Time	~ 2 hours
Learning Goals	• Make your code more readable

Unrefactored Code vs. Refactored Code

Code should be easy to read and understand. It should obey the various good practice we talked about. Separations-of-concern, DRY, loose coupling etc. are all important concepts to implement in your code base.

But your code is going to suck the first time you write it. Even experienced developers write horrible code. It takes discipline and ongoing effort to improve code. When building a new feature, developers write code that is good enough to make the feature work. But once the feature work, specs should accompany the implementation code. Then, go back and refactor the code for better readability or greater speed. The test will help ensure that you didn't break your app or introduce unwanted bugs. Tests are important!

Objectives

Refactor Code

In this challenge, fork and clone from this repo. You will see the following files:

- `extract_variable.rb`
- `extract_method.rb`
- `introduce_parameter_object.rb`
- `unfactored_password_controller.rb`
- `refactored_password_controller.rb`

We will focus on the first three files for now. For each file, you will be given some code that needs to be refactored. There will be a link given in each file which will point you to a resource that will help you with the refactoring. After you are done with the refactoring, pair up with a buddy and review each other's code.

Note: do not follow the syntax given in the resource as they are not Ruby code. Understand the concepts and read the examples given in the resource. You should be able to understand at least 80% of the code given in the resource by now as you should already have a pretty good background with reading code. Then based on what you understand, refactor the code given with our charming Ruby.

A More Complicated Example

We have extracted some code from one of our in-house apps. Have a look at both `unfactored_password_controller.rb` and `refactored_password_controller.rb` .

Now based on what you understand from the code, discuss some of these questions with your fellow classmates and mentors:

- Which example was easier to understand?
- Does good code mean shorter code?
- How should you name variables and methods?
- Besides changing the code so it's more efficient / maintainable / easier to read, we have to be sure that we are not change the functionality of the code. How do you ensure that you're not changing the functionality?
- Anything else?