**CIS 171**

**Final Project Documentation**

**John L. Duncan (jd0296887)**

**Design**

One should begin any project by having a solid understanding of the requirements. I thoroughly read and reread both the requirements and the rules for Blackjack; a basic outline of the design began to emerge. My other starting point in programming, before I even write the first line of the code, is to have a firm grasp on the data. If one truly understands the data they’re working with, the battle is half-won. From this, I began to discover what my classes would be – two Players (a Dealer and a Human Player), a Deck of cards, and the individual Games themselves. In opting for deriving the Dealer and the HumanPlayer from an abstract Player, I could pass either one into any method with a Player as a parameter – much simpler.

The game mechanics were more challenging. Ideally, one wants a firm separation of concerns between a display, and the mechanics behind the display. Since this is a GUI app, the display and much of the game rules became more enmeshed than I was comfortable with. The only way I could see around this problem would result in an even greater problem – circular dependencies. So, I had to live with my Game class confined to creating new instances of my players and deck, and allow the Display class to handle most of the heavy lifting of the game itself.

I opted to use an ArrayList object to handle the deck and hands. While there’s a little more processing overhead involved in the ArrayList versus a standard array, the increased flexibility to add or remove items at will and the ability to use the Collections class to handle shuffling was well worth it.

**Requirements**

Shuffling the deck is handled using the Collections.shuffle method in the Deck class. Dealing is handled by a joint effort of the Deck and Player classes – the draw method in the Player class receives a Card from the deal method of the Deck class. The current score is tracked in the Display method, displayed in the upper left corner of the screen. I also opted to implement Vegas decks. A message box is launched once the app is started to request a number of decks to be used in the game. This number is passed to the Deck class and determines the number of times a “for” statement runs through a deck creation sequence.

**Testing**

Testing was an on-going process throughout this exercise. I added a test class so that I could begin testing some of the game mechanics with the console directly without have to use the GUI. I would revert back to using this console “test box” a number of times throughout the process to get a more immediate response while debugging certain aspects of the game. Another helpful diagnostic tool was the message box; I could see what was happening to my data at critical points in my program in ways that might not manifest themselves in the GUI. I had volunteers run through using the nearly-finished app to help discover issues that I may have missed.

**Challenges**

The challenges were many, but were mitigated by giving some thought to my design architecture early-on. One hurdle encountered revolved around the Ace card – since it could represent either one or eleven, I had to implement a way to choose a value based upon circumstances. I represent the Ace as “1”, unless the total value of the hand is ten or less, in which case it’ll change to the greater value. The full solution may be found on lines 30 – 47 of the Player class. I also found the GUI challenging; creating those elements is much more difficult in Java than it is in C#. I also don’t really have the good sense of design to make an elegant GUI, but it is functional and “does the job”. My choice of which JPanel Layout to use was the one decision that I changed multiple times as I was creating the app.

**Conclusion**

This was a very enjoyable exercise that helped me to gain a better understanding of the language. I’m not as happy as I would like with the user interface; though it meets the requirements for included components, it just isn’t pretty. I was rather proud of using the modulus of thirteen to help generate all cards for each suit (a little trick I picked up a while back in one of my other classes). I also feel I could have done better to have a tighter separation of concerns between the user interface and the game itself, though on the whole I think at least some of that was unavoidable. On the whole, it was a great experience – challenging but worth the experience and knowledge gained.