Team 6 Testing Write Up

Milestone 2 (ConfigTest.java)

For this milestone, we wrote unit tests using TestFX in order to verify the functionality of our program. Since at this milestone there was not a whole lot to test, our main verifications were whether or not the user inputs were valid, if our labels displayed the correct information, and if our exit buttons were accurate. These components were selected since they are the areas of the program that we believed had the most risk in terms of having edge cases that we didn't think about. Below is a brief summary of how each test verifies the functions of our game:

testPlay: This tests that clicking the begin game button starts the game and moves to the configuration screen

testBackButton1: This checks to see that the back button on the configuration screen takes you back to the welcome screen by testing if you can click on the play button again.

testEmptyInput: This tests an edge case for the name text field where the test verifies that the name being entered isn't an empty string by checking if the input is null

testWhitespaceInput: This tests an edge case for the name text field in which the input is a bunch of spaces. The test checks and sees if the input field is null.

testNoGold: This checks that the program will not advance if the user does not click on an initial money value. It verifies that the label asking the user to select their difficulty is still there, signifying that the program has not advanced.

testNoPlayers: This test checks to see if that the program will not advance if the user does not choose a number of players. It checks to see if the text field is still on screen to signify that the program has not advanced.

testQuitButton1: This test tests the quit button in our first screen. This test verifies that the window closes when the button is pressed.

testLabels: This tests that the information that the user's input in the configuration screen is accurately translated to the initial room, where the user's name, weapon, and gold is displayed.

testQuitButton2: This tests that the quit button in the newly added toolbar correctly sends the player back to the initial welcome screen.

testBackButton2: This tests that the back button on the player configuration screens redirect the players back to the initial configuration screen

testWhitespacePlayer: This tests checks that a name consisting entirely of whitespaces is not allowed for players to choose and will not advance to the main game

testEmptyPlayer: This checks that the program will not advance if the user does not provide a name for the player (not an empty string), by checking if the player name is null

testNoCharacter: This checks that the program will not advance if the user does not provide a character for the player, by checking if the player character is null

testMainGame: This tests that the Main Game screen is displayed after player config. It verifies if certain nodes are contained in the Main Game screen.

testNumPlayers: This tests if the correct number of players are added. It verifies this by the numPlayers variable in the controller class.

Milestone 3 (BoardTest.java)

For this milestone, we wrote unit tests using TestFX in order to verify the functionality of our program. Since at this milestone there was not a whole lot to test, our main verifications were player movement via dice-rolling and turn-taking, assuring our labels displayed the correct information, and the monetary effects of tiles. These components were selected since they are the areas of the program that we believed had the most risk in terms of having edge cases that we didn't think about. Below is a brief summary of how each test verifies the functions of our game:

testMovement: This tests that players can choose to move when it is their turn and, as a result, their token is also shown to move.

testTurns: This tests that the game correctly takes turns between players so that after one moves or passes, it is then the next player's turn.

testGetMoney: This tests that green squares provide the player with additional money when landed on.

testLoseMoney: This tests that red squares subtract money from the player when landed on.

testSprites: This tests that the sprites shown on the toolbar matches with the current roll's player.

testToolbar: This tests that the data shown on the toolbar (Player name, gold and turn) is correct in accordance with which player is currently rolling.

testLabelsUpdate: This test verifies that the label on the toolbar that displays the information of the other players (not including the current player) is correctly changed each turn. This is

checked by taking the label text before and after a turn and asserting that they are not the same.

testDiceRolling: This test verifies that the dice rolling animation correctly correlates to the actual number rolled. This is checked by obtaining the integer value of the roll, and based on its value we check that the corresponding dice image is shown in the toolbar.