Föreläsning 3

Iteration

while-satsen
for-satsen
do-satsen

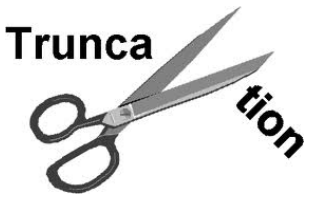
Datatypen double

De enkla datatyperna, som används för att lagra tal (t.ex. **int** och **double**), har en begränsad storlek och representerar således endast en delmängd av de verkliga talen.

Detta innebär att ett resultat eller ett mellanresultat från en beräkning av ett uttryck kan bli ett värde som inte kan lagras.

Resulterar uttrycket i ett för stort värde uppstår ***overflow*** och om uttrycket resulterar i ett för litet värde uppstår ***underflow***.

Reella tal lagras med ett bestämt antal signifikanta siffror, vilket innebär att det vid beräkningar uppstår ***trunkeringsfel*** och ***kancellationsfel***.



Datatypen double

Exempel:

Antag att vi lagrar decimala tal med 4 decimaler. Uttrycket

1.0/3.0 + 1.0/3.0 +1.0/3.0

kommer då att evalueras på följande sätt:

1.0/3.0 + 1.0/3.0 +1.0/3.0 =

0.3333 + 0.3333 + 0.3333 = 0.9999

resultatet blir inte 1.0, som man kanske trott!

Konsekvens:

När man handhar reella tal skall man aldrig jämföra på exakthet, utan istället jämföra på tillräcklig noggrannhet.

Problemexempel

I en triangel kan man beteckna sidorna *a*, *b* och *c*.

Om man känner till längden av sidorna *a* och *b* samt vinkeln β mellan dessa sidor, kan man räkna ut längden av sidan *c* med hjälp av formeln:

$$c = \sqrt{a^2 + b^2 - 2ab \cdot \cos\beta}$$

Skriv ett program som läser in längderna på två sidor i en triangel och vinkeln mellan dessa sidor (uttryckt i grader).

Programmet skall avgöra om triangeln är liksidig (alla sidor lika långa), likbent (två sidor lika långa) eller oliksidig (inga sidor är lika långa).

Programmet skall ge lämpliga utskrifter.

Analys:

Indata: Längderna *A* och *B* på två sidor i triangeln, samt den mellanliggande vinkeln *VG* i grader.

Utdata: Utskrift av huruvida triangeln är liksidig, likbent eller oliksidig.

Design:

Diskussion:

I klassen `Math` i Java anges parametrarna till de trigonometriska funktionerna i *radianer*. Därför är en omvandling av vinkeln från grader till radianer nödvändig att göra.

Denna omvandling kan antingen göras genom att använda metoden `toRadians` i klassen `Math`, eller genom att använda formeln

$$VR = VG \cdot \pi / 180$$

där *VR* är vinkeln uttryckt i radianer och *VG* är vinkeln uttryckt i grader.

För att två sidor skall betraktas som lika antas att sidornas längder skiljer sig med mindre än ε längdenheter.

Algoritm:

1. Läs längderna på sidorna *A* och *B*, samt gradtalet *VG* av den mellanliggande vinkeln.
2. Beräkna *VR* som den mellanliggande vinkeln uttryckt i radianer.
3. Beräkna längden *C* av den den okända sidan i triangeln med hjälp av formeln

$$C = \sqrt{A^2 + B^2 - 2AB \cdot \cos(VR)}$$

4. Om $|A - B| \leq \varepsilon$ och $|A - C| \leq \varepsilon$ och $|B - C| \leq \varepsilon$ så
   skriv ut att triangeln är liksidig
   annars om $|A - B| \leq \varepsilon$ eller $|A - C| \leq \varepsilon$ eller $|B - C| \leq \varepsilon$ så
   skriv ut att triangeln är likbent
   annars
   skriv ut att triangeln är oliksidig.

Varför måste vi göra jämförelsen $|B - C| \leq \varepsilon$?

Datarepresentation:

Längderna *A*, *B* och *C* samt vinklarna *VG* och *VR* är av datatypen **double**.
Konstanten PI, för att avbilda π finns tillgänglig i klassen Math.
Klassen Math har en klassmetod toRadians för att omvandla från grader till radianer.
Konstanten EPS = 0.001, för att avbilda ε.

Implementation:

```
/* Programmet läser in längderna av två sidor i en triangel samt mellanliggande vinkel,
 * och skriver ut huruvidatriangeln är liksidig, likbent eller oliksidig. */
import javax.swing.JOptionPane;
import java.util.Scanner;
public class Triangle {
   public static void main( String[] arg) {
      final double EPS = 0.001;
      String input = JOptionPane.showInputDialog("Ange längden av två sidor samt"
                                                 + " mellanliggande vinkel: ");
      Scanner sc = new Scanner(input);
      double a = sc.nextDouble();
      double b = sc.nextDouble();
      double angleGrades = sc.nextDouble();
      double angleRadians = Math.toRadians(angleGrades);
      double c = Math.sqrt(a*a + b*b - 2.0*a*b*Math.cos(angleRadians));
      if ((Math.abs(a - b) <= EPS) && (Math.abs(a - c) <= EPS) && (Math.abs(b - c) <= EPS))
         JOptionPane.showMessageDialog(null,"LIKSIDIG");
      else if ((Math.abs(a -b) <= EPS) || (Math.abs(a - c) <= EPS) || (Math.abs(b - c) <= EPS))
         JOptionPane.showMessageDialog(null,"LIKBENT");
      else
         JOptionPane.showMessageDialog(null, "OLIKSIDIG");
   }//main
}//Triangle
```

Styrstrukturerna i Java

Sekvens: tilldelningssatsen
selektionssatser
iterationssatser
return-satsen
anrop av **void**-metod
programblock
exception

Selektion: **if**-satsen
switch-satsen

Iteration: **while**-satsen
do-satsen
for-satsen

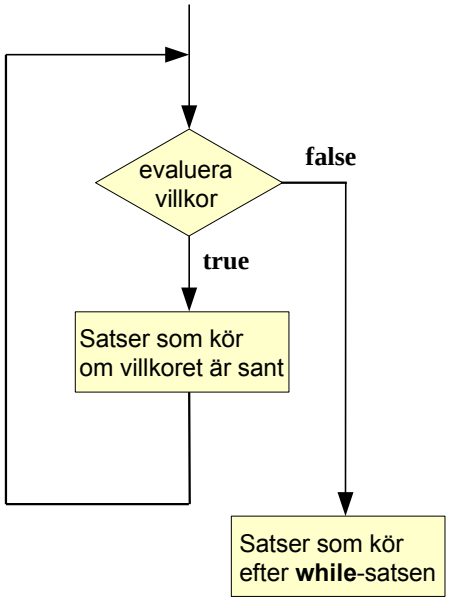
Olika typer av iteration

a) tills ett visst villkor blivit uppfyllt (***villkorsloop***)
b) ett på förhand bestämt antal gånger (***räkneloop***)

Exempel:

a) "Rör om tills smöret har smält" (villkorsloop).
b) "Gå fem steg framåt" (räkneloop)

Iteration: *while-satsen*



Upprepning av en sats:

```
while (villkor)
    sats;
```

Upprepning av ett programblock:

```
while (villkor) {
    sats1;
    . . .
    satsN;
}
```

Problemexempel

Antalet bakterier y_n i en bakterieodling efter *t* tidsenheter ges av formeln

$$y_n = y_s e^{1.386t}$$

där y_s är antalet bakterier vid $t = 0$.

Skriv ett program som beräknar hur många tidsenheter det tar innan en bakterieodling som innehåller en bakterie innehåller minst 1 miljard bakterier.

Analys:

Indata: Ingen

Utdata: Hur lång tid det tar innan bakterieodlingen innehåller minst 1 miljard bakterier.

Design:

Algoritm:

1. Sätt tillväxttiden *tid* till 0.
2. Sätt *yStart*, antalet bakterier vid tiden 0, till 1.
3. Sätt totala antalet bakterier i odlingen *yTotal* till *yStart*.
4. Upprepa så länge som *yTotal* är mindre än 1000000000
   4.1. Öka *tid* med 1.
   4.2. Beräkna *yTotal* mha formeln

   $$yTotal = yStart * e^{1.386*tid}$$
5. Skriv ut värdet av *tid*.

Datarepresentation:

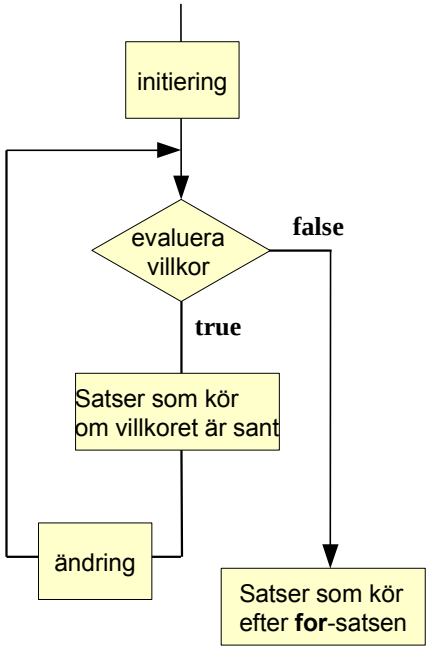
tid är av datatypen **int**.

yStart och yTotal är reella tal, dvs av datatypen **double**.

Implementation:

```
import javax.swing.JOptionPane;
public class Bacteria {
   public static void main( String[] arg) {
      int time = 0;
      double yStart = 1;
      double yTotal = yStart;
      while (yTotal < 1.0e9) {
         time = time + 1;
         yTotal = yStart*Math.exp(1.386*time);
      }
      JOptionPane.showMessageDialog(null, "Det tar " + time + " tidsenheter innan"
                                          + " odlingen innehåller 1 miljard bakterier.");
   } //main
} //Bacteria
```

Iteration: *for-satsen*



Upprepning av en sats:

```
for (initiering; villkor; ändring)
   sats;
```

Upprepning av ett programblock:

```
for (initiering; villkor; ändring)  {
   sats1;
   . . .
   satsN;
   }
```

for-satsen är en villkorsloop, men använd **for**-satsen *endast vid räkneloopar*!!!

for-satsen

```
for (int i = 1; i <= 5; i = i + 1)
    System.out.println(i);
```

Ger utskriften:

1
2
3
4
5

```
for (char c = 'd'; c >= 'a'; c = (char) (c - 1))
    System.out.println(c);
```

Ger utskriften:

d
c
b
a

```
for (double x = 0.5; x < 0.8; x = x + 0.1)
    System.out.println(x);
```

Ger utskriften:

0.5
0.6
0.7
0.7999999999999999

Troligen inte förväntat resultat!
*Den styrande variabeln i en **for**-sats bör vara en uppräkningsbar datatyp.*

Problemexempel

Att omvandla en temperatur angiven i grader Fahrenheit till grader Celsius sker med följande formel:

C = (F -32) • 5 / 9, där C anger grader Celsius och F grader Fahrenheit

Skriv ett program som skriver ut en omvandlingstabell från grader Fahrenheit till grader Celsius i intervallet från -10.0 till +10.0 grader Fahrenheit, där var 0.5:e grad anges.

Fahrenheit	Celsius
-10.0	-23.3
-9.5	-23.1
-9.0	-22.8
. . .	. . .
9.0	-12.8
9.5	-12.5
10.0	-12.2

Analys:

Indata: Ingen

Utdata: En konverteringstabell från grader Fahrenheit till grader Celcius som anger var 0.5:e grad i intervallet -10.0 till +10.0.

Design:

Algoritm:

1. Skriv ut rubriken
2. För f = -10.0 till 10.0 stega 0.5
   2.1. c = (f -32)*5/9
   2.2. Skriv ut f och c

Datarepresentation:

c och f är av datatypen **double**.

Implementation:

```
public class FahrenheitToCelsius {
   public static void main(String[] arg) {
      System.out.printf("%15s%12s", "Fahrenheit", "Celsius\n");
      for (double fahrenheit = -10; fahrenheit <= 10; fahrenheit = fahrenheit + 0.5) {
          double celsius = (fahrenheit -32) * 5.0 / 9.0;
          System.out.printf("%12.1f%13.1f\n", fahrenheit, celsius);
       }
   } //main
} //FahrenheitToCelsius
```

Problemexempel

Skriv ett program som läser in två heltal, som representerar antal rader respektive antal kolumner, och skriver ut nedanstående mönster i kommandofönstret:

Design:

Algoritm:

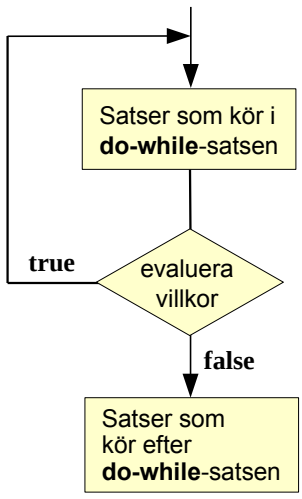
1. Läs antal rader *r* och antal kolumner *k*
2. För varje rad
   2.1. För varje kolumn
      2.1.1. Skriv ut tecknet '*'
   2.2. Skriv ut radslut

Implementation:

```
import javax.swing.*;
import java.util.*;
public class WriteStars {
   public static void main(String[] args) {
      String input = JOptionPane.showInputDialog("Ange antal rader och antal kolumner: ");
      Scanner sc = new Scanner(input);
      int nrOfRows = sc.nextInt();
      int nrOfColumns = sc.nextInt();
      for (int row = 1; row <= nrOfRows; row = row +1) {
         for (int col = 1; col <= nrOfColumns; col = col +1) {
            System.out.print("*");
         }
         System.out.println();
      }
   }//main
}//WriteStars
```

Iteration: *do-satsen*

I en **while**-satsen beräknas testuttrycket inför varje varv i loopen.
I en **do**-satsen beräknas testuttrycket *efter varje varv* i loopen.
En **do**-sats genomlöps *minst en gång*, medan en **while**-sats kan genomlöpas noll gånger



Upprepning av en sats:

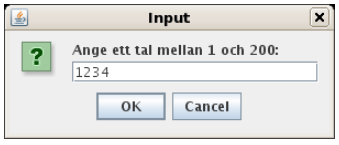
```
do
   sats;
while (villkor);
```

Upprepning av ett programblock:

```
do {
   satser
} while (villkor);
```

Exempel: Inläsningskontroll

```
import javax.swing.JOptionPane;
public class InspectInput {
   public static void main (String[] arg) {
      int number;
      do {
         String input = JOptionPane.showInputDialog("Ange ett tal mellan 1 och 200:");
         number = Integer.parseInt(input);
         if (number < 1 || number > 200)
            JOptionPane.showMessageDialog(null, "Ogiltigt tal! Försök igen!");
      } while (number <1 || number > 200);
      JOptionPane.showMessageDialog(null, "Det accepterade talet är " + number);
   }//main
}//InspectInput
```



Variablers räckvidd (scope)

En variabels räckvidd är det kodavsnitt inom vilket variabeln går att använda.

Grundprincip: En variabel är synlig endast inom det programblock där variabeln deklarerats.

```
public static void main (String[] arg) {
   int number;
   do {
      String input = JOptionPane.showInputDialog("Ange ett tal mellan 1 och 200:");
      number = Integer.parseInt(input);
      if (number < 1 || number > 200)
         JOptionPane.showMessageDialog(null, "Ogiltigt tal! Försök igen!");
   } while (number <1 || number > 200);
   JOptionPane.showMessageDialog(null, "Det accepterade talet är " + number);
}//main
```

number och input kända här!

input okänd här!

En variabels räckvidd skall begränsas så mycket som möjligt.

Evighetsloop och satsen break

Konstruktionen

```
while (true) {
   satser
}
```

innebär en ***evighetsloop***.

En evighetsloop uppkommer vanligtvis på grund av ett misstag hos programmeraren, men kan ibland vara avsiktlig.

En loop kan när som helst lämnas med hjälp av satsen **break**:

```
while (villkor) {
   ...
   if (villkor för att sluta)
      break;
   ...
}
```

*Att lämna en loop via en **break**-sats, skall användas mycket restriktivt!!!*

Varför???

Observera att vid nästlade loopar lämnas den loop i vilken **break**-satsen står!

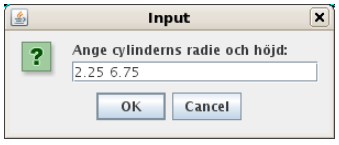
```
while (villkor) {
   ...
   while (villkor) {
      ...
      if (villkor för att sluta)
         break;
      ...
   }
   ...
}
```

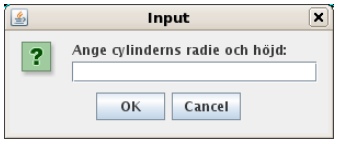
Upprepad programkörning

*Ett av få exempel där det är okej att lämna en loop via en **break**-sats.*

```
import javax.swing.JOptionPane;
import java.util.Scanner;
public class RepeatedExecution {
   public static void main (String[] arg) {
      while (true) {
         String input = JOptionPane.showInputDialog("Ange cylinderns radie och höjd:");
         if (input == null)
            break;
         Scanner sc = new Scanner(input);
         double radius  = sc.nextDouble();
         double height = sc.nextDouble();
         double volume = Math.PI * Math.pow(radius, 2) * height;
         String output = String.format("%s %.2f", "Volymen av cylindern är", volume);
         JOptionPane.showMessageDialog(null, output);
      }//while
   }//main
}//RepeatedExecution
```

*Cancel-knappen returnerar **null***





for-, while- eller do-satsen?

while-satsen är mest generella iterationssatsen, eftersom **for**- och **do**-satsen enkelt kan simuleras med en **while**-sats.

```
for (initiering; villkor; ändring)
   sats;
```

```
initiering;
while (villor) {
   sats;
   ändring;
}
```

```
do
   sats;
while (villkor);
```

```
sats;
while (villor)
   sats;
```

- använd **for**-satsen vid räkneloopar
- använd **do**-satsen om loopen skall gå minst ett varv
- använd **while**-satsen i alla övriga fall
- är du osäker på vilken iterationssats som skall väljas, välj **while**-satsen

Problemexempel

Skriv ett program som läser en indataserie bestående av N positiva heltal samt beräknar och skriver ut medelvärdet av dessa tal.

Analys:

Indata: Antalet tal i dataserien samt själva dataserien.

Utdata: Medelvärdet av talen i dataserien.

Speciella åtgärder: Om inga tal ingår i dataserien kan inte medelvärdet beräknas.

Design:

Utkast till algoritm:

1. Läs in antalet heltal som ingår i dataserien till variabeln *antal*.
2. Läs talen och beräkna talens sammanlagda summa i variabeln *summa*.
3. Beräkna medelvärdet *medel* mha formeln *medel* = *summa* / *antal*.
4. Skriv ut medelvärdet, dvs värdet av variabeln *medel*.

Mer detaljerad algoritm:

1. Läs in antalet heltal som ingår i dataserien till variabeln *antal*.
2. Sätt *summa* till 0.
3. Upprepa *antal* gånger
   3.1. Läs ett tal till variabeln *tal*.
   3.2. Addera *summa* och *tal* och spara resultatet i *summa*.
4. Om *antal* > 0 så
   4.1. Beräkna medelvärdet *medel* mha formeln
   medel = *summa* / *antal*
   4.2. Skriv ut medelvärdet *medel*

   annars
   4.3. Skriv ut att inga värden ingick i dataserien.

Datarepresentation:

antal, *summa* och *tal* är heltal av typen **int**.

medel är ett reellt tal av typen **double**.

Implementation: Med användning av en **for**-sats

```
import javax.swing.JOptionPane;
public class AddValues {
   public static void main(String[] arg) {
      String input = JOptionPane.showInputDialog("Ange antalet tal i serien:");
      int number = Integer.parseInt(input);
      int sum = 0;
      for (int i = 1; i <= number; i = i + 1) {
         input = JOptionPane.showInputDialog("Ange tal nr " + i + ": ");
         int value = Integer.parseInt(input);
         sum = sum + value;
      }
      if (number > 0) {
         double mean = (double) sum / (double) number;
         JOptionPane.showMessageDialog(null, "Medelvärdet av talen är "+ mean);
      }
      else {
         JOptionPane.showMessageDialog(null, "Inga tal ingick i serien!");
      }
   }//main
}//AddValues
```

Implementation: Med användning av en **while**-sats

```
import javax.swing.JOptionPane;
public class AddValues2 {
   public static void main(String[] arg) {
      String input = JOptionPane.showInputDialog("Ange antalet tal i serien:");
      int number = Integer.parseInt(input);
      int sum = 0;
      int i = 0;
      while (i < number) {
         i = i + 1;
         input = JOptionPane.showInputDialog("Ange tal nr " + i + ": ");
         int value = Integer.parseInt(input);
         sum = sum + value;
      }
      if (number > 0) {
         double mean = (double) sum / (double) number;
         JOptionPane.showMessageDialog(null, "Medelvärdet av talen är "+ mean);
      }
      else {
         JOptionPane.showMessageDialog(null, "Inga tal ingick i serien!");
      }
   }//main
}//AddValues2
```

Problemexempel

Skriv ett program som läsa och beräkna medelvärdet av *ett okänt antal positiva heltal*. Serien av heltalen *avslutas med ett negativt heltal* (vilket inte ingår i serien).

Analys:

Indata: Talen i dataserien som skall läsas in, samt ett negativt tal som avbryter inläsningen.

Utdata: Medelvärdet av talen som ingår i dataserien.

Speciella åtgärder: Om inga tal ingår i dataserien kan inte medelvärdet beräknas.

Design:

Algoritm:

1. Sätt *summa* till 0 och *antal* till 0.
2. Läs ett tal till variabeln *tal*.
3. Upprepa så länge som *tal* ≥ 0
   - 3.1. Öka *antal* med 1.
   - 3.2. Addera *summa* och *tal* och spara resultatet i *summa*.
   - 3.3. Läs ett tal till variabeln *tal*.
4. Om *antal* > 0 så
   - 4.1. Beräkna medelvärdet *medel* mha formeln

     medel = *summa*/*antal*
   - 4.2. Skriv ut medelvärdet *medel*

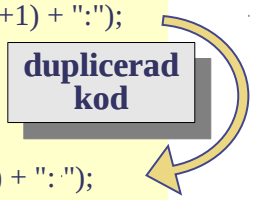
   annars
   - 4.3. Skriv ut att inga värden ingick i dataserien.

Datarepresentation:

antal, *summa* och *tal* är heltal av typen **int**.

Implementation:

```
import javax.swing.JOptionPane;
public class AddValues3 {
   public static void main(String[] arg) {
      int number = 0;
      int sum = 0;
      String input = JOptionPane.showInputDialog("Ange tal nr " + (number+1) + ":");
      int value = Integer.parseInt(input);
      while (value > 0) {
         number = number + 1;
         sum = sum + value;
         input = JOptionPane.showInputDialog("Ange tal nr " + (number + 1) + ": ");
         value = Integer.parseInt(input);
      }
      if (number > 0) {
         double mean = (double) sum / (double) number;
         JOptionPane.showMessageDialog(null, "Medelvärdet av talen är "+ mean);
      }
      else {
         JOptionPane.showMessageDialog(null, "Inga tal ingick i serien!");
       }
   }//main
}//AddValues3
```



Problemexempel

Skriv ett program som läsa och beräkna medelvärdet av ett okänt antal heltal.

Analys:

Diskussion: Nu kan alla heltal ingå i dataserien och hur skall vi då kunna markera slutet på serien? Ett sätt är att utnyttja Cancel-knappen i dialogrutan. När man trycker på *Cancel* returneras värdet **null**.

Indata: Talen i dataserien som skall läsas in. Inläsningen avbryts genom att trycka på *Cancel*-knappen i dialogrutan.

Utdata: Medelvärdet av talen som ingår i dataserien.

Speciella åtgärder: Om inga tal ingår i dataserien kan inte medelvärdet beräknas.

Design:

Algoritm:

1. Sätt *summa* till 0 och *antal* till 0.
2. Upprepa
   - 2.1. Gör en inläsning från dialogfönstret till variabeln *indata*.
   - 2.2. Om indata == **null** gå till punkt 3.
   - 2.3. Öka *antal* med 1.
   - 2.4. Konvertera *indata* till heltalet *tal*.
   - 2.5. Addera *summa* och *tal* och spara resultatet i *summa*.
3. Om *antal* > 0 så
   - 3.1. Beräkna medelvärdet *medel* med hjälp av formeln

     $$medel = summa/antal$$
   - 3.2. Skriv ut medelvärdet *medel*

   annars
   - 3.3. Skriv ut att inga värden ingick i dataserien.

Datarepresentation:

antal, *summa* och *tal* är heltal av typen **int**.

medel är ett reellt tal av typen **double**.

Implementation:

```
import javax.swing.JOptionPane;
public class AddValues4 {
   public static void main( String[] arg) {
      int number = 0;
      int sum = 0;
      while (true) {
         String input = JOptionPane.showInputDialog("Ange nästa tal i serien\n" +
                                                    "Avsluta ned Cancel");
         if (input == null)
            break;
         int value = Integer.parseInt(input);
         number = number + 1;
         sum = sum + value;
      }
      if (number > 0) {
         double mean = (double) sum / (double) number;
         JOptionPane.showMessageDialog(null, "Medelvärdet av talen är "+ mean);
      }
      else {
         JOptionPane.showMessageDialog(null, "Inga tal ingick i serien!");
       }
   }//main
}//AddValues4
```

Implementation:

```
import javax.swing.JOptionPane;
import java.util.Scanner;
public class AddValues5 {
   public static void main( String[] arg) {
      int number = 0;
      int sum = 0;
      while (true) {
         String input = JOptionPane.showInputDialog("Ange talen. Avsluta med Cancel!");
         if (input == null)
            break;
         Scanner sc = new Scanner(input);
         while(sc.hasNext()) {
            int value = sc.nextInt();
            number = number + 1;
            sum = sum + value;
         }
      }
      if (number > 0) {
         double mean = (double) sum / (double) number;
         JOptionPane.showMessageDialog(null, "Medelvärdet av talen är "+ mean);
      }
      else {
         JOptionPane.showMessageDialog(null, "Inga tal ingick i serien!");
      }
   }//main
}//AddValues5
```