Project: Multiplayer .NET Remoting Poker

Problem Description

The aim of this project is to develop a distributed .NET application using the .NET remoting infrastructure that implements the behavior of a multiplayer Poker game. The application will consist of two parts: a server console application that represents a Poker dealer and a Windows.Forms client application, which implements all necessary features of a Poker player.

Poker is a round-based game. That is, the dealer starts a new game by informing all participating players to make their initial bet. The dealer has to wait a certain amount of time before all players may have responded. If a player has failed to make his/her initial bet in the given time, he/she will be excluded from the next game. The dealer will then start to deal cards beginning with the first player (every player is assigned a position). Every eligible player will receive 5 cards.

In the next step, the dealer starts the first betting round (in clockwise order). The dealer will ask every player first whether he/she wants to discard any cards. A player can discard up to three cards. The dealer will replace the cards instantly. Secondly, the dealer will ask the player for his/her betting choice (Call, Raise, or Fold).

The betting process involves a rather complicated protocol. If Call is chosen, then the player has to put the amount he/she owes into the pot. No other player will be affected. If Fold is chosen, the current player will instantly drop out of the game. If Raise is chosen, then the current player has to put enough credit units into the pot to match what has been bet and then he/she would raise the pot by an additional amount. This choice has a direct effect on every other player, since they have to react immediately in order to stay in the game. That is, every other player has to choose either Call or Fold. The dealer will wait until every player has made his/her choice.

When the first betting round is completed the dealer will start a second one. In the second round the players cannot again discard cards. A player can choose either Call, Fold, or Raise. The chosen action will be handled with as described above.

Finally, the dealer will ask every remaining player to reveal his/her hand. The dealer will compare the hands and will determine how has won the current game. The winner will receive to pot, that is, the player’s credit will be increased by the pot’s value.

Every player has to return his/her cards to the dealer. If a player fails to return his/her cards, then the dealer’s deck will be invalidated. That is, the dealer will discard the actual deck and will create a new one (just like in real life, where the
dealer will start with a new deck, if the old one has been tampered with in some way).

The server console application has to register a server-activated factory remoting object with the .NET remoting services. The SAO will use a HTTP channel that will listen at port 7000. The class that implements the dealer has to be derived from \texttt{MarshalByRefObject}.

Additionally, the server has to provide support for a game chat room. That is, while the server is running, players should be able to exchange messages.

The player client implements a Windows.Form GUI. The client is event-based. That is, the client has to react on Windows events, dealer events, and chat events. A client has to register with the server in order to participate in the game. The player that is assigned to number one will be the \textit{leader}. The leader can ask the dealer to start the game. Note if the players are not ready yet, then it makes no sense to start it at all.

When a player registers with the server, the dealer will assign the initial credit to this player. The initial credit may be a default value or an existing balance that has been stored in a database. The latter optional approach will enable a scheme in which a player can keep his/her credit over time, and more importantly the dealer may grant players a loan in order to continue to play. Loans have to be repaid, may have attached an interest rate, and may have other effects on players like an imposed \textbf{Raise} limit.

You can use everything that has been shown in class or was part of an assignment. Every student has to work on his/her own project, but the discussion of ideas is allowed. Note, however, that your solution must be original. That is, it must reflect your own ideas and work.

You have to document your project. That is, you have to write a report that will contain:

- A requirement specification,
- A design specification,
- Charts that illustrate the interactions between dealer and player and players in general,
- Discussion of special features of your application, and
- Open problems and possible future work.

The report should consist of approximately 15-20 pages.
Basic Rules

Cards and Poker Hands

Standard Poker uses a pack of 52 playing cards. The card ranking is as follows: Ace (the highest), King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, and 2 (the lowest). There are four suits (Spades, Hearts, Diamonds, and Clubs).

The number of cards, which is dealt, depends on the type of the game. In the project we use “5 card stud”, that is every player is dealt 5 cards face down (i.e. the other players cannot see, which cards the current player is dealt). Therefore, every poker hand consists of five cards, and there are at most five hands, where the highest hand wins.

Ranking of Poker Hands

1. Royal Flush (score 10) - A straight to the Ace in the same suit. In case of a tie, that hand with the higher-ranking suit wins.
2. Straight Flush (score 9) - A straight in the same suit. In case of a tie, that hand with the higher-ranking card wins.
3. Four of a Kind (score 8) - Four cards of the same number. In case of a tie, the hand with the higher-ranking “Four of a Kind” wins.
4. Full House (score 7) - “Three of a Kind” with any pair. In case of a tie, the hand with the higher-ranking “Three of a Kind” wins.
5. Flush (score 6) - 5 cards of the same suit. In case of a tie, that hand with the higher-ranking card wins.
6. Straight (score 5) - 5 cards with consecutive numbers. In case of a tie, that hand with the higher-ranking card wins.
7. Three of a Kind (score 4) - Three cards of the same number. In case of a tie, the hand with the higher-ranking “Three of a Kind” wins.
8. Two Pair (score 3) - Any pair with any pair. In case of a tie, the hand with the higher-ranking pair wins.
9. Jacks or Better (score 2) - A pair of Jacks, Queens, Kings, or Aces. In case of a tie, the hand with the higher-ranking pair wins.
10. Highest Card (score 0) - A hand that does not even qualify for “Jacks or Better”. The hand, which is holding the highest-ranking card, wins. A tie in a game can always be broken, since suits are ordered (Spade is the most valuable suit).

Betting

A new game starts with a betting round. Every player has to put an initial ante into the game’s pot to qualify for the game. The ante is taken from the player’s credit, if the player has still enough credit. The game host (dealer) specifies the initial ante.
Betting is handled in a clockwise order. A player has three choices:

1. **Call** - A player has to put the amount that he/she owes into the pot in order to stay in the game. For example, if the current player has bet so far 25 credit units and somebody has bid 35 credit units he/she would owe 10 credit units.

2. **Raise** - First, a player has to put enough credit units into the pot to match what has been bet and then he/she would raise the pot by putting an additional amount. The host (dealer) specifies the maximal amount the pot can be raised. For example, assume the initial bet is 25 credit units, the current player has to match these 25 credit units first and than he/she will raise by 15 extra credit units (the amount must not exceed the current raise limit), which increases the pot’s value to 40 credit units.

3. **Fold** - The current player drops out of the game losing the possibility of winning the pot. The player does not need to put anything into the pot, but he/she will lose everything that he/she has already been put into the pot.

---

**Milestones**

Announced in class.

**Deliveries**

- A package (ZIP-file) that contains both the applications and the source code.
- A written report.

**Important Dates**

- 11-18-02/11-22-02: Lab-session - project presentation
- December 6, 2002: Submission of the project package.
- 12-09-02/12-13-02: Lab session - project presentation
- December 13, 2002: Submission of the written report.