Programmering 1 - Uppgift 3

Peter Holm

Juli 2020

1 Uppgifter

1.1 Ge tre olika exempel på logiska uttryck.

När man skall välja mellan olika vägar i ett program använder man för det mesta en if-sats. Denna sats kan se ut på olika sätt. Den enklaste versionen har formen:

```
if (logiskt uttryck)
    sats;
```

Exempel på logiska uttryck:

```
// Exempel 1
if (liquidLevel >= maxLevel)
    openValve();

// Exempel 2
if (liquidLevel < minLevel) {
    closeValve();
    fillReservoir();
}
// Exempel 3
if (waterTemp > thermostatValue || waterTemp < thermostatValue)
    adjustWaterTemp();
```

1.2 Beskriv den booleska datatypen och skriv ett exempel (källkod) där den används.

Variabler kan även deklareras till datatypen **boolean**. Det här en typ som används för att beskriva sanningsvärden.
En **boolean** kan enbart innehålla två olika värden, dessa värden är **true** eller **false**.
I programmering är det vanligt att man behöver en typ som kan innehålla en av två värden, som:

- Ja / Nej
- På / Av
- Sant / Falskt

```
// Exempel
boolean isJavaFun = true; // Boolean initiering

if (isJavaFun)
    System.out.println("Java is fun!"); // Utmatning
else
    System.out.println("Java is not fun..");
```

1.3 Konvertera följande while-sats till en for-sats, förklara koden i detalj och beskriv vad blir utskriften blir efter körning (efter lämplig komplettering av koden).

```
int k = 0; // Heltalsvariabeln k, initieringsdel.
String s = "k:"; // Textvariabeln s
while(k < 6) { // Logiskt uttryck, villkorsdel.
    s = s + " " + k; // sats1
    k = k + 2; // sats2, forandringsdel.

    System.out.println(s); // Utmatning i kommandotolken
}
```

- Heltalsvariabeln **k** sätts som räknare till **while**-satsens logiska uttryck. **while**-satsens uppgift är att repetera en eller flera programsatser, för var gång **while**-satsens logiska uttryck är sant, utförs satserna innanför **while**-satsens klamrar. I det här fallet utförs **while**-satsen 3 gånger tills det logiska uttrycket är falskt och inga fler varv i **while** utförs.
- Sats1 innanför klamrarna tilldelar strängvariabeln **s**, strängen, "`k:  <whitespace>`" samt värdet i "`k`" för varje varv **while**-satsen utförs.
- Sats2 utökar värdet genom att addera värdet 2 till heltalsvariabeln **k** för varje varv i **while**-satsen.
- Genom att placera en **println**-metod innuti **while** kan vi se hur det hela successivt byggs upp genom repetitionens gång, via kommandotolken. Efter exekvering får vi utmatningen;

```
k:  0
k:  0 2
k:  0 2 4
```

Ändrar vi det logiska uttrycket till **while**(k $<$ 12), får vi utmatningen;

```
k:  0
k:  0 2
k:  0 2 4
k:  0 2 4 6
k:  0 2 4 6 8
k:  0 2 4 6 8 10
```

Utförs istället **while** 6 gånger.

Nedan följer **while**-satsen tidigare, omskriven till en **for**-loop.

```
String s = "k:";

for (int k = 0; k < 12; k = k + 2) {
    s = s + " " + k;
    System.out.println(s);
}
// Utmatning:
// k: 0
// k: 0 2
// k: 0 2 4
// k: 0 2 4 6
// k: 0 2 4 6 8
// k: 0 2 4 6 8 10
```

Skillnaden mellan en **while** och en **for**-loop är att inuti **for**-loopens parenteser sker en initieringsdel en vilkorsdel(logisk uttryck) samt en förändringsdel.

- **while**-loopens initieringsdel sker innan repetitions-satsen.
- **while**-loopens villkorsdel sker efter **while**-deklarationen, inuti dess parenteser. Likt **for**.
- **while**-loopens förändringsdel sker inuti dess klamer-kropp.

1.4 Förklara skillnaden mellan jämförelseoperatorer och logiska operatorer med hjälp av några exempel.

Används jämförelseoperatorer(<, >, != etc) så bildas uttryck av data-typen boolean.

```
int a = 5, b = 10;
/*Lika med*/
a == b; // falskt
/*Icke lika med */
a != b // sant
/* less than */
a < b // sant
/* Greater than */
a > b // falskt
```

Det finns även tre andra operatorer som används till att bilda mer advancerade logiska uttryck, dessa heter logiska operatorer.

```
// && och-operator, || eller-operator, ! icke-operator.
int a = 5, b = 10;


a == 5 || b == 20; // sant
a == 20 || b == 5 // falskt

b > a && a < b // sant
b < a && a > b // falskt

!(a == 0 || a == 10) // sant
!(a == 5 || b == 10) // falskt
```

- && Ger värdet sant, ifall samtliga uttryck är sanna.
- || Ger värdet sant, ifall ett av uttrycken är sanna.
- ! Ger värdet sant, ifall inget utav uttrycken stämmer.

1.5 Förklara vad ökningsoperatorn (++) och minskningsoperatorn (- -) gör med hjälp av några exempel (källkod). Dessa operatorer finns i två varianter, prefix och postfix. Förklara också skillnaden mellan dessa två varianter med hjälp av ett exempel (källkod).

När man utför upprepningar i program förekommer det ofta man vill öka värdet av en tilldelningsvariabel som sätts som räknare. Där med kommer öknings och minskningsoperatorer väl till pass.

1.5.1 Ökning och minskningsoperatorn.

- ++variabelNamn ökar önskad heltalsvariabel med 1.
- - -variabelNamn minskar önskad heltalsvariabel med 1.

Nedan är ett program som räknar från 0 upp till 5 och sedan ner igen från 5 till 0 med hjälp av dessa operatorer. Operatorerna är definerade i **for**-loopernas förändringsdel.

```
public class Increment
{
   public static void main(String[] args)
   {
      String a = "incriment: ", b = "decriment: ";

      for (int i = 0; i < 5; ++i)
      {
         System.out.println(a + i);

         if (i == 4)
         {
            for (int d = 5; d >= 0; --d)
            {
               if (d == 5)
                  System.out.println("Peek! Going down..");
               else
                  System.out.println(b + d);
            }
         }
      }
   }
}
```

```
/*Utmatning:*/
incriment: 0
incriment: 1
incriment: 2
incriment: 3
incriment: 4
Peek! Going down..
decriment: 4
decriment: 3
decriment: 2
decriment: 1
decriment: 0
```

1.5.2 Prefix och postfix.

Prefix(++variabelNamn) och postfix(variabelNamn++) är väldigt lika, men inte riktigt desamma. Hade postfix används till exemplet ovan hade det inte blivit någon skillnad i resultatet. Båda ökar (och med - -, minskar) variablernas värden. Prefix (++i) ökar värdet **innan** det aktuella uttrycket utvärderas, medan postfix(i++) ökar värdet **efter** att uttrycket har utvärderats!
Exempel

```
public class Postfix
{
   public static void main(String[] arg)
   {
      int x, y, z, a, b, c;

      x = 8; y = 6;
      // Prefix
      z = ++x * --y;
      System.out.println(z); // 45 = 9 * 5

      a = 8; b = 6;
      // Postfix
      c = a++ * b--;
      System.out.println(c); // 48 = 8 * 6
   }
}
```

1.6 Skriv denna programsats på ett annat sätt (mindre kod):

```
summa = summa + delsumma;
```

Kan förenklas till,

```
summa += delsumma;
```