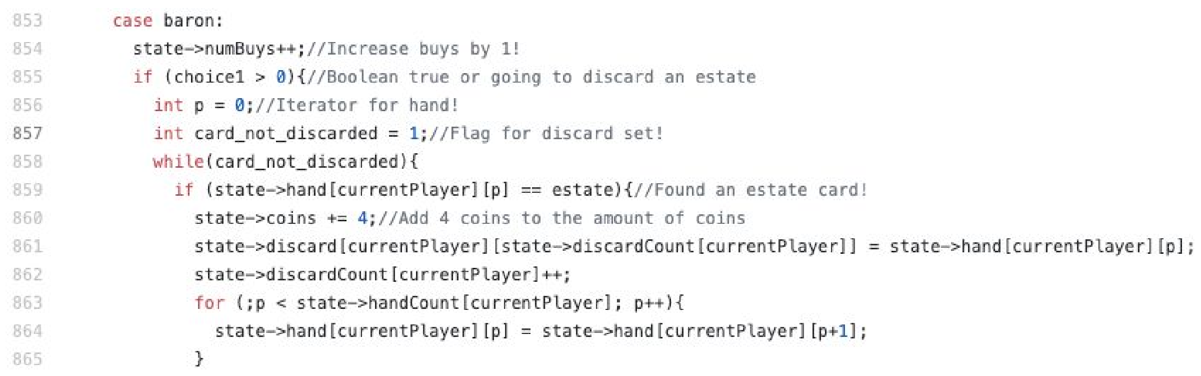
Danny Tran
SOFTWARE ENGINEERING II (CS_362_400_U2019)
Assignment-2.pdf

Refactor

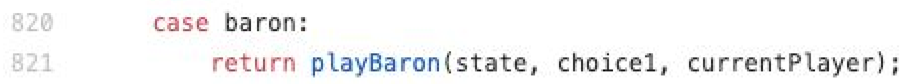
I refactored the following cards: minion, tribute, baron, ambassador and mine. This process involved creating new functions for the cards and function prototypes in the dominion.h file. The new functions also required parameters in order to maintain the card's functionality. All functions return 0 to inform cardEffect that a card from the switch case statement had been played.

Before Refactoring



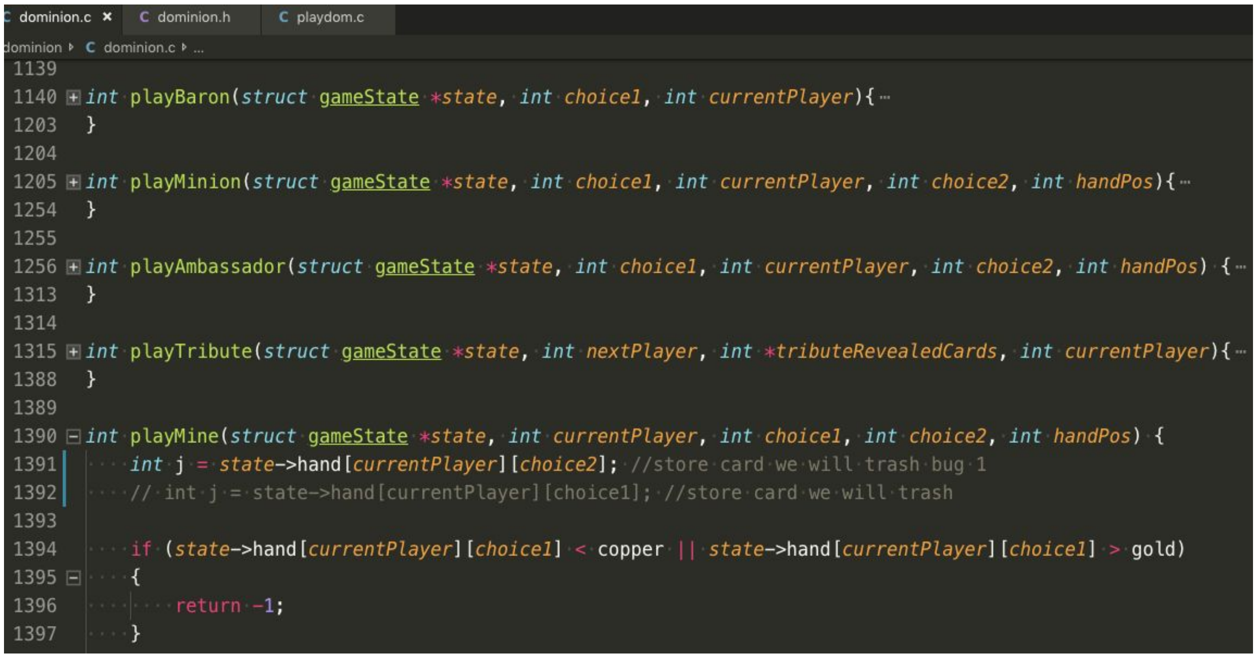
```
    case baron:
      state->numBuys++;//Increase buys by 1!
      if (choice1 > 0){//Boolean true or going to discard an estate
        int p = 0;//Iterator for hand!
        int card_not_discarded = 1;//Flag for discard set!
        while(card_not_discarded){
          if (state->hand[currentPlayer][p] == estate){//Found an estate card!
            state->coins += 4;//Add 4 coins to the amount of coins
            state->discard[currentPlayer][state->discardCount[currentPlayer]] = state->hand[currentPlayer][p];
            state->discardCount[currentPlayer]++;
            for (;p < state->handCount[currentPlayer]; p++){
              state->hand[currentPlayer][p] = state->hand[currentPlayer][p+1];
            }
```

Refactored dominion.c



```
    case baron:
       return playBaron(state, choice1, currentPlayer);
```

Danny Tran
SOFTWARE ENGINEERING II (CS_362_400_U2019)
Assignment-2.pdf



dominion.c × dominion.h playdom.c

dominion ▸ dominion.c ▸ ...

```
int playBaron(struct gameState *state, int choice1, int currentPlayer){…
}

int playMinion(struct gameState *state, int choice1, int currentPlayer, int choice2, int handPos){…
}

int playAmbassador(struct gameState *state, int choice1, int currentPlayer, int choice2, int handPos) {…
}

int playTribute(struct gameState *state, int nextPlayer, int *tributeRevealedCards, int currentPlayer){…
}

int playMine(struct gameState *state, int currentPlayer, int choice1, int choice2, int handPos) {
    int j = state->hand[currentPlayer][choice2]; //store card we will trash bug 1
    // int j = state->hand[currentPlayer][choice1]; //store card we will trash

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

Bugs

Minion

- The first bug was to swap the actions of either receiving 2 coins or the +4 cards actions. I swapped variables choice1 and choice2. This bug will go unnoticed by the program but the user will detect buggy behavior as their choices are not respected.
- The second bug is a subtle one where the drawCard function actually is called 5 times instead of 4. The player is supposed to receive 4 cards if they chose this effect.

Tribute

- Two subtle bugs: if the player chooses either receiving coins or actions the game state doesn't increment the player's coins or actions.

Baron

- The player gains an estate card regardless of whether they discard an estate card.
- Estates are incremented instead of decremented.

Ambassador

- The players don't receive the copy of the revealed card. gainCard is commented out.
- There is no break statement to stop the loop from discarding the cards. The player will have more cards discarded than is expected.

Mine

- The wrong card will get trashed. This is done by observing the trashed card that is stored in variable j is based on choice 2 instead of choice 1 which affects the card that the player discards.
- The second bug disables the effect that the player will gain any cards.