CSci 364
Spring, 2025
Programming Assignment #1
Due: See Blackboard

Implement the knock-knock joke message-exchange pattern using a client-server architecture and Java sockets (java.net.Socket and java.net.ServerSocket). Use a Java 14 JDK or higher.

**Server Set-up Details**
Your server shall read three command line arguments: 1) an integer TCP port to which the server will bind, 2) the name of a text file containing several knock-knock jokes, and 3) a long seed value for a random number generator to select the next joke.

Your server shall check that the correct number of arguments have been entered on the command line. Also, check that the numbers are parsable (java.lang.**Integer.parseInt(String)** and java.lang.**Long.parseLong(String)**) and that the jokes file exists (java.io.**FileReader(String)**). If any errors with command line arguments exists, catch the appropriate exceptions, print a helpful message to the console, and exit.

Each line in the jokes input file has a joke setup and a punchline separated by a ‘#’ character. Read all of the lines from the jokes input file into an appropriate data structure that can be indexed by an integer. You may find the java.lang.**String.split(regex)** method helpful in parsing each line in the file.

Your server should bind to its port and print a message to the console indicating on which port the server is listening. The server should print the client’s host name and port once the client connects. The server only needs to handle one client at a time (a single-threaded server). The server should set up a java.io.**PrintWriter** and a java.io.**BufferedReader** so it can read messages from and write messages to the client.

For the final step in preparing for the Knock-Knock message exchange is for the server to obtain a random index to the data structure and retrieve the set-up and punchline to a joke. See java.util.Random(long) and java.util.**Random.nextInt(int)**.

Be sure your server implementation only instantiates a single Random object.

**Client Set-up Details**
Your client shall read two command line arguments: the name or the IP address of the server and the server’s TCP port.

Your client shall check that the correct number of command line arguments have been entered. Also, check that the port number is parsable. If any errors exists with command line arguments, catch an appropriate exception, print a helpful error message, and exit.

Your client should connect to the server using the entered server host and port. The client should setup a java.io.**PrintWriter** and a java.io.**BufferedReader** so it can read messages from and write messages to the client. The client shall also setup a java.util.**Scanner** object so it can read messages typed at the keyboard.

**Message Exchange Details**
Your client and server should exchange messages following the Knock-Knock message exchange pattern below. The pattern starts with the server sending a message, “Knock knock.” The pattern ends with the server sending a “Bye.” message.

If the server or client receives a message that varies from the specified exchange pattern, it should print a message to its console and exit.

| Message Number | Client | Server |
|---|---|---|
| 1 | | Knock knock. |
| 2 | Who’s there? | |
| 3 | | <set-up line> |
| 4 | <set-up line> who? | |
| 5 | | <punch line> |
| 6 | | Bye. |

The server shall print messages to its console that are received from and sent to the client. The client shall read messages entered by the client user and send verbatim to the server. The client shall also print messages to its console that are received from the server. After the server sends a “Bye.” message it shall close its connection with the current client and wait to accept a client from another connection.

Assuming the following project directory structure, the following commands should allow you to compile and run your server and client.

```
hw1/
        build.xml
        jokes.txt
        src/<Java source files>
```

If you use the sample Ant script found on Blackboard, the following commands should run your server and client.

In a console window…

```
$ cd <directory where build.xml and jokes.txt are>
$ ant
$ java -cp ./build JokeServer 4444 jokes.txt 1
```

In another console window…

```
$ cd <directory where build.xml and jokes.txt are>
$ java -cp ./build JokeClient localhost 4444
```

**Submit to Blackboard**

Add a comment box to the top of each source code file. The comment box should include your name and a brief description of the file contents. Finally, make sure your code has consistent indentation.

Do not submit .class files. Clean your project before creating a .tar or .zip archive of your project directory.

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