|  |
| --- |
|  |
| Refactoring |
| Improving the Quality of Existing Code |

# 

## Objectives

After completing this lab, you should:

* Know how to use refactoring tools in the IDE (like Rename and Extract Method).
* Clearly see the benefit of having working unit tests, and learn to rely on the tests as a feedback mechanism.

## Overview

Here is the bad news: the new developer you hired has written some terrible, atrocious code. No one can understand what it does.

The good news: at least there are unit tests to prove the code is currently working.

You job is to refactor the code and make it readable, while keeping it in working order.

# How To Start

1. **Open** the solution file in the **Before\Algorithm** directory of this lab.
2. **Run** the tests to make sure everything is in working order.
3. Start refactoring! The primary goal is to refactor the code in Algorithm\Finder.cs – as it stands the code is incomprehensible.

Tips:

* Start with simple **rename** refactors so you can better understand the abstractions you are working with. Rename any class or any variable.
* Move on to **extract** methods and making the code more modular.
* See if you can also eliminate switch statements and multiple exit points from methods.

Anything is fair game in this lab – create new classes, new methods, and rename tests. The only restriction is that the existing tests have to keep working. Lean on the tests and run them after every small change to make sure you are on the right path.

# How to End

You can stop when you feel the code is good enough – something you can come back to in 6 months and understand.

After the lab, we’ll compare notes and discuss the different approaches everyone took.