Project 3: Poker Tournament

# **Due: Sunday of Week 9 (10/28) by 11:59 PM**

# **Introduction:**

A couple of springs ago an AI named Libratus defeated four professional poker players in a tournament. (For more details see this article: <http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/ai-learns-from-mistakes-to-defeat-human-poker-players> .) In this case, the players were playing Texas hold ‘em, which is considered a somewhat more difficult type of poker game.

For the next project we are going to use various decision-making techniques to play a simpler version of poker. I’ve created software that will guide the process of playing a two player version of poker. Your part in this will be to create a player that uses some of the decision-making techniques to guide three major parts of their gameplay: initial betting, replacing cards, and final betting.

The version of poker we are playing is called Five Card Draw Poker. For those of you not familiar with the game of poker the rules are given below. In order to test out everyone’s creations we will be holding a single elimination tournament. For this tournament each AI will be paired up with another AI, using the tournament software to play the game. The winner will advance to the next round. Eventually one AI will emerge as the winner.

# **Five Card Draw Poker Rules:**

From: http://www.pokerlistings.com/poker-rules-5-card-draw

**Five Card Draw is one of most basic forms of poker, and it's the kind of poker you're used to seeing in movies and on TV.**

Because the game has been around for such a long time, and has been played in countless home games and card rooms across the nation, there are a couple different variations on the traditional rules.

**Below you will find all of the most commonly used rules for playing Five Card Draw.**

The game is simple: make the best 5-card poker hand possible after one draw, and bet accordingly. The player with the best hand after the second betting round takes the pot.

## **Antes:**

The **ante method** is the original way the game was played, and is most commonly the system used in home games around the world. **In this version each player must pay a predetermined ante before being dealt any cards.**

## **The Deal:**

Once all players have paid their ante the dealer deals every player (starting on his left) five cards face down.

**After all players receive their five cards, the first betting round ensues.**

The betting must start with the player to the left of the dealer.

**In an ante game like this, the first player to act is allowed to check** (meaning they are not forced to bet, and can choose to stay in the hand for free).

So in the betting round there are certain commands players are allowed to do:

|  |  |
| --- | --- |
| **Command** | **Definition** |
| Check | The player does not bet anything but can remain in the game. Note that if both players check, the betting round is over. |
| Bet | The player states an amount that they wish to bet. Other players can call this amount or raise it. |
| Call | If a player calls a bet, it means they’re going to pay the same as what the other player just bet. A call ends a round of betting. |
| Raise | If a player says they’re going to raise, it means they are going to increase the amount of the bet by a certain amount. If the other player wishes to remain in the game they need to either call the bet or raise still further. |
| Fold | If a player says they are going to fold it means that they have decided not to play this hand any further and are conceding this round. The remaining player gets all the money from the pot and a new round starts. |

## **The Draw Round:**

When the betting round completes, the draw round begins with the player closest to the dealer's left. Assuming this player hasn't already folded, they have the option of changing any amount of cards they choose.

A player can "stand pat", meaning they keep all five cards, or **they can throw away any amount from 1-5 cards**, getting them replaced with an equal number of cards from the top of the deck.

It is almost never correct to call in the first betting round, only to throw away four or all five of your cards.

## **The Second Betting Round and Showdown:**

Once all players have received their new cards, each player must evaluate their hand and proceed to the second (and final) betting round. This betting round works the same way as the initial one. Once the second betting round is completed it's time for the showdown (assuming more than one player still has a hand).

Just like almost every other form of poker, the player with the best 5-card poker hand at showdown wins the pot. [**Head here for a complete list of poker hand rankings**](http://www.pokerlistings.com/poker-hand-ranking/)**.**

**Once the showdown is complete, the dealer collects all the cards, and passes the deal on to the player to his left.**

**Hand Strength**

|  |  |
| --- | --- |
| **Royal Flush** | A straight from a ten to an ace with all five cards of the same suit. In poker all suits are ranked equally. |
| **Straight Flush** | Any straight with all five cards of the same suit. |
| **Four of a Kind** | Any four cards of the same rank. |
| **Full House** | Any three cards of the same rank together with any two cards of the same rank. Example: 3 Kings and 2 Aces |
| **Flush** | Any five cards of the same suit (not consecutive). The highest card of the five determines the rank of the flush. Our example shows an Ace-high flush, which is the highest possible. |
| **Straight** | Any five consecutive cards of different suits. Aces can count as either a high or a low card. Our example shows a five-high straight, which is the lowest possible straight. |
| **Three of a Kind** | Any three cards of the same rank. Our example shows three-of-a-kind Aces, with a King and a Queen as side cards - the best possible three of a kind. |
| **Two Pair** | Any two cards of the same rank together with another two cards of the same rank. Our example shows the best possible two-pair, Aces and Kings. The highest pair of the two determines the rank of the two-pair. |
| **One Pair** | Any two cards of the same rank. |
| **High Card** | Any hand not in the above-mentioned hands. |

# **Tournament Software:**

The tournament software is written in C# using Visual Studio 2015. The software consists of a number of classes that are briefly summarized in the table below:

|  |  |
| --- | --- |
| **Class** | **Description** |
| **Program** | This class will be used to select the two players who were running in a section of the tournament. It will also be used to start the game and report the results. |
| **Card** | This class represents a single playing card. |
| **Deck** | This class represents a deck of cards. It has methods that will be used to shuffle the deck of cards and deal cards. |
| **Evaluate** | This is a class of static methods that you can call to evaluate the hand that you have been dealt. |
| **Game** | This class runs you through the steps of a poker game, from anteing, receiving cards, betting, exchanging cards, betting again, in determining the winner. |
| **Player** | This abstract class is the parent of the class you will be creating. There are three abstract methods that you need to implement that will each be used to make decisions at different parts of the game. |
| **PlayerAction** | This class is used to create objects that record player actions as a round of the game occurs. This information will be available to both players in the form of a list of player actions. |
| **Human** | This is a concrete subclass of the Player class. It is provided to allow you to test your code by manually playing against your AI. |

**Note: all of the classes that are listed in the table above cannot be altered by you as part of your development process. The tournament itself will be run using these classes and any changes you may make in your own personal development environment will not appear in the final tournament version.**

You should take some time to look through the code to see how the list of player actions is recorded and to see how the game itself actually gets played and evaluated. Should you see any issues please contact me immediately so that they can be addressed.

Make sure you look at the Human class carefully, because it shows you how player actions are created and sent back to the Game class for use by the tournament software.

# **Requirements:**

1. You will create a class that has the Player class as a parent. This class will implement the abstract methods defined inside of the Player class. You can use any of the decision-making approaches that we have discussed in this course to implement these methods.
2. Your class name will be PlayerN, where N is the team number that was assigned to you when teams were signed earlier in the course. You will find the team list at the end of this document in case you forgot your team number.
3. Make certain that your program works with the tournament software. This means that you should be testing it against the human class that was provided. This will allow you to interact with the AI.
4. As noted in the previous section, you may not alter the tournament software to get your program to work. Your code needs to work with the existing software and it will not be customized just to make your code work.
5. You are expected to follow the coding standards that you were taught in your introductory programming course. This means variable names should be self-descriptive, indentation should be used to show program structure, and your code should be commented.
6. Write a short two page summary of the decision-making approaches you used for the betting processes (you may have used different decision-making for the first and second betting steps) and for the decision which cards to discard.
7. You will be submitting a zip file that contains all of the code for your Player subclass. Do not submit the entire project. I only need the specific subclass that you created that implements the decision-making process. This zip file should also contain the two page summary from item 6 above.

# **Deliverables:**

The deliverable is a single zip file with the code for your custom subclass and the short summary of the decision-making process.

# **Grading Rubric:**

Names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Max Pts.** | **Earned Pts.** |
| Code follows programming standards in terms of indentation, variable names, and comments. | 10 |  |
| Code implements decision-making for the first betting process using one of the approaches from class | 20 |  |
| Code implements decision-making for the card replacement process using one of the approaches from class | 20 |  |
| Code implements decision-making for the second betting process using one of the approaches from class | 20 |  |
| Code works correctly with the existing tournament software | 10 |  |
| Code executes correctly and generates valid results | 10 |  |
| Two page summary of decision-making process was included | 10 |  |
| Total Grade: | 100 |  |

Comments: