1.) TestEnter, uses instances of Player, Room, and Key class to test method in Door class.

2.) Tests the enter() method. The initial state creates a door with an outside and inside room on either side of the door, and sets the key that opens the door. The input data is a Player who holds a Key.

3.) The output should be a message of whether or not the Key worked on the Door. This is checked with an assertTrue check that makes sure the output message is equal to what it should be.

4.) This test covers the entering a Room through a Door that has a specific Key functionality.

5.) JUnit 4 class: DoorTest, method: testEnter()

1.) TestGo, uses instances of Player and Room to test moving between rooms.

2.) Tests the go() method. The initial state creates a Player and four Rooms and sets the Player’s location to one of the rooms. The input data for the test is a Player in a Room and the direction the player would like to move.

3.) The output should be the entry message for the Room just entered. This is checked with an assertTrue check that makes sure the output message is equal to what it should be.

4.) This test covers the traversing through different Rooms in the adventure functionality.

5.) JUnit 4 class: PlayerTest, method: testGo()

1.) TestPickUp and TestDrop use instances of Player, Room, and Item class to test methods in Player class.

2.) Tests the pickUp() and drop() methods. The initial state creates a Player a Room and some Items, then sets the Player’s location to the room. The input data for the tests is a Player in a Room with some Items in it.

3.) The output should be an array of Items carried by the Player after the pickUp or drop. This is checked with an assertArrayEquals check that makes sure the Player’s Item array after the pickUp or drop is equal to the expected array.

4.) This test covers the ability of a Player to pick up or an Item from a Room.

5.) JUnit 4 class: PlayerTest, methods: testPickUp(), testDrop()

1.) TestAddItem and TestRemoveItem use instances of Room and Item to test methods in the Room class

2.) Tests the addItem() and removeItem() methods. The initial state creates a Room and some Items. The input data for the tests is a Room with some Items in it.

3.) The output should be an array of Items in the room after the add or remove. This is checked with an assertArrayEquals check that makes sure the Room's Item array after the add or remove is equal to the expected array.

4.) This test covers the ability of a Room to have an Item added to it or removed from it.

5.) JUnit 4 class: RoomTest, methods: testAddItem(), testRemoveItem()

1.) TestEnter and TestExit use instances of Player and Room to test methods in the Room class

2.) Tests the enter() and exit() methods. The initial state creates a Player and two Rooms and sets the Player's location to one of the rooms. The input data for the test is a Player in a Room and the direction the player would like to move

3.) The output should be the entry message for the Room just entered. This is checked with an assertTrue check that makes sure the output message is equal to what it should be.

4.) These tests cover the ability of a Player to enter or exit different Rooms.

5.) JUnit 4 class: RoomTest, methods: testEnter(), testExit()