**APCS Spring, 2002**

**Assignment 4:  Working with Multiple Interacting Classes**

**Files:** [**environ.h**](http://docs.google.com/cs102labs/lab4/environ.h)**,** [**environ.cpp**](http://docs.google.com/cs102labs/lab4/environ.cpp)**,** [**fish.dat**](http://docs.google.com/cs102labs/lab4/fish.dat)**,** [**fish.h**](http://docs.google.com/cs102labs/lab4/fish.h)**,** [**fish.cpp**](http://docs.google.com/cs102labs/lab4/fish.cpp)**,** [**FishDisplyG.h**](http://docs.google.com/cs102labs/lab4/FishDisplayG.h)**,**[**FishDisplayG.cpp**](http://docs.google.com/cs102labs/lab4/FishDisplayG.cpp)**,** [**nbrhood.h**](http://docs.google.com/cs102labs/lab4/nbrhood.h)**,** [**nbrhood.cpp**](http://docs.google.com/cs102labs/lab4/nbrhood.cpp)**,**

[**position.h**](http://docs.google.com/cs102labs/lab4/position.h)**,** [**position.cpp**](http://docs.google.com/cs102labs/lab4/position.cpp)**,** [**randomgen.h**](http://docs.google.com/cs102labs/lab4/randomgen.h)**,** [**randomgen.cpp**](http://docs.google.com/cs102labs/lab4/randomgen.cpp)**,** [**simulate.h**](http://docs.google.com/cs102labs/lab4/simulate.h)**,** [**simulate.cpp**](http://docs.google.com/cs102labs/lab4/simulate.cpp)**,** [**utils.h**](http://docs.google.com/cs102labs/lab4/utils.h)**,** [**utils.cpp**](http://docs.google.com/cs102labs/lab4/utils.cpp)**,** [**fishsim.cpp**](http://docs.google.com/cs102labs/lab4/fishsim.cpp)

**Purpose:**

The purpose of this lab is to learn to work with a moderate size program involving several interacting classes.  In this example, we will work with classes that already have been completed to change the functionality of the overall system.  The overall goal is to modify the fish simulation from the Marine Biology Case Study so that fish can breed and die with certain probablilities.  The code as given can be compiled and run, so you get a sense of how the simulation works before you start your modifications.  You should add all the above files (except fish.dat) to the project.  Also add the CMUgraphicslib.lib file to the project. You can use the fish.dat file to test.

**Preparation**:

* 1. Read the Marine Biology Case Study.

**Tasks:**

* 1. Modify the class Fish as follows
     1. add private data members myProbDie and myProbBreed which represent the probability of dying and breeding, as described below
     2. add a private data member myAge which keeps track of the fish's age, relative to the number of steps in the simulation
     3. add a public member function Age which returns the fish's age
     4. add a private data member which is a random number generator for the fish and remove the local variable random number generator from the Move member function

Each Fish member function which needs to generate random numbers can use this random number generator

* + 1. add a new Fish constructor so that the three new data members are correctly initialized from three parameters

The existing constructors should be kept, but the implementation changed to give reasonable default values to the new data members  
All constructors will need to initialize the random number generator by setting its seed, if a specific seed is wanted.

* + 1. add a new private member function Breed which is called whenever the fish is breeding age
    2. add a new public member function Act which replaces Move to determine what happens to a fish on each step of the simulation

Move should be made into a private member function which can be called by Act.

* 1. Modify the Environment class as follows
     1. Modify the implementation of the constructor so that in addition to the information already read in about each fish, it reads in values for the three new fish data members.

Fish data file format should be changed so each line includes, after the fish position, its age, probability of breeding, and probability of dying

* + 1. Modify the AddFish function so that in addition to a position, it takes an age, and the two probabilities as parameters and applies them appropriately.
    2. Add a new public member function KillFish which removes the fish at the given position

KillFish will be the inverse of AddFish  
KillFish can be called by the Fish member function Act when a fish dies

* 1. Modify the Step member function in the class Simulation so that each fish determines its next action by calling Act instead of Move
  2. Modify the fish.dat file or write your own input file to match the specifications needed for your revised code.

**Hand in:**

* 1. Copies of your finished fish.h, fish.cpp, environment.h, environment.cpp, simulation.cpp files

The new version of the simulation should have fish age one unit each step.  The fish can be given different initial ages from the fish data file.  When a fish is between 10 and 20 time units old it can breed as follows:  for each empty neighboring space, the fish produces an offspring in that space (using a call to AddFish) with the probability myProbBreed.  After breeding, the fish moves as before, if there is an empty space to move into.  The new fish which are created by breeding have zero age and the same probabilities of breeding and dying as the parent.

In addition, on each step of the simulation, there is a certain probability that a fish will die, given by myProbDie.  The simulation determines whether a fish will die before it breeds or moves.  If it dies, it cannot breed or move and it is removed from the simulation (by a call to KillFish).

After you have your fish breeding and dying properly, experiment with values of the probabilities of breeding and dying to see if you can get a population which stays relatively stable most of the time for simulations of a few hundred steps.  It should not grow to fill the whole space nor should it die out, at least not on every run.

Extra credit 1:  modify the way the fish die so that the probability of dying is dependent on the age of the fish as well as the data member myProbDie in such a way that older fish are more likely to die.

Extra credit 2: Adapt the simulation so that there are different species of fish.  Fish of one species all have the same probability of breeding and dying, but different species can have different probabilities.  Change the Fish::Showme function so that the character returned depends on the species of fish instead of the ID of the fish.  Change the display so that it shows different species as different colors.

Extra credit 3:  Change the display so that younger fish are very small and older fish are larger, but never outside the bounds of their cell in the grid.