Benjamin Tate
10/22/2017
CS 362 -- Section 400
Assignment 3

Bugs:

My unit tests of dominion functions (not cards) found no bugs. For my card tests, I chose to make tests for Smithy, Adventurer, Village, and Great Hall cards. Smithy and Adventurer were two of the cards that I introduced bugs in for assignment 2, so naturally my tests found bugs in them. For Smithy, I gave it a bug that makes it draw 2 cards instead of 3, so all tests failed, since it wasn't behaving at all how it was meant to. In the Adventurer function, I made it look for 3 treasure instead of 2, which showed up in my unit tests that had to do with how much treasure was gained, as well as how many cards were drawn, discarded, etc., since it altered how the function searched through the deck. The Village and Great Hall cards were not ones that I had introduced bugs to, so they passed all of the tests that I made for them.

Unit Testing:

Analyzing coverage with gcov yielded relatively low coverage percentages in general, but they seem much more reasonable when you take into account the fact that each call of gcov is only looking at tests for one function. When running the tests individually, we can manually look at how many times each line of each function was run during testing.

- updateCoins():
    - updateCoins() is a relatively simple function, and my tests were able to achieve complete statement and branch coverage for it.
- drawCard():
    - My tests were able to get 100% statement and branch coverage for drawCard(). However, with branch frequency splits at or around 93%/7% for all of its branches, it's safe to assume that its path coverage was relatively low.
- isGameOver():
    - Full statement and branch coverages were also achieved for isGameOver(), however, like with drawCard(), some of the branches had very uneven frequencies, so path coverage was not very good.
- fullDeckCount():
    - I got full statement coverage for fullDeckCount(), but there was one branch that was never taken by my tests. This is because I forgot to add card(s) including at least one Smithy to the discard before testing, so this would be a pretty easy fix.
- Smithy:
    - The smithyCase() function had very good coverage, with every statement being covered at least 55 times, and with the both sides of its one branch being taken (with a 67/33 split on frequency).
- Adventurer:

    - The adventurerCase() function had complete statement and branch coverage, but with all of its loops and branches, more complex types of coverage such as path were not as complete.
- Village:
    - Each line of the village case in the cardEffect() function was run exactly once. The case only has 3 simple statements, so that is full coverage all around.
- Great Hall:
    - Each line of the great_hall case in the cardEffect() function was run exactly once. The case only has 3 simple statements, so that is full coverage all around.

In the extremely straightforward units, such as the Village and Great Hall cards, all types of coverage were complete, but since the tests weren't well-designed for getting things like path and data-flow coverage, those got lower and lower the more branches and loops the unit had.

Unit Testing Efforts:

- updateCoins():
    - The suite for this unit has 3 tests, which check whether updateCoins works with bonuses from 0-4 and copper, silver, and gold coins, respectively.
- drawCard():
    - I used 4 tests for this function. The first is to draw all cards from the deck. The second is to attempt to draw when both the deck and discard are empty, checking that it is unsuccessful and handCount doesn't change. The third is to draw from the discard pile when the deck is empty. The fourth is to once again draw all cards, as at this point there should be no change.
- isGameOver():
    - A game ends when either the province supply pile is empty or there are 3 empty supply piles of other types, so I simply tested each of these cases and made sure that the game ended when appropriate and not when it shouldn't.
- fullDeckCount():
    - First, I attempted to count smithy cards when I knew there were none in the deck, discard, or hand. Next, I used gainCard() to add one smithy to the discard, deck, and hand, checking between each to make sure that the number of smithys went up, as well as the appropriate pile.
- Smithy:
    - For the Smithy card, I just tested to see if the hand gained 2 cards (+3, -1) and the discard lost 1.
- Adventurer:
    - For Adventurer, I used a loop to test with 0-4 gold in the deck, and on each loop, checked how many treasure were gained and how many cards were discarded, removed from the deck, and added to the hand, compared to the appropriate numbers, given the known quantity and placement of the gold cards.
- Village:

    - Since the Village is usch a simple card, all that I had to do was test the change in the number of cards in the hand, and in numActions.
- Great Hall:
    - Great Hall behaves very similarly to Village, so the only difference in my test was how many gained actions I was checking for.