FÖRELÄSNING 3

Variabler

L&L avsn. 1.4, 2.1 - 2.6

- Ett **namn**, den heter något, t ex rot.
- Ett **värde**, det värde som just nu associeras med variabeln.
- En **typ**, anger vilka värden variabeln kan anta, t ex heltal, reella tal, tecken.
- En **adress**, anger var i primärminnet värdet finns (okänt för programmeraren).

Namn

Giltiga namn Kurt, kurt48, Kurt_med_streck, a, A, å, ä, ö, $.

Ogiltiga namn 48d, x+2 - de måste börja med en bokstav, bara tillåtna tecken

Reserverade ord, fördefinierade public, main, class, for ...

Java är *case-sensitive*.

Namnkonventioner

- Namn på klasser börjar med versal
- Konstanter skrivs med versaler (t ex PI för π).
- Metoder och variabler skrivs med gemener
- Versaler används för att öka läsligheten (t ex linesPerPage)
- Använd namn som ökar läsbarheten i programmet.

Titta på appendix F(L&L) för att få guidning hur man skriver snygga program. Titta gärna på Appendix A för att få korta definitioner av alla programmeringsrelaterande ord.

Primitiva datatyper, L&L avsn. 2.3

heltal Exempel: 0; -7; 3;
`byte` (1 byte) Talområde: -128 - +127
`short` (2 bytes) Talområde: -32 768 - +32 767,
`int` (4 bytes) största talet: 2.1475e+09
`long` (8 bytes) största talet: 9.22e+18

reella tal Exempel: 3.14; 7.9; -0.1;
`float` (4 bytes)
`double`(8 bytes)

FLOAT
noggrannhet: 7 siffror
Största talet: 3.40282e+38

DOUBLE
noggranhet 15 siffror
Största talet: 1.79769e+308

boolean: logiskt, true eller false

tecken Exempel: 'a', 'A', '+', `char` (2 bytes).

Variabeldeklaration och initiering L&L avsn. 2.2

```
int ränta;
double x, y; int putte;
int musse, mimmi=-6; //mimmi blir -6
char tecken='C'; //obs apostroferna
int pluto=7+4; //startvärde elva
double tal=2.3*4; // startvärde 9,2
final double ÅR = 17;// ÅR blir en konstant
```

Tilldelningssatsen L&L avsn. 2.2

Satsen ser ut på följande sätt:
variabel=uttryck Läs från vänster till höger, exempel: int x = 3;
OBS! Man kan *inte* skriva som i matematik 3 = x FEL!!!

`int` a, b, c;
`double` x, y;
$a = 2$;
$b = 3 * 2 + 4$;
$c = a + b$; //(c=2+10=12)
$x = 4.5$;

Aritmetiska operatorer, L&L avsn. 2.4

Högerledet kan vara någonting som blir ett värde genom uträkning, vi kallar detta för ett *uttryck*. Ett uttryck kan innehålla tal, variabler, funktion och *operationer* på sådana. Det finns en speciell prioritetsordning mellan de olika operatorerna. *, /, % (modulsoperatorn) har högre prioritet än + och -.

Ex:

`int` a, b, c, d;

`double` x, y, z;

$a = 2$;

$b = 3 * a$; // b blir 6

$c = 5$;

d=c/b+a; // d blir 2 eftersom a,b,c,d är int, $\frac{5}{6} + 2 = 2.83$

$a = 7.45$; // a får värdet 7 ty a är deklarerad till int.

$x = 2$;

$y = a/x$; // y blir 3.5

$y = a/c$; // y blir 1 ($\frac{7}{5} = 1$)

$c = c + 5$; //"Det nya värdet på c = det gamla värdet på c+5

$a = a * 8$;

$d = 15\%7$; // d=1; $15 - 2 * 7 = 1$, resten från heltalsdivision.

Speciella operatorer

Ex:

`int` i=1, x=6;

$i++$; //++ operatorn ökar värdet med 1.

$i = i + 1$; //Samma som ovan

$i+ = 4$; //Samma som i=i+4

$x--$; // $--$ operatorn minskar värdet med 1.

$x- = 1$; //Samma som ovan

$x = x * 4$;

$x* = 4$; //Samma som ovan

$x = x/2$;

$x/ = 2$; //Samma som ovan

Exempel: Datum

```
public class F3E1 {
  public static void main(String [] arg) {
    int datum = 931215, år, månad, dag, temp;

    temp  = datum/100;  // 9312
```

```
    år    = temp/100;   // 93
    månad = temp%100;   // Använd modulus för att komma
                        // åt resten.  9312 - 93*100 = 12
    dag   = datum%100;  // 15

    System.out.println(Datumet 931215 blir på ny form:\n"
               + dag + "/" + månad + -" +  år);
  }
}
```

Math klassen, L&L avsn. 3.5

Klassen `Math` innehåller trigonometriska, exponentiella och andra operationer. Man har alltid tillgång till Math-klassen.

$sinx$ `------>` `Math.sin(x)`
x^2 `------>` `Math.pow(x,2)`
$\sqrt{3+6}$ `------>` `Math.sqrt(3+6)`
Se sid. 99.

Det finns ett stort bibliotek i Java, Java-dokumentationen på
`http://java.sun.com/javase/7/docs/api/`

Explicit typkonvertering (conversions) L&L avsn. 2.5

```
public class F3E2 {
  public static void main (String [] arg) {
    int a=2, b=5;
    double d;
    d = a; //tillåtet - en int får plats i en double
    //a = d är inte tillåtet
    //MEN a = (int)d är tillåtet
    d = a/b;          // d har värdet 0!!
    System.out.println(" d= " + d);
    // Variant.
    d = 1.0*(a)/b;    // Java konverterar till double,
                      // dock inte så snyggt.
```

```
    System.out.println(" d= " + d);
    // Istället.
    d = (double)a/b; // d får nu värdet 0.4.
    // Explicit typkonvertering.
    System.out.println(" d= " + d);
  }
}
```

Med **explicit typkonvertering** tvingar vi Java att byta typen. Expl. typom. kan göras till alla typer.

Paket och bibliotek

De allra flesta Javainstallationer kommer med en stor mängd bibliotek som innehåller olika klasser. Dessa klasser kan användas för att underlätta programmeringen eftersom många vanliga funktioner finns implementerade i dessa bibliotek. Relaterade bibliotek är ofta sammansatta till paket, så att de ska vara enklare att hålla reda på. För att använda bibliotek och paket måste dessa importeras med kommandot *import*, som alltid ska skrivas först i ett program. Paketet *java.lang*, som bl a innehåller klasserna *String* och *System*, importeras automatiskt eftersom den är av fundamental betydelse. Om ni t ex behöver mata in från tangentbordet måste ett objekt av klassen *Scanner* skapas. Scanner-klassen måste importeras. Skriv *import java.util.Scanner;* överst i klassen.

In- och utmatning

Klassen `Scanner`, L&L avsn. 2.6

Hittills har vi bara skrivit ut text på skärmen, inte 'läst in' data till programmen. I Java använder man sig av *strömmar* för att läsa in respektive skriva ut data.

- *Utström*: en *ström* med tecken skickas till skärm eller fil.
- *Inström*: en *ström* med tecken kommer från tangentbord eller fil.

.

Vi kommer i huvudsak använda två av de färdiga strömobjekten:

- `System.out` representerar *standard output*-strömmen som normalt är kopplad till skärmen. `System.out` är ett objekt av kalssen `PrintStream`. Där finns de vanlig metoderna för utskrift, `print` och `println`, se exempel nedan. Strömmen `System.out` använder en buffert. Det betyder att det kan hända att utskriften inte omedelbart syns i textfönstret. För att vara säker på att texten syns direkt kan man använda metoden `flush` efter print-satsen:

  ```
  System.out.flush();
  ```

  För att skriva ut tal och text på önskat sätt, t ex med tre decimaler, se sid 161 – 164, `DecimalFormat`-klassen respektive sid 135, `printf`-metoden.

- `System.in` representerar *standard input*-strömmen som normalt är kopplad till tangentbordet. Till klassen `Scanner` finns metoder som kan läsa in siffror, tecken och texter, se sid 115.

  Exempel:

```
import java.util.Scanner; //OBS!

public class InputDemo {
   public static void main (String[] arg) {
     //Skapa ett Scanner-objekt för inmatning:
     Scanner scan = new Scanner(System.in);

     String namn;
     System.out.print(Änge namn: ");
     namn = scan.nextLine(); //inmatning av ord eller meningar (strängar)

     System.out.println(Änge ålder: ");
     int ålder = scan.nextInt(); //inmatning av heltal

     System.out.println(Änge årsinkomst: ");
     double inkomst = scan.nextDouble(); //inmatning av decimaltal
   }
}
```

Annan viktig information

- Det är semikolon som skiljer sater i Java, inte radslut (enter).
- Man bör kommentera sin kod, men "lagom" mycket. //gör att resten av *raden* blir en kommentar som inte exekveras. Om man vill kommenreta flera rader kan man börja med /* och sluta med */.

- Indentera din kod MEDANS DU SKRIVER, inte efteråt!! Ett dåligt indenterat program kan vara omöjligt att felsöka!

  Exempel:

```
public class Hello {
   public static void main (String[] arg) {
      System.out.println(‘‘Hello World’’);
   }
}
```

  Samma exempel dåligt indenterat:

```
  public class
Hello
{      public
    static         void
main
(String[     ]  arg)
{     System.out.println(
  "Hello World"
)
;}
  }
```

- *Problem med blandad inläsning* av text och data, se exemplet `Class TestIO`