Datavetenskap I

Föreläsning 7

Booleska uttryck och sanningstabeller

A few good men

Regissör: Rob Reiner

Screen play: Aron Sorkin

Skådespelare: Tom Cruise, Jack Nicholson

Föreläsning 7

- Innehåll
  - Föreläsning 7
    - Booleska uttryck
    - Sanningstabeller
  - Läsanvisningar

Varför booleska uttryck?

- Programkod har ett flöde:
  - Main-metod
    - Första satsen i main-metoden
    - Andra satsen i main-metoden
    - Och så vidare
- Vi har kanske ett metod-anrop till en annan metod. Då hoppar vi dit och utför de satserna i ordning. Sedan hoppar vi tillbaka i koden till satsen efter metod-anropet.

Varför booleska uttryck?

- Men om vi har en bit kod vi vill exekvera ibland och ibland inte? Eller en bit kod vi vill upprepa flera gånger?
- Vi behöver booleska uttryck (villkor) som styr
  - selektion
  - iteration
- Vi styr flödet i koden med dessa.
- Det är användandet av selektioner och iterationer som gör programkod verkligt användbar.

Den enklaste formen av selektion är if-satsen. Vi frågar oss om ett villkor är sant och om det är det så utför vi ett block kod som hör till if-satsen. Strukturen följer mönstret:

OM (villkor som ska utvärderas är sant)
    utför det här kod-blocket

Mer om selektioner och iterationer i problemlösning och kommande föreläsningar.

Booleska uttryck

- Ett boolesk uttryck kan utvärderas till:
  - sant
  - falskt
- Vi använder olika operatorer för att bygga upp uttryck som ska utvärderas.
- Uttryck behöver också operander som innehåller data.
- Ett booleskt uttryck innehåller minst en operand och en operator. Oftast så behöver vi två operander.

Exempel:

```
int count = 30;
if (count > 20)
    System.out.println (”Count exceeded”);
```

I fallet ovan så är varibeln count större än 20 och villkoret utvärderas till sant. Raden med println kommer att exekveras.

```
int count = 10;
if (count > 20)
    System.out.println (”Count exceeded”);
```

I fallet ovan så är varibeln count mindre än 20 och villkoret utvärderas till falskt. Raden med println kommer inte att exekveras utan vi hoppar till nästa rad efter if-satsen (syns inte här).

Variabeln count:s värde är antingen större än 20 eller inte. Det finns inget mitt emellan.

I uttrycket count>20 så är count och 20 operander och > operator

Booleska uttryck

- Operander i programkod utgörs av variabler som innehåller värden eller värden skrivna direkt i uttrycket.
- Operatorer kan vara av olika sorter:
  - Likhetsoperatorer
  - Relativa operatorer
  - Logiska operatorer
- Vi använder också parenteser för att gruppera delar i mer komplexa uttryck.

```
int count = 30;
if (count > 20)
    System.out.println (”Count exceeded”);
```

I uttrycken ovan är variablen count och värdet 20 operander. Operatorn utgörs av större än-tecknet.

Värden kan också hämtas via metod-anrop. Detta är något som kommer att komma efterhand som ni programmerar och lär er skriva kod i lite kortare former.

Parenteser fungerar precis som för matematiska uttryck. De talar om vilka delar som hör ihop och ska beräknas/utvärderas först.

Likhetsoperatorer

- Bland likhetsoperatorerna hittar vi:
  - == som jämför om två operander är exakt lika
  - != som jämför om två operander är olika
- Observera att operatorn för lika med är två likhetstecken i följd. Ett likhetstecken innebär en tilldelning.

```
int total = 20;
int sum = 20;

if (total == sum)
    System.out.println(”total equals sum”);
```

I fallet ovan så kommer textsträngen att skrivas ut. Om vi istället har:

```
int total = 10;
int sum = 20;

if (total == sum)
    System.out.println(”total equals sum”);
```

Så kommer textsträngen inte att skrivas ut.

Om vi ändrar operatorn från lika (==) till olika (!=) så blir det tvärt om:

```
int total = 20;
int sum = 20;

if (total != sum)
    System.out.println(”total not equal to sum”);
```

I fallet ovan så kommer textsträngen inte att skrivas ut. Om vi istället har:

```
int total = 10;
int sum = 20;

if (total != sum)
    System.out.println(”total not equal to sum”);
```

Så kommer textsträngen att skrivas ut.

Relativa operatorer

- Vi har flera relativa operatorer:
  - < mindre än
  - <= mindre än eller lika med
  - > större än
  - >= större än eller lika med

Ordningsföljd på operatorer

- Aritmetiska operatorer (+, -, /, …) utvärderas före likhet och relativa operatorer.
- 2+4<6+7 blir (2+4)<(6+7) blir 6<13
- Var tydlig i dina uttryck och sätt ut parenteser även om de inte är nödvändiga för programkoden.
- **Lättläst kod minskar risken för fel.**

Logiska operatorer

- Vi har operatorer som vi använder för att binda samman olika delar i ett booleskt uttryck:
  - ! Logiskt ICKE (unär operator)
  - && logiskt OCH
  - || logiskt ELLER (det ena eller det andra eller bägge)
- Dessa operatorer fungerar med booleska operander. Det betyder att operanden måste gå att utvärdera till att vara sann eller falsk.
- Om den första operanden i ett uttryck med logiska operatorer kan ange vad uttrycket som helhet kommer att ha för värde så beräknas inte uttrycket för nästa operand i Java. Detta gäller dock inte alla programspråk.

! Är en unär operator vilket innebär att den endast används med en operand. Du placerar ! Framför den operand du vill få motsatsen till.

De logiska operatorerna här förändrar inte variablerna eller värdena som används med dem. De påverkar endast hur vi utvärderar uttrycket som helhet. ICKE beräknas före OCH och sist ELLER

```
boolean isWindy = true;
boolean itRains = false;

if (isWindy && itRains)
    System.out.println(”Use rain coat”); //skrivs inte ut
med de nuvarande värdena på isWindy och itRains

if (!isWindy && itRains)
    System.out.println(”Use umbrella”); //skrivs inte ut
med de nuvarande värdena på isWindy och itRains

if (!itRains)
    System.out.println(”no rain protection needed”);
//skrivs ut med de nuvarande värdena på isWindy och
itRains
```

Sanningstabeller

- Sanningstabeller är ett verktyg för att skriva upp och titta på olika delar i ett booleska uttryck med logiska operatorer.
- Den enklaste sanningstabellen vi kan göra:

a	!a
S	F
F	S

S står för sant
F står för falskt

Sanningstabeller blir större ju fler operander som är inblandade. Du skapar en sanningstabell genom att först skriva upp alla operander och göra de möjliga kombinationerna av dessa: Två operander ger 4 rader, 3 operander 8 rader, 4 operander 16 rader, o.s.v.

Sanningstabeller

- Om vi har uttrycket
  (a&&b) || c
  som vi vill utvärdera hur gör vi?
- Först skriver vi upp alla kombinationer av värden på operanderna:

a	b	c
S	S	S
S	S	F
S	F	S
S	F	F
F	S	S
F	S	F
F	F	S
F	F	F

Sanningstabeller

- (a&&b) || c : Nästa steg så fyller vi på med den första delen av uttrycket som ska utvärderas:

a	b	c	a&&b
S	S	S	S
S	S	F	S
S	F	S	F
S	F	F	F
F	S	S	F
F	S	F	F
F	F	S	F
F	F	F	F

Sanningstabeller

- (a&&b) || c : I sista steget får vi nu hela uttrycket:

a	b	c	a&&b	(a&&b)|| c
S	S	S	S	S
S	S	F	S	S
S	F	S	F	S
S	F	F	F	F
F	S	S	F	S
F	S	F	F	F
F	F	S	F	S
F	F	F	F	F

Läsanvisningar

- Java Foundations
  - Kapitel 4.1: Viktigt. Du behöver kunna behärska detta senare. Du ska kunna förstå och använda olika operatorer för att skapa booleska uttryck och kunna utvärdera booleska uttryck. Du ska kunna läsa och konstruera sanningstabeller.
- Extra materiel
  - Till power point-bilderna finns kommentarer som utökar informationen i JF 4.1. Dessa ska läsas som en del av kurslitteraturen.

Som övning rekommenderas att du själv gör sanningstabeller för exemplen i boken.