Apocalypse Test Document

To the test the functionality of the game and to test the correct function of all the interface there are JUnit tests and manual console based tests. There are JUnit tests for GameSet, CustomGameSet, DefualtGameSet, Piece, Coord, AIPlayer, HumanPlayer, Board, GameDynamics and Movement. To test each of these classes with their corresponding JUnit tests a new folder must be created. Then hamcrest-core-1.3.jar and junit-4.12.jar must be downloaded, and placed in the folder. Both the test code, code being tested and any other code that is used in them put into this folder as well. All the code must then be compiled, using the command javac -cp .:junit-4.12.jar:hamcrest-core-1.3.jar *.java . Then the test can be run using the command java -cp .:junit-4.12.jar:hamcrest-core-1.3.jar org.junit.runner.JUnitCore <Test class> . Whether the tests pass or fail will be printed to the terminal and can be read to see if the game is functioning correctly.

Other Console Based Tests

Much of the chess logic is difficult to test using j-unit due to the fact that the only metric we can use is something that we have written already. The following tests are “sanity checks” that print out information pertaining to the game to the console.
Ensure all tests are compiled using the command javac *.java in the Tests folder
Run the following tests in the test folder using the normal java <Test class> command. The following is a brief description of each:

TestPieceMovement

-This just test to see if the available moves generated by the Movement class make sense with what is on the board

TestBoardWithGameSets

- This is just to test if the board will properly update with different created Gameset objects to ensure that the board will update and display correctly.

TestGameDynamicsMovement

- This shows the analysis of the GameDynamics and the moves, as well as the AI level 0 choice

TestGameDynamicsStalemate

- In this test you must move the piece forward to test if the game dynamics logic engine correctly determines a stalemate condition

TestGameWhiteWin

- This game the user must input a move, but the game can be won in 1 move. This is just to test the gamedynamics win condition logic

TestAIMovement

- This is another “sanity check” which tests to see if available moves generated by the AI movement generator make sense. This can be run several times and the choices should be different.

TestAIVsAI

- This test prompts the user to run a number of iterations for running the AI versus the AI at different difficulty. This required modifications to the game dynamics which should be included in the test folder also. At the end of the iteration it prints out a list of the recorded termination of the game: either stalemate, draw, white, or black, representing white winning or black winning correspondingly