Outside-in Test Driven Development (London School TDD)

Software Quality Days 2019

Peter Kofler, 'Code Cop'
@codecopkofler
www.code-cop.org

Peter Kofler

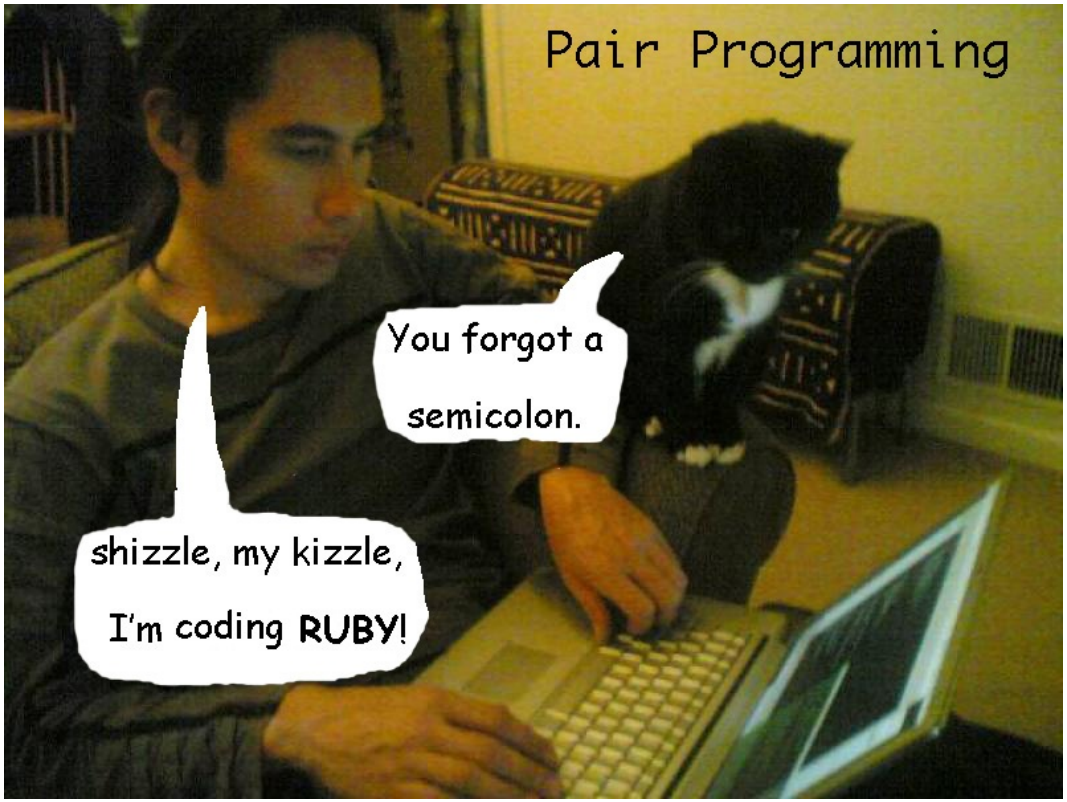
- Ph.D. (Appl. Math.)
- Professional Software Developer for 20 years
- “fanatic about code quality”
- Independent Code Quality Coach

I help development teams with

- Professionalism
- Quality and Productivity
- Continuous Improvement

Mentoring

- Pair Programming
- Programming Workshops
- Deliberate Practice, e.g. Coding Dojos



Developing Quality Software Developers

Agenda

- Recap Classic TDD
- Introduction
  Outside-in TDD
- Coding Exercise
- “Bank OCR“
- Retrospective

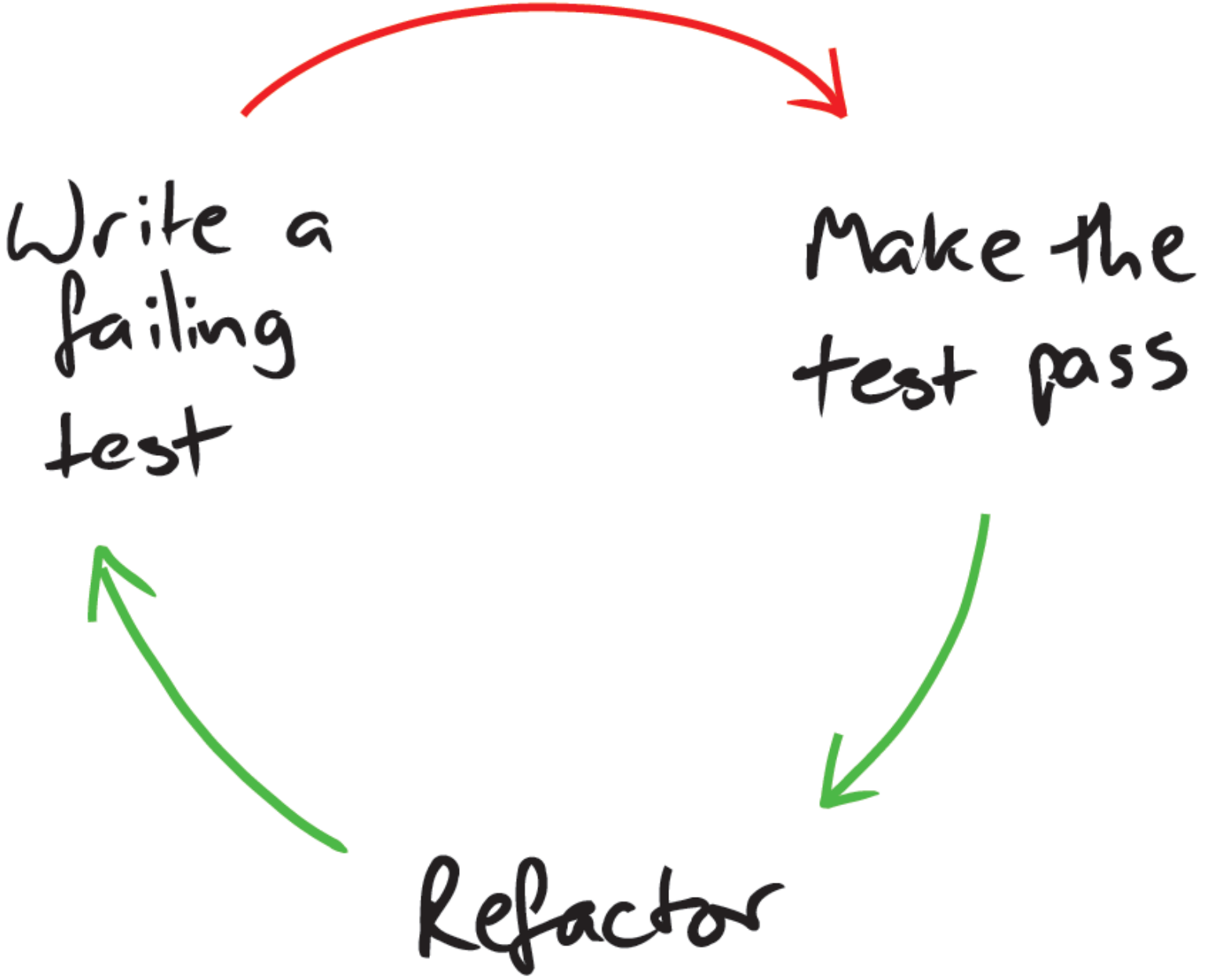
Test Driven Development

Your experience with TDD?

- When and how are you applying TDD?
- What are you using every day?
- Any problems?

Test-Driven Development is

- a programming practice in which **all** production code is written in response to a failing test.
- a practice for designing and coding software applications.
- not a replacement for testing.



Nathaniel Pryce, http://www.doc.ic.ac.uk/~np2/teaching/

TDD Cycle

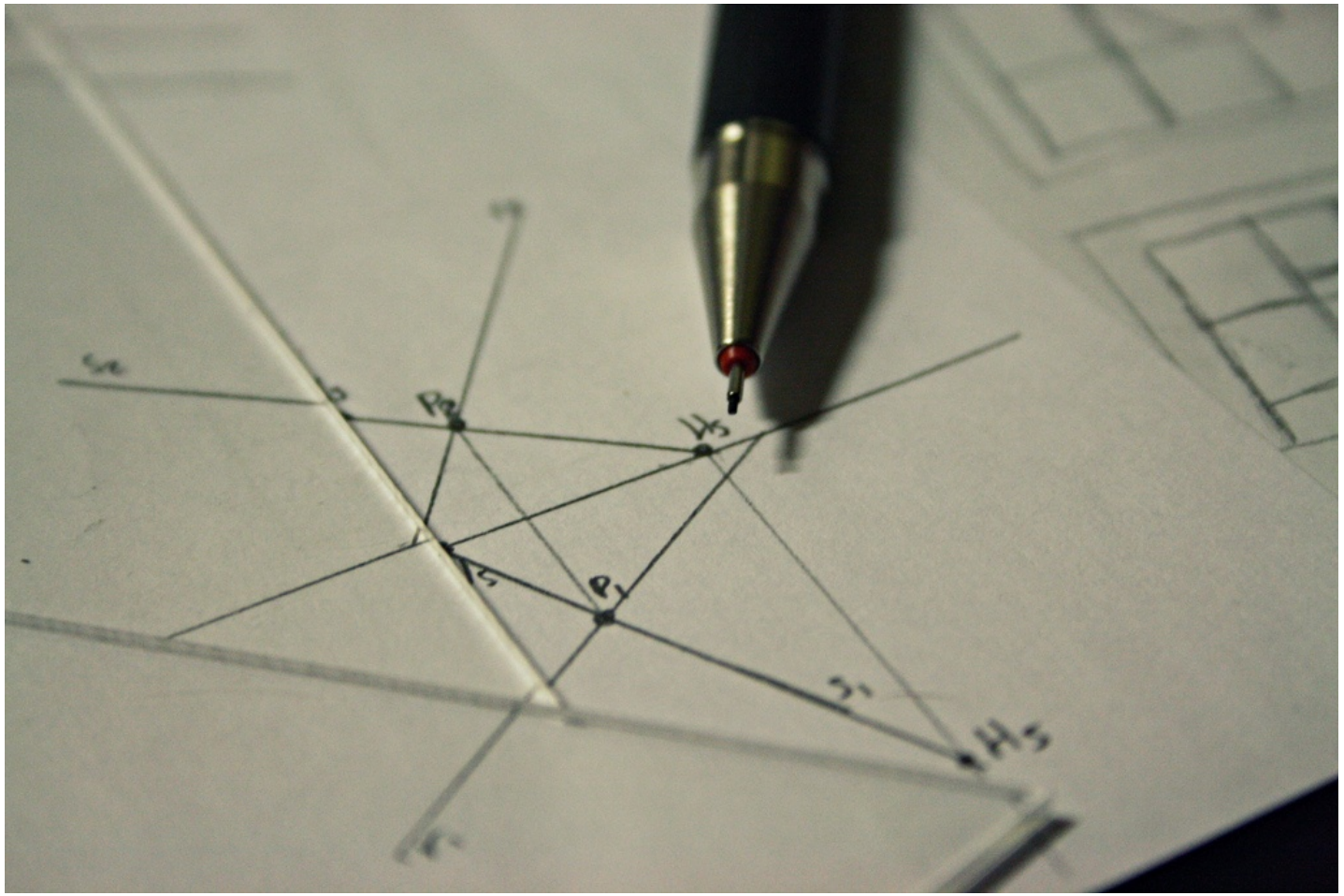
- add a test
- run all tests and see if the new one fails
- write little code
- run all tests and see them succeed
- refactor code **mercilessly**
- repeat

Uncle Bob's 3 Laws of TDD

- You are not allowed to write any ...
  - ... production code
    unless to make a failing test pass.
  - ... more of a unit test
    than is sufficient to fail the test.
  - ... more production code
    than is sufficient to pass the test.

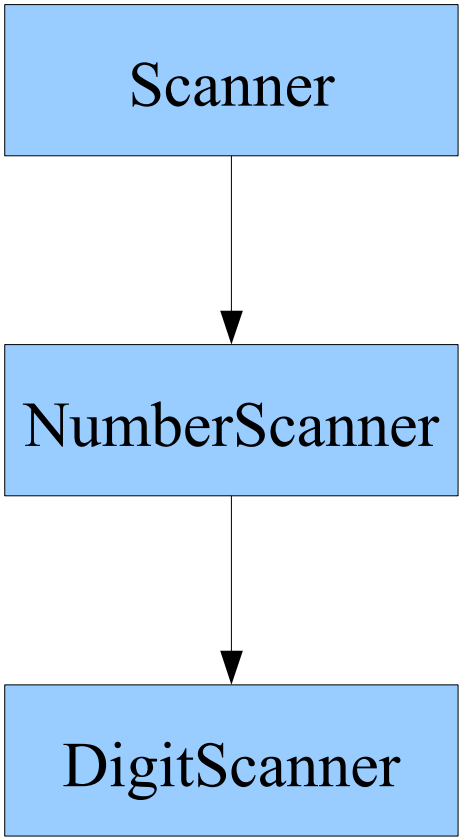
http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd

How do we design using TDD?



e.g. Scanning Numbers

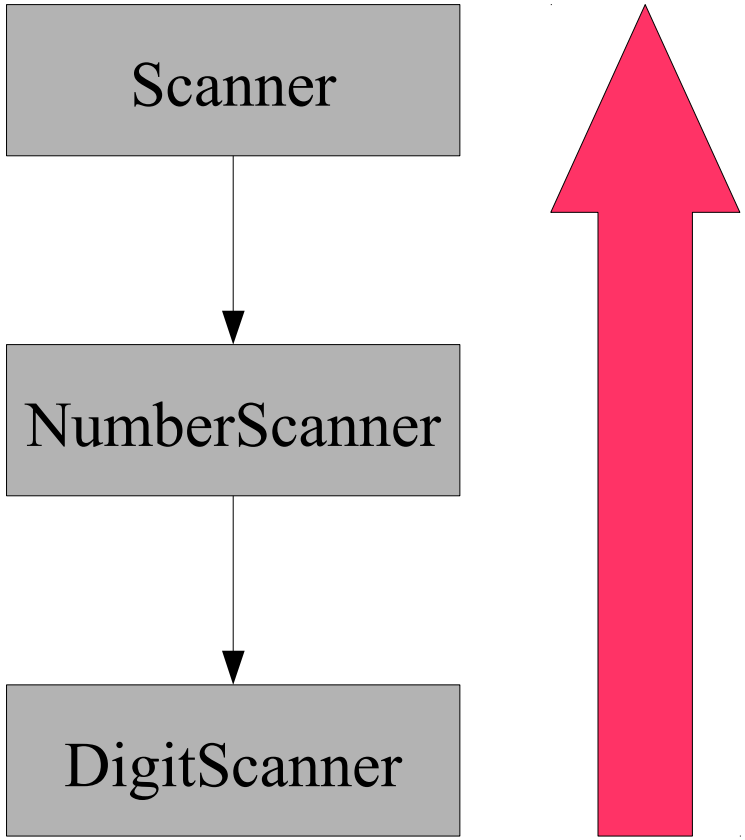
- We need to scan documents.
- Documents contain numbers.
- Numbers consist of digits.



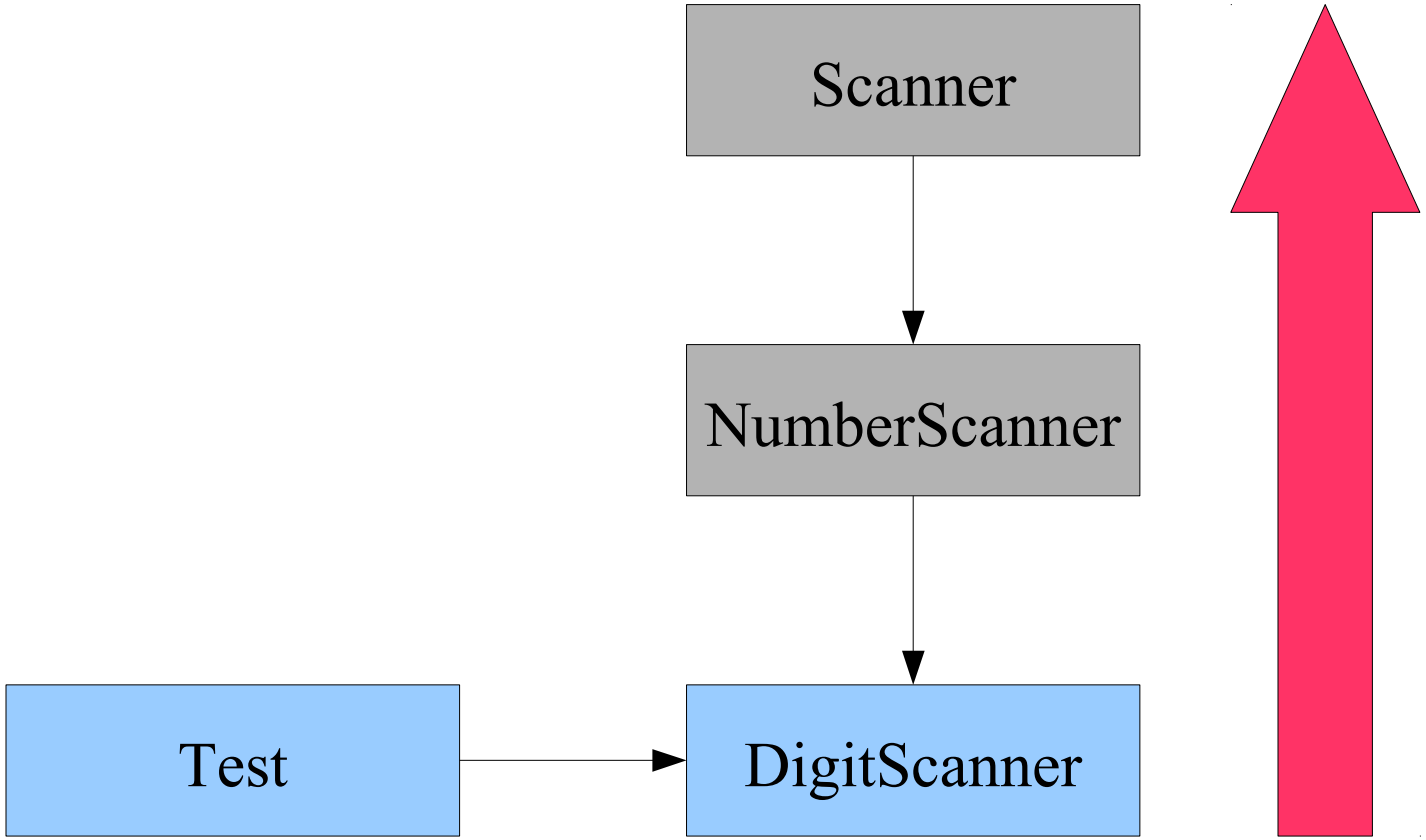
“Chicago” School TDD

Scanning Numbers #1

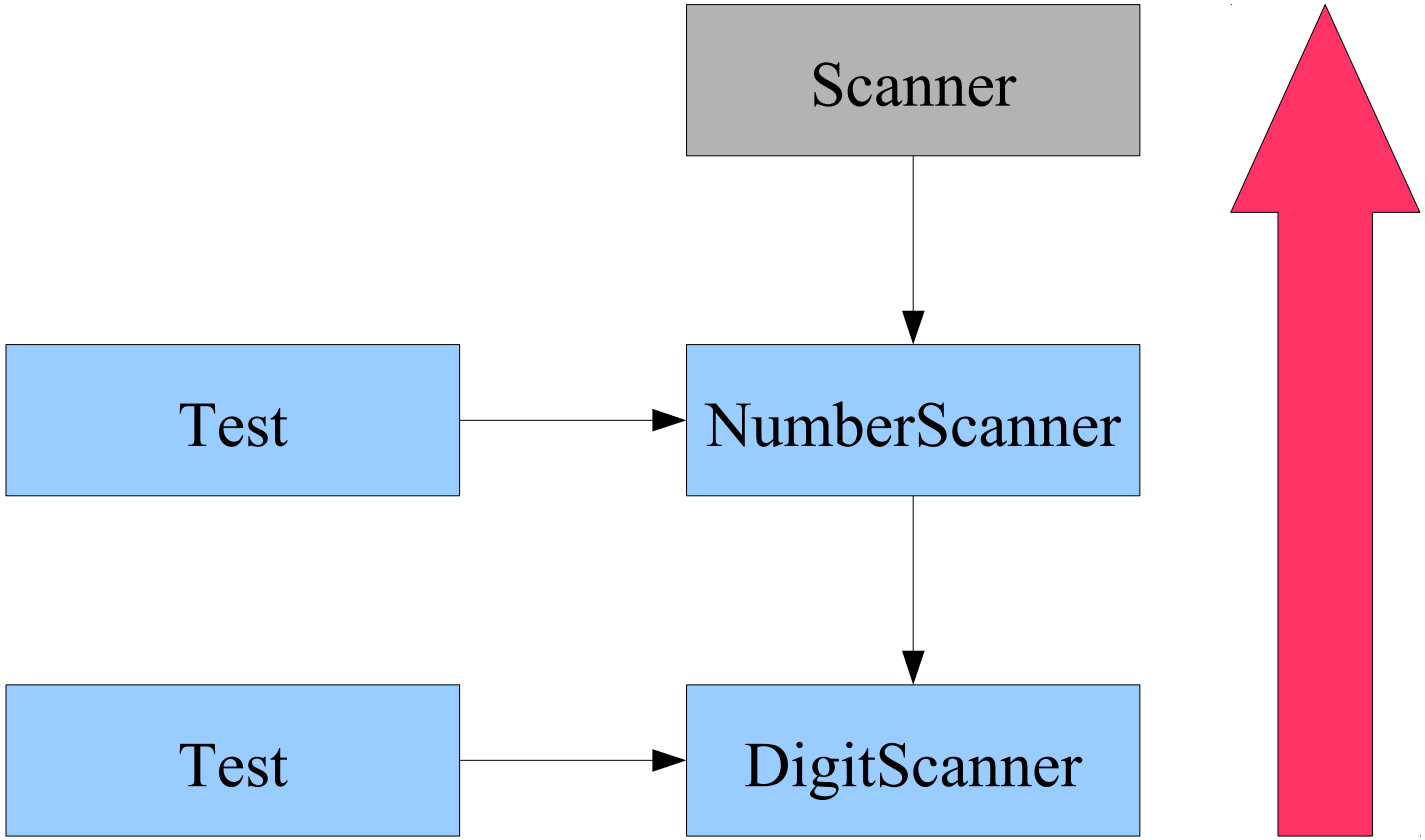
- Classic TDD aka "Chicago" or "Detroit School"
- Inside-out = working from „bottom“ up



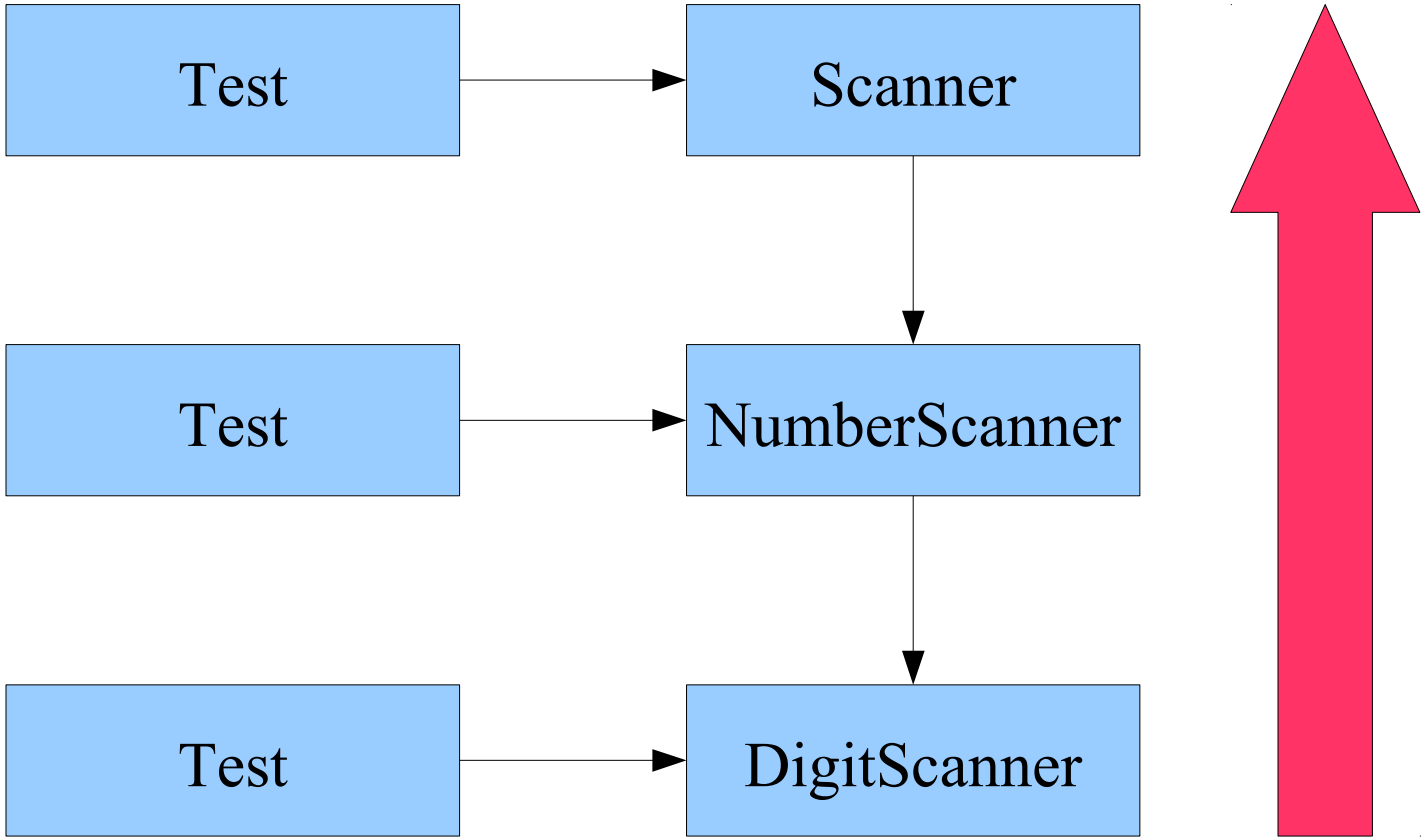
Scanning Numbers #2



Scanning Numbers #3



Scanning Numbers #4



Summary of Classic TDD

- Working from „bottom“ up
- Collaborators usually not mocked (just used)

- State-based tests

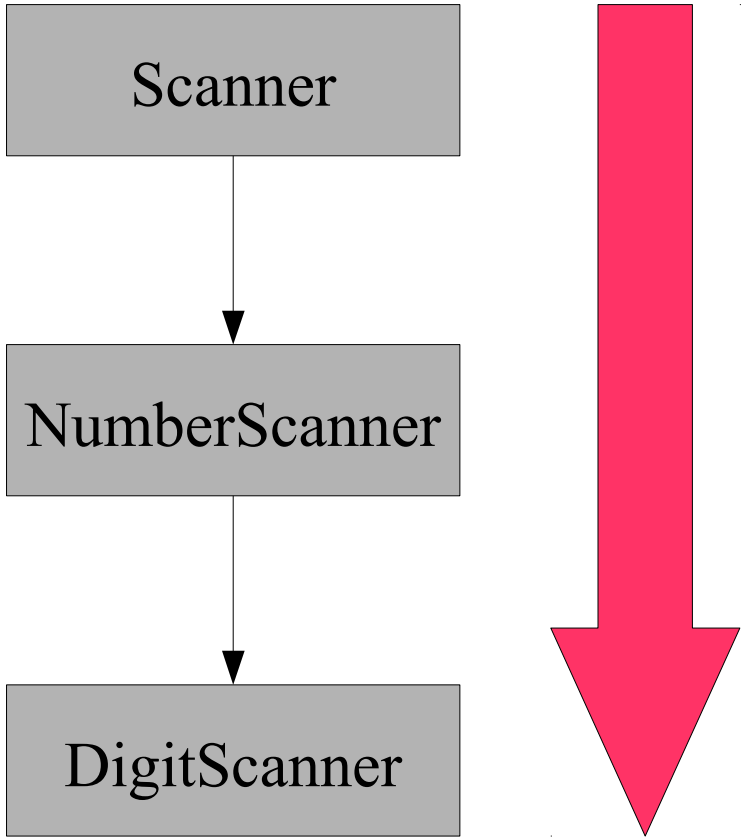
- Emergent design during refactoring
- Avoids over-engineering

Franziska Sauerwein, http://slides.com/franziskasauerwein/outside#/

“London” School TDD

Scanning Numbers #1

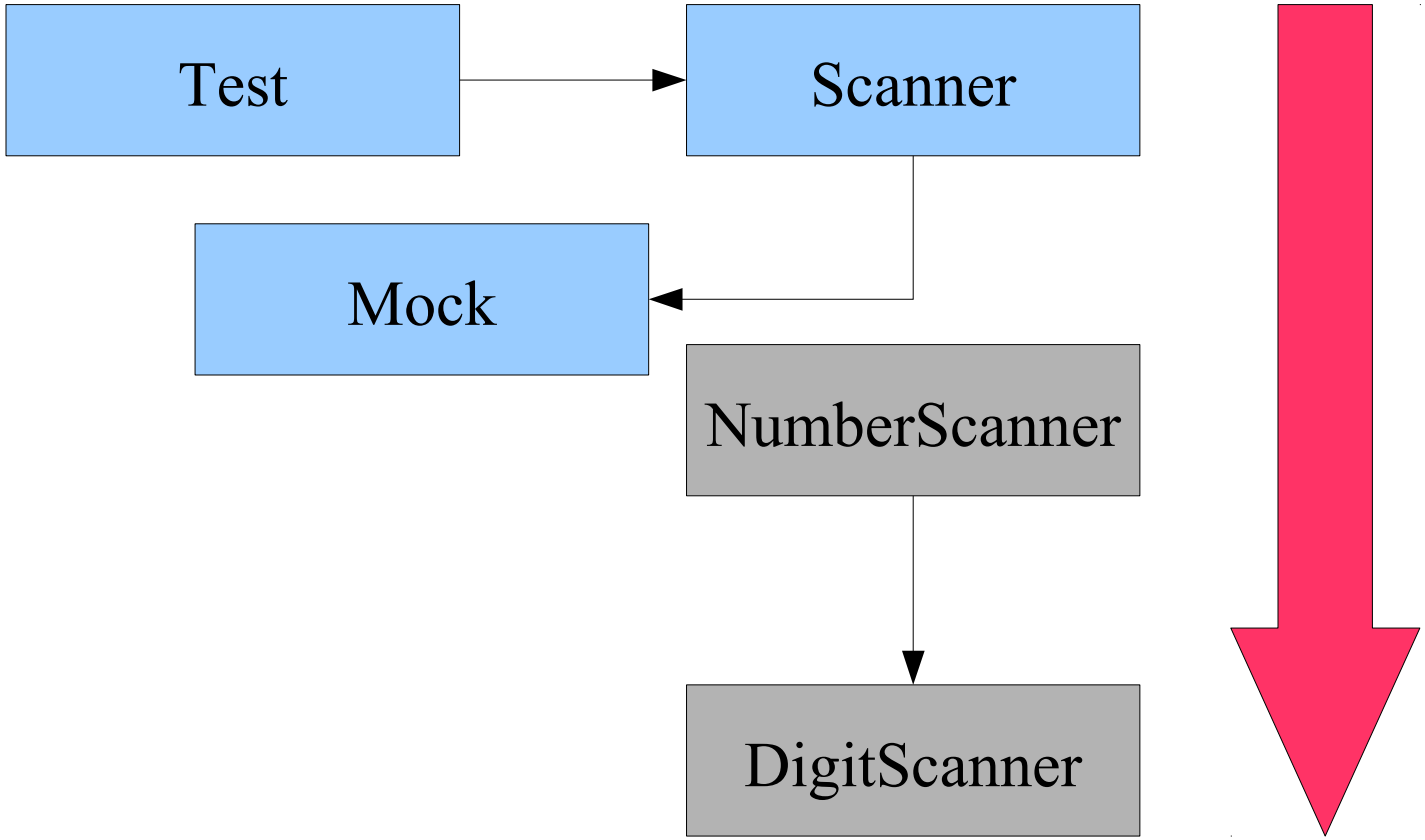
- Mockist TDD aka "London School"
- Outside-in = following the user interaction



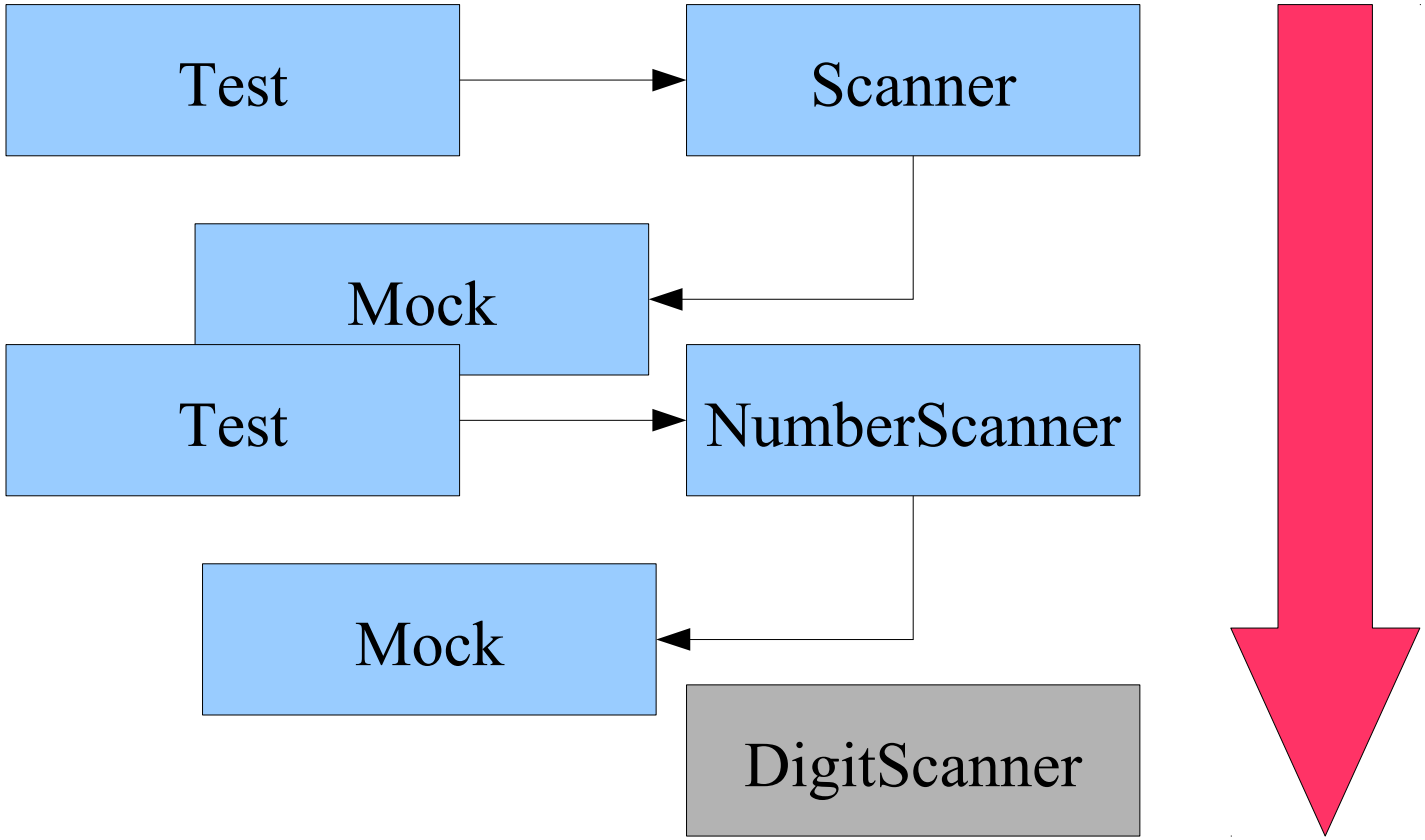
Outside-In

- build the system from the "outside-in"
- helps identify top level function/class, entry point to the desired functionality,
  - e.g. widget in GUI, link on a web page, or command line flag
- following the user interaction through all the parts of the system

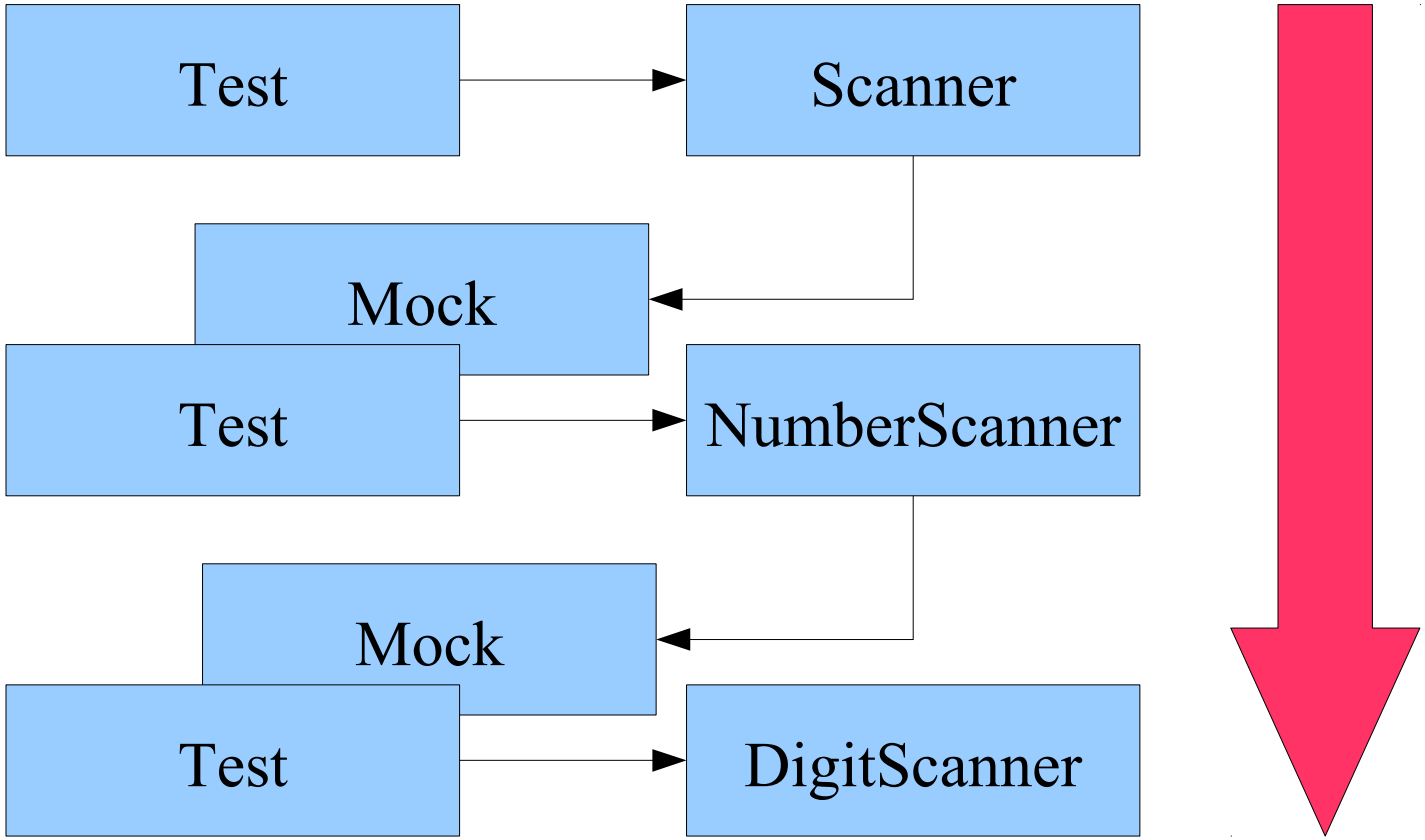
Scanning Numbers #2



Scanning Numbers #3



Scanning Numbers #4



Summary of Outside-In TDD

- Working from „outside“ in.
- Assume collaborators and mock them.

- Verify behaviour, not state.

- Design in Red stage.
- Follow Tell-Don't-Ask principle.

Franziska Sauerwein, http://slides.com/franziskasauerwein/outside#/

What is a “Mock”?

Five Types of Test Doubles

- Dummy (Object)
- Fake (Object)
- Stub
    - Partial Stub
- Spy
- Mock
    - Partial Mock
- Test-Specific Subclass

How to “Mock” an Object

- by hand
  - implement its interface (Eclipse Ctrl-1)
  - subclass it (beware complex constructors)
- with `java.lang.reflect.Proxy`
  - since Java 1.3
  - only for interfaces
  - nasty for more than 1 method

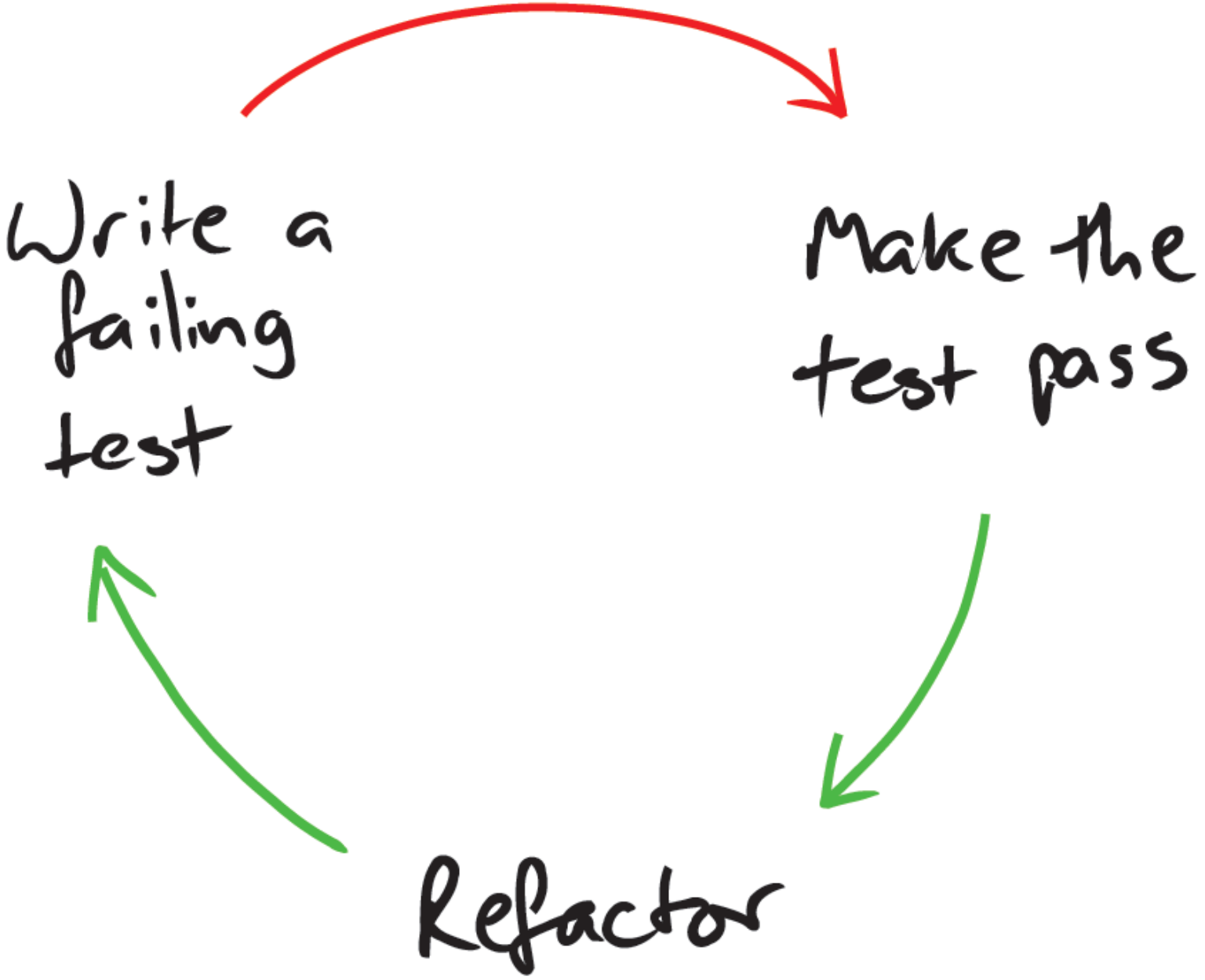
Mocking Frameworks

- e.g. **Mockito, moq, Sinon.JS**, ...
  - mock interfaces (Proxy)
  - mock non final classes (cglib)

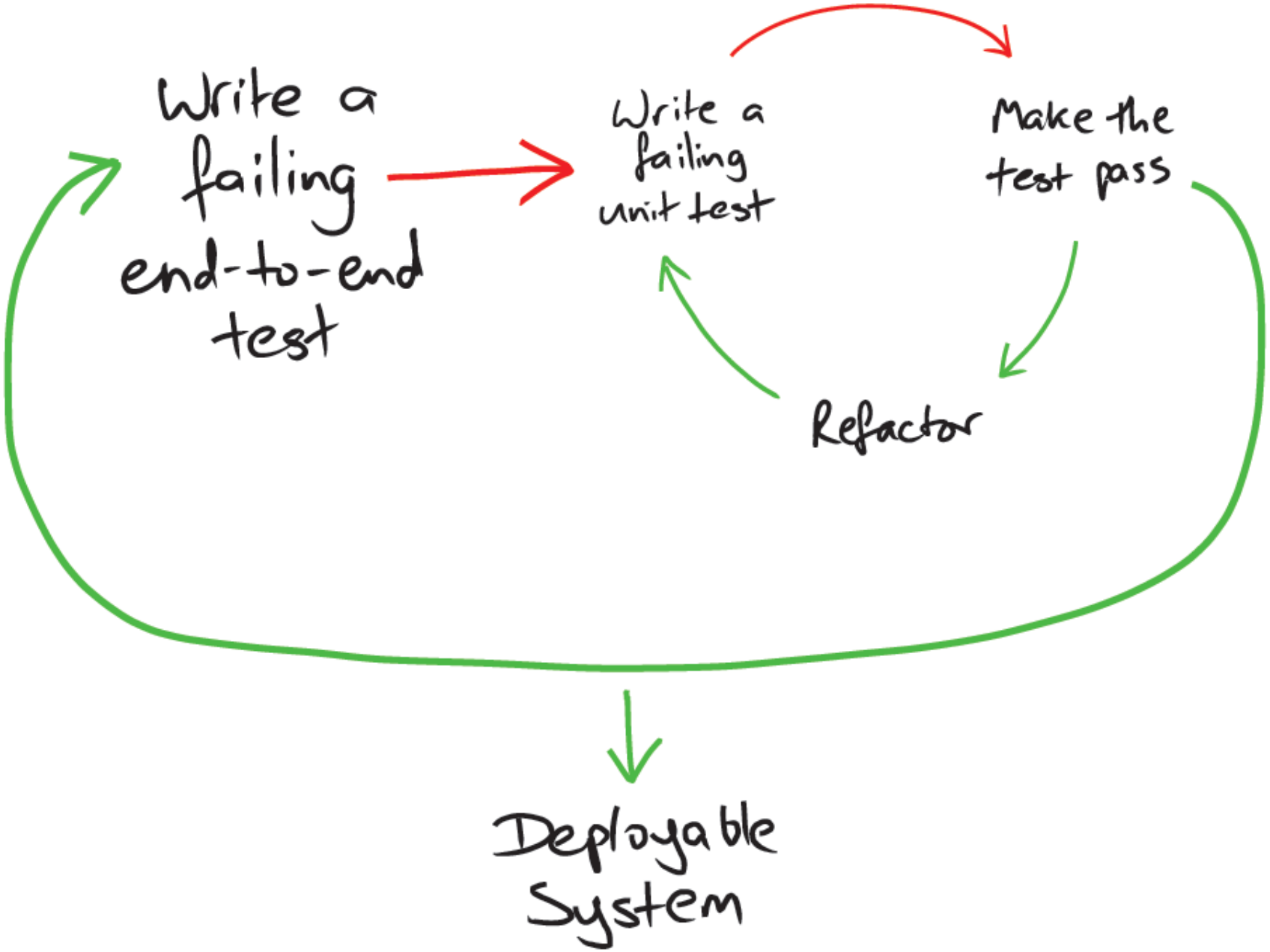
```
import static org.easymock.EasyMock.*;

SomeInt mock = createMock(SomeInt.class);
expect(mock.someMethod("param")).andReturn(42);
replay(mock);
// run the test which calls someMethod once
verify(mock);
```

Double Loop TDD



Nathaniel Pryce, http://www.doc.ic.ac.uk/~np2/teaching/



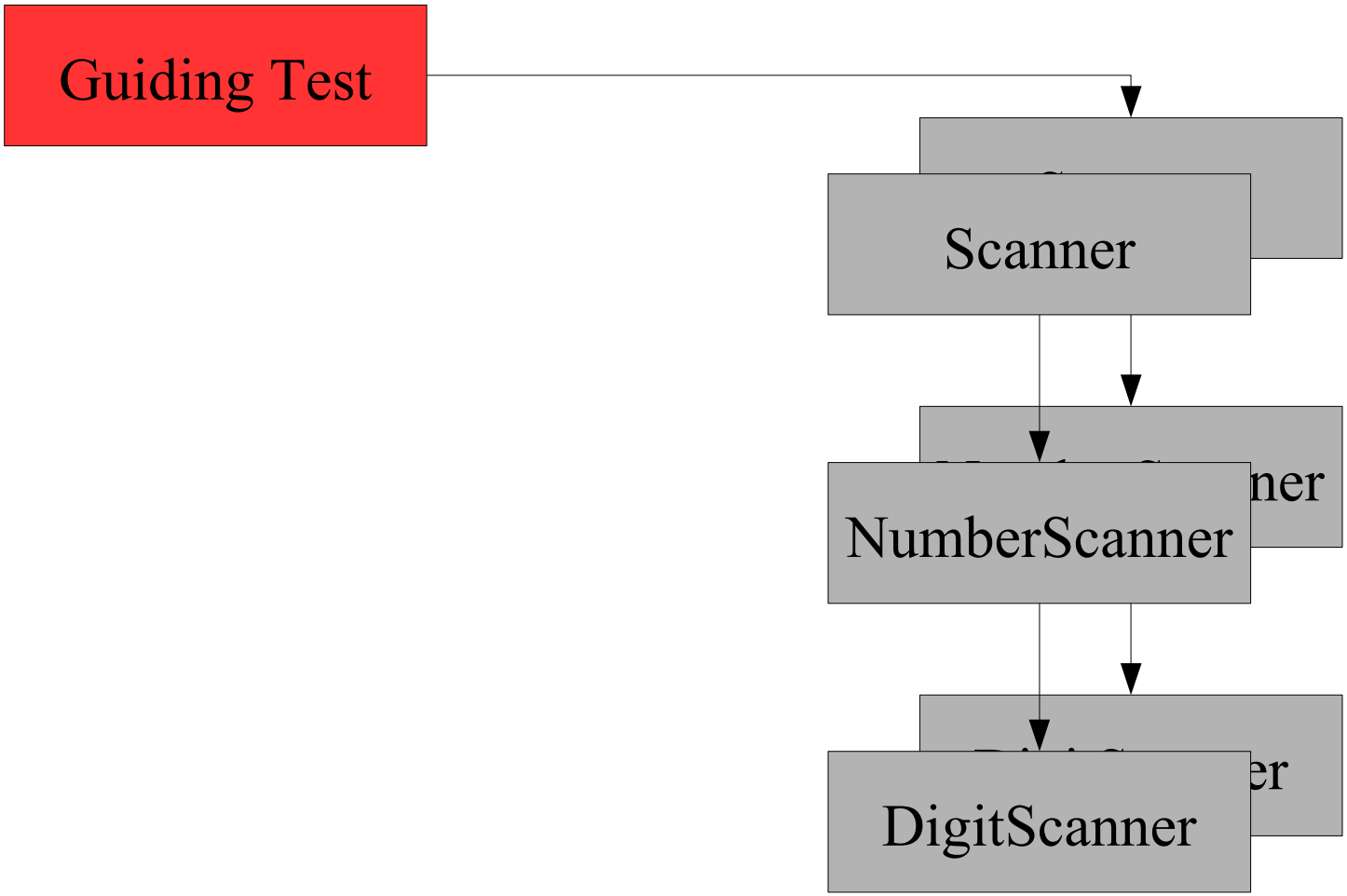
Nathaniel Pryce, http://www.doc.ic.ac.uk/~np2/teaching/

Double Loop TDD Design Process

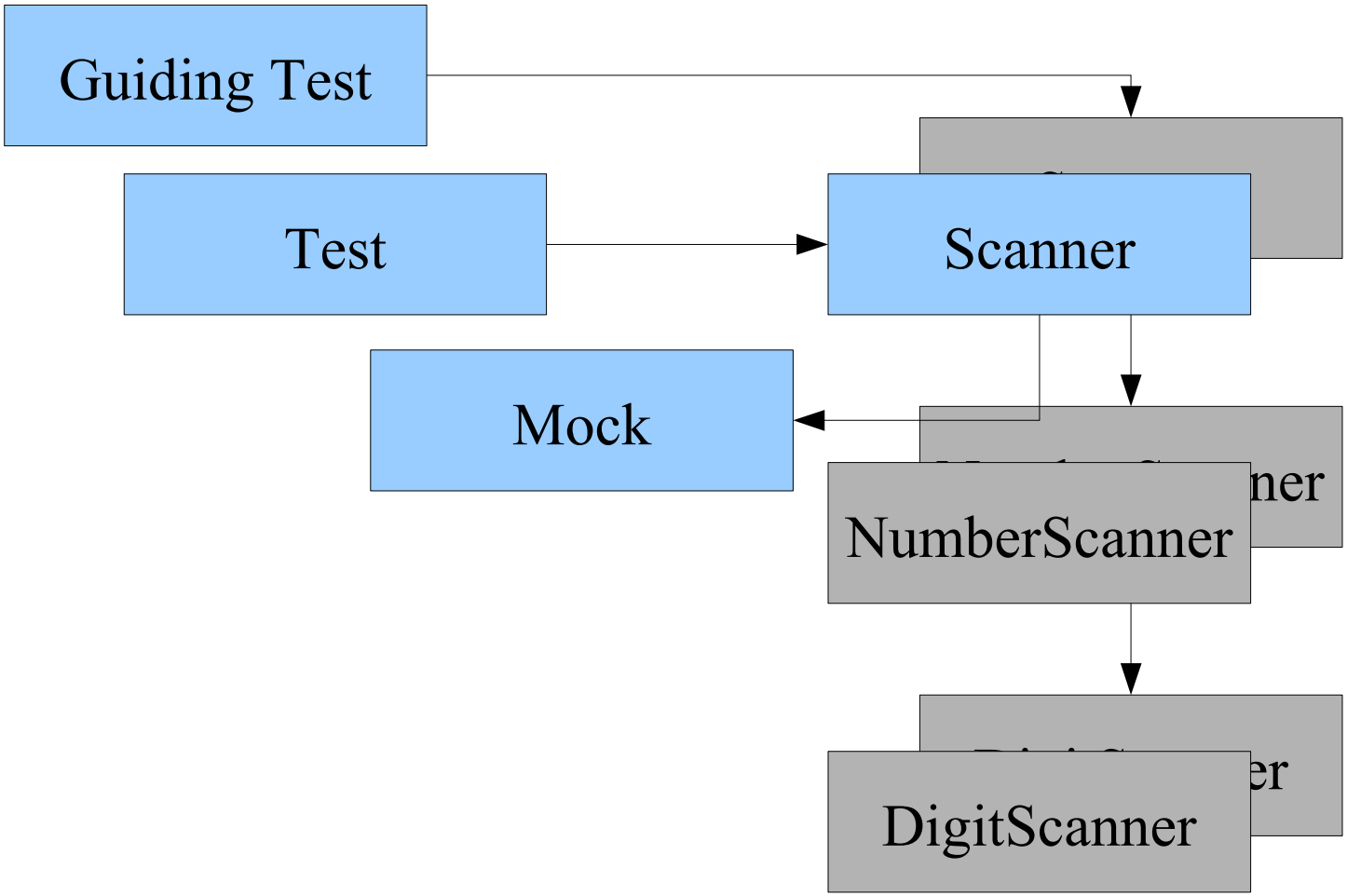
- create an Acceptance/Guiding Test (red)
- start with top level interaction (from UI)
  - discover/design needed collaborators
  - stub/mock these dependencies
  - implement using TDD
- run Guiding Test to see where to go next
- while Guiding Test is still red
- move down to previously mocked collaborator

Emily Bache, http://coding-is-like-cooking.info/2013/04/outside-in-development-with-double-loop-tdd/

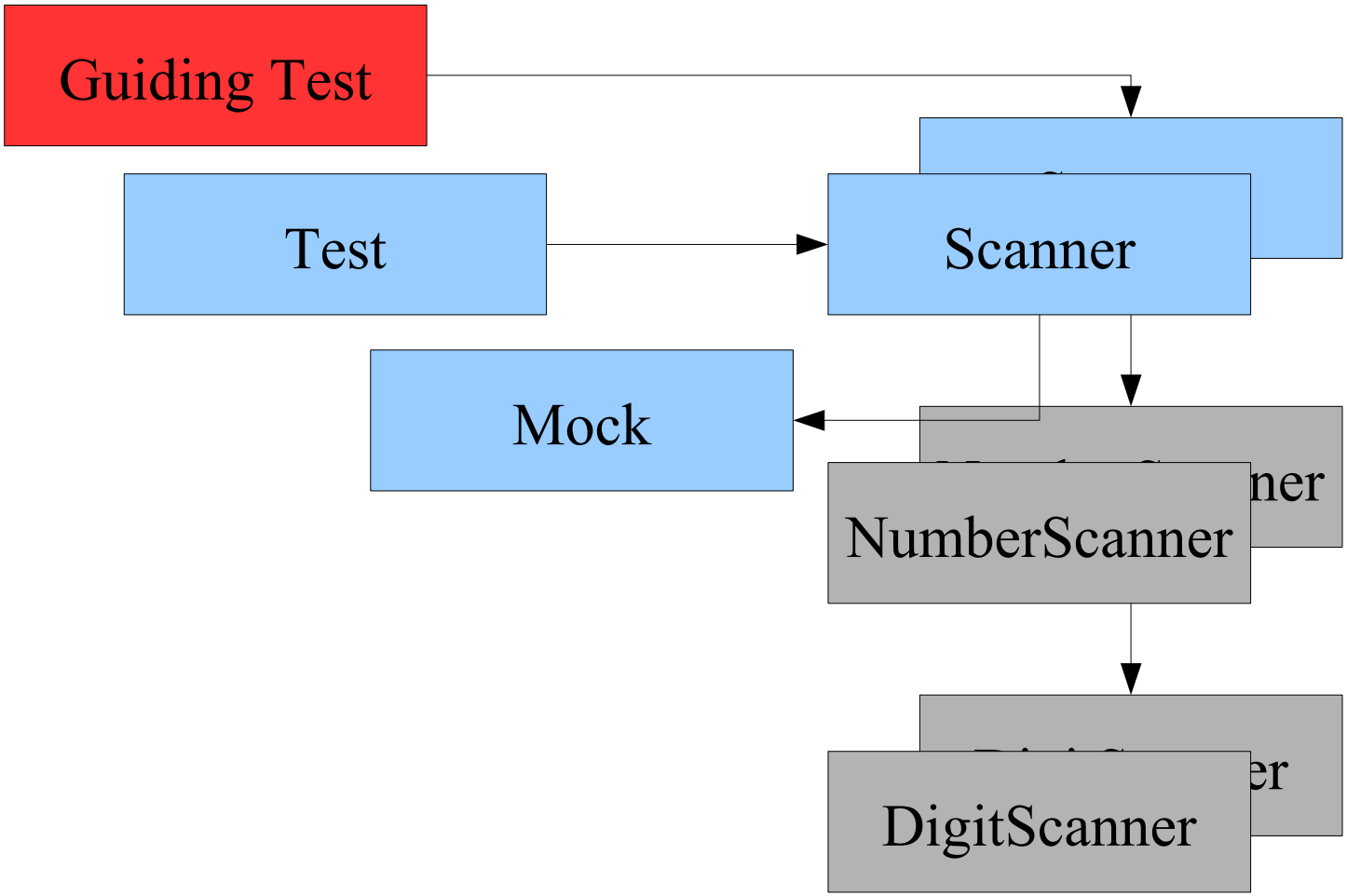
Scanning Numbers #1



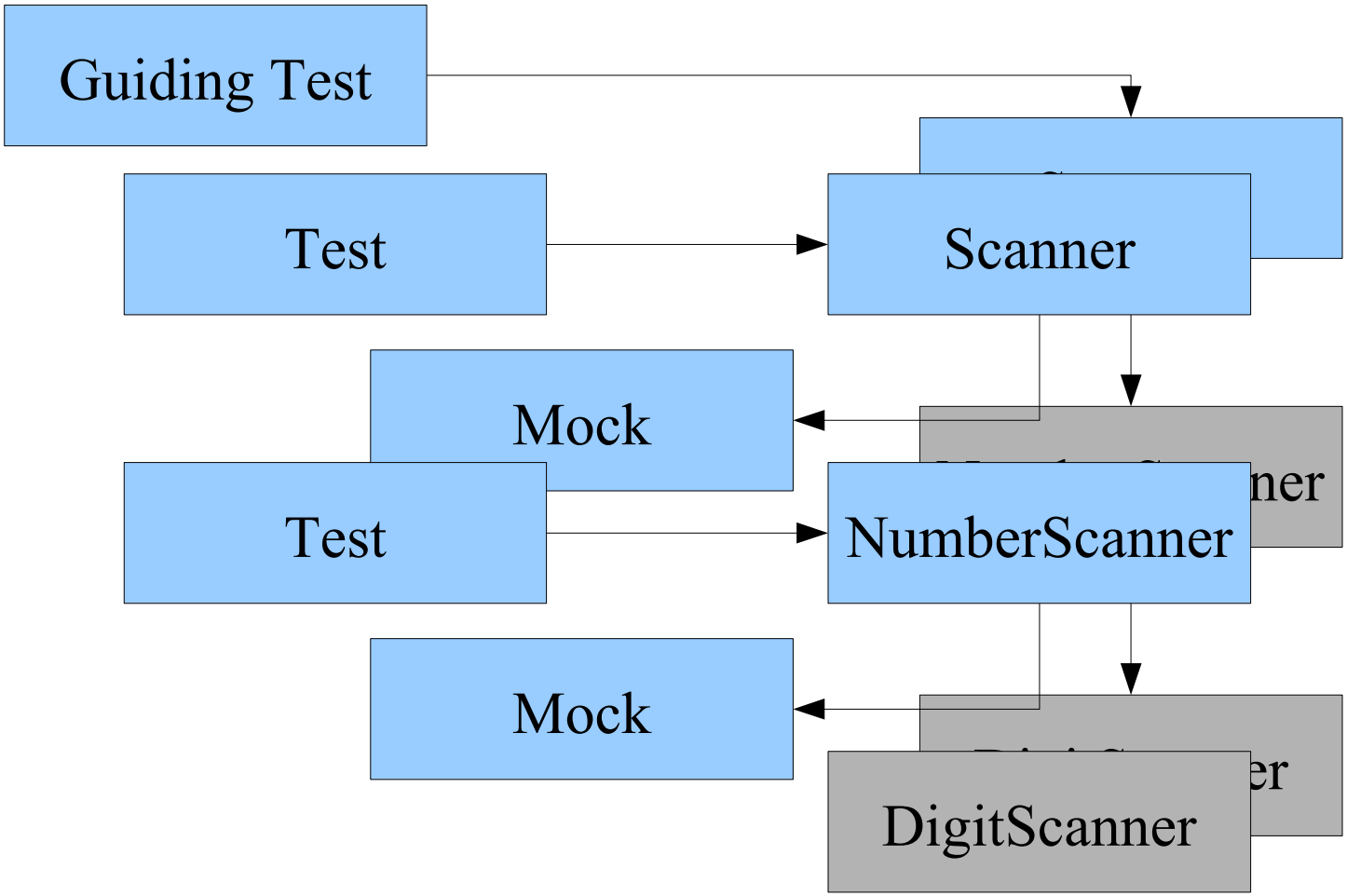
Scanning Numbers #2



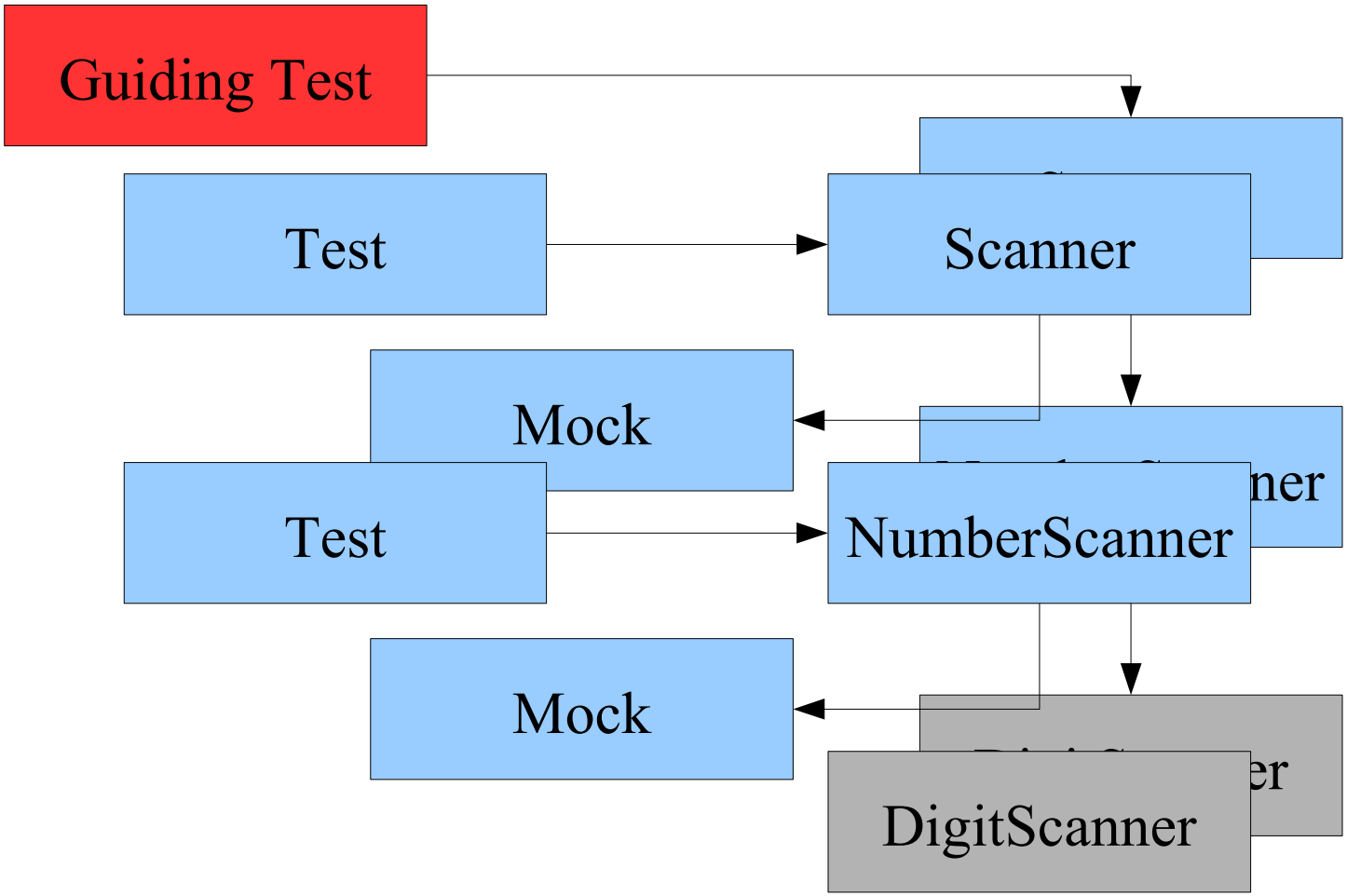
Scanning Numbers #3



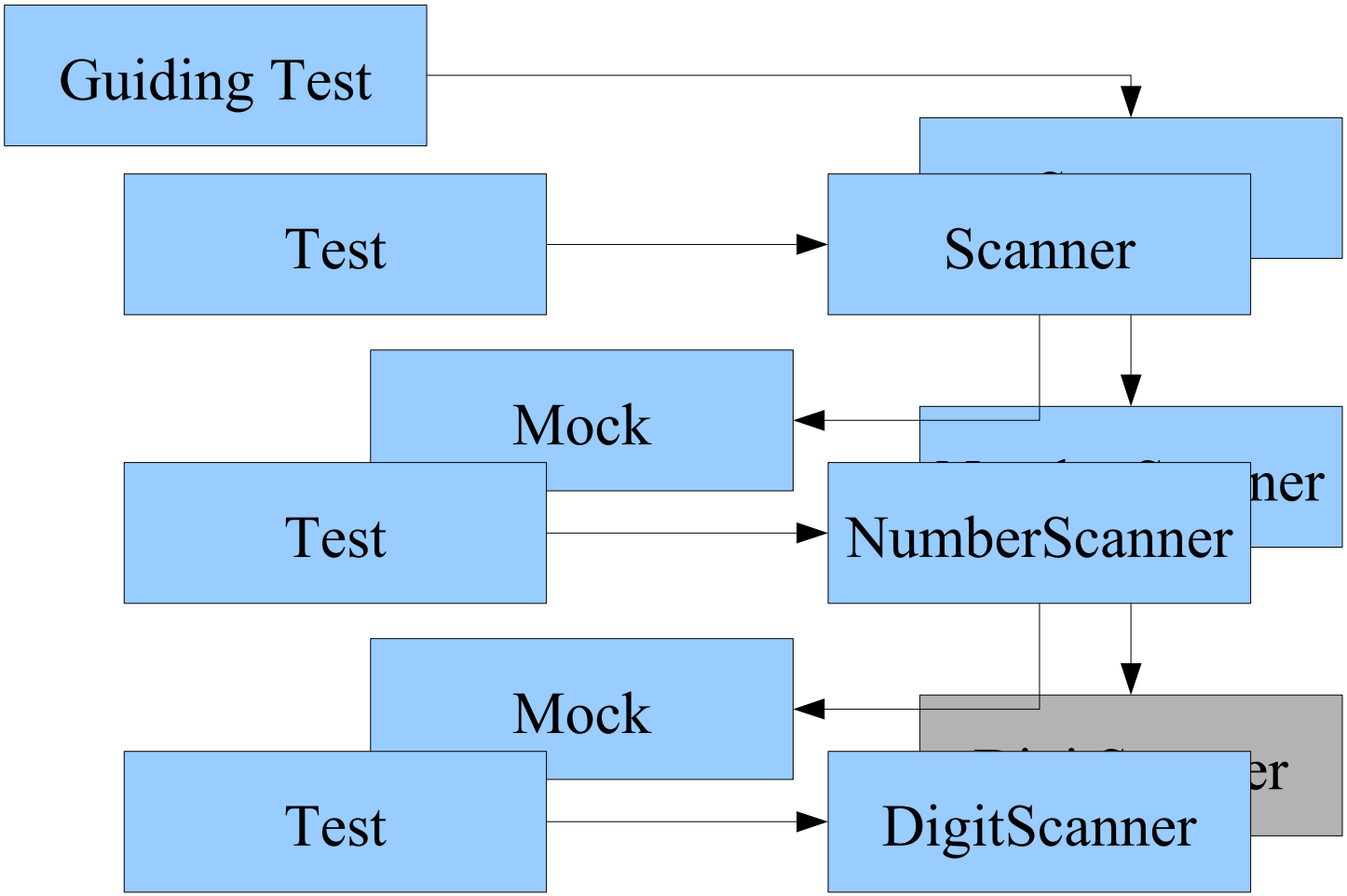
Scanning Numbers #4



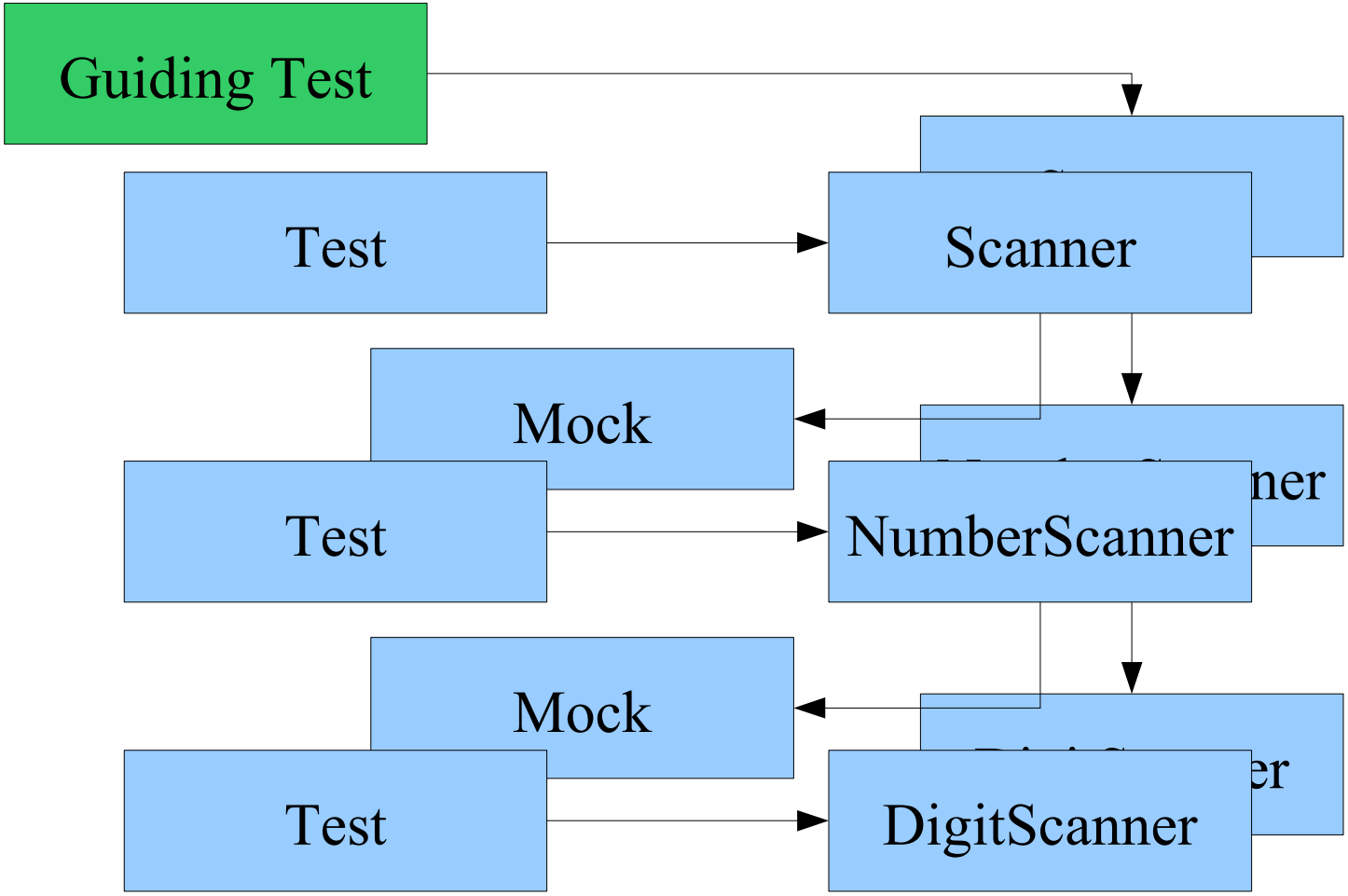
Scanning Numbers #5



Scanning Numbers #6



Scanning Numbers #7

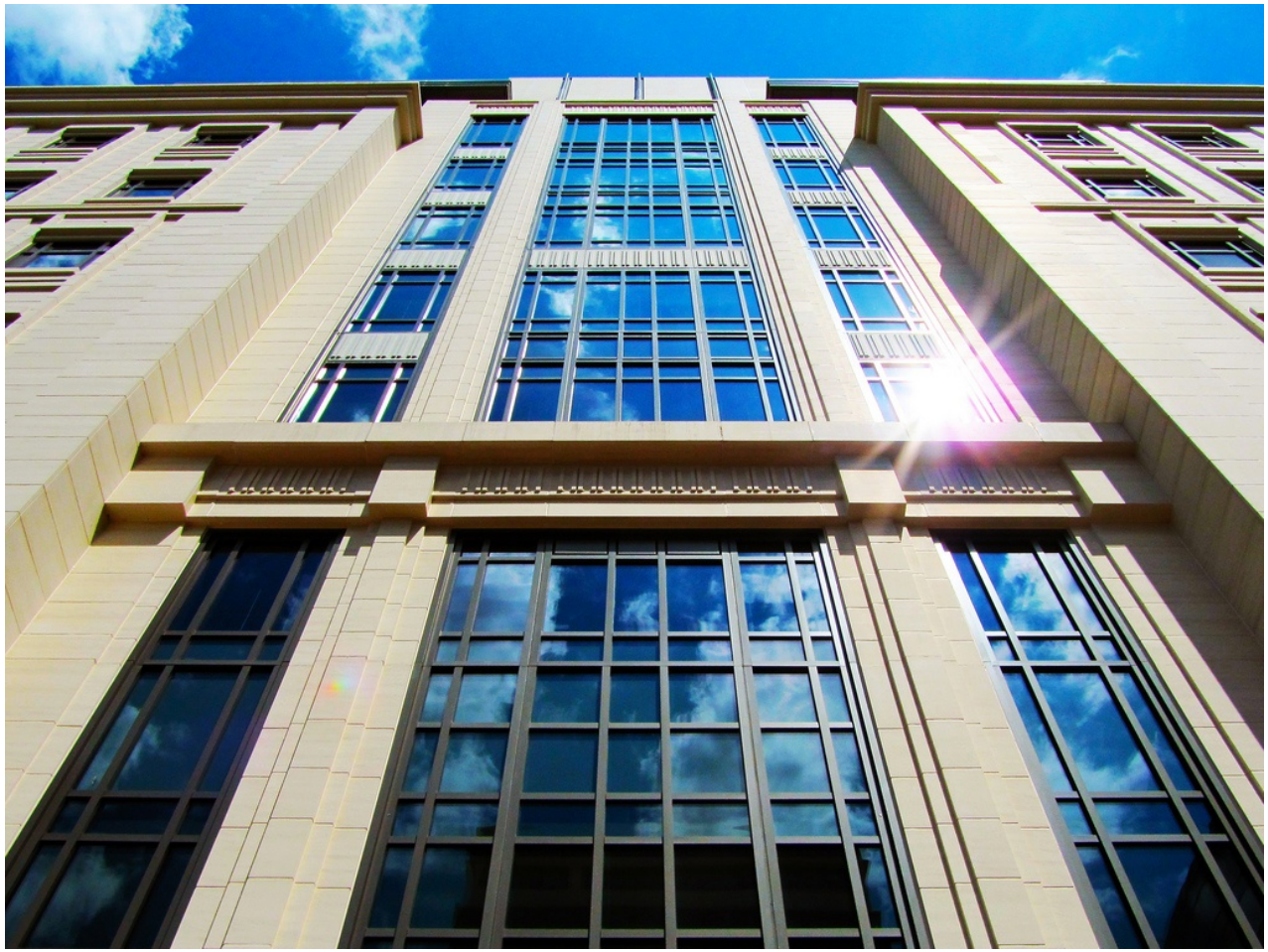


Try it yourself

Coding Dojo Mindset

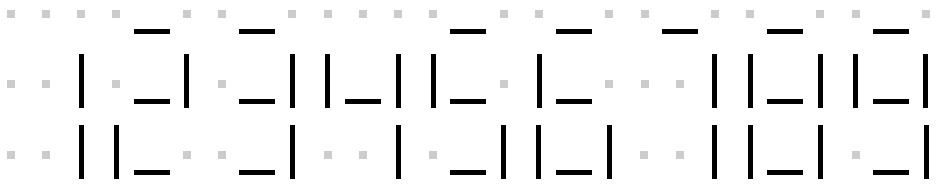
- Safe place outside work
- We are here to learn
- Need to slow down
- Focus on doing it right
- Collaborative Game

Assignment



Bank OCR

- You work for a bank, which has a machine to assist in reading letters. The machine scans the paper documents, and produces a file with a number of entries which each look like this:



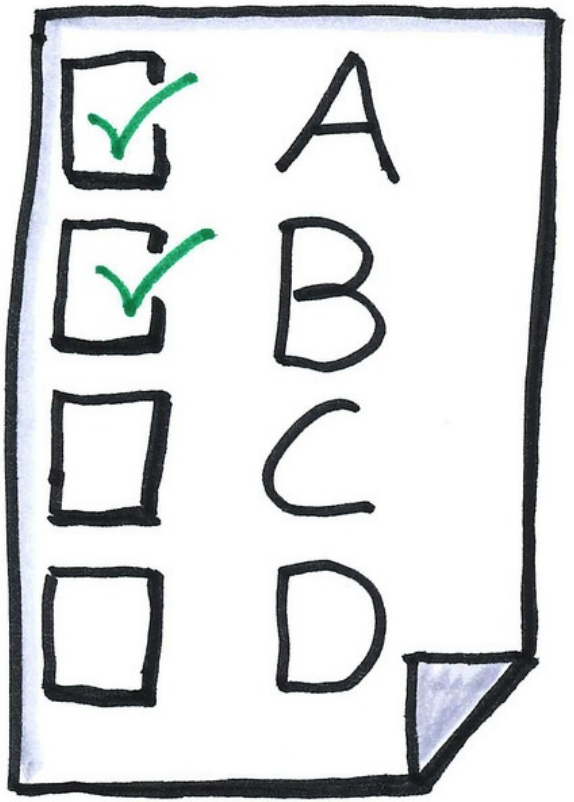
- Each entry is 4 lines long, each line has 27 characters. The first 3 lines contain an account number written using **pipes and underscores**, and the fourth line is blank. Each account number should have 9 digits, all of which should be in the range 1-9.
- Write a program that can take this file and parse it into actual account numbers.

Prepare

- Find a pair.
- Agree on a programming language.
- Get the project from
  https://bitbucket.org/pkofler/bankocr-kata-setup
- See `GuidingTest` (failing test)
  - Guiding Test is the starting point.
- Work through outer API, outside-in.
- Implement Bank OCR.

Recommended: A Test List

- Use first ten minutes to create list of acceptance test cases (on paper)
- Start each TDD cycle with at least three test cases before beginning to code (paper or text file)



Apply: Outside-In TDD

- build the system from the "outside-in", following the user interaction through all the parts of the system

- create a Guiding Test
- start with top level interactions
- mock dependencies
- implement using TDD until all tests green
- move inside previously mocked collaborator

Don't Focus on Getting it Done. Focus on Doing It Perfectly.

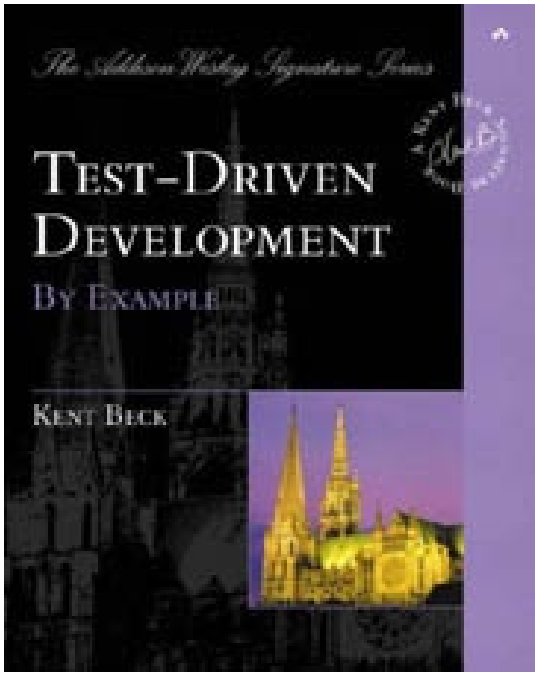
Practice

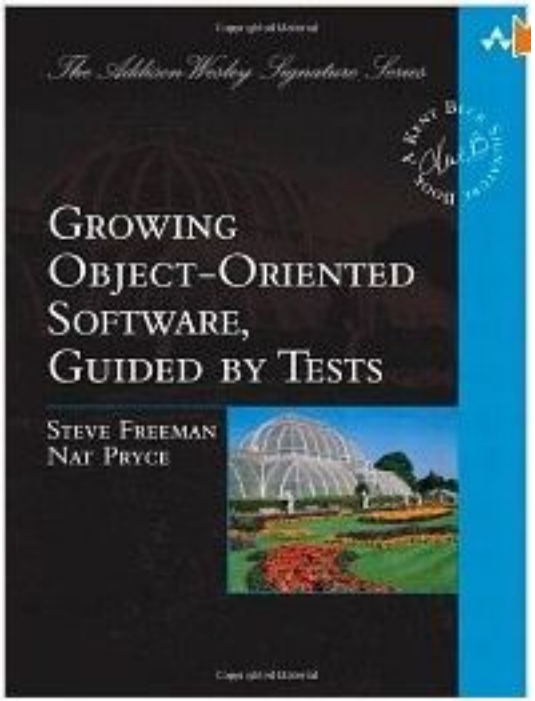
Closing Circle

- What did you learn today?
- What surprised you today?
- What will you do differently in the future?

Study!

- TDD is important.
- You need to study it.
- Good books:





Peter Kofler
@codecopkofler
www.code-cop.org

EXPERIENCE THE VALUE OF QUALITY

CC Images

- London https://www.flickr.com/photos/damski/8019978119
- Bruce http://www.flickr.com/photos/sherpas428/4350620602/
- pairing http://www.flickr.com/photos/dav/94735395/
- agenda http://www.flickr.com/photos/24293932@N00/2752221871/
- wants you http://www.flickr.com/photos/shutter/105497713/
- drawing https://www.flickr.com/photos/msk13/4108489367
- Chicago https://www.flickr.com/photos/pedrosz/3488261555/
- mocks http://www.flickr.com/photos/sneddon/2413980712/
- loops https://www.flickr.com/photos/fitzharris/7592626086/
- hands https://www.flickr.com/photos/ninahiironniemi/497993647/
- dojo http://www.flickr.com/photos/49715404@N00/3267627038/
- bank https://www.flickr.com/photos/bigmacsc99/4325336251