```
import java.net.*;
import java.io.*;
import java.util.*;

/** A simple HTTP server that generates a Web page showing all
 *  the data that it received from the Web client (usually
 *  a browser). To use this server, start it on the system of
 *  your choice, supplying a port number if you want something
 *  other than port 8088. Call this system server.com. Next,
 *  start a Web browser on the same or a different system, and
 *  connect to http://server.com:8088/whatever. The resultant
 *  Web page will show the data that your browser sent. For
 *  debugging in a servlet or other server-side program, specify
 *  http://server.com:8088/whatever as the ACTION of your HTML
 *  form. You can send GET or POST data; either way, the
 *  resultant page will show what your browser sent.
 *  <P>
 *  Taken from Core Servlets and JavaServer Pages 2nd Edition
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.coreservlets.com/.
 *  &copy; 2003 Marty Hall and Larry Brown.
 *  May be freely used or adapted.
 */

public class EchoServer extends NetworkServer {
  protected int maxRequestLines = 50;
  protected String serverName = "EchoServer";

  /** Supply a port number as a command-line
   *  argument. Otherwise, use port 8088.
   */

  public static void main(String[] args) {
    int port = 8088;
    if (args.length > 0) {
      try {
        port = Integer.parseInt(args[0]);
      } catch(NumberFormatException nfe) {}
    }
    new EchoServer(port, 0);
  }

  public EchoServer(int port, int maxConnections) {
    super(port, maxConnections);
    listen();
  }

  /** Overrides the NetworkServer handleConnection method to
   *  read each line of data received, save it into an array
   *  of strings, then send it back embedded inside a PRE
   *  element in an HTML page.
   */

  public void handleConnection(Socket server)
      throws IOException{
    System.out.println
        (serverName + ": got connection from " +
         server.getInetAddress().getHostName());
    BufferedReader in = SocketUtil.getReader(server);
    PrintWriter out = SocketUtil.getWriter(server);
    String[] inputLines = new String[maxRequestLines];
    int i;
    for (i=0; i<maxRequestLines; i++) {
      inputLines[i] = in.readLine();
      if (inputLines[i] == null) // Client closed connection.
        break;
```

```
      if (inputLines[i].length() == 0) { // Blank line.
        if (usingPost(inputLines)) {
          readPostData(inputLines, i, in);
          i = i + 2;
        }
        break;
      }
    }
    printHeader(out);
    for (int j=0; j<i; j++) {
      out.println(inputLines[j]);
    }
    printTrailer(out);
    server.close();
  }

  // Send standard HTTP response and top of a standard Web page.
  // Use HTTP 1.0 for compatibility with all clients.

  private void printHeader(PrintWriter out) {
    out.println
      ("HTTP/1.0 200 OK\r\n" +
       "Server: " + serverName + "\r\n" +
       "Content-Type: text/html\r\n" +
       "\r\n" +
       "<!DOCTYPE HTML PUBLIC " +
         "\"-//W3C//DTD HTML 4.0 Transitional//EN\">\n" +
       "<HTML>\n" +
       "<HEAD>\n" +
       "  <TITLE>" + serverName + " Results</TITLE>\n" +
       "</HEAD>\n" +
       "\n" +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +
       "<H1 ALIGN=\"CENTER\">" + serverName +
         " Results</H1>\n" +
       "Here is the request line and request headers\n" +
       "sent by your browser:\n" +
       "<PRE>");
  }

  // Print bottom of a standard Web page.

  private void printTrailer(PrintWriter out) {
    out.println
      ("</PRE>\n" +
       "</BODY>\n" +
       "</HTML>\n");
  }

  // Normal Web page requests use GET, so this server can simply
  // read a line at a time. However, HTML forms can also use
  // POST, in which case we have to determine the number of POST
  // bytes that are sent so we know how much extra data to read
  // after the standard HTTP headers.

  private boolean usingPost(String[] inputs) {
    return(inputs[0].toUpperCase().startsWith("POST"));
  }

  private void readPostData(String[] inputs, int i,
                            BufferedReader in)
      throws IOException {
    int contentLength = contentLength(inputs);
    char[] postData = new char[contentLength];
    in.read(postData, 0, contentLength);
    inputs[++i] = new String(postData, 0, contentLength);
```

```
  }

  // Given a line that starts with Content-Length,
  // this returns the integer value specified.
  private int contentLength(String[] inputs) {
    String input;
    for (int i=0; i<inputs.length; i++) {
      if (inputs[i].length() == 0)
        break;
      input = inputs[i].toUpperCase();
      if (input.startsWith("CONTENT-LENGTH"))
        return(getLength(input));
    }
    return(0);
  }

  private int getLength(String length) {
    StringTokenizer tok = new StringTokenizer(length);
    tok.nextToken();
    return(Integer.parseInt(tok.nextToken()));
  }
}
```