Assignment 2J – TimeServer

1. **Write a server called TimeServer.java to serve the time of day to any client that wishes to connect.**

When a client connects, enter a loop that sends the time to the client every second. The message should be formatted like this (with a semi-colon at the end to denote the end of the message):

May 5, 2008 8:03:00 AM;

After writing the message to the OutputStream, use .flush( ) on the OutputStream to force the output to actually get sent over the network. Leave all the I/O objects and the Socket open, wait a second and send the time again (and continue repeating). Note: when you leave the OutputStream open, data written to it will remain in server-side buffers until you .close() the OutputStream or .flush() on the OutputStream. Flushing keeps it open while forcing the data out of buffers and across the wire.

1. **Write a client to connect to TimeServer and get the time. Call it TimeClient.java.**

Connect to the server and read the message from it. If you try to read, and no data is available yet, the read will block (meaning it will stop and wait; your program will be suspended, waiting). When data becomes available, the read will un-block and read the data.

Read data up through the semi-colon (that’s part of the protocol, so you can expect it), print the data to the screen (on a new line). Then read again (which will wait for more data).

# Some APIs for date/time in Java

**Date** class (in the java.util library):

Constructor with no parameters will create a Date object that represents the current date and time.

.**toString**() will return a human readable representation of the date in the form ‘Fri May 10 11:42:00 CDT 2013’

**DateFormat** class (in the java.text library):

.**getDateTimeInstance**() – a static method that will return a DateFormat object that will format a Date object into a human readable string.

.**format**(Date d)- will return a human readable string for the date in the form ‘May 10, 2013 11:42:00 AM’