Master of Software Systems Program

CICS 525 | Real-Time and Distributed Systems | May – July 2011

Assignment #1

General guidelines for homework: Before starting on this homework, review the homework guidelines provided on the first day of class. Remember that it is encouraged to discuss the problems with others in the class, but the assignment group only should complete all write-ups. **Homework grades will be based not only on getting the "correct answer," but also on good writing style and clear presentation of your solution**. It is your responsibility to make sure that the graders can easily follow your line of reasoning. Even if you can't solve the problem, you will receive partial credit for explaining why you got stuck on a promising line of attack. More importantly, you will get valuable feedback that will help you learn the material. Please acknowledge the people with whom you discussed the problems and what sources you used to help you solve the problem (e.g., books from the library).

Part 1: Java Sockets

Your teaching assistant will introduce you to the notion of sockets in Java and you will study the application of sockets in the development of client-server applications. The first client-server application that you will be introduced to will be the Echo Client-Server. On the course web page you will find two Java source files: EchoServer.java and EchoClient.java (see Reference 5 for details). The following information has been extracted from Reference 5.

Java Sockets are a mechanism for communication over the Internet. All the classes discussed in this lab are in the java.net package. A socket is an endpoint for communication. There are two kinds of socket, depending on whether one wishes to use a connectionless or a connection-oriented protocol. The connectionless communication protocol of the Internet is called UDP. The connection-oriented communication protocol of the Internet is called TCP. UDP sockets are also called datagram sockets.

Internet Addresses and Ports

Each socket is uniquely identified on the entire Internet with two numbers. The first number is a 32-bit integer called the Internet Address (or IP address). The second number is a 16-bit integer called the port of the socket. The IP address uniquely identifies one machine (also called a host or node) on the Internet. The new version of the Internet protocol (IP version 6 or IPv6) increases the size of the IP address to 128 bits. Java encapsulates the concept of an IP address with the class InetAddress. Java represents ports with 32-bit integers, even though a 16-bit integer would suffice.

While IP addresses uniquely identify machines, the reverse is not true. A machine can have several IP addresses. Note that each 16-bit port number actually represents two distinct ports: a UDP port and a TCP port.

Connectionless Sockets

The simplest kind of socket is a UDP socket. Such a socket is analogous to a mailbox. Data is sent and received in units called datagrams (analogous to letters and parcels) from any UDP socket to any other UDP socket. For this reason UDP sockets are also called datagram sockets. Java encapsulates the concept of a UDP socket with the class DatagramSocket, and the concept of a datagram with the class DatagramPacket.

A DatagramPacket consists of a fixed-length array of bytes together with an IP address and port. When one sends a DatagramPacket, its array of bytes is sent to the socket with the specified IP address and port (if it exists). When one receives a DatagramPacket, the data is copied into its array of bytes, and the IP address and port of the sender are copied into the IP address and port of the DatagramPacket.

Connection-Oriented Sockets

A TCP socket is analogous to (one side of) a telephone connection. TCP sockets are of two kinds: ordinary sockets and server sockets. A server socket is never used for transmission of information. Its sole purpose is to listen for incoming connection requests. When a client process wishes to make a connection with a server, it first constructs an ordinary socket and then it asks for a connection with the server. When a server socket receives a connection request, it constructs an ordinary socket with an unused port number which completes the connection. The server socket then goes back to listening for connections. Once the connection is established, the two connected sockets can communicate with each other using ordinary read and write operations in either direction.

Java encapsulates the concept of an ordinary TCP socket with the class Socket, and the concept of server socket with the class ServerSocket. Input to and output from a socket is encapsulated in Java using the InputStream and OutputStream classes, respectively.

Connections to Web Servers

One special case of a TCP socket is a socket that is communicating with a Web server. Web servers normally listen on TCP port 80. One could construct a socket and connect directly to a Web server if one knows its IP address. However, when one connects to a Web server, it is usually because one is interested in a document. Documents on the Internet are identified in a very different manner than one uses for identifying sockets on the Internet. The standard identifier for a document on the Internet is its Universal Resource Locator (URL). Java encapsulates the concept of a URL with the class URL. When one constructs a URL object, the URL is checked to make sure that it is a well formed URL. Interacting with the URL requires that one first establish a connection with the Web server that is responsible for the document identified by the URL. The TCP socket for the connection is constructed by invoking the openConnection method on the URL object. This method also performs the name resolution necessary to determine the IP address of the Web server. This method returns an object of type URLConnection. The connection to the Web server is requested by calling connect on the URLConnection object. Input to (when defined) and output from the document identified by the URL is encapsulated in Java using the InputStream and OutputStream classes, respectively. If the URL specifies the http protocol, then the URLConnection will actually be an object of the subclass HttpURLConnection which has additional methods specific to HTML documents. Other protocols are also supported, but only http has a public subtype associated with it. "

Part 2: A Client-Server Application - Simple Stock Trading System

You must implement a stock trading client-server application in Java which uses Java TCP sockets. The application involves two clients and one server (no need for threads).

Requirements

Here is a description of the functionality of your application:

1. The server keeps track of stock prices according to stock names. It returns stock prices when queried with the name of the stock.
2. One client can update the price of individual stocks.
3. The other client queries stock prices by passing the stock name.

The application could be based on the Sun Java Knock Knock Joke Socket tutorial.

Part 3: A Multithreaded Client-Server Application for Stock Trading

You must implement a stock trading client-server application in Java which uses Java TCP sockets. The application involves two clients and one server. This is a multithreaded implementation and requires synchronization between threads for different operations.

Requirements

Here is a description of the functionality of your application:

1. The server keeps track of stock prices according to stock names. It returns stock prices when queried with the name of the stock. The server is assumed to be a non-terminating application.
2. The server also retains information about the stock owned by individual users and their cash balance.
3. At the server, stock prices are regularly updated for those stock tickers that clients have expressed an interest in. Initially the server is not tracking any stock. When the first client connects and requests a quote, the server begins to track that stock, and other stocks from then onward. The server retrieves stock prices using the Google Finance API (the Teaching Assistants will give you the details). The prices are updated periodically every two minutes. For every new stock that the server tracks, it starts with 1000 shares to trade. Clients may purchase from these 1000 shares and no more.
4. The server makes all data persistent: when the server is terminated all relevant data is stored to disk and the server restores the data when restarted.
5. Each client is associated with a user. When a client connects to the server the first message from the client to the server is the username, which is indicated by the string USER <USERNAME>. If the user is new, then $1000 is credited to the user and the username is added to the server's data set.
6. A client can query stock prices by passing the stock name. The server returns the price of the stock if the ticker symbol is valid. A query is placed using the string QUERY <TICKERNAME> where <TICKERNAME> is a stock identifier.

7. Clients can also buy or sell stocks with the message BUY <TICKERNAME> <NUM STOCKS> or SELL <TICKERNAME> <NUM STOCKS>. The server should verify that the user has a sufficient balance for a BUY transaction and that the user owns stock for the SELL transaction. The transactions alter the user's cash balance appropriately.

The application could be based on Sun's Multithreaded Knock Knock Joke Socket tutorial. Your assignment must be accompanied by a text document which describes how you came up with your implementation, what are the limitations of your program and how can you extend it in the near future.

Submission

You must submit your assignment using `handin`. The name of the course is `cics525` and the assignment name is `assignment1`. You only need to submit your work for Part 3 although completing Part 2 will help in your understanding the content better.

You must submit the following files (please use these names for your files):

1. Source code: queryClient.java, updaterClient.java, server.java
2. A makefile with compilation instructions: Makefile
3. A readme file in plain ASCII text format: Readme.txt
4. The readme file must contain the following information:

    Lab number

    Names and student numbers

    Operating system

    Java compiler (version)

    Compilation instructions

    Execution instructions

    Known bugs

Recommended References

1. Java Sun, All About Sockets - http://java.sun.com/docs/books/tutorial/networking/sockets/
2. Java Socket Tutorial - http://www.ccs.neu.edu/home/kenb/com1335/socket_tut.html
3. Java Sun, Knock Knock Joke Client-Server Tutorial - http://java.sun.com/docs/books/tutorial/networking/sockets/clientServer.html
4. Sockets: Basic Client-Server Programming in Java (Rick Proctor) http://bdn.borland.com/article/0,1410,31995,00.html
5. Java TCP Sockets - http://www.csc.villanova.edu/mdamian/Sockets/JavaSocketsNoThread.htm