Disain

Erik Jõgi
erik.jogi@hansa.ee

Hansabank

Mis on disain?

Miks on disain oluline?

Hansabank

Heaks disaineriks ei ole võimalik koolis õppida

Kogemus, kogemus, kogemus

Hansabank

Hea disain, halb disain

Kas disaini headust on võimalik mõõta?

- Lihtne
- Lühike
- Loogiline

Anton Litvinenko: Software Metrics – 13. Nov

Hansabank

Hea disaini alustala – suhtlemine

- Pidev kommunikatsioon
  - ei piisa, kui alguses räägitakse asjad läbi ja siis läheb igaüks oma nurka nokitsema
- Arendajate vahel:
  - koos asju läbi arutades saavad olulised asjad selgemaks
  - väheneb oht, et sama asja mõistetakse erinevalt
- Tellija ja arendajate vahel:
  - aitab mõista tellija (äri)valdkonda
  - tagab, et tulemus on selline, mis kliendile reaalselt tarvis on
- Ärge kunagi arvake või eeldage midagi kui te kirjutate tarkvara
  - Küsige tellija/teise arendaja käest alati üle!

"Assumption is the mother of all fuckups"

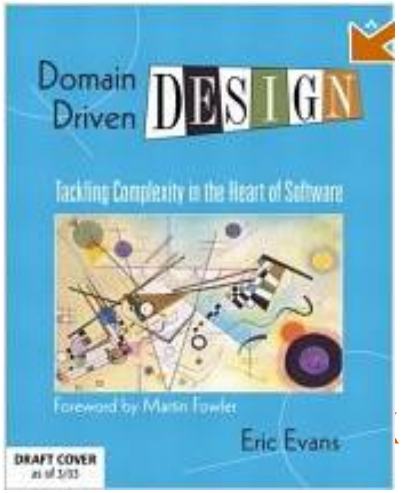
Hansabank

Domain-driven design

- Edu alus on valdkonna tundmine
  - eesmärk ei ole saada valdkonna eksperdiks vaid olla võimeline aru saama valdkonna sisust

- Ühine keel valdkonna ekspertidega
  - objektmudelis kasutatakse vastava valdkonna termineid – ei leiutata uus nimesid
  - vastasel korral kulub palju energiat "tõlkimisele" ja tekib tihti vääritimõistmisi

"Domain-Driven Design:
Tackling Complexity in the Heart of Software"
Eric Evans, 2003



TDD – Test Driven Design

- Unit Testing, Test First Development
- Aitab oluliselt kaasa hea disaini saavutamisele
  - Ask a question of a system by writing a test
  - Respond by writing code to pass the test
    - do the simplest thing possible
  - Refine the response
  - Repeat
- Testid jäävad koodiga kokku
- Alati on võimalik teste käivitada peale muudatuste tegemist ning veenduda, et midagi ei läinud "katki"
- Testid on omamoodi dokumentatsioon, mis näitab, kuidas koodi kasutada

Hansabank

Refactoring

- Olemasoleva koodi strktuuri muutmine ilma välist käitumist muutmata
- Väikeste sammudega
  - iga sammu lõpus peab kood toimima
  - vigade tekke tõenäosus väike
- Oluline on testide olemasolu
- Mitme järjestikuse sammu tulemusena võib saavutada olulise sammu paremuse poole koodi loetavuses

- Peab tegema regulaarselt – mida rohkem seda edasi lükata, seda keerukamaks selle tegemine muutub.

"You know you've achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away“
-- Antoine de Saint-Exupery

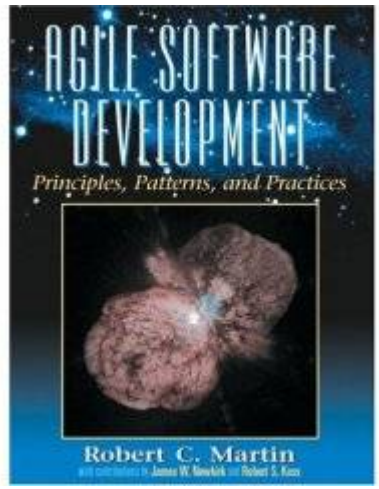
Hansabank

Olulised OO disaini printsiibid

Raamat

"Agile Software Development: Principles, Patterns, and Practices"
Robert Martin, 2003

- **The Open-Closed Principle**
- **The Dependency-Inversion Principle**
- **The Single Responsibility Principle**
- The Interface-Segregation Principle
- The Liskov Substitution Principle



http://blog.objectmentor.com/articles/category/uncle-bobs-blatherings

Olulised OO disaini printsiibid

The Open-Closed Principle

A module should be open for extension but closed for modification

- Avatud laiendusteks:
  Moodulit saab laiendada, kui tekib uusi vajadusi
- Suletud muutusteks:
  Mooduli laiendamiseks ei pea moodulit ennast muutma

- Tuleb kasutada abstraktsiooni
  - interface
  - abstract class/method

Hansabank

Olulised OO disaini printsiibid

The Dependency-Inversion Principle

Depend on abstractions,
do not depend on implementations

- Mooduli realisatsioon ei tohiks sõltuda temast madalama taseme moodulite realisatsioonist vaid nende abstraktsioonidest
- Kuidas saada viide realisatsioonile?
  - Inversion-of-Control (IOC)
  - Dependency Injection (DI)

Hansabank

Olulised OO disaini printsiibid

The Single-Responsibility Principle

A class should have only one reason to change

- Igal klassil peaks olema ainult üks kohustus
  - Erinevad kohustused põhjustavad muutusi erinevates kohtades ja kui need on ühes klassis koos, siis võib ühe suunas muutusi tehes tekkida konflikt teise kohustuse täitmisega
- SRP tulemuseks on rohkem väikseid klasse:
  - Väike klass ei saa olla väga keeruline
  - Väikeste klassidega on lihtsam jälgida teisi OO põhimõtteid
  - Väikseid klasse on lihtsam tesida

Hansabank

Oluline osa koodis: nimed

- Klassid, väljad, muutujad
  - nimi aitab hiljem oluliselt kaasa koodi mõistetavusele
- Pange nimi, mis annab kõige selgemalt edasi sisu
- Nime valikule kulutatud aeg ei ole raisatud aeg
  - küsige kolleegidelt, kas nad saavad sellest nimest sama moodi aru kui teie
  - kasutage vajadusel sõnaraamatut
  - kontrollige (äri)valdkonna eksperdilt terminite tähendusi

Hansabank

Muutused

- Muutused on elu igapäevane osa
- Oleme muutusteks valmis, mitte ei võitle nende vastu
  - ”aga me ju leppisime teisiti kokku?”
- Muutusi sisse viies jälgime, et disain ei kannataks
  - refactoring
- Koodi kustutamisest:

  Aga ma ju nägin palju vaeva selle kirjutamiseks?
  - kulutatud aja eest makstakse raha
  - vaeva nähes (loodetavasti) õppisid sa midagi
  - vähem koodi – vähem potensiaalseid kohti vigadele

Hansabank

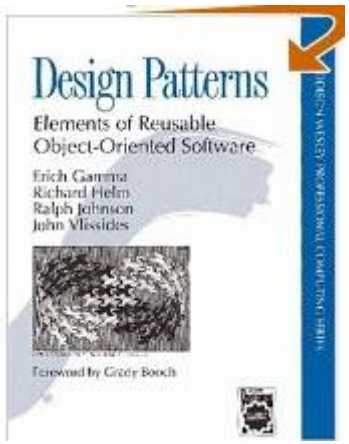
Design Patterns – disaini mustrid

Raamat

"**Design Patterns**: Elements of Reusable Object-Oriented Software"
[Gang of Four Book (**GOF**)] 1994
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

- Tüüplahendused objektorienteeritud disainis olulistele ja tihti esile kerkivatele probleemidele
- Kogenumate arendajate oskuste ja kogemuste süstematiseeritud kataloog
- Lihtsustab arendajate suhtlemist – "me kasutame *Strategy* mustrit"

- Pattern:
  - nimi
  - probleem
  - lahendus
  - mõjud, tagajärjed



Design Patterns

- Jagatud kolme gruppi:
- **Creational patterns** concern the process of object creation
  Factory method, Abstract factory, Builder, Prototype, Singleton
- **Structural patterns** deal with the composition of classes or objects
  Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy
- **Behavioral patterns** characterize the ways in which classes or objects interact and distribute responsibility
  Interpreter, Memento, Template method, Observer, Chain of Responsibility, State, Command, Strategy, Iterator, Visitor, Mediator

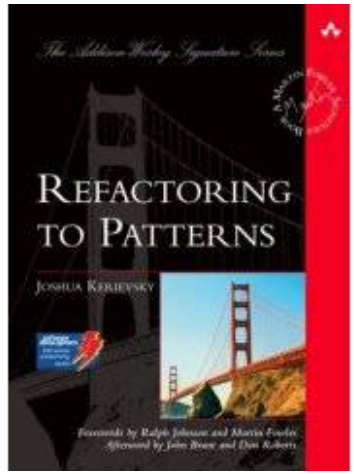
Hansabank

Refactoring to Patterns

- Over-engineering
  - katse liiga palju mustreid kasutada ilma reaalse vajaduseta (YAGNI)
- Under-engineering
  - väga palju koodi (copy-paste), mida saaks mustri abil vähendada

- Tihtipeale tuleb mingi mustri kasutamise vajadus/mõttekus välja alles töö käigus, kui osa koodi on juba kirjutatud
- Mustri kasutuselevõtt aitab suurendada koodi loetavust ning lihtsustab edasiste muudatuste tegemist

"Refactoring to Patterns"
Joshua Kerievsky, 2004



Halva disaini tunnused – design smells

- jäikus (rigidity)
  - raske muutusi teha – isegi lihtsaid, iga muudatus põhjustab vajaduse muuta teisi sõltuvaid osi
- haprus (fragility)
  - läheb katki iga kord kui midagi muuta, tihtipeale kuskil mujal - esmapilgul sõltumatus kohas
- liikumatus (immobility)
  - ei suuda taaskasutada
- viskoossus (viscosity)
  - raske on muutusi teha disaini järgides, lihtsam on *hack*'e teha

Hansabank

Halva disaini tunnused - design smells

- tarbetu keerukus (needless complexity)
  - tuleviku tarbeks ette tehtud asjad
  - YAGNI – *you ain't gonna need it*
- tarbetud kordused (needless repetition)
  - copy-paste liigne kasutamine
- läbipaistmatus (opacity)
  - raskesti arusaadavus

Hansabank

Disain enne koodi kirjutamist?

- Paljud vanemad metoodikad propageerivad seda
  - disainer teeb dokumentatsiooni
  - programmeerijad kirjutavad selle järgi koodi
- Reaalsuses väga hästi ei toimi
  - disainer ei suuda kõiki nüansse ette näha
  - programmeerijad muutuvad laisaks ja ei mõtle ise
- Tulemusena kulutatakse aega asjade peale, mis ei loo tellijale väärtust
- Ära püüa liiga pikalt ette mõelda
- Enne konreetset ülesannet lahendama asumist tasub siiski mõelda ning arutada teistega keerukamad kohad üle
- Disaini ei tohi segi ajada analüüsiga
  - ärivaldkonna tundmaõppimine

Hansabank

Dokumentatsioonist

- Pidage silmas kellele ja miks te seda teete
  - pole kõigile sobivat ühtset viisi
  - mõelge, mis on selle dokumendi tarbijale olulised aspektid

- Joonistest:
  - hea joonis on lihtne joonis
  - enne programmi kirjutamist tehtud joonis on vaid visand, idee
  - milline on parim vahend jooniste tegemiseks?

Hansabank

Kokkuvõte

- Lihtsad lahendused
- Pidev pürgimine hea disaini poole
- Valdkonna tundmine
- Pidev enese täiendamine
  - teiste koodi lugemine (JDK, open source)
  - uute/paremate lahenduste otsimine

Hansabank