Test List

AllTest.java

AllTest.java is the test suite used to perform all the tests. Included within AllTest.java are test classes PlayerTest.java, RoomTest.java and DoorTest.java.

PlayerTest.java

testGetLocation()

This tests the getLocation method in the Player class of the AdventureGame program. To determine correctness while not using any other methods, the initial state of the Player location was checked. When a Player is created and a new location has not been set, player location should be null. The assertNull JUnit4 method was used to test that getLocation returned a null value. getLocation method was tested because it is used in the required test methods.

testSetLocation()

This method tests the setLocation method of the Player class. The initial state was tested using the assertNull Junit4 method. setLocation(r1) was then used to set the player location to a new room and correctness was tested using the assertEquals JUnit4 method with r1 and p.getLoc() as arguments. setLocation was tested because it is used in the required test methods.

testGo()

This method tests the go method of the Player class. First movement of the player from one room to the other was tested by setting the player location to Room r1 with r1 having a new Room r2 set as it's side. the go method was then called on the side that r2 was located on and the assertEquals JUnit4 method was used to check that r2 was the new location of the player. The go method was then called in the direction of a wall with assertEquals being used to check that the player location had not changed and was still r2. The test is implemented to make sure that a player is successfully moving from one room to another when they are supposed to.

testShowMyThings()

This method tests the showMyThings method of the Player class. For testing, the method was called to check that the content items were being displayed correctly. The assertEquals JUnit4 method was used with "" and the result of showMyThings as arguments. The test was carried out because the method was used in the required testing.

testPickup()

This method tests the pickup method of Player class. To make sure that the item contents were empty, assertEquals JUnit4 method was used to check that the showMyThings return was equal to "", or nothing. assertEquals was used again after calling the pickUp(item) method to check that the return from showMyThings was equal to "1: key ". pickUp and assertEquals were called two more times to check that a second item would be added but not a third. testPickup() is used to make sure that the player can add up to two items to the carried contents but no more.

testDrop()

This method tests the drop method of Player class. The initial state was tested using assertEquals with "" and showMyThings as arguments to establish that nothing was in the player carried contents. One item was then added and assertEquals was used to check that it had been added. drop(2) was then called on the second item and assertEquals was used to check that nothing had happened. drop(1) was then called and assertEquals was used to make sure that the item had been removed from the carried contents. Two items were then added using pickup and drop(2) was again checked using assertEquals to check that the items were removed from showMyThings.

RoomTest.java

testSetSide()

This method tests the setSide method of Room class. Initially, all room sides are set to a Wall value. setSide(0, r2) and setSide(1, r3) were then called to set initial side values to those of newly created rooms. The player was then moved in the direction that the room had been set to and assertEquals was used to check that the player location was equal to the r2 and r3 values. setSide is used to set adjacent rooms in the AdventureGame program.

testGetRoomContents()

This method tests the getRoomContents method of Room class. Only the initial content state was tested in this method with an array being set to equal the return of the getContents method. assertEquals was then used to check that the length of the array was equal to zero, as expected. getRoomContents returns the contents of a given room and the test of this method is used to make sure that items are being successfully added and removed.

testAddItem()

This method tests the addItem method of Room class. The initial state of the room was tested using assertEquals and getRoomContents to make sure that the room contents were equal to 0. items were then added one at a time using the addItem method and then checking that they were in fact added by using assertEquals on the item added and the first and second position of the contents array. This test method is used to make sure that items are being added to rooms when they are supposed to be.

testRemoveItem()

This method tests the removeItem method of Room class. To test the initial state, two items were added to a room and then their presence checked using the assertEquals method to see if the length of the content array was equal to 2. removeItem was used to remove an item, one at a time and check that the length of the content array had been reduced using assertEquals on the length of the content array and the predicted values. This method was used to make sure that items would be removed from the room after being picked up by a player.

testEnter()

This method tests the enter method of Room class. To test the initial state, player location was set to null and then checked using assertEquals to make sure that it was in fact null. The enter method was then used to move a player into a room and checked using assertEquals to make sure that the player location was equal to the room that was entered.

This test was used to make sure that a player could successfully move from one room to another and that the new location would be equal to the desired room.

testExit()

This method tests the exit method of Room class. The initial state was tested by setting player location equal to the first room and checking that location had been set using the assertEquals method on room, and player location. A new room, r2 was created and set as one of the first rooms sides. The exit method was then called on that direction and checked using assertEquals to make sure that r2 was equal to the player location. This test was used to verify that a player could successfully exit a room and that player location would be set to the new room.

DoorTest.java

testEnter()

This method tests the enter method of Door class. The initial state was set up by adding a key to a room and setting player location to that room. a new door was then created between room r1 and room r2 requiring key k1. The enter method from r1 to r2 was first called without picking up the key and assertEquals was used to make sure that player location was equal to the initial room. The key was picked up and enter was again called from r1 to r2 and assertEquals was used to make sure that player location was equal to r2. This same process was then repeated but in the opposite direction, from r2 to r1, again using assertEquals to check for room location correctness. This test was carried out to determine the correct function of the doors between rooms as well as the key requirement.