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

/** A multithreaded variation of EchoServer.
 *  <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 ThreadedEchoServer extends EchoServer
                                implements Runnable {
  public static void main(String[] args) {
    int port = 8088;
    if (args.length > 0) {
      try {
        port = Integer.parseInt(args[0]);
      } catch(NumberFormatException nfe) {}
    }
    ThreadedEchoServer echoServer =
      new ThreadedEchoServer(port, 0);
    echoServer.serverName = "Threaded EchoServer";
  }

  public ThreadedEchoServer(int port, int connections) {
    super(port, connections);
  }

  /** The new version of handleConnection starts a thread. This
   *  new thread will call back to the <I>old</I> version of
   *  handleConnection, resulting in the same server behavior
   *  in a multithreaded version. The thread stores the Socket
   *  instance since run doesn't take any arguments, and since
   *  storing the socket in an instance variable risks having
   *  it overwritten if the next thread starts before the run
   *  method gets a chance to copy the socket reference.
   */

  public void handleConnection(Socket server) {
    Connection connectionThread = new Connection(this, server);
    connectionThread.start();
  }

  public void run() {
    Connection currentThread =
      (Connection)Thread.currentThread();
    try {
      super.handleConnection(currentThread.getSocket());
    } catch(IOException ioe) {
      System.out.println("IOException: " + ioe);
      ioe.printStackTrace();
    }
  }
}

/** This is just a Thread with a field to store a Socket object.
 *  Used as a thread-safe means to pass the Socket from
 *  handleConnection to run.
 */

class Connection extends Thread {
  private Socket serverSocket;

  public Connection(Runnable serverObject,
                    Socket serverSocket) {
    super(serverObject);
    this.serverSocket = serverSocket;
  }

  public Socket getSocket() {
    return serverSocket;
  }
}
```