Eddie C. Fox

January 26, 2017

CS 362: Software Engineering II

Assignment 2

Documentation Section:

The documentation should contain your understanding of the code of smithy and adventurer card. It should also contain documentation of your understanding of the code of **discardCard**() and **updateCoins**() methods. The documentation section should contain statements to check **preconditions** and **postconditions** of each function. In other words, you need to write what it is required or must be true before the function is called, and what will be true/changed when the function is finished/returned.

Coding of Cards in General:

Before a card is played, there is a function called playCard() that is called that validates if the card being played can be played at the time. It checks various things such as if the player has enough actions, and if it is the right phase. The card variable in the playCard() function is assigned the result of the handCard() function based on the parameter passed to it. The cardEffect() function, after some initial set up, has a bunch of switch functions for each possible card that can be played. Based on the value of the card variable in playCard(), the appropriate switch statement is executed, and then cardEffect() returns. Adventurer and Smithy are just two of the cards that can be found in the switch statements here.

Adventurer Card:

The adventurer card is a 6 coin cost kingdom action card that allows a person to reveal cards from their deck until 2 treasure cards are revealed. After putting those into your hand, the rest are discarded.

The adventurer code has a while loop that occurs as long as the drawntreasure variable is less than 2. First, it will check the number of cards in the players deck. If it is empty, the discard pile is shuffled and added to the deck. Cards are drawn in the while loop and added to the top of the players hand. If it is copper, silver, or gold, the drawntreasure variable is incremented by 1. Otherwise, the card is added to a temporary array so that the game knows what to discard. After finding two treasures is complete, the cards that were added to the temporary array are discarded.

Smithy:

The adventurer card is a 4 coin cost kingdom action card that allows a person to draw 3 extra cards that turn.

The Smithy card uses a for loop that iterates 3 times, calling the drawCard() function for the currentPlayer 3 times, which draws the player 3 cards.

*int discardCard(int handPos, int currentPlayer, struct gameState *state, int trashFlag)*:

This function is used to discard cards. It has a trash flag to distinguish between cards that are trashed due to effects and those are discarded due to being played. The code contains extra conditions for dealing if the last card in the hand is the one being played.

Preconditions:

Must be passed appropriate parameters. The hand position of the card to discard, the current player, the gameState struct, and the trashFlag integer. There must be at least one card in the hand to discard.

Postconditions:

This function discards cards after being passed the appropriate parameters. If the card isn't trashed, it is added to the played pile, and the count of played cards is incremented. The card that has its parameter passed to the function is removed by being set to -1, and the number of the cards in a players hand is decremented by 1.

int updateCoins(int player, struct gameState *state, int bonus):

This function manipulates the coins variable in the gameState. The first it adjusts the number of coins in the hand to 0, and then goes through the players hand and adds to the coin variable, 1 for a bronze card, 2 for a silver card, and 3 for a gold card. Finally, it adds bonus coins from the effect of cards.

Preconditions: Must be passed appropriate parameters. Generally only called when a card is played, by the effect of certain cards, and at the end of a turn.

Postconditions: Coin variable in state accurately reflects the number of coins a player has in their hand.

Refactor Section:

Refactored Adventurer, Smithy, Mine, Great_Hall, and Council_Room.

Changes as such:

Adventurer: Created `int playAdventurer(struct gameState *state, int currentPlayer)` function which takes state and current player as parameter from the cardEffect() function. This

function mainly takes code from the switch statement concerning the adventurer card and puts it into the playAdventurer() function instead. This function is called in the switch statement. I also changed the temphand, drawntreasure, cardDrawn, and z variables from the switch statement into the playAdventurer() function. I deleted the drawntreasure variable from the main cardEffect() function because this was the only card that actually used the variable, so I thought it made more sense to make it local.

Smithy: Created `int playSmithy(struct gameState *state, int currentPlayer, int handPos)` function which takes state, current player, and hand position as parameters from the cardEffect() function. Took the code from the switch statement in cardEffect() for playing the smithy card.

Mine: Created `int playMine(struct gameState *state, int choice1, int choice2, int currentPlayer, int handPos)` function. It takes the state, choice1, choice2, currentPlayer, and handPos parameters. Took code from switch statement in cardEffect() for playing the mine card.

Great_Hall: Created `int playGreat_Hall(struct gameState *state, int currentPlayer, int handPos)` function which takes state, current player, and hand position as parameters from the cardEffect() function. Took the code from switch statement in cardEffect() for playing the Great_Hall card.

Council_Room: Created `int playCouncil_Room(struct gameState *state, int currentPlayer, int handPos)` function which takes state, current player, and hand position as parameters from the cardEffect() function. Took the code from switch statement in cardEffect() for playing the Council_room card.

Bugs Section:

Adventurer:
Changed the if condition from the deck count being < 1 to > 1. This means that after every card draw while searching for 2 treasures, the deck shuffles itself, unless the deck only would have one or fewer cards after drawing.

```
if (state->deckCount[currentPlayer] > 1)

{

//if the deck is empty we need to shuffle discard and add to deck
                    shuffle(currentPlayer, state);
}
```

Smithy:

In the for loop for this function, I changed it from `for (i = 0; i < 3; i++)`

To

```
for (i = 0; i <= 3; ++i)
```

By changing the comparison and converting from postfix to prefix (unsure if this is needed), this should have the effect of only drawing 2 cards.

Mine:

Changed the following section:

```
if (state->hand[currentPlayer][choice1] < copper || state->hand[currentPlayer][choice1] > gold)
	{
		return -1;
	}
```

Changed the if condition to: `if (state->hand[currentPlayer][choice1] < copper && state->hand[currentPlayer][choice1] > gold)`

This means that the function will never check to make sure the name of the treasure is in between copper and gold. Instead, it asks the impossible to return -1, that the name must simultaneously be alphabetically before copper and after gold.

Great_Hall: The code for this function was so straightforward, I couldn't figure out how to bug it. That should be acceptable as only 4 out of 5 cards need to be bugged, and all of my other 4 cards are bugged.

Council_Room:

Changed the section:

```
for (i = 0; i < state->numPlayers; i++)
	{
		if (i != currentPlayer)
		{
			drawCard(i, state);
		}
	}
```

So that the if condition within the for loop, if (I != current player) becomes

```
if (i == currentPlayer)
```

This makes it so that the current instead of the other players drawing cards, the player who plays Council Room will draw even more cards, making it very powerful.