Lösningsförslag: Instuderingsfrågor, del B

Uppgift 1.

a) Utskriften blir:
```
10 minutes has passed.
30 minutes has passed.
50 minutes has passed.
```

b) Utskriften blir:
```
30
```

c) Utskriften blir:
```
Resultat: 1
Resultat: 2
Resultat: 0
Resultat: 1
Resultat: 2
```

d) Utskriften blir:
```
a = 1 b = 2
a = 2 b = 2
a = 3 b = 2
a = 4 b = 4
a = 5 b = 6
a = 6 b = 8
```

e) Utskriften blir:
```
Sum is 20
Count is 5
```

Uppgift 2.

b)
```
double i = 0.25;
while (i <= 5.0) {
   System.out.print(i + " ");
   i = i + 0.25;
}
System.out.println();
```

b)
```
int i = 1, sum = 0;;
while (i <= 8) {
   sum = sum + i;
   System.out.print(sum + " ");
   i = i + 1;
}
System.out.println();
```

c)
```
while (!finished) {
   //
}
```

d)
```
while (counter > LOWER && counter < UPPER) {
   //
}
```

Uppgift 3.

Det uppkommer en division med 0 i uttrycket 1.0/i, eftersom **while**-satsen går ett varv längre än vi tänkt oss. Detta kan givetvis åtgärdas genom att ändra villkoret i **while**-satsen så den löper ett varv mindre, men det egentliga felet är inte villkoret i **while**-satsen utan startvärdet på variabeln i! Varför sätta denna till 11 när den borde vara 10? Denna felaktiga sätta att initiera startvärdet innebär att det måste korrigeras inne i **while**-satsen genom att räknas ner med 1 innan själva beräkningarna utförs. En korrekt programsekvens har följande utseende:

```
int i = 10;
double sum = 0;
while (i >= 1) {
    sum = sum + 1.0 / i;
    i = i-1;
}
```

Uppgift 4.

b) Terminerar inte, detta pga att det uppstår avrundningsfel när vi gör beräkningar på reella tal.

c) Terminerar inte, eftersom `x` får aldrig värdet 55.

d) Terminerar inte. Villkoret säger att loopen skall fortsätta så länge som `i < 10` eller `sum != 15`, dvs loopen bryts när `x>=10` och `sum == 15`. Detta inträffar dock aldrig. Visserligen kommer `sum` att under exekveringen ha värdet 15, men dock inte samtidigt som `x >= 10`.

e) Terminerar inte. Det står nämligen ett semikolon (;) efter villkoret i **while**-satsen, vilket med all säkerhet är misstag av programmeraren. **while**-satsen består alltså endast av en tom sats och i en tom sats görs inget, varför utfallet av villkoret som styr **while**-satsen inte kommer att förändras.

Om vi indenterar programmet som det logiskt tolkas syns detta tydligare:

```
int k = 1;
while (k != 10)
    ;
{
    System.out.println(k);
    k = k +1;
}
```

Uppgift 5.

a)	b)	c)	d)	e)	f)
1	4	-8	1	5	15
2	8	-4	2	7	12
3	12	0	4	9	9
0	16	4	8	11	6
1	20	8	16	13	

Uppgift 6.

Om man använder en for-sats som en generell villkorsloop är det mycket lätt att få obegriplig kod, en while-sats är att föredra vid villkorsloopar då dess syntax är mycket enklare att förstå.

Uppgift 7.

Den styrande variabeln i förändras inuti **for**-satsen! När man läser initieringen av for-satsen

```
for (int i = 1; i <= 10; i = i + 1)
```

förväntar man sig att satsen skall genomlöpas då styrande variabeln i har värdena 1,2,3,...,10. MEN, satsen genomlöps då i har värdena 1,3,5,7 och 9!

Vi har här ett bra exempel på hur man inte skall använda en **for**-sats. **for**-satsen skall enbart nyttjas som en räkneloop, vilket innebär att man aldrig inom **for**-satsen skall påverka den styrande variabeln i **for**-satsen eller villkoret i **for**-satsen. För att erhålla det önskade resultatet borde man istället använt följande sats:

```
for (int i = 1; i <= 9; i = i + 2) {
    System.out.println(i);
}
```

Uppgift 8.

a) **for**-satsen har skrivits på ett felaktigt sätt! Delarna i **for**-satsen skall separeras med semikolon (;) och inte med komma (,). Satsen skall se ut på följande sätt:

```
for (int k = 1; k <= 10; k = k +1)
    System.out.println(k);
```

b) Variabeln i deklareras två gånger inom samma programenhet. Hur detta skall rättas till beror på om variabeln i enbart skall existera inom **for**-satsen eller om den även skall existera utanför **for**-satsen.

Fall 1: variabeln i är definierad enbart inom **for**-satsen

```
for (int i = 1; i <= 10; i = i +1) {
    //här kommer en eller flera programsatser
}
```

Fall 2: variabeln i är definierad inom hela det aktuella programblocket.

```
int i;
for (i = 1; i <= 10; i = i +1) {
    //här kommer en eller flera programsatser
}
```

c) Variabeln j är odefinierad i utskriftssatsen! Det står ett semikolon efter själva **for**-satsen, vilket innebär att **for**-satsen enbart innehåller en tom sats. Eftersom variabeln j är lokal inom **for**-satsen och utskriftssatsen inte ligger i **for**-satsen är således j odefinierad i utskriftssatsen. Om vi indenterar programmet som det logiskt tolkas syns detta tydligare:

```
for (int j = 1; j <= 10; i = j +1)
    ;
{
 System.out.println(j);
      //här kommer eventuellt flera programsatser
}
```

Ett programblock får lägg in varsomhelst i ett program där en sats får förekomma.

d) Kompileringsfelet uppkommer pga att variabeln i redan är deklarerad när den deklareras i den inre **for**- satsen. Avsikten med kodavsnittet är troligen att den yttre **for**-satsen skall löpa 10 gånger och den inre **for**-satsen också skall löpa 10 gånger, varför koden borde ha följande utseende:

```
int antalVarv = 0;;
for (int i = 1; i <=10; i = i + 1)
    for (int j = 1; j <=10; j = j + 1)
        antalVarv = antalVarv + 1;
```

Uppgift 9.

a)
```
******
******
******
```

b)
```
*
**
***
```

c)
```
5670
5671
5672
5673
```

Uppgift 10.

a)
```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

b)
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

c)
```
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5
```

Uppgift 11.

a) 27 gånger

b) 18 gånger

Uppgift 12.

a)
```
for (int i = 1; i <= 15; i = i +2)
    System.out.println(i);
```

b)
```
for (double i = 0.25; i <= 5.0; i = i + 0.25) {
    System.out.println(i);
```

c)
```
for (int i = 10; i >= -10; i = i -2) {
    System.out.println(i);
```

d)
```
for (int i = 0; i <= 9; i = i + 1) {
    System.out.println(i * i);
```

Uppgift 13.

a) En **for**-sats är lämpligast, eftersom vi har en räkneloop.

b) En **while**-sats är lämpligast, eftersom vi har en villkorsloop.

c) En **for**-sats är lämpligast, eftersom vi har en räkneloop.

d) En **while**-sats är lämpligast, eftersom vi har en villkorsloop.

Uppgift 14.

Utskriften blir:

```
i = 4
```

Uppgift 15.

Variabeln more är okänd utanför programblocket, i vilket variabeln är deklarerad. Variablen more måste således deklareras utanför **do-while**-blocket:

```
        boolean more;
        do {
            i = i + 1;
            sum = sum * i;
            if (i > 6)
                more = false;
            else
                more = true;
    } while (more);
```

Uppgift 16.

Lämpligen används en **do**-sats enligt:

```
do {
    String indata = JOptionPane.showInputDialog("Ge talet: ");
    int tal = Integer.parseInt(indata);
    if (tal < 1 || tal > 7)
        JOptionPane.showMessageDialog(null, "Fel indata! Försök igen");
}while ( tal < 1 || tal > 7);
```

En alternativ lösning är att lägga inläsningen i en evighetssats och hoppa ut när ett korrekt värde blivit inläst:

```
while(true) {
    String indata = JOptionPane.showInputDialog("Ge talet: ");
    int tal = Integer.parseInt(indata);
    if (tal >= 1 && tal <= 7)
        break;
    JOptionPane.showMessageDialog(null, "Fel indata! Försök igen");
}
```

Denna lösning är sämre eftersom vi lämnar loopen via en *bakdörr*. En **while**-loop är det normalt (≈alltid) villkoret som skall bestämma när loopen skall lämnas.

Uppgift 17.

```
    import javax.swing.*;
    public class MetersToYards {
        public static void main(String[] arg) {
            final double METERS_PER_YARD = 0.9144;
            while (true) {
                String indata = JOptionPane.showInputDialog("Ange antalet meter: ");
                if (indata == null)
                    break;
                double lengthInMeters = Double.parseDouble(indata);
                double lengthInYards = lengthInMeters / METERS_PER_YARD;
                int yards = (int) lengthInYards;
                double rest = lengthInYards - yards;
                int feet = (int) (3*rest);
                rest = rest -feet/3.0;
                int inches = (int) (3*12*rest);
                JOptionPane.showMessageDialog(null, lengthInmeters + " är " + yards + " yards, " +
                                                    feet + " feeet och " + inches + " inches.");
            }
        } // main
    }//MetersToYards
```