Objektorienterad Programmering (TDDC77)

Föreläsning II: utmatning, variabler, typer

Ahmed Rezine

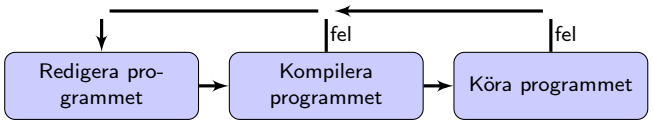
IDA, Linköpings Universitet

Hösttermin 2016

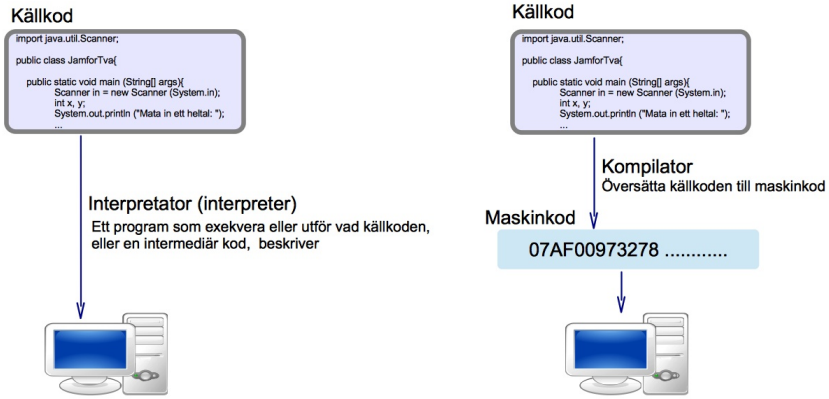
Kompilera och köra programmet

under terminal 2:

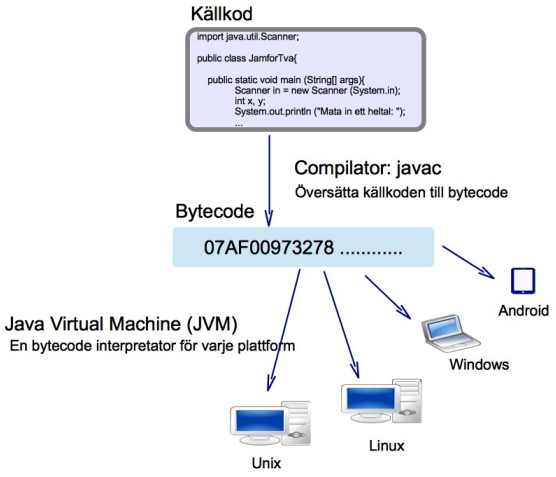
- Skapa Hej.java och skriv programmet
- Kompilera med **"javac Hej.java"**
- Rätta fel och repetera tills du lyckas kompilera ditt program
- Kör med **"java Hej"**
- Syntax och Semantiken



"Interpretera" eller "kompilera" källkod

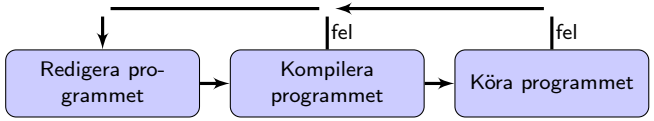


Kompilatorer och JVMer



Kompilera och köra programmet

- Skapa Hej.java och skriv programmet
- Kompilera med **"javac Hej.java"**
- Rätta fel och repetera tills du lyckas kompilera ditt program
- Kör med **"java Hej"**
- Syntax och Semantiken



Ge exakta instruktioner till datorn

- Exempel: hur många ord finns det i "programmering är kul" ? hur gjorde ni?
- Hur många ord i " cd-skivan , som ligger i laptop-väskan,är trasig !" ?
- Datorer måste få exakta "instruktioner" !!

Det är viktigt att förstå att en dator gör precis vad den är sagt att göra, och INGENTING ANNAT ! Därför, måste datorer få exakta "instruktioner" !!

Exakta instruktioner

- Analysera din uppgift och försök att hitta en lösning i ett språk du kan, t.ex. svenska !
- Översätt din lösning till programkod, t.ex. Java
- Försök inte att lösa allt på en gång
- Glöm inte att:
  - Använda papper och penna
  - Prova
  - Ha kul !

Outline

Ett enkelt program

```
/* Hej.java
 * Demonstrera hur man kan mata ut texter
 * till den standard outputen
 */
class Hej
{
    // Skriv ut ett enkelt meddelande
    public static void main(String[] args)
    {
     System.out.println("Hej");
     System.out.println("... IT1 studenter!");
    }
}
```

- Filen måste ha samma namn som programmetsnamn
- Kommentarer med `/* ... */` och `//...`
- Program definition med `class Namn { .... }`
- Main metoden är också definierad med `{...}`
- En metod är en gupp av instruktioner. Här två stycken.
- Varje instruktion slutar med en semikolon `;`

Ett enkelt program: Identifierare och reserverat ord

```
/* Hej.java
 * Demonstrera hur man kan mata ut texter
 * till den standard outputen
 */
class Hej
{
    // Skriv ut ett enkelt meddelande
    public static void main(String[] args)
    {
     System.out.println("Hej");
     System.out.println("... IT1 studenter!");
    }
}
```

`class`, `Hej`, `public`, `main`,`System`, `println` är identifierare

- Några har vi valt själva (`Hej`, `args`)
- Andra har andra programmerare valt (`String`, `out`, ...)
- Andra är reserverat i språket (`class`, `public`, `static` ...)
  `http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html`

Egna Identifierare

- Kombination av bokstäver, siffror, _, och $
- Kan inte börja med en siffra
- Exempel: `Hej`, `total`, `numberOfElements`, `$var`, `NUM_SEATS`
- Men inte: `2th`, `var#5`
- Skillnad mellan stora och små bokstäver: `sum` och `Sum` och `SUM` är inte samma identifierare

Egna Identifierare

- Det finns konventioner. Till exempel: stora bokstäver för varje ord: `Hej`, `MessagePrinter`, `AllElementsInspektor` i Program namn.
- Alltid välj meningsfulla namn. `x` eller `y` är sällan bra namn.
- Ni kommer att snabbt glömma vad ni menade med `smUpTi`.
- För långa namn är också värdelösa.

Blanktecken (white space)

- Java användar blanktecken för att separera ord
- Man kan formaterra sin text på olika sätt
- Man ska sträva efter att göra programmet så läsbart som möjligt

```
/* Demonstrera hur man kan mata ut texter
 * till den standard outputen
 */ class Hej{ // Skriv ut ett enkelt meddelande
public static void main(String[] args){ System.out.println("Hej");
System.out.println("... IT1 studenter!"); }}
```

```
/* Hej.java
 * Demonstrera hur man kan mata ut texter
 * till den standard outputen
 */
class Hej
{
    // Skriv ut ett enkelt meddelande
    public static void main(String[] args)
    {
     System.out.println("Hej");
     System.out.println("... IT1 studenter!");
    }
}
```


Outline

Utmatning: **println**

```
/* Klara.java
 *
 * Illustrera skillnaden mellan print och println
 */

class Klara{

    // Metoden skriver ut "klara färdiga gå!"
    public static void main(String[] args){
     System.out.println("Klara" + "... färdiga" + "... ... gå!!!");
    }

}
```


Utmatning: **print** och **println**

- Att skriva till skärmen, en fil, nätverk ... alltså data som kommer från programmet
- Brukar vara ganska lätthanterligt
- **print** och **println** kommer ni att använda ofta.

Att konkatenera sträng litteraler (string litterals)

```java
/* KlaraPlus.java
 *
 * Illustrera Strängskonkatenering
 */

public class KlaraPlus
{
    // Metoden konkatenera och skriver ut "klara färdiga gå!"
    public static void main(String[] args){
     System.out.print("Klara" + "... färdiga" + "... ... gå!!!");
    }
}
```


Outline

Variabler och datatyper

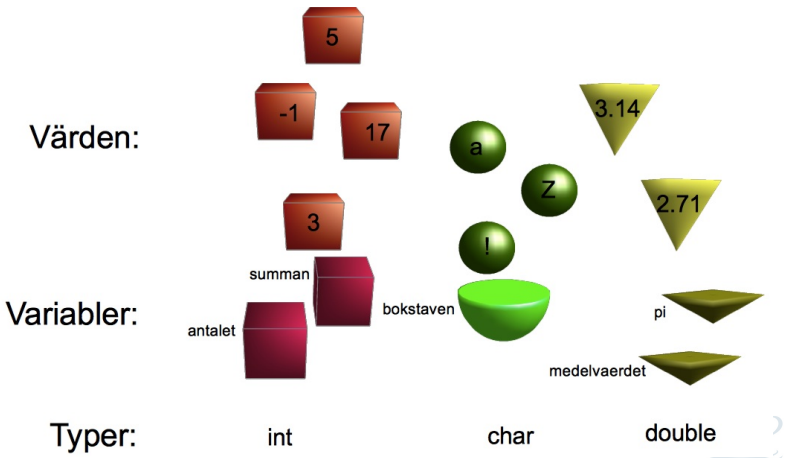
- En variabel är ett namn för en plats i dator minnet.
- Den används för att hålla ett datavärde
- Skilj på variabelns namn (plats i datorminnet), och värdet den representerar (värdet som står i platsen)

Utmatning: **println**

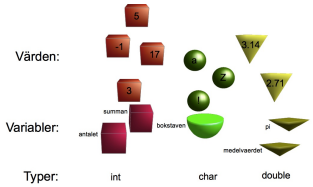
```java
/* StraengVar.java
 *
 * Illustrera konceptet av en sträng variabeln
 */
public class StraengVar{

    // Metoden deklarera och tilldela en sträng variabel och skriver ut
    public static void main(String[] args){
     String exampleNamn="TDDC77";
     String otherName = exampleNamn;
     System.out.println("Kursen heter: ");
     System.out.println(exampleNamn);
     System.out.println(otherName);
    }
}
```

Variabler och datatyper



Variabler och datatyper




```
public class DeklarationOchTilldelning
{
    public static void main(String[] args)
    {
     int talet=4, sum=0;
     double pi=3.14159265359, meddelvaerde;
     char bokstaven='!';
     String namnet;

     System.out.println("talet värde är: " + talet);
    }
}
```


Deklaration

- Variabler måste deklareras innan man får använda de
- Deklarationer associerar en typ (ex. int, float, String...) till ett namn (ex. talet, meddelvaerde, namnet ...)
- Namnet ska vara meningfult och i små bokstäver, t.ex: **namn, kursNamn, hourOfTheDay**
- Man använder **final** för att deklarera en konstant. Konstanternamnen skrivs i stora bokstäver, t.ex: PI, MAX_PLACES

Tilldelning

- Ett sätt att ge en variabel ett värde
- Man kan tilldela en litteral (".5", "3", "a", "hej!", etc), en variabel, eller ett uttryck ("1-4", "TDD" + "C77", etc) till en annan variabel
- variabeln och det man tilldela kan inte har vilken typ som helst

Datatyper

- Ett sätt att skilja på data av olika typ
- Både data och variabler har en typ som måste matcha
- Det finns åtta primitiva datatyper i Java:
  - heltal: byte, short, int, long; t.ex. $0, -128, 32767, \ldots$
  - flyttal: float, double; t.ex. $-3.4, 1.7$
  - boolean; t.ex. *false*
  - char; t.ex. 'a', 'Ö', ';'
- Alla andra datatyper i Java tillhör inte de primitiva datatyper och kallas för referens datatyper

Heltalstyper

Typ	Exempel	Storlek	Min	Max
byte	-5, 3, 127, ...	8 bits	-128	127
short		16 bits	-32768	32767
int*		32 bits	-2,147,483,648	2,147,483,647
long		64 bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

```
class Fib{

    static public void main(String[] args){

     int fib0=1;
     int fib1=1;
     int tmp;

     System.out.println("Första talet i Fibonaccis sekvensen är: " + fib0);
     System.out.println("Andra talet i Fibonaccis sekvensen är : " + fib1);

     tmp= fib0 + fib1;
     fib0= fib1;
     fib1= tmp;
     System.out.println("Nästa talet i Fibonaccis sekvensen är : " + fib1);

     tmp= fib0 + fib1;
     fib0= fib1;
     fib1= tmp;
     System.out.println("Nästa talet i Fibonaccis sekvensen är : " + fib1);
    }
}
```

Flyttalstyper

Typ	Exempel	Storlek	Min	Max
float	-1.5, 3.0, ...	32 bits	-3.4 E+38 (7 sig. siffror)	3.4 E+38 (7 sig. siffror)
double		64 bits	-1.7 E+308 (15 sig. siffror)	1.7 E+308 (15 sig. siffror)

- **double** är den standard typen.
- Man läggar **F** eller **f** för att mena ett **float** istället för ett **double**.
- Det som skiljer är framför allt precisionen. Kommer att förklaras närmare i kursen numeriska beräkningar.

```
class Pi{
    static public void main(String[] args){
     double pi1 = 3.14159265;
     float pi2 = 3.14159265F;

     System.out.println("Pi i en double: " + pi1);
     System.out.println("Pi i en float : " + pi2);
    }
}
```


Char typen

- Variabler av typen char tar 16 bits i minnet och kan innehålla en av 65536 tecken.

```
class Char{
    static public void main(String[] args){
     char t1='H', t2='e', t3='j', t4=' ';
     char t5='T', t6='D', t7='D', t8='C';
     char t9='7', t10='7', t11='!';

     System.out.println("Sammanbindningen ger: " + t1 + t2 + t3 + t4 + t5 + t6
                 + t7 + t8 + t9 + t10 + t11);

    }
}
```

Boolean typen

- Variabler av typen boolean kan anta värdena **true** (dvs. sant) och **false** (dvs. falskt)

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

     boolean klar= false;
     boolean foerStor, aerMindre;

     System.out.println("Är vi klara ? " + klar);
    }
}
```


Primitiva datatyper

Det finns 8 primitiva datatyper i Java:

Typ	Exempel	Storlek	Min	Max
byte	-5, 3, 127, ...	8 bits	-128	127
short		16 bits	-32768	32767
int*		32 bits	-2,147,483,648	2,147,483,647
long		64 bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
float	-1.5, 3.0, ...	32 bits	-3.4 E+38 (7 sig. siffror)	3.4 E+38 (7 sig. siffror)
double		64 bits	-1.7 E+308 (15 sig. siffror)	1.7 E+308 (15 sig. siffror)

Variabler av typen char tar 16 bits i minnet och kan innehålla en av 65536 tecken.

Variabler av typen boolean kan anta värdena **true** (dvs. sant) och **false** (dvs. falskt)

Outline

Typkonvertering

- typkonvertering byter typ på ett värde (inte alltid exakt!!)
- den kan ta en av tre former:
  - tilldelning: sker automatiskt, t.ex. tilldela ett byte värde till en int variabel
  - promotion (befordran?): sker också automatiskt i vissa operationer, t.ex när man dividera en float per en int, eller när man konkatenera en sträng och en int0
  - casting: explicit genom att skriva önskade type i parenthese innan värden som ska konverteras. Om en konvertering är alls möjlig, casting borde klara det.

Implicit typkonvertering

```
class ImplicitKonvertering{
    static public void main(String[] args){
     byte b = 77;
     int i = b; //123456789;
     float f = i;
     double d = f;

     System.out.println("b= " + b);
     System.out.println("d= " + d);
    }
}
```


Explicit typkonvertering

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

     int i1 = 77; //123456789;
     //char c1 = i1; //error
     char c2 = (char) i1;
     System.out.println("i1=  " + i1);
     System.out.println("c2= " + (int)c2);
     System.out.println();

     //int i2 = 1.5; // error
     int i3 = (int) -2.75;
     System.out.println("i3 = " + i3);
     System.out.println();

     //float f1 = 0.5; //error
     double f2 = 0.5;
     float f3 = 0.5F; //ingen typekonvertering !
     float f4 = (float) 0.5;
     System.out.println("f2= " + f2);
     System.out.println("f3= " + f3);
     System.out.println("f4= " + f4);
    }
}
```