Imperativ programmering

1DL126 3p

Föreläsning 4

Imperativa paradigmer

- Ostrukturerad programmering
- Strukturerad programmering
  - Procedurell programmering
  - Objektorienterad programmering
    - Klassbaserad programmering
    - Prototypbaserad programmering
    - Konceptorienterad programmering
  - Aspektorienterad programmering
  - Attributorienterad programmering
  - ... och så vidare ...

OOP

- Procedurell programmering bryter ner ett problem till en samling datastrukturer och procedurer.
- Objektorienterad programmering bryter ner problemet till objekt som innehåller data och kod.
- Båda metoderna fungerar lika bra - det är bara olika sätt att tänka.

OOP

- Procedurell programmering - Programmet är en lista av instruktioner.
- Objektorienterad programmering - Programmet är ett antal objekt som samarbetar.
- Objekten kan ses som fristående maskiner med var sin given uppgift.
- Objekten kommunicerar med varandra genom att skicka meddelanden.

OOP

- Slår samman flera gamla tekniker.
  - Arv, modularitet, polymorfism, inkapsling
- Formulerades på 60-talet men slog inte igenom förrän på 90-talet.
  - Restriktivt tankesätt - förbjuder fulkod
  - Kräver mer av hårdvaran
  - Kräver avancerade tekniker för kompilering, minneshantering m.m.
  - Hårdvarupris vs. Utvecklingskostnad

OOP

- Simula var det första språket som använde koncepten som bygger OOP. (objekt, klasser, subklasser, virtuella metoder, automatisk minneshantering) Detta som en utbyggnad av Algol.
- Smalltalk var det första språket som kallades objektorienterat.

OOP

- Syftet är att skapa program som är mer flexibla och lätta att underhålla.
- Stark betoning på modularitet.
- Tydlig koppling mellan program och verklighet gör programmet lätt att förstå för de som inte programmerar själva.
- UML - Generera kod.

Terminologi

OOP	Procedurell
metoder	procedurer / funktioner
klasser	moduler
meddelande	anrop
medlem	variabel / funktion
klassvariabel	global variabel

Viktiga termer

- Klass
- Objekt
- Metod
- Meddelande
- Arv
- Inkapsling
- Abstraktion
- Polymorfism

Klass

- En abstrakt beskrivning av en sak.
- Beskriver egenskaper och förmågor.
- Exempel: Klassen Bil
  - Antal dörrar
  - Hastighet
  - Färg
  - Förmågan att gasa / bromsa
  - Förmågan att tuta
  - ...

Klass

- Erbjuder modularitet och struktur.
- Ska vara lättförståelig - andra än programmerare ska kunna förstå vad det är och vad den kan göra.
- Ska vara 'self-contained' - Allt som behövs för dess funktion ska finnas i klassen.

Objekt

- Ett specifikt föremål - en instans - som är byggt enligt en klass.
- Klassen Bil beskriver alla möjliga bilar - WWB829 är en specifik bil - en instans.
- Bilar har en färg - WWB829 har färgen grå.

Metod

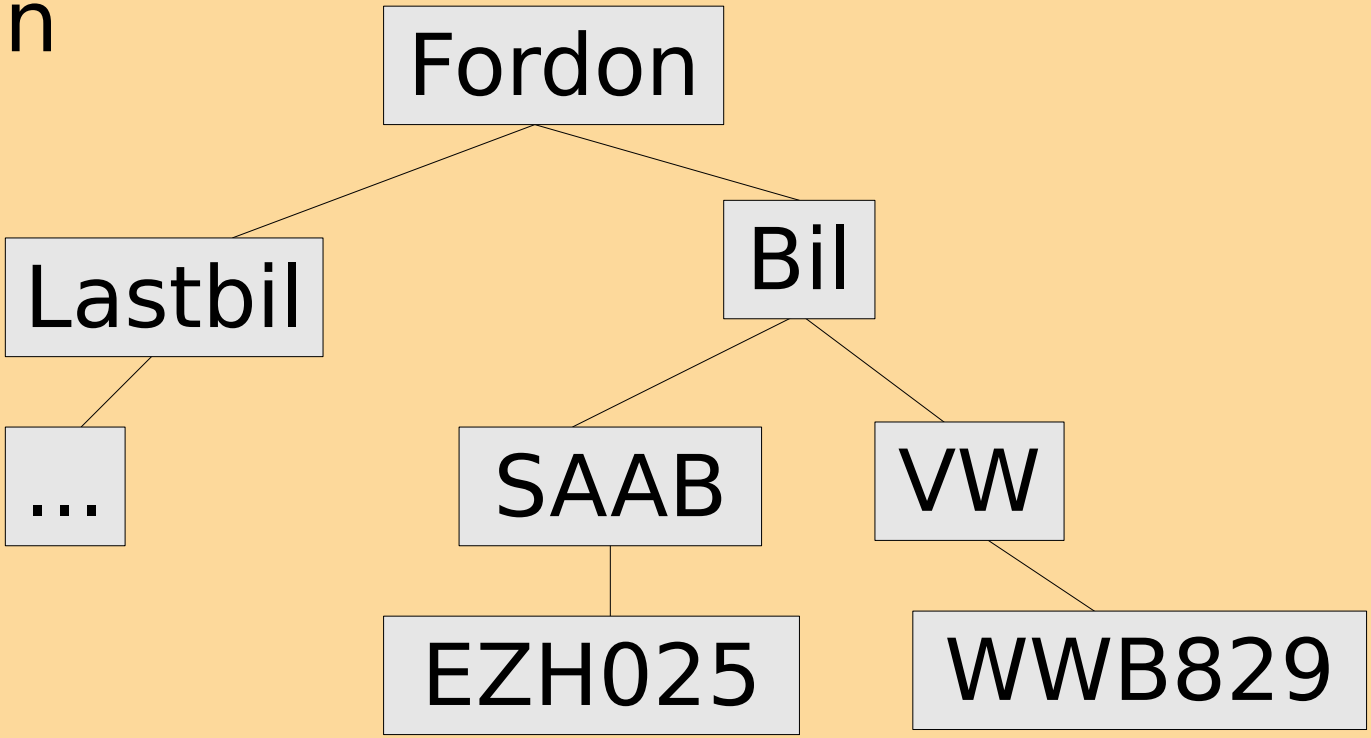
- Ett objekts förmågor implementeras med metoder (jfr. funktioner).
- En bil kan tuta, den har alltså en metod som heter tuta().
- Ett anrop till en metod ska endast påverka ett objekt. Alla bilar kan tuta - men när vi tutar i vår bil ska endast den tuta.

Meddelande

- Används när ett objekt vill att ett annat objekt ska agera, eller för att skicka information från ett objekt till ett annat.
- I Java: Metodanrop.

Arv

- Bygger ut - specialiserar - en klass.
- VW är en speciell typ av bil - klassen VW ärver från klassen Bil.
- 'Är'-relation



Inkapsling

- Dölja implementationen.
- Allt data är privat och åtkomstmetoder används för att sätta eller plocka ut.
- Ger ett rent gränssnitt och gör det lättare att uppdatera koden.
- private, protected, public
  - Java: package
  - C#: internal, VB.NET: Friend
  - C++ & Eiffel: Speca vilka klasser

Abstraktion

- Använd alltid en så abstrakt nivå som möjligt när ni refererar till objekt i koden.
- I de flesta sammanhang kan WWB829 betraktas som en Bil.

  En parkeringsplats...

  ```
  pplats.parkera(Bil b)
  ```

  Hos VW-försäljaren...

  ```
  affären.sälja(VW ny)
  ```

Polymorfism

- Polymorfism = Överlagring
- Möjligheten för en metod att operera på olika typer av indata.
  - heltal + heltal, flyttal + flyttal
  - Bil.tanka(Blyfri drivmedel)
    Bil.tanka(Diesel drivmedel)
    Bil.tanka(RH5 drivmedel)

Java

- Språk med stöd för objektorienterad programmering.
- En (publik) klass per källkodsfil. Filnamn = klassnamn.
- Automatisk minneshantering.

Java

- Oak
- Java
- Java 1.1
- Java 1.2 = Java 2
- Java 2 1.2.x
- Java 2 1.3.x
- Java 2 1.4.x
- Java 2 1.5.0 = Java 2 5.0 = Java 5.0
- Java 6.0

Java

```
public class Tomte
{
    int ålder;
    String namn;
    boolean skägg;

    public Tomte(int å, String n)
    {
        ålder = å;
        namn = n;
        skägg = true;
    }
}
```

Instansvariabler
(= egenskaper)

Java

```
public class Tomte
{
    int ålder;
    String namn;
    boolean skägg;

    public Tomte(int å, String n)
    {
        ålder = å;
        namn = n;
        skägg = true;
    }
}
```

Konstruktor

Java

Java i praktiken

Obligatorisk uppgift

Skriv ner det viktigaste / det du minns bäst från dagens föreläsning