```java
// $Id: multiserver.java,v 1.4 2012-05-29 20:54:57-07 - - $

//
// Multi server.  Usage:  java multiserver hostport.  Accept a
// connection from many client and echo back any input.  Each
// client is assigned a worker thread.

//

import java.io.*;
import java.net.*;
import java.util.*;
import static java.lang.System.*;

class multiserver {

   static void quit (String format, Object... params) {
      err.printf (format, params);
      exit (1);
   }

   static String ident (options opts) {
      return String.format ("%s: port %d", opts.progname,
                            opts.portnumber);
   }

   static String get_jarname() {
      String jarpath = getProperty ("java.class.path");
      int lastslash = jarpath.lastIndexOf ('/');
      if (lastslash < 0) return jarpath;
      return jarpath.substring (lastslash + 1);
   }

   static class options {
      final String progname = get_jarname();
      int portnumber;
      options (String[] args) {
         try {
            if (args.length != 1) throw new NumberFormatException();
            portnumber = Integer.parseInt (args[0]);
         }catch (NumberFormatException exn) {
            quit ("Usage: %s portnumber%n", progname);
         }
      }
   }


   static class worker implements Runnable {
      static int worker_count = 0;
      options opts;
      Socket client;
      int worker_id = ++worker_count;
      worker (options opts, Socket client) {
         this.opts = opts;
         this.client = client;
      }
      public void run() {
         out.printf ("%s: worker %d: starting%n",
                     ident (opts), worker_id);
         try {
            Scanner client_in = new Scanner (client.getInputStream());
            PrintWriter client_out =
                        new PrintWriter (client.getOutputStream());
            for (int count = 1; client_in.hasNextLine(); ++count) {
               if (client.isInputShutdown()
                || client.isOutputShutdown()) break;
               String line = client_in.nextLine();
               out.printf ("%d[%d]%s%n", worker_id, count, line);
               client_out.printf ("%d[%d]%s%n", worker_id, count, line);
               client_out.flush();
            }
            client.close();
            out.printf ("%s: worker %d: finished%n",
                        ident (opts), worker_id);
         }catch (IOException exn) {
            quit ("%s: %s%n", ident (opts), exn);
         }
      }
   }

   public static void main (String[] args) {
      options opts = new options (args);
      try {
         ServerSocket socket = new ServerSocket (opts.portnumber);
         out.printf ("%s: waiting for client%n", ident (opts));
         for (;;) {
            Socket client = socket.accept();
            out.printf ("%s: socket.accept OK%n", ident (opts));
            Thread worker = new Thread (new worker (opts, client));
            worker.start();
         }
      }catch (IOException exn) {
         quit ("%s: %s%n", ident (opts), exn);
      }catch (IllegalArgumentException exn) {
         quit ("%s: %s%n", ident (opts), exn);
      }
   }

}
```

```java
// $Id: roboclient.java,v 1.8 2013-08-13 20:36:49-07 - - $

//
// Roboclient hostname port username delaysec cycles message message...
//
// The roboclient connects to hostname:port with the username.
// Then it writes messages given by the trailing words in args
// the number of cycles followed by a certain delay in seconds.
// Then it quits.  Useful for testing the server.
//

import java.io.*;
import java.net.*;
import java.util.*;
import static java.lang.System.*;

class roboclient {

   static void quit (String format, Object... params) {
      err.printf (format, params);
      exit (1);
   }

   static String ident (options opts) {
      return String.format ("%s: %s %d", opts.progname,
                            opts.hostname, opts.portnumber);
   }

   static class options {
      final String progname = "roboclient";
      String hostname;
      int portnumber;
      String username;
      long delaysec;
      int cycles;
      String robomessage;
      options (String[] args) {
         try {
            if (args.length < 5) throw new NumberFormatException();
            hostname = args[0];
            portnumber = Integer.parseInt (args[1]);
            username = args[2];
            delaysec = Long.parseLong (args[3]);
            cycles = Integer.parseInt (args[4]);
            robomessage = "";
            for (int i = 5; i < args.length; ++i) {
               robomessage += " " + args[i];
            }
         }catch (NumberFormatException exn) {
            quit ("Usage: %s hostname port username delaysec cycles "
                + "message message%n", progname);
         }
      }
   }


   static class reader implements Runnable {
      Socket socket;
      Scanner scanner;
      reader (Socket _socket, Scanner _scanner) {
         scanner = _scanner;
         socket = _socket;
      }
      public void run() {
         while (! socket.isInputShutdown() && scanner.hasNextLine()) {
            out.printf ("%s%n", scanner.nextLine());
         }
         scanner.close();
      }
   }

   static class writer implements Runnable {
      Socket socket;
      options opts;
      PrintWriter writer;
      writer (Socket _socket, options _opts, PrintWriter _writer) {
         socket = _socket;
         opts = _opts;
         writer = _writer;
      }
      public void run() {
         writer.printf ("%s%n", opts.username);
         writer.flush();
         for (int count = 0; count < opts.cycles; ++count) {
            if (socket.isOutputShutdown()) break;
            try {
                Thread.sleep (opts.delaysec * 1000);
            }catch (InterruptedException error) {
            }
            writer.printf ("%s[%d] %s (%s %d) --%s%n",
                           opts.progname, count, opts.username,
                           opts.hostname, opts.portnumber,
                           opts.robomessage);
            writer.flush();
         }
         writer.close();
      }
   }

   public static void main (String[] args) {
      Scanner stdin = new Scanner (System.in);
      options opts = new options (args);
      try {
         Socket socket = new Socket (opts.hostname, opts.portnumber);
         out.printf ("%s: socket OK%n", ident (opts));
         Thread reading = new Thread (new reader (socket,
                          new Scanner (socket.getInputStream())));
         Thread writing = new Thread (new writer (socket, opts,
                          new PrintWriter (socket.getOutputStream())));
         reading.start();
         writing.start();
      }catch (IOException exn) {
         quit ("%s: %s%n", ident (opts), exn);
      }catch (IllegalArgumentException exn) {
         quit ("%s: %s%n", ident (opts), exn);
      }
   }

}

```

```
::::::::::::::
roboserver.log
::::::::::::::
bash-2$ multiserver 8888
multiserver: port 8888: waiting for client
multiserver: port 8888: socket.accept OK
multiserver: port 8888: worker 1: starting
1[1]Hello
1[2]roboclient[0] Hello (localhost 8888) -- Message from World.
1[3]roboclient[1] Hello (localhost 8888) -- Message from World.
1[4]roboclient[2] Hello (localhost 8888) -- Message from World.
1[5]roboclient[3] Hello (localhost 8888) -- Message from World.
multiserver: port 8888: socket.accept OK
multiserver: port 8888: worker 2: starting
2[1]Foo-Bar
2[2]roboclient[0] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
1[6]roboclient[4] Hello (localhost 8888) -- Message from World.
2[3]roboclient[1] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[4]roboclient[2] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
1[7]roboclient[5] Hello (localhost 8888) -- Message from World.
2[5]roboclient[3] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[6]roboclient[4] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
1[8]roboclient[6] Hello (localhost 8888) -- Message from World.
2[7]roboclient[5] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[8]roboclient[6] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
1[9]roboclient[7] Hello (localhost 8888) -- Message from World.
2[9]roboclient[7] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[10]roboclient[8] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
1[10]roboclient[8] Hello (localhost 8888) -- Message from World.
2[11]roboclient[9] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
multiserver: port 8888: worker 2: finished
1[11]roboclient[9] Hello (localhost 8888) -- Message from World.
multiserver: port 8888: worker 1: finished
^C
bash-3$ exit
exit

::::::::::::::
roboclient1.log
::::::::::::::
bash-2$ roboclient localhost 8888 Hello 2 10 Message from World.
roboclient: localhost 8888: socket OK
1[1]Hello
1[2]roboclient[0] Hello (localhost 8888) -- Message from World.
1[3]roboclient[1] Hello (localhost 8888) -- Message from World.
1[4]roboclient[2] Hello (localhost 8888) -- Message from World.
1[5]roboclient[3] Hello (localhost 8888) -- Message from World.
1[6]roboclient[4] Hello (localhost 8888) -- Message from World.
1[7]roboclient[5] Hello (localhost 8888) -- Message from World.
1[8]roboclient[6] Hello (localhost 8888) -- Message from World.
1[9]roboclient[7] Hello (localhost 8888) -- Message from World.
1[10]roboclient[8] Hello (localhost 8888) -- Message from World.
bash-3$ exit
exit

::::::::::::::
roboclient2.log
::::::::::::::
bash-2$ roboclient localhost 8888 Foo-Bar 1 10 Message from Foo Bar Baz Qux.
roboclient: localhost 8888: socket OK
2[1]Foo-Bar
```

```
2[2]roboclient[0] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[3]roboclient[1] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[4]roboclient[2] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[5]roboclient[3] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[6]roboclient[4] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[7]roboclient[5] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[8]roboclient[6] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[9]roboclient[7] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
2[10]roboclient[8] Foo-Bar (localhost 8888) -- Message from Foo Bar Baz Qux.
bash-3$ exit
exit

```