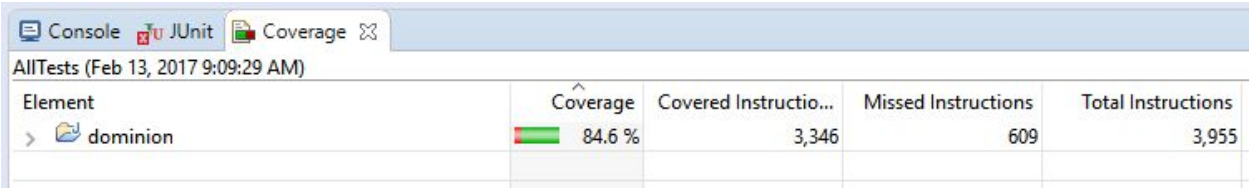
Test Cases:

DO	Expected Results	Actual results
Smithy: 1. Check size of deck 2. Run function	Check size of deck is 3 more	deck is 3 more
Laboratory 1. Check deck size 2. Run function	Check deck size is 2 higher Check player action amount is still 1	deck size is 2 higher player action amount is still 1
Library 1. Run function	Check size of hand is greater than 6	size of hand is 7
Festival 1. Check amount of coins 2. Run function	Player should now have 2 actions Player should now have 2 buys Player should now have 2 more coins	Player has 2 actions Player has 2 buys Player has 2 more coins
Village 1. Check size of deck 2. Run function	Check size of deck is 1 more	Deck size is one higher
Bureaucrat 1. Count number of coins 2. Run function	Make sure number of coins is 2 higher	Player has 2 more coins
Chapel 1. Check size of hand 2. Run function	Check size of hand did not change	Size of hand did not change
Cellar 1. Check size of deck 2. Run function once	Check deck is 1 higher check still has 1 action	Size of deck is 1 higher Player gains an action
Chancellor 1. Check amount of coins	Check amount of coins is raised by 2	Player has 2 more coins

2. Run function		
Council room 1. Check size of deck 2. Run function	Check size of deck is 3 higher	Deck has 3 more cards
Feast 1. Check size of deck 2. Check total value of deck 3. Run function	Deck is same size Deck value has not raised by more than 5	Nothing happened
Turn Test 1. Set amount of turns	Player's score is the same as the winners score	pass

I was getting a lot code breaking bugs with the Adventurer, feast, and garden cards, so they are not yet in release. I believe I was going outside the hashmap when calling something with them as I was getting a null draw.

Because I was not able to get any outside coverage tool to work with my code I used the native eclipse coverage tool which stated my tests covered between 78% and 86% covered.



The main difference I would do to run these tests automated would be to change the print functions to assert methods. On each test I would use:

assertEquals("OneTwo" , one+two);

to show whether they pass or fail. For different numbers, I would just change the amount of players and other changeable variables by assigning them a random variable in their respective ranges using random and modulo. This would allow me to check and change all tests to randomly fit the "moveable parts".