Table of Contents

This document contains descriptions and reasons for the tests implemented in our code base so far. The tests test the code we have written that we intend to present for this iteration.

The test code classes this document will cover are:

1. TestBoard.java
2. TestGame.java
3. TestTile.java
4. TestToken.java

TestBoard.java

* testConstructor()
  + This method is designed to test that the constructor properly instantiates a new object and all accessible class variables.
* testGetToken()
  + This method tests that, given some board with some token on some tile, when we call getToken() with the proper variables we can receive the indicated token.
* testGetTile()
  + Tests in the same way testGetToken() does, but for a tile instead of a token.
* testResetBoard()
  + This method tests that the board was reset. In doing so, it also tests that we are capable of correctly flipping tokens and removing tokens from play (and then ultimately undoing such things).

TestGame.java

* testConstructor()
  + This test that the constructor works properly, instantiating an object in the proper fashion. It also tests, as possible, that all class variables have been correctly instantiated.
* testGameLogic()
  + This method tests the various pieces of the logic of the game – moving tokens, flipping tokens, attacking with tokens, ensuring that after a move the active player is set to the other player, etc. These pieces of logic could be tested individually, but we wanted to keep the game flowing in one test case so that we wouldn’t have to create a new board/etc for each different piece of logic we were testing.
  + The specific methods being tested are switchPlayer(), flipToken(), moveToken(), isValidMove(), and isValidAttack(). They are tested for ‘correct’ and ‘incorrect’ inputs – e.g.; isValidMove() is tested for cases where it returns true as well as false, etc.
* testGettersSetters()
  + There are also many getters and setters tested – to ensure that they were implemented correctly, as well as to increase code coverage. Usually getters/setters don’t need to be tested, but we decided to do so to keep with the high standards of code we’ve set for ourselves.
  + To keep the test code simple (and uncluttered), all tests for getters and setters have been put together in one testing method. This felt appropriate to us as all the same tests are conducted, simply in order of one after the other.

TestTile.java

* testMethodName()
  + Description of test method.

TestToken.java

* testMethodName()
  + Description of test method.