Swedbank

Tarkvaratehnika

Disain

Erik Jõgi
erik.jogi@swedbank.ee

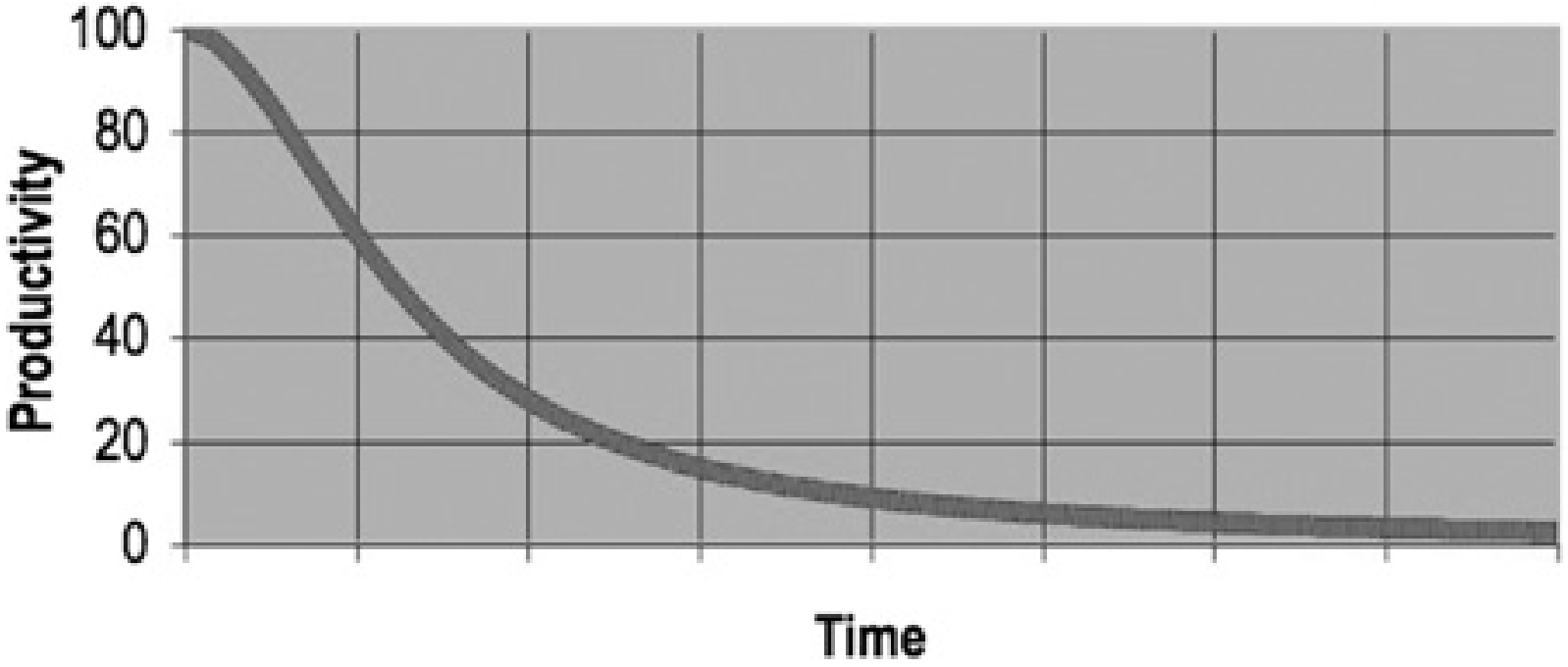
Mis on disain?

Miks on see oluline?

- Enamus koodil, mida me kirjutame, on üllatavalt pikk eluiga
- Me unustame, et lisaks sellele, et me peame koodi kompileeruma ja jooksma saama nüüd kohe, peab see olema kasutatav ka (mõne teise programmeerija poolt) mitme kuu või aasta pärast

Bad Code

- Mis juhtub, kui mitte hoolida koodi disainist?



Heaks disaineriks ei ole kahjuks võimalik koolis õppida.

Kogemus, kogemus, kogemus

Vaatame erinevaid disaini aspekte

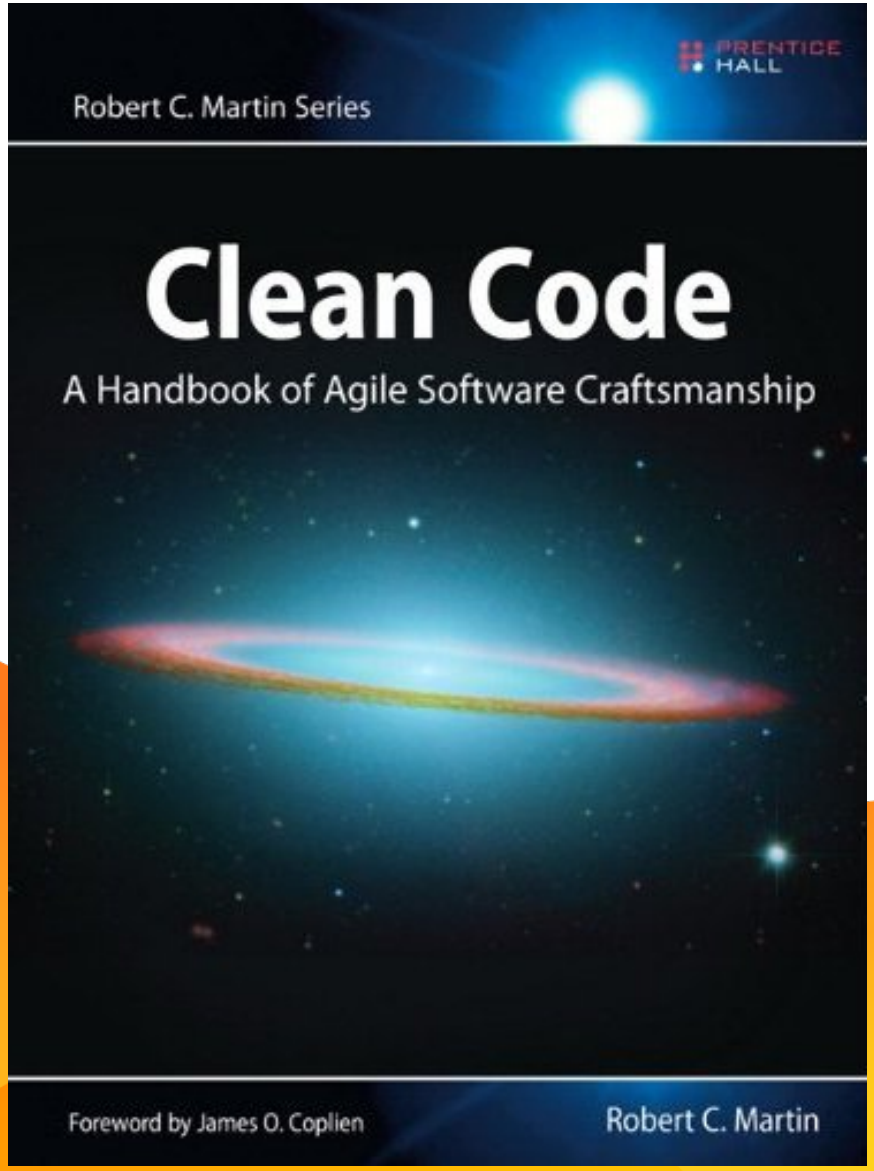
Liigume madalamalt tasemelt kõrgemale – muutujatest ja funktsioonidest klasside ja nende vaheliste seoste poole

Swedbank

Clean Code

Clean Code.
A Handbook of Agile
Software Craftmanship

Robert C. Martin, 2008



Swedbank

Clean Code

You know you are working on clean code when each routine you read turns out to be pretty much what you expected.

You can call it beautiful code when the code also makes it look like the language was made for the problem.

Ward Cunningham

inventor of Wiki and Fit, co-inventor of Extreme Programming

Swedbank

Clean Code

Naming
Nimed

Swedbank

Mõistetavad nimed

Mida see meetod teeb?

```
public boolean a(String b, char c, char d, String e, String f, String g) {
  A h = new A(b, c, d, e, f, g);
  List i = j.n(h.o());
  if (i != null) {
    for (Iterator k = i.iterator(); k.hasNext();) {
      A l = (A) k.next();
      if (m(l, f, g)) {
        return l.n();
      }
    }
  }
  return false;
}
```

Swedbank

Mõistetavad nimed

Aga see?

```
public boolean isServiceAllowed(String serviceCode, char accountType, char accountSubtype,
                                String accountState, String clientType, String source) {
  AllowedServiceRule ruleToFind =
      new AllowedServiceRule(serviceCode, accountType, accountSubtype,
                             accountState, clientType, source);
  List matchingRules = accountServiceRulesCache.findAccountServiceRules(ruleToFind.getKey());
  if (matchingRules != null) {
    for (Iterator iterator = matchingRules.iterator(); iterator.hasNext();) {
      AllowedServiceRule rule = (AllowedServiceRule) iterator.next();
      if (matches(rule, clientType, source)) {
        return rule.isAllowed();
      }
    }
  }
  return false;
}
```

Swedbank

Nimi peab aitama mõista sisu ja konteksti

Stuff, Data, X, list, number, hp, a1, b2,
theSet, aList, someThings

vs.

AccountAlias, FixedPaymentCard,
flaggedCustomers, maxTransactionCount,
queryIntervalSeconds, agreementsByType,
findCustomerByRegistrationCode

Correct English Only, Please!

- Sõna-sõnalt tõlge ei pruugi anda õiget ingliskeelset vastet
- Kasutage õiged antud valdkonna termineid, uurige valdkonna ekspertidelt, mis on õiged terminid
- Grammatika ja õigekiri!

  Sõnastikud, õigekirjakontrollijad, Google

- Proovige, kas koodi saab lugeda nagu lauseid tekstis?

Kasutage ühte terminit sama mõiste jaoks igal pool

- Customer, Client, Party, Counterparty
- state, status
- Contract, Agreement
- get, load, fetch, find, retrieve
- regCode, regNumber, personalCode

- … ja kasutage teise mõiste jaoks erinevat terminit

Swedbank

Vältige müra!

- Vältige kodeeringuid: Hungarian Notation, klassi väljadel kindel prefiks, jne...
- Ära lisa liigset konteksti
  - ee.bank.agreement package'is ei pea algama kõik klassid Agreement...
  - SomeContract klassis ei pea olema muutujad contractType, contractNumber, contractEndDate
- Ära ürita olla naljakas
  - killEmAll(), findThosePoorCustomers(), BombSquad

Swedbank

Clean Code

Functions
Funktsioonid

Swedbank

Funktsioonid

Hea funktsioon on:

- lühike
- vähe trepiastmeid {indentation}
- tegeleb ainult ühe asjaga (üks abstraktsiooni tase)

Functions should do one thing.
They should do it well.
They should do it only.

Funktsiooni argumendid

- “Less is more” – mida vähem, seda parem

- Ideaalne funktsioon ei võta ühtegi argumenti
- Ühe ja kahe argumendiga on ka OK.
- Kolm on üldjuhul juba liiga palju ja üle kolme ei tohiks mingil juhul olla

Swedbank

Kuidas argumentide arvu vähendada?

- Selle asemel, et järjest edasi anda ühte argumenti, tehke klassi tasemel väli
- Kui funktsioon vajab rohkem kui 2 argumenti, siis võib-olla peaks need argumendid omaette klassi panema?

- Vältige üksikuid boolean argumente (lippe)

savePayment(boolean validate)

vs.

savePayment(), validateAndSavePayment()

Swedbank

Vältige kõrvalmõjusid

Funktsioon, mis on definitsiooni (nime) järgi read-only, ei tohi muuta olekut

- getAmount()
- isConfirmed()
- checkPassword(String)

Kui on ikkagi vaja olekut muuta, siis pange funktsioonile sobivam nimi

- checkPasswordAndLoadSettings(String)

Swedbank

Clean Code

Comments
Kommentaarid

Swedbank

Comments Are Evil

Comments lie:

- They are not tied to a specific piece of code
- They cannot be refactored
- They cannot be unit tested
- They get out of date very easily

Truth can only be found in the code

Don't use a comment when you can use a function or a variable

Swedbank

Millal võib kommentaare kasutada?

The proper use of comments is to compensate
for our failure to express ourselves in code

- Legal comments
- Warning of consequences
- TODO comments
- Amplification
- javadocs in public APIs

Swedbank

The Boy Scout Rule:

Leave the campground cleaner than you found it.

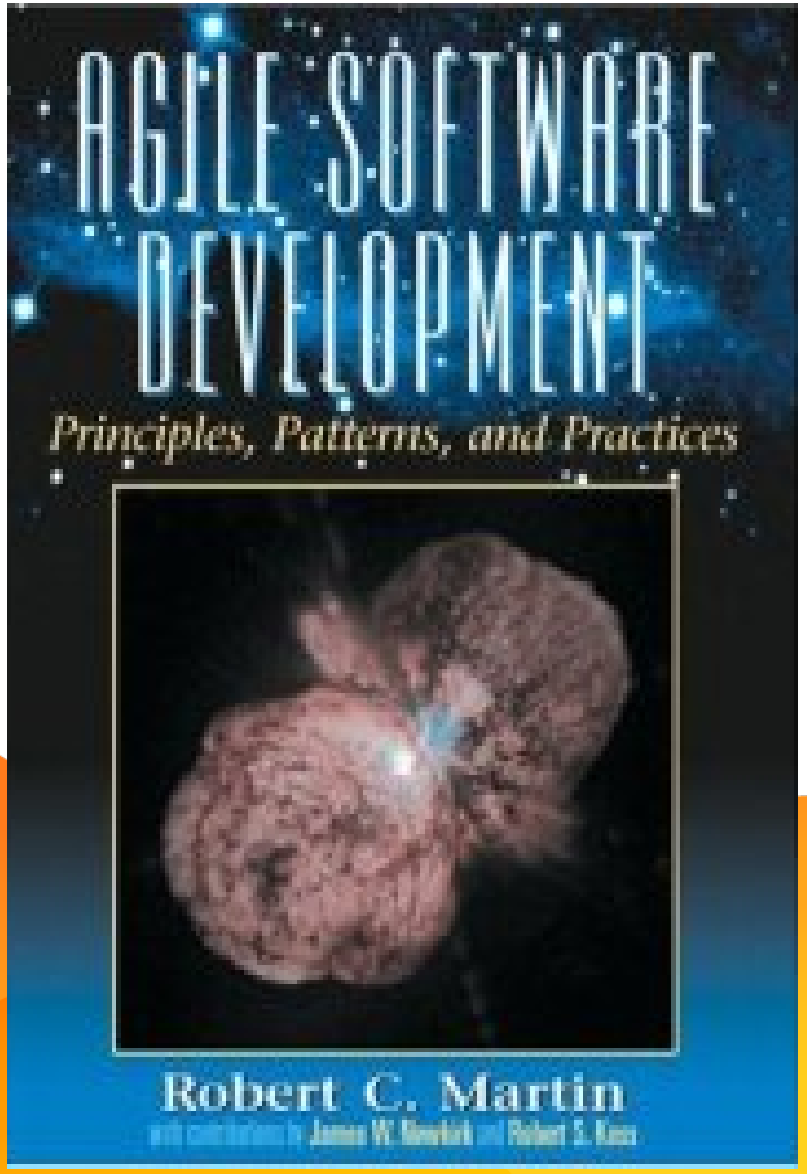
- Can you imagine working on a project where code simply got better as time passed?
- Do you believe that any other option is professional?

Swedbank

Object-Oriented Design

Agile Software Development:
Principles, Patterns,
and Practices

Robert C. Martin, 2003



Swedbank

Interfaces and Abstract Classes

- 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.

- Program to interfaces

Swedbank

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.
- Selgem disain
- Lihtsam testida
  - Mock objects
- Spring Framework – www.springframework.org

Swedbank

Composition Over Inheritance

- Inheritance is for “is-a” relationships:
  - Apple is a Fruit
  - Button is a Jcomponent
- When the same interface is most logical

- Composition is for “consists-of”, “contains”, “uses”, “has” relationships:
  - SalesSystemUI uses JFrame
  - Customer conains an Address

- Composition means decoupling – independence of each other's changes
- Java does not allow multiple inheritance

Swedbank

Immutable Objects

- 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

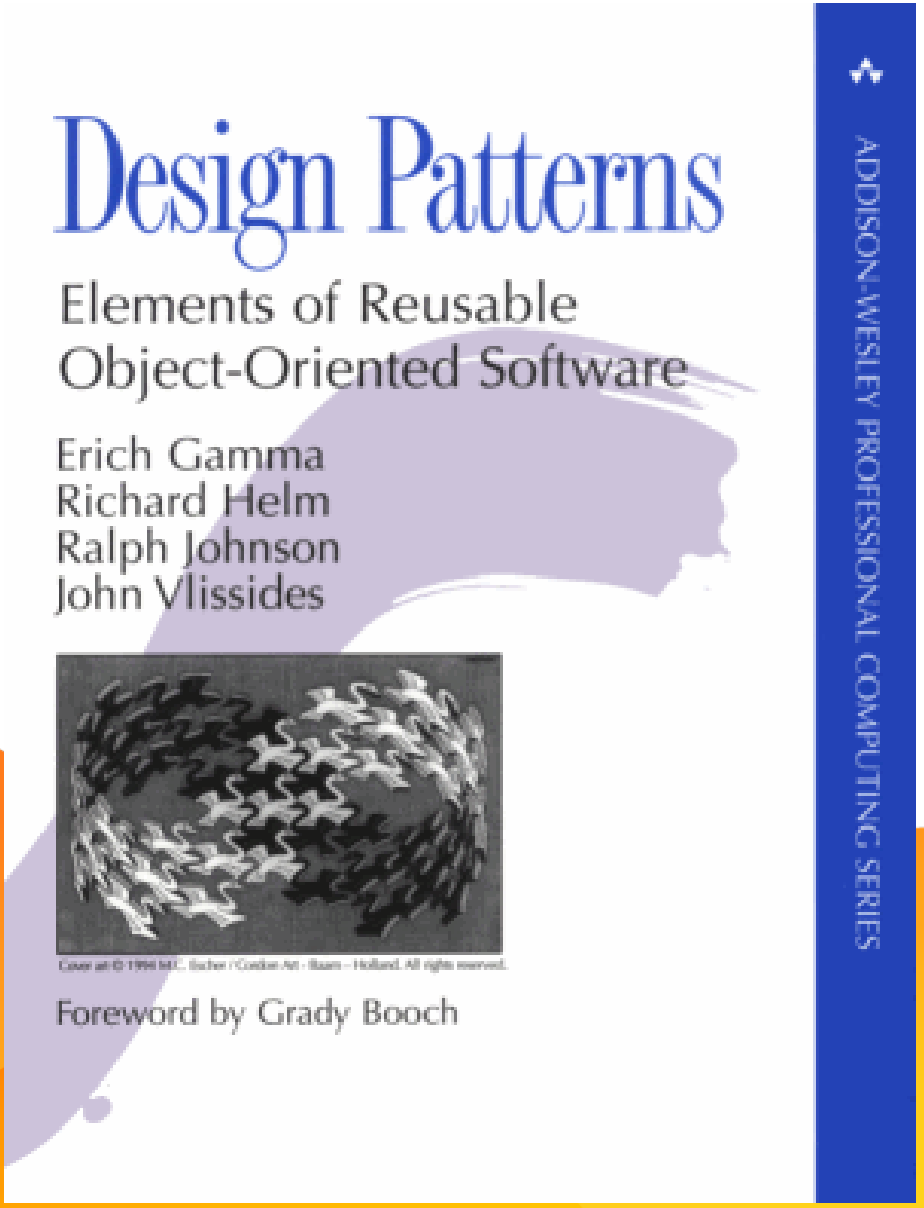
- Võimaldab kompilaatoril/virtuaalmasinal koodi optimeerida

Swedbank

Design Patterns

Design Patterns
Elements of Reusable
Object-Oriented Software"
[Gang of Four Book (**GOF**)]

Erich Gamma, Richard Helm,
Ralph Johnson, John Vlissides
1994



Design Patterns – disaini mustrid

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

Swedbank

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

Swedbank

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

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

Swedbank

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

Swedbank

Behavioral patterns

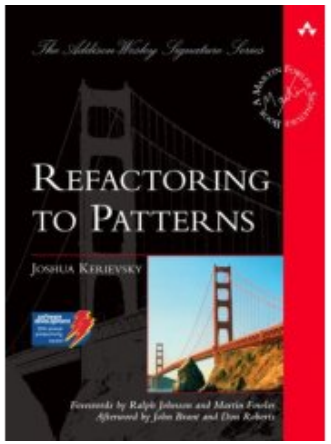
Behavioral patterns characterize the ways in which classes or objects interact and distribute responsibility

- 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
- Mediator

Refactoring to Patterns

- Over-engineering - katse liiga palju mustreid kasutada ilma reaalse vajaduseta
  - YAGNI – You Ain't Gonna Need It
- 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
- Suurendab koodi loetavust
- Lihtsustab edasiste muudatuste tegemist



"Refactoring to Patterns"
Joshua Kerievsky, 2004

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

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

- Clean Code
- The Boy Scout Rule
- Pidev pürgimine hea disaini poole
- Valdkonna tundmine
- Pidev enese täiendamine
  - teiste koodi lugemine (JDK, open source)
  - uute/paremate lahenduste otsimine