Criando Chat

Multithread com Socket em Java

Alunos: **Rômulo C. Guerra - 1611275**

Definições

**Socket** :

De acordo com a própria Oracle: “Socket é um ponto de comunicação entre duas máquinas”, ou seja, podemos enviar mensagens entre a máquina A e a máquina B através de uma conexão estabelecida com o Socket.

***Thread :***

Thread é um pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se autodividir em duas ou mais tarefas. É o termo em inglês para Linha ou Encadeamento de Execução. Essas tarefas múltiplas podem ser executadas simultaneamente para rodar mais rápido do que um programa em um único bloco ou praticamente juntas, mas que são tão rápidas que parecem estar trabalhando em conjunto ao mesmo tempo.

***Multithread :***

Multithread é a capacidade que o sistema operacional possui de executar várias threads simultaneamente sem que uma interfira na outra. Estas threads compartilham os recursos do processo, mas são capazes de ser executadas de forma independente.

***Servidor :***

O servidor servirá como unidade centralizadora de todas as conexões recebidas via socket e terá como responsabilidade o envio de uma mensagem (recebida de um cliente) para todos os demais conectados no servidor. Quando um cliente se conecta a ele o mesmo cria uma Thread para aquele cliente, ou seja, cada conexão terá sua respectiva Thread e o servidor fará a gestão disso.

**Cliente :**

Cada usuário criará uma instância do cliente e fará uma conexão com o servidor socket. O cliente deverá informar o endereço do server socket e a respectiva porta.

Código

Nosso código será dividido em duas Classes **Servidor.java** e **Cliente.java.**

**Servidor.java:**

import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.io.OutputStream;  
import java.io.OutputStreamWriter;  
import java.io.Writer;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.JTextField;  
  
public class Servidor extends Thread {  
  
 private static ArrayList<BufferedWriter>*clientes*;  
 private static ServerSocket *server*;  
 private String nome;  
 private Socket con;  
 private InputStream in;  
 private InputStreamReader inr;  
 private BufferedReader bfr;  
  
 public Servidor(Socket con){  
 this.con = con;  
 try {  
 in = con.getInputStream();  
 inr = new InputStreamReader(in);  
 bfr = new BufferedReader(inr);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void run(){  
  
 try{  
  
 String msg;  
 OutputStream ou = this.con.getOutputStream();  
 Writer ouw = new OutputStreamWriter(ou);  
 BufferedWriter bfw = new BufferedWriter(ouw);  
 *clientes*.add(bfw);  
 nome = msg = bfr.readLine();  
  
 while(!"Sair".equalsIgnoreCase(msg) && msg != null)  
 {  
 msg = bfr.readLine();  
 sendToAll(bfw, msg);  
 System.*out*.println(msg);  
 }  
  
 }catch (Exception e) {  
 e.printStackTrace();  
  
 }  
 }  
  
 public void sendToAll(BufferedWriter bwSaida, String msg) throws IOException  
 {  
 BufferedWriter bwS;  
  
 for(BufferedWriter bw : *clientes*){  
 bwS = (BufferedWriter)bw;  
 if(!(bwSaida == bwS)){  
 bw.write(nome + " -> " + msg+"\r\n");  
 bw.flush();  
 }  
 }  
 }  
  
 public static void main(String []args) {  
  
 try{  
 JLabel lblMessage = new JLabel("Porta do Servidor:");  
 JTextField txtPorta = new JTextField("12345");  
 Object[] texts = {lblMessage, txtPorta };  
 JOptionPane.*showMessageDialog*(null, texts);  
 *server* = new ServerSocket(Integer.*parseInt*(txtPorta.getText()));  
 *clientes* = new ArrayList<BufferedWriter>();  
 JOptionPane.*showMessageDialog*(null,"Servidor ativo na porta: "+  
 txtPorta.getText());  
  
 while(true){  
 System.*out*.println("Aguardando conexão...");  
 Socket con = *server*.accept();  
 System.*out*.println("Cliente conectado...");  
 Thread t = new Servidor(con);  
 t.start();  
 }  
  
 }catch (Exception e) {  
  
 e.printStackTrace();  
 }  
 }  
}

**Cliente.java :**

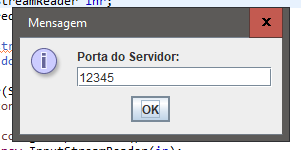
import java.awt.Color;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.io.OutputStream;  
import java.io.OutputStreamWriter;  
import java.io.Writer;  
import java.net.Socket;  
import javax.swing.\*;  
  
public class Cliente extends JFrame implements ActionListener, KeyListener {  
  
 private static final long *serialVersionUID* = 1L;  
 private JTextArea texto;  
 private JTextField txtMsg;  
 private JButton btnSend;  
 private JButton btnSair;  
 private JLabel lblHistorico;  
 private JLabel lblMsg;  
 private JPanel pnlContent;  
 private Socket socket;  
 private OutputStream ou ;  
 private Writer ouw;  
 private BufferedWriter bfw;  
 private JTextField txtIP;  
 private JTextField txtPorta;  
 private JTextField txtNome;  
  
 public Cliente() throws IOException{  
 JLabel lblMessage = new JLabel("Verificar!");  
 txtIP = new JTextField("127.0.0.1");  
 txtPorta = new JTextField("12345");  
 txtNome = new JTextField("Cliente");  
 Object[] texts = {lblMessage, txtIP, txtPorta, txtNome };  
 JOptionPane.*showMessageDialog*(null, texts);  
 pnlContent = new JPanel();  
 texto = new JTextArea(10,20);  
 texto.setEditable(false);  
 texto.setBackground(new Color(240,240,240));  
 txtMsg = new JTextField(20);  
 lblHistorico = new JLabel("Hist�rico");  
 lblMsg = new JLabel("Mensagem");  
 btnSend = new JButton("Enviar");  
 btnSend.setToolTipText("Enviar Mensagem");  
 btnSair = new JButton("Sair");  
 btnSair.setToolTipText("Sair do Chat");  
 btnSend.addActionListener(this);  
 btnSair.addActionListener(this);  
 btnSend.addKeyListener(this);  
 txtMsg.addKeyListener(this);  
 JScrollPane scroll = new JScrollPane(texto);  
 texto.setLineWrap(true);  
 pnlContent.add(lblHistorico);  
 pnlContent.add(scroll);  
 pnlContent.add(lblMsg);  
 pnlContent.add(txtMsg);  
 pnlContent.add(btnSair);  
 pnlContent.add(btnSend);  
 pnlContent.setBackground(Color.*LIGHT\_GRAY*);  
 texto.setBorder(BorderFactory.*createEtchedBorder*(Color.*BLUE*,Color.*BLUE*));  
 txtMsg.setBorder(BorderFactory.*createEtchedBorder*(Color.*BLUE*, Color.*BLUE*));  
 setTitle(txtNome.getText());  
 setContentPane(pnlContent);  
 setLocationRelativeTo(null);  
 setResizable(false);  
 setSize(250,300);  
 setVisible(true);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 }  
  
 public void conectar() throws IOException{  
  
 socket = new Socket(txtIP.getText(),Integer.*parseInt*(txtPorta.getText()));  
 ou = socket.getOutputStream();  
 ouw = new OutputStreamWriter(ou);  
 bfw = new BufferedWriter(ouw);  
 bfw.write(txtNome.getText()+"\r\n");  
 bfw.flush();  
 }  
  
 public void enviarMensagem(String msg) throws IOException{  
  
 if(msg.equals("Sair")){  
 bfw.write("Desconectado \r\n");  
 texto.append("Desconectado \r\n");  
 }else{  
 bfw.write(msg+"\r\n");  
 texto.append( txtNome.getText() + " diz -> " + txtMsg.getText()+"\r\n");  
 }  
 bfw.flush();  
 txtMsg.setText("");  
 }  
  
 public void escutar() throws IOException{  
  
 InputStream in = socket.getInputStream();  
 InputStreamReader inr = new InputStreamReader(in);  
 BufferedReader bfr = new BufferedReader(inr);  
 String msg = "";  
  
 while(!"Sair".equalsIgnoreCase(msg))  
  
 if(bfr.ready()){  
 msg = bfr.readLine();  
 if(msg.equals("Sair"))  
 texto.append("Servidor caiu! \r\n");  
 else  
 texto.append(msg+"\r\n");  
 }  
 }  
  
 public void sair() throws IOException{  
  
 enviarMensagem("Sair");  
 bfw.close();  
 ouw.close();  
 ou.close();  
 socket.close();  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 try {  
 if(e.getActionCommand().equals(btnSend.getActionCommand()))  
 enviarMensagem(txtMsg.getText());  
 else  
 if(e.getActionCommand().equals(btnSair.getActionCommand()))  
 sair();  
 } catch (IOException e1) {  
 // *TODO Auto-generated catch block* e1.printStackTrace();  
 }  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
  
 if(e.getKeyCode() == KeyEvent.*VK\_ENTER*){  
 try {  
 enviarMensagem(txtMsg.getText());  
 } catch (IOException e1) {  
 // *TODO Auto-generated catch block* e1.printStackTrace();  
 }  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent arg0) {  
 // *TODO Auto-generated method stub* }  
  
 @Override  
 public void keyTyped(KeyEvent arg0) {  
 // *TODO Auto-generated method stub* }  
  
 public static void main(String []args) throws IOException{  
  
 Cliente app = new Cliente();  
 app.conectar();  
 app.escutar();  
 }  
  
}

**Passo - a - Passo**

**1º -** Iniciar o **Servidor.java**:

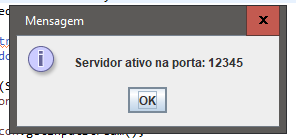
Execute primeiramente a classe servidor.

Aparecerá a caixa de texto (Figura 1). Nela vamos informar a porta da rede que nosso servidor (***Socket)*** ira executar. Neste exemplo colocaremos a porta “12345”, após informar a porta aperte “OK”.



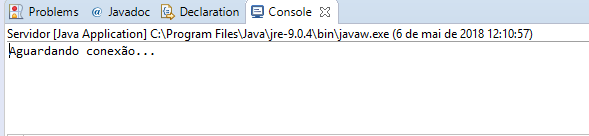
Assim que informado a porta nosso servidor ira executar o ***Método*** ***RUN***: Toda vez que um cliente novo chega ao servidor, esse método é acionado e alocado numa Thread e também fica verificando se existe alguma mensagem nova.

E aparecera a seguinte mensagem (Figura 2:)



Aperte novamente “OK”. Pronto nosso servidor esta configurado.

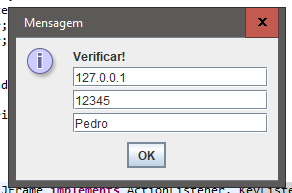
Podemos ver no terminal que o nosso servidor esta esperando um cliente se conectar:



**2º** - Iniciar o **Cliente.java**:

Execute a classe cliente.java

Aparecerá a caixa de texto (Figura 4). Nela vamos informar o endereço IP, a porta da rede e o nome do cliente.



Neste exemplo nossa maquina é o servidor e ao mesmo tempo cliente, então no primeiro campo informaremos o endereço IP local “127.0.0.1”. Caso queira colocar o servidor em outro computador

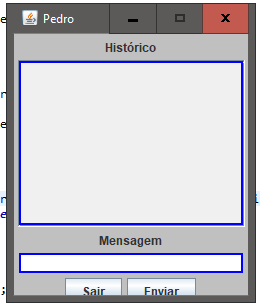
Coloque neste campo o endereço IP do computador que será o servidor. No segundo campo informe a porta que colocamos no nosso servidor, no caso “12345”. Agora no terceiro campo informe o nome deste cliente, este nome será como os outros clientes iram te ver quando mandar uma mensagem.

Após informar os três campos aperte “OK”

**3º** - Utilizar o chat:

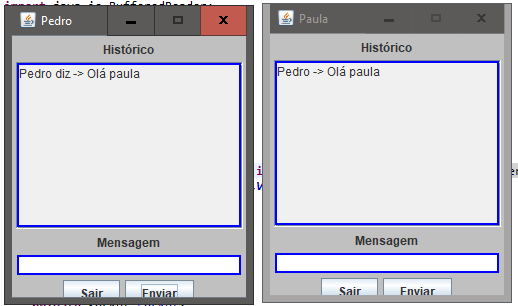
Para esse exemplo execute o passo 2º pelo menos uma segunda vez, para termos 2 cliente.

Após o passo 2º, aparecera a tela de chat (Figura 5).



Para enviar uma mensagem, digite no campo Mensagem e aperte “Enviar”.

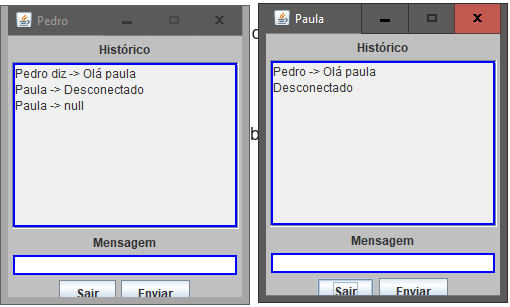
Quando enviamos uma mensagem, a mesma parecera no campo Histórico tanto no seu quanto no de todos os clientes conectados neste chat. Conforme apresenta na Figura 6.



Você pode colocar quantos clientes quiser em cada servidor.

**4º** - Desconectar

Para se desconectar do chat basta apertar o botão “Sair”.



Depois pare o serviço do servidor para desligar o chat.