1 Semester SWK Modul 6

Formål: Fokus i dag er Syntaks og afprøvning ved kodning af **For-løkker**

Emner: **Conditional Execution**

- For..Next

Litteratur: Reges: Kap.: 4 Fokus: 4.1, 4.2

Lidt om For løkker

Vi har set på **If..Then** syntaksen. Og med denne introduktion kan vi nu udføre vores kode med **betingelser**. I stedet for alt kaldes i en forudbestemt sekvens, kan vi nu ændre på, hvordan koden eksekveres, afhængigt at de data der arbejdes med!

Noget tilsvarende gør sig gældende med **løkker**. Vi vil i dag se på en simpel opbygning. Når vi skriver koden ved vi ikke nødvendigvis hvor mange gange vi skal gentage en proces!

Gentagelserne er betinget af slutværdien

For..Next kan altså bruges med en ukendt værdi som slutværdi – en variabel der først er givet undervejs i afviklingen af programmet. Men dog er en kendt værdi på kørselstidspunktet!!!

Eksempel: For løkker

```
import java.util.*;  //Inkludér java's package! * Reges kap.3 :-)
public class ForNext {   //koden her er uden underklasse med objekter :-))
  public static void main (String[] args){
    int dice = -1; //Terningen er minus 1  så vi ved om den har været slået inden vi kigger på den :-)
    Random r = new Random();

    System.out.println("Taelleren cntX behoever ikke foelge cntX++ - her taeller vi 2 op ad gangen");
    for (int cntX=2; cntX <= 20; cntX=cntX+2){
      System.out.println("Her er cntX = "+ cntX);
    }
    for (int cntLoop1=1; cntLoop1<=10; cntLoop1++){     //Computeren prøver 10 gange at slå en sekser på 3 slag
      System.out.print(cntLoop1+" gang faar vi: ");
      for (int cntRollDice=1; cntRollDice<=3; cntRollDice++){
        dice = r.nextInt(6)+1; //tilfældigt tal mellem 1 og seks
        System.out.print(" slag#"+cntRollDice +": " +dice);
        if (dice==6 ){
          System.out.println("<--- Her lykkedes det!!! :-)");
          break; //alternativt kunne vi sætte tælleren cntRollDice=3 så loppet slutter ☺
        } else {
          if (cntRollDice>=3 & dice != 6){ //Vi har her en ifsætning hvor BEGGE udsagn skal være opfyldt
            System.out.println("<---- Ingen sekser her :-(");
          }
        }
      }
    }
  }
}
```

Mest simple løkke
Bemærk i stedet for cnt++ som man altid ser hopper vi to ad gangen
(normalt bruger vi cnt++ og ganger bare tæller med de steps vi ønsker)

En nested konstruktion – en løkke i en løkke (bemærk at vi ændrer på den indre løkke for at komme ud – her ville man måske bruge en while opbygning – kommer vi til senere☺)

Loops

- For-løkker der ligger inden i en anden for-løkke som vi så tidligere kaldes **Nestede Loops**
- Bemærk konstruktionen i loops **For <startværdi> til <slutværdi> <tællerjustering>** og herefter kommer så den kode vi vil eksekvere.
- Vi vil her arbejde med **For..next** kodning efter denne model og får føling med kode der går ind i gentagelsesmønstre.

Øvelse#1: Skriv et program Tabel
En lille hurtig øvelse ;)

1. Programmet skal udskrive den lille tabel
2. Benyt nestede løkker.
3. Forklar sidemanden hvad der sker
4. Vi gennemgår et par af jeres løsninger på tavlen !!

Efter pausen kører vi Øvelse #2!!

Øvelse#2: Skriv et program DiceGame

1. Skriv et program der kan slå med en terning
2. Lad hovedprogrammet være spillet og:
3. Opret en klasse Dice der definerer et terningeobjekt med:
   - Konstruktør der initialiserer terningen til -1
   - Getter der henter terningens værdi
   - Metode, shuffle der slår med terningen.
4. Lad hovedprogrammet kører 10 spil hvor der skal rammes en sekser på tre forsøg
5. Udskriv forløbet!

Tip: Husk at dice-klassen skal bruge Java.Util pakken!

Repetition

Vi har nu gennemgået de indledende og mest centrale begreber i OOP vedrørende Klasser, objekter og metoder.

- Hvad er en **Klasse**?
  - En skabelon der definer identifiers og methods som objekter skabes ud fra
- Hvad er et **Objekt**? (uddyb forskellen på Class og Object)
  - Når en identifier skabes med **new** kommandoen skabes et object efter den givne klasses anvisning Objektet er den forekomst der kan arbejdes med, hvor klassen bare er en model for objektet. Fx: Dice dice1 = **new** Dice();
- Hvad er en **Metode**?
  - Metoder er en række handlinger der kan kaldes, hvorefter metoderne så foretager noget med objektet.
- Hvad er en **Konstruktør**? (hvad adskiller Constructor fra Method)
  - En konstruktør kan erklæres og vil så blive udført i forbindelse med 'NEW'kommandoen. Typisk bruges konstruktøren til at initiailsere data med en given startværdi.

Note: På nuværende tidspunkt skal man have en **viden** om, hvad begreberne dækker over, og have en **grundlæggende forståelse** af konceptet med at **oprette og arbejde med klasser og objekter**.
Diskutér emnerne, læs litteraturen og tal med din underviser hvis der er begrebsmæssig forvirring.

Repetition

Vi har nu gennemgået de indledende og mest centrale begreber i OOP vedrørende Klasser, objekter og metoder.

- Hvad er en **private variabel**?
  - En variabel der erklæres private er beskyttet mod direkte tilgang. Kun objektets metoder kan se den
- Hvad menes med **Getter og Setter** ?
  - Metoder der oprettes for at hente (en getter) en variabel, henholdsvis sætte variablen kaldes gettere og settere, men er blot metoder som vi kender allerede.
- Hvad gør **this**?
  - This kommandoen sikrer at vores objekts variabel kan skelnes fra variablen i metoden, da de belejligt hedder det samme. Med **this.var1** henvises til objektes egen variabel, **var1** mens var1 er den der optræder i metodens parameter;
- Hvad betyder **Encapsulation**?
  - Encapsulation, eller indkapsling, er når vi isolerer objektets variable så de ikke kan tilgås direkte. Kommandoen private er en måde at sikre dette.

Note:

På nuværende tidspunkt skal man have en **viden** om, hvad begreberne dækker over, og have **en grundlæggende forståelse** af konceptet med at **oprette - og arbejde med - klasser og objekter**.

Diskutér emnerne, læs litteraturen og tal med din underviser hvis der er begrebsmæssig forvirring.

Repetition

Vi har nu gennemgået Conditional Execution. Hvad lærte vi?

- Hvad er gør **IF..then..else**?
  - IF kommandoen tillader os at styre om en kode skal afvikles ud fra en given betingelse
- Hvad gør en **For..next**?
  - For..next er en kommandoen der tillader os at gentage en sekvens et givent antal gange.
- Hvad gør **Import**?
  - JAVA har en lang række værktøjer som vi kan importere. Det er klasser som kan hjælpe os med metoder. Det kan være en metode der udregner kvadratroden, eller en metode der kan returnerer et tilfældigt tal. Ved at skrive **import java.util.*** får vi adgang til alle de klasser der ligger i java.util;

Note:

På nuværende tidspunkt skal man have en **viden** om, hvad begreberne dækker over, og have **en grundlæggende forståelse** af konceptet med at **operere med betingelser og loops i sin kode og kunne importere andre frameworks**.
Diskutér emnerne, læs litteraturen og tal med din underviser hvis der er begrebsmæssig forvirring.

1 Semester SWK Modul 6

Formål: At forankre vores kompetencer og skrive god kode til vores projekter i ☺

NÆSTE GANG i Modul 7&8

Emner: Vi repeterer **ALT** og vi bruger det vi har lært i vores kodeopgave:

Litteratur:Reges: Reges!!!

*Vi arbejder med **klasser**, **objekter** og **metoder**. Vi bruger **parametre** og **returværdier** og husker **private**-variable og **this**-begrebet så vores metoder kan bruge samme navne i parameterlisterne. Vi initialiserer vores objekter med en **konstruktør** og vores kode rummer nu **If..then** (med logiske operatorer som **&&** og **||**) og løkker af typen **For..Next**.*

KEA - Københavns ErhvervsAkademi // James Hindsgavl Brink // James.brink@gmx.com