



Programmering för språkteknologer I

Markus Saers
markus.saers@lingfil.uu.se
9-2040
stp.ling.uu.se/~markuss/vt09/pst1



Innehåll

Kapitel 5, 6, 7
Repetition
Datatyper
Uttryck – metoder för primitiva typer
Kontrollsatser



Repetition

Variabeltyper
Klasser
 Implementation
 Struktur
Metodanrop
Statiska metoder och variabler



Variabeltyper

Objektsreferenser
 Utgående pil
 ...till potentiellt komplicerad struktur
Värdetyper
 Enkla värden
 Heltal – `int`
 Decimaltal – `double`
Gör stor skillnad vid tilldelning



Klasser

Specifikation → Implementation
Ritning för hur objekt ska vara
 Lexikal definition
Metoder
 Vad kan man göra med objekt av klassen?
Attribut
 Hur representeras objektens tillstånd?



Metodanrop

Hur fungerar de "på riktigt"?
Vilka variabler finns tillgängliga vid ett anrop?
Vad är skillnaden mellan att skicka med objekt och värden som argument?
Metodsignatur



Statiska metoder och variabler

Finns utan konkreta objekt
Tillhör klassen

```
public static void main(String[] args)
```



Datatyper

Samma som variabeltyper
Fokuserar på det data som lagras i variabeln

Två grupper

Enkla/primitiva/värden
Komplexa/strukturer/objekt

Datorer kan bara hantera enkla datatyper

All komplexa datatyper kan reduceras till enkla



Enkla datatyper

Tal av olika slag

Heltal
Decimaltal
Index i teckentabeller
Sant/falskt

Olika mycket minne används för representationen

Kan reduceras till en serie ettor och nollor (bitar)



Javas enkla datatyper

Heltal

byte (8 bitar): -128–127
short (16 bitar): -32 768–32 767
int (32 bitar): -2 147 483 648–2 147 483 647
long (64 bitar): -9,2e18–9,2e18
-9 223 372 036 854 775 808–9 223 372 036 854 775 807

Decimaltal

float (32 bitar): -3,4e38–3,4e38 (7 siffrors noggrannhet)
double (64 bitar): -1,7e308–1,7e308 (15 siffrors noggrannhet)

Index i teckentabeller

char (16 bitar): 0–65 535

Sant/falskt

boolean (1 bit): true/false



Komplexa datatyper

Byggs upp av enkla datatyper och andra komplexa datatyper

Klasser

```
public class Square {  
    private int x;  
    private int y;  
    private int side;  
    // Konstruktörer och metoder  
}
```



Allt är ettor och nollor!

Alla program är uppbyggda av komplexa datatyper

Alla komplex datatyper kan reduceras till enkla datatyper

Alla enkla datatyper kan representeras med en serie ettor och nollor (bitar)

En bit har antingen värden 1 eller 0

1/0, på/av, sant/falskt



Uttryck

Specialsatser

Tilldelning

Har vi redan gått igenom

Aritmetiska uttryck

Logiska uttryck

Bitaritmetiska uttryck

Hoppar vi över

new

Har vi redan gått igenom



Aritmetiska uttryck

Skrivs som man förväntar sig

De fyra räknesätten: + - * /

Gruppering: ()

Negation: -

"Resten": %

Viktigt att hålla reda på vilka datatyper som är inblandade

Vid heltalsdivision kastas resten (det som man beräknar decimalsiffrorna med) bort



Aritmetiska uttryck

```
int a = 0;
```

```
int b = 12;
```

```
int c = 20;
```

```
a = 2 * (b + c) + 4; // a = 68
```

```
b = a / 10; // b = 6
```

```
c = a % 10; // c = 8
```



Aritmetiska uttryck

```
double x = 1.4;
```

```
double y = 2 * x; // y = 2.8
```

```
double z = x / 2; // z = 0.7
```

```
int a = 25;
```

```
double b = a / 2; // b = 12
```



Aritmetiska uttryck

```
double x = 1.4;
```

```
double y = 2 * x; // y = 2.8
```

```
double z = x / 2; // z = 0.7
```

```
int a = 25;
```

```
double b = (double)a / 2;
```

```
// Explicit konvertering
```

```
// av a från int till double
```

```
// b = 12.5
```



Aritmetiska uttryck

Mer matte?

Kolla i dokumentationen för standard klassen `Math` (i `java.lang`)

Massor med bra statiska metoder för att beräkna mer komplexa saker (upphöjt till, kvadratroten ur, sinus, cosinus, logaritmer)

Tilldelningsversioner finns också:

```
a += 1; // a = a + 1
```

```
b -= 2; // b = b - 2
```

```
c *= 3; // c = c * 3
```

```
d /= 4; // d = d / 4
```



Logiska uttryck

Boolesk algebra
Kombinerar värden av typen `boolean`

Jämförelseoperatorer
Jämför två värden



Boolesk algebra

Konjunktion (och, \wedge)
`&&`

Disjunktion (eller, \vee)
`||`

Negation (icke, \neg)
`!`

Konstanterna sant och falskt
`true`
`false`



Jämförelseoperatorer

Jämför två enkla typer

Mindre än ($<$) ...eller lika med (\leq)
`<` `<=`

Större än ($>$) ...eller lika med (\geq)
`>` `>=`

Lika med ($=$)
`==`

Skiljt från (icke lika med, \neq)
`!=`



Logiska operatorer

Låter oss analysera enkla typer



Kontrollsatser

Kontrollerar programflödet
Vad är programflödet?



Programflödet

Den ordning saker sker i
Går att utläsa ur programmet
Kan vara bökigt med komplexa program
Kan göra det svårt att felsöka

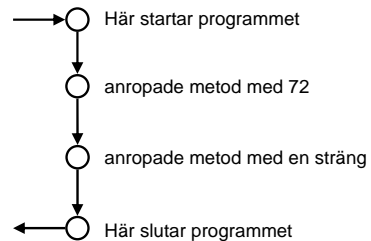


Exempel

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Här startar programmet");
        aMethod(72);
        anotherMethod("en sträng");
        System.out.println("Här slutar programmet");
    }
    public static void aMethod(int a) {
        System.out.println("anropade metod med " + a);
    }
    public static void anotherMethod(String s) {
        System.out.println("anropade metod med " + s);
    }
}
```



Exempel



Programflöde

Vad saknas?

Alternativa vägar

Hur gör man för att få utskrifter av typen "72 är ett jämt tal"?

Repetitioner

Hur gör man för att förklara exempelvis "gör det här för alla ord i SUC-korpusen"?

Man kontrollerar programflödet!

Med kontrollsatser



Alternativa vägar

Om-så-satser, villkorssatser

Motsvarar implikation i satslogik (\rightarrow)

Om x är jämt så skriv "x är ett jämt tal"

```
if (isEven(x)) {
    System.out.println(x +
        " är ett jämt tal");
}
```



Villkorssatser

Kan byggas ut till om-så-annars

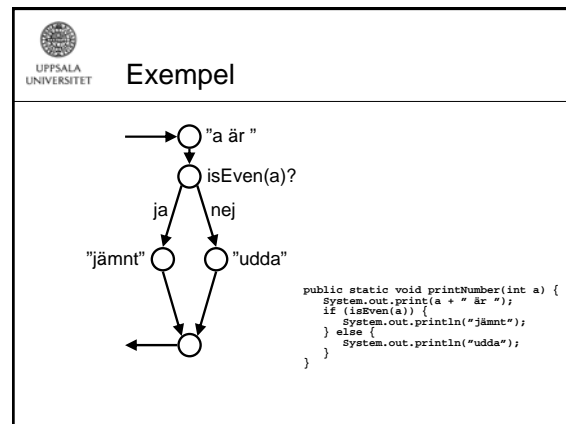
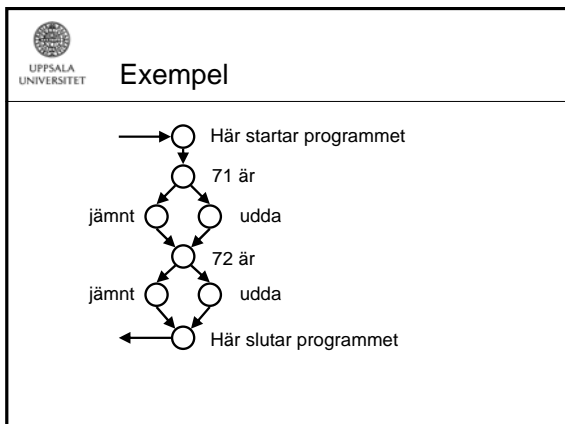
Om x är jämt så skriv "x är ett jämt tal", annars skriv "x är ett udda tal"

```
if (isEven(x)) {
    System.out.println(x +
        " är ett jämt tal");
} else {
    System.out.println(x +
        " är ett udda tal");
}
```



Exempel

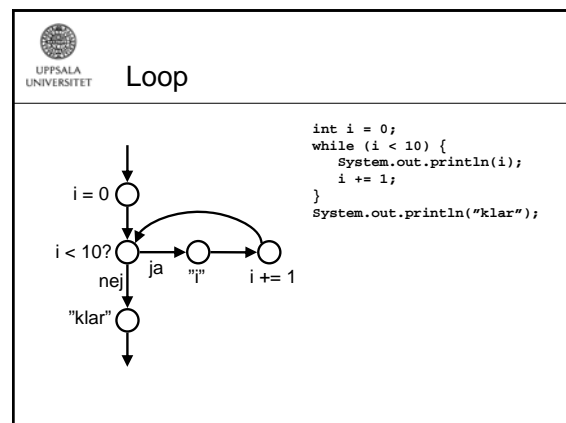
```
public class TestIf {
    public static void main(String[] args) {
        System.out.println("Här startar programmet");
        printNumber(71);
        printNumber(72);
        System.out.println("Här slutar programmet");
    }
    public static void printNumber(int a) {
        System.out.print(a + " är ");
        if (isEven(a)) {
            System.out.println("jämnt");
        } else {
            System.out.println("udda");
        }
    }
    public static boolean isEven(int a) {
        // ger true om a är jämt, annars false
    }
}
```



UPPSALA UNIVERSITET

Repetitioner

Låter oss upprepa ett kodblock
 Måste ha ett villkor
 När det inte är uppfyllt avbryts repetitionen
 Kallas också loop
 "Medan villkoret är uppfyllt: utför följande block"



UPPSALA UNIVERSITET

Loop

Vanligaste loopen

```

int i = 0;
while (i < 10) {
    // Gör något
    i++;
}
  
```

UPPSALA UNIVERSITET

Loop

Vanligaste loopen
 Initiering

```

int i = 0;
while (i < 10) {
    // Gör något
    i++;
}
  
```



Loop

Vanligaste loopen
Initiering
Villkor

```
int i = 0;
while (i < 10) {
    // Gör något
    i++;
}
```



Loop

Vanligaste loopen
Initiering
Villkor
Steg

```
int i = 0;
while (i < 10) {
    // Gör något
    i++;
}
```



For-loop

Innehåller alla loop-komponenter i blockhuvudet

```
int i = 0;
while (i < 10) {
    // Gör något
    i++;
}

for (int i = 0; i < 10; i++) {
    // Gör något
}
```



Exempel

```
public class TestWhileIf {
    public static void main(String[] args) {
        int i = 0;
        while (i <= 10) {
            System.out.print(i + " är ");
            if (isEven(i)) {
                System.out.println("jämnt");
            } else {
                System.out.println("udda");
            }
            i++;
        }
    }
    public static boolean isEven(int a) {
        return a % 2 == 0;
    }
}
```



Sammanfattning

Kapitel 5, 6, 7
Datatyper
Aritmetiska och Booleska uttryck
Kontrollsatser
if-else
while
for