Joseph Devaney – OOP, Semester 2 Assignment: Test Plan

For a detailed walkthrough of this project and the critical classes and methods, please see my blog at <http://josephdevaney1.wordpress.com/>.

# Testing

Due to a limited amount of variable user input, there should not be many unexpected exceptions. In this test plan, I will test the start screen by inputting data in an incorrect format. Then I will play through the game a number of times trying to hit all of the possible situations. I will document correct behaviour and any bugs that need to be further tested and resolved.

Start Screen

This screen has four inputs for screen name, hero’s chipstack, villain’s chipstack and the size of the small bet for the game. Name relates to a string and the rest relate to integers. There is no explicit sanitization set to these text boxes, but WPF does most of this. Testing name with ‘Joe” drop tables’, and all of the rest with a character ‘z’ results in the name being applied, and the other fields being assigned the default values of 100, 100 and 2. One issue here is that if the name is too long, the textbox in the next window cuts the end off.

In Game

On the first run through under normal circumstances, the gameplay is as expected. The interface has some problems that will need addressing.

* There is no way to see what card(s) you have selected.
* There is no way to deselect cards.
* There is nothing documenting how many draws are left.
* There needs to be a lasting account of the previous actions taken by both hero and villain. This should last at least two full hands.

For the last point, there are plans to implement a text box in the lower left corner that displays all of the actions in text format. There will also be a new class to store the entire information of each hand. These will be stored in a collection in memory while the current game is ongoing, and afterwards they will be written to a text file for easy review by the user.

All user inputs act as they should, and the GUI updates as data changes.

All-in Edge Case

In poker, you can only play the chips you have on the table at the start of the hand. This leads to the concept of being all-in. This effectively means that if somebody else has made a bet for more chips than you have left, you can just call for your amount of chips. The remainder gets returned to the other player. This is a case that is far removed from the norm and as such, requires special testing.

**Case 1: Can’t cover mandatory blind**

**The player (user or computer) should just post any chips they have, and then if called, finish the hand sans betting.**

**Result: Works correctly**

**If the computer is the player that is all-in, the user is given the option to raise, despite that being a legal choice. If the user raises, the window stops because the computer knows they are all-in and so cannot make a decision.**

**Case 2: Call an all-in with less chips than the bet**

**In this scenario, player a makes a bet of, for example, 20 while player b only has 15 chips left. Player b is able to call, and if so, 5 chips need to be refunded from the pot and put into player a’s chipstack.**

**Result: Worked Correctly**

**Cases where a player bets or raises all-in works exactly the same way as case 1.**

**Player out of Chips**

If a player runs out of chips, the game is over and the other player wins. User should be allowed to start a new game from the start screen. Currently there is no functionality to make this work. The application must be restarted.

A future development of this will be to allow the user to continue the game by reloading his chipstack, or that of the computer.

Other

All of the data bindings between the GUI and the code appear to work correctly. Both chipstacks get updated, as does the pot. Hero’s name shows as whatever he/she chose at startup. The computers actions get shown via a textblock beside his seat. All Card Images show and update correctly.