**Final Report**

**State of Features**

* Our code supports a game with 2-4 players.
* Each game will be played with 10 randomly selected kingdom cards.
* During each turn, the player will be allowed to see his hand of cards.
* He will then be allowed to select which card he wants to play. If he does not have any action cards, the game will immediately proceed to his buy phase.
* The number of actions, buys and money he has will be modified based on which action card he played.
* He will be allowed to select which card(s) he wants to purchase.
* The card(s) that he purchases are added to his discard pile.
* The game will automatically transfer to the next player with a message box stating which player’s turn it is.
* When a witch is played, the game will be able to tell which players have moats in their hands and which players must obtain a curse. **A message will be displayed by the players who got curses.**
* When a militia is played, the game will be able to tell which players have moats in their hands and which players must discard down to 3 cards. **If a player must discard, the game will wait until his turn and then prompt him to do so.**
* The game will display a count of how many of each type of card is available so that players can keep track of how soon the game will be over.
* At the end of the game, the game will tally points and declare a winner or state which players tied.

**Testing Strategies Employed**

All of the tests for this project were created in a white box manner, because the code was visible as the tests were developed.

The project was run primarily on test driven development. Most of the tests throughout the project were unit tests. The Visual Studio built-in tests were used to run the unit tests. The unit tests were also used as regression tests, because if a previously passing unit test suddenly failed, it was clear that the newly written code was not functioning properly.

The cards that required the use of checkbox messages to appear were difficult to unit test. Tests for these cards were written as scripted tests where the tester was given a strict set of instructions to follow to execute each test (ie…check 1 box, check 2 boxes, check the gold card).

As the project progressed, system tests became extremely important to test that the whole system was able to work cohesively together. Finally, once the project was ending, exploratory tests (in the form of repeatedly playing the game) were used to find minor bugs within the entire project.

**Testing Strategies Not Employed**

Black box testing was never used during this project because all tests were written by people with access to the code.

**Testing Examples**

***Unit Testing:*** This unit test tests that treasure cards return no victory points.

[TestMethod]

public void testTreasureReturnsNoVP()

{

Card c = new Silver();

Assert.AreEqual(0, c.getVictoryPoints());

}

***Scripted Testing:*** This scripted test tests that when a mine card is played and the player chooses to give up a silver, that player then receives a gold in his hand. The tester is instructed to select the silver card from the checkbox menu that appears when the test is run.

[TestMethod]

public void TestMineGiveUpSilverForGold()

{

Dictionary<Card, int> cards = new Dictionary<Card, int>();

cards.Add(new Gold(), 1);

GameBoard board = new GameBoard(cards);

Card c = new Mine();

Player p1 = new HumanPlayer(1);

board.AddPlayer(p1);

ArrayList newHand = new ArrayList();

newHand.Add(new Silver());

newHand.Add(new Silver());

newHand.Add(new Silver());

newHand.Add(c);

p1.setHand(newHand);

int cardsInHand = p1.getHand().Count;

int moneyInHand = p1.getTotalMoney();

p1.playCard(c);

Assert.IsTrue(p1.getHand().Contains(new Gold()));

Assert.AreEqual(moneyInHand + 1, p1.getTotalMoney());

Assert.AreEqual(cardsInHand - 1, p1.getHand().Count);

}

***System Testing:*** System testing was used in the middle of the project to ensure that the cards and graphics were communicating correctly.

Exploratory Testing: Exploratory testing was performed by playing through the game several times in an effort to find all bugs.



**Thoroughness of Testing**

//TODO