CSci 364
Spring, 2023
Programming Assignment #1
Due: See Blackboard
Points: See Blackboard

Implement the TCP client of a client-server architecture that plays the high-low guessing game (as discussed in class). You must implement your client using Java 17 JDK (or higher) and utilize a Java socket (java.net.**Socket**). Your client must inter-operate with the server provided with this assignment.

Server Details
The server reads two command line arguments: a TCP port number, and a value greater than zero representing the maximum of the guessing range. The server will exit if either of the command line arguments is invalid.

The server selects a random target integer between 0 and the maximum value from the command line, binds to the specified TCP port, and manages the game on behalf of your client. The server will compare the client's guess with the target and send messages to your client indicating the direction for the next guess. If the client sends an invalid guess (out of range or a string that cannot be converted to an integer), the server will send a message beginning with "Error."

After the client sends to the server the correct guess, the server will send a message acknowledging the correct guess and the number of guesses. The server will then ask the client if the client user wants to play another game. If the client sends "y" or "Y", the server will generate a new random target for the next game.

Client Details
Your client shall read read two command line arguments: the server hostname or IP address and an integer representing the server's TCP listening port. After validating the command line arguments, your client should connect to the server.

All messages sent from the server must be echoed to the client console.

After receiving a welcome prompt from the server, your client should read an integer guess from standard input (entered by you via the keyboard) and send that guess to the server. If the client user enters a non-integer guess, the server will respond with a message beginning with "Error." Your client should read another guess from standard input and send that to the server. When the client user guesses the target, the server will acknowledge the win and send a question whether to play again. If the client sends "y" or "Y", the client and server can play another game. If the client sends something else, both client and server should close objects and exit.

Testing
Develop and test your client by running the server in one console window and running your client in another console window.

Assume the server is started as follows.

```
[in a console window]
$ cd [directory where GuessServer.class has been placed]
$ java GuessServer 4444 15
```

The client console window should have output similar to the following.

```
[in another console window on same computer]
$ cd [directory where build.xml is]
$ mkdir src
```

Create and develop your GuessClient.java file in the src directory. The Ant script presumes your source file is in the src/ directory.

```
$ ant
$ java -cp ./build GuessClient localhost 4444
Server: Welcome. Guess a number [0, 15)
Client: 5
Server: Guess lower
Client: 3
Server: Guess lower
Client: 1
Server: Correct. 3 guesses.
Server: Play again? y|n
Client: n
$
```

Grading Rubric

- Each source code file must begin with a comment block stating name of file, a short summary, and the author's name.
- Source code must be consistently and properly indented.
- https://en.wikipedia.org/wiki/Indentation_style#Variant:_1TBS_(OTBS)
- Programs that do not compile will receive 0.
- If your program crashes while I run it, I will do some investigation. Most likely, this will result in a 0.
- Do not import packages with a wildcard: "import java.io.*;" You are required to name each individual class imported: "Import java.io.BufferedReader;"

Submit to Blackboard

Make sure your name is in your source code files. Use meaningful filename (not Client.java).

Submit your src/ directory (with Java source code file(s)) and your Ant build script using the directory structure below. Your Ant script should have targets to clean and compile the source code. These files should be submitted as a zipped or tarred file:
ex. **lastname-firstname.hw1.zip** or **lastname-firstname.hw1.tar**.

```
hw1/
        build.xml
        src/[Java source files]
```