Legacy Code Renovation

Impressions and Guidance

by Pascal von Arx

Motivation

Why do systems in generally degenerate to spaghetti Code?

Is it because people are lazy?
Is it because people are incapable?
Is it because people like it?

Is clean code just to boring?

There are to ways to adress systems degeneration

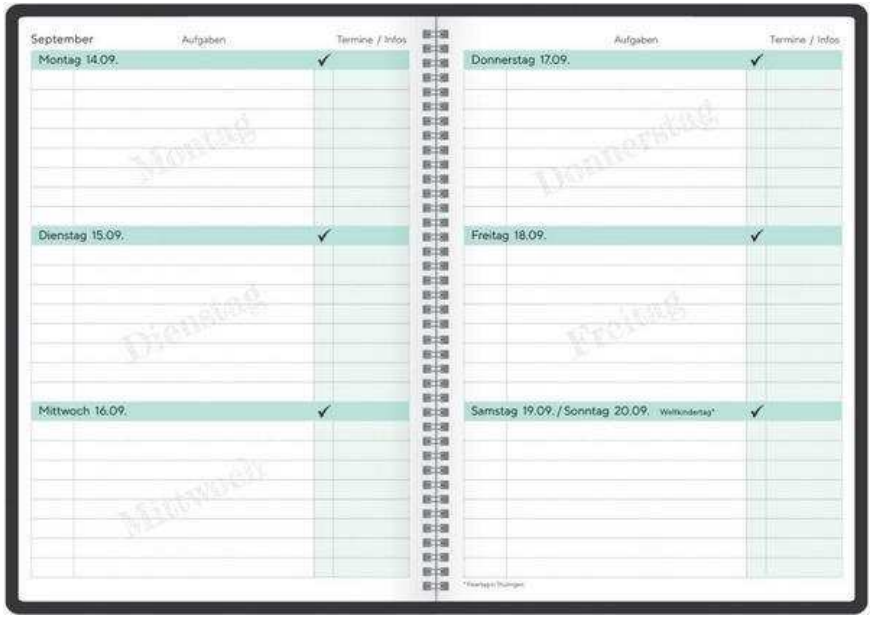
1. Rebuild -> rarely possible
2. Refactor -> lets at least refactor then

   Refactoring is about structural improvement without behavioural change
   For me it is about making something degenerated beautiful again.

- Refresher of the course content
- Give some alternative guidance for code refactoring
- Don’t do a real world code refactoring demo. There is no way for me to do it in a short presentation.

Agenda

- Consinder when changing (legacy) systems
- Questions for making legacy code testable
- Scenarios and pattern to apply
- Conclusion



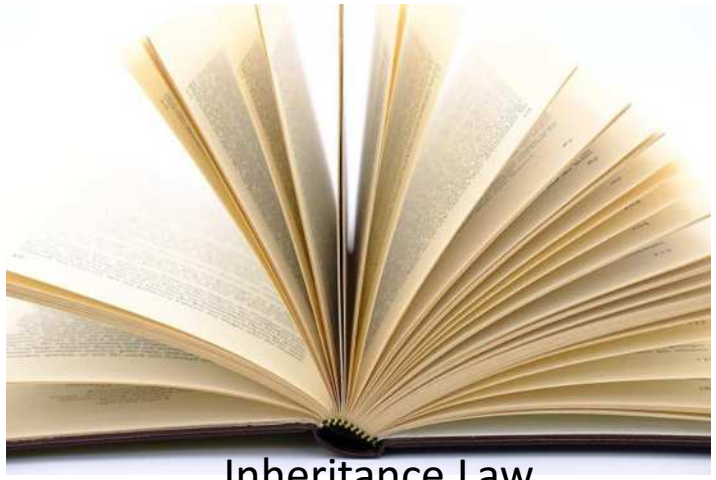
Consider when changing (legacy) systems

- Never extend/refactor without tests!

=> We need to make a system testable first

- We need a high test-coverage

=> Use combinatorial tests to get there



Inheritance Law

- We normally don’t know the specified behaviour of the system

=> Use the behaviour of the system in production as the baseline for approval testing

- We dont know the critical boundaries in the system for developping good testdata

=> Use mutation testing to cover them

Questions for making legacy code testable

1. Can I overview the class i need to test?
2. Do I need to instantiate the class for my tests?
3. Am I able to instantiate the class i need to test?
4. Am I able to test the method under test?
    a) Am I able to instantiate the method parameters?
    b) Do I need more flexibility for objects instatiated inside of the constructor?
    c) Do I need more flexibility for objects instatiated inside of the method under test?
    d) Do I need to control objects created inside of the method under test?
    e) Do I need to manipulate the result of a method called by the method under test?
    f) Do I need to change the behaviour of a static method called in the method under test?

Test need to match the system under test. It is never the other way round.

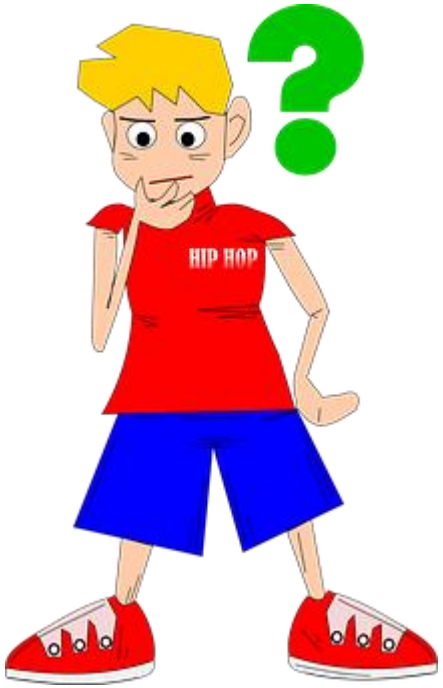
Can I overview the class i neet to test?

If not, then make it smaller by
extracting long methods
into break out objects

Do I need to instantiate the class for my tests?

If not, then test against static methods by

exposing static methods



Am I able to instantiate the class i need to test? (1)

- If needed dependencies that are easy to fake are accessed directly, then
  - introduce **Getters** and override the Getters in a subclass to provide fakes or
  - set a **superseeded instance variable** or
  - provide it as a new Constructor argument or
  - **Introcuce static setter** in case of singleton

Am I able to instantiate the class i need to test? (2)

- If unneeded dependencies are the cause then **pull up the feature** into a super class that is free of the problematic dependencies (or **push down problematic dependencies**)

- If dependencies are instantiated in the constructor then
  - apply **extract and override factory method** or
  - **Parametrize constructor**

Am I able to instantiate the class i need to test? (3)

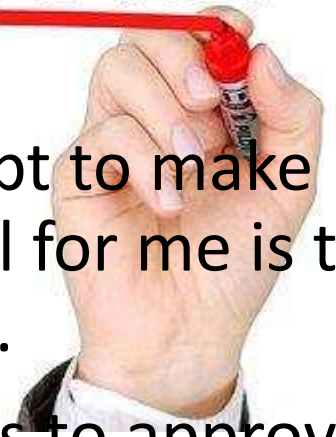
- If the constructor fails because it instantiates uneeded dependencies for my test
  - Apply **extract an override getter** with lazy instantiation

- If it stays impossible to instantiate the class then try to extract some algorithmig logic into a static method with **primitive parameters**

Am I able to test the method under test?

- Am I able to instantiate the method parameters?
  - If not, then **adapt the parameters** to a simpler interface or apply **extract implementor**
- Do I need more flexibility for objects instatiated inside of the constructor?
  - Apply **extract and override factory method**
- Do I need more flexibility for objects instatiated inside of the method under test?
  - Apply **extract and override call**
- Do I need to manipulate the result of a method called by the method under test?
  - Apply **subclass and override method**
- Do I need to change the behaviour of a static method called in my method under test?
  - **Introduce an instance delegator (method)**
- Do I need to control objects created inside of the method under test?
  - Apply **parametrize method**

CONCLUSION



- When I attempt to make legacy Code more testable, the most important tool for me is the cdi-container for breaking some dependencies.
- When it comes to approval testing, is getting rather difficult, because test inputs are far more complex than in the course. I often see deep data structures in input parameters which are very hard to mutate.

- I’m still looking for the silver bullet, but i didn’t find it so far.

Thanks for your attention