|  |  |
| --- | --- |
| **CMSC 131** |  |
|  |

Assignment:     Project #7

Due Date:     Monday 12/10, 11:00PM

Open/Closed policy:     CLOSED

**Fish Club**

****

The first rule of Fish Club is:  *You do not talk about Fish Club.*

The second rule of Fish Club is:  *You DO NOT talk about Fish Club.*  (After all, it's a closed project...)



Other rules:

* Each fish travels in one direction until it encounters a rock, at which point it turns randomly to move in a direction where there is no rock.  Plants and rocks don't move.
* If one fish runs into another fish, the larger one eats the smaller one, accumulating it's mass.
* If a fish passes over a plant, it eats part of the plant -- the fish gets bigger; the plant gets smaller (or disappears if it was already pretty small.)
* Over time, fish shrink (unless they eat something) and plants grow (unless something eats them).
* If a fish gets too big, it will explode, creating 4 to 8 smaller fish.  The combined mass of the smaller fish is equal to the mass of the original.
* If a plant gets too big, it will explode creating 2 to 9 smaller plants.  The combined mass of the smaller plants is equal to the mass of the original.
* If the size of a fish or a plant reaches zero, it is gone forever.

**Objective**

To practice exception handling; practice using the ArrayList class; practice using for-each loops; practice using two-dimensional arrays.

**Overview**

*Fish Club* is a simulation of a very primitive "Ecosystem."

If you watch *Fish Club* in "slow motion", you can see some interesting interactions and patterns occurring among the pond's inhabitants.  If you slide the speed slider all the way to the right (maximum speed) and watch patiently, you may observe some interesting long-term patterns in the populations and locations of the fish and plants.  The interactions can be very complex!  It's fun to try to formulate "theorems" for what sorts of situations will yield an "ecosystem" that can survive for a very long time.  Can you think of any scenarios where you can PROVE that the fish will never die?

You will be writing portions of the Model, Fish, and Plant classes. We have distributed skeleton versions of these classes to your CVS repositories -- look in the package called "fishPond".

**Note that the main method which runs the project is in the Controller class.**

**Specifications**

See the [JavaDoc](http://docs.google.com/doc/index.html) for a complete description of the project.   This information is also duplicated in the source code distribution that you have received.  A few items of great importance:

* **Your work must go in the "fishPond" package.**
* **There are a few *private* methods that you must implement, but that are not described in the JavaDoc -- please see the instructions embedded in the source code distribution!**
* **Please note that reading one of the "Summaries" in the JavaDoc does not show you all of the information about that particular member -- you have to read the full descriptions of each member to see all of your instructions!  (Or read the comments in the source code files.)**

**Style Hint**

We have provided a  lot of built-in named constants.  For example, the two-dimensional array that keeps track of where all the rocks are uses the values "Model.ROCK" and "Model.WATER" to keep track of where there are rocks and where there is just water.  Even though you can easily see that these constants represent the values "true" and "false", it is a HUGE mistake to simply use "true" and "false" in your code when differentiating between rock and water.  That defeats the entire purpose of using symbolic constants:  the symbolic constants make the code more readable, and also facilitate possible changes to the project design at some later time.  You will lose style points if you don't use symbolic constants properly!

**Requirements**

* Before submitting, be sure that your project does not produce any unexpected text outputs. (If you have included trace output using System.out.print, you must remove those lines before submitting.) Extraneous text output will cause your project to fail lots of release tests on the submit server.
* As with all programming assignments:
  + You must use meaningful variable names and good indentation.
  + You must avoid code duplication, by calling appropriate methods (rather than cutting and pasting code).
  + Because we are using an automatic testing system, it is important the you adhere closely to the specifications. In particular, **please do not change the names of any of the classes you are to implement, and use signatures given above for your methods**.
* **You are encouraged to use for-each loops as much as possible when cycling through the elements in your ArrayLists.** You will lose points if you don't use them most of the time!
* You must use **Random131.getRandomInteger()**, and not Java's built-in random number generator, for all random numbers.
* Follow the instructions provided in the JavaDoc very carefully!  If you don't, you will not pass our tests.

[Web Accessibility](https://www.umd.edu/web-accessibility)