Swedbank

Tarkvaratehnika

Disain

Erik Jõgi
erik.jogi@swedbank.ee

Mis on disain?

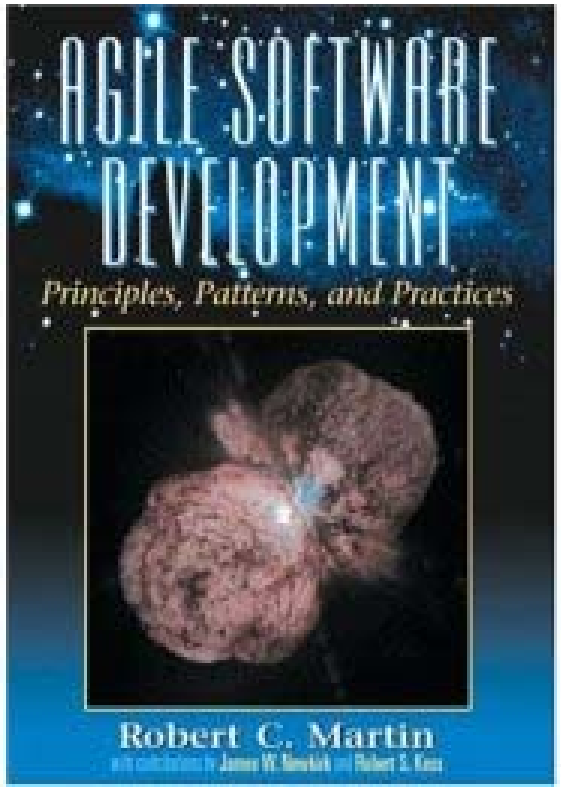
Miks on see oluline?

Heaks disaineriks ei ole võimalik koolis õppida

Kogemus, kogemus, kogemus

Üks väga hea raamat

- Agile Software Development: Principles, Patterns, and Pract
  Robert C. Martin, 2003



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

Hea disain, halb disain?

- Kas disaini headust on võimalik mõõta?

- Lihtne
- Lühike
- Loogiline

- Software metrics
  - http://en.wikipedia.org/wiki/Software_package_metrics
  - http://en.wikipedia.org/wiki/Software_metric

Halva disaini tunnused

Design smell / Code smell

http://en.wikipedia.org/wiki/Code_smell

- Keerukus
  - Tuleviku tarbeks ette tehtud asjad – over-engineering
  - YAGNI – you ain't gonna need it!
- Copy-paste
  - Ka "peaegu copy-paste"
- Rigidity, fragility, immobility, viscosity

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

Be agile!

- Agile = väle, paindlik
  - Loeng 21. okt – Agile Methods
- Muudatused
  - elu igapäevane osa
  - “... aga me ju leppisime kokku...”
  - koodi kustutamine
- Kommunikatsioon
  - teiste arendajatega
  - tellija/kliendiga
  - pidevalt – mitte ainult alguses

TDD – Test Driven Design/Development

- Unit testing, test-first development

- 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, neid on võimalik käivitada peale koodi muutmist ning veenduda, et midagi ei läinud "katki"
- Testid on omamoodi dokumentatsioon, mis näitab, kuidas koodi kasutada

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.

  http://en.wikipedia.org/wiki/Refactoring

A designer knows he has achieved perfection
not when there is nothing left to add,
but when there is nothing left to take away.

Antoine de Saint-Exupery

Interfaces & abstract classes

- Olulised hea disaini alustalad
- Võimaldab ignoreerida implementatsiooni/realisatsiooni detaile

- Väga laialdaselt kasutusel JavaSE APIs
  - Collections API
  - I/O
  - SPIs

- Abstraktsed klassid võimaldavad kirjeldada tegevuste järjekorra jättes vabaks tegevuste implementatsiooni.

Inversion of Control / Dependency Injection

- The Hollywood Principle:

  Don't call us – we'll call you
- Klass ei lähe ise "otsima" omale vajalike ressursse vaid need antakse talle ette.
  - üldjuhul läbi interface'ide
- Selgem disain
- Lihtsam testida
  - Mock objects

- Spring Framework

  http://www.springframework.org

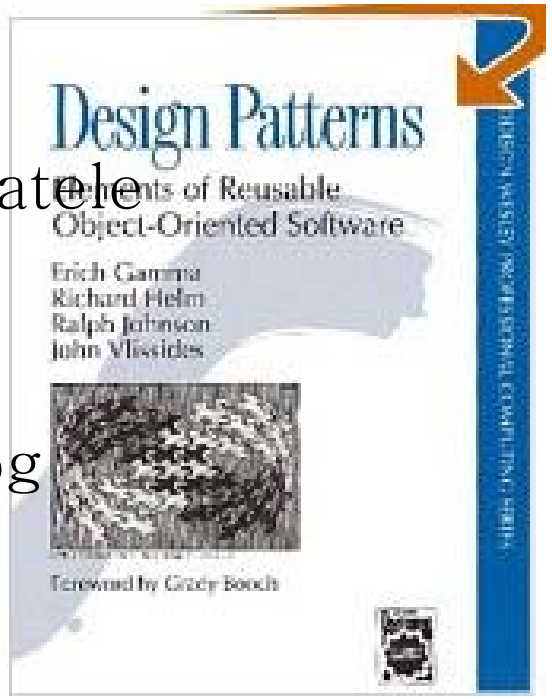
Swedbank

Mutable vs. Immutable

- An **immutable object** is an object whose state cannot be modified after it is created
  - Kõige levinum Java **immutable** objekti näide?
- Lihtsustab programmeerimist
  - Caching
  - Thread-safety
  - Bad code
  - Map keys
- Kiirendab programmi jooksmist

Design Patterns – disaini mustrid

"**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"

Creational patterns

Creational patterns concern the process of object creation

- Factory Method
  - abstract classes use to create objects
- Abstract Factory
- Builder
  - separate construction from its representation
  - used to immutable objects
- Prototype
- Singleton
  - ensure a single instance with global access

Structural patterns

Structural patterns deal with the composition of classes

- Adapter (Wrapper)
  - lets classes work together that couldn't otherwise because of incompatible interfaces
- Bridge
- Composite
  - lets clients treat individual objects and compositions of objects uniformly
- *Null Object**
  - lets to avoid if(o==null) constructs for special cases

Swedbank

Structural patterns

Structural patterns deal with the composition of classes

- Decorator
  - attach additional responsibilities to an object dynamically
- Façade
  - defines a higher-level interface that makes the subsystem easier to use
- Flyweight
- Proxy
  - a surrogate or placeholder for another object to control access to it

Behavioral patterns

Behavioral patterns characterize the ways in which classes or objects interact and distribute responsibility

- Interpreter
- Memento
- Template method
  - define the skeleton of an algorithm in an operation, deferring some steps to subclasses
- Observer (Publish-Subscribe)
  - when one object changes state, all its dependents are notified
- Chain of Responsibility
  - giving more than one object a chance to handle the request

Behavioral patterns

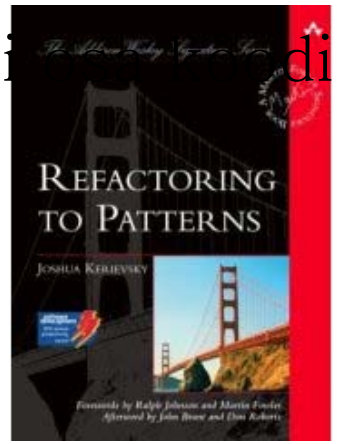
Behavioral patterns characterize the ways in which classes or objects interact and distribute responsibility

- State
  - allow an object to alter its behavior when its internal state changes
- Command
  - encapsulate a request as an object
- Strategy
- Iterator
  - access the elements sequentially without exposing its representation
- Visitor

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 di on juba kirjutatud
- Suurendab koodi loetavust
- Lihtsustab edasiste muudatuste tegemist



"Refactoring to Patterns"

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 konreetse ülesande lahendama asumist tasub siiski mõelda ning arutada teistega keerukamad kohad üle
- Disaini ei tohi segi ajada (äri)analüüsiga
  - ärivaldkonna tundmaõppimine

Kokkuvõte

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