Assignment 4 - Refactoring and Unit Testing
Due Friday, April 25th at 8:00pm

Purpose

1. Perform refactorings to improve existing code.
2. Use the xUnit framework to test code during refactoring.

Description

You are working on a complex ecosystem simulation. At the moment, you have developed part of a Fish class, and have been asked to refactor the code before adding to the functionality. In addition, there is a test suite which tests some of the Fish's functional requirements, so any changes you make need to pass all tests!

Tasks

You are to perform three refactorings to the provided Fish class. You need to maintain a report of the *detailed* steps you performed in your refactorings, as described in the **Report** section of this handout.

Refactoring #1: Replace Magic Numbers with Symbolic Constants (20 points)

In the constructor and *move* method, there are several fixed numbers hard-coded as initializations or in conditional statements. Since the parameters of a simulation are likely to change in the future, it would be best to consolidate these numbers, and replace them in the actual code with symbolic constants. You should replace a total of two symbolic constants in the constructor, and four in the *move* method.

Refactoring #2: Replace Conditional Calculations with Strategy (40 points)

In the *move* method, there is a complex conditional statement to determine how a fish should move, based on its size and hunger level. Perform the *Replace Conditional Calculations with Strategy* on this method, generating new Strategy classes and simplifying the *move* code.

After performing this, create a private method *updateMoveStrategy* in *Fish*, which sets / creates which strategy to use based on the hunger and size variables. This method should be called at the end of the *age* method in Fish. If done properly, all tests should still pass

Refactoring #3: Replace Hard-Coded Notifications with Observer (40 points)

There is a *FishReport* class, which is used to maintain reports on the fish's attributes. For the moment, *FishReport* is the only class the fish reports its attributes to, but a GUI is expected in the future, which will require similar capabilities. Rather than have a Fish class maintain instances of many different object for this type of task, the Observer pattern should be implemented instead. Refactor the code so that *Fish* and *FishReport* objects are part of an Observer pattern instead.

Report

You are to create a report detailing each refactoring above. This report should clearly describe each step of the refactoring, as shown in the slides on "Refactoring" and "Refactoring to Design Patterns".

For example, poor documentation would simply have:
"I performed the "Replace Conditional Calculations with Strategy" refactoring."

Good documentation indicates steps taken, and lines removed and added to the code:

1. I identified the conditional calculation in method *foo*
2. I created the class *Foo*
3. I created a method *bar* in class *Foo*
4. I removed the line "if (x=2) { x++;}" from method *foo*, and added "if (y=4) {y--;}" to method *bar*.
5. I ran the test suite. All tests passed. etc.

Make sure each refactoring is given its own section, and that you fully complete a refactoring before starting another.

Deliverables

You need to submit the following to moodle:

1. Your report as a PDF. Please do not submit in other formats.
2. A zip or tar.gz file containing your refactored code.

Provided Files

You are provided with five files total: *Fish.java*, *FishReport.java*, *Pond.java*, *FishTests.java* and *FishTestRunner.java*.
Do **not** modify *Pond.java*, *FishTests.java* or *FishTestRunner.java*
FishTests.java is a jUnit test case, and *FishTestRunner.java* is used to run the test case.
All files come pre-compiles.

To run the test suite from the command line, use the command:

```
$ java -cp .:lib/junit.jar:lib/hamcrest-core.jar FishTestRunner
```

If you need to compile, use the command

```
$ javac -cp .:lib/junit.jar:lib/hamcrest-core.jar FishTestRunner
```

You should make sure you can run the test cases prior to starting any refactorings! If you have any trouble, email Dana or Liz, or post on Piazza, or ask in class.