Objects First With Java
A Practical Introduction Using BlueJ

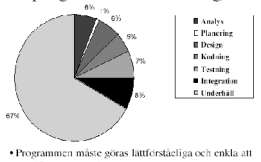
Konstruktion av klasser

Hur skriver man Java-klasser så att de blir lätta att förstå, lätta att underhålla och lätta att återanvända?

Huvudbegrepp

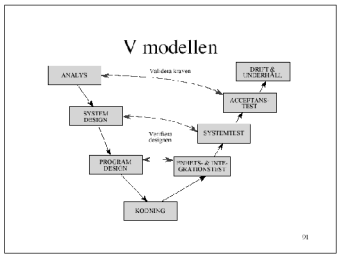
- Ansvarsstyrd design
- Koppling (coupling)
- Samhörighet (cohesion)
- Omkonstruktion (refactoring)

Kodning är bara en liten del i programvaruutvecklingen



• Programmen måste göras lättförståeliga och enkla att modifiera, rätta och bygga ut

V modellen



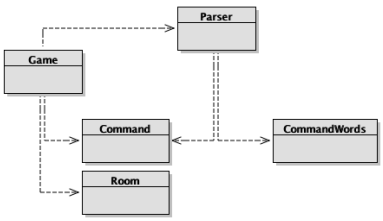
Programvara förändras

- Programvara för verkliga applikationer skrivs inte en gång för alla.
- Programvara utvidgas, korrigeras, underhålls, flyttas, anpassas, ...
- Ofta är många människor involverad i en viss programvara under olika perioder (kan vara 10-tals år).

Underhåll eller överge

- Det finns egentligen bara två möjligheter för ett program:
  - Antingen underhålls det kontinuerligt
  - eller så överges det.
- Program som inte kan underhållas eller är för arbetskrävande att underhålla - överges.

World of Zuul



Vad kännetecknar en bra klass?

- Motsvarar en odelad, väldefinierad abstraktion.
- Avsikten kan dokumenteras kort och tydligt.
- Klassnamnet är ett substantiv (eller adjektiv) som beskriver abstraktionen på ett bra sätt.
- Klassen har ett väl genomtänkt gränssnitt.
- Metoderna i klassen ska ha en stark samhörighet.
- Klassen skall vara löst kopplad till andra klasser.

Kodkvalitet

Två viktiga begrepp när det gäller att bedöma kvaliten hos programkod:

- Koppling (eng. Coupling)
- Samhörighet (eng. Cohesion)

Koppling

- (Samman)koppling handlar om att olika enheter i ett program refererar varandra.
- Om två klasser är skrivna så att båda beror på detaljer i den andra, så säger vi att klasserna är tätt kopplade (eng. tightly coupled).
- Målet är att klasser skall vara löst kopplade (eng. loose coupling).

Lös koppling

Lös koppling gör det lättare att:

- förstå en klass utan att också behöva förstå en annan;
- ändra en klass utan att också behöva ändra en annan klass.

Alltså: Underlättar underhåll.

Samhörighet (Cohesion)

- Samhörighet handlar om antalet olika uppgifter som en enhet ansvarar för.
- Om varje enhet bara ansvarar för en enda logisk uppgift säger vi att dess samhörighetsfaktor är hög (eng. high cohesion.
- Samhörighet är ett mått för klasser och metoder.
- Vi vill ha en hög grad av samhörighet.

Hög grad av samhörighet

En hög grad av samhörighet gör det lättare att:

- förstå vad en klass eller metod gör;
- hitta beteckningar och benämningar som passar uppgiften;
- återanvända klassen eller metoden.

Hög grad av samhörighet

Metoder:

En metod skall utföra en och endast en uppgift.

Klasser:

En klass skall representera en och endast en självständig enhet i en tillämpning.

Samhörighet

Exempel 1

```
public void setNameAndAge(String name, int age);
```

Bättre

```
public void setName(String name);
public void setAge(int age);
```


Samhörighet

Exempel 2

```
/* Anropas när en person fyller år */
public void calculateHoliday()
{
    holiday = new Holiday();
    age = age + 1;
}
```

Bättre

```
public void calculateHoliday()
public void incrementAge();
```

Samhörighet

Exempel 3

```
public void setFirstName(String name){
        firstName = name;
}

public void setLastName(String name){
        lastName = name;
        fullName = firstName + " " + lastName;
}
```

Bättre

```
public void setFirstName(String name){
        firstName = name;
        fullName = firstName + " " + lastName;
}

public void setLastName(String name){
        lastName = name;
        fullName = firstName + " " + lastName;
}
```


Koduppprepning

Koduppprepninging

- är ett tecken på dålig konstruktion,
- gör underhållet svårare,
- ökar sannolikheten för fel när koden ändras.

Ansvarsbaserad konstruktion

- Varje klass skall bara ansvara för uppdateringar av sina egna datastrukturer.
- Den klass som innehåller en datastruktur skall också ansvara för de beräkningar som kan behövas göras på datastrukturen.

Leder till att vi får en löst koppling mellan klasserna.

Ändringar skall vara lokala

- En effekt av låg koppling och ansvarsbaserad konstruktion är att ändringar kan begränsas lokalt.
- När en ändring behövs göras så skall så få klasser som möjligt bli påverkade.

Att tänka efter före

- När en klass konstrueras så skall man försöka att tänka efter vilka ändringar som antagligen kommer att göras i framtiden.
- Målet är att försöka underlätta framtida ändringar.

Omkonstruktion (Refactoring)

- När klasser underhålls lägger man ofta till ny kod.
- Klasser och metoder tenderar att växa
- Då och då behöver man konstruera om klasser och metoder för att bibehålla en hög grad av samhörighet och en låg kopplingsfaktor.

Omkonstruktion och testning

- När man konstruerar om kod, håll isär omkonstruktionen från andra typer av förändringar.
- Genomför omkonstruktionen först utan att förändra funktionaliteten som sådan.
- Kör tester innan och efter omkonstruktionen för att säkerställa att allt fungerar som tidigare.

Konstruktionsfrågor

Frågor som:

- Hur stor skall en klass vara?
- Hur stor skall en metod vara?

- Kan nu besvaras i termer av koppling och samhörighet.

Riktlinjer för programkonstruktion

- En metod är för stor om den utför mer än en logiskt samhörig uppgift.
- En klass är alltför komplex om den representerar mer än en logisk enhet.
- Obs: Detta är riktlinjer - konstruktören måste också använda sitt eget omdöme.

Sammanfattning 1

- Program som används behöver ofta ändras för att möta nya krav.
- Det är viktigt att konstruktionen underlättar sådana ändringar.
- Bra programkod innebär mer än att bara kunna utföra en uppgift på ett bra sätt vid ett tillfälle i tiden.
- Programkod skall vara lätt att förstå och lätt att underhålla.

Sammanfattning 2

- Bra kod undviker upprepningar, har en hög grad av samhörighet och låg koppling.
- Programmeringsstil (kommentarer, namnval, struktur, etc.) är också viktigt.