PLEASE NOTE

While working on this assignment, I had issues getting Junit to work as well as mavn. As a result, I made a separate class called “CardTest.java” which is with my original code and this report covers. My friend helped me move the project to a new IDE however that supports mavn and Junit and this allowed me to give the testing suite a much more comprehensive investigation into my code. So I ended up with a problem: I have extensively worked with my original code but it has little coverage, and the version with really good test coverage is barely tested on the OSU servers and has a makefile I made only 30 minutes before the due date! To resolve this issue, there are two projects that are essentially the same: One is located in /bin and this is the one I made in Eclipse. This version works for sure but doesn’t even have an option to use it’s test class while compiling. In /testing however, is a version I just made using NetBeans, and this one has a much better test suite I recently made. The makefile is new but barely tested by myself, but at least works enough to compile and allow testing. If you read the make file included with it it will tell you the commands you can use, the best one is “make coverage” which does all the test coverage stuff and posts it somewhere on the internet. I’m sure you know how to access it but I’m still trying to learn this portion as much as possible. Anyways, go ahead and run that and you should be able to access all the test statistics you need, which unfortunately I can’t include in this document since I have no idea how to access an html page through terminal (is that even possible???)

Anyways, sorry to turn in such a mess to ya’ll but good luck and thanks!

A: Unit tests:

Tests cover the valid creation of a new card, card assignment, and holding information within that card relevant to the program. Major functions are asserted to be correctly functioning. Assigncard assigns a card values based on the number passed in, this is asserted to correctly assign values. StartRound and findDeckSizes are tested together; StartRound assigns cards to all the piles and populates them with the correct card type, and findDeckSizes iterates through the piles to find how many assigned cards are within the piles. Each pile used throughout the entire game is asserted to have the correct amount of cards fitting for the start of the game, relative to the official Dominion rule set. Input validation is asserted to correctly function. MoveCard moves a card from one pile to the top of the other, and is asserted to correctly move a card from the hand to the discard pile. actionCard activates a card’s action and is asserted to correctly activate coin adding from a generated card. buyCard buys a card from a pile and adds it to the discard pile, and is asserted to correctly place a colony card in the discard pile.

B Actual outcomes of executed tests

Card objects confirmed valid and accessible through array, applicable to hand, deck, discard pile, and card piles. All functions work as intended (as far as I can tell).

C Choice of input

Input for program was chosen through number commands 1-n. Validity checks were utilized throughout. Input for test suite were functions assertTrue() returning boolean. Input returns error upon entering string.

D bugs

Since the instructions required bugs to be present, 5 bugs within the program are as follows:

-Special actions for certain cards do not activate

-platinum cards if in great number do not correctly add up in display at times, hasn't been recreated

-computer cheats to get extra Victory points

-choosing card within 100 but beyond hand size accesses an empty card of no value

-Error upon string input instead of number commands

E commentary

Dynamic testing covered the bulk of the project until completion. Dynamic testing consisted of variable watching, sysouts throughout the functions monitoring progress, and manual breakpoint and step ins. Upon completion CardTest.java was developed to cover the major functions of the project within a testing environment. A more indepth testing suite is being developed within Mavn to be run for easier access on NetBeans but due to unfamiliarity with the environment may not be included. Overall the program functions well as long as user input remains valid, and runs from start to finish according to official Dominion rulesets. Cards chosen to be used within the game are the basic cards recommended by the official manual for a first game to be played with and have values corresponding to their physical counterparts.

F auto test system

```
Method(parameters){
Switch param{
        Case1: assertTrue(method works as desired); break;
        Case2: assertTrue(method works as desired); break;
        Case3: assertTrue(method works as desired); break;
```

```
        Case4: assertTrue(method works as desired); break;


        }
}
```

This method takes in parameters that would include flags for what method should be tested, as well as a flag to determine if the method could be tested in a more specific way (thus the different cases).