Agenda

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

- Announcements
- Lecture
  - Test-Driven Development
  -

Announcements

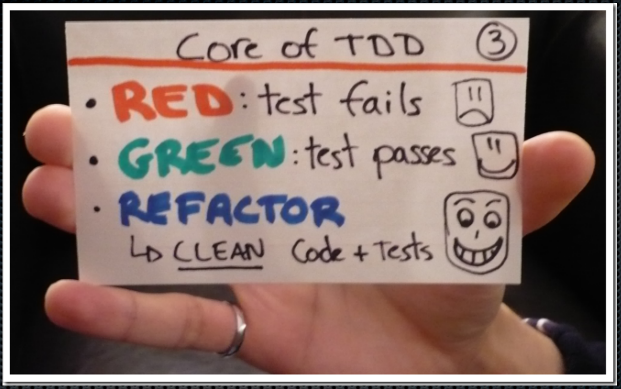
- Agile Tour Toronto
  - Looking for volunteers
  - I have 2 free tickets

Understanding TDD and Refactoring with LEGO

- Material adapted from Bryan Beecham
- @BillyGarnet

Why TDD

- It improves the lives of the users of your software.
- It lets your teammates count on you, and you on them.
- It feels good to write it.



The Mantra

Red - Green - Refactor

photo from doolwind.com

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 tree
  - An animal
  - A vehicle

Let’s Practice!

- Work together to come up with some new tests
- Keep building to minimally pass tests
- Don’t worry about Refactoring for now
- Here’s a few if you get stuck:
  - Is the house at least x inches/cm tall?
  - Is the tree the same size as the house?
  - Is the animal smaller than the person?

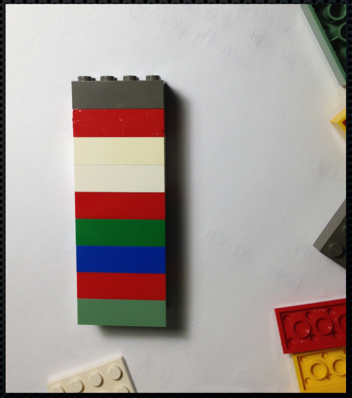
Refactoring

- Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.
~Martin Fowler

LEGO Refactoring

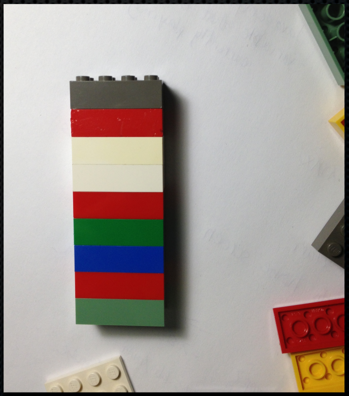
Build our program

- Stack bricks like this:
- gray-red-white-white-red-green-blue-red-green
- gray on top and green at the very bottom



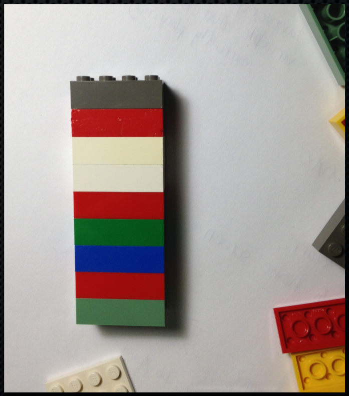
Class and Methods

- Your group of Lego bricks represents a **class**
- Each one of these colour bands represents a **method**



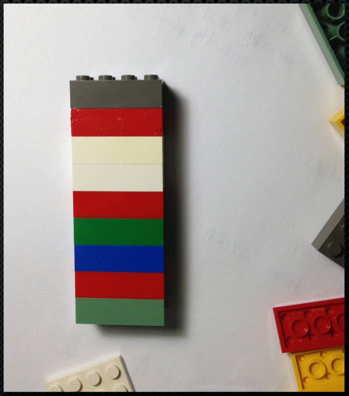
Duplication

- Notice how there are repeats of the same colour bands. This is duplication in our code. Let’s fix this!



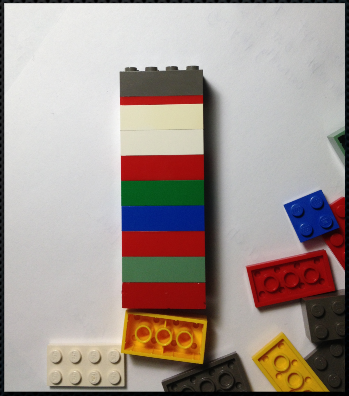
Extract Method

- We are going to start by extracting the method. We will replace the first red brick with a red plate.
- Put the structure back together and put the red brick on the bottom.
- This red plate tells the program to go to run the red brick code.



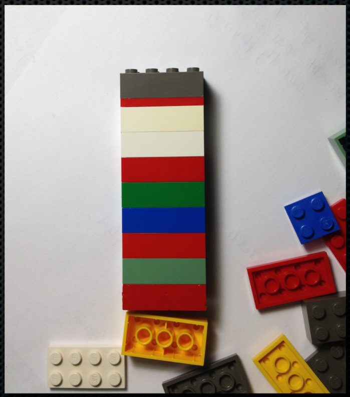
Extract Method

- We are going to start by extracting the method. We will replace the first red brick with a red plate.
- Put the structure back together and put the red brick on the bottom.
- This red plate tells the program to go to run the red brick code.



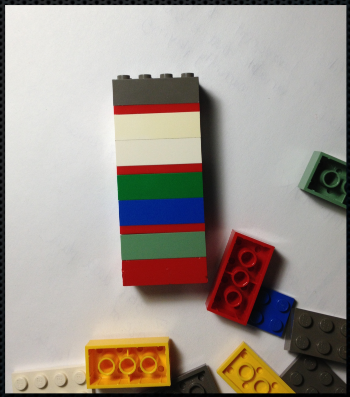
Remove Duplication

- Now let’s replace the other red bricks with the red plates.
- We don’t need to move these red bricks to the bottom since we already have that code there.



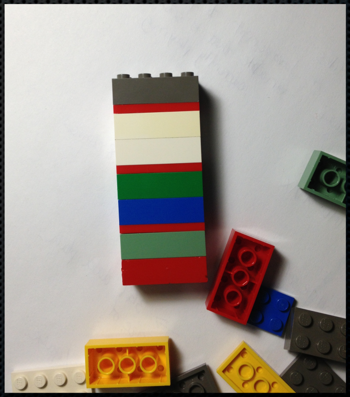
Remove Duplication

- Now let’s replace the other red bricks with the red plates.
- We don’t need to move these red bricks to the bottom since we already have that code there.



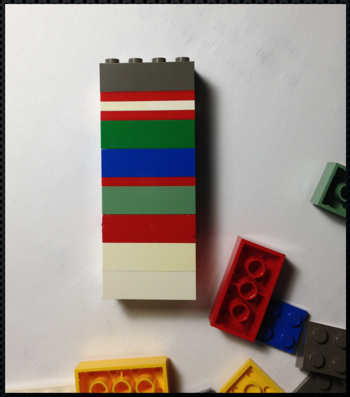
Keep Going!

- Let’s do (Extract Method) on that big white section.
- Even though it’s not repeated, it would be easier to read the program with that white blob at the bottom.

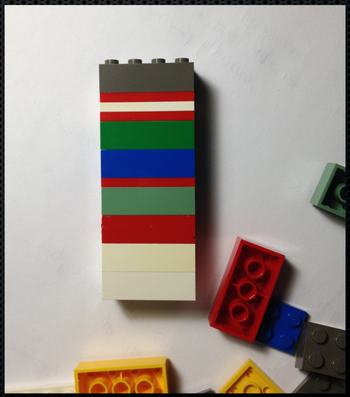


Keep Going!

- Let’s do (Extract Method) on that big white section.
- Even though it’s not repeated, it would be easier to read the program with that white blob at the bottom.

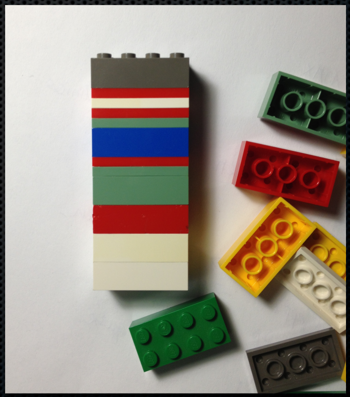


Any other duplication?



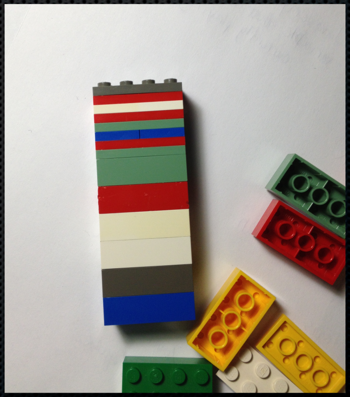
Readability

- I really don't like the way we are mixing big and flat pieces at the top of our program.
- Let's extract gray and blue as well



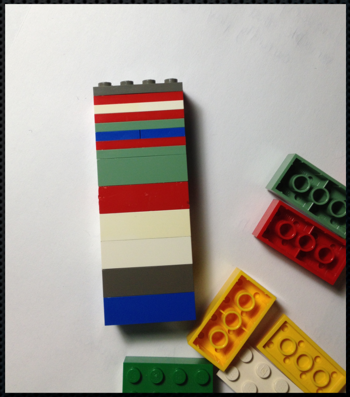
Readability

- I really don't like the way we are mixing big and flat pieces at the top of our program.
- Let's extract gray and blue as well



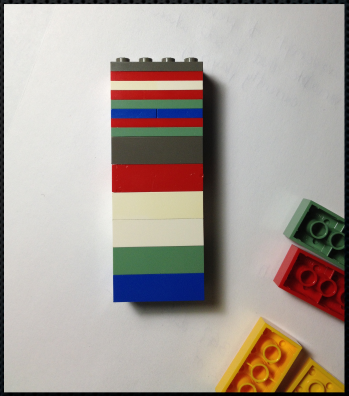
Clarity

- Remember one of the requirements for good software is that it's easy to update.
- Let's re-arrange these bottom bricks to match the order of the plates that are above them.

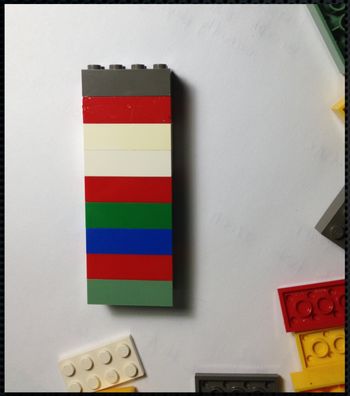


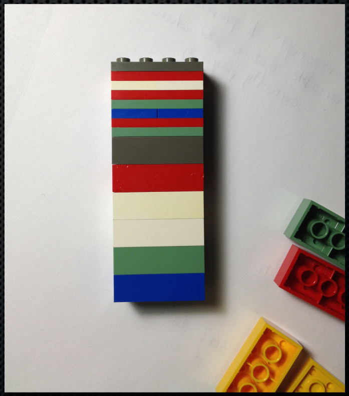
Clarity

- Remember one of the requirements for good software is that it's easy to update.
- Let's re-arrange these bottom bricks to match the order of the plates that are above them.



Before and After





Exercise 4 -Team Build

- Each team will come up with a large structure to build.
- You will be working in pairs, contributing to a large structure (the build).

Decide on an idea

- Keep it to yourselves for now

Break it into components

- Take a minute to write out each of the components that need to be built.
- Do this in sprints
  - 2 min for planning, 6 min for executing, and 2 for review
- Build in pairs and then integrate

Demo time!

- Each team will now present their creation.
- Please share a few tests that you came up with.
- Admire your work
  - Take a photo.
  - Upload to Twitter or Facebook.
  - Brag to your friends.

Review

- Test-Driven Development / Design
- Refactoring
- Pair Programming

For Sprint 1

- Planning Poker on user stories
  - Use the one story that you implemented from Sprint 0 as an anchor
- Have Product Owner prioritize stories
- Select user stories to be implemented
- Get details by talking to the Product Owner
- Create test cases (or plans) first

- All while using agile development