RMIT University

COSC1176/1179 – Network Programming

Assignment

Aim: To apply the learnt networking concepts to a practical application

Method: This assignment will be attempted by students individually.

Due date: 18 May, 1:00 pm

Submission Method

- Your submission should consist of two parts: (i) all java files and (ii) documentation. Each part should be in a separate directory of its own. You should submit separate Java code for the different stages of the solution, as explained in the specification.
- You should put everything into one zip file, and submit the zip file. That file should be named as ***sxxxxxxx.zip***, where ***xxxxxxx*** is your student number.

Marks: The assignment is marked out of 100, and contributes 25% to your final mark.

Late submission penalties apply
Unless an extension has been granted (for procedures and grounds see http://www.rmit.edu.au/students/assessment/extension), a penalty of 10% of the total project score will be deducted per calendar day, and no submissions will be accepted 5 days beyond the due date.

Special Consideration

With the exception of dire circumstances, no extension requests will be considered within 5 working days of the submission date. ("Dire Circumstances" means things like hospitalization of you or a close relative, etc.) Persons requesting a late extension may be required to prove that a significant body of the work has already been completed.

Introduction

In this assignment, you are to develop a card game, a slightly modified version of Blackjack, played over the network.

General Specification

Your task is to implement a simple game with separate client and server machines. There are two roles in the game: player and dealer, each represented by a program. You have to implement the game in two stages.

> **Stage 1:** Single-player game: one player against the dealer. When you have completed stage one, save a copy of it for submission.
>
> **Stage 2:** Multiplayer game: you need to modify the program so that several (less then five) players can play against the dealer and against each other at the same time.

The game

It is played with a pack of French cards that consists of 52 card decks. Each deck has four suits and each suit has 13 ranks. The value of face cards (king, queen and jack) is counted as ten points, an ace can be counted as one or 11 points (whichever is more favourable). Other ranks have their values written on them explicitly, from two to ten. In this game the suit of a card has no role. The pack can be considered as containing a large enough (infinite) number of decks, so you don't need to keep track of cards already played and all cards come up with the same probability. There are no jokers in the pack.

The result of the game is determined by evaluating the hands: adding the values of the cards in the hand together. If a hand's value is over 21, it is considered 'busted' and that means losing the game.

At start both player and dealer get a two-card hand. In the next phase the player can ask for one or more cards or may opt for no additional cards, the decision is entirely up to the player. Once the player has finished, it is the dealer's turn to ask for more cards. The dealer, however, must ask for more cards if its hand is 17 or below. Once the dealer has also finished asking for more cards, they evaluate the hands. If neither player nor dealer has busted, the winner is the hand with the higher value. In case both have the same number of points, it is a draw i.e. no-one wins or loses.

Multiplayer version

In this case there are up to five players and one dealer. The result is evaluated the same way as before, i.e. the highest hand value among the non-busted participants (players and dealer) wins the game.

Players have to sign up for the game, and the dealer starts a game with those who have signed up before. If a new player signs up while a game is played, the new applicant will be put on a waiting list, and will be accepted for the next game. Players who finished a game are not signed up automatically for the next game, but they can do so manually.

The sequence of the game is as follows. The two-card hand is given to all players and then to the dealer. Next, the players can ask for new cards. You can develop this in two steps.

(i) The players go in sequence, i.e. the first player gets the chance to ask for new cards, once finished, the second player can start and so on, until all players had a turn, and finally the dealer can ask for more cards.

(ii) There is no synchronization between the players, they can send requests that overlap in time. Once all players have finished, it is the dealer's turn for getting more cards. When the dealer has finished, the hands are evaluated and the result announced.

Again, save a copy of the step 1 solution for submission.

You need to submit the second step only, if it works. If the second step doesn't work, you can submit the first step.

Program components

The client

You will have two types of clients: player and dealer. The dealer has a double role. First, it takes care of the card pack, and every participant has to ask the dealer for cards. At the same time, the dealer is also playing the game, with slightly more restrictions, i.e. has to ask cards if under 17. The dealer should be able to work without any human input.

The players are just playing the game as per the specification. The players should be directed by humans, i.e. a human operator can decide when to sign up for a game and when to ask for more cards.

The server

The orchestration of the game should be done by the server. It includes keeping track of players, managing the sign-up list, conducting the game and evaluating/announcing the result. It can be considered as an intermediary between players and dealer:

Logging

The server logs the events in the game in both single player and multiplayer cases. There will be two separate logs, one for gaming and one for communication. The separation of logs is for security reasons, those having access to the communication log will not be able to learn about the intricacies of the game and vice versa. Each log entry should contain the date and time, the remote socket address and the action performed (e.g. "Card given").

The platform

You have to use Java for programming all components. Threads have to be used. The communication is via sockets. You need to choose the appropriate socket for each task, and explain why that socket is the most suitable.

Documentation

You need to prepare a short user instruction sheet, and a longer document describing how the program operates. You can attach a class diagram, if it makes it easier to explain your program's operation. You also need to insert comments into your program, as appropriate. The documentation should be no longer than two pages.

Proper commenting of the source code is expected.

Miscellaneous

A simple text user interface will suffice. The user (player) should be notified about events and messages sent by the server. This includes information sent by the server about other players in the multiplayer game.

The program should have a reasonable timeout (up to one minute) when waiting for user input or expecting a server message, when appropriate.

Notes

- In the multiplayer version, you can assume that no client request comes between starting the game and registering the game's state as being played. This is a simplification for this assignment only, in real life cases you need to use appropriate concurrency control.
- Your program will be tested on the Moonshot servers.

Mark allocation

Single player game: 60 marks

Multiplayer game

 Stage 1: 25 marks

 Stage 2: 15 marks

The End