Agenda

- Sit with your teams
- One baggie of Lego per person

- Announcements
- Lecture
  - Test-Driven Development

- Tutorial
  - Scrum with your TA
  - Quiz

Announcements

- Peer Evaluation Forms

- Preliminary Deliverables now due on Tuesdays at 10pm
  - TAs need more time to provide feedback

Understanding TDD and Refactoring with LEGO

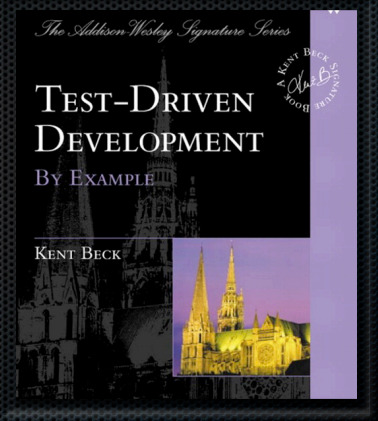
- Material adapted from Bryan Beecham
- @BillyGarnet

Exercise - 1

- Open up one LEGO packet
- Build a person and a house out of Lego
- Admire your work
  - Take a photo.
  - Upload to Twitter or Facebook.
  - Brag to your friends.

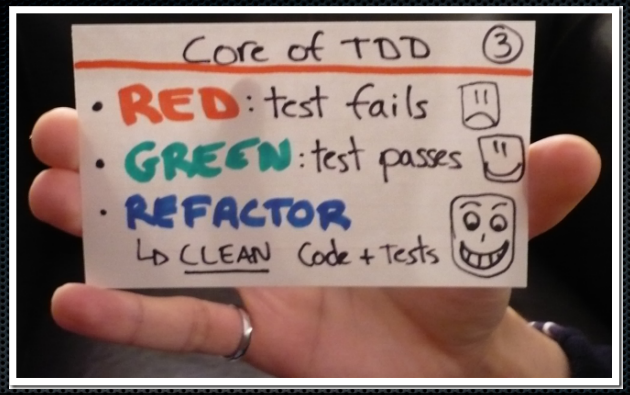
Reference

Test-Driven
Development
By Example
Kent Beck



Why TDD

- Clean code that works ~ Ron Jefferies
- It is a predictable way to develop. You know when you are finished, without having to worry about a long bug trail
- It gives you a chance to learn all of the lessons that the code has to teach you. If you only slap together the first thing you think of, then you never have time to think of a second, better thing.



The Mantra
Red - Green - Refactor
photo from doolwind.com

The Mantra - Red

- Write a small test that doesn't work

The Mantra - Green

- Do the minimum to make the test work

The Mantra - Refactor

- Eliminate duplication

Exercise - 2

- Build a person and a house with TDD

Prepare your environment

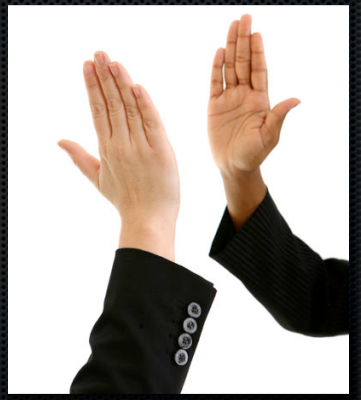
- Clear the area in front of you. This is your program.

First Test

- Does the person exist? No?

Hurray!!!

We failed the test!
Celebrate! High 5s!



Minimum to pass the test

- Add a brick. Can that be a person?

Hurray!!!

- The person now exists!
- Not very impressive but it could represent a person.
- We passed the test! We are rocking now!

Refactor

- Remove any duplication. In this case we're good.

Same thing for house

- Steps?
- Prepare your environment.
- First test.
- Test fails.
- Minimum to pass the test.
- Refactor.

We need a new test

- The house is taller then the person.
- Assert.IsTrue(house.height > person.height);

Hurray!!! - More Failure

The person is the same size so we fail this test.
Well done!

Failure = Learning Opportunity

- If you're not failing, you're not learning.

Minimum to pass the test

Hurray!!! - Success

- Alright, we passed the test.

Refactor

- Still very simple. Still nice and clean.

Software Requirements

- Software must do three things:
  - It must work
  - It must be understandable
  - It must be updatable
  - (Robert C. Martin...I think)

We need a new test

- Is the house wider then the person? No?
- We failed another test! Awesome! We are learning a lot about improvements that are needed to our code.
- Let's do the minimum to pass the test.
- Any duplication to remove?

We need a new test

- Can your person fit in the house? Yikes! No.
- We failed another test! Awesome! We are learning so much about what our customer needs.
- Let's do the minimum to pass the test.
- Any duplication to remove?