Programmeringsteknik för S1
Övning 2

Övningsgrupp 4

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/2D1311/

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

Dagens program

- Java är objektorienterat
  Hur klasser och instanser hänger ihop
- Kort repetition av metoder
  Parametrar, lokala variabler, synlighet
- Hur skapar vi klasser och instanser?
  Några korta illustrativa exempel
- Mer om konstruktorn
  Vår klass kan ha mer än en konstruktor
- Sammanfattning inför LAB3
  Kolla dig själv!

Objektorienterad programmering

Java är ett objektorienterat programmeringsspråk. Det betyder att vi använder oss av objekt som får representera verkliga eller fiktiva saker.

Dessa objekt innehåller dels data om den saken de representerar, men inte bara det! Vi har även metoder som ligger i våra objekt. Dessa metoder kan manipulera datan som objektet innehåller.

Objekt innehåller alltså både data och metoder.

Vad är vitsen med att ha både metoder och data i objekten?

Den kanske viktigaste fördelen med objekt är att vi kan höja abstraktionsnivån.

Antag att vi ska skriva ett stort program. Vi vill då behöva hålla så lite som möjligt i minnet, då är risken för fel som minst.

Efter att vi en gång skrivit koden för ett objekt behöver inte bry oss hur det fungerar inuti, det räcker att det fungerar.

Det enda som är intressant är alltså hur objektet ser ut utifrån. Det vill säga hur vi skapar objektet och sedan använder det.

Klasser och instanser

Instans är ett annat ord för objekt.

För att kunna skapa och använda en instans behöver vi ha en mall för hur den ska se ut och fungera.

Vilka variabler och metoder innehåller instansen?

Ni har redan sett dessa mallar på föreläsningen.

Vi brukar kalla mallen för en klass.

Kort repetition av metoder

I LAB2 använde vi oss av metoder för att strukturera upp koden och göra den mer övergriplig.

```
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 Lab3?

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 *variabler* och *metoder*.

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.

TIPS: 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`.

TIPS: 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
```

En kopia av variablerna per instans

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`.

Anrop av metoder i andra instanser...

Metoder ska ha en instans framför sig

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

För exempel på hur de ser ut se `main`-metoden i exemplet `TestJacuzzi`.

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

Figur 1: Alice i sin instans av `Jacuzzi`.

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 inparametertyp 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 LAB3.

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 anropar metoder och när man måste skriva en instans framför metodnamnet?

Om något är oklart fråga!