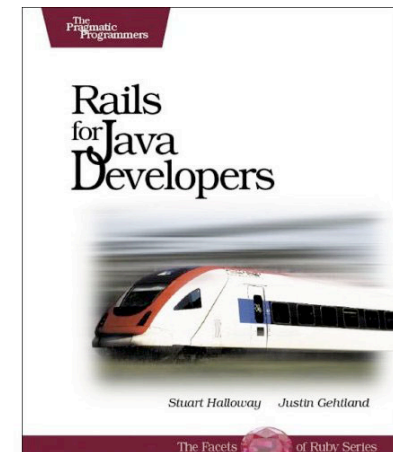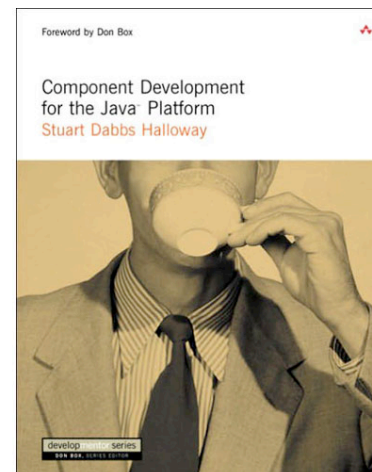# Test-Driven Development

## Course Authors:
Scott Davis and Stuart Halloway

# About the Authors

Scott: Editor of AboutGroovy.com, Davisworld Consulting LLC
Stu: CEO of Relevance, Inc.

# Why Test?

# Reasons Devs Don't Test

- tests take too much time to write

- tests take too much time to run

- it's not my job to test code

- aren't acceptance tests enough?

- but it compiles...

relevance

# Untested code is legacy code.

# Testing Tools

- XUnit

- mock objects

- code coverage

- low-ceremony languages

relevance

# The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more..

Source: **http://www.agilemanifesto.org**/

# Course Materials

http://github.com/relevance/java-tdd

readings

examples

labs

additional resources

# Section Reviews

# XUnit

- test classes extend a base test class

- test methods marked by annotation or convention

- setup and teardown for shared test config

- run suites in graphical or console runners

- test FIRST:

  http://blog.objectmentor.com/articles/2007/08/02/which-came-first

# BDD

- behavior-driven development

- common domain vocabulary

- closer to human language

- executable documentation

- helps enforce "first"

# Coverage

- more than one kind of coverage

- low coverage is bad

- high coverage proves nothing

- interpret coverage in context!

# Refactoring

- keep code DRY

- make code intentional and cohesive

- remove code smells

- red, green, refactor

- consult the catalog

relevance

# Mocking

- stubs isolate code

- mocks test behavior

- mock-friendly code is good code

  (except when it isn't)

- know when to quit mocking

relevance

# Refactotum

- start with a clean checkout and build

- use tests and metrics to find code smells

- red, green, refactor

- contribute in small increments