```java
import java.io.*;
import java.net.*;
import java.util.*;

/** Trivial Chat Server to go with our Trivial Chat Client.
 *
 * WARNING -- this code is believed thread-safe but has NOT been 100% vetted
 * by a team of world-class experts for Thread-safeness.
 * DO NOT BUILD ANYTHING CRITICAL BASED ON THIS until you have done so.
 * See the various books on Threaded Java for design issues.
 *
 * @author      Ian F. Darwin, http://www.darwinsys.com/
 * @version $Id: ChatServer.java,v 1.3 2006/04/11 22:40:33 ian Exp $
 */
public class ChatServer {
        /** What I call myself in system messages */
        protected final static String CHATMASTER_ID = "ChatMaster";
        /** What goes between any handle and the message */
        protected final static String SEP = ": ";
        /** The Server Socket */
        protected ServerSocket servSock;
        /** The list of my current clients */
        protected ArrayList<ChatHandler> clients;
        /** Debugging state */
        private static boolean DEBUG = false;

        /** Main just constructs a ChatServer, which should never return */
        public static void main(String[] argv) throws IOException {
                System.out.println("DarwinSys Chat Server 0.1 starting...");
                if (argv.length == 1 && argv[0].equals("-debug"))
                        DEBUG = true;
                ChatServer w = new ChatServer();
                w.runServer();                  // should never return.
                System.out.println("**ERROR* Chat Server 0.1 quitting");
        }

        /** Construct (and run!) a Chat Service
         * @throws IOException
         */
        ChatServer() throws IOException {
                clients = new ArrayList<ChatHandler>();

                servSock = new ServerSocket(Chat.PORTNUM);
                System.out.println("DarwinSys Chat Server Listening on port " +
                                Chat.PORTNUM);
        }

        public void runServer() {
                try {
                        while (true) {
                                Socket us = servSock.accept();
                                String hostName = us.getInetAddress().getHostName();
                                System.out.println("Accepted from " + hostName);
                                ChatHandler cl = new ChatHandler(us, hostName);
                                synchronized (clients) {
                                        clients.add(cl);
                                        cl.start();
                                        if (clients.size() == 1)
                                                cl.send(CHATMASTER_ID, "Welcome! you’re the first one here");
                                        else {
                                                cl.send(CHATMASTER_ID, "Welcome! you’re the latest of " +
                                                        clients.size() + " users.");
                                        }
                                }
                        }
                } catch(IOException e) {
                        log("IO Exception in runServer: " + e);
                }
        }

        protected void log(String s) {
                System.out.println(s);
        }

        /** Inner class to handle one conversation */
        protected class ChatHandler extends Thread {
                /** The client socket */
                protected Socket clientSock;
                /** BufferedReader for reading from socket */
                protected BufferedReader is;
                /** PrintWriter for sending lines on socket */
                protected PrintWriter pw;
                /** The client’s host */
                protected String clientIP;
                /** String form of user’s handle (name) */
                protected String login;

                /* Construct a Chat Handler */
```

```
		public ChatHandler(Socket sock, String clnt) throws IOException {
			clientSock = sock;
			clientIP = clnt;
			is = new BufferedReader(
				new InputStreamReader(sock.getInputStream()));
			pw = new PrintWriter(sock.getOutputStream(), true);
		}

		/** Each ChatHandler is a Thread, so here's the run() method,
		 * which handles this conversation.
		 */
		public void run() {
			String line;
			try {
				while ((line = is.readLine()) != null) {
					char c = line.charAt(0);
					line = line.substring(1);
					switch (c) {
					case Chat.CMD_LOGIN:
						if (!Chat.isValidLoginName(line)) {
							send(CHATMASTER_ID, "LOGIN " + line + " invalid");
							log("LOGIN INVALID from " + clientIP);
							continue;
						}
						login = line;
						broadcast(CHATMASTER_ID, login +
							" joins us, for a total of " +
							clients.size() + " users");
						break;
					case Chat.CMD_MESG:
						if (login == null) {
							send(CHATMASTER_ID, "please login first");
							continue;
						}
						int where = line.indexOf(Chat.SEPARATOR);
						String recip = line.substring(0, where);
						String mesg = line.substring(where+1);
						log("MESG: " + login + "-->" + recip + ": "+ mesg);
						ChatHandler cl = lookup(recip);
						if (cl == null)
							psend(CHATMASTER_ID, recip + " not logged in.");
						else
							cl.psend(login, mesg);
						break;
```

```
                                case Chat.CMD_QUIT:
                                        broadcast(CHATMASTER_ID,
                                                "Goodbye to " + login + "@" + clientIP);
                                        close();
                                        return;         // The end of this ChatHandler

                                case Chat.CMD_BCAST:
                                        if (login != null)
                                                broadcast(login, line);
                                        else
                                                log("B<L FROM " + clientIP);
                                        break;
                                default:
                                        log("Unknown cmd " + c + " from " + login + "@" + clientIP);
                                }
                        }
                } catch (IOException e) {
                        log("IO Exception: " + e);
                } finally {
                        // the sock ended, so we’re done, bye now
                        System.out.println(login + SEP + "All Done");
                        synchronized(clients) {
                                clients.remove(this);
                                if (clients.size() == 0) {
                                        System.out.println(CHATMASTER_ID + SEP +
                                                "I’m so lonely I could cry...");
                                } else if (clients.size() == 1) {
                                        ChatHandler last = (ChatHandler)clients.get(0);
                                        last.send(CHATMASTER_ID,
                                                "Hey, you’re talking to yourself again");
                                } else {
                                        broadcast(CHATMASTER_ID,
                                                "There are now " + clients.size() + " users");
                                }
                        }
                }
        }

        protected void close() {
                if (clientSock == null) {
                        log("close when not open");
                        return;
                }
                try {
```

```
                                clientSock.close();
                                clientSock = null;
                        } catch (IOException e) {
                                log("Failure during close to " + clientIP);
                        }
                }

                /** Send one message to this user */
                public void send(String sender, String mesg) {
                        pw.println(sender + SEP + mesg);
                }

                /** Send a private message */
                protected void psend(String sender, String msg) {
                        send("<*" + sender + "*>", msg);
                }

                /** Send one message to all users */
                public void broadcast(String sender, String mesg) {
                        System.out.println("Broadcasting " + sender + SEP + mesg);
                        for (int i=0; i<clients.size(); i++) {
                                ChatHandler sib = (ChatHandler)clients.get(i);
                                if (DEBUG)
                                        System.out.println("Sending to " + sib);
                                sib.send(sender, mesg);
                        }
                        if (DEBUG) System.out.println("Done broadcast");
                }

                protected ChatHandler lookup(String nick) {
                        synchronized(clients) {
                                for (int i=0; i<clients.size(); i++) {
                                        ChatHandler cl = (ChatHandler)clients.get(i);
                                        if (cl.login.equals(nick))
                                                return cl;
                                }
                        }
                        return null;
                }

                /** Present this ChatHandler as a String */
                public String toString() {
                        return "ChatHandler[" + login + "]";
                }
```

```
        }
}
```