

Föreläsning 2

- Om kursen
- Enkla program i Java
- Sekvens
- Variabler
- Tilldelning, operatorer

Deitel: kapitel 2.4-2.7, 4.12 – 4.13

Om kursen

Under kursens gång läggs det ut material under rubriken **Kursmaterial**:

- **Föreläsningar**
En PowerPoint-presentationer med en hel del kodexempel. Kör kodexemplen på egen hand. Och ändra i koden och se vad som händer vid exekveringen. Det är viktigt att experimentera.
- **Laboration**
Kursen innehåller ett antal laborationer vilka du ska genomföra.
- **Programmeringsuppgifter**
Du ska redovisa **fem programmeringsuppgifter** under kursens gång. Diskutera gärna uppgiften med kursare och samarbeta med lösningen. Men det är mycket viktigt att du skriver din lösning på egen hand så du säkert förstår vad du gör.

På kursen är 2 skrivningstillfällen (prel):

Lördagen den 16/5 – 2009, kl 10.00 – 15.00

Lördagen den 22/8 – 2009, kl 10.00 – 15.00

Ett litet Java-program

```
package f2;
```

Welcome1.java

```
public class Welcome1 {  
    public void printMessage() {  
        System.out.println("Välkommen till ");  
        System.out.println("Java-kurs vid");  
        System.out.println("Malmö högskola!");  
    }  
}
```

Klassen Welcome1 innehåller metoden **printMessage**. En metod är ett antal programsatser som givits ett namn.

```
package f2;
```

StartWelcome1.java

```
public class StartWelcome1 {  
    public static void main(String[] args) {  
        Welcome1 w1 = new Welcome1();  
        w1.printMessage();  
    }  
}
```

Klassen StartWelcome1 innehåller metoden **main**. Ett java-program startar alltid sin exekvering i en main-metod.

Ett Java-program

Skriva ett program som skriver ut:

Mitt namn är Nina.

Jag är 13 år och väger 42 kilo.

```
public class Nina {  
  
}
```

Varje program ska bestå av minst två klasser (class). Deklarationen av en klass börjar med:

public class Klassnamn

Klassens namn ska börja med stor bokstav.

Denna klassen heter Nina.

Blockparenteserna, { och }, markerar start och slut på klassen.

Ett Java-program

Metoden presentation beskriver vad som ska uträttas av programmet.

```
public class Nina {  
    public void presentation() {  
  
    }  
}
```

Blockparenteserna markerar start och slut på presentation-metoden.

Ett Java-program

`System.out.println(...);` och
`System.out.print(...);`
skriver ut text i ett textfönster
(meddelandefönstret i JBX).
`println` gör att nästkommande
utskrift startar på ny rad.

```
public class Nina {  
    public void presentation() {  
        System.out.println("Mitt namn är Nina.");  
        System.out.println();  
        System.out.println("Jag är 13 år och ...");  
    }  
}
```

Nina.java

`System.out.println();`
ger en tomrad i textfönstret.

Metoden består av
tre satser med kod
(tre statement).
Varje sats avslutas
med semikolon ;.

Ett Java-program

Det behövs en klass som skapar ett objekt av klassen Nina och som sedan anropat presentation-metoden.

StartNina.java

```
package f2;
```

```
public class StartNina {  
    public static void main(String[] args) {  
        Nina nina = new Nina();    // Skapa objekt  
        nina.presentation();        // anropa presentation  
    }  
}
```

```
package f2;
```

```
public class Nina {  
    public void presentation() {  
        System.out.println("Mitt namn är Nina.");  
        System.out.println();  
        System.out.println("Jag är 13 år och ...");  
    }  
}
```

Ett Java-program

```
public class Nina {  
    public void presentation() {  
        System.out.println("Mitt namn är Nina.");  
        System.out.println();  
        System.out.println("Jag är 13 år och ...");  
    }  
}
```



Satserna med kod utförs **uppfifrån och ned**.

1. `Nina nina = new Nina();`
2. `nina.presentation();` // **Exekvera presentation**
3. `System.out.println("Mitt namn är Nina.");`
4. `System.out.println();` // **Tomrad**
5. `System.out.println("Jag är 13 år och ...");`

Mitt namn är Nina

Jag är 13 år och väger 42 kilo.

Vad händer om vi skriver raderna i presentation i annan ordning?

Källkod, bytekod och exekvering

- En kompilator skapar en fil med **bytekod** med hjälp av **källkoden**. Namnet på bytekodsfilen slutar med **.class**. Programmeraren utför kompileringen.



- **Bytekoden** exekveras på en dator med hjälp av en interpretator. Numera är det vanligt att interpretatorn översätter bytekoden till maskinkod. Interpretatorn finns i användarens dator.



- NetBeans / JBuilder sköter både kompilering och exekvering då du väljer **Run**. (**Build** ger enbart kompilering)

Kommentarer

Ett program ska innehålla **kommentarer** som beskriver vad det gör.
Kompilatorn bryr sig inte om kommentarer.

**/* Anrop till presentation-metoden ger ett par utskrifter om Nina.
Utskrifterna hamnar i ett textfönster. */**

```
public class Nina {  
    public void presentation() {  
        System.out.println("Mitt namn är Nina");  
        System.out.println();           // Tomrad  
        System.out.println("Jag är 13 år gammal ...");  
    }  
}
```

Flerradskommentar
/* */

Enradskommentar
//

**/* Detta program skapar ett objekt av typen Nina och anropar
presentation-metoden för att erhålla utskrifter. */**

```
public class StartNina {  
    public static void main(String[] args) {  
        Nina nina = new Nina();  
        nina.presentation();  
    }  
}
```

Inmatning av text

Med hjälp av klassen **JOptionPane** underlättas inläsning av text från tangentbordet. Även att visa en text i ett fönster underlättas. Nedanstående program läser in en text från tangentbordet och skriver sedan ut text i ett fönster.

```
package f2;
import javax.swing.JOptionPane; // eller import javax.swing.*;

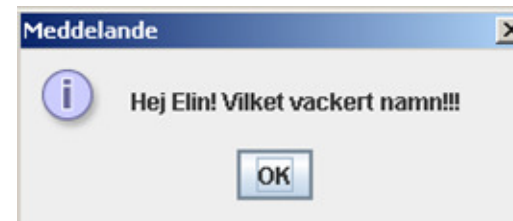
public class Inmatning {

    public void inmatningExempel() {
        String meddelande, namn; // String lagrar ett antal tecken

        namn = JOptionPane.showInputDialog( "Ange ditt namn" );
        meddelande = "Hej " + namn + "! Vilket vackert namn!!!";
        JOptionPane.showMessageDialog( null, meddelande );
    }
}
```

Inmatning.java

StartInmatning.java



Inmatning av tal

Metoden bmiMethod beräknar BMI för en person. I bmiMethod används bl.a. metoderna Integer.parseInt(String) och Double.parseDouble(String)

```
package f2;  
import javax.swing.*;
```

```
public class BMI {  
    public void bmiMethod() {  
        int vikt;  
        double längd,bmi;  
        String viktStr;
```

BMI.java

StartBMIEx.java

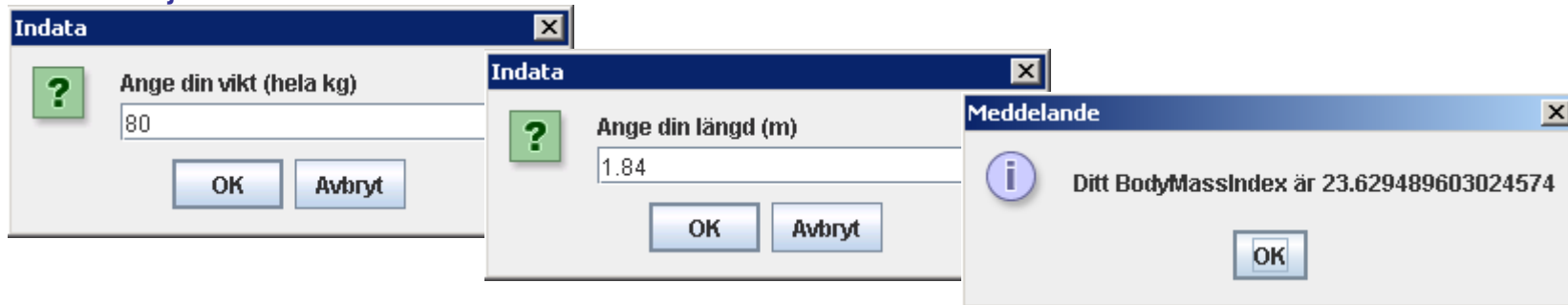
```
        viktStr = JOptionPane.showInputDialog( "Ange din vikt (hela kg)" );  
        vikt = Integer.parseInt( viktStr );  
        längd = Double.parseDouble( JOptionPane.showInputDialog( "Ange din längd (m)" ) );
```

```
        bmi = vikt/(längd*längd);
```

```
        JOptionPane.showMessageDialog( null, "Ditt BodyMassIndex är "+bmi );
```

```
    }
```

```
}
```



Hur källkoden ska skrivas

```
package f2;
import javax.swing.*;

public class Welcome {
    public void printMessage() {
        String name = JOptionPane.showInputDialog("Ange ditt namn");
        JOptionPane.showMessageDialog( null, "Hej " + name );
    } // printMessage
} // Welcome
```

- Startparentesen, {, ska stå på samma rad som (dvs efter) eller rakt under klassnamn, metodhuvud, styrstruktur etc.
- Efter startparentes (eller underförstådd startparentes) ska koden indenteras ett par steg (indenteras = flyttas åt höger).
- Slutparentesen, }, ska stå rakt under klassnamn, metodnamn, styrstruktur el.dyl.

Identifierare och reserverade ord

```
package f2;
import javax.swing.*;

public class Welcome {
    public void printMessage() {
        String name = JOptionPane.showInputDialog("Ange ditt namn");
        JOptionPane.showMessageDialog( null, "Hej " + name );
    } // printMessage
} // Welcome
```

- **Klassens namn** ska alltid börja med **stor bokstav** (Welcome)
- **Paketets namn** ska alltid börja med **liten bokstav** (javax.swing)
- **Metodnamn** ska alltid börja med **liten bokstav** (printMessage)
- **Variabelnamn** ska alltid börja med **liten bokstav** (name)

Ovanstående namn kallas för **identifierare**. En identifierare får bl.a. innehålla bokstäver, siffror och `_`. Identifierare ska vara beskrivande, t.ex. skatt och inte s.

Java innehåller ett antal **reserverade ord**, ord som har en speciell betydelse i språket. Exempel på sådana ord är `import`, `public`, `class` och `void`. Reserverade ord får inte användas som identifierare.

Datatyper och variabler

De data som ett program arbetar med kan vara av olika typ. De kan t.ex. utgöras av

- Heltal
- Flyttal (decimaltal)
- Text (kallas för strängar)

I programmet lagrar man data i s.k. **variabler**.

I ett program som ska räkna ut medelvärdet av tre heltal kan man ha en variabel för vart och ett av heltalen och en variabel för det beräknade medelvärdet. Programmet kommer alltså att innehåll minst fyra variabler.

Variabler

En variabel är en storhet som man själv inför i programmet. Variabeln kan lagra värde av en viss typ.

I Java finns det två sorters variabler, enkla variabler och referensvariabler:

- Enkla variabler – kan lagra någon typ av data, t.ex. heltal eller flyttal.
- Referensvariabler – refererar till någon typ av objekt. En referensvariabel kan alltså lagra en referens till någon typ av objekt. T.ex. hanteras strängar (text) av objekt av typen String. När du jobbar med en sträng så använder du en referensvariabel.

Du kommer att använda referensvariabler under kursens gång och du får veta mer om referensvariabler senare på kursen.

Enkla datatyper

Olika enkla variabeltyper

data	datatyp
heltal	byte, short, int , long
flyttal	float, double
tecken	char (kan lagra ett tecken, t.ex. 'a', '&', '9')
true/false	boolean

De färgade typerna är de som används på kursen.

Deklaration av variabel

typ **namn (= värde);**

int alder;

double langd,vikt=72.3;

En variabel har **typ**, **namn** och **värde**. Man måste alltid ange typ och namn. Om man vill kan man dessutom initiera variabeln med ett värde.

Lokala variabler

En variabel som deklareras i en metod kallas för en lokal variabel. En lokal variabel måste ges ett värde innan den kan användas i metoden.

Variabeln kan ges ett värde när den deklareras (kallas initiering, heltal, decimaltal, santFalskt initieras vid deklarationen) eller senare i programmet (mycketStortTal, tecken).

Variabler.java

```
public class Variabler {  
    public void variabler() {  
        int heltal=10000000;           //-2*10^9 - 2*10^9  
        long mycketStortTal;           //-9*10^18 - 9*10^18  
        double decimaltal=285.43 ;     //15 siffrors noggrannhet  
        char tecken;                   //Ett tecken  
        boolean santFalskt=true;       //true eller false  
        String mångaTecken="Hej!";    //Få eller många tecken  
  
        mycketStortTal=252332000000L;  //Går ej utan L  
        tecken='w';  
  
        System.out.println("int: "+heltal);  
        System.out.println("long: "+mycketStortTal);  
        System.out.println("double: "+decimaltal);  
        System.out.println("char: "+tecken);  
        System.out.println("boolean: "+santFalskt);  
        System.out.println("String: "+mångaTecken);  
    }  
}
```

Tilldelning

En **tilldelningssats** används för att ge en variabel ett nytt värde. Tilldelning anges med symbolen **=**.

```
int heltal;           // skapas utrymme för en int.  
double tal, andel;    // skapas utrymmer för två double.  
long stortTal, skatt=4300; // skapas utrymme för två long. skatt ges värdet 4300.  
heltal = 1250;        // heltal tilldelas värdet 1250. 1250 lagras i heltal.
```

variabel = uttryck;

```
tal = 3.25 + heltal;    // tal får värdet 1253.25.
```

Uttryck till höger om **=** beräknas först ($3.25 + \text{heltal}$). Sedan lagras värdet (1253.25) i variabeln till vänster om **=** (vänster led). Dvs. variabeln tilldelas värdet av uttrycket.

Eftersom resultatet av beräkningen i höger led är ett flyttal (1253.25) så måste tal vara av en flyttalstyp, t.ex. double.

En variabel kan lagra samma datatyp och ”mindre” datatyper

```
stortTal = skatt;      // skatt och stortTal är samma datatyp  
stortTal = heltal;     // heltal (int) är ”mindre” datatyp än stortTal (long)
```

Heltalstyper kan lagras i en flyttalstyp

```
tal = heltal;
```

Typkonvertering

När man vill ändra datatyp på ett värde så kallas det för typkonvertering.

Om man vill byta till en datatyp med större noggrannhet så sker detta automatiskt, s.k. **implicit typkonvertering**.

```
int tal1 = 23;
```

```
long tal2;
```

```
float tal3 = 34.6
```

```
double tal4;
```

```
tal2 = tal1;    // värdet i tal1 görs först om till ett long-värde. Detta värde lagras i tal2.
```

```
tal4 = tal3;    // värdet i tal3 görs först om till ett double-värde. Detta värde lagras i tal4.
```

```
tal3 = tal2;    // värdet i tal2 görs först om till ett float-värde. Detta värde lagras i tal3.
```

```
tal4 = tal1;    // värdet i tal1 görs först om till ett double-värde. Detta värde lagras i tal4
```

I exemplet ovan ser du att

- * mindre typ kan lagras i större typ (typ med större noggrannhet)
(**tal2 = tal1;** och **tal4 = tal3;**)
- * heltalstyp kan lagras i flyttalstyp
(**tal3 = tal2;** och **tal4 = tal1;**)

Typkonvertering

Om man vill byta till en datatyp med mindre noggrannhet så måste man ange detta i sin kod, s.k. **explicit typkonvertering**. Skälet till detta är att man riskerar att förlora i noggrannhet, t.ex. bli av med decimalerna i ett flyttal.

```
int tal1 = 23;  
long tal2;  
float tal3 = 34.6  
double tal4;
```

```
tal1 = (int)tal2;    // värdet i tal2 görs om till en int. Detta värde lagras sedan i tal1.  
                  // Är värdet i tal2 i intervallet  $-2 \cdot 10^9 - 2 \cdot 10^9$  så är resultatet  
                  // oförändrat annars är det förändrat (avhugget).  
tal3 = (float)tal4; // värdet i tal4 görs om till en float. Detta innebär att noggrannheten  
                  // i det nya värdet minskar till ca 7 siffror.  
tal2 = (long)tal3;  // tal3 görs om till en long. Detta innebär att decimalerna tas bort.  
tal1 = (int)tal4;   // tal4 görs om till en int. Detta innebär att decimalerna tas bort.
```

När ett flyttal typkonverteras till ett heltal så sker ej avrundning.
Decimalerna tas helt enkelt bort.

Aritmetik, räkneoperatorer

+ addition

- subtraktion

* multiplikation

/ division $45.0 / 10.0 = 4.5$
heltalsdivision $45 / 10 = 4$ (heltalsresultat)

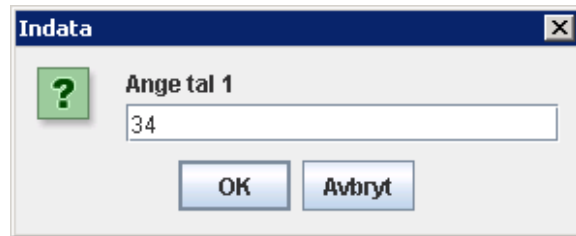
% rest vid heltalsdivision $25 \% 10 = 5$

$\frac{45}{10} = 4$ Rest: 5 $45 \% 10 = 5$

$\frac{7}{3} = 2$ Rest: 1 $7 \% 3 = 1$

Addera och subtrahera

Nu ska vi skriva ett program vars körresultat kan se ut så här:

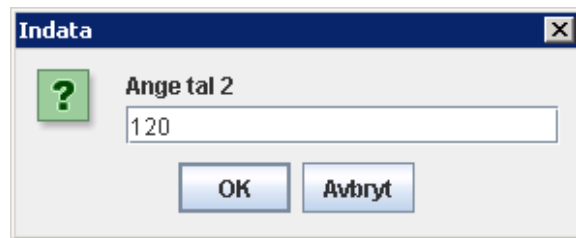


Indata

? Ange tal 1

34

OK Avbryt



Indata

? Ange tal 2

120

OK Avbryt

34+120=154

120+34=154

34-120=-86

120-34=86

Hur ska programmet se ut?

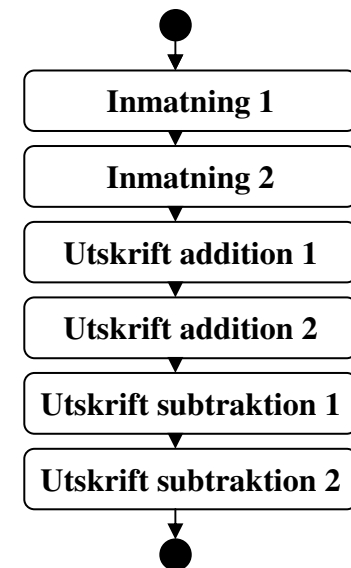
Addera och subtrahera - halvkod

Halvkod

- Deklarera variablerna tal1 och tal2
- Inmatning av tal 1
- Inmatning av tal 2
- Utskrift av addition 1
- Utskrift av addition 2
- Utskrift av subtraktion 1
- Utskrift av subtraktion 2

AddSub.java

StartAddSub.java



Prioritet

När uttryck beräknas så sker detta efter en speciell prioritet. Dessa regler följer reglerna inom matematiken.

Prioritetsordning vid beräkning av uttryck

1. Uttryck inom parenteser
2. Multiplikation, division och rest
3. Addition och subtraktion

Vid lika prioritet sker beräkningar från vänster till höger.

Exempel:

```
long resultat;  
int tal = 23;  
resultat = tal + 5 * (3+1);
```

1. **resultat = 23 + 5 * 4**
2. **resultat = 23 + 20**
3. **resultat = 43**

I **Liang** hittar du en exaktare lista över prioritet.

Vanliga räkneoperationer

Vardagsspråk

Öka variabeln **antal** med 1

Minska variabeln **antal** med 1

Öka variabeln **antal** med 4

Minska variabeln **antal** med 7

Lägg till värdet av **tal**

Minska med värdet av **tal**

Uttryck

```
antal = antal + 1;  
antal++;  
++antal;  
antal += 1;
```

```
antal = antal - 1;  
antal--;  
--antal;  
antal -= 1;
```

```
antal = antal + 4;  
antal += 4;
```

```
antal = antal - 7;  
antal -= 7;
```

```
antal = antal + tal;  
antal += tal;
```

```
antal = antal - tal;  
antal -= tal;
```

Vanliga räkneoperationer

Vardagsspråk

Uttryck

Lägg på moms

```
belopp = belopp + belopp*moms;  
belopp += belopp*moms;
```

Ta ut pengar

```
saldo = saldo - transaktion;  
saldo -= transaktion;
```

Dela 37 äpplen på 5 personer

Antal äpplen var

```
antal = applen / 5;
```

Ej utdelade äpplen

```
kvar = applen % 5;
```

Fördubbla

```
tal = 2 * tal;  
tal *= 2;
```

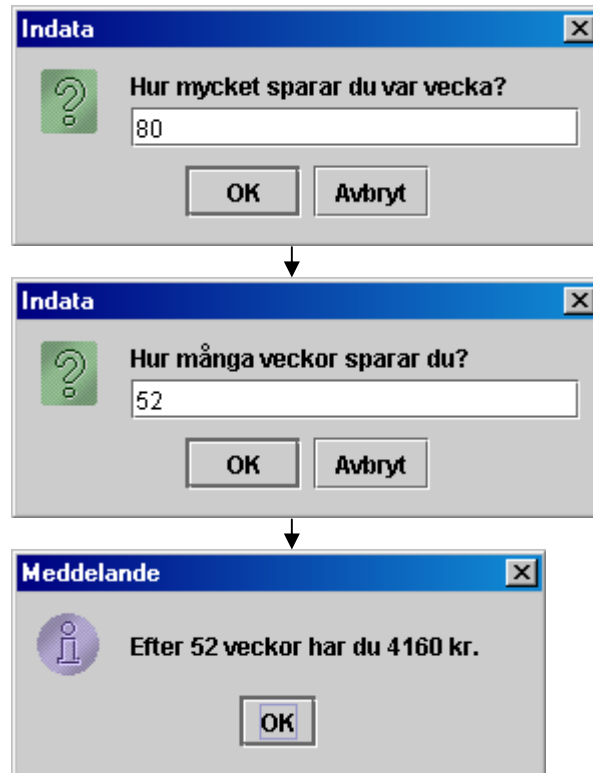
Halvera

```
tal = tal / 2;  
tal /= 2;
```

Byt tecken

```
tal = -tal;  
tal *= -1;
```

Körresultat



Hur ser programmet ut?

Sparande.java