testBuildable:

- This test is designed to see if the method testBuildable returns the correct boolean, depending on what tile is selected. The test checks three different types of tiles, an empty tile, path tile, and buildable tile and asserts that false, false, and true are returned respectively. This is important to the M3 implementation as towers are only allowed to be placed in designated locations and not on the path; this method checks that tiles are assigned correctly.

testExistingTower:

- This test is designed to check if a tower can be built on a spot occupied by another tower. The test first places a tower in a specific location, then attempts to build another tower in that specific location. It asserts that this action cannot be done by returning false. This relates to the M3 implementation as our game should not allow towers to occupy the same position, and this test checks if that situation is possible.

testOccupySpot:

- This test is designed to check if placing a tower occupies a spot on the grid. This is done by building a tower in a specific buildable location, then asserting that the specific location on the grid is no longer buildable. This relates to the M3 implementation as towers need to occupy a grid space so they can exist in the game world and block any other objects from existing in the same space. This test checks if this is the case.

testDifferentMoney

- This test is designed to check if selecting different difficulties changes the starting money given to the player. Three players are given, each having selected different difficulties. Then the test checks if each player has different amounts of money compared to each other. This test relates to the M2 implementation, as selecting different difficulties is supposed to change the starting money a player has. This method confirms this occurs.

testTowerCosts

- This test is designed to check if selecting different difficulties changes the cost of a tower. This is done by creating three towers, each of which is done under a different difficulty. Then their tower costs are compared to each other to confirm that each tower has a different cost. This is relevant to the M3 implementation as towers are supposed to cost different amounts based on the difficulty chosen. This test cases confirms if this occurs.

testInsufficientFunds

- This test is designed to check that a player cannot buy a tower if they lack the money to purchase it. A player is first created, then has the money set to 0. After this, the player attempts to purchase a tower. This should result in false, as the tower costs more than 0 currency. This is important to the M3 implementation, as buying towers is supposed to cost money, and a player is not allowed to purchase a tower if they don't have the money for it. This test checks if this is the case.

testSpentMoney

- This test is designed to check if buying a tower changes the money a player holds. A player is first created, and then buys several towers. Between each purchase, a comparison is made between the current player money and the expected player money to ensure the player's money is reflective of the purchases made. This test relates to the M3 implementation as successfully buying towers is supposed to cost money and change the player's money to reflect a purchase. This test confirms this is occurring.

testName

- This test is designed to check if a player inputted a valid name. A valid name is one that is not empty or only white space. The test first checks if an empty string representing a name returns false, and then if an only white space name returns false. Finally the test checks a normal name, including whitespace, to see if it returns true. This test is relevant to the M2 implementation as names cannot be empty or contain only white space. This test confirms that names cannot be whitespace / empty.

testDifficulty

- This test is designed to check if a player has selected a difficulty. The test first checks if an empty difficulty string returns false, then tests that difficulties "Easy", "Medium", and "Hard" return true. These three difficulties are the options a player is able to select. This test is important to the M2 implementation as the player is required to select a difficulty in order to proceed with the game. This test confirms that selecting a difficulty is required.

testHealth

- This test is designed to check if the monument's health is different based on the difficulty selected. This test first creates 3 players, each having different difficulties selected. Then the health of all player's monuments is compared to the other healths and assert that they all equal different amounts. This test is relevant to the M2 implementation as selecting different difficulties is meant to change the starting health the player's monument has. This test confirms if this is true.