Test Driven Development Plan

Definition

“Test-driven development” refers to a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design (in the form of refactoring).

It can be succinctly described by the following set of rules:

* write a “single” unit test describing an aspect of the program
* run the test, which should fail because the program lacks that feature
* write “just enough” code, the simplest possible, to make the test pass
* “refactor” the code until it conforms to the simplicity criteria
* repeat, “accumulating” unit tests over time

Using the definition of Unit testing [here](https://www.xenonstack.com/insights/what-is-unit-testing/#:~:text=Unit%20testing%20is%20a%20type,and%20produces%20a%20single%20output.)

Junit has been used to perform the following Unit tests in netbeans.

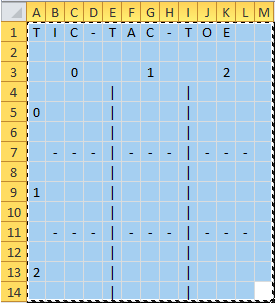
Unit Tests:

1. Draw Game Board
2. Display markers from coordinates
3. Accept 2 Human Opponents
4. Accept 1 Human Opponents
5. Accept 0 Human Opponents

[UNIT TEST 1 ]Draw Game Board

The program will draw a blank game board , working our way to successfully being able to drawn a game board of any size.

Goal :



Pseudocode

For a given number of row

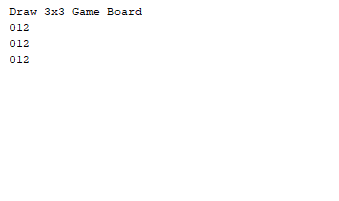
For given number of columns

print cell

if col not last print vert line

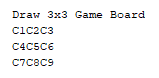
if row not last print horizontal line

initial Test outcome :



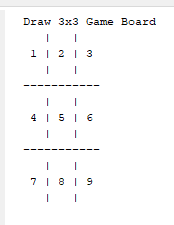
Test 2

Draw a 3x3 Game Board: Draws 3 columns for 3 rows , each cell in a row numbered 1 to 3



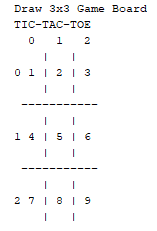
Draws 3 columns for 3 rows, for each cell and order value is called from a list of values

Test 3



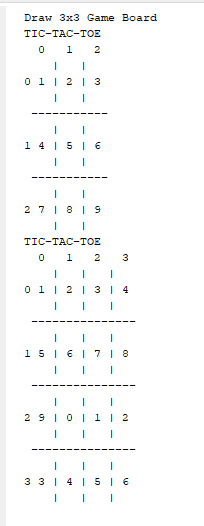
Draws 3 columns for 3 rows , for each cell an order value is called from a list of values , also some added templates for the game board

Test 4



Successfully draws a 3 by 3 game board, each cell is populated with a value from an ordered array. The template is as described in the requirements. The values in the cells are left in as a template for player markers .

Test 5



Successfully draws both a 3x3 and 4x4 game board game with the given number of rows and columns, each cell is populated with a value from an ordered array. The template is as described in the requirements. The values in the cells are left in as a template ready for player markers.

At this point the Acceptance test has been satisfied for this unit test.

[UNIT TEST 2] Display markers from coordinates

The Program will draw markers in the correct position on the board based on some coordinates, only valid inputs allowed. Only valid moves are allowed

*Pseudocode*

Get input

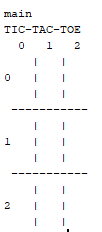
Check input string is valid coordinate

Check coordinate is a valid move

Add Marker to board

Draw board

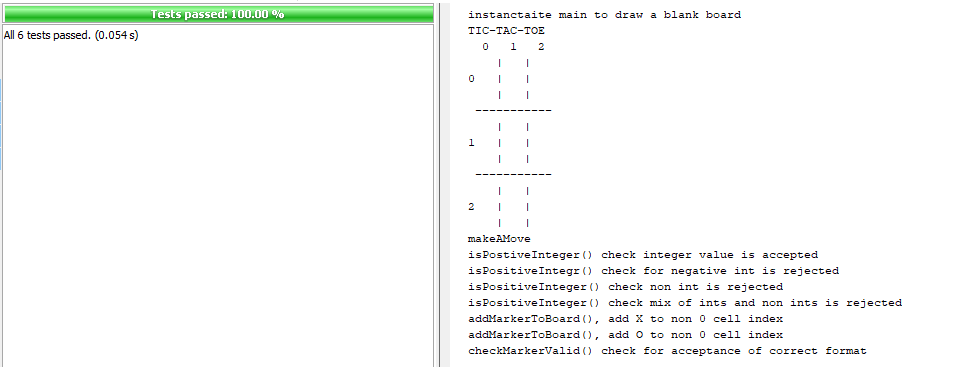
Test 1:



Some initial functions for input checking and , a blank initialisation of the game board

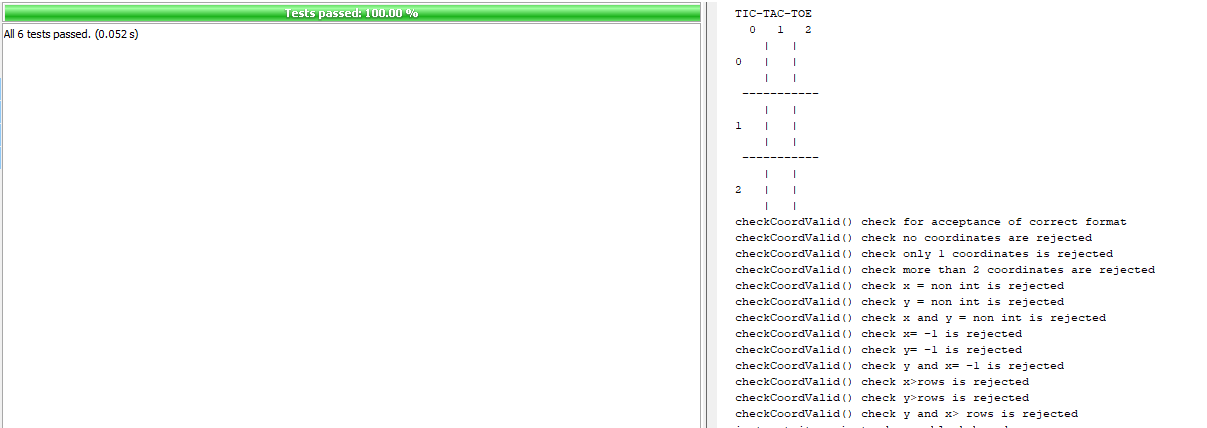
Test 2

Using JUNIT I have performed coverage test for the various input checks



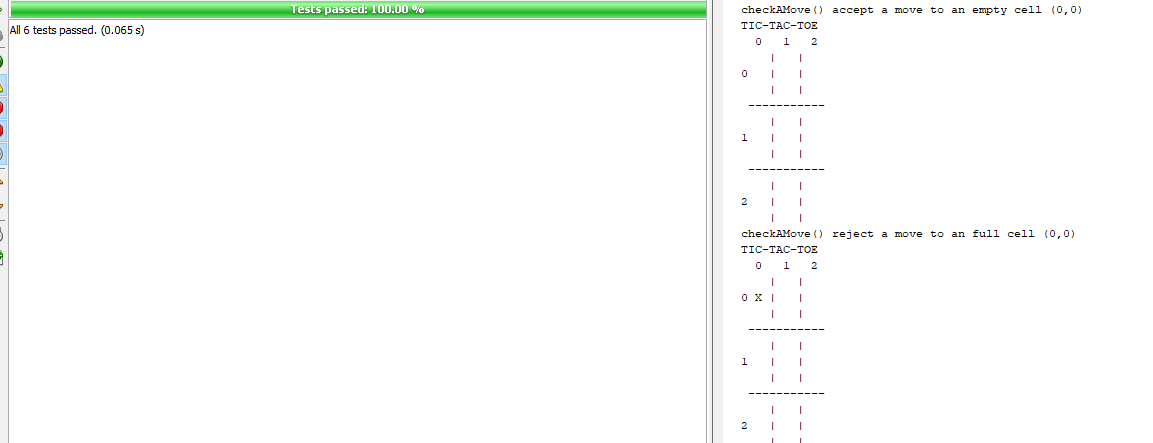
Test 3

Check that checkCoordValid () checks console input against valid coordinates for the board.



Test 4

Check that a move can be made by checking if the board already has a marker in the position indicated by the coordinates.



At this point the Acceptance test has been satisfied for this unit test.

[UNIT TEST 3] Accept 2 Human Opponents

The Purpose of this test is to build and test the run time routine to allow for 2 humans to play several moves.

Pseudocode:

Args(row = 3 col = x humans = 2)

Draw board

Display PLAYER message + token

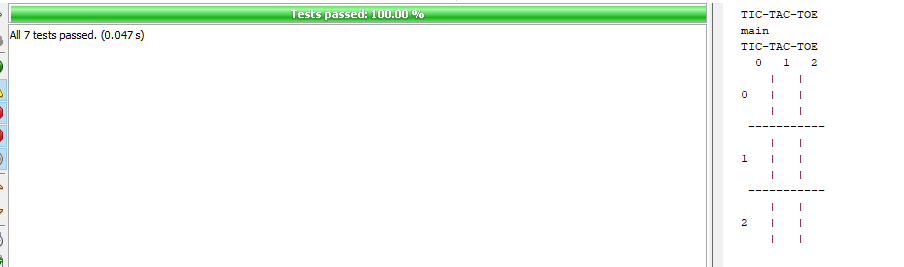
Wait for input

Accept input

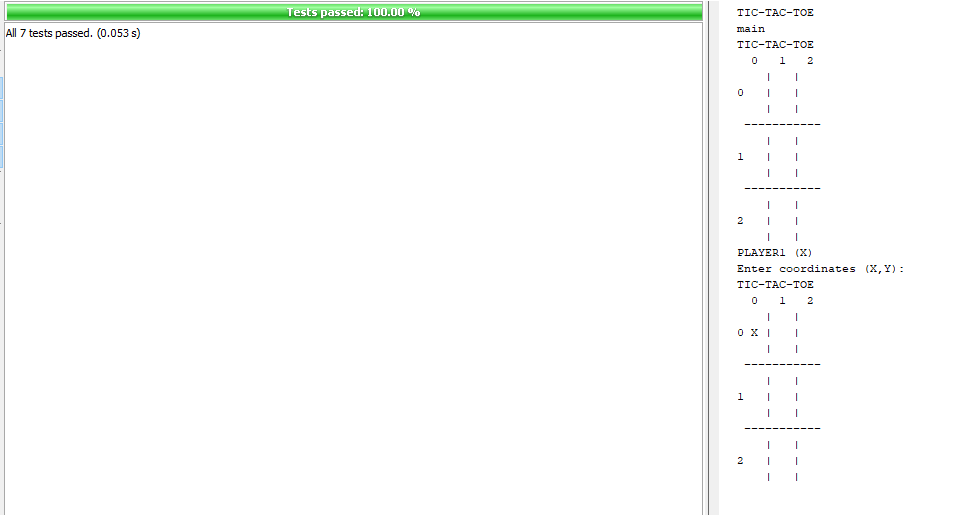
Update board

Test 1

Initial Runtime routine implement with some untested player input capture routine



Test 2



Used Junit to pass Player 1 ‘s first move and displayed it on the board.

[UNIT TEST 4] Accept 1 Human Opponents

[UNIT TEST 5] Accept 0 Human Opponents