1. Chat UDP:

**Receiver:**

public class Receive extends javax.swing.JFrame implements Runnable {  
  
private DatagramSocket socket;

private DatagramPacket psend;

private DatagramPacket preceive;

private InetAddress inet;

private int port;

public Receive() throws SocketException {

initComponents();

this.setTitle("Receiver");

socket = new DatagramSocket(5000);

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

send(jTextField1.getText().trim());

}

private void jTextField1KeyPressed(java.awt.event.KeyEvent evt) if(evt.getKeyCode()==KeyEvent.VK\_ENTER){

send(jTextField1.getText().trim());

}

}

public void send(String data) {

if (!data.equals("")) {

try {

byte[] bdata = new byte[1024];

bdata = data.getBytes("UTF-8");

psend = new DatagramPacket(bdata, bdata.length, inet, port);

socket.send(psend);

jTextArea1.append("Receive say: " + data + "\n");

jTextField1.setText("");

} catch (IOException ex) {

Logger.getLogger(Sender.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static void main(String args[]) throws SocketException {

Receive re = new Receive();

re.setVisible(true);

Thread thread = new Thread(re);

thread.start();

}

@Override

public void run() {

byte[] receive ;

while (true) {

try {

receive = new byte[1024];

preceive = new DatagramPacket(receive, 1024);

socket.receive(preceive);

inet = preceive.getAddress();

port = preceive.getPort();

String sreceive = new String(preceive.getData(), 0, preceive.getData().length, "UTF-8");

System.out.println(sreceive);

if(!sreceive.trim().equals("\*\*\* Connect \*\*\* success \*\*\*"))

jTextArea1.append("Sender say: " + sreceive + "\n");

String[] matric = sreceive.split(",");

int line;

int value;

for(int i=0; i<matric.length;i++){

line = matric[i].split("\n").length;

}

} catch (IOException ex) {

Logger.getLogger(Receive.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

**Sender:**

private DatagramSocket socket;

private DatagramPacket pSend;

private DatagramPacket precieve;

private byte[] bdata = new byte[1024];

private byte[] bdatare = new byte[1024];

private final InetAddress inet;

private final int port = 5000;

public Sender() throws UnknownHostException, SocketException {

this.inet = InetAddress.getByName("127.0.0.1");

initComponents();

this.setTitle("Sender");

socket = new DatagramSocket();

send("\*\*\* Connect \*\*\* success \*\*\*");

}

public void send(String ssend) {

if (!ssend.equals("")) {

try {

byte []a=new byte[1024];

a = ssend.getBytes("UTF-8");

pSend = new DatagramPacket(a, a.length, inet, port);

socket.send(pSend);

if(!ssend.equals("\*\*\* Connect \*\*\* success \*\*\*"))

jTextArea1.append("Sender say: " + ssend + "\n");

jTextField1.setText("");

} catch (IOException ex) {

Logger.getLogger(Sender.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

send(jTextField1.getText().trim());

}

private void jTextField1KeyPressed(java.awt.event.KeyEvent evt)

if (evt.getKeyCode() == KeyEvent.VK\_ENTER) {

send(jTextField1.getText().trim());

}

}

public static void main(String args[]) throws UnknownHostException, SocketException {

Sender fs = new Sender();

fs.setVisible(true);

Thread threadSend = new Thread(fs);

threadSend.start();

}

@Override

public void run() {

while (true) {

try {

precieve = new DatagramPacket(bdatare, 1024);

socket.receive(precieve);

String receive = new String(precieve.getData(), 0, precieve.getLength(), "UTF-8");

jTextArea1.append("Receive say: " + receive + "\n");

} catch (IOException ex) {

Logger.getLogger(Sender.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

1. RMI

Interface:

public interface SoHHInterface extends Remote{

public String check(int a)throws RemoteException;

}

IMPL:

public class SoHH extends UnicastRemoteObject implements SoHHInterface {

public SoHH() throws RemoteException {

super();

}

@Override

public String check(int a) throws RemoteException {

int s = 0;

String text="Số hoàn hảo "+a+" = ";

for (int i = 1; i < a; i++) {

if (a % i == 0) {

s += i;

text +=" + "+i;

}

}

if (s == a) {

text+=".";

return text;

}

return "";

}

}

Server:

public static void main(String[] args) throws RemoteException, AlreadyBoundException {

Registry reg = LocateRegistry.createRegistry(5678);

SoHH soHH = new SoHH();

reg.bind("soHH", soHH);

System.out.println("Service ready ...");

}

Client:

private Registry reg;

private SoHHInterface soHHInter;

public JClient() throws RemoteException, NotBoundException {

initComponents();

this.setTitle("Client");

reg = LocateRegistry.getRegistry("127.0.0.1", 5678);

soHHInter = (SoHHInterface) reg.lookup("soHH");

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed

if (soHH.getText().trim().length() != 0) {

show.setText("Danh sách các số hoàn hảo nhỏ hơn " + Integer.parseInt(soHH.getText().trim()) + " là: ");

for (int i = 1; i < Integer.parseInt(soHH.getText().trim()); i++) {

try {

String valueData = soHHInter.check(i);

if (valueData.trim().length() != 0) {

show.append("\n" + valueData);

}

} catch (RemoteException ex) {

Logger.getLogger(JClient.class.getName()).log(Level.SEVERE, null, ex);

}

}

}else{

JOptionPane.showMessageDialog(null, "Hãy nhập số lượng xét!", "Thông báo", JOptionPane.INFORMATION\_MESSAGE);

}

public static void main(String args[]) throws RemoteException, NotBoundException {

JClient jc = new JClient();

jc.setVisible(true);

}

Giai thừa:

@Override

public int giaithua(int n) throws RemoteException {

if (n > 0) {

return n \* giaithua(n - 1);

} else {

return 1;

}

}

Số mũ:

@Override

public int pow(int a, int b) throws RemoteException {

int a1= (int) Math.pow(a, b);

return a1;

}

Chuỗi:

@Override

public boolean soNT(int n) throws RemoteException {

int sqrt = (int) Math.sqrt(n);

if(n<=2){

return false;

}

for(int i=2;i<sqrt;i++){

if(n%i==0){

return false;

}

}

return true;

}

@Override

public Map<String, Integer> token(String e) throws RemoteException {

Map<String, Integer> hashMap = new HashMap<String, Integer>();

String[] words = e.split("[ .!?]");

for (int i = 0; i < words.length; i++) {

if (words[i].length() > 1) {

if (hashMap.get(words[i]) != null) {

int value = hashMap.get(words[i]).intValue();

value++;

hashMap.put(words[i], value);

} else {

hashMap.put(words[i], 1);

}

}

}

Map<String, Integer> treeMap = new TreeMap<String, Integer>(hashMap);

System.out.print(treeMap);

return treeMap;

}

}

Số hoàn hảo:

for(int i = 1;i<b;i++)

{

int tong=0;

for(int j=1; j<i;j++)

{

if(i%j==0) tong+=j;

}

if(i==tong)

{

System.out.println(i);

}

}

Hoặc

int KTSHH(int n)

{

int s = 0;

for (int i = 1; i < n; i++)

if (n%i == 0)

s += i;

if (s == n)

return 1; // đúng trả về 1

else

return 0; // sai trả về 0

}

Số nguyên tố:

|  |
| --- |
| public static boolean isPrimeNumber(int n) { |

|  |
| --- |
| // so nguyen n < 2 khong phai la so nguyen to |

|  |
| --- |
| if (n < 2) { |

|  |
| --- |
| return false; |

|  |
| --- |
| } |

|  |
| --- |
| // check so nguyen to khi n >= 2 |

|  |
| --- |
| int squareRoot = (int) Math.sqrt(n); |

|  |
| --- |
| for (int i = 2; i <= squareRoot; i++) { |

|  |
| --- |
| if (n % i == 0) { |

|  |
| --- |
| return false; |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
| return true; |