HTTP/1.0
slut  Tue Nov 28 15:37:42 GMT+01:00 2000 GET /xxx.html HTTP/1.0
slut  Tue Nov 28 15:37:49 GMT+01:00 2000 GET /yyy.html HTTP/1.0
start Tue Nov 28 15:37:50 GMT+01:00 2000 GET /qqq.html HTTP/1.0
slut  Tue Nov 28 15:37:53 GMT+01:00 2000 GET /zzz.html HTTP/1.0
slut  Tue Nov 28 15:38:01 GMT+01:00 2000 GET /qqq.html HTTP/1.0
```

Lektion 12

Hændelser

Klassevariabler og variablers virkefelt

Arrays

Evt. intro til flertrådet programmering

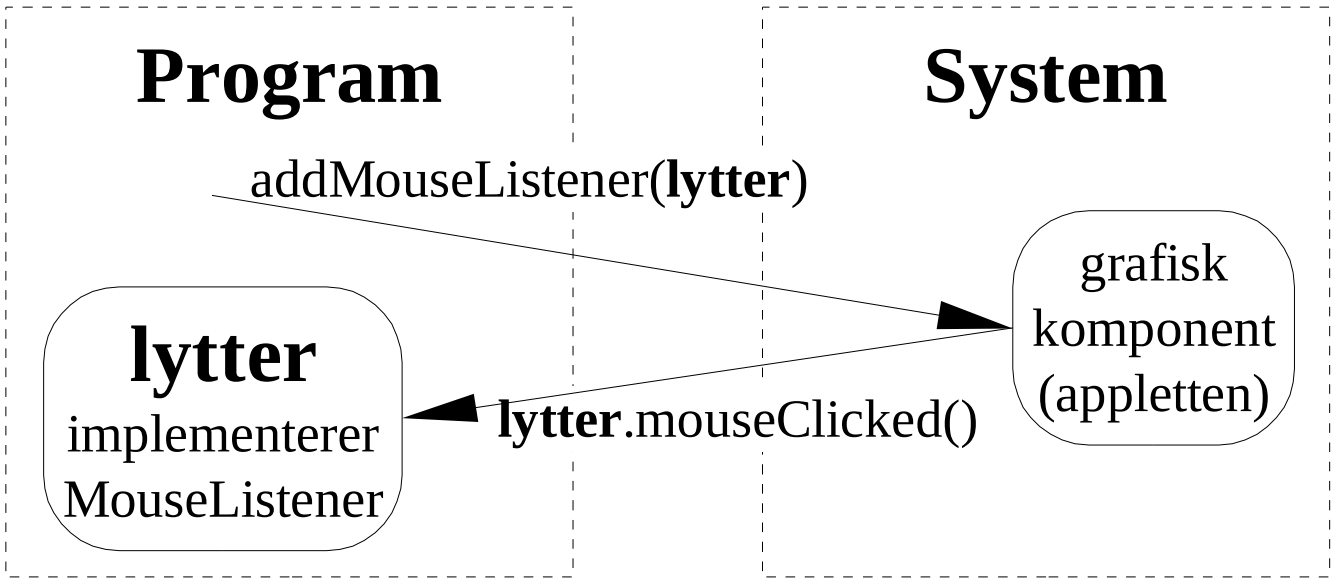
Efterfølgende ITD−kurser

Trinvis gennemgang
(det *skal* I prøve i kurset her!)

Hændelser

Når brugeren foretager en handling, f.eks. bevæger musen, klikker, trykker en knap ned, ændrer i et tekstfelt osv., opstår der en *hændelse*.

Hvis man vil behandle en bestemt type hændelser fra en komponent, skal man ”lytte” efter den hændelse. Det gøres ved at registrere en *lytter* på komponenten.



Lytte efter muse−hændelser på en applet:

```
import java.applet.*;
public class LytTilMusen extends Applet
{
  public void init()
  {
    Muselytter lytter = new Muselytter();
    this.addMouseListener(lytter);    // this er appletten selv
  }
}
```

Lytteren skal i dette tilfælde implementere interfacet **MouseListener**

```
import java.awt.*;
import java.awt.event.*;

public class Muselytter implements MouseListener
{
  public void mousePressed(MouseEvent hændelse)
  {
    Point trykpunkt = hændelse.getPoint();
    System.out.println("Mus trykket ned i "+trykpunkt);
  }

  public void mouseReleased(MouseEvent hændelse)
  {
    Point slippunkt = hændelse.getPoint();
    System.out.println("Mus sluppet i "+slippunkt);
  }

  public void mouseClicked(MouseEvent hændelse)
  {
    System.out.println("Mus klikket i "+hændelse.getPoint());
  }

  //  Ubrugte hændelser
  // (skal defineres for at implementere MouseListener)
  public void mouseEntered (MouseEvent event) {}
  public void mouseExited (MouseEvent event) {}
}
```

Uddata fra apletten kan ses i konsolvinduet
(Netscape: Communcator/Tools/Java Console):

```
Mus trykket ned i java.awt.Point[x=132,y=209]
Mus sluppet i java.awt.Point[x=139,y=251]
Mus trykket ned i java.awt.Point[x=101,y=199]
Mus sluppet i java.awt.Point[x=101,y=199]
Mus klikket i java.awt.Point[x=101,y=199]
```

Linietegning

Hvordan får vi information fra lytteren til appletten?

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class Linietegning extends Applet
{
  public Point trykpunkt;
  public Point slippunkt;

  public void init()
  {
    Linielytter lytter = new Linielytter();
    // initialiserer lytterens reference til appletten
    lytter.appletten = this;
    this.addMouseListener(lytter);
  }

  public void paint(Graphics g)
  {
    g.drawString("1:"+trykpunkt+"  2:"+slippunkt,10,10);
    if (trykpunkt != null && slippunkt != null)
    {
      g.setColor(Color.blue);
      g.drawLine(trykpunkt.x,trykpunkt.y,slippunkt.x, slippunkt.y);
    }
  }
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

public class Linielytter implements MouseListener
{
  public Linietegning appletten;

  public void mousePressed(MouseEvent hændelse)
  {
    appletten.punkt1 = hændelse.getPoint();
  }

  public void mouseReleased(MouseEvent hændelse)
  {
    appletten.punkt2 = hændelse.getPoint();
    appletten.repaint();
  }

  //-----------------------------------------------------------------------
  //  Ubrugte hændelser (skal defineres for at implementere interfacet)
  //-----------------------------------------------------------------------
  public void mouseClicked(MouseEvent event) {} // kræves af MouseListener
  public void mouseEntered(MouseEvent event) {} // kræves af MouseListener
  public void mouseExited (MouseEvent event) {} // kræves af MouseListener
}
```

Tekstredigering

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class LytTilKnap extends Applet implements ActionListener
{
  private TextArea t1, t2;
  private Button kopierKnap;

  public void init()
  {
    String s = "Her er en tekst.\nMarkér noget og tryk Kopier...";
    t1 = new TextArea(s, 5,20);
    add(t1);
    kopierKnap = new Button("Kopiér>>");
    kopierKnap.addActionListener(this);
    add(kopierKnap);
    t2 = new TextArea( 5,20);
    t2.setEditable(false);
    add(t2);
  }

  // kræves af ActionListener
  public void actionPerformed(ActionEvent e)
  {
    t2.setText(t1.getSelectedText() );
  }
}
```

JBuilder

Værktøjet genererer i jbInit() koden:

```
    buttonKast.addActionListener(
      new java.awt.event.ActionListener() {

      public void actionPerformed(ActionEvent e) {
        buttonKast_actionPerformed(e);
      }
    });
```

Det er en anonym indre klasse der implementerer ActionListener()− interfacet, der bliver tilføjet som lytter til buttonKast. Den anonyme klasse har kun én metode (actionPerformed), og der sker kun én ting, nemlig at buttonKast_actionPerformed bliver kaldt.

Den er defineret andetsteds i koden, og her kan vi sætte vores kode ind:

```
  void buttonKast_actionPerformed(ActionEvent e) {
   // vores programkode her...
  }
```

Arrays

Et array er en række data af samme type

```
public class ArrayEksempel1
{
  public static void main(String[] args)
  {
    int[] arr = new int[6];
    arr[0] = 28;
    arr[2] = 13;

    arr[3] = arr[0] + arr[1] + arr[2];

    int længde = arr.length;

    for (int i=0; i<længde; i=i+1)
       System.out.print( arr[i] + " " );
    System.out.println();
  }
}
```

```
28 0 13 41 0 0
```

Initialisering med startværdier

```
public class Maaneder
{
  public static void main(String[] args)
  {
    int[]måneder= {31,28,31,30,31,30,31,31,30,31,30,31};

    System.out.println("Længden af januar er: " + måneder[0]);
    System.out.println("Længden af april er: " + måneder[3]);

    for (int i=0; i < måneder.length; i=i+1)
      System.out.print(måneder[i] + ", ");

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

```
Længden af januar er: 31
Længden af april er: 30
31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31,
```

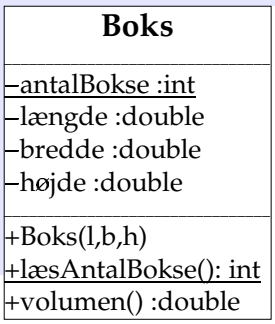
Arrays af objekter

```
public class MaanedersNavne
{
  public static void main(String[] args)
  {
    String[] måneder = {
      "januar", "februar", "marts", "april", "maj", "juni",
      "juli", "august", "september", "oktober", "november",
"december" };

      System.out.println("Den 1. måned er " + måneder[0] );
      System.out.println("Den 6. måned er " + måneder[5] );
      System.out.println("Den 9. måned er " + måneder[9] );
  }
}
```

```
Den 1. måned er januar
Den 6. måned er juni
Den 9. måned er oktober
```

Lokale, objekt− og klassevariable

```
public class Boks4
{
  private double længde;              // objektvariabel
  private double bredde;              // objektvariabel
  private double højde;               // objektvariabel
  private static int antalBokse;      // klassevariabel

  public Boks4(double l, double b, double h)
  {
    // l, b og h er lokale variabler
    længde = l;
    bredde = b;
    højde  = h;
    antalBokse = antalBokse + 1;
  }

  public static int læsAntalBokse()   // klassemetode
  {
    return antalBokse;
  }

  public double volumen()
  {
    // vol er en lokal variabel
    double vol;
    vol = længde*bredde*højde;
    return vol;
  }
}
```




```
public class BenytBoks4
{
  public static void main(String args[])
  {
    System.out.println("Antal bokse: "+ Boks4.læsAntalBokse());

    Boks4 boksen;
    boksen = new Boks4(2,5,10);

    System.out.println("Antal bokse: "+ Boks4.læsAntalBokse());

    Boks4 enAndenBoks, enTredjeBoks;
    enAndenBoks = new Boks4(5,5,10);
    enTredjeBoks = new Boks4(7,5,10);

    System.out.println("Antal bokse: "+ Boks4.læsAntalBokse());
  }
}
```

```
Antal bokse: 0
Antal bokse: 1
Antal bokse: 3
```

Variabler erklæret **static** kaldes statiske variabler eller klassevariable.

Metoder erklæret **static** kaldes statiske metoder eller klassemetoder.

Vi har brug for statiske metoder og variabler i 2 tilfælde:

- Hvis vi ønsker at have fælles variabler for samtlige objekter af en klasse.
- Hvis vi ønsker at kunne anvende variabler eller metoder uden at skulle lave nye objekter.

Tilknytning

Objektvariabler og −metoder er altid tilknyttet et konkret objekt, oprettet med f.eks.

```
boksen = new Boks().
```

En klassevariabel eksisterer kun ét sted i hukommelsen, uanset hvor mange objekter der oprettes, og alle objekterne deler klassevariablen.

Klassemetoder og −variabler er ikke tilknyttet noget konkret objekt og kan altid anvendes.

Adgang til variabler og metoder

Objektvariabler anvendes uden for objektet ved at skrive objektnavn.variabel.

Objektmetoder anvendes uden for klassen ved at skrive objektnavn.metode(), f.eks.

```
boksen.volumen()
```

Klassevariabler anvendes uden for klassen ved at skrive Klassenavn.variabel.

Klassemetoder anvendes uden for klassen ved at skrive Klassenavn.metode(), f.eks.

```
Boks.læsAntalBokse()
```

Adgang fra metoder

Almindelige metoder kan uden videre anvende andre metoder, variabler, klassemetoder og klassevariable.

Klassemetoder kan kun anvende klassemetoder og klassevariable. De kan ikke anvende objektvariabler eller −metoder (til disse skal der være et objekt).

Lektion 13

Lektion 14

public, protected og private

Variabler og metoder erklæret public er altid tilgængelige, både inden og uden for klassen.

Variabler og metoder erklæret protected er tilgængelige for alle klasser inden for samme pakke.
Klasser i andre pakker kan kun få adgang hvis de er nedarvinger.

Skriver man ingenting er det kun klasser i samme pakke der har adgang til variablen eller metoden.

private er det mest restriktive. Hvis en variabel eller metode er erklæret private, kan den kun benyttes indenfor samme klasse (og derfor kan den ikke tilsidesættes med nedarving).

Adgang	***public***	***protected***	***(ingenting)***	***private***
i samme klasse	ja	ja	ja	ja
klasse i samme pakke	ja	ja	ja	nej
arving i en anden pakke	ja	ja	nej	nej
klasse der ikke arver i en anden pakke	ja	nej	nej	nej

Holder man sig inden for samme pakke er der altså ingen forskel mellem public, protected og ingenting.

Nøgleordet final

En variabel der er erklæret final kan ikke ændres når den først har fået en værdi.

```
public class X
{
  public final int a=10;

  //..
}
```

Herover kan a's værdi ikke ændres. final styrer ikke synlighed, men kan bruges sammen med public, protected og private.

Det kan også bruges på lokale variable (hvor public, protected og private aldrig kan bruges):

```
  public static void main(String args[])
  {
    final Vector v = new Vector();

    //v = new Vector();  // ulovligt! v kan ikke ændres.
```

```
  public static void main(String args[])
  {
    final Vector v = new Vector();

    // lovligt, v refererer stadig til samme objekt
    v.addElement("Hans");
```

Nøgleordet abstract

Noget der er erklæret abstract er ikke implementeret, og skal defineres i en nedarving.

Klasser

```
public abstract class X
{
  public void a()
  {
    //..
  }
}
```

Det er forbudt at oprette objekter fra en abstrakt klasse

```
  public static void main(String args[])
  {
    X x = new X();  // ulovligt! X er abstrakt
```

Metoder

En metode erklæret abstract har kun et metodehovede men ingen krop. Den kan kun erklæres i en abstrakt klasse

```
public abstract class X
{
  public abstract void a();
}
```

Nedarvinger skal definere de abstrakte metoder, eller selv også være abstrakte

```
public class Y extends X
{
  public void a()
  {
    //..
  }
```

```
}
```

Indre klasser

- (Almindelige) indre klasser
- Lokale klasser
- Anonyme klasser

Almindelige indre klasser

```
public class YdreKlasse
{

  class AlmindeligIndreKlasse
  {
  }
}
```

Koden i den indre klasse kan derfor anvende alle den ydre klasses variabler og metoder − også de private. Den er altid knyttet til en forekomst af den ydre klasse.

Eksempel – Linietegning

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class LinietegningIndre extends Applet
{
  private Point trykpunkt = null;
  private Point slippunkt = null;

  public void init()
  {
    Linielytter lytter = new Linielytter();
    this.addMouseListener(lytter);
  }


  // En indre klasse
  class Linielytter implements MouseListener
  {
    public void mousePressed (MouseEvent event)
    {
      // sæt den ydre klasses variabel
      trykpunkt = event.getPoint();
    }

    public void mouseReleased (MouseEvent event)
    {
      slippunkt = event.getPoint();
      repaint();
    // kald den ydre klasses metode
    }

    // kræves af MouseListener:
    public void mouseClicked (MouseEvent event) {}
    public void mouseEntered (MouseEvent event) {}
    public void mouseExited (MouseEvent event) {}
  } // slut på indre klasse

  // en metode i den ydre klasse
  public void paint (Graphics g)
  {
    if (trykpunkt != null && slippunkt != null)
      g.drawLine (trykpunkt.x, trykpunkt.y,
        slippunkt.x, slippunkt.y);
  }
}
```

Læg mærke til at den indre klasse uden videre har adgang til den ydre klasses variabler og metoder.

Lokale klasser

En lokal klasse er defineret i en blok programkode, ligesom en lokal variabel.

```
public class YdreKlasse
{
  public void metode()
  {
    // ...

    class LokalKlasse
    {
      // metoder og variabler her ...
    }

    LokalKlasse objektAfLokalKlasse = new LokalKlasse();

    // ...
  }
}
```

```
public class YdreKlasseMedLokalKlasse
{

  private char c = 'c';
  private char d = 'd';

  public void prøvLokaltObjekt(final char e)
  {
    final char f= 'f';

    class LokalKlasse {
      char g = 'g';
      public void udskriv()
      {
        System.out.println( g );
        System.out.println( f );
        System.out.println( e );
        System.out.println( d );
        System.out.println( c );
      }
    } // slut på lokal klasse

    LokalKlasse lokal = new LokalKlasse();
    lokal.udskriv();
  }

  public static void main(String args[]){
    YdreKlasseMedLokalKlasse ydre
       = new YdreKlasseMedLokalKlasse();
    ydre.prøvLokaltObjekt('e');
  }
}
```

```
g
f
e
d
c
```

Anonyme klasser

En anonym klasse er en lokal klasse uden klassenavn, som der oprettes netop ét objekt ud fra:

```
public class YdreKlasse
{
  public void metode()
  {

    // ... programkode for metode()

    X objektAfAnonymKlasse = new X()
    {
      void metodeIAnonymKlasse()
      {
        // programkode
      }
      // flere metoder og variabler i anonym klasse
    }

    objektAfAnonymKlasse.metodeIAnonymKlasse();

    // mere programkode for metode()

  }
}
```

Eksempel – filtrering af filnavne

```
import java.io.*;
public class FilnavnfiltreringMedAnonymKlasse
{
  public static void main(String arg[])
  {
    File f = new File( "." );

    FilenameFilter filter;

    filter = new FilenameFilter()
       { // En anonym klasse
         public boolean accept( File f, String s)
         {
           return s.endsWith( ".java");
         }
       } // slut på klassen;
     ; // slut på tildelingen filter = new ...

    // brug objektet som filter i en liste af filer
     String[] list = f.list( filter );

    for (int i=0; i<list.length; i=i+1)
     System.out.println( list[i] );
  }
}
```

```
IndreKlasser1.java
A.java
FilnavnfiltreringMedAonymKlasse.java
LinietegningAnonym.java
YdreKlasseMedLokalKlasse.java
LinietegningIndre.java
```

Eksempel − Linietegning

Udviklingsværktøjer benytter ofte anonyme klasser i forbindelse med at lytte efter hændelser. Her er Linietegning−eksemplet igen hvor vi bruger en anonym klasse som lytter.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class LinietegningAnonym extends Applet
{
  private Point trykpunkt = null;
  private Point slippunkt = null;

  public void init()
  {
    this.addMouseListener(
      new MouseListener()
      {
        public void mousePressed (MouseEvent event)
        {
          trykpunkt = event.getPoint();
        }

        public void mouseReleased (MouseEvent event)
        {
          slippunkt = event.getPoint();
          repaint();
        }

        public void mouseClicked (MouseEvent event) {}
        public void mouseEntered (MouseEvent event) {}
        public void mouseExited (MouseEvent event) {}
      } // slut på anonym klasse
    ); // slut på kald til addMouseListener()

    System.out.println("Lytter−objekt oprettet");
  }

  public void paint (Graphics g)
  {
    if (trykpunkt != null && slippunkt != null)
      g.drawLine (trykpunkt.x, trykpunkt.y,
        slippunkt.x, slippunkt.y);
  }
}
```

Eksempel − tråde

Her opretter vi 5 tråde som anonyme klasser. Der oprettes ét tråd−objekt per gennemløb af for−løkken

```
public class AnonymeTraade
{
  public static void main(String arg[])
  {

    for (int i=0; i<5; i++)
    {
      final prioritet = i;
      Thread t = new Thread() {
        public void run()
        {
          setPriority(Thread.MIN_PRIORITY + prioritet);
          char tegn = 'a' + prioritet;
          for (int j=0; j<50; j=j+1)
            System.out.print(tegn);
          System.out.print("Færdig med "+tegn+".");
        }
      }
      t.start();
    }
  }
}
```