N Queens Reflection

Charles Bailey

04/25/2015

CMPT 390

One of the earliest programs I had to write for a Computer Science Class was a program to determine where to place N number of queens on a chessboard so that none of them could capture each other. This was a program created in CMPT 202, where I was just getting introduced to data structures and applicable problem solving. While this program may not be as impressive as many of the other programs I have done for Upper Division Computer Science Courses, It was one of the first steps of learning how to implement an algorithm and testing it with unit tests. The algorithm needed a few methods to perform basic functions like clearing out a board, displaying the board in a command window, setting a Queen on the board, and removing a queen from the board. The hardest part of writing this code was being inexperienced with problem solving and understanding Java, since I had only taken the introductory computer science course. I had to get help to understand the importance of having methods to perform certain functions in a program, and how to apply the theoretical algorithm into an actual program.

Once the methods that did basic functions were written, I had to implement an algorithm to determine if another queen could capture a queen. To do this, I had to have a for loop check and see if there were any other queens in it. Once the method worked for columns, I had to think about how to check diagonally. This involved two for loops check right and left of the current queen chess piece. The entire method was written as a private Boolean method to return true or false. I also had to implement a method for placing a queen and use the method for determining if a queen was under attack. Once this method was completed, the main method was created to display the board and show a person if there was a solution for a board of size N. In addition to learning how to problem solve and implementing an algorithm, the program also had to pass a unit test. I had some experience with unit tests in the introductory course, but did not have a good grasp on them. As I wrote the N-Queens problem, I had to periodically run the unit tests to see if my program was working. I was glad when some of the tests were passing, and frustrated when other tests were still failing. Working with unit tests taught me that you can’t expect your program to function perfectly the first time you write it, and that it is alright to fail a few unit tests as you write your solution one step at a time. It is easy to create something and say it works when it does what you want it to, but sometimes having a unit test program can challenge you to rethink how you are solving the problem at hand. Once all the unit tests were being passed by my solution, I felt proud of my accomplishment because I had solved a challenging (at the time) problem and it passed the unit tests. When looking back on the program, I learned about persevering through implementing a tough algorithm, how to temporarily use other ways of getting the methods to work, and how you don’t need to pass all of the unit tests at once to make progress.