```
 1
 2 import java.net.*;
 3 import java.io.*;
 4 import java.util.*;
 5
 6 public class JHTTP extends Thread {
 7
 8     private File documentRootDirectory;
 9     private String indexFileName = "index.html";
10     private ServerSocket server;
11     private int numThreads = 50;
12
13     public JHTTP(File documentRootDirectory, int port,
14             String indexFileName) throws IOException {
15
16         if (!documentRootDirectory.isDirectory()) {
17             throw new IOException(documentRootDirectory
18                     + " does not exist as a directory");
19         }
20         this.documentRootDirectory = documentRootDirectory;
21         this.indexFileName = indexFileName;
22         this.server = new ServerSocket(port);
23     }
24
25     public JHTTP(File documentRootDirectory, int port)
26             throws IOException {
27         this(documentRootDirectory, port, "index.html");
28     }
29
30     public JHTTP(File documentRootDirectory) throws IOException {
31         this(documentRootDirectory, 80, "index.html");
32     }
33
34     public void run() {
35
36         for (int i = 0; i < numThreads; i++) {
37             Thread t = new Thread(
38                     new RequestProcessor(documentRootDirectory, indexFileName));
39             t.start();
40         }
41         System.out.println("Accepting connections on port "
42                 + server.getLocalPort());
43         System.out.println("Document Root: " + documentRootDirectory);
44         while (true) {
45             try {
46                 Socket request = server.accept();
47                 RequestProcessor.processRequest(request);
48             } catch (IOException ex) {
49             }
50         }
51
52     }
53
54     public static void main(String[] args) {
55
56         // get the Document root
57         File docroot;
58         try {
59             docroot = new File(args[0]);
60         } catch (ArrayIndexOutOfBoundsException ex) {
61             System.out.println("Usage: java JHTTP docroot port indexfile");
```

```
62              return;
63          }
64
65          // set the port to listen on
66          int port;
67          try {
68              port = Integer.parseInt(args[1]);
69              if (port < 0 || port > 65535) {
70                  port = 80;
71              }
72          } catch (Exception ex) {
73              port = 80;
74          }
75
76          try {
77              JHTTP webserver = new JHTTP(docroot, port);
78              webserver.start();
79          } catch (IOException ex) {
80              System.out.println("Server could not start because of an "
81                      + ex.getClass());
82              System.out.println(ex);
83          }
84
85      }
86  }
```

```
101
102 import java.net.*;
103 import java.io.*;
104 import java.util.*;
105
106 public class RequestProcessor implements Runnable {
107
108     private static List pool = new LinkedList();
109     private File documentRootDirectory;
110     private String indexFileName = "index.html";
111
112     public RequestProcessor(File documentRootDirectory,
113             String indexFileName) {
114
115         if (documentRootDirectory.isFile()) {
116             throw new IllegalArgumentException(
117                     "documentRootDirectory must be a directory, not a file");
118         }
119         this.documentRootDirectory = documentRootDirectory;
120         try {
121             this.documentRootDirectory = documentRootDirectory.getCanonicalFile();
122         } catch (IOException ex) {
123         }
124         if (indexFileName != null) {
125             this.indexFileName = indexFileName;
126         }
127     }
128
129     public static void processRequest(Socket request) {
130
131         synchronized (pool) {
132             pool.add(pool.size(), request);
133             pool.notifyAll();
134         }
135
136     }
137
138     public void run() {
139
140         // for security checks
141         String root = documentRootDirectory.getPath();
142
143         while (true) {
144             Socket connection;
145             synchronized (pool) {
146                 while (pool.isEmpty()) {
147                     try {
148                         pool.wait();
149                     } catch (InterruptedException ex) {
150                     }
151                 }
152                 connection = (Socket) pool.remove(0);
153             }
154
155             try {
156                 String filename;
157                 String contentType;
158                 OutputStream raw = new BufferedOutputStream(
159                         connection.getOutputStream());
160                 Writer out = new OutputStreamWriter(raw);
161                 Reader in = new InputStreamReader(
```

```
                        new BufferedInputStream(
                        connection.getInputStream()), "ASCII");
                StringBuffer requestLine = new StringBuffer();
                int c;
                while (true) {
                    c = in.read();
                    if (c == '\r' || c == '\n') {
                        break;
                    }
                    requestLine.append((char) c);
                }

                String get = requestLine.toString();

                // log the request
                System.out.println(get);

                StringTokenizer st = new StringTokenizer(get);
                String method = st.nextToken();
                String version = "";
                filename = st.nextToken();
                if (filename.endsWith("/")) {
                    filename += indexFileName;
                }
                contentType = guessContentTypeFromName(filename);
                if (st.hasMoreTokens()) {
                    version = st.nextToken();
                }

                if (method.equals("GET")) {
                    File theFile = new File(documentRootDirectory,
                            filename.substring(1, filename.length()));
                    if (theFile.canRead()
                            // Don't let clients outside the document root
                            && theFile.getCanonicalPath().startsWith(root)) {
                        DataInputStream fis = new DataInputStream(
                                new BufferedInputStream(
                                new FileInputStream(theFile)));
                        byte[] theData = new byte[(int) theFile.length()];
                        fis.readFully(theData);
                        fis.close();
                        if (version.startsWith("HTTP/")) {  // send a MIME header
                            out.write("HTTP/1.0 200 OK\r\n");
                            Date now = new Date();
                            out.write("Date: " + now + "\r\n");
                            out.write("Server: JHTTP/1.0\r\n");
                            out.write("Content-length: " + theData.length + "\r\n");
                            out.write("Content-type: " + contentType + "\r\n\r\n");
                            out.flush();
                        }  // end if

                        // send the file; it may be an image or other binary data
                        // so use the underlying output stream
                        // instead of the writer
                        raw.write(theData);
                        raw.flush();
                    } // end if
                    else {  // can't find the file
                        if (version.startsWith("HTTP/")) {  // send a MIME header
                            out.write("HTTP/1.0 404 File Not Found\r\n");
                            Date now = new Date();
```

```
223                             out.write("Date: " + now + "\r\n");
224                             out.write("Server: JHTTP/1.0\r\n");
225                             out.write("Content-type: text/html\r\n\r\n");
226                         }
227                         out.write("<HTML>\r\n");
228                         out.write("<HEAD><TITLE>File Not Found</TITLE>\r\n");
229                         out.write("</HEAD>\r\n");
230                         out.write("<BODY>");
231                         out.write("<H1>HTTP Error 404: File Not Found</H1>\r\n");
232                         out.write("</BODY></HTML>\r\n");
233                         out.flush();
234                     }
235                 } else {  // method does not equal "GET"
236                     if (version.startsWith("HTTP/")) {  // send a MIME header
237                         out.write("HTTP/1.0 501 Not Implemented\r\n");
238                         Date now = new Date();
239                         out.write("Date: " + now + "\r\n");
240                         out.write("Server: JHTTP 1.0\r\n");
241                         out.write("Content-type: text/html\r\n\r\n");
242                     }
243                     out.write("<HTML>\r\n");
244                     out.write("<HEAD><TITLE>Not Implemented</TITLE>\r\n");
245                     out.write("</HEAD>\r\n");
246                     out.write("<BODY>");
247                     out.write("<H1>HTTP Error 501: Not Implemented</H1>\r\n");
248                     out.write("</BODY></HTML>\r\n");
249                     out.flush();
250                 }
251             } catch (IOException ex) {
252             } finally {
253                 try {
254                     connection.close();
255                 } catch (IOException ex) {
256                 }
257             }
258 
259         } // end while
260 
261     } // end run
262 
263     public static String guessContentTypeFromName(String name) {
264         if (name.endsWith(".html") || name.endsWith(".htm")) {
265             return "text/html";
266         } else if (name.endsWith(".txt") || name.endsWith(".java")) {
267             return "text/plain";
268         } else if (name.endsWith(".gif")) {
269             return "image/gif";
270         } else if (name.endsWith(".class")) {
271             return "application/octet-stream";
272         } else if (name.endsWith(".jpg") || name.endsWith(".jpeg")) {
273             return "image/jpeg";
274         } else {
275             return "text/plain";
276         }
277     }
278 } // end RequestProcessor
279 
```