

```
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: }
```

```
1: // $Id: roboclient.java,v 1.8 2013-08-13 20:36:49-07 - - $
2:
3: //
4: // Roboclient hostname port username delaysec cycles message message...
5: //
6: // The roboclient connects to hostname:port with the username.
7: // Then it writes messages given by the trailing words in args
8: // the number of cycles followed by a certain delay in seconds.
9: // Then it quits. Useful for testing the server.
10: //
11:
12: import java.io.*;
13: import java.net.*;
14: import java.util.*;
15: import static java.lang.System.*;
16:
17: class roboclient {
18:
19:     static void quit (String format, Object... params) {
20:         err.printf (format, params);
21:         exit (1);
22:     }
23:
24:     static String ident (options opts) {
25:         return String.format ("%s: %s %d", opts.progname,
26:                                opts.hostname, opts.portnumber);
27:     }
28:
29:     static class options {
30:         final String progname = "roboclient";
31:         String hostname;
32:         int portnumber;
33:         String username;
34:         long delaysec;
35:         int cycles;
36:         String robomessage;
37:         options (String[] args) {
38:             try {
39:                 if (args.length < 5) throw new NumberFormatException();
40:                 hostname = args[0];
41:                 portnumber = Integer.parseInt (args[1]);
42:                 username = args[2];
43:                 delaysec = Long.parseLong (args[3]);
44:                 cycles = Integer.parseInt (args[4]);
45:                 robomessage = "";
46:                 for (int i = 5; i < args.length; ++i) {
47:                     robomessage += " " + args[i];
48:                 }
49:             } catch (NumberFormatException exn) {
50:                 quit ("Usage: %s hostname port username delaysec cycles "
51:                     + "message message%n", progname);
52:             }
53:         }
54:     }
55:
```

```
56:
57: static class reader implements Runnable {
58:     Socket socket;
59:     Scanner scanner;
60:     reader (Socket _socket, Scanner _scanner) {
61:         scanner = _scanner;
62:         socket = _socket;
63:     }
64:     public void run() {
65:         while (! socket.isInputShutdown() && scanner.hasNextLine()) {
66:             out.printf ("%s\n", scanner.nextLine());
67:         }
68:         scanner.close();
69:     }
70: }
71:
72: static class writer implements Runnable {
73:     Socket socket;
74:     options opts;
75:     PrintWriter writer;
76:     writer (Socket _socket, options _opts, PrintWriter _writer) {
77:         socket = _socket;
78:         opts = _opts;
79:         writer = _writer;
80:     }
81:     public void run() {
82:         writer.printf ("%s\n", opts.username);
83:         writer.flush();
84:         for (int count = 0; count < opts.cycles; ++count) {
85:             if (socket.isOutputShutdown()) break;
86:             try {
87:                 Thread.sleep (opts.delaysec * 1000);
88:             } catch (InterruptedException error) {
89:             }
90:             writer.printf ("%s[%d] %s (%s %d) --%s\n",
91:                             opts.progname, count, opts.username,
92:                             opts.hostname, opts.portnumber,
93:                             opts.robomessage);
94:             writer.flush();
95:         }
96:         writer.close();
97:     }
98: }
99:
100: public static void main (String[] args) {
101:     Scanner stdin = new Scanner (System.in);
102:     options opts = new options (args);
103:     try {
104:         Socket socket = new Socket (opts.hostname, opts.portnumber);
105:         out.printf ("%s: socket OK\n", ident (opts));
106:         Thread reading = new Thread (new reader (socket,
107:             new Scanner (socket.getInputStream())));
108:         Thread writing = new Thread (new writer (socket, opts,
109:             new PrintWriter (socket.getOutputStream())));
110:         reading.start();
111:         writing.start();
112:     } catch (IOException exn) {
113:         quit ("%s: %s\n", ident (opts), exn);
114:     } catch (IllegalArgumentException exn) {
115:         quit ("%s: %s\n", ident (opts), exn);
116:     }
```

```
117:    }  
118:  
119: }  
120:
```

```
1: ::::::::::::::
2: roboserver.log
3: ::::::::::::::
4: bash-2$ multiserer 8888
5: multiserer: port 8888: waiting for client
6: multiserer: port 8888: socket.accept OK
7: multiserer: port 8888: worker 1: starting
8: 1[1]Hello
9: 1[2]roboclient[0] Hello (localhost 8888) -- Message from World.
10: 1[3]roboclient[1] Hello (localhost 8888) -- Message from World.
11: 1[4]roboclient[2] Hello (localhost 8888) -- Message from World.
12: 1[5]roboclient[3] Hello (localhost 8888) -- Message from World.
13: multiserer: port 8888: socket.accept OK
14: multiserer: port 8888: worker 2: starting
15: 2[1]Foo-Bar
16: 2[2]roboclient[0] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
17: 1[6]roboclient[4] Hello (localhost 8888) -- Message from World.
18: 2[3]roboclient[1] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
19: 2[4]roboclient[2] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
20: 1[7]roboclient[5] Hello (localhost 8888) -- Message from World.
21: 2[5]roboclient[3] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
22: 2[6]roboclient[4] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
23: 1[8]roboclient[6] Hello (localhost 8888) -- Message from World.
24: 2[7]roboclient[5] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
25: 2[8]roboclient[6] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
26: 1[9]roboclient[7] Hello (localhost 8888) -- Message from World.
27: 2[9]roboclient[7] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
28: 2[10]roboclient[8] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
29: 1[10]roboclient[8] Hello (localhost 8888) -- Message from World.
30: 2[11]roboclient[9] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
31: multiserer: port 8888: worker 2: finished
32: 1[11]roboclient[9] Hello (localhost 8888) -- Message from World.
33: multiserer: port 8888: worker 1: finished
34: ^C
35: bash-3$ exit
36: exit
37:
38: ::::::::::::::
39: roboclient1.log
40: ::::::::::::::
41: bash-2$ roboclient localhost 8888 Hello 2 10 Message from World.
42: roboclient: localhost 8888: socket OK
43: 1[1]Hello
44: 1[2]roboclient[0] Hello (localhost 8888) -- Message from World.
45: 1[3]roboclient[1] Hello (localhost 8888) -- Message from World.
46: 1[4]roboclient[2] Hello (localhost 8888) -- Message from World.
47: 1[5]roboclient[3] Hello (localhost 8888) -- Message from World.
48: 1[6]roboclient[4] Hello (localhost 8888) -- Message from World.
49: 1[7]roboclient[5] Hello (localhost 8888) -- Message from World.
50: 1[8]roboclient[6] Hello (localhost 8888) -- Message from World.
51: 1[9]roboclient[7] Hello (localhost 8888) -- Message from World.
52: 1[10]roboclient[8] Hello (localhost 8888) -- Message from World.
53: bash-3$ exit
54: exit
55:
56: ::::::::::::::
57: roboclient2.log
58: ::::::::::::::
59: bash-2$ roboclient localhost 8888 Foo-Bar 1 10 Message from Foo Bar Baz Qux.
60: roboclient: localhost 8888: socket OK
61: 2[1]Foo-Bar
```

```
62: 2[2]roboclient[0] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
63: 2[3]roboclient[1] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
64: 2[4]roboclient[2] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
65: 2[5]roboclient[3] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
66: 2[6]roboclient[4] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
67: 2[7]roboclient[5] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
68: 2[8]roboclient[6] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
69: 2[9]roboclient[7] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
70: 2[10]roboclient[8] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
71: bash-3$ exit
72: exit
73:
```