Code Smells and Refactoring

One of the core code smells that we encountered in Iteration 3 was the use of duplicate code. One specific instance of duplicate code was in our use of the menubarmenu (hamburger menu), in which the code was duplicated in all of the different activities. We realized that this was an unacceptable option, when we wanted to add a “Home” button into the menu and needed to make edits to each of the various classes to implement it in all of our activities. To fix this, we moved the instances of duplicate code into a single domain item with a static method to handle menu generation.

Another problem that we encountered in Iteration 3 was the use of long methods and classes. This got in the way of our ability to adequately unit test our application, and required us to break down larger methods and classes into smaller, more manageable code. Similarly, when we began refactoring, we realized we had a significant amount of dead, or commented-out, code. During the refactoring we removed this dead code, which allowed us to more clearly see the actual functioning code, and assisted us with development and refactoring generally.

The final problem that we encountered, which we are in the process of fixing, was using far too many constructors in our domain object classes. For example, in the Bird class, we were using a different constructor depending on which of the many attributes possible of being included for an individual bird might be included in the addition. This caused a few different bugs which cost us development time, so we worked on developing a builder class to handle the different cases.