Subhei Shaar
CS 162
2/20/15

Assignment 3: Design/Reflection

Understanding the Problem

Create Stud Poker

- Use a class for StudPoker
- Enable 2 – 6 players.
- Write functions to detect hand strength
- Enable betting and folding (you can bet more points than you have)
- Play n games specified by user

Create Poker class

- Make the base class Poker and have it include all of the protected member variables
- Virtual functions for play() etc.
- Stud Poker and Draw Poker are derived from Poker

Create Draw Poker

- Same thing as StudPoker except rules are changed
- 2 rounds
- 1st round each player gets 5 cards and must bet
- Able to replace 4 cards in hand after first round

Inputs:

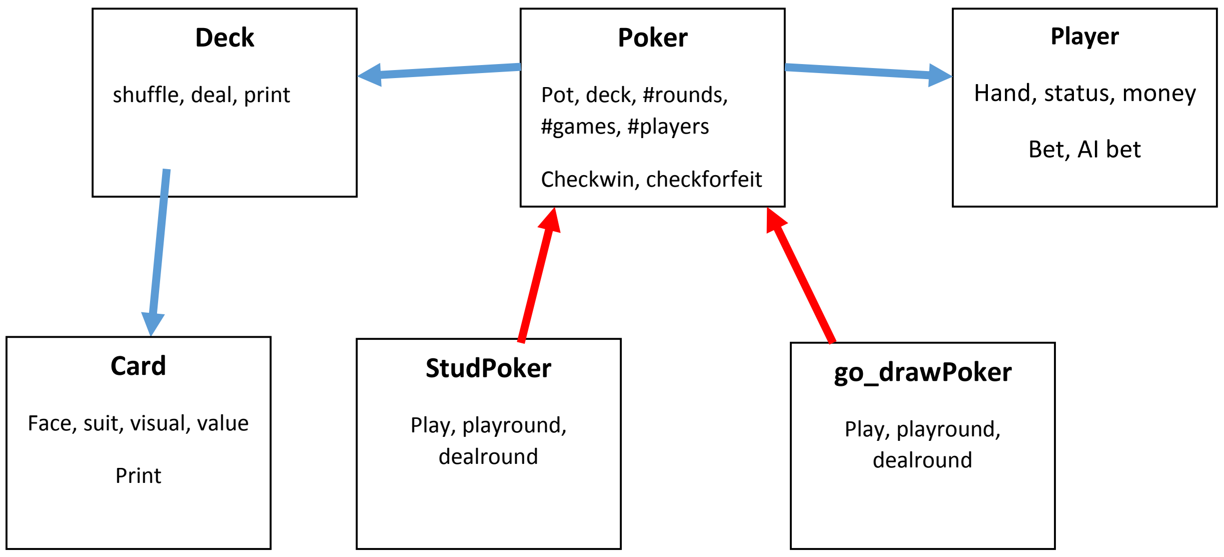
- Game type (Stud Poker or Draw Poker)
- Number of players
- Number of games
- Bet amount

Output:

- Cards in hand
- Betting amount
- Pot amount
- Display winner

Devising a Plan/Design

Class hierarchy



The blue arrows represent a “has-a” relationship between classes.

The red arrows represent an “is-a” relationship between classes.

Reflection:

In this program I followed the conclusion I came to during the last time that the problem becomes much easier to tackle when you separate your header files from your source files. I took it a step further and had a source file for each class containing the functions used by that class. All together I had 6 classes, Player, Deck, Card, Poker, StudPoker, go_drawPoker. This is the most logical class hierarchy I came to for playing poker. I think designing the classes really well allows for your code to be reused to create similar projects that need a certain object. After I had finished the Stud Poker coding I dreaded repeating the whole process again for draw poker. Once I started to create draw poker I realized I had built all the tools I needed for pretty much any type of card game and all I needed to do was to redefine the play function.

Testing:

Since this program has a lot of different components to test I focused on what I thought to be most important, determining the winner. The most challenging part of this assignment was to evaluate the hand to determine if the player hand a pair, 2 pair, triple, etc. I had to first sort the hand by value from least to greatest of the face. I assumed that aces were high and 2’s were low. After sorting I wrote many conditional statements to check if the next card in the hand was the same value as the current card in the hand. If this was true then the hand is said to have a pair.

After determining the strength of each players hand I also needed a way to resolve ties. For example if two players both had pairs who would win? I wrote another function that would check if another player also had the highest strength for that round then evaluated the decks side by side to determine who had the stronger deck. I also implemented suit detection which means there can’t be a tie in the game. I realize that suit detection is a problem that doesn’t matter most of the time because it is extremely rare for 2 players to have the same pair in a round.