TÀI LIỆU THỰC TẬP LẬP TRÌNH MẠNG: LAB 09

DANH MỤC THUẬT NGỮ TIẾNG ANH

Từ	Nghĩa của từ
abstract	Trừu tượng
break	Dừng vòng lặp
catch	Từ khóa đầu của một khối bắt ngoại lệ
continue	Bỏ qua phần cuối vòng lặp, tiếp tục sang bước tiếp theo
default	Giá trị mặc định của phương thức switch()
extends	Kế thừa
final	Một hằng số, phương thức hay một lớp không được ghi đè
finally	Một phần của khối xử lý ngoại lệ try luôn được thực hiện
implements	Thực hiện giao diện
import	Khai báo một gói thư viện
instanceof	Kiểm tra một đối tượng là một thể hiện của lớp
interface	Giao diện
new	Tạo một đối tượng mới của lớp
null	Tham chiếu rỗng
package	Gói
private	Tiền tố chỉ được truy cập bởi phương thức của lớp
protected	Tiền tố được truy cập bởi phương thức của lớp, lớp con của và các lớp khác trong cùng một gói
public	Tiền tố có thể được truy cập bởi phương thức của tất cả các lớp
return	Trả về của một phương thức
super	Gọi phương thức của lớp cha
synchronized	Đồng bộ
this	Tham chiếu đến đối tượng hiện tại

DANH MỤC CHỮ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
URL	Uniform Resource Locator
CSDL	Cơ Sở Dữ Liệu
JDBC	Java Database Connectivity
CNTT	Công Nghệ Thông Tin
HĐH	Hệ Điều Hành
MVC	Model-View-Control
DNS	Domain Name System
API	Application Programming Interface
FTP	File Transfer Protocol
JDK	Java Development Kit
GB	GigaByte
UCLN	Ước Chung Lớn Nhất
BCNN	Bội Chung Nhỏ Nhất
RAM	Random Access Memory
RMI	Remote Method Invocation
JVM	Java Virtual Machine
NIC	Network Interface Card
ĐH KTKT CN	Đại học Kinh tế Kỹ thuật Công nghiệp

LỜI NÓI ĐẦU

Ngày nay do nhu cầu thực tế và do sự phát triển mạnh mẽ của nhiều công nghệ tích hợp, dẫn đến các chương trình ứng dụng hầu hết đều có khả năng thực hiện trên môi trường mạng. Ngôn ngữ JAVA là ngôn ngữ phù hợp để viết các ứng dụng mạng. So với lập trình thông thường, lập trình mạng đòi hỏi người lập trình hiểu biết và có kỹ năng tốt để viết các chương trình giao tiếp và trao đổi dữ liệu giữa các máy tính với nhau.
Để hỗ trợ sinh viên chuyên ngành CNTT trong nhà trường tiếp cận với kỹ thuật lập trình mới này, tiếp theo cuốn tài liệu học tập lý thuyết "**Công nghệ JAVA**", chúng tôi xây dựng cuốn "**Bài tập lập trình mạng**", nhằm cung cấp cho sinh viên những kiến thức và kỹ thuật cơ bản nhất để phát triển các chương trình ứng dụng mạng, thông qua các dạng bài tập từ cơ bản đến nâng cao qua các chủ đề: lập trình cơ bản, lập trình hướng đối tượng, lập trình CSDL JDBC, lập trình mạng dùng socket, lập trình phân tán với RMI. Sinh viên sẽ thực hiện các bài thực hành này trên phòng máy nhà trường.

Nội dung cuốn tài liệu bao gồm 12 bài lab chia thành các chủ đề khác nhau. Trong mỗi chủ đề chúng tôi đưa ra tóm tắt lý thuyết, bài tập mẫu, sau đó là bài tập tương tự, và bài tập tổng hợp. Kết quả qua những bài lab, sinh viên được rèn và thành thạo các kỹ năng lập trình hướng đối tượng, lập trình CSDL, lập trình với giao thức truyền thông có sẵn và khả năng tích hợp trong các ứng dụng khác nhau, nhất là các giao thức truyền thông thời gian thực, từ đó sinh viên có thể viết được các phần mềm quản lý theo mô hình MVC, xây dựng được các ứng dụng mạng, các ứng dụng tích hợp và triệu gọi lẫn nhau trên mạng Intranet (mạng cục bộ), mạng Internet (mạng toàn cầu), các hệ thống xử lý truy xuất dữ liệu phân tán hoàn chỉnh. Nội dung biên soạn phù hợp với chuẩn đầu ra của ngành CNTT và ngành mạng máy tính và truyền thông dữ liệu về kỹ năng và kiến thức. Sau khi học xong học phần này sinh viên có thể viết phần mềm quản lý, truyền thông.

Chúng tôi xin chân thành cảm ơn Thầy Nguyễn Hoàng Chiến, phó chủ nhiệm khoa, phụ trách khoa CNTT trường ĐH KTKT CN cùng với các đồng nghiệp đã đóng góp ý kiến cho cuốn tài liệu này. Vì tài liệu được biên soạn lần đầu, chúng tôi đã cố gắng hoàn chỉnh, song không tránh khỏi thiếu sót. Rất mong nhận được sự góp ý của bạn đọc để tài liệu học tập được hoàn thiện hơn.

Xin trân trọng cảm ơn!

Nhóm tác giả

LAB 9. LẬP TRÌNH SERVER PHỤC VỤ NHIỀU CLIENT [4,6,10]

A. MỤC TIÊU

Trang bị cho sinh viên kỹ thuật lập trình các ứng dụng một Server có khả năng đáp ứng nhiều Client.

Kết hợp lập trình giao diện với java swing, xử lý đa luồng, lập trình Client-Server theo cơ chế TCP viết các ứng dụng đa luồng trên mạng.

B. NỘI DUNG

- Viết các ứng dụng Client-Server kết hợp sử dụng luồng.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

- Viết được chương trình kết nối TCP thực hiện trao đổi dữ liệu giữa Client và Server.
- Xây dựng ứng dụng chat LAN.

D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

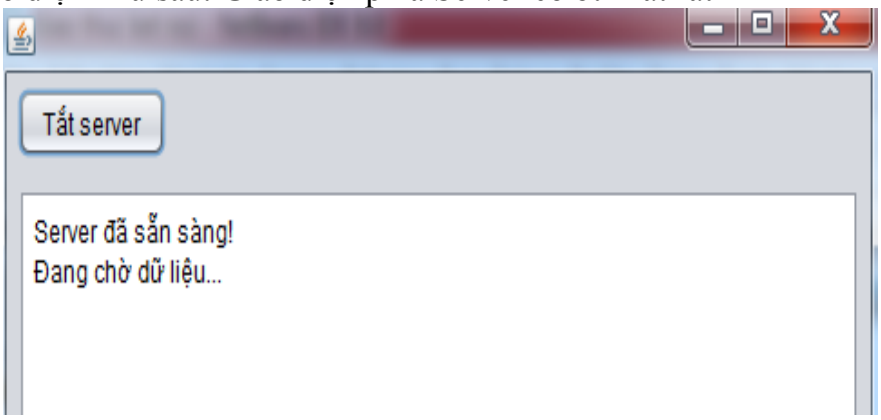
- Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8.

E. HƯỚNG DẪN

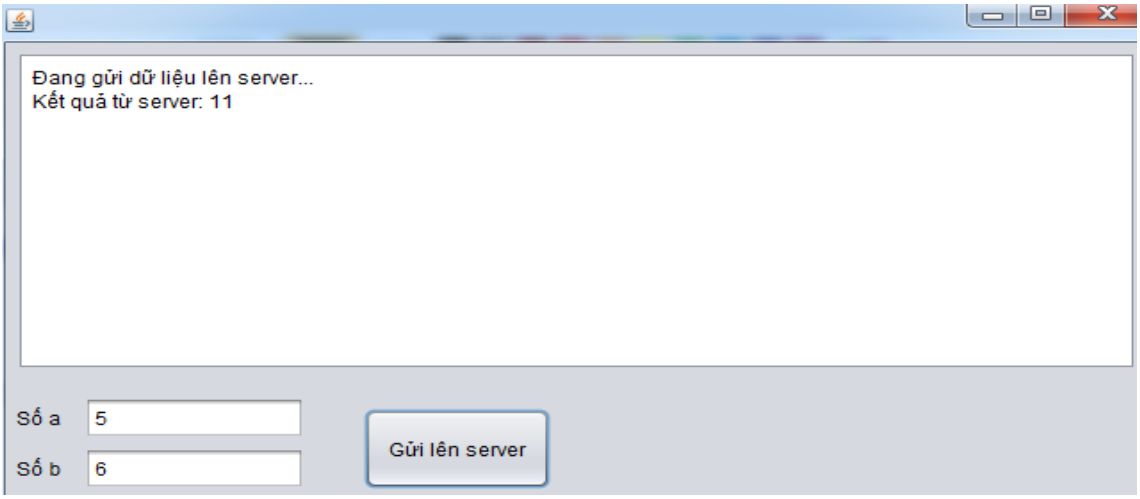
Để cài đặt chương trình Server TCP phục vụ nhiều Client đồng thời, trong lập trình mạng sử dụng kỹ thuật phổ biến đó là: Xây dựng chương trình đa luồng (đa tiểu trình). Chương trình Server kiểu này sẽ sinh ra một luồng mới mỗi khi có một Client gửi yêu cầu tới đòi phục vụ.

Chạy chương trình Server trên một máy và Client trên máy khác nếu hệ thống có mạng. Nếu thử nghiệm trên một máy đơn ta có thể mô phỏng mô hình khách/chủ bằng cách mở hai cửa sổ dòng lệnh DOS.

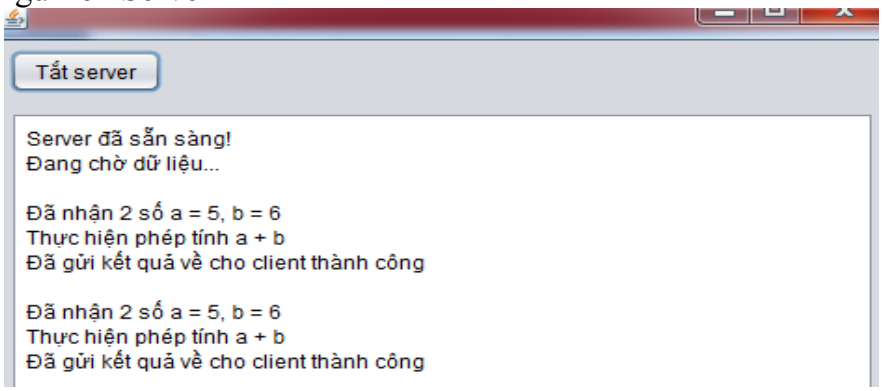
Bài 1. Viết chương trình minh họa giao tiếp Client-Server sử dụng TCP, xử lý tính tổng hai số có giao diện như sau: Giao diện phía Server có btnBatTat



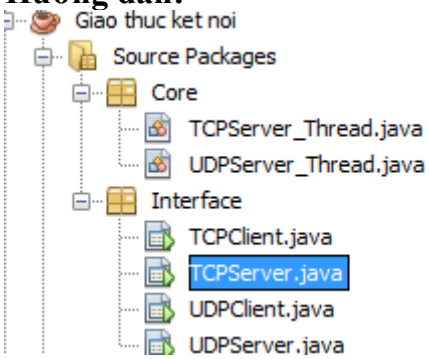
Phía Client: có btnSendToServer



Xử lý sự kiện gửi lên Server



Hướng dẫn:



Bước 1: Tạo file TCPServer

```
TCPServer_Thread mThreadServer;
boolean mTurnOn = false;
private void btnBatTatActionPerformed(java.awt.event.ActionEvent evt) {
        if(mTurnOn==false)
        {
            mThreadServer= new TCPServer_Thread(txaStatus);
            mThreadServer.start();
            mTurnOn = true;
            btnBatTat.setText("Tắt Server");
        }
   else
```

```
        {
            mThreadServer.StopServer();
            mTurnOn = false;
            btnBatTat.setText("Bật Server");
            txaStatus.append("Đã tắt Server\n\n");
        }
    }
```

Bước 2: File TCPClient

```
private void btnSendToServerActionPerformed(java.awt.event.ActionEvent
evt) {
    txaStatus.append("Đang gửi dữ liệu lên Server...\n");
        try
        {
            Socket ClientSocket = new Socket("Localhost", 1234);
            //tao luong nhan du lieu tu Server
            DataInputStream inFromServer = new
DataInputStream(ClientSocket.getInputStream());
            //tao luong gui du lieu len Server
            DataOutputStream outToServer = new
DataOutputStream(ClientSocket.getOutputStream());
            // nhap du lieu tu ban phim gui len Server
            outToServer.writeBytes(txtSoA.getText()+'\n');
            outToServer.writeBytes(txtSoB.getText()+'\n');
 txaStatus.append("Kết quả từ Server:"+inFromServer.readLine() + "\n\n");
            //dong luong gui du lieu len Server
            outToServer.close();
            //dong luong nhan du lieu tu Server
            inFromServer.close();
            //dong socket Client
            ClientSocket.close();
        }catch(UnknownHostException e)
        {
            txaStatus.append("Server Not Found\n\n");
        }catch(IOException e)
        {
            txaStatus.append("Cannot make a connection\n\n");
        }
    }
private void SoABKeyTyped(java.awt.event.KeyEvent evt) {
        JTextField txt =  (JTextField)evt.getSource();
```

```
        String after = txt.getText() + evt.getKeyChar();
        try
        {
            Integer.parseInt(after);
        }catch(NumberFormatException ex)
        {
            evt.consume();
        }
    }
```

Bước 3: Tạo file TCPServer_Thread: Lớp này khởi động Server trong 1 thread khác để tránh giao diện bị treo khi Server đang đợi kết nối.

```
public class TCPServer_Thread extends Thread
{
    ServerSocket mServer;
    JTextArea mTxaStatus;    //JTextArea để lưu status của Server
    public TCPServer_Thread(JTextArea txaStatus)
    {
        mTxaStatus = txaStatus;
    }
    @Override
    public void run()
    {
        try
         {
             String so1,so2,so3;
             int tong;
             mServer = new ServerSocket(1234);
mTxaStatus.append("Server đã sẵn sàng!\nĐang chờ dữ liệu...\n\n");
             while(true)
             {
                 Socket connectionSocket = mServer.accept();
                 DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
                 DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
                 so1 = inFromClient.readLine();
                 so2 = inFromClient.readLine();
       mTxaStatus.append("Đã nhận 2 số a = " + so1 +", b = "+ so2 +"\n");
                 int a = Integer.parseInt(so1);
                 int b = Integer.parseInt(so2);
```

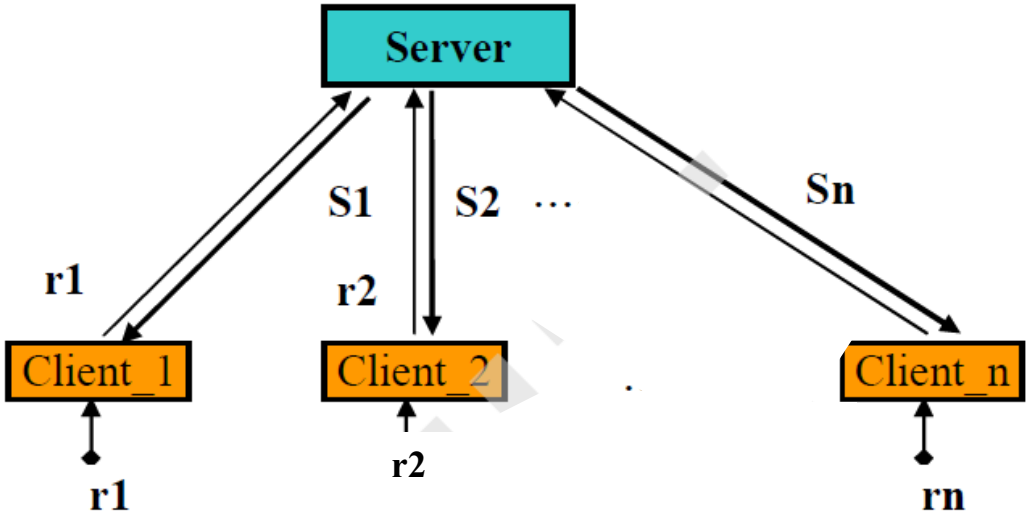
```
                tong = a + b;
                so3 = String.valueOf(tong);
                mTxaStatus.append("Thực hiện phép tính a + b\n");
         outToClient.writeBytes(so3+'\n');
        mTxaStatus.append("Đã gửi kết quả về cho Client thành công\n\n");
             }
         }
         catch(Exception ex)
         {
  Logger.getLogger(TCPServer_Thread.class.getName()).log(Level.SEVERE,
null, ex);
             mTxaStatus.append("Server đã xảy ra lỗi\n");
         }
    }
    public void StopServer()
            {
        super.stop();
        try {
            mServer.close();
          } catch (IOException ex) {
 Logger.getLogger(TCPServer_Thread.class.getName()).log(Level.SEVERE,
          null, ex);
        }
    }
}
```

Bài 2.

Viết chương trình Server phục vụ nhiều Client đồng thời sử dụng giao thức truyền thông TCP. Chương trình cho phép nhận bán kính đường tròn gửi đến từ các Client, tính diện tích hình tròn, hiển thị tên, địa chỉ IP, số cổng, bán kính r, diện tích của Client tương ứng. Sau đó trả kết quả về cho Client.

Hướng dẫn:

Để viết chương trình này sử dụng kỹ thuật đa luồng trong Java. Mỗi khi một chương trình Client gửi yêu cầu kết nối đến Server, Server sẽ sinh ra một luồng mới để phục vụ kết nối đó. Sau khi phục vụ xong kết nối nào thì luồng đó được giải phóng. Mô hình xây dựng chương trình thể hiện như sau:



Bước 1: Tạo **class** areaClient
Chương trình Client thực hiện các công việc sau:

1. Gửi kết nối tới Server
2. Nhập bán kính r từ bàn phím
3. Gửi bán kính tới Server
4. Nhận kết quả trả về và hiển thị
5. Kết thúc chương trình

```
class areaClient{
public static void main(String[] args)
{
   Socket cl=null;
   BufferedReader inp=null;//luong nhap
   PrintWriter outp=null;//luong xuat
   BufferedReader key=null;//luong nhap tu ban phim
   String ipServer= "127.0.0.1";//Chuoi dia chi Server
   int portServer=3456; //dia chi cong Server
   String r; //Tao socket va ket noi toi Server
  try{
      cl=new Socket(ipServer,portServer);
     //tao luong nha/xuatp kieu ky tu cho socket
 inp=new BufferedReader(new InputStreamReader(cl.getInputStream()));
 outp=new PrintWriter(cl,getOutputStream(),true);
//tao luong nhap tu ban phim
  key=new BufferedReader(new InputStreamReader(System.in));
  //Nhap ban kinh r tu ban phim
  System.out.print("r=");
  r=key.readLine().trim();
  //gui r toi Server
```

```
  outp.println(r);
 //Nhan dien tich tra ve tu Server va hien thi
 System.out.println("Area:"+inp.readLine());
 //ket thuc chuong trinh
 if(inp!=null)
 inp.close();
 if(key!=null)
 key.close();
  if(outp!=null)
  outp.close();
  if(cl!=null)
 cl.close();
}
cacth(IOException e)
    {
     System.out.println(e);
    }
  }
}
```

Bước 2: Tạo chương trình phía Server

Chương trình Server phục vụ nhiều Client thực hiện các công việc sau:

1. Khởi tạo đối tượng ServerSocket và nghe tại số cổng 3456.
2. Thực hiện lặp lại các công việc sau:
3. Nhận kết nối mới, tạo socket mới
4. Phát sinh một luồng mới và nhận socket
5. Nhận bán kính gửi tới từ Client
6. Tính diện tích
7. Hiển thị số thứ tự luồng, tên, địa chỉ IP, số cổng, bán kính r, diện tích của Client
8. Gửi diện tích về cho Client
9. Kết thúc luồng

Tạo class NewThread

```
import java.io.*;
import java.net.*;
//Khai báo lớp NewThread cho phép tạo ra luồng mới
class NewThread extends Thread
{
  private int count;
  private Socket cl=null;
  private BufferedReader inp=null;//luong nhap
  private PrintWriter outp=null;//luong xuat
```

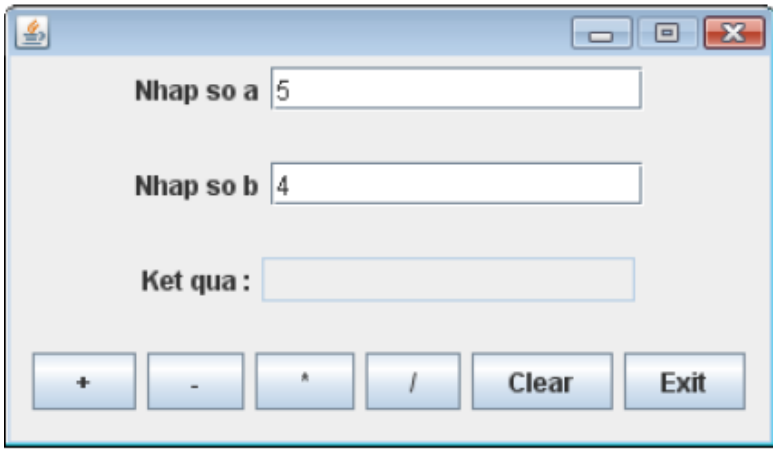
```
NewThread(Socket cl, int count)
{
  super();//Truy xuất cấu tử lớp Thread
  this.cl=cl;
  this.count=count;
  start();
}
//cai dat phuong thuc run-Luong moi
public void run()
{
  try{
        //tao luong nhap /xuat cho socket cl
inp=new BufferedReader(new InputStreamReader(cl.getInputStream()));
        outp=new PrintWriter(cl.getOutputStream(),true);
        //Doc ban kinh gui toi tu Client
        double r=Double.parseDouble(inp.readLine().trim());
        // lay dia chi Client
        InetAddress addrClient=cl.getInetAddress();
        //lay so cong phia Client
        int portClient=cl.getPort();
        //Tinh dien tich
        double area=3.14*r*r;
        //Hien thi
System.out.println("Luong thu:"+count, Client:"+addrClient.getHostName()+
", ip:"+addrClient.getHostAddress()+",port:"+portClient+",
r="+r+",area:"+area);
        //Gui dien tich ve cho Client tuong ung
        outp.println(area);
        //ket thuc luong
        inp.close();
        outp.close();
        cl.close();
   }
    catch(IOException e)
      {
        System.out.println(e);
      }
   }
}
```

Bước 3: Tạo **class** AreaThreadServer

```
class AreaThreadServer{
public static void main(String[] args)
{
//Khai bao bien
    int count;
    ServerSocket svr=null;
    Socket cl=null;
    int portServer=3456;
   try{
       svr=new ServerSocket(portServer);
      count=0;
     while(true){
   cl=svr.accept();
    new NewThread(cl, count);
  count++;
  }
}
  catch(IOException e)
   {
    System.out.println(e);
    }
  }
}
```

Bài 3.

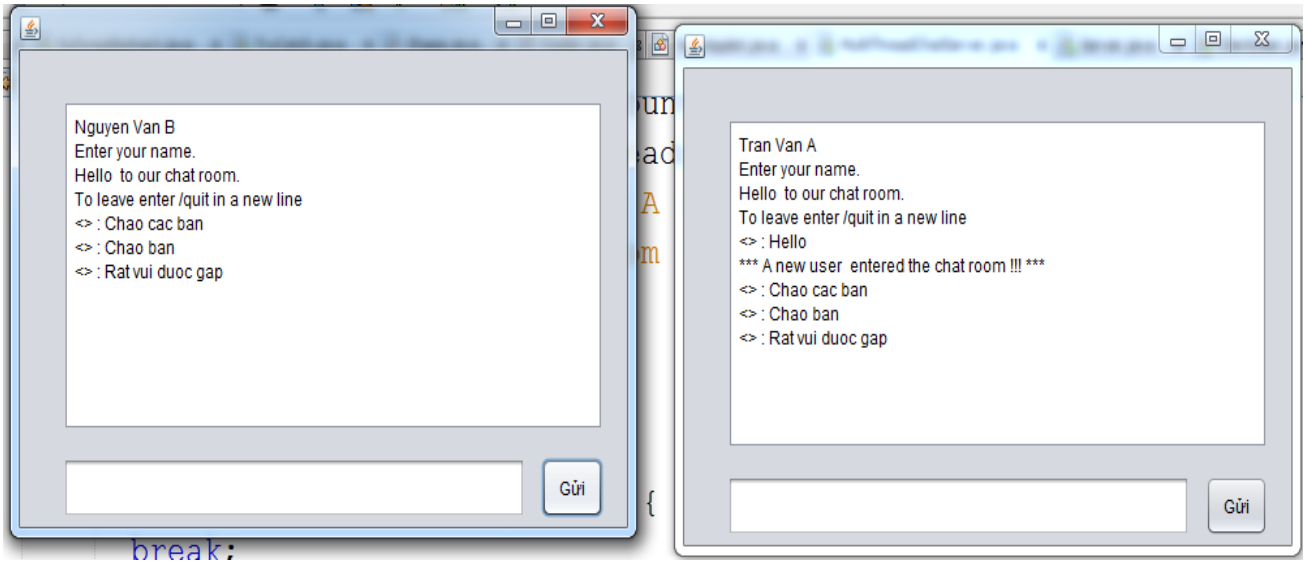
Viết 1 chương trình Calculator_server nhận 1 biểu thức gồm 2 chữ số và 1 phép toán sau đó thực thi biểu thức này và gửi kết quả lại cho client.
Sửa chữa chương trình cho phép nhiều client kết nối cùng lúc.
Phía client, viết giao diện gồm 2 JtextField cho việc nhập số, 1 JLabel cho việc xuất kết quả. Các nút Cộng, trừ, nhân, chia, clear và thoát.
Giao diện client như sau:



Hướng dẫn: Sinh viên thực hiện tương tự bài 2.

Bài 4. Viết ứng dụng chat đa người dùng sử dụng TCP và luồng.

Yêu cầu: Giao diện các Client tham gia chat như sau:



Hướng dẫn: Kết hợp lập trình java swing và xử lý đa luồng trong java để giải quyết.

Bước 1: Tạo **class** MultiThreadChatServer sử dụng cổng 2222, đáp ứng đồng thời 10 Client. (file thực thi)

```
public class MultiThreadChatServer extends javax.swing.JFrame{
  private static ServerSocket ServerSocket = null;
  private static Socket ClientSocket = null;
  private static final int maxClientsCount = 10;
  private static final ClientThread[] threads = new
ClientThread[maxClientsCount];
  public static void main(String args[]) {
    int portNumber = 2222;
    if (args.length < 1) {
    System.out.println("Usage: java MultiThreadChatServer <portNumber>\n"
```

```
            + "Now using port number=" + portNumber);
    } else {
      portNumber = Integer.valueOf(args[0]).intValue();
    }
    try {
      ServerSocket = new ServerSocket(portNumber);
    } catch (IOException e) {
      System.out.println(e);
    }
    while (true) {
      try {
        ClientSocket = ServerSocket.accept();
        int i = 0;
        for (i = 0; i < maxClientsCount; i++) {
          if (threads[i] == null) {
          (threads[i] = new ClientThread(ClientSocket, threads)).start();
            break;
          }
        }
        if (i == maxClientsCount) {
  DataOutputStream os = new
DataOutputStream(ClientSocket.getOutputStream());
          os.writeUTF("Server too busy. Try later.");
          os.close();
          ClientSocket.close();
        }
      } catch (IOException e) {
        System.out.println(e);
      }
    }
  }
}
```

Bước 2: Tạo **class** ClientThread. Mỗi Client khi tham gia vào nhóm chat tương ứng với 1 thread.

```
class ClientThread extends Thread {
  private DataInputStream is = null;
  private DataOutputStream os = null;
  private Socket ClientSocket = null;
  private final ClientThread[] threads;
```

```
private int maxClientsCount;
public ClientThread(Socket ClientSocket, ClientThread[] threads) {
  this.ClientSocket = ClientSocket;
  this.threads = threads;
  maxClientsCount = threads.length;
}
public void run() {
  int maxClientsCount = this.maxClientsCount;
  ClientThread[] threads = this.threads;
  try {
    is = new DataInputStream(ClientSocket.getInputStream());
    os = new DataOutputStream(ClientSocket.getOutputStream());
    os.writeUTF("Enter your name.");
    String name = is.readUTF().trim();
    os.writeUTF("Hello " + name
        + " to our chat room.\nTo leave enter /quit in a new line");
    for (int i = 0; i < maxClientsCount; i++) {
      if (threads[i] != null && threads[i] != this) {
          threads[i].os.writeUTF("*** A new user " + name
            + " entered the chat room !!! ***");
      }
    }
    while (true) {
      String line = is.readUTF();
      if (line.startsWith("/quit")) {
        break;
      }
      for (int i = 0; i < maxClientsCount; i++) {
        if (threads[i] != null) {
            System.out.println(line);
          threads[i].os.writeUTF("<" + name + "> : " + line);
        }
      }
    }
    for (int i = 0; i < maxClientsCount; i++) {
      if (threads[i] != null && threads[i] != this) {
        threads[i].os.writeUTF("*** The user " + name
            + " is leaving the chat room !!! ***");
      }
    }
```

```
      os.writeUTF("*** Bye " + name + " ***");
      for (int i = 0; i < maxClientsCount; i++) {
        if (threads[i] == this) {
          threads[i] = null;
        }
      }
      is.close();
      os.close();
      ClientSocket.close();
    } catch (IOException e) {
    }
  }
}
```

Bước 3: Tạo **class** Server (chạy trên Server)

```
 public class Server extends javax.swing.JFrame {
  private static ServerSocket ServerSocket = null;
  private static Socket ClientSocket = null;
  private static final int maxClientsCount = 10;
  private static final ClientThread[] threads = new
ClientThread[maxClientsCount];
      public Server() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    private void initComponents() {
     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE));
        layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );
        pack();
    }
  public static void main(String args[]) {
```

```
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
       javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        }
  java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Server().setVisible(true);
                    int portNumber = 2222;
                    if (args.length < 1) {
     System.out.println("Usage: java MultiThreadChatServer
<portNumber>\n"  + "Now using port number=" + portNumber);
                } else {
                  portNumber = Integer.valueOf(args[0]).intValue();
                }
                try {
                  ServerSocket = new ServerSocket(portNumber);
                } catch (IOException e) {
                  System.out.println(e);
                }
                while (true) {
                  try {
                    ClientSocket = ServerSocket.accept();
                    for (i = 0; i < maxClientsCount; i++) {
```

```
                    if (threads[i] == null) {
       (threads[i] = new ClientThread(ClientSocket, threads)).start();
                      break;
                    }
                  }
                  if (i == maxClientsCount) {
DataOutputStream os = new
DataOutputStream(ClientSocket.getOutputStream());
                    os.writeUTF("Server too busy. Try later.");
                    os.close();
                    ClientSocket.close();
                  }
                } catch (IOException e) {
                  System.out.println(e);
                }
              }
            }
          });
        }
    }
```

Bước 4: Tạo **class** ClientUser minh họa giao diện người dùng (chạy trên Client)

```
  public class ClientUser extends javax.swing.JFrame implements Runnable{
  private static Socket ClientSocket = null;
  private static DataOutputStream os = null;
  private static DataInputStream is = null;
  private static BufferedReader inputLine = null;
      public ClientUser() {
        initComponents();
    }
private void msg_sendActionPerformed(java.awt.event.ActionEvent evt) {
         try{
             os.writeUTF(msg_text.getText().trim());
             msg_text.setText("");
         }catch(Exception e){}
    }
 public static void main(String args[]) {
          try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```

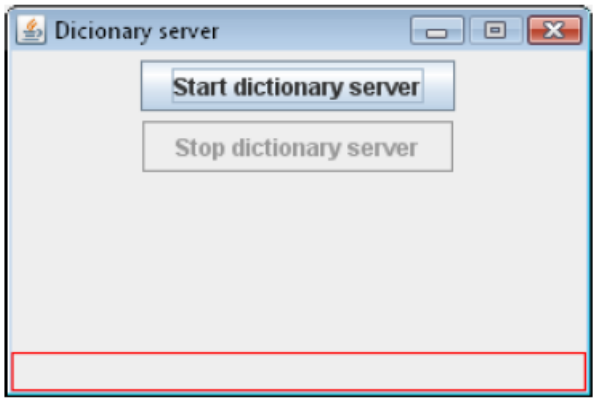
```
                if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        }
java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
               int portNumber = 2222;
               String host = "localhost";
               try {
                  ClientSocket = new Socket(host, portNumber);
                  inputLine = new BufferedReader(new
InputStreamReader(System.in));
             os = new DataOutputStream(ClientSocket.getOutputStream());
             is = new DataInputStream(ClientSocket.getInputStream());
               } catch (UnknownHostException e) {
                 System.err.println("Don't know about host " + host);
               } catch (IOException e) {
System.err.println("Couldn't get I/O for connection to the host "+ host);
               }
                 if (ClientSocket != null && os != null && is != null) {
                     ClientUser c = new ClientUser();
                     c.setVisible(true);
                     new Thread(c).start();
               }
            }
        });
```

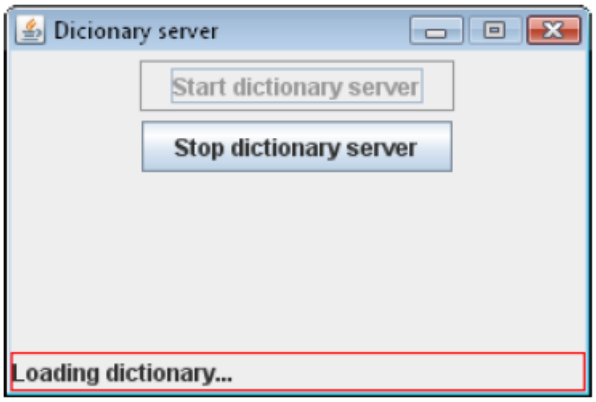
```
    }
    private javax.swing.JScrollPane jScrollPane1;
    private static javax.swing.JTextArea msg_area;
    private javax.swing.JButton msg_send;
    private javax.swing.JTextField msg_text;
    @Override
    public void run() {
         String responseLine;
        try {
              while (true) {
              responseLine = is.readUTF();
              System.out.println(responseLine);
         msg_area.setText(msg_area.getText().trim()+ "\n"+ responseLine);
          }
        } catch (IOException e) {
          System.err.println("IOException:  " + e);
        }
    }
```

Bài 5. Tổng hợp

Viết chương trình từ điển cho phép tra từ qua mạng, việc tra từ này phải đảm bảo nhiều người có thể thực hiện cùng một lúc. Thiết kế chương trình gồm 2 phần: Phần server và client.

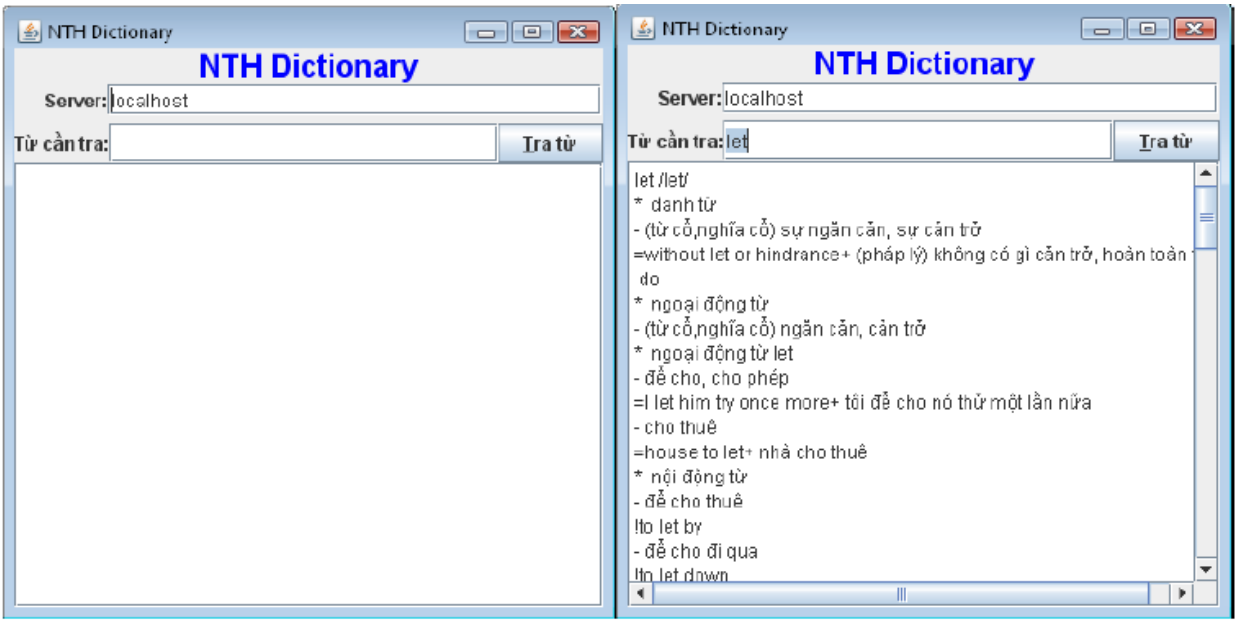
Phần server: Chạy trên server có giao diện như sau:





Khi người dùng start server từ điển, server này sẽ lắng nghe trên cổng 2520 và sẽ nhận đầu vào là từ cần tra sau đó thực hiện tra từ và trả kết quả về cho client hoặc là 1 đối tượng từ đã tra trong trường hợp tra cứu thành công, hoặc là null nếu từ không tồn tại.

Phần Client: được triển khai ở phía client, có giao diện như sau



Hướng dẫn:

Sử dụng đa luồng kết hợp với lập trình mạng sử dụng TCP, sinh viên triển khai tương tự bài 3.

TÀI LIỆU THAM KHẢO

[1]. Cay S. Horstmann, *Core Java Volum I - Fundamentals, Tenth Edition*, NewYork : Prentice Hall, 2016.

[2]. Cay S. Horstmann. *Core Java Volum II - Advanced Features, Tenth Edition*, New York : Prentice Hall, 2017.

[3].Eng.haneen Ei-masry, *Java database connection*, Islamic University of Gaza Faculty of Engineering Department of Computer Engineering ECOM 4113: DataBase Lab, 2014.

[4]. Angelos Stavrou, *Advanced Network Programming Lab using Java*, Network Security, ISA 656, Angelos Stavrou.

[5]. Marenglen Biba, Ph.D, *Manual for Lab practices, Remote Method Invocation Three Tier Application with a Database Server*, Department of Comsputer Science, University of New York.

[6].Elliotte Rusty Harold, *Java Network Programming, Fourth Edition*, O'Reilly Media, 2013.

[7]. Đoàn Văn Ban, *Lập trình hướng đối tượng với JAVA*, Nhà xuất bản Khoa học và Kỹ thuật, 2005.

[8]. ThS. Dương Thành Phết, *Bài tập thực hành Chuyên đề 1 CNPM- Java*, Khoa CNTT- Trường ĐH Công nghệ TP.HCM.

[9]. https://www.oracle.com/technetwork/java/socket-140484.html#

[10]. https://personales.unican.es/corcuerp/java/Labs/LAB_22.htm

[11]. http://www.nrcmec.org/pdf/Manuals/CSE/student/2-2%20java16-17.pdf

[12]. http://cse.mait.ac.in/pdf/LAB%20MANUAL/JAVA.pdf

[13]. https://www.academia.edu/35283541/Bài tập môn lập trình hướng đối tượng