Understanding TDD with LEGO

Bryan C. Beecham
@BillyGarnet

Refactoring in Python (Video Training)

Test Driven Development (TDD), Refactoring and Pair Programming (Workshop)

Exercise - 1

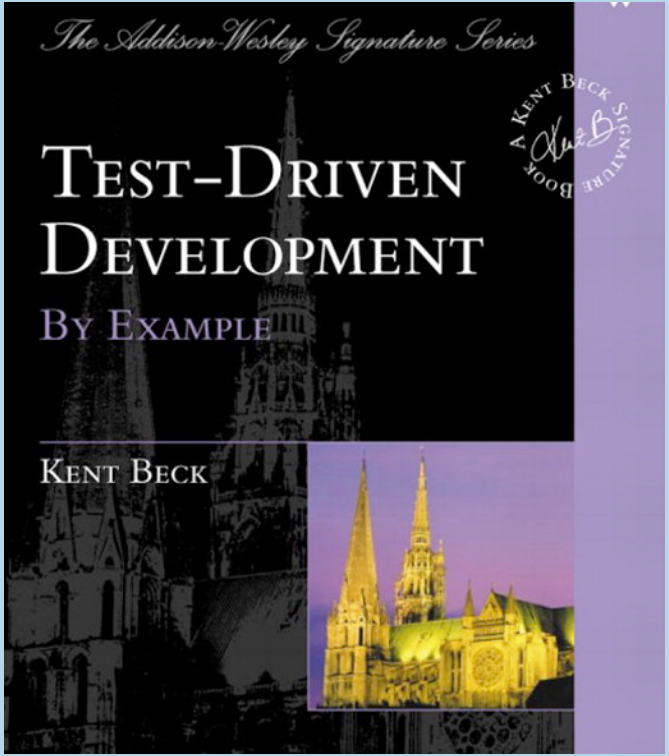
Build a person and a house out of LEGO

Admire your work

Take a photo. Send it to your friends.

What is the goal of TDD?

It creates clean code that works
~ Ron Jeffries


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck

Why do we do TDD?

- It is a predictable way to develop. You know when you are finished, without having to worry about a long bug trail.

TDD

- 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.

TDD

- It improves the lives of the users of your software.

TDD

- It lets your teammates count on you, and you on them.

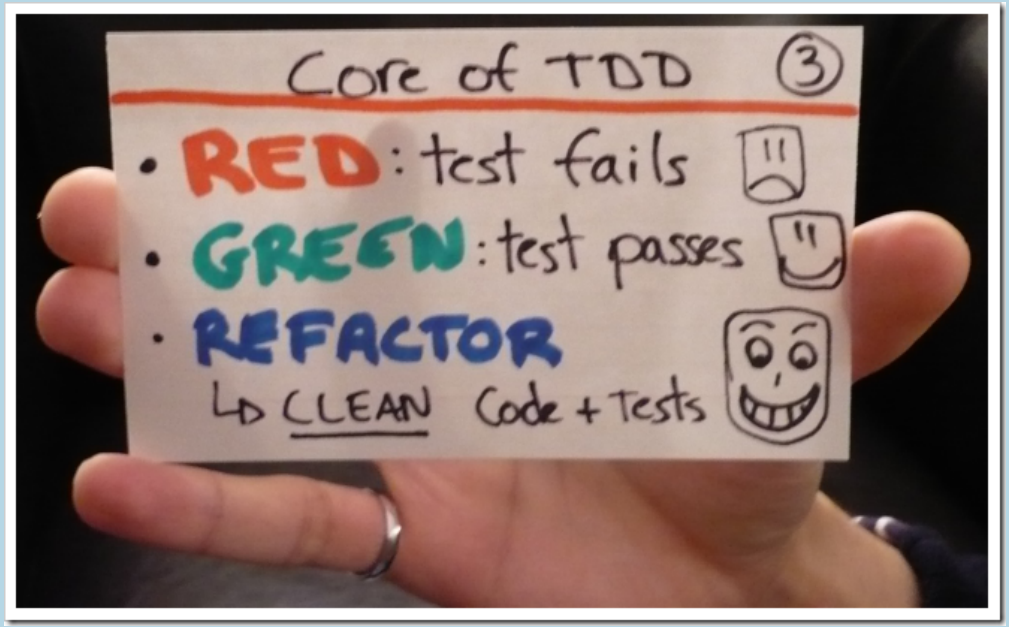
TDD

- It feels good to write it.

Red - Green - Refactor

photo from doolwind.com

THE MANTRA



Red

Write a small test that doesn’t work

Green

Do the minimum to make the test work

Refactor

Eliminate Duplication

Improve Readability and Understanding

Simplify

Exercise - 2

Build a person and a house with TDD

Prepare your environment

- Clear the area in front of you. This is your program.
- The perfect program! Wait…what? why?

First Test

- Does the person exist? No?

We failed the test! Celebrate! High 5s!

HURRAY!!!

Minimum to pass the test

- Add a block. 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

- Blah, blah, awesome re-creation by speaker...everyone is extremely impressed and are thinking this might be the best session they have every seen. (I wonder if anyone is reading this…)

We need a new test

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

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

HURRAY!!!
MORE FAILURE

Failure = Learning Opportunity

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

Minimum to pass the test

- From the audience this time. Anyone? Bueller?

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

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?

Exercise - 3 Partners!

- Break into groups of two for this next exercise.
- When developers do this we call it Pair Programming

- Berkley photo from the web

New Requirements

Your new program needs to have:

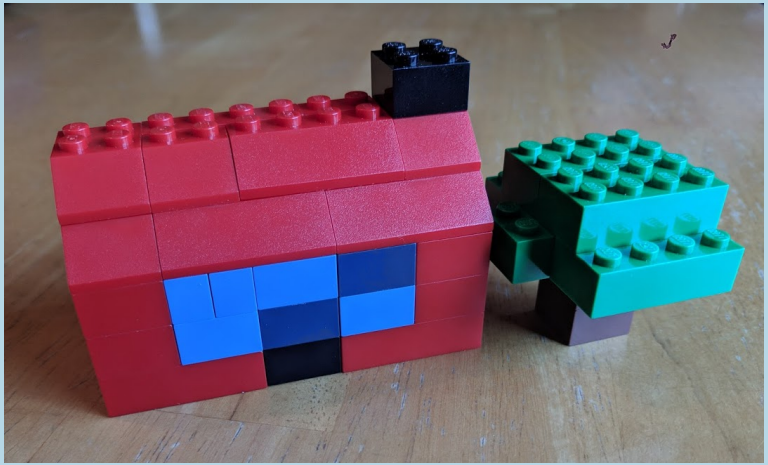
- A person
- A house
- A plant
- An animal
- A vehicle

What's the TDD order again?

- Write a (one) test on the paper.
- Check that it fails and mark an  beside it.
- Build only what's needed to make it pass.
- Check that it passes and mark a  beside it.
- Look for ways to improve the design.
- When you're satisfied, mark an R beside the test.

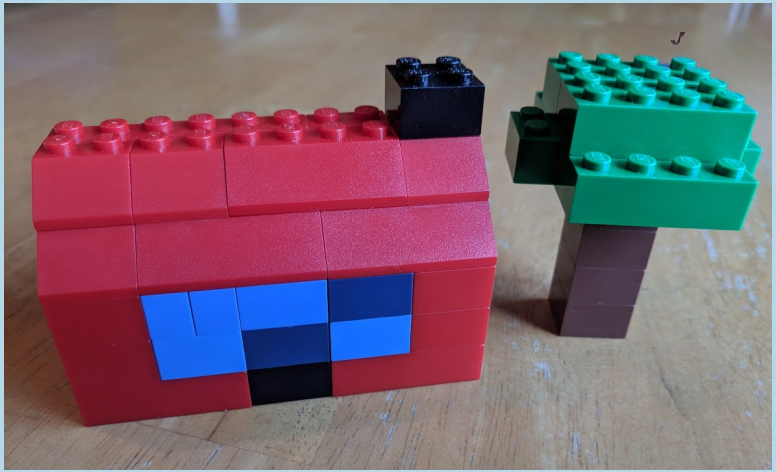
What's the TDD order again?

Is the tree the same size as the house? ✗



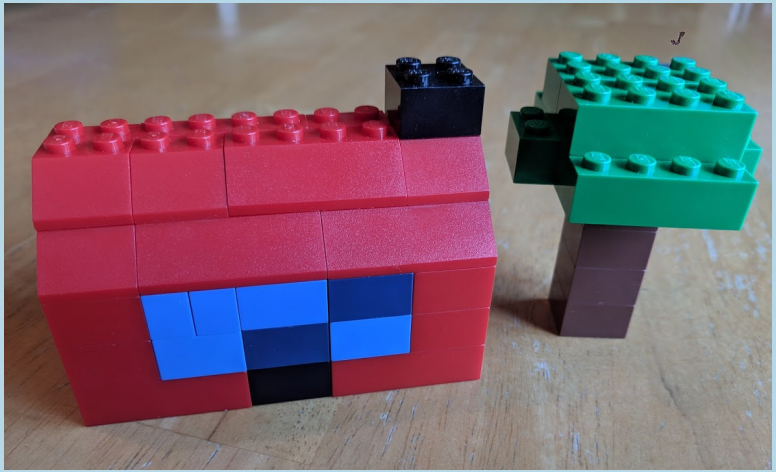
What's the TDD order again?

Is the tree the same size as the house? ✗ ✓



What's the TDD order again?

Is the tree the same size as the house? ✗ ✓ R



Let's Practice!

- Work together and write down a new test
- Build minimally pass the test
- Refactor
- Here's a couple if you get stuck:
  - *Is the tree the same size as the house?*
  - *Is the animal smaller than the person?*

New Requirements

Your new program needs to have:

- A person
- A house
- A tree
- An animal
- A vehicle

Review

- Let’s take a minute to walk around and look at everybody’s creations!
- Taking breaks and stretching your muscles (eyes, legs, arms, back) is important to stay healthy.

Apology

- You may have to break your wonderful new structures for the components for the next section.
- Don’t worry, there is more building to come!
- Please do take a photo. Send an tweet to your friends that you’re playing with LEGO! #jealous

Exercise 4 -Team Build

- Each table will come up with a large structure with lots of parts to build. (airport, zoo, amusement park...)
- You will be a software team, working in pairs, contributing to the large team structure (the build) at your table.

Decide on an idea

- Shh...don’t tell the other tables.
- (Jeopardy music...)

Break it into components

- Take a minute to write out a few of the components that need to be built (product backlog).
- Each pair will build a component through TDD.
- We will then add the sections to the big project. Make sure you don’t break your teams tests!
- Write your test on paper, fail, then add lego to make the test pass.

Demo time!

- A few teams will now present their creation.
- Please share the tests that you came up with and point out the solution.

Review

- Test-Driven Development / Design
- Pair Programming
- A bit of developer speak
- Experience working on a software team

Questions?

Bryan Beecham
@BillyGarnet
bryan@icebergideas.com
https://www.linkedin.com/in/bryanbeecham