Programmeringsteknik för Bio1 och I1
Övning 3

Övningsgrupp 3 (Sal E33)

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/2D1310/

Övningsanteckningar och diagnostiska prov:
http://www.nada.kth.se/∼hjorth/teaching/

Administrativt

Kontrollera att ni har fått er Lab2 inrapporterad i res-systemet genom att skriva

```
>res show prgibio04
```

```
RAPPORTERADE RESULTAT FÖR: ÅÅMMDD-XXXX Efternamn, Förnamn
ANVNAMN: användarnamn STUDIESTATUS: I-03 GRUPP: 3
Moment   Nr    Datum      Resultat   Rapp av

lab      1     040123     G          auto
lab      2     040204     G          hjorth
```

Prata med mig eller föreläsaren om det är något fel!

Kom ihåg de diagnostiska proven! Frågorna som kommer på tentan kommer vara snarlika, men inte exakt likadana.

Jag länkar till proven från min websida

`http://www.nada.kth.se/∼hjorth/teaching`

Kort repetition av metoder

Vi gick igenom hur metoder skrivs och anropas.

```
import java.io.*;

public class MetodAnrop {

    public static void main(String[] inparametrar) {
        int x = 4, y = 7;
        int summa = sum(x,y); // Metodanrop!

        System.out.println(x + " + " + y + " = " + summa);
    }

    public static int sum(int a, int b) {
        return a + b;
    }
}
```

Variabler existerar bara innanför de måsvingar inom vilka de är deklarerade. I exemplet ovan syns inte `x` och `y` i metoden `sum`.

Notera att värdet på parametrarna kopieras vid anrop, de lokala kopiorna inuti `sum` heter `a` och `b`. Ändrar vi värdet på `a` ändras inte värdet på `x`.

Vid metodanrop måste parametrarna ha rätt datatyp. I vårt fall vill `sum` ha två heltal (`int`).

Vad behöver vi veta för Lab4?

Idag ska vi lära oss hur man skapar nya klasser.

Tidigare har vi bara använt oss av en klass i vilken vi har skrivit en `main`-metod. Nu ska vi dessutom skriva en extra klass.

Den nya klassen ska ha både *instansvariabler* och *klassvariabler* samt även *klassmetoder* och *instansmetoder*.

Sedan ska vi använda vår nya klass som mall och skapa objekt utifrån den. Det vill säga vi ska *instansiera* flera objekt av den nya klassen.

Vårt program kommer alltså använda sig av två klasser, en med `main`-metod och en utan någon `main`-metod.

Vad är en klass?

En klass är vårt sätt att i programkoden beskriva hur objekten ska vara. I filen Jacuzzi.java står det,

```
public class Jacuzzi {

    int temperatur; // instansvariabel

    Jacuzzi(int parameter) { // konstruktor
        temperatur = parameter;
    }

    public void setTemperatur(int t) { // instansmetod
        temperatur = t;
    }

    public String toString() { // instansmetod
        return "Jacuzzin har temperatur "  + temperatur + " grader";
    }
}
```

Här har vi en `Jacuzzi` vilken har en instansvariabel `temperatur`, en konstruktor och dessutom två instansmetoder.

Ha bara en klass per fil samt se till att klassnamn och filnamn stämmer överrens.

Vad är en instans?

Instanserna är de objekten som skapas utifrån klassens mall med hjälp av konstruktorn.

```
Jacuzzi palmsprings = new Jacuzzi(41);
```

När du skriver `new` anropar du klassens konstruktor, vilken skapar (konstruerar) en ny instans av klassen.

Det går att skapa fler olika instanser utifrån en och samma klass.

Vi skapar vår första instans

I filen TestJacuzzi.java skapar vi en instans av vår klass `Jacuzzi`, ändrar dess temperatur och skriver ut resultatet.

```
import java.io.*;

public class TestJacuzzi {

    public static void main(String[] argument) {

        Jacuzzi palmsprings = new Jacuzzi(41); // Konstruktorn anropas

        System.out.println(palmsprings);

        palmsprings.setTemperatur(42); // Nu höjer vi temperaturen

        System.out.println(palmsprings);
    }
}
```

Notera hur vi höjer temperaturen. Vår instansmetod ändrar på temperaturen hos instansen. Eftersom det kan finnas många olika instanser, måste vi säga vilken instans den tillhör, här är det `palmsprings`.

Om en klass har en `toString`-metod så är det möjligt att skriva ut den direkt som vi gör ovan.

Vi kompilerar och kör...

När vi kompilerar och sedan kör programmet ser det ut så här

```
>javac TestJacuzzi.java
>java TestJacuzzi
Jacuzzin har temperatur 41 grader
Jacuzzin har temperatur 42 grader
```

Vi har här skapat en instans av vår `Jacuzzi` klass med temperaturen 41 och skrivit ut den.

Därefter har vi höjt dess temperaturen till 42 och skrivit ut instansen igen.

Vi kan instansiera flera objekt

Vi kan enkelt skapa flera instanser av samma klass i vårt testprogram TestFleraJacuzzi.java

```
import java.io.*;

public class TestFleraJacuzzi {

    public static void main(String[] argument) {
        Jacuzzi palmsprings = new Jacuzzi(41);
        Jacuzzi helsinki    = new Jacuzzi(33);
        Jacuzzi pingvin     = new Jacuzzi(4);

        System.out.println("Palmspringshotell: " + palmsprings);
        System.out.println("Helsinki: " + helsinki);
        System.out.println("Sydpolen: " + pingvin);

    }
}
```

De olika instanserna `palmsprings`, `helsinki` och `pingvin` skiljer sig åt eftersom vi anropade deras konstruktorer med olika parametervärden.

Vi kompilerar och kör

```
>javac TestFleraJacuzzi.java
>java TestFleraJacuzzi
Palmspringshotell: Jacuzzin har temperatur 41 grader
Helsinki: Jacuzzin har temperatur 33 grader
Sydpolen: Jacuzzin har temperatur 4 grader
```

Gemensamma variabler

Vi har nu sett hur man skapar både en och flera instanser av en klass.

Vår första klass innehöll en instansvariabel `temperatur` samt två instansmetoder.

Den ena instansmetoden `setTemperatur` ändrade på temperaturen och den andra instansmetoden `toString` returnerade en sträng med temperaturen, vilken vi skrev ut.

I vårt exempel hade våra `Jacuzzi`-instanser olika temperatur. Varje instans har sin egen lokala kopia av instansvariabeln `temperatur`.

I Lab4 behöver vi även kunna skapa variabler som är gemensamma för alla instanser av vår klass, så kallade klassvariabler

Det vill säga, ändrar någon av instanserna på variabeln ser alla instanser det nya värdet.

Hur skapar vi klassvariabler?

Genom att skriva `static` framför en variabel så blir det en klassvariabel. Alla instanser har en gemensam kopia av variabeln. Ändrar en instans på den ser alla ändringen eftersom de utnyttjar samma minnescell.

```
public class LyxJacuzzi {

    int temperatur;      // instansvariabel
    static int lyxskatt = 0; // klassvariabel

    LyxJacuzzi(int parameter) { // konstruktor
        temperatur = parameter;
    }

    public String toString() {
        return "Temperatur "  + temperatur + " grader"
            + " och lyxskatt " + lyxskatt + " kronor";
    }

    public static void nySkatt(int pengar) { // klassmetod
        lyxskatt = pengar;
    }

    public int doppaTån() { // instansmetod
        return temperatur;
    }
}
```

Här är klassvariabeln `lyxskatt` gemensam för alla instanser, vilket markeras av ordet `static`.

Exempel med en klassvariabel

Detta program skapar tre instanser och ändrar sedan på klassvariabeln `lyxskatt`.

```
import java.io.*;

public class TestLyxJacuzzi {

    public static void main(String[] argument) {
        LyxJacuzzi palmsprings = new LyxJacuzzi(41);
        LyxJacuzzi lotta       = new LyxJacuzzi(45);
        LyxJacuzzi hummer      = new LyxJacuzzi(100);

        System.out.println("* Från början ser det ut så här");

        System.out.println("  Palmspringshotell: " + palmsprings);
        System.out.println("  Lotta: " + lotta);
        System.out.println("  Hummergrytan: " + hummer);

        System.out.println("* Lyxskatten sätts till 10000");

        LyxJacuzzi.nySkatt(10000); // Vi ändrar skatten för alla!

        System.out.println("  Palmspringshotell: " + palmsprings);
        System.out.println("  Lotta: " + lotta);
        System.out.println("  Hummergrytan: " + hummer);
    }
}
```

Observera att klassen `TestLyxJacuzzi` ligger i TestLyxJacuzzi.java filen och `LyxJacuzzi` finns inuti LyxJacuzzi.java filen.

Vi kör vårt exempel...

Notera hur skatten hos alla våra instanser ändras när vi anropar `nySkatt`.

```
>javac TestLyxJacuzzi.java
>java TestLyxJacuzzi
* Från början ser det ut så här
  Palmspringshotell: Temperatur 41 grader och lyxskatt 0 kronor
  Lotta: Temperatur 45 grader och lyxskatt 0 kronor
  Hummergrytan: Temperatur 100 grader och lyxskatt 0 kronor
* Lyxskatten sätts till 10000
  Palmspringshotell: Temperatur 41 grader och lyxskatt 10000 kronor
  Lotta: Temperatur 45 grader och lyxskatt 10000 kronor
  Hummergrytan: Temperatur 100 grader och lyxskatt 10000 kronor
```

Anrop av metoder i andra klasser...

Klassmetoder ska ha klassnamnet framför sig

```
LyxJacuzzi.nySkatt(10000);
```

Instansmetoder ska ha en instans framför sig

```
palmsprings.setTemperatur(42);
```

För exempel på hur de ser ut se `main`-metoderna i de två föregående exemplena `TestJacuzzi` och `TestLyxJacuzzi`.

Tillhör metoden den klass eller instans du gör anropet *ifrån* behöver du inte skriva något före metodnamnet vid anrop.

Mer om konstruktorn

Konstruktorn instansierar nya objekt av en klass.

Den ser till att variablerna i den nya instansen har de värden de ska ha.

```
public class Jacuzzi {

    int temperatur; // instansvariabel
    String märke;   // instansvariabel

    Jacuzzi() { // konstruktor A
        temperatur = 0;
        märke = "okänd";
    }

    Jacuzzi(int temp, String märke) { // konstruktor B
        temperatur = temp;
        this.märke = märke;           // exempel på hur this används
    }

}
```

Du kan ha flera olika konstruktorer till samma klass.

När du anropar konstruktorn med `new` väljs den konstruktor vars inparameter typ passar bäst.

```
Jacuzzi hawaii = new Jacuzzi();               // anropar konstruktor A

Jacuzzi palmsprings = new Jacuzzi(41, "IKEA"); // anropar konstruktor B
```

Nu har vi det vi behöver...

Nu har vi allt vi behöver för att klara Lab4.

Kolla er själva...

Ni vet hur man skriver en ny klass?

Ni vet hur man instansierar objekt av den nya klassen med hjälp av `new`?

Ni vet hur man skapar klassvariabler och klassmetoder med hjälp av `static`?

Ni vet hur man anropar både klassmetoder och instansmetoder?

Om något är oklart fråga!