Playing Nim in Client-Server Mode

This project will be challenging because it involves merging features from two programs you have examined in the tutorial. You have studied the game Nim that can be played locally on a computer. You will now distribute this program by putting most of the components on the client-side and a few components on the server side. The server-side main method will be modeled after the main method in the BroadcastEchoServer example you have studied. The client handler will be similar but easier since, in the initial version, the clients do not have to talk to each other.

(Step 1) Changes to Nim on the client side
- For clarity, rename the class, constructor, and file to NimClient
- Add import of objects from java.net and java.io
- Add an init method that performs the following tasks:
  - Creates a new socket (use localhost)
  - Creates a DataInputStream and DataOutputStream for this socket
  - Reads an integer from the server and stores it as the client’s id
  - Sends an integer, the maximum move amount, to the server
  - Uses System.out.println to write a message displaying the client id
- Add a sendMove method with an int parameter that sends the parameter value to the server as the player’s move; you may want to print a message to the screen to help trace program execution
- Add a receiveMove method that returns an int value representing the computer move, which is received from the server; add a print message to trace execution

(Step 2) Changes to the NimPanel on the client side
- Put in a getter method, getMaxMove, that will be called by NimClient
- Remove the method getMachineMove, but save it as you will be putting it in on the server side shortly
- Remove the if command that called getMachineMove; in its place put in a call to thisGame.sendMove(numPieces); followed by machineMove = thisGame.receiveMove();

(Step 3) Creating a NimServer class
- This class is almost identical to the BroadcastEchoServer shown in class
- The only changes needed are in names and messages, such as BroadcastEchoServer becomes NimServer and the System.out.println messages are worded to fit the Nim game

(Step 4) Changes to the NimClientHandler
- Change names and messages as appropriate for the Nim environment
- Remove the putMessage method; in its place you will need two methods: putInt and getInt to communicate integer values to and from the client; these methods are comparable to sendMove and receiveMove in NimClient
- Put the getMachineMove method inside this class
- The run method is completely different, as indicated in the description below; fortunately it is also much simpler
  - Declare two int variables for numPieces and maxMoves
  - Print a message that the player with the specified id has been started
  - Put the following sequence of operations inside an if command that insures the input and output streams are not null
Send the player id to the client
Receive the value for maxMove from the client
There are only two operations that are put inside of an infinite for loop: receive numPieces from the client and send to the client the result of calling machineMove passing in the values numPieces and maxMove

Test your program and make sure it works correctly by playing many games. Try to win the game then try to lose the game. Make sure the messages displayed as you play the game make sense.

You should be able to run several clients at the same time where each of the games are totally independent. Try this and make sure everything works properly.

**BONUS SECTION: Redo the server side so that two human clients can play the same game against each other.**

To keep it simple, let’s assume the first player that signs onto the server is number 1 and all subsequent players are one higher than the previous player. Also no player id’s are reused once they are assigned. The opponent for an odd number player will be the next higher number and the opponent for an even number player will be the next lower number (i.e., 1 plays 2, 3 plays 4, etc.). Rather than have a getMachineMove method, the nimClientHandler will pass the move from one player to the other, as appropriate.

Good luck on this bonus. Once you have the program working, you can challenge your friends to a game. With a little more work and access to a server machine, you can even play opponents over the Internet.

**CONGRATULATIONS! Your have completed the lab project associated with the third (and final) session of the JavaJumpStart Tutorial.**