Ashley Bertrand
Testing – A3

---

testEnterDoor()
Class: AllTest

Description: This test is checking for correct functionality with regards to moving through a door. The biggest caveat to consider is that the player must be carrying the key in order to enter. The door constructor also takes two CaveSite parameters and each of the following cases must be checked:

1. Player moving from first CaveSite parameter to second without the key

   Output: player's location is the first parameter because player isn't carrying the key to get through the door

2. Player moving from first CaveSite parameter to second with the key

   Output: player's location is the second parameter because player has the key to get through the door

3. Player moving from second CaveSite parameter to first without the key

   Output: player's location is second parameter because player isn't carrying the key to get through the door

4. Player moving from second CaveSite parameter to first with the key

   Output: player's location is first parameter because player has the key to get through the door

Method tested: enter(Player p) in class Door

Functionality: this test is covering functionality for enter() when a player is moving through a door; it also calls setLoc(), getLoc(), addItem(), pickUp(), and drop() to accomplish test coverage

Initial state of door: takes an outside Room object, an inside Room object, and a Key object

Initial state of player: player's location is set to outside

Input data: two Room objects, a Key object, a Door object, and a Player object; enter() takes a Player object

---

testGo()
Class: AllTest

Description: This test is checking for correct functionality with regards to moving between rooms according to an int direction passed as a parameter to go(). A player can move from one room to another through the direction according to which the sides of the rooms are set up. When this is done correctly, the player's new location becomes the adjacent room from the passed direction.

Output: player's location is the parameter passed into the side of the initial room in relation to the direction (int) that gets passed to go()

Method tested: go(int direction) in class Player

Functionality: this test is covering functionality for go() when a player is moving from one room to another; it also calls setSide(), setRoom(), getLoc(), look(), and getDesc() to accomplish test coverage

Initial state of player: player's location is set to room1

Initial state of room1: room1's side at direction 0 is set to room 2

Initial state of room2: a new room object gets instantiated to be passed to setSide()

Input data: a Player object and two Room objects; go() takes an int

---

testPickUp()
Class: AllTest

Description: This test is checking for correct functionality with regards to a player picking up items. Important measures to consider are the item the player picks up must be located in the room the player is currently located in. Also, the player can carry a maximum of 2 items; any additional items that the player gets called to pick up will not be added to the player's myThings array.

Output: player is carrying the item with which it was called to pick up; player will pick up a second item if called to do so; player will not pick up any items after hands are full (carrying 2 items)

Method tested: pickUp(Item i) in class Player

Functionality: this test is covering functionality for pickUp() when a player is going to grab an item; it also calls setDesc(), addItem(), setLoc(), getFirstItem(), getDesc(), handsEmpty(), handsFull(), numItemsCarried(), haveItem(), showMyThings(), and getSecondItem() to accomplish test coverage

Initial state of player: player's location is set to room

Initial state of room: room gets item1, item2, and item3 added to it

Input data: a Player object, three Item objects, and a Room object; pickUp() takes an Item object

---

testDrop()
Class: AllTest

Description: This test is checking for correct functionality with regards to a player dropping items. A player can only drop an item that is currently being carried and cannot drop anything when handsEmpty() is true. Since a player can carry a maximum of 2 items, an int 1 or 2 gets passed to drop(), telling the player which item to drop, the first item or the second item respectively.

Output: player is no longer carrying the item with which it was called to drop; when player is carrying nothing the player's hands will be empty

Method tested: drop(int itemNum) in class Player

Functionality: this test is covering functionality for drop() when a player is going to let go of an item; it also calls setDesc(), setLoc(), pickUp(), numItemsCarried(), haveItem(), handsEmpty(), getFirstItem(), getDesc(), and handsFull() to accomplish test coverage

Initial state of player: player's location is set to room

Input data: a Player object, two Item objects, and a Room object; drop takes an int

---

testAddItem()
Class: AllTest

Description: This test is checking for correct functionality with regards to adding an item to a room. Items that get added are put into an ArrayList, and there is no capacity limit to the number of items that a room can hold. An item gets added to a room either with the initialization of the game or when a player drops an item.

Output: Item i gets placed in a room when addItem(Item i) is called on a Room object; the room will not be empty

Method tested: addItem(Item i) in class Room

Functionality: this test is covering functionality for addItem() when placing or leaving an item in a room; it also calls getRoomContents() and roomEmpty() to accomplish test coverage

Initial state of room: a new room object gets instantiated

Initial state of item: a new item object also gets instantiated so it can be passed to addItem()

Input data: a Room object and an Item object; addItem() takes an Item object

---

testRemoveItem()
Class: AllTest

Description: This test is checking for correct functionality with regards to removing an item from a room. An item gets removed from a room when a player picks up an item. Items that do not exist in a room cannot be removed from a room.

Output: Item i gets removed from a room when removeItem(Item i) is called on a Room object; the room will be empty if there are no more items left to be 'removed'

Method tested: removeItem(Item i) in class Room

Functionality: this test is covering functionality for removeItem() when an item gets picked up from a room; it also calls addItem() and roomEmpty() to accomplish test coverage

Initial state of room: an item is added to room

Input data: a Room object and an Item object; removeItem() takes an Item object

---

testEnterRoom()
Class: AllTest

Description: This test is checking for correct functionality with regards to a player entering a room. When enter() is called on an instance of a Room, the player that is moving locations is passed as a parameter. The player's location becomes this new room.

Output: player's location becomes the room with which enter() was called on

Method tested: enter(Player p) in class Room

Functionality: this test is covering functionality for enter() when a player moves to another room; it also calls getLoc(), look(), and getDesc() to accomplish test coverage

Initial state of room: a new room object gets instantiated

Initial state of player: a new player object also gets instantiated so that it can be passed to enter()

Input data: a Room object and a Player object; enter() takes a Player object

---

testExitRoom()
Class: AllTest

Description: This test is checking for correct functionality with regards to a player exiting a room. When exit() is called on an instance of a Room, the player that is moving locations is passed as a parameter. In addition, a directional int is also passed, and the room at the adjacency of the direction becomes the player's new location.

Output: player's location becomes the room on the adjacent side of the room which exit() was called on in the direction of the int passed to exit()

Method tested: exit(int direction, Player p) in class Room

Functionality: this test is covering functionality for exit() when a player leaves a room; it also calls setSide(), getLoc(), look(), and getDesc() to accomplish test coverage

Initial state of room1: its side at direction 0 is set to room2

Initial state of room2: a new room object gets instantiated to be passed to setSide()

Input data: two Room objects and a player object; exit() takes an int and a Player object