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

/** A starting point for network servers. You'll need to
 *  override handleConnection, but in many cases listen can
 *  remain unchanged. NetworkServer uses SocketUtil to simplify
 *  the creation of the PrintWriter and BufferedReader.
 *  <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 NetworkServer {
  private int port, maxConnections;

  /** Build a server on specified port. It will continue to
   *  accept connections, passing each to handleConnection until
   *  an explicit exit command is sent (e.g., System.exit) or
   *  the maximum number of connections is reached. Specify
   *  0 for maxConnections if you want the server to run
   *  indefinitely.
   */

  public NetworkServer(int port, int maxConnections) {
    setPort(port);
    setMaxConnections(maxConnections);
  }

  /** Monitor a port for connections. Each time one is
   *  established, pass resulting Socket to handleConnection.
   */

  public void listen() {
    int i=0;
    try {
      ServerSocket listener = new ServerSocket(port);
      Socket server;
      while((i++ < maxConnections) || (maxConnections == 0)) {
        server = listener.accept();
        handleConnection(server);
      }
    } catch (IOException ioe) {
      System.out.println("IOException: " + ioe);
      ioe.printStackTrace();
    }
  }

  /** This is the method that provides the behavior to the
   *  server, since it determines what is done with the
   *  resulting socket. <B>Override this method in servers
   *  you write.</B>
   *  <P>
   *  This generic version simply reports the host that made
   *  the connection, shows the first line the client sent,
   *  and sends a single line in response.
   */

  protected void handleConnection(Socket server)
      throws IOException{
    BufferedReader in = SocketUtil.getReader(server);
    PrintWriter out = SocketUtil.getWriter(server);
    System.out.println
      ("Generic Network Server: got connection from " +
```

```
        server.getInetAddress().getHostName() + "\n" +
        "with first line '" + in.readLine() + "'");
    out.println("Generic Network Server");
    server.close();
  }

  /** Gets the max connections server will handle before
   *  exiting. A value of 0 indicates that server should run
   *  until explicitly killed.
   */

  public int getMaxConnections() {
    return(maxConnections);
  }

  /** Sets max connections. A value of 0 indicates that server
   *  should run indefinitely (until explicitly killed).
   */

  public void setMaxConnections(int maxConnections) {
    this.maxConnections = maxConnections;
  }

  /** Gets port on which server is listening. */

  public int getPort() {
    return(port);
  }

  /** Sets port. <B>You can only do before "connect" is
   *  called.</B> That usually happens in the constructor.
   */

  protected void setPort(int port) {
    this.port = port;
  }
}
```