PROJECT 1: TALK PROJECT

A simple unidirectional talk program consisting of two Java classes (TalkClient and TalkServer) was presented in class. For this project you will have to implement a bidirectional Talk program consisting of only one Java class (Talk.java) as specified below. Only two-party conversations are to be supported by your program.

Description

Your program should accept command-line options and run in any of the following modes (and report an error for **any** invalid invocation):

Talk –h [*hostname* | *IPaddress*] [–p *portnumber*]

The program behaves as a client connecting to [*hostname* | *IPaddress*] on port *portnumber*. If a server is not available your program should exit with the message "Client unable to communicate with server". Note: *portnumber* in this case refers to the server and not to the client.

Talk –s [–p *portnumber*]

The program behaves as a server listening for connections on port *portnumber*. If the port is not available for use, your program should exit with the message "Server unable to listen on specified port".

Talk –a [*hostname*|*IPaddress*] [–p *portnumber*]

The program enters ``auto'' mode. When in auto mode, your program should start as a client attempting to communicate with *hostname*|*IPaddress* on port *portnumber*. If a server is not found, your program should detect this condition and start behaving as a server listening for connections on port *portnumber*.

Talk –help

The program prints your name and instructions on how to use your program.

Once a two-party connection has been established, messages received through the socket should be prepended with ``[remote]'' when displaying the message on the screen to differentiate it from messages that are typed locally. In addition, the string STATUS is considered to be a keyword and will not be transmitted. If a user types STATUS your program should print information about the state of the connection (IP numbers, remote and local ports).

Arguments in brackets are optional. In case a *hostname* or *IPaddress* is not provided their default value should be that of the computer where Talk is being executed. If a *portnumber* is not provided its default value should be 12987. Once a connection is established you will have to obtain suitable stream objects such as a BufferedReader and a PrintWriter (from the socket) as your input and output streams respectively. The unit of data sent/received must be a line of text as returned by the readLine() method. Users of your program will be using the keyboard to type their messages and their screens to display incoming and outgoing messages.

Note: the readLine() method is a blocking operation and the call does not return until something is read from the input stream. This may result in undesired effects on your program such as not being able to

receive and display remote messages on your screen while you are typing. Your program must not suffer from this problem and your solution should be described in your report.

Implementation
The program must be implemented in Java and your submission should consist of one file (Talk.java). I ask that packages not be used (I will compile every single submission using scripts and I cannot duplicate your directory structure in a consistent manner). Note that the use of inner classes is allowed.

Report
A brief report should be typed and turned in by the due date at class time. The report does not need to be long and should only describe your approach (and challenges) in solving the problem and any methods or variables of relevance in your class. The report should also include a ``feedback'' section about this project.

Submission
The report and electronic submissions are due at class time on the due date.

Submissions will be done via Harvey (Harvey.utulsa.edu). If your name is not in the class homepage (at Harvey), please send me your name, your student ID number, the reason why you are not enrolled in the class and I'll make sure it is added to our website. Your submission should consist of a zip file containing your report and your Talk.java program(s).

DUE DATE: Friday, Sept. 20, 2013 at 11:00am