LITT OM PROGRAMMERING

Noen råd om programmering knyttet til oblig. 1

For forelesning 18/9 2006

MAT-INF1100

Hvorfor programmering i MAT-INF1100

- Programmering er viktig i realfag. Vi vil integrere det i kurs fra starten av.
  - Mange får bruk for programmering i masteroppgaver o.l.
  - Kunnskaper om et språk som Java nyttig bakgrunn for all databehandling
- Koding av algoritmer gir bedre følelse med hva de er og hva som kan gjøres med dem. Noen ganger økes også forståelsen av matematikken bak

Programmering ingen heksekunst, men krever trening

Programmer ukesoppgaver i INF1000 og MAT-INF1100

Siktemål med oblig. 1

- Få dere til å skrive noen programmer i Java *og få dem til å virke*
- La dere utforske fenomener som "overflow", udefinerte størrelser og avrundingsfeil.
  Det blir mye fokus på det nå fordi det er tidlig i kurset.
- Dere gjør dere litt kjent med de ulike typene (**float**, **double** etc.)
- Koding av de første løkkestrukturene – utregning av ting man ikke får til greit for "hånd" (primitiv kalkulator)

Samarbeid gjerne, men mitt råd er: kod til slutt alt selv.

Javating

Numeriske ting ikke spesielt vektlagt i "Rett på Java", men du finner det du trenger

- De innledende "besvergelser" i tallrike eksempler
- Deklarasjoner typer og konstanter i 2.4
- Matematiske funksjoner i tabell 2.4 – klassen Math
- Lokker og forgreninger i 4.2-4.3 (**if**, **while**, **for**)

Se også kompendium og eget skriv med tips (peker fra obligsiden)

Klassen Math

Rett på Java, tabell 2.4

y=Math.sqrt(x);	$y = \sqrt{x}$
z=Math.pow(x,y);	$z = x^y$
y=Math.sin(x);	$y = \sin x$
...	...

Se skriv "Litt om Javas..." (tips til oblig) og http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Math.html for detaljer.

Obs: **z=Math.pow(x,y)** er generelt *ikke* egnet for heltallspotenser

Typer og konvertering

Hierarki

int	32 bits heltall
long	64 bits heltall
float	32 bits flyttall
double	64 bits flyttall

1. Automatisk konvertering går nedover i listen
2. Konvertering oppover må gjøres eksplisitt
3. Standard type for konstanter er **int** og **double**
   2134234356L angir **long**, mens **4.56F** angir **float**
4. Funksjoner **Math.*** kan ha diverse input.
   Ofte returneres **double** selv om argumentet er **float**

Typetukling i Java 1

```
float a=1.5;            /* feil */
float a=1.5F;           /* ok */
a=2.37;                 /* feil */
a=2.37F;                /* ok */
a=(float)2.37;          /* ok */
a=(float)2.37*3.1;      /* feil */
a=(float)(2.37*3.1);    /* ok */
a=Math.cos(a);          /* feil */
a=(float)Math.cos(a);   /* ok */
```

Typetukling i Java 2

```
float a=1.5F;           /* ok */
double b=4.76;          /* ok */
a=b;                    /* feil */
b=a;                    /* ok */
a=(float)b;             /* ok */
b=1/2;                  /* gir b=0.0 */
b=1.0/2.0;              /* gir b=0.5 */
long ii=101;            /* ok */
long ii=123123243434;   /* feil */
long ii=123123243434L;  /* ok */
a=ii;                   /* ok ???? */
```

Regneoperasjoner og typer

- Addisjon, subtraksjon og muliplikasjon virker ganske likt for alle typer
- Divisjon "/" er ulik for heltall og flytende tall
  NB! *Heltallsdivisjon*⇒ resten sløyfes
  Eksempler: **13/8** gir 1, **-8/3** gir -2

Når en operasjon mellom to tall utføres gjøres begge om til den mest generelle typen. feks. vil

3.0/8

utføres som divisjon mellom to **double**.

Programmeringseksempel

Kompendium, oppgave 2.1(e): Beregn

$$\sum_{i=1}^{1000} \sin\left(\frac{i}{50}\right)$$

1. Skriv program **sum.java**
2. Kompiler **javac sum.java**
3. Kjør **java sum** (sum er klassenavn i sum.java)