

Föreläsning 4

- Ett javaprogram
- String
- Variabel
- Tilldelning, operatorer

Ett Java-program

Ett program består av en *sekvens* av instruktioner vilka en dator ska utföra (exekvera).

Instruktion 1

Instruktion 2

:

Instruktion n

Instruktionerna som ska exekveras i programmet ska placeras i en *klass*. Instruktionerna skrivs i en eller flera *metoder*.

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

Welcome1 är namnet på en *klass*. *todo* är namnet på en *metod*.

Programmet startar alltid i en speciell metod, *main*-metoden.

```
public static void main( String[] args ) {  
    Welcome1 prog = new Welcome1();  
    prog.todo();  
}
```

Klass

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

Welcome1 är namnet på en *klass*. En klass kan bl.a. innehålla en eller flera metoder. *Welcome1* innehåller metoden *todo*.

Namnet på en klass bestämmer du själv. Men följande regler ska du hålla dig till:

- * Namnet ska alltid börja med stor bokstav.
- * Namnet ska endast innehålla bokstäver och siffror
- * Bokstäverna ska hålla sig till a-z (A-Z). Använd t.ex. inte å, ä och ö i klassnamn.

Klassen ska till vidare börja med:

public class Klassnamn

Metod

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

todo är namnet på en *metod*. En metod innehåller instruktioner.

Metoden *todo* innehåller tre instruktioner:

1. `System.out.println("Välkommen till ");`
2. `System.out.println("java-kurs vid ");`
3. `System.out.println("Malmö högskola");`

INSTRUKTION
Utskrift i Output-fönstret
`System.out.println(String)`

Instruktionerna i en metod utförs alltid uppifrån och ner.

Namnet på metoden bestämmer du själv. Men följande regler ska du hålla dig till:

- * Namnet på metoden ska alltid börja med liten bokstav.
- * Namnet ska endast innehålla bokstäver och siffror.

I laboration 2 ska metoder börja med **public void**:
public void metodnamn()

Metoden main

Main-metoden ska innehålla instruktioner som startar programmet.

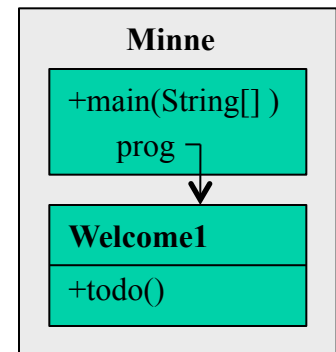
```
public static void main( String[] args ) {  
    Welcome1 prog = new Welcome1();  
    prog.todo();  
}
```

1. main-metoden ska skapa ett objekt av klassen där programmets instruktioner är placerade. Det sker med *new*-operatoren.

INSTRUKTION
Skapa objekt
new Klassnamn()

```
Welcome1 prog = new Welcome1();
```

prog är en *referensvariabel* och håller reda på objektet av *typen* Welcome1. Variabelns namn bestämmer du själv men den ska börja med liten bokstav.



2. main-metoden ska anropa metoden med instruktionerna som ska exekveras. Vid anropet flyttar exekveringen till instruktionerna i metoden.

INSTRUKTION
Anropa metod
variabel.metodnamn()

```
prog.todo();
```

Metoden main

main-metoden ska placeras i en klass. Du bestämmer själv vilken klass som ska innehålla *main*-metoden. En klass kan innehålla högst en *main*-metod.

Klassen *Welcome1* kan innehålla *main*-metoden:

```
public class Welcome1 {  
    public void todo() {  
        System.out.println("Välkommen till ");  
        System.out.println("java-kurs vid ");  
        System.out.println("Malmö högskola");  
    }  
  
    public static void main( String[] args ) {  
        Welcome1 prog = new Welcome1();  
        prog.todo();  
    }  
}
```

Welcome1.java

Metoden main

main-metoden kan placeras i en separat klass:

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

StartWelcome1.java

I det här fallet behövs ingen main-metod i klassen Welcome1.

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

Flera metoder i en klass

En klass kan innehålla flera metoder.

```
public class Person {  
    public void printName() {  
        System.out.println("My name is Lisa");  
    }  
  
    public void printInformation() {  
        System.out.println("I am 9 years old.");  
    }  
  
    public void printFamilyName() {  
        System.out.println("Bengtsson.");  
    }  
}
```

Person.java

Metoderna exekveras i den ordning de anropas.

```
public class StartPerson {  
    public static void main( String[] args ) {  
        Person person = new Person();  
        person.printName();  
        person.printFamilyName();  
    }  
}
```

**My name is Lisa
Bengtsson.**

StartPerson.java

```
public class StartPerson {  
    public static void main( String[] args ) {  
        Person person = new Person();  
        person.printFamilyName();  
        person.printInformation();  
        person.printName();  
        person.printInformation();  
    }  
}
```

**Bengtsson.
I am 9 years old.
My name is Lisa
I am 9 years old.**

Flera metoder i samma program

Metoder i en klass kan anropa varandra.

```
public class Person2 {  
    public void printName() {  
        System.out.println("My name is Lisa");  
    }  
  
    public void printInformation() {  
        System.out.println("I am 9 years old.");  
    }  
  
    public void printFamilyName() {  
        System.out.println("Bengtsson.");  
    }  
  
    public void presentation() {  
        printName();  
        printFamilyName();  
        printInformation();  
    }  
}
```

4
8
6
3
5
7

Person2.java

```
public class StartPerson2 {  
    public static void main( String[] args ) {  
        Person2 pers = new Person2();  
        pers.presentation();  
    }  
}
```

1
2

StartPerson2.java

Siffrorna till höger om instruktionerna anges exekveringsordningen.

The `println` method

- In the Lincoln program from Chapter 1, we invoked the `println` method to print a character string
- The `System.out` object represents a destination (the monitor) to which we can send output

`System.out.println ("Whatever you are, be a good one.");`

The diagram illustrates the components of the `println` method call. A red bracket under `System.out` is labeled **object**. A red arrow points to `println`, which is labeled **method name**. A red bracket under the string `"Whatever you are, be a good one."` is labeled **information provided to the method (parameters)**.

String Concatenation

- The *string concatenation operator* (+) is used to append one string to the end of another
`"Peanut butter " + "and jelly"`
- It can also be used to append a number to a string
`"Kaffe: " + 12 + " kr"`
- A string literal cannot be broken across two lines in a program
`System.out.println("Strängen kan delas upp" +
" i flera olika delar" +
" och på olika rader");`

CountDown.java

CountDown1.java

String Concatenation

- The + operator is also used for arithmetic addition
- The function that it performs depends on the type of the information on which it operates
- If both operands are strings, or if one is a string and one is a number, it performs string concatenation

```
"Rolf " + "Axelsson"
"Kaffe: " + 12
11 + " fotbollsspelare"
```
- If both operands are numeric, it adds them
- The + operator is evaluated left to right, but parentheses can be used to force the order

Addition.java

Escape sequences

<u>Escape Sequence</u>	<u>Meaning</u>
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\"</code>	double quote
<code>\'</code>	single quote
<code>\\</code>	Backslash (\)

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 innehålla 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. En referensvariabel som inte refererar till ett objekt har värdet **null**.

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.

Numeric Primitive Data

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

<u>Type</u>	<u>Storage</u>	<u>Min Value</u>	<u>Max Value</u>
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	$\pm 3.4 \times 10^{38}$ with 7 significant digits	
double	64 bits	$\pm 1.7 \times 10^{308}$ with 15 significant digits	

Lokal variabel

En variabel som deklareras i en metod kallas för en *lokal variabel*. En lokal variabel måste ges ett värde innan den kan användas. Variabeln kan tilldelas ett värde när den deklareras (initiering) eller senare i programmet.

```
public class LocalVariable {  
    public void example() {  
        int number=100000;  
        double real;  
        long bigNbr;  
  
        real = 13.25;  
        bigNbr = -37222654987L;  
  
        System.out.println( "int: " + number );  
        System.out.println( "double: " + real );  
        System.out.println( "long: " + bigNbr );  
    }  
  
    public static void main( String[] args ) {  
        LocalVariable lv = new LocalVariable();  
        lv.example();  
    }  
}
```

// Exempel på lokala variabler:
// number, real, bigNbr
// number initieras till 100000
// real har inget värde
// bigNbr har inget värde

// lv är en lokal variabel

LocalVariable.java

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;    // tilldelningssats  
tal = 3.25 + heltal;   // tal ges värdet 1253.25.
```

uttryck beräknas först (3.25 + heltal) till 1253.25.
Sedan lagras 1253.25 i variabeln **tal**.

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

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.

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

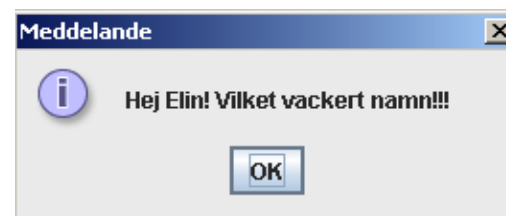
public class InputString {

    public void example() {
        String message, name; // String lagrar ett antal tecken

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

    public static void main(String[] args) {
        InputString prog = new InputString();
        prog.example();
    }
}
```

InputString.java



Inmatning av tal

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

```
import javax.swing.*;
```

```
public class BMI {  
    public void bmiMethod() {  
        int weight;  
        double length,bmi;  
        String str;
```

BMI.java

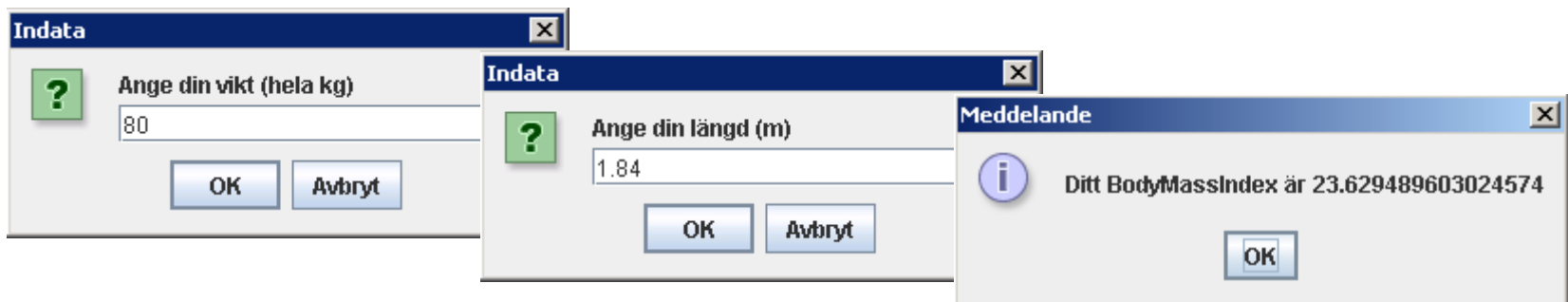
StartBMI.java

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

```
        bmi = weight/(length*length);
```

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

```
    }  
}
```



Aritmetik, räkneoperatorer

+ addition

- subtraktion

* multiplikation

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

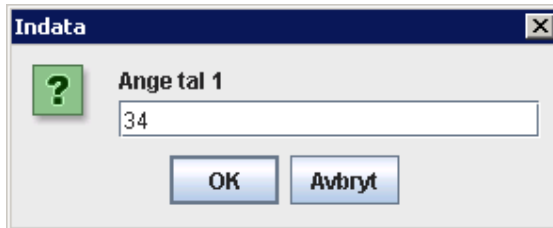
% rest vid heltalsdivision $25 \% 10 = 5$

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

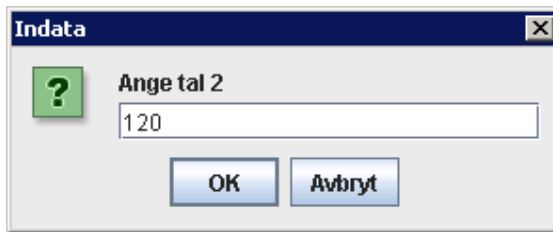
$\frac{7}{3} = 2$ Rest: 1 $7 \% 3 = 1$

Addera och subtrahera

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



A screenshot of a Windows-style dialog box titled "Indata". It contains a green square icon with a white question mark. To the right of the icon is the text "Ange tal 1". Below this text is a text input field containing the number "34". At the bottom of the dialog are two buttons: "OK" and "Avbryt".



A screenshot of a Windows-style dialog box titled "Indata". It contains a green square icon with a white question mark. To the right of the icon is the text "Ange tal 2". Below this text is a text input field containing the number "120". At the bottom of the dialog are two buttons: "OK" and "Avbryt".

34+120=154

120+34=154

34-120=-86

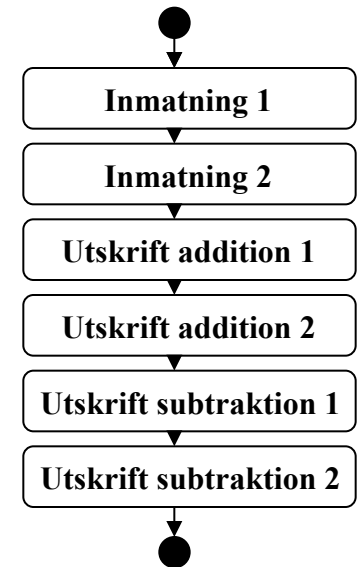
120-34=86

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

Hur källkoden ska skrivas

```
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.

```
public class Welcome {    eller    public class Welcome  
                                {
```

- 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.

```
public class Welcome {    eller    public class Welcome  
                                {  
                                }  
                                }
```

Identifierare och reserverade ord

```
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.