Föreläsning 10

- Villkor
- Iteration - for-sats
- Iteration - while-sats
- Iteration – do-while-sats
- continue, break

JF: 4.5, 4.7, 4.8 (ej ”Iterators and for Loops”)

Villkor

```
int number;
Random rand = new Random();
String resTrue = "true: ", resFalse = "false: ";

number = rand.nextInt(51);
if( true ) {
   resTrue +=  number + " ";
} else {
   resFalse += number + " ";
}
JOptionPane.showMessageDialog(null, resTrue + "\n" + resFalse);
```

Conditions.java

1 number är delbart med 7

2 number är i intervallet 5-20

3 number är ej i intervallet 5-20

Programmet slumpar ett tal i intervallet 0-50. Om villkoret i if-satsen beräknas till true så ändras strängen *resTrue*, annars ändras *resFalse*.

Trevligt vore det om detta skedde t.ex. 100 ggr. Då testas villkoret bättre. Det behövs en upprepning, **iteration**, i programmet.

Iteration – upprepning av instruktioner

```
    int number;
    Random rand = new Random();
    String resTrue = "true: ", resFalse = "false: ";

1   number = rand.nextInt(51);
2   if( true ) {
3      resTrue +=  tal + " ";
4   } else {
5      resFalse += tal + " ";
6   }
    JOptionPane.showMessageDialog(null, resTrue + "\n" + resFalse);
```

Vi vill upprepa raderna 1-6 ett antal gånger.

Iteration - for-loop

```
    int number;
    Random rand = new Random();
    String resTrue = "true: ", resFalse = "false: ";

    for(int i=1; i<=100; i++) {
1       number = rand.nextInt(51);
2       if( true ) {
3          resTrue +=  tal + " ";
4       } else {
5          resFalse += tal + " ";
6       }
    }
    JOptionPane.showMessageDialog(null, resTrue + "\n" + resFalse);
```

Raderna 1-6 kommer att upprepas 100 gånger. I ovanstående exempel används en for-loop.

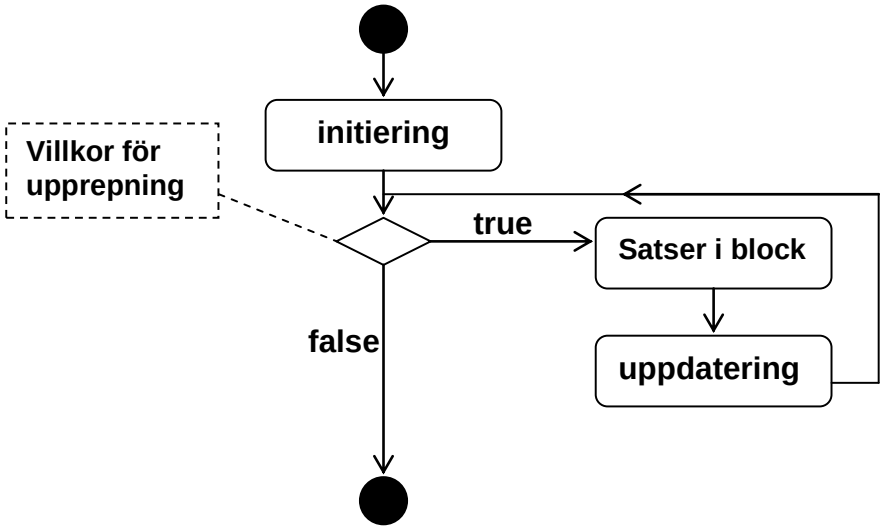
For1.java

for-loop

Ordinär for-sats:

```
for( initiering ; villkor för upprepning ; uppdatering ) {
   // block med kod som upprepas om villkor är sant
}
```

Aktivitetsdiagram



1. **Initiering utförs från början. Det är enda gången initieringen utförs.**
2. **Sedan beräknas villkoret.**
   - **Blir resultatet true så utförs koden inuti for-loopen.**
     Sedan utförs upp-dateringen.
     Slutligen upprepas steg 2
   - **Blir resultatet false så fortsätter programmet efter loopen.**

for-loop, exempel

For-satsen används med fördel när något ska upprepas ett känt antal gånger.

- Du vill skriva ut talen 5, 6, 7, …, 12 så här:
  5 6 7 8 9 10 11 12

```
for( int i = 5 ; i <= 12; i++ ) {
   System.out.print( i + ” ” );
}
```

- Du vill skriva ut talen 10, 9, 8,…,1 så här:
  10 9 8 7 6 5 4 3 2 1

```
for( int i = 10 ; i >= 1 ; i-- ) {
   System.out.print( i + ” ” );
}
```

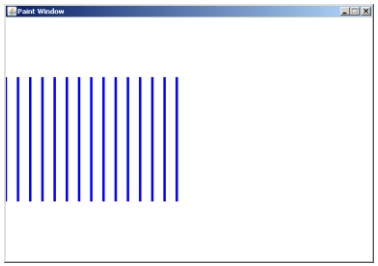
- Du vill skriva ut vart femte tal, första talet ska vara 0 och 8 tal ska skrivas ut
  0, 5, 10, 15, 20, 25, 30, 35

```
 int start = 0, n=8;
for( int i = 1 ; i <= n ; i++ ) {
   if( i < n ) {
      System.out.print( start + ", " );
   } else {
      System.out.print( start );
   }
   start += 5;
}
```

- Du vill rita n st vertikala linjer i PaintWindow

```
int distance = 20;
for(int i=0; i<n; i++) {
   window.line(i*distance, 100, i*distance, 300, Color.BLUE, 4);
}
```

For2java



PWExamples.java

Iteration - while-loop

```
    int number, i;
    Random rand = new Random();
    String resTrue = "true: ", resFalse = "false: ";

    i = 1;
    while( i<=100 ) {
1       number = rand.nextInt(51);
2       if( true ) {
3           resTrue +=  tal + " ";
4       } else {
5           resFalse += tal + " ";
6       }
        i++;
    }
    JOptionPane.showMessageDialog(null, resTrue + "\n" + resFalse);
```

Raderna 1-6 kommer att upprepas 100 gånger.

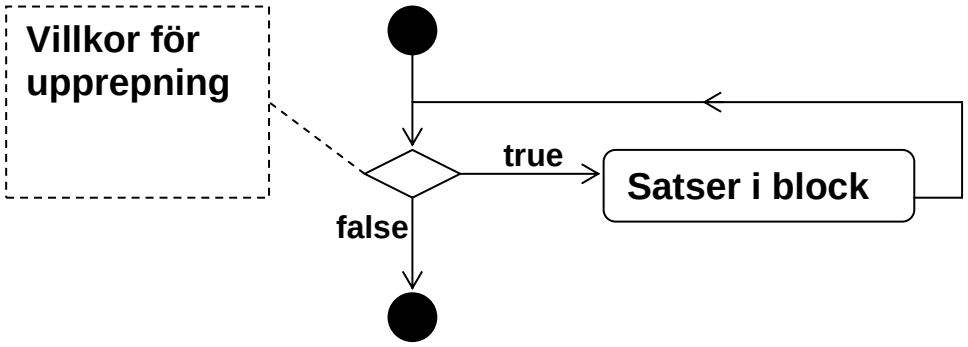
While1.java

while-loop

Ordinär while-sats:

```
while(villkor för upprepning) {
   // block med kod som upprepas om villkor är sant
}
```

Aktivitetsdiagram



1. **Villkoret beräknas.**
   - **Blir resultatet true så utförs koden inuti while-loopen. Sedan upprepas steg 1**
   - **Blir resultatet false så fortsätter programmet efter loopen.**

While2.java

while-loop, exempel

while-satsen används med fördel när något ska upprepas ett okänt antal gånger, dvs. när man inte med säkerhet vet hur många gånger något ska upprepas, men går även att använda vid ett känt antal upprepningar.

1 Användaren får ange positiva tal. När användaren anger 0 eller ett negativt tal skrivs de inmatade talens summa ut.

```
double nbr, sum=0;
nbr = Double.parseDouble( JOptionPane.showI…( ”Ange ett…” ) );
while( nbr > 0 ) {
   sum = sum + nbr;
   nbr = Double.parseDouble( JOptionPane.showI…( ”Ange ett…” ) );
}
JOptionPane.showM…( null, ”Summan av talen är ” + sum );
```

While3.java

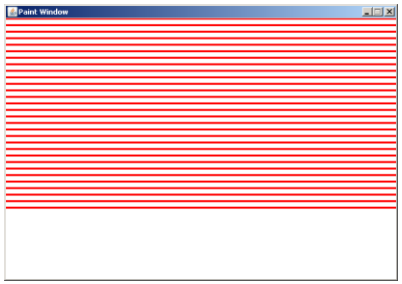
2 Du vill öka ett kapital tills det uppgår till ett visst belopp

```
while( sum < amount ) {
   sum = sum + …;
}
```

While4.java

3 Du vill rita n st horisontella linjer i PaintWindow

```
int i=0, distance=10;
while(i<n) {
    window.line(0, i*distance, window.getBackgroundWidth(), i*distance, Color.RED, 3);
    i++;
}
```



PWExamples.java

Iteration - do-loop

```
        int nbr, i;
        Random rand = new Random();
        String resTrue = "true: ", resFalse = "false: ";

        i = 1;
        do {
1           nbr = rand.nextInt(51);
2           if( true ) {
3              resTrue +=  tal + " ";
4           } else {
5              resFalse += tal + " ";
6           }
            i++;
        } while( i<=100 );
        JOptionPane.showMessageDialog(null, resTrue + "\n" + resFalse);
```

Raderna 1-6 kommer att upprepas 100 gånger.

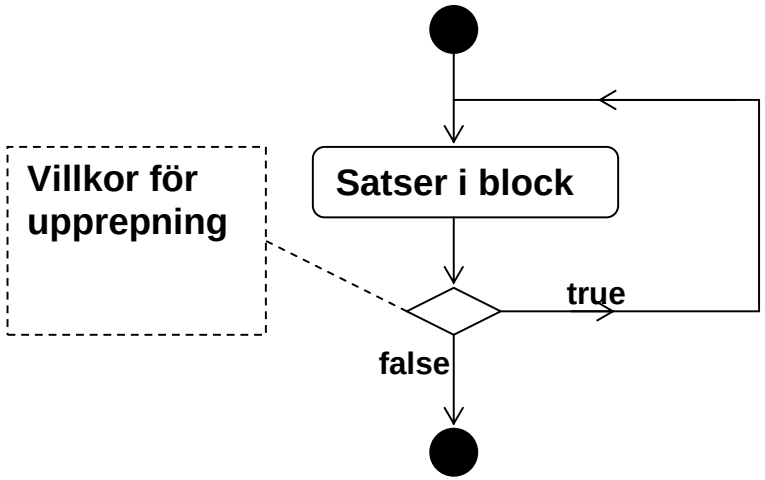
Do1.java

do-loop

Ordinär do-sats:

```
do {
   // block med kod som upprepas om villkor är sant
} while(villkor för upprepning);
```

Aktivitetsdiagram



1. **Koden i loopen utförs.**
2. **Sedan beräknas villkoret.**
   - **Blir resultatet true så upprepas steg 1**
   - **Blir resultatet false så fortsätter programmet efter loopen.**

Do2.java

do-loop, exempel

do-satsen används med fördel när något ska upprepas ett okänt antal gånger, men minst en gång.

Do-satsen är användbar t.ex. vid kontroll av värden.

1. Användaren ska mata in ett positivt heltal

```
do {
   nbr = Integer.parseInt( JOptionPane.showI…( ”Mata in ett positivt heltal” ) );
} while(nbr<=0)
```

Do3.java

2. Användaren ska välja ett meny-alternativ:

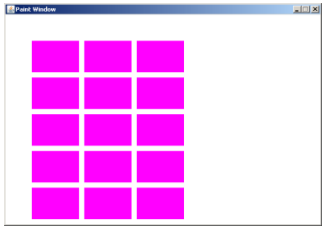
```
String message = "1. Addera\n2. Subtrahera\n3. Multiplicera\n\nVälj alternativ";
int choice;

// Inmatning av ett av menyalternativen
do {
   choice = Integer.parseInt( JOptionPane.showInputDialog( message ) );
} while( choice < 1 ) || ( choice > 3 ) );
```

Do4.java

Loopar kan ”nästlas”

En loop-sats kan utgöra en del av kroppen till en annan loop. Detta kallas för att nästla loopar.



Exempel: Rektanglar, *rows* rader och *cols* kolumner

```
for(int row = 0; row < rows; row++) {
   for(int col = 0; col < cols; col++) {
      window.fillRect(50 + row*100, 50 + col*70, 90, 60, Color.MAGENTA);
   }
}
```

PWExemples.java

Exempel: Multiplikationstabell, ett antal rader och kolumner ska skrivas ut

```
for( int factor = 1 ; factor <= 10 ; factor++ ) {
   for( int table = 1 ; table <= 12 ; table++ )
      System.out.print( String.format( ”%4d”, factor * table ) );
   System.out.println();
}
```

```
   1   2   3   4   5   6   7   8   9  10  11  12
   2   4   6   8  10  12  14  16  18  20  22  24
   3   6   9  12  15  18  21  24  27  30  33  36
   4   8  12  16  20  24  28  32  36  40  44  48
   5  10  15  20  25  30  35  40  45  50  55  60
   6  12  18  24  30  36  42  48  54  60  66  72
   7  14  21  28  35  42  49  56  63  70  77  84
   8  16  24  32  40  48  56  64  72  80  88  96
   9  18  27  36  45  54  63  72  81  90  99 108
  10  20  30  40  50  60  70  80  90 100 110 120
```

Multiplication.java

break

Ett **break** inuti en loop-sats avbryter loopen.

```
for(int i=1; i<10; i++) {
   if(i==4)
      break;
   System.out.print(i+" ");
}
```

Utskriften blir: **1 2 3**

Om loopar är nästlade avbryts den loop som innehåller break-satsen.

```
for(int rows=1; rows<=2; ++) {
   for(int i=1; i<10; i++) {
      if(i==4)
         break;
      System.out.print(i+" ");
   }
   System.out.println();
}
```

Break1.java

Utskriften blir: **1 2 3**
1 2 3

continue

Ett **continue** inuti en loop-sats avbryter aktuell iteration. Exekveringen "hoppar" till uppdateringsdelen/villkoret för iterationen.

```
for(int i=1; i<10; i++) {
   if(i==4)
      continue;
   System.out.print(i+" ");
}
```

Continue1

Utskriften blir: **1 2 3 5 6 7 8 9**

Satserna **break** och **continue** kan **alltid undvikas**. Detta är oftast lämpligt eftersom kodens läsbarhet kan minska om de används.

Looparna kan ersätta varandra

Loop-strukturerna är olika lämpliga i olika situationer. Men man kan alltid välja fritt mellan for-loopen och while-loopen.

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

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

Ett program om primtal

Ett primtal är ett positivt tal som ej är delbart med något annat heltal än 1 och sig själv.

Exempel på primtal är 2, 3, 5, 7, 11 och 13

Några exempel på tal som inte är primtal är

4 (4/2 = 2) 12 (ex 12/2 = 6) 15 (ex 15/3 = 5)

Ett sätt att konrollera om ***n*** är ett primtal är att testa om ***n*** är delbart med något av talen 2, 3, 4, .., ***n***-1.

Tänkbar algoritm:

- **Deklarera en boolean prime=true;**
- **Inmatning av heltal (med inmatningskontroll)**
- **Testning av heltal. Om primtal så sätt prime till true.**
- **Skriv ut resultat. Använd prime för korrekt utskrift.**

Körresultat:

```
Ange ett tal större än 1: 17
17 är ett primtal

Ange ett heltal större än 1: -4
Ange ett heltal större än 1: 18
18 är inte ett primtal
```

Kontroll av inmatning. Tal <=1 ger uppmaning till ny inmatning (do-loop)