JavaLife

David Robertson

CS 1632 - DELIVERABLE 5: Performance Testing Conway's Game of Life

To profile this app, I used vmsim to sample hotspots while running all of the possible commands. I then looked into the code for the hotspots to see what was intentional, such as sleeps between printing images, and what was unintentional.

While profiling the game, the method that stood out the most in terms of expenses seemed to be the MainPanel.convertToInt(int x) method. After looking at this method, it appears that the method is completely useless. After looping unnecessarily, it simply returns the int that was originally passed into the method. I wrote pinning tests for positive maxint, negative maxint, and 0. I decided that my first refactor should be an attempt to simply pass the int back to the caller. My first attempt to refactor ended up with the tests for the positive passing, as well as the zero passing. The negative however, failed. Upon looking over the method again, I discovered that the reason appeared to be because of the way the concatenation occurs in the method. It was causing a number formatting error due to the concatenation of a negative number at the end of a string. For example, it could be 00…0000000-1. This means that my initial thought was wrong. The original version of this method will crash if given a negative number. This means that if given a negative number, I had to ensure that the method exits in error. I left in the test that failed, and created a new test designed to ensure that a number formatting exception is thrown.

The next largest hotspot I found was the cell.toString method. Upon examining the code, it seemed that it was running an unnecessary loop. After running, it seemed to simply check the first value of the string that it was creating. I took out the loop to simplify this method. I then wrote tests to make sure that the output would still be the same for alive and dead, with and without the use of the constructor to set the Boolean.

The only hotspot I found after these modifications was with the sleep method. I chose not to do anything about this one however, because the run continuous already runs at a rate I can barely keep track of. I assume this hotspot is intentional to keep that speed down.

Images and pinning tests available at <https://github.com/RobertsonDavid/SlowLifeGUI/tree/master/test>

Refactored code available at

<https://github.com/RobertsonDavid/SlowLifeGUI>