Dominic Phan
Assignment 3
CS 362

Bugs

Out of the eight unit tests I made, I discovered three errors, each belonging to cards. The unit tests I did for various functions of dominion were error free. The various functions were error free to an extent. I had to make an assumption to discardCard() due to the bad documentation it had. It wasn't clear whether or not the card was supposed to be put in the discard pile, as it was only moved to the play pile. I assumed that a separate function would take the card out of the play pile and put it in the player's discard pile. This is based on the instructor's response on Piazza:

> "As I can see from the comments of the discardCard in the dominion.c, the discardCard does two things based on the int trashFlag.
> 1- if the trashFlag=0, discardCard adds the played card to the palyer pile, and then it removes the card from the player hand.
> 2- if the trashFlag=1, discardCard removes the card from the player hand.
>
> I should generate my unit tests based on those scenarios. I mean if I want to test discardCard function, I'll, for example, generate a player with three cards and then call the discardCard function, finally assert if the the card is removed from the player hand, the handCount is reduced by 1. I might need to generate three (or more) different unittests to cover all the scenarios (i.e., branches) and generate the proper assertions for each unit test." – Ali Aburas

As for the 3 card bugs I discovered, two of them were bugs I introduced into the adventurer and smithy class, and the last was a bug that I did not introduce.

Smithy Bug:
Smithy failed my unit test that determined whether or not three cards were added to the hand. It failed four times (tested once for each player). This makes sense as the bug I introduced was turning the conditional statement in the for loop from a less-than to a less-than-or-equal-to statement, thus producing an extra card.

Adventurer Bug:
Adventurer failed a unit test that checked whether or not two cards were added to the hand. It also failed a unit test that checked whether or not adventurer was added to the play pile. The first bug makes sense, because I introduced that error when changing the conditional statement of the for loop, similar to Smithy. The second bug, however, is due to the fact that the function did not include a discardCard function call to remove the card from the hand.

Mine Bug:
The Mine card actually had many bugs. When trying to use mine to convert a treasure to the same treasure from supply or a treasure to a lower treasure from supply, my unit tests failed. This is a failure in the implementation of the card because according to the Dominion wiki, players are able to sacrifice treasures for the same one, or one with lower value. The

implementation passed the test when converting the treasure to one that was +3 in value though.

Unit Testing

This is a table of my code coverage:

	Unit1	Unit2	Unit3	Unit4	Card1	Card2	Card3	Card4
Testcov	84.85%	89.29%	95.16%	94.12%	92.42%	96.27%	89.41%	90.00%
Domcov	16.33%	18.03%	20.34%	22.34%	30.35%	33.59%	34.21%	36.98%
branchcov	16.31%	16.31%	17.75%	19.18%	20.14%	30.46%	30.46%	40.53%

From this data, I believe my unit tests were decently written, but much improvement could be made. The statement coverage for the individual tests had a high percentage, so my unit tests mostly ran. The statement coverage for the dominion file is moderately covered. It seems that depending on what card/function, I test, statement coverage for the file goes up 2-8%. After inspecting the gcov data file, I can see clearly that most of the lines that were never executed, represented as "#####", belong to cases for cardCost and cases for cardEffect. I believe once unit tests are written for each card, statement coverage should improve greatly.

My branch coverage is actually kind of low. It's strange that there wasn't an increase in branch coverage from unittest1 to unittest2. The handCard() function returns a card at a certain position from the player's hand, so I believe the reason for no change in branch coverage is due to the simplicity of the function. It simply returns a value from the gamestate. There was also quite the jump in branch coverage when testing the "Mine" card. I believe it's due to the implementation of mine… There are many if-else statements used and functions called that have if-else statements.

My boundary coverage was simplistic, but I believe it worked in most cases. When testing the cards, I made sure to consider boundary cases such as what happens if there are no cards left in the deck when smithy is called, or what happens when there's one treasure in the deck and multiple treasures in the discard pile when adventurer is called.

Unit Testing Efforts

supplyCount()

The supplyCount() function must return he correct amount of supply cards available. This is important because once supply for three cards = 0, the game end. I developed tests by creating supply piles consisting of victory cards, curses, resource cards, and nonvictory kingdom cards because these card's values differ based on the number of players. I made sure everything functioned correctly and that all values returned were value for the differing number of players.

handCard()

This was a very simple card to test because all it does it return the card as a certain position from the current player's hand. To test thing, I simply added 5 unique cards to the hand and verified that the cards were returned in the right position when calling handCard(). I did not need to test for invalid inputs (out of range) because it's only used to discard or get the card that will be played, so the parameter handPos will always be valid.

gainCard()

gainCard has three different options to where the card is going to. I created a test for each destination (hand, deck, discard) and I also created a test that checks whether the supplyCount has decreased after using the card. I also made sure to use a test to check what happens during invalid values (supply out or name is incorrect).

discardCard()

This function takes a card from the player's hand and either trashes it or moves it to the played pile. To test this, I created tests for both functionalities and ensured that it worked for all four players and all hand positions. I had to create an array consisting of all coppers and check that each copper was sent to the discard pile. For the cards that were trashed, I could not check where they went, but I could check that they were not in the played pile and that they were not still in the hand.

Smithy

To check smithy, I created tests that tracked whether or not other player's gamestates were changed by running this card effect. I also created a test that checked that three cards were added to the hand and that smithy was removed from the hand. I also checked that smithy was added to the played pile.

Village

Village allows the player to draw one card and gain two actions. To test this card, I kept track of player's gamestates and made sure the card did not mutate other gamestates because it only affected the current player. I had to create a test htat checked a card was drawn and added to the hand. I also checked that two actions were added to the gamestate. I also had to ensure that the village card was added to the discard pile.

Adventurer

The adventurer card is supposed to search the deck until two treasures are found. The rest of the cards area discarded. I had to test a few situations: two top cards of the deck are treasures, treasures somewhere in the middle of the deck, an empty deck with treasures in the discard pile and a deck with one treasure in the deck and one treasure in the discard pile. I also created a test to check that other player's gamestates were not mutated. I also checked that the adventurer was put into the played pile after being used.

Mine

The mine card allows the player to trash a treasure from their hand to gain a treasure that is up to 3 cost of the trashed treasure. To test this, I had to check that the card was removed from play, so I had to check the hand, deck, and discard pile to make sure it did not remain. I also had to check that the new treasure was added to the hand. I also checked that other player’s gamestates were not mutated. I ran all these tests for these scenarios: trashing copper for silver or copper, trashing silver for copper, silver, or gold, trashing gold for copper, silver or gold.