

for-loop

for-each-loop

while-loop

do-while-loop

Iterationer används inom programmering för att upprepa en sekvens av instruktioner ett visst antal gånger eller tills ett visst villkor uppfylls. En iteration kan användas för att bearbeta data, söka efter specifika värden, eller för att utföra en uppgift ett visst antal gånger.

# Iterationer

For-loop använder sig av en räknare

Initierar variabel  
med ett startvärde

Villkor för loop

Förändring av variabelvärde

```
for (int i = 0; i < 10; i++)  
{  
    System.out.println(i);  
}
```

Skriver ut 0-9 i konsolen

# Iterationer

```
String[] names = new String[]{"Clark", "Jenny", "Hulda"};
```

```
for (int i = 0; i < names.length; i++) {  
    if(i == 1)  
        break;  
    System.out.println(names[i]);  
}
```

Skriver ut Clark i konsolen

```
for (int i = 0; i < names.length; i++){  
    if(i == 1)  
        continue;  
    System.out.println(names[i]);  
}
```

Skriver ut Clark och Hulda i konsolen

# Iterationer

`break;`

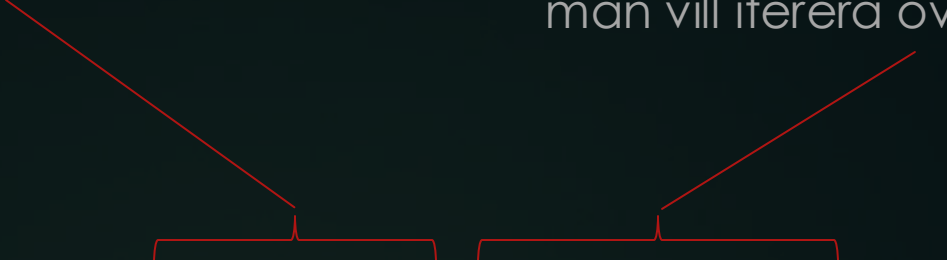
Används för att bryta sig ur loop

`continue;`

Används för att hoppa över iterationen den befinner sig i och resterande kodrader utförs inte

Deklarera variabel av rätt datatyp

Variabelnamnet för den samling  
man vill iterera över elementen i



```
for (String name:nameSuggestions) {  
    System.out.println(name);  
}
```

```
String[] nameSuggestions = new String[]{"Clark", "Jenny", "Hulda"};
```

# Iterationer

Villkoret måste vara sant för att loopen ska starta samt upprepa kodblocket

```
int counter = 10;  
while (counter >= 0) {  
    System.out.println(counter);  
    counter--;  
}  
System.out.println("Gott nytt år!");
```

# Iterationer

Kodblocket körs minst en gång, därefter upprepas kodblocket så länge villkoret är sant

```
int counter = 10;  
do {  
    System.out.println(counter);  
    counter--;  
} while (counter >= 0);  
System.out.println("Gott nytt år!");
```

# Iterationer

# STATISKA METODER



En statisk metod, klassmetod...

- tillhör en klass
- kan anropas direkt utan att instansiera ett objekt
- kan bara komma åt andra statiska medlemmar
- kan ej komma åt icke-statiska medlemmar
- används för att utföra en uppgift som inte kräver information från ett objekt

(Alternativet är en icke-statisk metod)

En instansmetod kan användas för att manipulera data som är specifikt för varje instans av klassen, eller för att få den att exekvera en sekvens av kod.



## STATISKA METODER



Entrypoint

Kommandoradsargument

Returtyp void

Första metoden som körs

main [ENTER]

Måste vara static

Måste vara public

```
public static void main(String[] args)
```

```
1  ▶ public class Main {  
2  ▶     public static void main(String[] args) {  
3  
4      }  
5  }
```

# Utgångspunkt

```

2  ▶ public class Main {
3  ▶     public static void main(String[] args) {
4      speak(sentence: "Java är kul", numOfReps: 2);
5      drawSeparatorLine();
6      String mySentence = createSentenceByColor("gul");
7      // System.out.println(mySentence);
8      speak(mySentence, numOfReps: 2);
9  }
10 private static void drawSeparatorLine(){
11     System.out.println("-----");
12 }
13 private static void speak(String sentence, int numOfReps) {
14     for (int i = 0; i < numOfReps; i++) {
15         System.out.println(sentence);
16     }
17 }
18 @ private static String createSentenceByColor(String color){
19     return "Vinner jag på lotto ska jag bygga en " + color + " villa på en öde ö.";
20 }
21 }

```

```

C:\Users\MichaelGranbäck\.jdk\corretto-17.0.8\bin\java.exe
Java är kul
Java är kul
-----
Vinner jag på lotto ska jag bygga en gul villa på en öde ö.
Vinner jag på lotto ska jag bygga en gul villa på en öde ö.

Process finished with exit code 0

```

# Statiska metoder



# Vi har kikat igenom:

- Datatyper
- Variabler, konstanter
- Array
- Input / Output
- Operatorer
- Selektioner
- Iterationer
- Statisk metod

Denna powerpoint, videos samt  
övningsuppgifter finns redan nu tillgängligt  
i läroplattformen

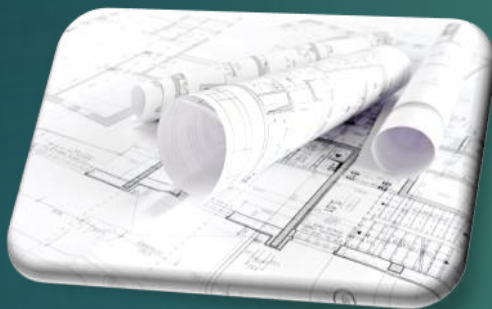


Inför nästa vecka...

13

# Några fördelar med OOP

Klasser



- Samla specifika egenskaper och funktionalitet, samla data och kod som hör ihop på ett och samma ställe.
- Önskad påverkan mellan programmets olika delar minskas
- Återanvända kod, dels genom arv men även mellan olika applikationer

Objekt



- Kan kommunicera med, och påverka, varandra (Vi kan sätta begränsningar)
- Lättare att visualisera och modellera
- Bryter ned komplexa system i mindre byggstenar där varje objekt har en specifik uppgift

# Tack för idag

break;



- ▶ Stay in school
- ▶ May the source be with you!
- ▶ May your code compile on the first try!