

Et Simpelt Eksempel

Michael R. Hansen

mrh@imm.dtu.dk

Informatics and Mathematical Modelling
Technical University of Denmark

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 1

Oversigt

- Eksempel: Styklister
 - Uformel beskrivelse
 - Design
 - Unified Modelling Language
 - Objekt model, Klassediagrammer
 - Java program
- Observationer
 - Klassediagrammer: bindeled mellem beskrivelse og program
 - Et lille, men ikke-trivielt, eksempel
 - Udgangspunkt for opgave i databaren i dag.
 - benyttes til at vurdere sværhedsgrad

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 2

Uformel beskrivelse: Styklister

I produktionsomgivelser bliver komplekse produkter samlet ud fra komponenter, som selv kan være simple dele eller komplekse produkter. Et lagerstyringssystem skal

- indeholde et katalog for simple dele,
- holde styr på hvordan produkterne stykkes sammen,
- kunne beregne prisen for et produkt, og
- meget andet.

Eksempel:

- Cykel
 - Stel
 - Hjul (2 styk)
 - Dæk
 - Slange
 - ...
 - ...

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 3

Væsentlige begreber

- | | |
|---------------------|-----------------|
| • komponent | component |
| • simpel del | part |
| • komplekst produkt | assembly |
| • katalogindgang | catalogue entry |
| • katalog | udelades her |
| • prisberegning | cost |

En *begrebsliste*, bestående af begreber med tilhørende definitioner, er et vigtigt del-resultat af en analyse.

Kandidater kan fås via en markering af navneord og verber i teksten

- *find de væsentligste begreber og gør dem så enkle som muligt — kompleksitet af begreber vokser med detaljeringsgraden.*

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 4

Design

Skal give en oversigt over de væsentligste systemkomponenter, herunder deres

- tilstand,
- grænseflade, og
- indbyrdes sammenhænge.

To formål

- vise hvordan problemstillingens væsentlige begreber repræsenteres og behandles
- give et overblik over programmet

abstraktion

UML diagrammer benyttes ofte til at beskrive design af objekt-orienterede programmer, f.eks. Java programmer.

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 5

Simpelt Klassediagram

Klassediagram for katalogindgange:

CatalogueEntry	Navn
-name: String -number: long -cost: double	Attributter
+getName(): String +getNumber(): long +getCost(): double	Operationer

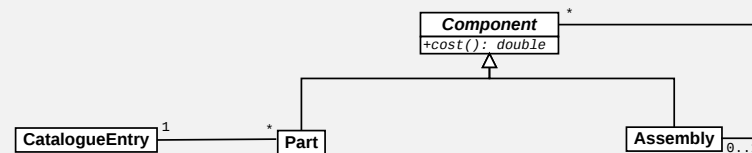
Bemærkninger:

- '-'et ved attributterne angiver at de er *private*
- '+'et ved operationer angiver at de er *public*
- for et objekt, der er en instans af klassen, angiver
 - attributternes værdi objektets *tilstand*, og
 - operationerne objektets *grænseflade*

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 6

Klassers indbyrdes sammenhænge

- en simpel del *er en* komponent *is-a, generalization*
- et komplekst produkt *er en* komponent *is-a, generalization*
- en simpel del *er tilknyttet* en katalogindgang *association*
- et komplekst produkt *er tilknyttet* en liste af komponenter *association*



Component og *cost* er *abstrakte* størrelser (klasse og operation).

Såvel *Part* som *Assembly* er komponenter. En abstrakt klasse specificerer fælles egenskaber ved subclasserne, her en *COST* operation. En abstrakt klasse kan ikke instantieres.

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 7

Implementerings skelet

Implementeringens struktur kan direkte afledes af diagrammet

```
abstract class Component {
    public abstract double cost();
}

class Part extends Component {}

class Assembly extends Component {
    private Collection<Component> components = new ArrayList<Component>();
    ...
}

...
```

Diagrammet giver derfor et overblik over koden.

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 8

Implementering af Part

Prisen for en del findes i katalogindgangen for delen

```
class Part extends Component {
    private CatalogueEntry entry;

    public Part(CatalogueEntry ent)
    {
        entry = ent;
    }

    public double cost()
    {
        return entry.getCost();
    }
}
```

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 9

Implementering af Assembly

Prisen for et komplekst produkt findes vha. komponenternes pris:

```
class Assembly extends Component {

    private Collection<Component> components = new ArrayList<Component>();

    public void add(Component c)
    { components.add(c); }

    public double cost()
    {
        double total = 0.0;
        for (Component c : components)
            total += c.cost();
        return total;
    }
}
```

- **Polymorfi** via den abstrakte klasse Component.
Den rigtige **COST** operation vælges under kørslen.

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 10

Eksempel på kørsel

```
CatalogueEntry ce1 = new CatalogueEntry("p1", 1, 10);
CatalogueEntry ce2 = new CatalogueEntry("p2", 2, 20);
```

```
Part p1 = new Part(ce1);
Part p2 = new Part(ce2);
Part p3 = new Part(ce1);
```

```
Assembly a1 = new Assembly();
Assembly a2 = new Assembly();
```

```
a1.add(p1);
a1.add(p2);
```

```
a2.add(p3);
a2.add(a1);
```

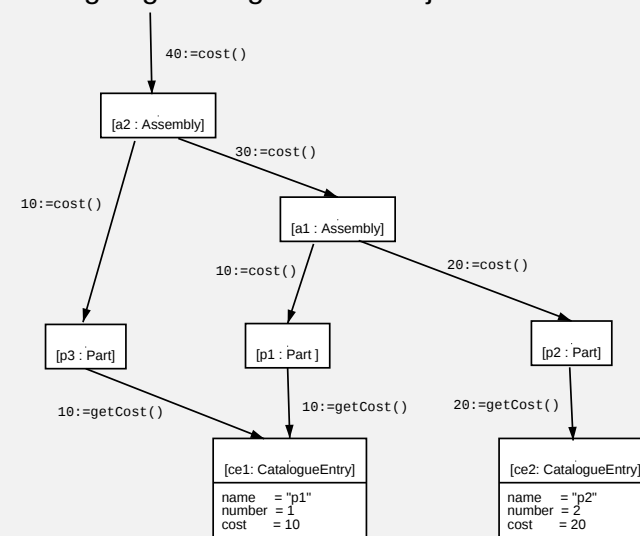
```
System.out.println(a1.cost());
System.out.println(a2.cost());
```

Hvad udskrives?

02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 11

Objekt Model

En samling objekter, hver med en selvstændig tilstand.
Beregninger foregår ved at objekterne udveksler meddelelser.



02161 Software Engineering 1 ©Michael R. Hansen, Spring 2008 – p. 12

Eksempel på JUnit test case

```
public class TestComputeCosts extends TestCase {
    CatalogueEntry stel, daek, slange;

    public void setUp() {
        stel = new CatalogueEntry("Stel",1,700);
        daek = new CatalogueEntry("Dk",2,100);
        slange = new CatalogueEntry("Slange",3,50);
    }

    public void testCostsPart() {
        Component stelPart = new Part(stel);
        assertEquals(700.0,stelPart.cost());
    }

    public void testCostsAssembly() {
        Component daekPart1 = new Part(daek);
        Component slange1 = new Part(slange);
        Assembly hjul1 = new Assembly();
        hjul1.add(daekPart1); hjul1.add(slange1);
        assertEquals(150.0,hjul1.cost());
    }
}
```

02161 Software Engineering 1 ©Anne Haxthausen, Spring 2008 – p. 13

Opsummering

- Uformel beskrivelse
- Design
- Java program
- Test