Zac McClain
Jon Weldon
David Grisham
Arnold Filliat

Testing Documentation:

testMove(){

This test is designed to test the move functionality of each player. The test does this by creating a new testPlayer of type SoccerPlayer. It then sets its x and y coordinates so that the players location is known. The test then calls the move method in Player on testPlayer with to move the player to the origin it then test that the player has moved to the origin.

}

testPass(){

This is a test designed to test the functionality of the pass method found in Player. The test starts by setting a player's boolean ball variable to true. The test then calls the player's pass method and with another player as an argument to pass the ball to them. The test then counts (in a iterative for loop) all of player with a ball with a ball and all the players without one. Using assert statements the game then checks that the correct player has the ball, That only one player has a ball and the rest do not.

}

testThrowIn(){

This tests the throwIn method found in SoccerGame. First this test puts all of the player form both teams into one ArrayList called palyers and then creates a throwingPlayer sets the player to be at index 0. Next if uses a for loop to insure that if the function is called over and over it will still be reliable. Inside of the for loop the throwingPlayer is given the ball calls the throwInBall() method and sets a flag and a count. There is then another for loop the iterates through the players to count how many players have a ball. Asser statements are then used to make sure that the throwingPlayer has the ball, that someone has a ball, and that the number of players with a ball is only one.

}

testBounds(){

This test is to make sure that the game recognizes when the ball/players are out of bounds. It does this by creating a field of type rectangle with a known width and height. It also creates a generic array to called vertices to hold the corner vertices. The test then uses several assert

statements to insure that the field contains the origin and all of the corners as well as checking each of the quadrants and areas that are out of bounds to make sure they are not included in the field.

}

testLoadField(){

This test is for the loading of the game field. Because all we do not yet have the source images for the field or the dimensions the only thing this test does is check that the width and height are correct.

}

testGetInFormation(){

Tests that teams can make formations of players on the field. It does this by calling the getInFormation method with the formation enumerated type (for the type of formation to be formed). It then uses assertTrue statements to ensure that the desired formation has been formed. The test does this for several different teams and formation types as well as two assertFalse statement to test that only the most current call to getInFormation sets the type.

}

testLocationFinder(){

Tests to see if players will know in what part of the field they will be in. The test does this by initializing a rectangular field and then iterating through all of the player in the game and using assertTrue statements that the field contains the players position. The test also initializing several counts to ensure that players will return the correct enumerated type if they are in a special location on the field. Unfortunately we do not have the field built yet so we have been unable to connect field position to special areas so the test has not yet been written.

}

testGoal(){

This test is designed to see if a goal is make that the game knows that a goal has been scored. This is done by having different players form different teams score goals. The test then uses assertEquals to ensure that the correct number of goals has been scored.

}