```
 1: // $Id: multiserver.java,v 1.4 2012-05-29 20:54:57-07 - - $
 2:
 3: //
 4: // Multi server.  Usage:  java multiserver hostport.  Accept a
 5: // connection from many client and echo back any input.  Each
 6: // client is assigned a worker thread.
 7:
 8: //
 9:
10: import java.io.*;
11: import java.net.*;
12: import java.util.*;
13: import static java.lang.System.*;
14:
15: class multiserver {
16:
17:    static void quit (String format, Object... params) {
18:       err.printf (format, params);
19:       exit (1);
20:    }
21:
22:    static String ident (options opts) {
23:       return String.format ("%s: port %d", opts.progname,
24:                             opts.portnumber);
25:    }
26:
27:    static String get_jarname() {
28:       String jarpath = getProperty ("java.class.path");
29:       int lastslash = jarpath.lastIndexOf ('/');
30:       if (lastslash < 0) return jarpath;
31:       return jarpath.substring (lastslash + 1);
32:    }
33:
34:    static class options {
35:       final String progname = get_jarname();
36:       int portnumber;
37:       options (String[] args) {
38:          try {
39:             if (args.length != 1) throw new NumberFormatException();
40:             portnumber = Integer.parseInt (args[0]);
41:          }catch (NumberFormatException exn) {
42:             quit ("Usage: %s portnumber%n", progname);
43:          }
44:       }
45:    }
46:
```

```
47:
48:    static class worker implements Runnable {
49:       static int worker_count = 0;
50:       options opts;
51:       Socket client;
52:       int worker_id = ++worker_count;
53:       worker (options opts, Socket client) {
54:          this.opts = opts;
55:          this.client = client;
56:       }
57:       public void run() {
58:          out.printf ("%s: worker %d: starting%n",
59:                      ident (opts), worker_id);
60:          try {
61:             Scanner client_in = new Scanner (client.getInputStream());
62:             PrintWriter client_out =
63:                         new PrintWriter (client.getOutputStream());
64:             for (int count = 1; client_in.hasNextLine(); ++count) {
65:                if (client.isInputShutdown()
66:                 || client.isOutputShutdown()) break;
67:                String line = client_in.nextLine();
68:                out.printf ("%d[%d]%s%n", worker_id, count, line);
69:                client_out.printf ("%d[%d]%s%n", worker_id, count, line);
70:                client_out.flush();
71:             }
72:             client.close();
73:             out.printf ("%s: worker %d: finished%n",
74:                         ident (opts), worker_id);
75:          }catch (IOException exn) {
76:             quit ("%s: %s%n", ident (opts), exn);
77:          }
78:       }
79:    }
80:
81:    public static void main (String[] args) {
82:       options opts = new options (args);
83:       try {
84:          ServerSocket socket = new ServerSocket (opts.portnumber);
85:          out.printf ("%s: waiting for client%n", ident (opts));
86:          for (;;) {
87:             Socket client = socket.accept();
88:             out.printf ("%s: socket.accept OK%n", ident (opts));
89:             Thread worker = new Thread (new worker (opts, client));
90:             worker.start();
91:          }
92:       }catch (IOException exn) {
93:          quit ("%s: %s%n", ident (opts), exn);
94:       }catch (IllegalArgumentException exn) {
95:          quit ("%s: %s%n", ident (opts), exn);
96:       }
97:    }
98:
99: }
```