Big Program 2: Wa-Tor

In this project you will complete and extend an implementation of the Wa-Tor population dynamics simulation devised by A. K. Dewdney in a Scientific American article entitled "Computer Recreations: Sharks and fish wage an ecological war on the toroidal planet Wa-Tor" (December 1984).

Wa-Tor is implemented as a two-dimensional grid showing fish, sharks and empty water. Swimming past the edge of the grid results in the creature appearing on the opposite side. Sharks eat fish and may also starve. Sharks and fish move, reproduce and die according to simple rules. From these simple rules complex emergent behavior can arise.

> "Time passes in discrete jumps, which I shall call chronons. During each chronon a fish or shark may move north, east, south or west to an adjacent point, provided the point is not already occupied by a member of its own species. A random-number generator makes the actual choice. For a fish the choice is simple: select one unoccupied adjacent point at random and move there. If all four adjacent points are occupied, the fish does not move. Since hunting for fish takes priority over mere movement, the rules for a shark are more complicated: from the adjacent points occupied by fish, select one at random, move there and devour the fish. If no fish are in the neighborhood, the shark moves just as a fish does, avoiding its fellow sharks." (Dewdney, 1984).

For the fish

1. At each chronon, a fish moves randomly to one of 4, adjacent unoccupied squares. If there are no unoccupied squares, no movement takes place.
2. At each chronon, each fish ages. Once a fish's age exceeds a certain number of chronons (fish_breed parameter) it may reproduce. Reproducing is done by leaving behind a new fish as it moves to a new square. Both the original fish and new fish have their age reset to zero.

For the sharks

1. At each chronon, a shark moves randomly to an adjacent square occupied by a fish. If there is none, the shark moves to a random adjacent unoccupied square. If there are no free squares, no movement takes place.
2. At each chronon, each shark ages. Once a shark has survived a certain number of chronons (shark_breed parameter) it may reproduce in exactly the same way as the fish.
3. If a shark moves to a square occupied by a fish, it eats the fish and a certain amount of energy. When this energy is depleted then the shark dies. This is implemented by resetting the time since a shark last ate to zero. When the time the shark last ate reaches a certain time (shark_starves parameter) then the shark dies.

What are the parameters for the longest running simulation you can find?

Program Grading: GradingBigPrograms.pdf

Working Together

BP2 is an individual assignment and so there are not teams.

Acceptable

With others you are encouraged to discuss:

- general algorithms
- the behavior or syntax for specific Java programming constructs

NOT Acceptable

With others you are NOT allowed to share code for this assignment in any form. All of the code that you submit with this assignment should be conceived-of and typed into the computer by you, or provided to you through the CS 200 website.

AVOID leaving any code in publicly accessible places, like: outside the L drive on a lab computer, on an unsecured computer, or on an unsecured website. Remember, all Piazza posts that include assignment code fragments must be marked private (visible only to Instructors).

Academic Integrity

Milestone 1: Getting Started

In this milestone, setup the project, read the code provided and implement a few short methods.

1. Create a new project in Eclipse. (WaTor would be a sensible choice for the project name.)
2. Download Config.java, put in your project src folder and review the contents to become familiar with the constants.
3. Download WaTor.java and TestWaTor.java and save them in your project src folder.
4. ~~**Read through the files to see what code is provided to you. A key skill is to be able to read and extend someone else's code rather than write it all yourself. Make sure if a method exists you call it rather than writing your own method for the same purpose.**~~
5. ~~Read and comment the testing methods: `testClearMoves`, `testEmptyArray`, `testCountCreatures`.~~
6. Write the method bodies of the following methods: ~~`showFishAndSharks`~~, ~~`placeFish`~~, ~~`placeSharks`~~, and ~~`countCreatures`~~. Note that placeFish and placeSharks have very similar implementations.
7. Modify the code in the main method to call these methods.
8. Some sample output: M1a.txt, M1b.txt, M1c.txt (with DEBUG constant true), M1d.txt
9. Submit your **WaTor.java** and **TestWaTor.java** files to zyBooks for feedback and grading. We recommend you complete commenting and styling your code for each milestone as described in the Program Grading section of this document. Your highest scoring results prior to 6:00pm or (11:59pm with a 10% deduction) **Wednesday, April 11th** will be recorded as your grade for this weekly milestone.
   Note: We may use different values in the Config.java for testing.

Milestone 2: Simulation Runs

In this milestone, complete the methods that get the simulation to run. There are many methods already completed. aFishMoves, sharkMoves, etc. Read the code to understand how they work and call them rather than writing the code yourself.

1. Add comments within the following test methods and headers to describe the tests: `testUnoccupiedPositions`(), `testChooseMove`(). Implement and comment the testFishPositions() method.
2. Implement the following methods: ~~`unoccupiedPositions`~~, ~~`chooseMove`~~, `fishSwimAndBreed`, `fishPositions` and `sharksHuntAndBreed`. Note that fishPositions is very similar to unoccupiedPositions and sharksHuntAndBreed is very similar to fishSwimAndBreed but simply adapted for shark behavior.
3. When the simulation is working when advancing 1 chronon at a time, then implement the feature that the user can type the number of chronon to run before displaying the ocean again.
4. Example output M2a.txt, M2b.txt (with DEBUG constant true), M2c.txt
5. Submit your **WaTor.java** and **TestWaTor.java** files to zyBooks for feedback and grading. We recommend you complete commenting and styling your code for each milestone as described in the Program Grading section of this document. Your highest scoring results prior to 6:00pm or (11:59pm with a 10% deduction) **Thursday, April 19th** will be recorded as your grade for this weekly milestone.
   Note: We may use different values in the Config.java for testing.

Milestone 3: Saving, Loading and Charting

In this milestone implement the saving and loading of simulation parameters and the creation of a population chart. The population chart shows how the numbers of fish and shark change over time.

1. Implement the methods `saveSimulationParameters`, `loadSimulationParameters`, and `savePopulationChart`. See header comments and comments in the main method for details on how these methods should work.
2. Complete the main method to collect history information, call these methods, and handle thrown exceptions appropriately.
3. Thoroughly test your code.
4. Example output: M3a.txt, chart1.sim, param1.sim, M3b.txt, chart2.sim, M3c.txt, M3d.txt, param3.sim, chartZ3.sim
5. Submit your **WaTor.java** and **TestWaTor.java** files to zyBooks for feedback and grading. Make sure your files are completely commented and styled as described in the Program Grading section of this document. Your highest scoring results prior to 6:00pm or (11:59pm with a 10% deduction) **Thursday, April 26th** will be recorded as your grade for this weekly milestone.
   Note: We may use different values in the Config.java for testing.

Change Log

Initial Version: 2018-4-2

Project designed by Jim Williams