完善的javasocketserver程序计算机二级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/597/2021_2022__E5_AE_8C_E5_96_84_E7_9A_84j_c97_597178.htm 编辑特别推荐: 全国计算机等级考试（等考）指定教材 全国计算机等级考试学习视频 全国计算机等级考试网上辅导招生 全国计算机等级考试时间及科目预告 百考试题教育全国计算机等级考试在线测试平台 全国计算机等级考试资料下载 全国计算机等级考试论坛 计算机等级考试四级应用题解析汇总 2009年下半年全国计算机二级考试报名时间从6月1日起已经开始报名。详情点击：2009年下半年全国计算机等级考试各地报名点汇总。2009年下半年全国计算机二级考试时间是2009年9月19日至23日。更多优质资料尽在百考试题论坛 百考试题在线题库。 /* * Copyright (c) 2000 David Flanagan. All rights reserved. * This code is from the book Java Examples in a Nutshell, 2nd Edition. * It is provided AS-IS, WITHOUT ANY WARRANTY either expressed or implied. * You may study, use, and modify it for any non-commercial purpose. * You may distribute it non-commercially as long as you retain this notice. * For a commercial use license, or to purchase the book (recommended), */ package com.davidflanagan.examples.net. import java.io.*. import java.net.*. import java.util.*. /** * This class is a generic framework for a flexible, multi-threaded server. * It listens on any number of specified ports, and, when it receives a * connection on a port, passes input and output streams to a specified Service * object which provides the actual service. It can limit the number of * concurrent connections, and logs activity to a specified

stream. **/ public class Server { /** * A main() method for running the server as a standalone program. The * command-line arguments to the program should be pairs of servicenames * and port numbers. For each pair, the program will dynamically load the * named Service class, instantiate it, and tell the server to provide * that Service on the specified port. The special -control argument * should be followed by a password and port, and will start special * server control service running on the specified port, protected by the * specified password. **/ public static void main(String[] args) { try { if (args.length lt. args.length) { if (args[i].equals("-control")) { // Handle the -control arg i . String password = args[i]. int port = Integer.parseInt(args[i]). // add control service s.addService(new Control(s, password), port). } else { // Otherwise start a named service on the specified port. // Dynamically load and instantiate a Service class String serviceName = args[i]. Class serviceClass = Class.forName(serviceName). Service service = (Service)serviceClass.newInstance(). int port = Integer.parseInt(args[i]). s.addService(service, port). } } } catch (Exception e) { // Display a message if anything goes wrong System.err.println("Server: " e). System.err.println("Usage: java Server " "[-control gt. gt.] " "[gt. gt. ...]"). System.exit(1). } } // This is the state for the server Map services. // Hashtable mapping ports to Listeners Set connections. // The set of current connections int maxConnections. // The concurrent connection limit ThreadGroup threadGroup. // The threadgroup for all our threads PrintWriter logStream. // Where we send our logging output to /** * This is the Server() constructor. It must be passed a stream * to send log output

to (may be null), and the limit on the number of * concurrent connections. **/ public Server(OutputStream logStream, int maxConnections) { setLogStream(logStream). log("Starting server"). threadGroup = new ThreadGroup(Server.class.getName()). this.maxConnections = maxConnections. services = new HashMap(). connections = new HashSet(maxConnections). } /** * A public method to set the current logging stream. Pass null * to turn logging off **/ public synchronized void setLogStream(OutputStream out) { if (out != null) logStream = new PrintWriter(out). else logStream = null. } /** Write the specified string to the log */ protected synchronized void log(String s) { if (logStream != null) { logStream.println("[" new Date() "] " s). logStream.flush(). } } /** Write the specified object to the log */ protected void log(Object o) { log(o.toString()). } /** * This method makes the server start providing a new service. * It runs the specified Service object on the specified port. **/ public synchronized void addService(Service service, int port) throws IOException { Integer key = new Integer(port). // the hashtable key // Check whether a service is already on that port if (services.get(key) != null) throw new IllegalArgumentException("Port " port " already in use."). // Create a Listener object to listen for connections on the port Listener listener = new Listener(threadGroup, port, service). // Store it in the hashtable services.put(key, listener). // Log it log("Starting service " service.getClass().getName() " on port " port). // Start the listener running. listener.start(). } /** * This method makes the server stop providing a service on a port. * It does not terminate any pending

connections to that service, merely * causes the server to stop accepting new connections **/ public synchronized void removeService(int port) { Integer key = new Integer(port). // hashtable key // Look up the Listener object for the port in the hashtable final Listener listener = (Listener) services.get(key). if (listener == null) return. // Ask the listener to stop listener.pleaseStop(). // Remove it from the hashtable services.remove(key). // And log it. log("Stopping service " listener.service.getClass().getName() " on port " port). } /** * This nested Thread subclass is a "listener". It listens for * connections on a specified port (using a ServerSocket) and when it gets * a connection request, it calls the servers addConnection() method to * accept (or reject) the connection. There is one Listener for each * Service being provided by the Server. **/ public class Listener extends Thread { ServerSocket listen_socket. // The socket to listen for connections int port. // The port were listening on Service service. // The service to provide on that port volatile boolean stop = false. // Whether weve been asked to stop /** * The Listener constructor creates a thread for itself in the * threadgroup. It creates a ServerSocket to listen for connections * on the specified port. It arranges for the ServerSocket to be * interruptible, so that services can be removed from the server. **/ public Listener(ThreadGroup group, int port, Service service) throws IOException { super(group, "Listener:" port). listen_socket = new ServerSocket(port). // give it a non-zero timeout so accept() can be interrupted listen_socket.setSoTimeout(600000). this.port = port. this.service = service. } /** * This is the polite way to get a Listener to

stop accepting * connections ***/ 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com