CSSE 375

Milestone 3

Kyle Bippus, Matthew Spurr, Jacob Schuenke, Tyler Shelton

Below is the table of refactorings that we aimed to complete for this milestone:

Smell	Location	Team Member
Duplicated Code	Generator	Jake Schuenke
Large Class	Generator	Jake Schuenke
Duplicated Code	PasswordViewController, NoteViewController	Matt Spurr
Data Class	Password, Note	Matt Spurr
Duplicated Code	WebDAVAPI	Tyler Shelton
Long Methods	WebDAVAPI	Tyler Shelton
Conditional Complexity	GeneratorViewController	Kyle Bippus
Dead Code	WebDAVAPI	Kyle Bippus

Duplicated Code in Generator Class

In the Generator class, there was a series of conditional statements that had very similar code, but not similar enough to directly Extract Method. Our solution was to use Extract Method in the bodies of the conditionals. This made the method shorter and easier to read.

Larege Class in Generator Class

In the Generator class, we had many methods that were used to test functionality. This would be a lot better in a class that actually uses test cases, so we moved those methods out of the Generator class.

Duplicated Code in PasswordViewController and NoteViewController

In PasswordViewController and NoteViewController, there is a method called prepareForSegue that is extremely similar, but cannot be directly extracted and pulled up. The two classes share a common superclass called CoreDataTableViewController. The biggest problem with the method in these two classes is that they each reference two other specific classes: Add*ViewController and *DetailViewController (where * is either Note or Password), and these also subclass one superclass called UITableViewController. The method calls within this prepareForSegue method are the exact same between the two; the only difference is the receiving objects, which are specific to either Note or Password. Because of this, we cannot Pull Up as we would otherwise like to.

Data Class in Password and Note Classes

The Note and Password classes can be considered Data Classes because they only have properties, getters, and setters. Unfortunately, we cannot change this without fundamentally altering the logic behind tracking individual Password and Note objects. The classes must remain this way because they subclass NSManagedObject, and subclasses are required to be objects just to store data, so that we do not need to interact with the tables directly.

Duplicated Code in WebDAVAPI Class

The duplicated code in WebDAVAPI was caused because there were two different methods to handle uploading and downloading from the server. The approach to get rid of the duplication was to just get rid of the longer and more problematic implementation.

Long Method in WebDAVAPI Class

Most of WebDAVAPI's funtionality was contained within a few. very large functions. Additionally, there were sections of each method that were never reached or done through several unneeded steps. By removing or consolidating these sections of code, as well as splitting up these large methods, we've eliminated this bad smell.

Conditional Complexity in GeneratorViewController Class

In the GeneratorViewController class, we have a method that adjusts the values of the sliders based on the values for the other sliders. Our method had four sets of "if...then" statements to get the maximum value for each slider. Our solution is to extract out the body of the conditional into its own method. Before calling it we create an array that contains all of the sliders, and then pass into the method the array minus the slider we want the value from. This way we can call our new method four times and have only one conditional insdie a separate method

Dead Code in Multiple Classes

Previously, we had many methods that were implemented at the beginning of the project, and as we built the system we left a lot of code commented out and never used. We cleaned up this code to make it much cleaner than before.