Programmeringsteknik och Matlab
Övning 2

Övningsgrupp 2 (Sal Q22/E32)

Johannes Hjorth
hjorth@nada.kth.se
Rum 4538 på plan 5 i D-huset
08 - 790 69 02

Kurshemsida:
http://www.nada.kth.se/kurser/kth/2D1312

Övningsanteckningar:
http://www.nada.kth.se/∼hjorth/teaching/prgi05

Dagens program

- Programmall, vad behöver vi i ett javaprogram?
- Hur skriver vi ut text på skärmen?
- Hur läser vi in från tangentbordet?
- Hur skriver vi `if` och `else` satser?
- Hur fungerar `for`-loopar i java?
- När använder vi `while`-loopar?
- Vad betyder felmeddelandet?

Viktiga datum

2005-09-23
Sista dagen för redovisning av `Lab3` med bonus.

2005-09-27
Inlämning av frivilliga Hemtal 1 i Java.

Ett första exempel

Programmet nedan sparas i filen `Hej.java`, notera att klassnamnet och filnamnet måste matcha.

```
public class Hej {
    public static void main(String[] inParametrar) {
        System.out.println("Titta det fungerar!");
    }
}
```

För att vi ska kunna köra programmet måste vi först översätta det till ett språk datorn kan förstå.

```
>javac Hej.java
```

Efter att ha *kompilerat* koden, så kan vi nu köra vårt nyskrivna program på följande sätt:

```
>java Hej
Titta det fungerar!
```

Dissektion av `Hej.java`

I java kan vi bygga våra program med hjälp av olika byggstenar. Detta underlättar när vi ska göra stora program. Dessa byggstenar kallas i java för klasser.

Första raden i `Hej.java`

```
public class Hej
```

skapar en klass som här börjar vid första måsvingen { och slutar vid sista måsvingen }.

När vi kör vårt program behöver java veta var den ska börja någonstans. Den vill ha en `main`-metod.

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

All kod som står mellan måsvingarna { och } körs rad för rad. I vårt exempel är det bara en rad:

```
System.out.println("Titta det fungerar!");
```

Den raden skriver ut en text på skärmen.

Hur läser vi in från tangentbordet?

Många program som vi använder bygger på att man frågar användaren om något:

```
import java.util.*;

public class Ålder {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("Hej vad heter du?");
        String namn = scan.nextLine();

        System.out.println(namn + " hur gammal är du?");
        int ålder = scan.nextInt();

        System.out.println("Nästa gång fyller du "
                           + (ålder + 1) + " år");
    }

}
```

Så här gjorde vi...

För att få tillgång till den byggsten (klass) som har hand om inläsning behöver vi först importera den:

```
import java.util.*;
```

Detta ger oss tillgång till `Scanner`-komponenten som bland annat används för att läsa in från tangentbordet.

Vi skapar sedan en `Scanner` genom att skriva

```
Scanner scan = new Scanner(System.in);
```

och använder den sedan för att läsa in från tangentbordet:

```
String namn = scan.nextLine();
```

Notera att vi måste ha någonstans att spara den inlästa strängen, därför skapar vi en variabel `namn` som kan innehålla en sträng (`String`).

Vi testkör `Ålder.java`

Vi kompilera och köra `Ålder.java`, då händer följande:

```
>javac Ålder.java
>java Ålder
Hej vad heter du?
Johannes
Johannes hur gammal är du?
26
Nästa gång fyller du 27 år
>
```

Notera att vi gjorde olika saker för att läsa in namn och ålder! Vad är skillnaden?

Titta också på hur vi gör för att sätta ihop utskriften av dels strängar och tal.

Användaren bestämmer!

Nästa steg är att göra olika saker beroende på vad användaren svarar, för att kunna göra det behöver vi `if`-satser.

```
import java.util.*;

public class IfÅlder {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.print("Vilket år är du född: ");
        int svar = scan.nextInt();

        if(svar > 1899) {
            System.out.println("Ungtupp!");
        } else {
            System.out.println("Gammeltant...");
        }
    }
}
```

I det här exemplet skrivs olika svar ut beroende på vilken ålder användaren anger.

Gammelmormor svarar...

Om min gammelmormor hade kompilerat och kört det här exemplet hade det sett ut så här:

```
>javac IfÅlder.java
>java IfÅlder
Vilket år är du född: 1899
Gammeltant...
```

Om istället killen med glasögon längst bak i övningssalen kör programmet:

```
>java IfÅlder
Vilket år är du född: 1985
Ungtupp!
```

Programmet ger alltså olika utskrift beroende på hur användaren svarar. Precis som vi ville.

Vad hade hänt ifall vi tog bort `else`-satsen?

Olika jämförelseoperatorer

I en `if`-sats kan vi jämföra till exempel två tal med varandra. Nedan är `ålder`, `pris`, `längd`, `test` variabler av typen `int` med heltalsvärden.

```
ålder < 65
pris >= 2.1e6
längd == 2.0
test != 0
```

Det går också att kombinera dem:

```
ålder < 65 && pris >= 2.1e6
längd == 2.0 || test != 0
```

Här betyder `&&` logiskt-och medan `||` betyder logiskt-eller.

En `if`-sats kan till exempel se ut så här:

```
if(tid < 45 && risk == 0) {
    System.out.println("Schyst!");
}
```

Samma sak flera gånger, `for`-loopar

Antag att vi vill skriva ut texten "Hej" flera gånger. Ett sätt är att skriva `System.out.println` fem gånger, men det finns ett bättre sätt:

```
public class HejFleraGånger {

    public static void main(String[] inParametrar) {
        for(int i = 0; i < 5; i++) {
            System.out.println("Hej " + i);
        }
    }
}
```

Kompilerar vi och kör koden händer följande:

```
>javac HejFleraGånger.java
>java HejFleraGånger
Hej 0
Hej 1
Hej 2
Hej 3
Hej 4
```

Vad betyder de olika delarna?

En `for`-loop skrivs på följande sätt:

```
for(int i = 0; i < 5; i++)
```

Texten `int i = 0` skapar en loop-variabel som håller räkning på hur många gånger vi kört loopen.

Villkoret `i < 5` testas i varje varv, så länge det är sant fortsätter loopen.

Slutligen är `i++` ett kortare sätt att skriva `i = i + 1`, det vill säga den beskriver hur vi ska öka (eller minska!) loop-variabeln till nästa varv.

Det som ska utföras i loopen skrivs på raden under.

Har vi bara en rad i loopen kan den skrivas direkt efter `for`-raden. Är det flera rader måste vi rama in dem med måsvingar.

`while`-loopar

Ifall vi vet hur många gånger en loop ska köras så använder vi en `for`-loop, men ifall vi inte vet det så passar `while`-loopar bättre.

```
public class EnWhileLoop {
    public static void main(String[] parametrar) {
        while(Math.random() < 0.9) {
            System.out.print("*");
        }
    }
}
```

Det här programmet (som ligger i filen `EnWhileLoop.java`) skriver ut en stjärna för varje varv den kör i `while`-loopen.

```
>javac EnWhileLoop.java
>java EnWhileLoop
*******
>java EnWhileLoop
************
>java EnWhileLoop
******
```

Mera `while`-loopar

Antag att vi skulle vilja översätta en `for`-loop till en `while`-loop istället. Då inför vi en loop-variabel `i`:

```
public class EnTillLoop {
    public static void main(String[] parametrar) {
        int i = 0;

        while(i < 5) {
            System.out.print("*");
        }
    }
}
```

Kompilerar vi och kör filen händer följande:

```
>javac EnTillLoop.java
>java EnTillLoop
************************************************
************************************************
************************************************
****************^C
```

Programmet får tokspel... vad gjorde vi för fel?

Vad gick snett?

Först skapade vi en loop variabel `i` och satte den till noll. Sedan startade vi en `while`-loop som körs så länge villkoret `i < 5` är sant.

Första varvet är `i` noll och villkoret är uppfyllt, andra varvet är `i` fortfarande noll... Problem!

Vi har glömt att öka värdet på loop-räknaren `i`!

Villkoret i `while`-satsen kommer aldrig bli falskt!

Vi har skapat en oändlig loop.

Hur ska vi göra för att vårt program ska fungera?

Felmeddelanden

Antag att vi får följande felmeddelande när vi kompilerar filen `EnAnnanLoop.java`:

```
EnAnnanLoop.java:5: cannot find symbol
symbol  : variable i
location: class EnAnnanLoop
        while(i < 5) {
              ^
1 error
```

Vi har då gjort fel i filen `EnAnnanLoop.java`, och problemet är `cannot find symbol`. Vilken symbol är det strul med, jo; `variable i`.

Med andra ord, den kan inte hitta variabeln `i`. Har vi skapat en variabel `i`? Om inte, gör det!

Felet uppdagades när kompilatorn försökte förstå `while`-slingan där variablen `i` användes för första gången.

Mer felmeddelanden

Vi lägger till raden `int i;` ovanför `while`-loopen och får nu följande felmeddelande när vi kompilerar:

```
EnTillLoop.java:5: variable i might not have been initialized
        while(i < 5) {
              ^
1 error
```

Vi har skapat variabeln `i` men glömt ge den ett startvärde (initiera). Det ordnas lätt:

```
public class EnAnnanLoop {
    public static void main(String[] parametrar) {
        int i = 0; // Vi måste ge i ett startvärde!

        while(i < 5) {
            System.out.print("*");
            i++;
        }
    }
}
```

Inför labben

Vi har nu allt som behövs för att klara av lab 3 och hemtal 1 i Java.

- Vi kan läsa in från tangentbordet och skriva ut på skärmen
- Vi kan läsa och förstå javas felmeddelanden
- Vi vet skillnaden på `print` och `println`
- Vi vet hur `if`/`else`-satser fungerar
- Vi vet hur `for`-loopar skrivs
- Vi kan `while`-slingor samt förstår hur man grupperar rader med måsvingar.

Observera att i stort sett alla rader i Java slutar med semikolon!

Lycka till!