ITCS 3166

Final Project

This project is an extension of the ideas used on the Midterm Project. By now you are expected to fully understand the implementation of a server-client program. Therefore, we'll now take it a step further.

So for this project, we're going to make a game. This game will make use of a server to join *two* clients together for a modified game of Blackjack. I'll explain how this modified game works below.

Here's your task: You're going to create exactly two classes. One class will act as the functional server. The main method of this server will setup the connection on a localhost, find the two clients, let them know which one is Player 1 and which one is Player 2 (depending on the order they connect), and then manage the game as it's being played.

This is important: For each player's turn they should start by displaying their current score to the user. Then, the user should be prompted to select whether they want to "Hit" or "Stand". This option selection can be done any way you choose, but invalid entries **must** be accounted for*. If the player chooses to "Hit", the server should deal a "card" to the player, which will consist of choosing a random number between 1 and 13. The server sends this "card" to the player, and the player receives the card and adds it to their score. If the card is above a 10, then add a 10 to the player's score instead of the original value (In a standard Blackjack game, Jacks, Queens, and Kings still count as 10). After they receive the card or if they choose to "Stand", the next player should take their turn. If either player takes a "Hit", and the card they're dealt raises their score total above 21, then the game should immediately end and the other player is the winner (this is known as a "Bust"). Likewise, if both players choose to "Stand", then both players should be shown the final scores and the game is then over.

*for example, if you choose to say "Enter (1) For a Hit or enter (2) to Stand", then any entry other than 1 or 2 should be counted as invalid and should loop to allow the player another entry. If you choose something like "Enter (H)it or (S)tand", then valid entries should be "H", "S", and ideally "h" and "s".

You May: Get user input in any way you choose, though you must check for invalid entries and prompt a re-entry if invalid data is entered; you may choose to implement an additional feature that allows a 1 (Ace) to be worth either 1 or 11, however this is not necessary; you may choose to deal two cards to each player to begin the game, however this is not necessary; you may choose to display the first card drawn to each player to both players, however this is not necessary; you may use driver classes and additional methods other than the main method in each class, however this is not necessary to complete this program.

You May Not: Exceed the scope of a console-based application – this means no GUIs, etc; you may not allow for more than two players by any means; you may not use two different client classes – one client must function as either player; you may not use or copy code from ***any*** source, including but not limited to other students, textbooks, online sources, activities from this class or any other class, or any other source. Copying code will result in a 0 grade on this assignment and possible other consequences.

While you're writing this program, keep the following things in mind:

- Socket, ServerSocket, and all forms of I/O must be closed when your connection is terminated.
- Your main method must catch any possible exceptions that may be thrown; it cannot throw an exception as this is bad practice.
- Please keep your program code neat and organized. Good use of white space goes a long way.
- Do not keep any unused variables or imports around – this is bad practice.
- Sample Output is included on the next page.

Grading Rubric

This assignment is worth 30 points. The point values are graded based on this rubric.

Connectivity	10 points	The server opens a port on a localhost, and the client successfully connects. The program successfully maintains the connection between the server and the two clients while the game is running.
User-Input	4 points	The client program successfully obtains user-input, and data validation is done correctly to validate the input obtained. The user is asked for new input if invalid input is given.
User-Friendliness	6 points	The look-and-feel of the program runs well; the client sends good user-friendly output; the server relays information to the console. Generally output to the server's console any time data is sent or received, or a connection is made or terminated.
App Reliability	5 points	The program correctly plays the modified Blackjack game in accordance to the above rules.
Good Practice	5 points	No dead code; no unused imports or variables, IO and Sockets are correctly closed. Good documentation.
Total	30 points	

Client Output (First Run of Client Class)

```
Fetching server connection...
Connection Established.
Fetching player ID...
Game Found.  You're Player 1.
Waiting for server...
It's your turn.  Enter 'H' for a Hit
or 'S' to Stand.
Score:0
H
Drew a 9.
Waiting for Player 2...
Player 2 took a Hit.
It's your turn.  Enter 'H' for a Hit
or 'S' to Stand.
Score:9
H
Drew a 10.
Waiting for Player 2...
Player 2 took a Hit.
It's your turn.  Enter 'H' for a Hit
or 'S' to Stand.
Score:19
a
Input is invalid.
It's your turn.  Enter 'H' for a Hit
or 'S' to Stand.
Score:19
S
Chose to Stand at 19.
Waiting for Player 2...
Player 2 took a Hit.
It's your turn.  Enter 'H' for a Hit
or 'S' to Stand.
Score:19
S
Chose to Stand at 19.
Waiting for Player 2...
Player 2 took a Stand.
Game Over.
Player 1's Score:19
Player 2's Score:18
Player 1 won!
```

Server Output

```
Server started.  Finding Clients...
Found Client 1.
Found Client 2.
Initiating Game...
Player 1's Turn.
Got 'H' from Client.
Player 1's Score: 9
Player 2's Turn.
Got 'H' from Client.
Player 2's Score:6
Player 1's Turn.
Got 'H' from Client.
Player 1's Score: 19
Player 2's Turn.
Got 'H' from Client.
Player 2's Score:10
Player 1's Turn.
Got 'S' from Client.
Player 2's Turn.
Got 'H' from Client.
Player 2's Score:18
Player 1's Turn.
Got 'S' from Client.
Player 2's Turn.
Got 'S' from Client.
Game Over.
Player 1's Score:19
Player 2's Score:18
```