

# ITU-VOOP-E2001

## Design by Contract

### Error Handling

Lars Thorup

[lars.thorup@aam.org](mailto:lars.thorup@aam.org), Oct 07, 2001

1

## Dagens overblik

- Opsamling fra sidst
- Design by Contract
- Error Handling
- Opsamling på første del af kurset
- Til næste gang

2

## Design by Contract

- Afsløre programfejl tidligt
- Større tryghed ved (gen)brug af fremmed klasse, hvis man kender dens contract
- For metoder:
  - precondition = forudsætning = krav
  - postcondition = følge = løfte
- For klasser:
  - invariant = uforanderligt løfte

3

## Contracts: eksempel – 1

- ```
class TANK
  invariant
    is_full = (0.97 * capacity <= gauge)
    and gauge <= 1.03 * capacity)

  fill is
    -- Fill tank with liquid
  require
    in_valve.open
    out_valve.closed
  ensure
    in_valve.closed
    out_valve.closed
    is_full
end
```

4

## Contracts: eksempel – 2

```
• class Conference
  public void add(Shape shape) {
    int oldCount = 0;
    if(Debug.isOn) {
      invariant();
      Assert.assertNotNull(shape);
      Assert.assertTrue(!_drawing.contains(shape));
      oldCount = _drawing.getShapeCount();
    }
    _drawing.add(shape);
    fireOnAdded(shape);
  }
  if(Debug.isOn) {
    int newCount = _drawing.getShapeCount();
    Assert.assertEquals(oldCount+1, newCount);
    Assert.assertTrue(!_drawing.contains(shape));
    invariant();
  }
}
```

5

## Contracts: issues

- Brug af old i postbetingelser
- Invarianter: betingelser der indgår i alle postbetingelser.
- Nedrivning
  - Prebetingelse må blive svagere (færre krav)
  - Postbetingelser og invarianter må blive stærkere (flere løfter)
  - Følger af reglen om substitutability

6

## Contracts vs autotest

- Contracts er mere generelle
  - og derfor sværere at skrive rigtigt
- Unit Tests er mere specifikke
  - og derfor ikke så sikre
- Contracts er bøvlede at compilere væk i Java.
  - med mindre man bruger jdk1.4 med indbygget assert
- Contracts burde med i javadoc

7

## Error Handling Techniques

- Detektion
- Klassifikation
- Håndtering
- Rapportering

8

## Detektion af fejl

- Teste fejlkoder
- Fange exceptions
- Timeout

9

## Klassifikation af fejl – 1

- Eksterne fejl
  - fx Full Disk, Ukendt Server, Nøgle ikke fundet, Timeout, Out of memory
- Interne fejl
  - fx Null Pointer Exception, Asserts (dem skal der ikke være nogen af)
- Brugerfejl
  - fx Ej 29 dage i februar, Ingen blok markeret, Kan ikke printe –4 kopier, Mangler end–quote

10

## Klassifikation af fejl – 2

- Definer exception hierarki til dette.
- InternalError skal arve fra Runtime Exception så de er unchecked.

11

## Håndtering af fejl – 1

- Forhindre
  - ofte for brugerfejl og visse eksterne fejl
  - altid for interne fejl
  - altid kun et supplement
- Luk ned ved at throw'e en speciel exception der ikke fanges
  - ofte for visse eksterne fejl
  - altid for interne fejl

12

## Håndtering af fejl – 2

- Gentag ved at fange exceptions lokalt
  - for visse eksterne fejl, fx timeout
- Afbryd operationen og rapporter ved at fange exceptions globalt
  - ofte for eksterne fejl og brugerfejl

13

## Rapportering – 1

- Information fra detektionstidspunktet
  - hvad præcis gik galt, fx ukendt server
- Information ind i mellem
  - hvad kan brugeren gøre ved det, fx tjekke sin printer definition
- Information fra håndteringstidspunktet
  - hvilken brugerfunktion kunne ikke udføres, fx print dokument

14

## Rapportering – 2

- Stack Trace
  - nyttig ved interne fejl
- Brug string parameter til Exception-klassen til alle disse ting.

15

## Timeout detektion

- Eksplicit timer thråd (nok stadig nødvendig ved fx database kald)
- Implicit timeout håndtering i network pakken.

16

## Opsamling på første del af kurset

- Process
  - Analyse
  - Design
  - Implementation
  - Gentag og uddyb
  - Nye krav
- Teknik

17

## Proces: analyse

- Materiale
- Domænemodel
- GUI-mockup
- Use-case
- Klassesdiagram

18

## Proces: Design

- Sekvensdiagram
- Klassesdiagram
- Design Patterns
- Interfaces
- Performance
- Fejlhåndtering

19

## Proces: implementation

- Automatiseret test
- Continuous Integration
- Brug scripts
- 
- Gentag og uddyb
- Nye krav

20

## Teknik

- Persistens
- Reflektion
- Trådprogrammering og synkronisering
- Optimering

21

## Til næste gang

- Implementer enkeltbrugerfunktionalitet i tegnekonferencesystemet.
- Anden delaflevering
- Læs til næste gang:
  - Thinking in Java, Bruce Eckel
    - kapitel 11, afsnittet "Object Serialization"
    - kapitel 15, afsnittet om "RMI"
  - "Yet Another java.lang.Class", Chiba & Tatsubori.

lars.thorup@acm.org, Oct 07, 2001

22