```
 1: // $Id: miniserver.java,v 1.5 2013-08-13 20:18:57-07 - - $
 2:
 3: //
 4: // Mini server.
 5: //    Usage:  miniserver hostport
 6: // Accept a connection from a client and echo back any input.
 7: //
 8:
 9: import java.io.*;
10: import java.net.*;
11: import java.util.*;
12: import static java.lang.System.*;
13:
14: class miniserver {
15:
16:    static void quit (String format, Object... params) {
17:       err.printf (format, params);
18:       exit (1);
19:    }
20:
21:    static String ident (options opts) {
22:       return String.format ("%s: port %d", opts.jarname,
23:                             opts.portnumber);
24:    }
25:
26:    static String get_jarname() {
27:       String jarpath = getProperty ("java.class.path");
28:       int lastslash = jarpath.lastIndexOf ('/');
29:       if (lastslash < 0) return jarpath;
30:       return jarpath.substring (lastslash + 1);
31:    }
32:
33:    static class options {
34:       final String jarname = get_jarname();
35:       int portnumber;
36:       options (String[] args) {
37:          try {
38:             if (args.length != 1) throw new NumberFormatException();
39:             portnumber = Integer.parseInt (args[0]);
40:          }catch (NumberFormatException exn) {
41:             quit ("Usage: %s portnumber%n", jarname);
42:          }
43:       }
44:    }
45:
```

```
46:
47:
48:    public static void main (String[] args) {
49:       options opts = new options (args);
50:       try {
51:          ServerSocket socket = new ServerSocket (opts.portnumber);
52:          out.printf ("%s: waiting for client%n", ident (opts));
53:          Socket client = socket.accept();
54:          out.printf ("%s: socket.accept OK%n", ident (opts));
55:          Scanner client_in = new Scanner (client.getInputStream());
56:          PrintWriter client_out =
57:                      new PrintWriter (client.getOutputStream());
58:          for (int count = 1; client_in.hasNextLine(); ++count) {
59:             String line = client_in.nextLine();
60:                    out.printf ("Client sent: [%d]%s%n", count, line);
61:             client_out.printf ("Client sent: [%d]%s%n", count, line);
62:             client_out.flush();
63:          }
64:          socket.close();
65:          client.close();
66:          out.printf ("%s: finished%n", ident (opts));
67:       }catch (IOException exn) {
68:          quit ("%s: %s%n", ident (opts), exn);
69:       }catch (IllegalArgumentException exn) {
70:          quit ("%s: %s%n", ident (opts), exn);
71:       }
72:    }
73:
74: }
```

```
 1: // $Id: miniclient.java,v 1.8 2013-08-13 20:22:20-07 - - $
 2:
 3: //
 4: // Mini client.
 5: //    Usage:  miniclient hostname hostport
 6: // Reads stdin and copies to the port.  Then copies the answer
 7: // back from the port to stdout.  Runs only in lock step with
 8: // strict alternation for the server.
 9: //
10:
11: import java.io.*;
12: import java.net.*;
13: import java.util.*;
14: import static java.lang.System.*;
15:
16: class miniclient {
17:
18:    static void quit (String format, Object... params) {
19:       err.printf (format, params);
20:       exit (1);
21:    }
22:
23:    static String ident (options opts) {
24:       return String.format ("%s: %s %d", opts.jarname,
25:                             opts.hostname, opts.portnumber);
26:    }
27:
28:    static String get_jarname() {
29:       String jarpath = getProperty ("java.class.path");
30:       int lastslash = jarpath.lastIndexOf ('/');
31:       if (lastslash < 0) return jarpath;
32:       return jarpath.substring (lastslash + 1);
33:    }
34:
35:    static class options {
36:       final String jarname = get_jarname();
37:       String hostname = "localhost";
38:       int portnumber;
39:       options (String[] args) {
40:          try {
41:             if (args.length < 1) throw new NumberFormatException ();
42:             portnumber = Integer.parseInt (args[0]);
43:             if (args.length > 1) hostname = args[1];
44:          }catch (NumberFormatException exn) {
45:             quit ("Usage: %s hostname portnumber%n", jarname);
46:          }
47:       }
48:    }
49:
```

```
50:
51:    public static void main (String[] args) {
52:       Scanner stdin = new Scanner (System.in);
53:       options opts = new options (args);
54:       try {
55:          Socket socket = new Socket (opts.hostname, opts.portnumber);
56:          out.printf ("%s: socket OK%n", ident (opts));
57:          Scanner serve_in = new Scanner (socket.getInputStream ());
58:          PrintWriter serve_out =
59:                      new PrintWriter (socket.getOutputStream ());
60:          for(;;) {
61:             out.printf ("%s: ", opts.jarname);
62:             if (! stdin.hasNextLine ()) break;
63:             String line = stdin.nextLine();
64:                   out.printf ("Stdin read: %s%n", line);
65:             serve_out.printf ("Stdin read: %s%n", line);
66:             serve_out.flush ();
67:             if (serve_in.hasNextLine ()) {
68:                out.printf ("Server said: %s%n", serve_in.nextLine ());
69:             }else {
70:                quit ("%s: no reply%n", ident (opts));
71:             }
72:          }
73:          socket.close ();
74:          out.printf ("%s: finished%n", ident (opts));
75:       }catch (IOException exn) {
76:          quit ("%s: %s%n", ident (opts), exn.getMessage());
77:       }catch (IllegalArgumentException exn) {
78:          quit ("%s: %s%n", ident (opts), exn.getMessage());
79:       }
80:    }
81:
82: }
83:
```

```
 1: ::::::::::::::
 2: miniserver.log
 3: ::::::::::::::
 4: bash-2$ miniserver 8888
 5: miniserver: port 8888: waiting for client
 6: miniserver: port 8888: socket.accept OK
 7: Client sent: [1]Stdin read: This is the first message.
 8: Client sent: [2]Stdin read: Second message.
 9: Client sent: [3]Stdin read: THird message.
10: miniserver: port 8888: finished
11: bash-3$ exit
12: exit
13:
14: ::::::::::::::
15: miniclient.log
16: ::::::::::::::
17: bash-2$ miniclient 8888
18: miniclient: localhost 8888: socket OK
19: miniclient: This is the first message.
20: Stdin read: This is the first message.
21: Server said: Client sent: [1]Stdin read: This is the first message.
22: miniclient: Second message.
23: Stdin read: Second message.
24: Server said: Client sent: [2]Stdin read: Second message.
25: miniclient: THird message.
26: Stdin read: THird message.
27: Server said: Client sent: [3]Stdin read: THird message.
28: miniclient: miniclient: localhost 8888: finished
29: bash-3$ exit
30: exit
31:
```