

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC CÔNG NGHỆ TP.HCM

THỰC HÀNH
LẬP TRÌNH MẠNG

Biên soạn:

www.hutech.edu.vn

THỰC HÀNH LẬP TRÌNH MẠNG

Ấn bản 2021

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập:

tailieuhoctap@hutech.edu.vn

MỤC LỤC

MỤC LỤC	I
HƯỚNG DẪN.....	V
BÀI 1. LẬP TRÌNH KẾT NỐI CƠ SỞ DỮ LIỆU.....	1
1.1 BÀI TẬP THỰC HÀNH:	ERROR! BOOKMARK NOT DEFINED.
1.1.1 Yêu cầu:	Error! Bookmark not defined.
1.1.2 Hướng dẫn:	1
1.1.3 Kết nối CSDL mysql:	1
1.1.4 Kết nối với cơ sở dữ liệu SQL Server:	11
1.2 BÀI TẬP THAM KHẢO:	ERROR! BOOKMARK NOT DEFINED.
1.2.1 Bài tập tham khảo 01:	Error! Bookmark not defined.
1.2.2 Bài tập tham khảo 02:	Error! Bookmark not defined.
BÀI 2. THAO TÁC VỚI FILE	Error! Bookmark not defined.
2.1 THAO TÁC VỚI THƯ MỤC:	17
2.1.1 Yêu cầu:	17
2.1.2 Hướng dẫn:	17
2.2 THAO TÁC VỚI FILE:	20
2.2.1 Yêu cầu:	20
2.2.2 Hướng dẫn:	20
2.3 BÀI TẬP THAM KHẢO:	22
2.3.1 Bài tập tham khảo 01:	Error! Bookmark not defined.
2.3.2 Bài tập tham khảo 02:	Error! Bookmark not defined.
BÀI 3. INETADDRESS	24
3.1 INETADDRESS:	24
3.1.1 Lớp INETADDRESS:	24

3.1.2 Tạo đối tượng INETADDRESS:	25
3.2 BÀI TẬP THỰC HÀNH:	25
3.2.1 Bài thực hành 01:.....	25
3.2.2 Bài thực hành 02:.....	26
3.2.3 Bài thực hành 03:.....	27
3.2.4 Bài thực hành 04:.....	28
3.2.5 Bài thực hành 05:.....	29
3.2.6 Bài thực hành 06:.....	30
BÀI 4. XỬ LÝ LUỒNG.....	32
4.1 LUỒNG TRONG JAVA:	32
4.1.1 Giới thiệu Thread:.....	32
4.1.2 Đa nhiệm (Multitasking):	32
4.1.3 Ưu điểm của đa luồng:	33
4.1.4 Nhược điểm của đa luồng:	33
4.2 BÀI TẬP THỰC HÀNH:	34
4.2.1 Bài thực hành 01:.....	34
4.2.2 Bài thực hành 02:.....	37
4.3 BÀI TẬP THAM KHẢO:	40
4.3.1 Bài tập tham khảo 01:.....	40
4.3.2 Bài tập tham khảo 02:.....	40
4.3.3 Bài tập tham khảo 03:.....	40
BÀI 5. TCP Socket, FTP.....	41
5.1 TCP SOCKET TRONG JAVA:	41
5.1.1 Socket:	41
5.1.2 Lập trình TCP Socket trong Java:	42
5.2 FTP:.....	43
5.2.1 Giao thức FTP:	43

5.2.2 Hoạt động của FTP:	44
5.2.3 Các phương thức truyền dữ liệu trong FTP:	44
5.3 BÀI TẬP THỰC HÀNH:	45
5.3.1 Bài thực hành 01:	45
5.3.2 Bài thực hành 02:	49
5.3.3 Bài thực hành 03:	53
5.3.4 Bài thực hành 04:	56
5.3.5 Bài thực hành 05:	57
5.3.6 Bài thực hành 06:	57
5.3.7 Bài thực hành 07:	Error! Bookmark not defined.
5.3.8 Bài thực hành 08:	Error! Bookmark not defined.
5.3.9 Bài thực hành 09:	Error! Bookmark not defined.
5.3.10 Bài thực hành 10:.....	Error! Bookmark not defined.
BÀI 6. UDP SOCKET.....	72
6.1 UDP SOCKET:.....	72
6.1.1 Lập trình socket với UDP:	72
6.1.2 Hoạt động của UDP Socket:	73
6.1.3 So sánh UDP và TCP:.....	74
6.2 BÀI TẬP THỰC HÀNH:	75
6.2.1 Bài thực hành 01:	75
6.2.2 Bài thực hành 02:	78
6.2.3 Bài thực hành 03:	81
6.2.4 Bài thực hành 04:	83
6.3 BÀI TẬP THAM KHẢO:.....	87
6.3.1 Bài tập tham khảo 01:	87
6.3.2 Bài tập tham khảo 02:	88
BÀI 7. LẬP TRÌNH ĐỐI TƯỢNG PHÂN TÁN (RMI)	Error! Bookmark not defined.

7.1 JAVA RMI:	90
7.1.1 Khái niệm:.....	90
7.1.2 Đặc tính của RMI:.....	91
7.1.3 Kiến trúc RMI:.....	91
7.2 BÀI TẬP THỰC HÀNH:	92
7.2.1 Bài thực hành 01:.....	92
7.2.2 Bài thực hành 02:.....	97
TÀI LIỆU THAM KHẢO	98

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Môn học này cung cấp các kiến thức cơ bản cho sinh viên về lập trình trên môi trường mạng; các kiến thức về xử lý file, kết nối và thao tác với hệ quản trị cơ sở dữ liệu; các kiến thức về TCP Socket, UDP Socket; các kiến thức về giao thức truyền dẫn file FTP; lập trình đối tượng phân tán (RMI) và áp dụng vào các bài tập thực tế.

NỘI DUNG MÔN HỌC

Bài 1: LẬP TRÌNH KẾT NỐI CƠ SỞ DỮ LIỆU

Sau khi thực hành xong bài này, sinh viên sẽ có khả năng lập trình các ứng dụng có kết nối cơ sở dữ liệu thông qua môi trường mạng. Sinh viên được ôn tập các kiến thức về lập trình giao diện bằng ngôn ngữ Java và các kiến thức về hệ quản trị cơ sở dữ liệu mySQL và SQL Server.

Bài 2: LUỒNG NHẬP XUẤT

Sau khi thực hành xong bài này, sinh viên sẽ có khả năng thao tác với file trong việc xây dựng các ứng dụng mạng, đặc biệt là các ứng dụng truyền nhận dữ liệu. nội dung của bài thực hành này gồm 2 phần chính: thao tác với thư mục và thao tác với file (đọc/viết file nhị phân và file văn bản).

Bài 3: INETADDRESS

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về INETADDRESS, cách tạo đối tượng INETADDRESS trong Java và áp dụng vào những bài tập thực tế.

Bài 4: XỬ LÝ LUỒNG

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các khái niệm cơ bản về luồng (Thread), đa nhiệm (Multitasking), đa luồng (MultiThread), cách tạo ra luồng trong Java và ứng dụng vào các bài tập thực tế.

Bài 5: TCP SOCKET, FTP

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về TCP Socket, FTP, mô hình kết nối client-server; biết cách khởi tạo, xây dựng các ứng dụng kiểu client-server thông qua 2 giao thức TCP, FTP bằng ngôn ngữ Java.

Bài 6: UDP SOCKET

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về UDP Socket, mô hình kết nối client-server; biết cách khởi tạo, xây dựng các ứng dụng kiểu client-server thông qua giao thức UDP bằng ngôn ngữ Java.

Bài 7: LẬP TRÌNH RMI

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về lập trình đối tượng phân tán (RMI - Remote Method Invocation) trong Java và áp dụng vào các bài tập thực tế.

KIẾN THỨC TIỀN ĐỀ

Sinh viên có kiến thức khái quát về mạng máy tính và lập trình cơ bản.

YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi lên lớp và làm bài tập đầy đủ ở nhà.

CÁCH TIẾP CẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần đọc trước các nội dung chưa được học trên lớp; tham gia đều đặn và tích cực trên lớp; hiểu các khái niệm, tính chất và ví dụ tại lớp học. Sau khi học xong, cần ôn lại bài đã học, làm các bài tập và câu hỏi. Tìm đọc thêm các tài liệu khác liên quan đến bài học và làm thêm bài tập.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

- Điểm quá trình: 50%. Điểm kiểm tra thường xuyên trong quá trình học tập. Điểm kiểm tra giữa học phần, điểm làm bài tập trên lớp, hoặc điểm chuyên cần.
- Điểm thi: 50%. Hình thức bài thi thực hành trong 90 phút, không được mang tài liệu vào phòng thi. Nội dung gồm các câu hỏi và bài tập tương tự như các câu hỏi và bài tập về nhà.

BÀI 1. LẬP TRÌNH KẾT NỐI CƠ SỞ DỮ LIỆU

1.1 BÀI TẬP THỰC HÀNH

Viết chương trình quản lý danh sách người sử dụng. Thông tin mỗi người sử dụng gồm có: họ tên, mật khẩu, đường dẫn thư mục, quyền truy xuất (đọc, viết, cả hai).

Chương trình có thể thực hiện các chức năng sau:

- Cho phép user đăng ký vào hệ thống.
- Cho phép user đăng nhập vào hệ thống.

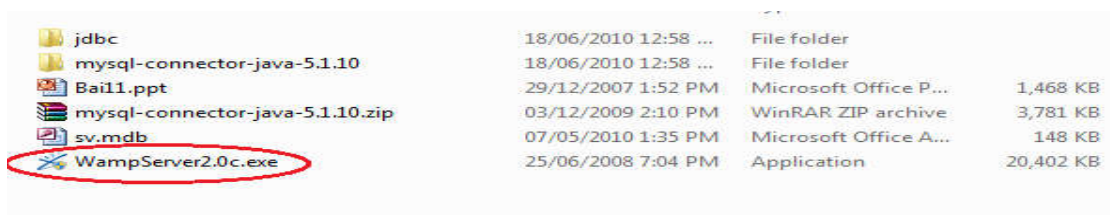
1.2 HƯỚNG DẪN

Hiện nay, có rất nhiều hệ quản trị CSDL đang được sử dụng rộng rãi, trong phần lab này ta cùng tìm hiểu hai hệ quản trị CSDL được sử dụng phổ biến hiện nay là: mysql và SQL Server.

1.2.1 Kết nối CSDL mysql

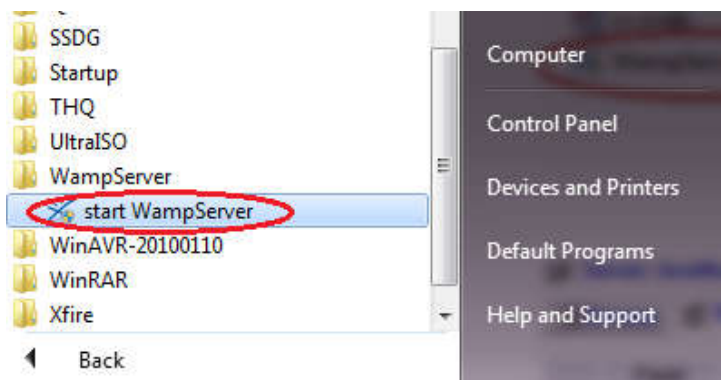
1. Cài đặt mysql thông qua gói WampServer và tạo CSDL:

a. Bước 1: Cài đặt wampserver

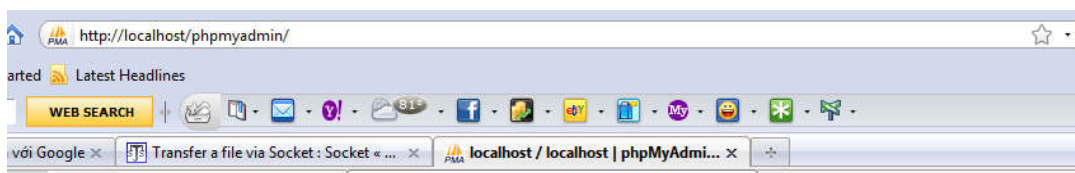


jdbc	18/06/2010 12:58 ...	File folder	
mysql-connector-java-5.1.10	18/06/2010 12:58 ...	File folder	
Bai11.ppt	29/12/2007 1:52 PM	Microsoft Office P...	1,468 KB
mysql-connector-java-5.1.10.zip	03/12/2009 2:10 PM	WinRAR ZIP archive	3,781 KB
sv.mdb	07/05/2010 1:35 PM	Microsoft Office A...	148 KB
WampServer2.0c.exe	25/06/2008 7:04 PM	Application	20,402 KB

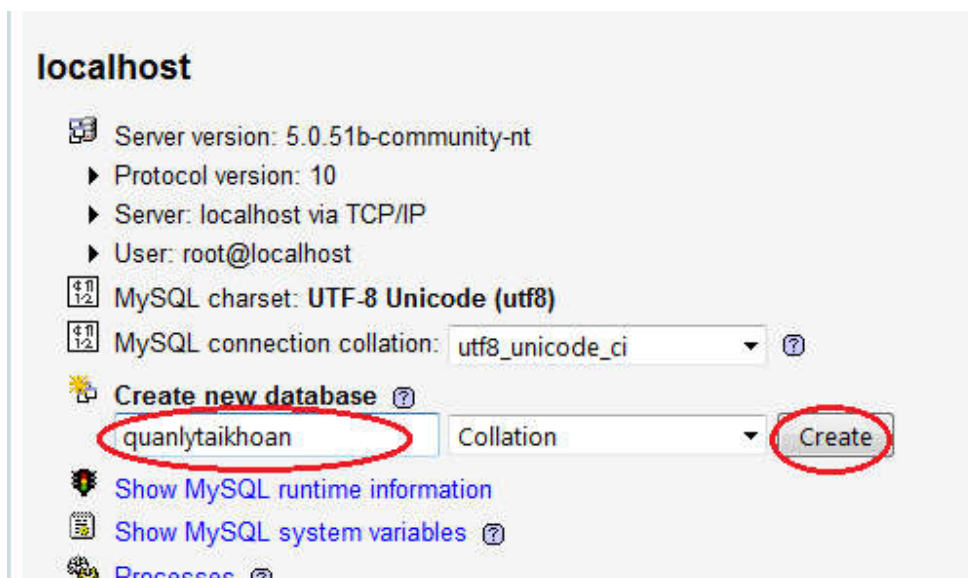
b. Bước 2: Sau khi cài xong WampServer ta vào Start → Program file → WampServer → start WampServer.



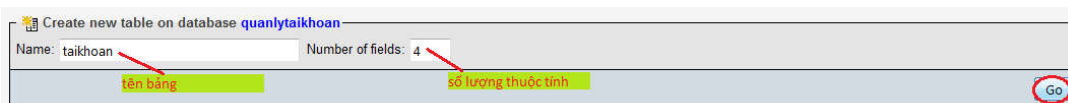
c. Bước 3: Vào trình duyệt Firefox gõ địa chỉ của trang phpmyadmin.



d. Bước 4: Tạo database quanlytaikhoan.



e. Bước 5: Vào Database quanlytaikhoan, tạo bảng taikhoan.



f. Bước 5: Tạo các thuộc tính cho bảng dữ liệu.

Server: localhost Database: quanlytaikhoan Table: taikhoan

Field	Type	Length/Values	Collation	Attributes
username	VARCHAR	50		
password	VARCHAR	50		
path	VARCHAR	200		
per	INT			

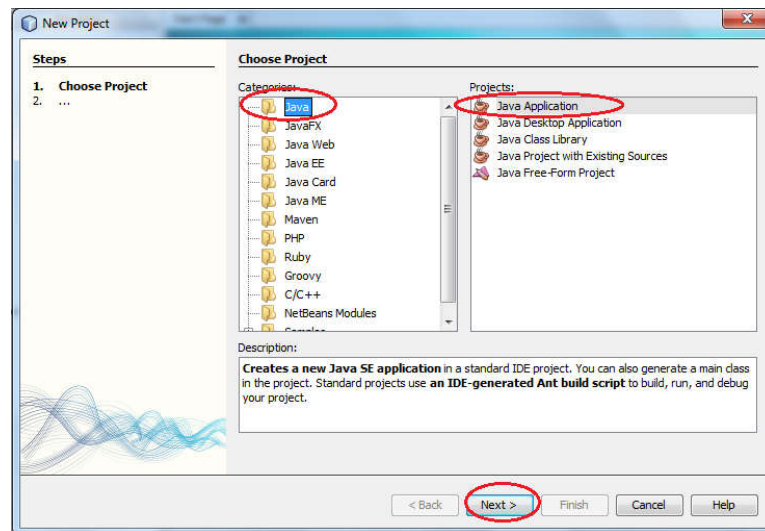
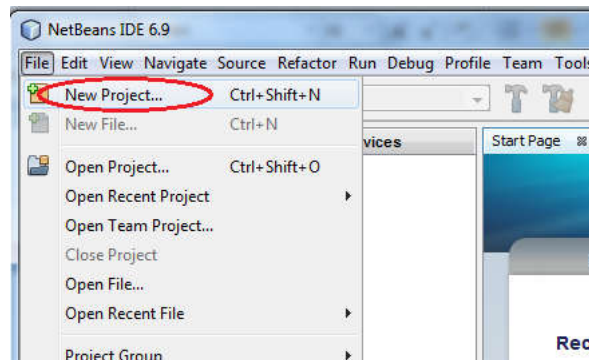
Table comments:

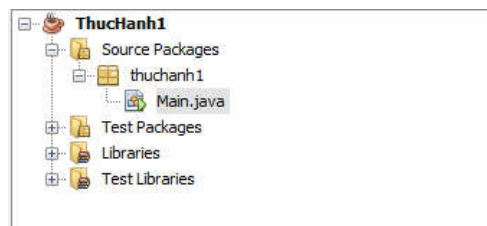
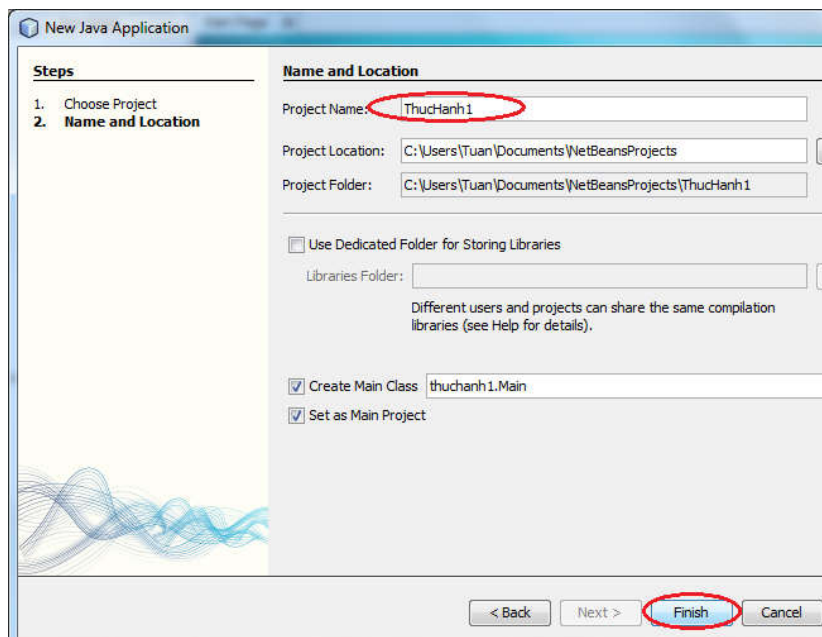
Storage Engine: InnoDB Collation:

Save Or Add 1 field(s) Go

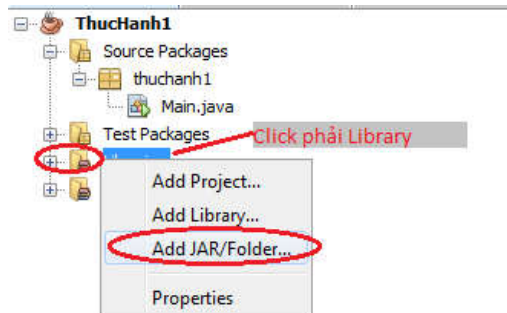
2. Tạo ra các lớp đối tượng truy xuất đến CSDL:

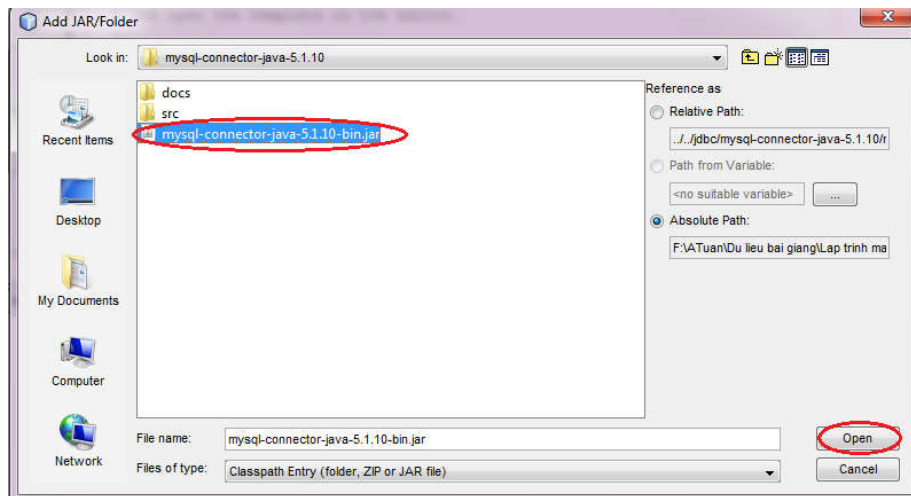
a. Bước 1: Tạo project mới: File → New Project



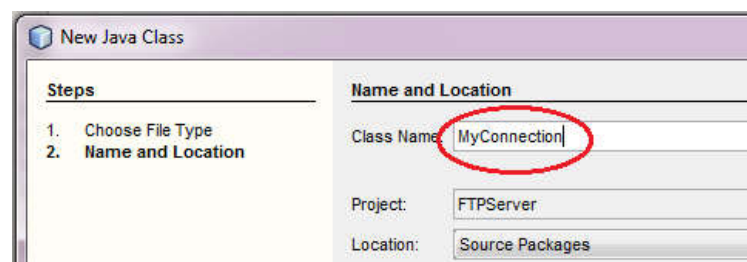
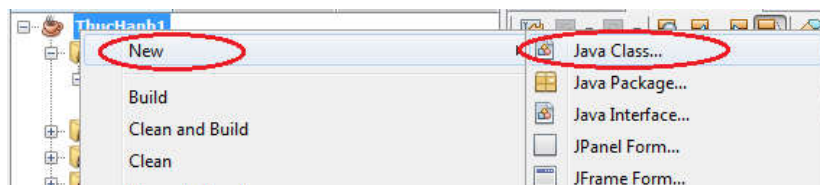


b. Bước 2: Thêm thư viện vào.





c. Bước 3: Tạo lớp chứa kết nối từ chương trình và mysql.



```
import java.sql.*;
import javax.swing.*;

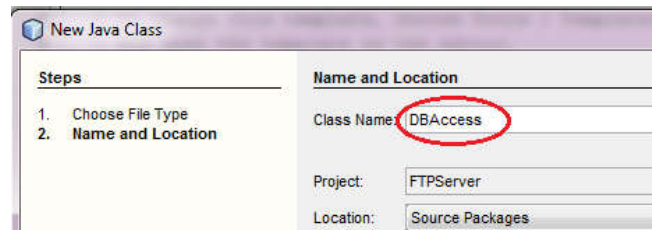
public class MyConnection {

    public Connection getConnection() {
        try {
            Class.forName("com.mysql.jdbc.Driver");//.newInstance();
            String URL = "jdbc:mysql://localhost/quanlytaikhoan?user=root&password=";
            Connection con = DriverManager.getConnection(URL);
            return con;
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.toString(), "Lỗi", JOptionPane.ERROR_MESSAGE);
            return null;
        }
    }
}
```

Diagram illustrating the database connection parameters in the code:

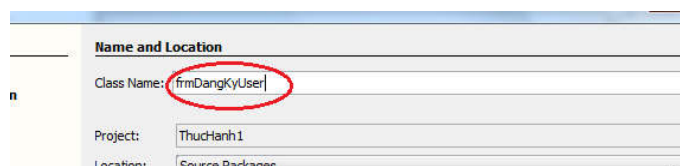
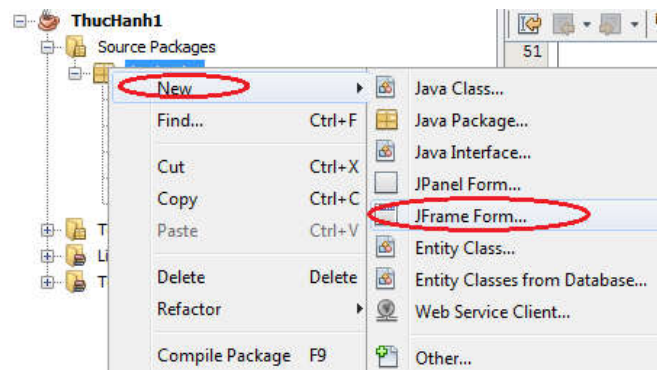
- Tên database** (Database Name): Points to the database name in the URL.
- Địa chỉ server** (Server Address): Points to the host part of the URL.
- Username**: Points to the user part of the URL.
- Password**: Points to the password part of the URL.

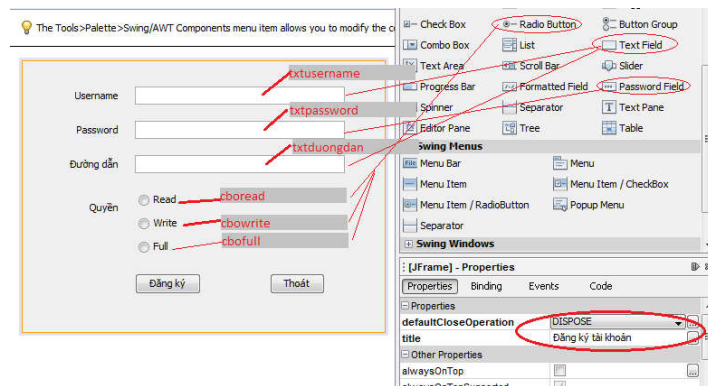
d. Bước 4: Tạo lớp truy xuất CSDL, và sau này khi cần truy xuất CSDL, ta chỉ cần gọi lớp này ra và thực thi các hàm bên trong nó.



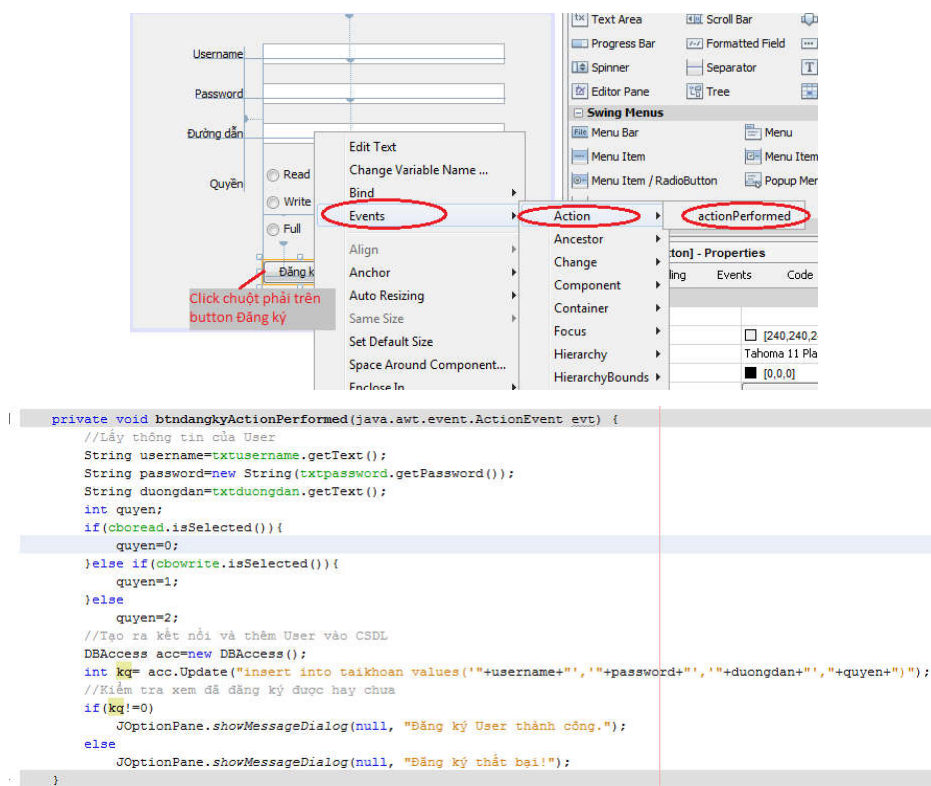
```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
public class DBAccess {
    private Connection con;
    private Statement stmt;
    public DBAccess() {
        try {
            MyConnection mycon=new MyConnection();
            con = mycon.getConnection();
            stmt=con.createStatement();
        } catch (Exception e) {
        }
    }
    public int Update(String str){
        try{
            int i=stmt.executeUpdate(str);
            return i;
        }catch(Exception e){
            return -1;
        }
    }
    public ResultSet Query(String str){
        try {
            ResultSet rs=stmt.executeQuery(str);
            return rs;
        } catch (Exception e) {
            return null;
        }
    }
}
```

e. Bước 5: Tạo giao diện đăng ký user.

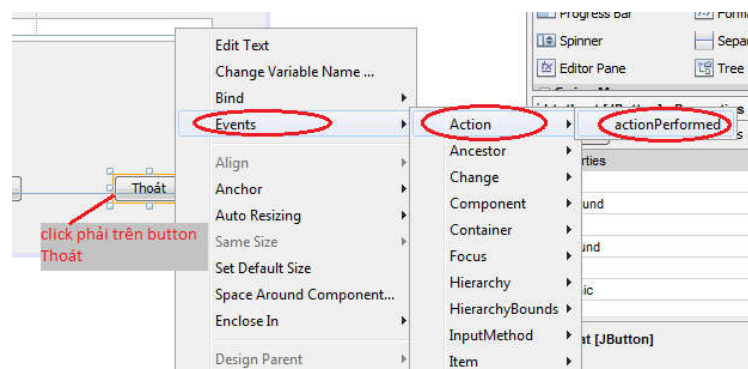




f. **Bước 6:** Thêm sự kiện khi click chuột vào nút Đăng ký.

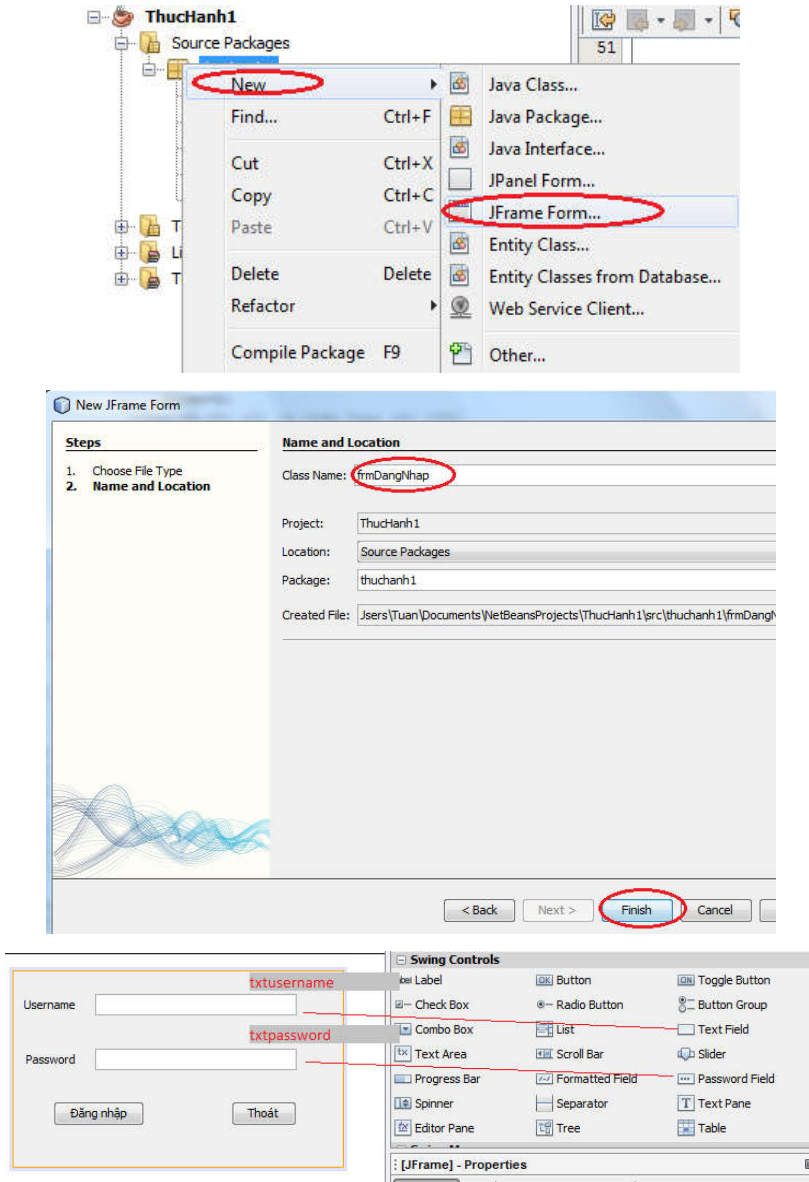


Sự kiện cho nút Thoát

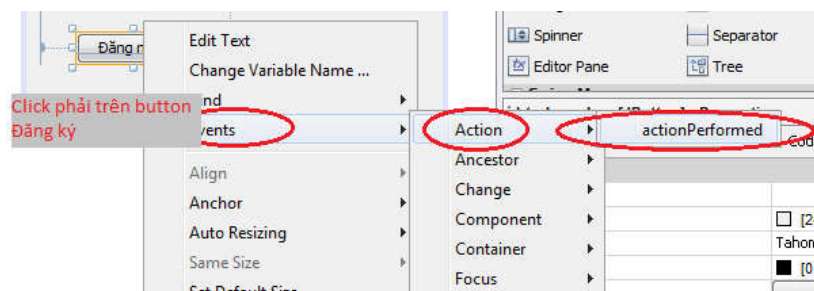


```
private void btnthoatActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

g. Bước 7: Tạo giao diện cho phần đăng nhập.



h. Bước 8: Thêm sự kiện cho nút đăng nhập.

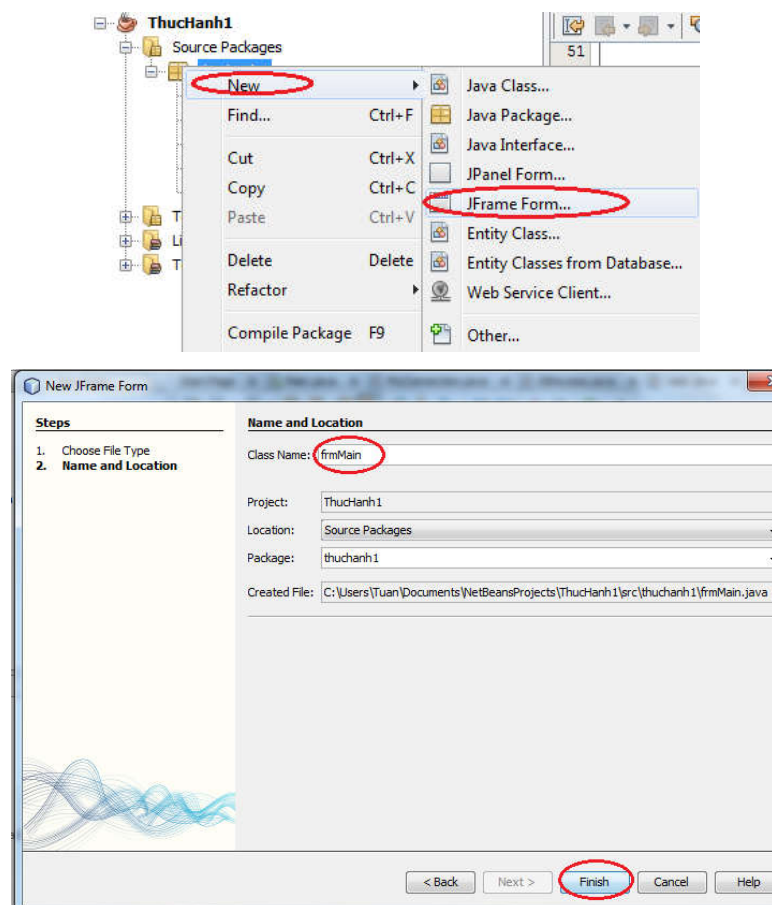


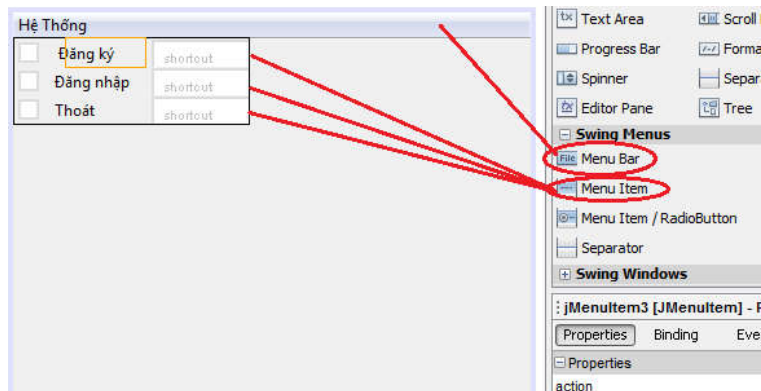

```
private void btndangnhapActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        //lấy thông tin của User
        String username=txtusername.getText();
        String password=new String(txtpassword.getPassword());
        //Tạo ra đối tượng giao tiếp CSDL
        DBAccess acc=new DBAccess();
        ResultSet rs=acc.Query("select * from taikhoan where username='"+username+"' and password='"+password+"'");
        if(rs.next()){
            JOptionPane.showMessageDialog(null, "Đăng nhập thành công!");
        }else{
            JOptionPane.showMessageDialog(null, "Đăng nhập thất bại!");
        }
    }catch(Exception e){}
}
```

Thêm sự kiện cho button Thoát

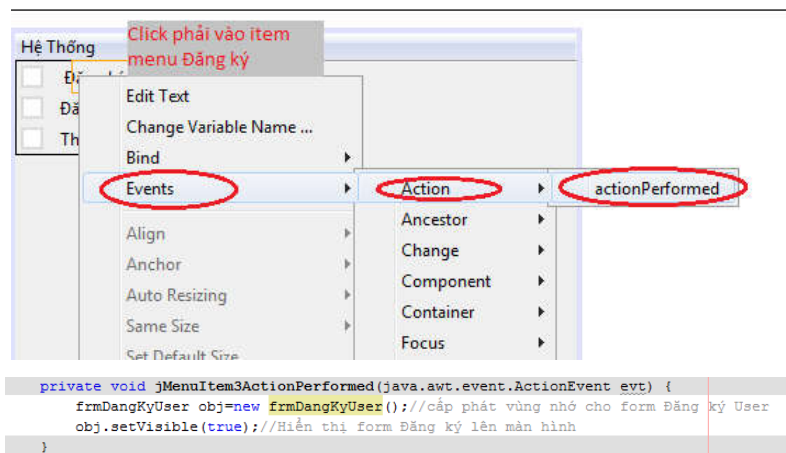
```
private void btnthoatActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

i. Bước 9: Tạo lớp giao diện cho chương trình.

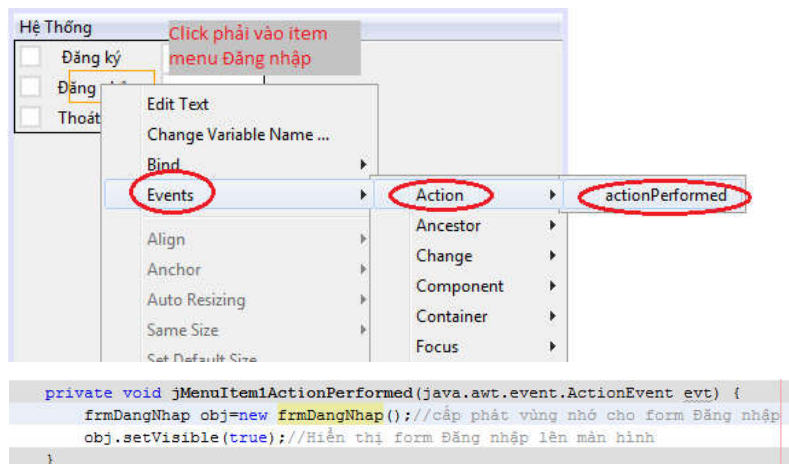




j. Bước 10: Thêm sự kiện cho các itemmenu
Thêm sự kiện cho itemmenu Đăng ký



Thêm sự kiện cho itemmenu Đăng nhập



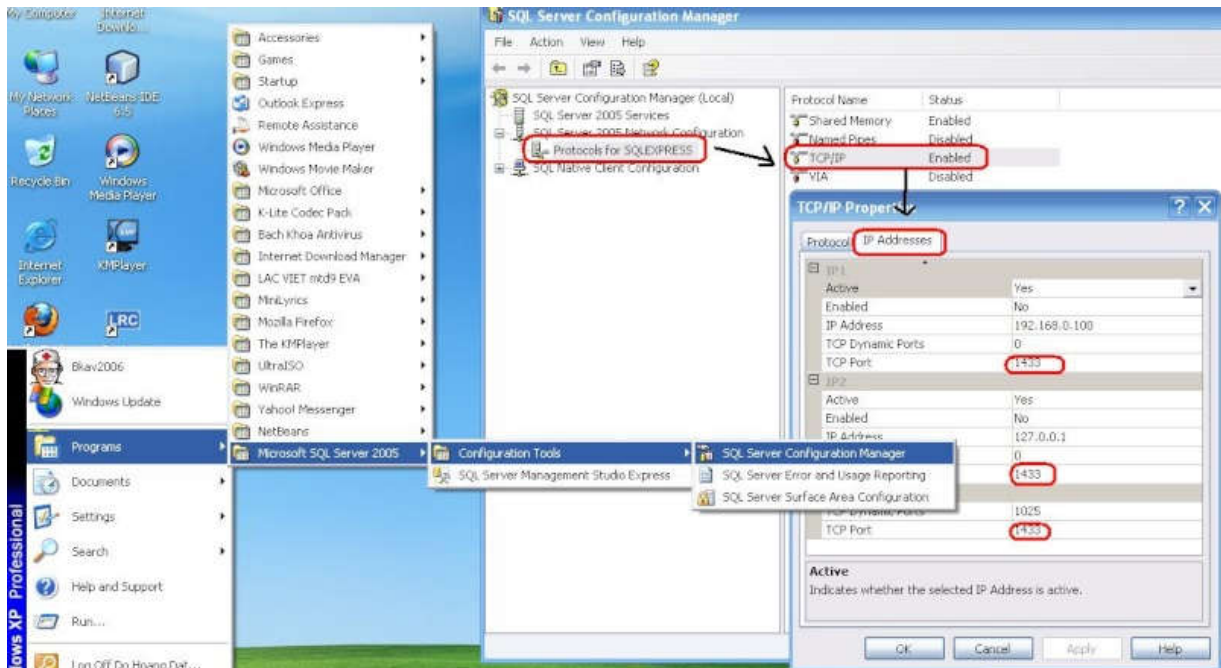
Thêm sự kiện cho itemmenu Thoát

```
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

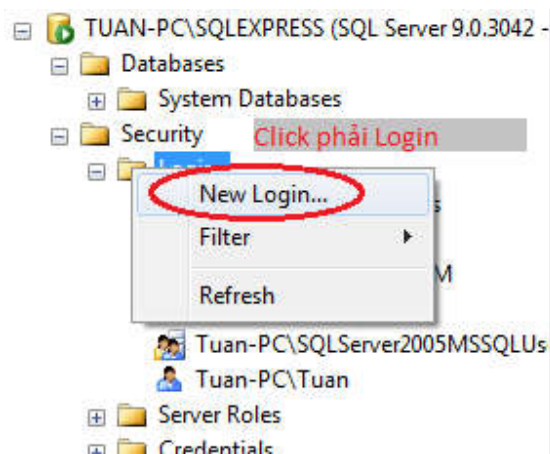
1.2.2 Kết nối với cơ sở dữ liệu SQL Server:

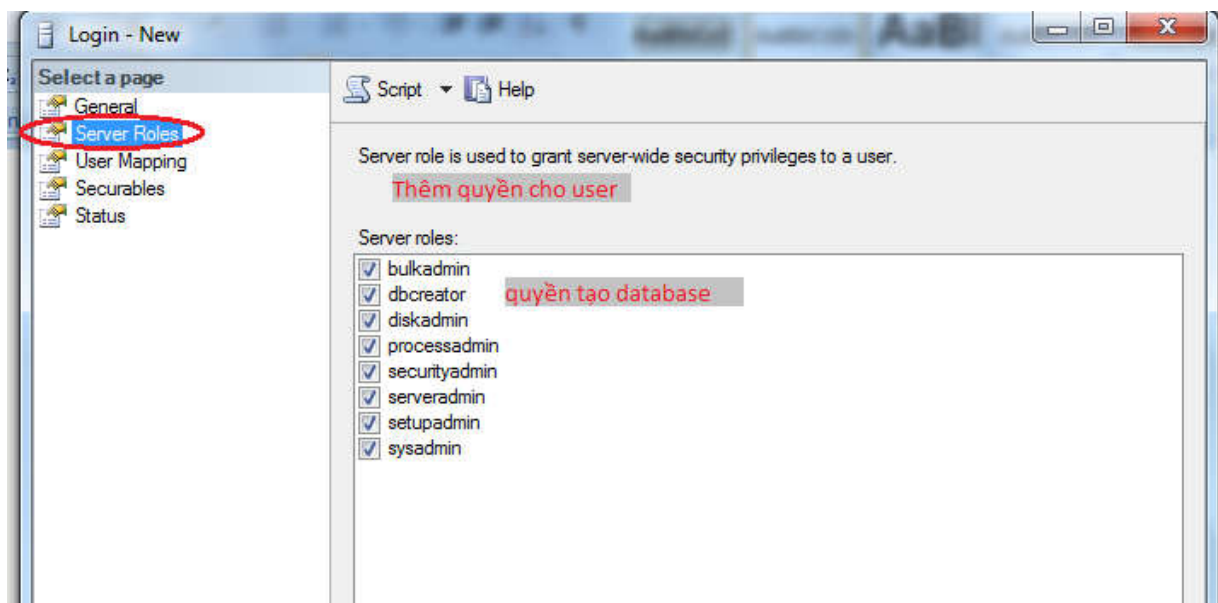
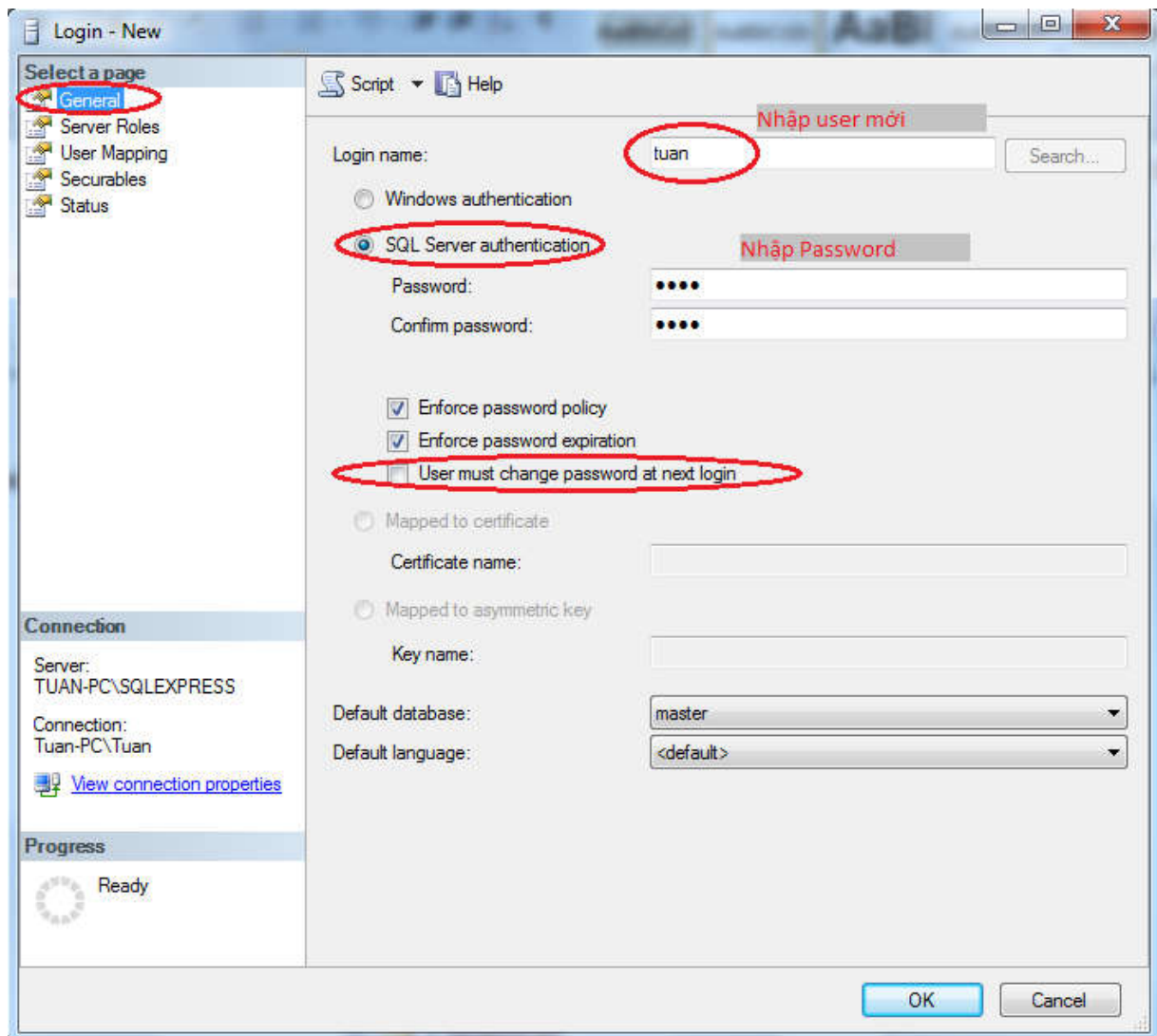
1. Tạo CSDL SQLServer:

a. **Bước 1:** Chỉnh lại port cho SQLEXPRESS.

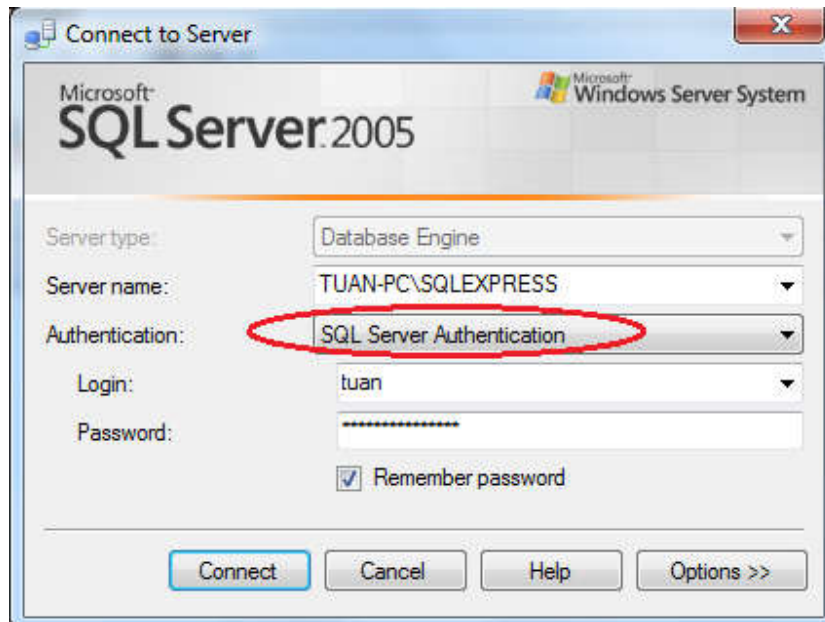


b. **Bước 2:** Tạo User mới trong sqlserver

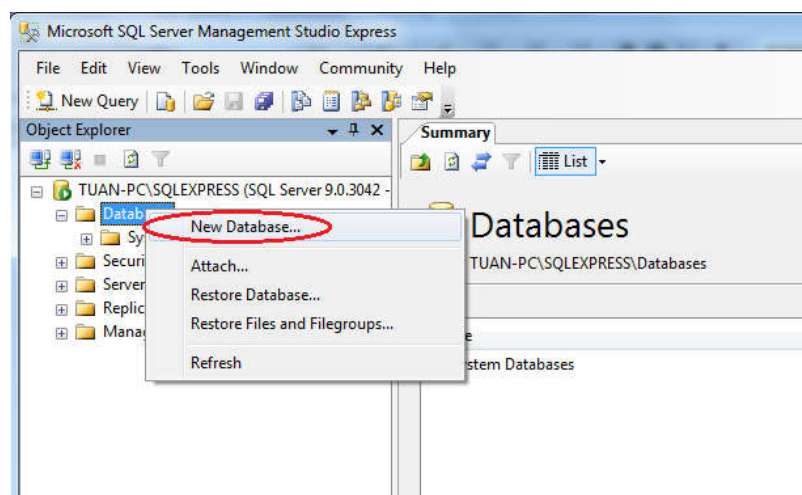


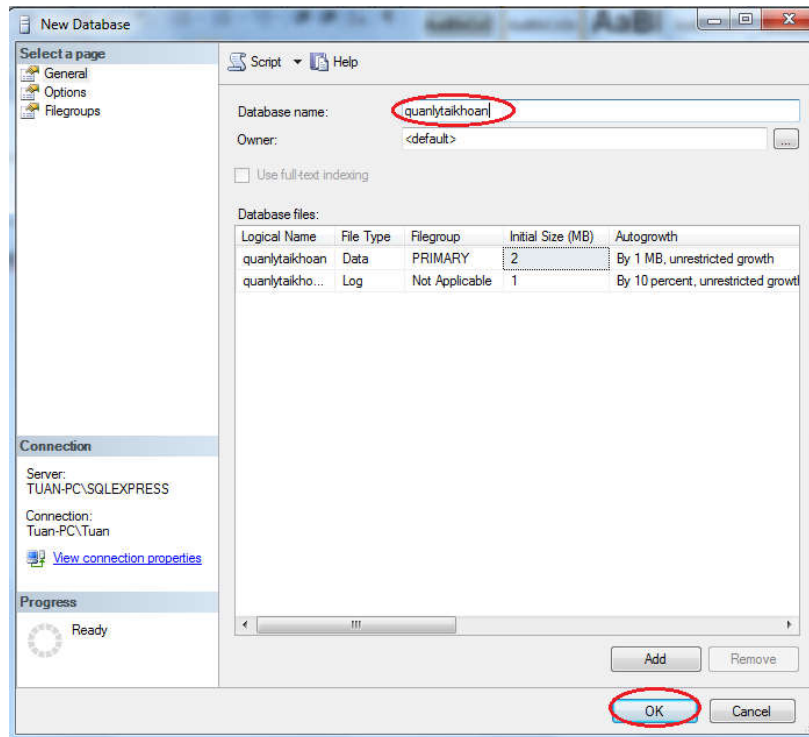


c. **Bước 3:** Đăng nhập vào Database với User mới tạo.

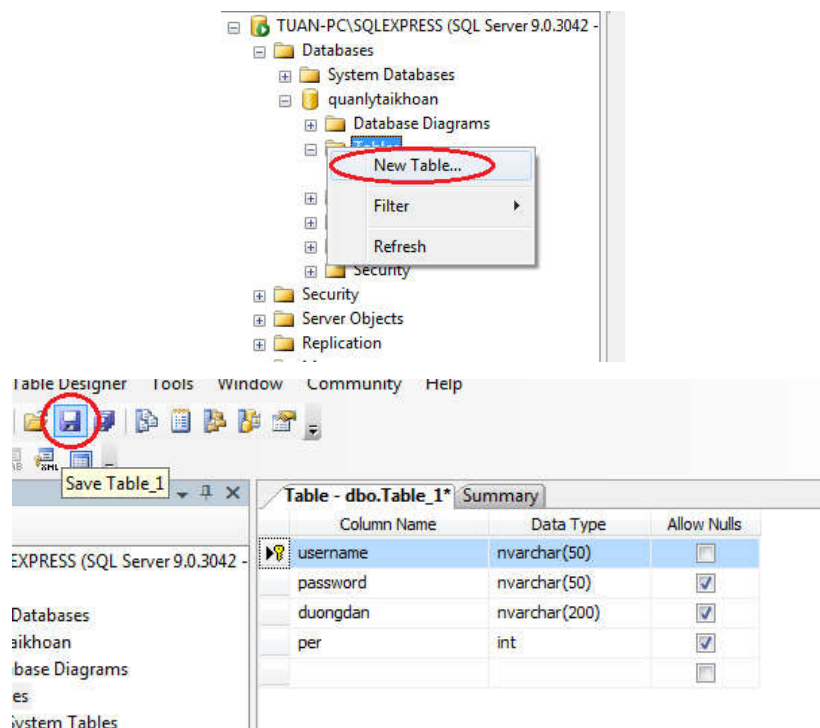


d. **Bước 4:** Tạo database mới quanlytaikhoan



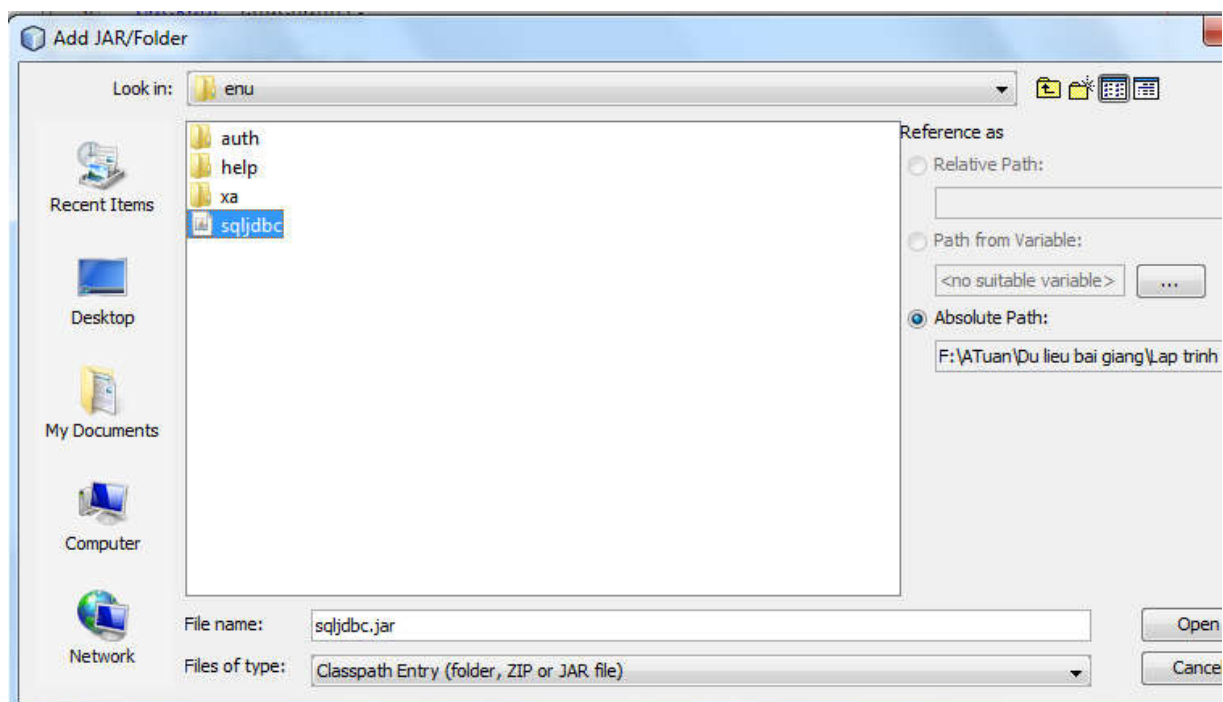
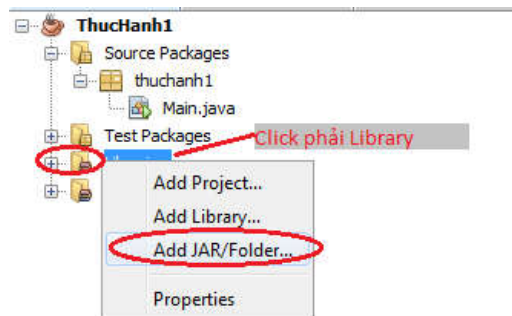


e. **Bước 5:** Tạo Table mới.





- f. Bước 6:** Tiếp theo ta thực hiện lại các bước như phần kết nối CSDL ở trên. Nhưng thay đổi phần add library như sau



- g. Bước 7:** Đồng thời thay đổi lớp MyConnection.java như sau

```
package thuchanh1;

import java.sql.*;
import javax.swing.JOptionPane;

/**
 *
 * @author Tuan
 */
public class MyConnection {
    public Connection getConnection(){
        try{
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String url = "jdbc:sqlserver://localhost:1433;Database=quanlytaikhoan;user=tuan;password=tuan";
            Connection con=DriverManager.getConnection(url);
            return con;
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, e.toString(), "Lỗi", JOptionPane.ERROR_MESSAGE);
            return null;
        }
    }
}
```



- h. Bước 8:** Sau khi làm xong các bước như phần kết nối CSDL mysql, ta kiểm tra chương trình đăng ký tài khoản và đăng nhập thử.

BÀI 2. LUỒNG NHẬP XUẤT

2.1 THAO TÁC VỚI THƯ MỤC

2.1.1 Yêu cầu

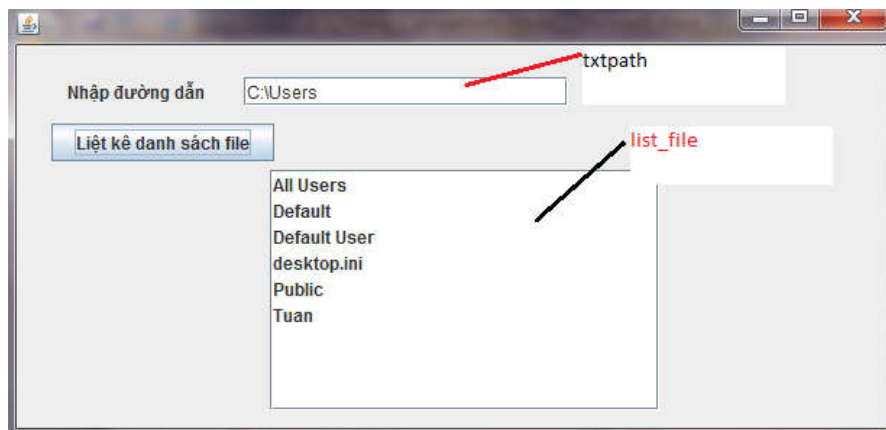
Viết chương trình nhập vào tên đường dẫn của một thư mục. Hiển thị tên tất cả các tập tin trong thư mục lên màn hình (sử dụng phương thức `fileList` của lớp đối tượng `File` và lớp `FileFilter`).

2.1.2 Hướng dẫn

Thao tác với File là một phần không thể thiếu trong các ứng dụng mạng, nhất là các ứng dụng truyền nhận dữ liệu.

1. Chương trình đơn giản, không có chức năng lọc danh sách.

a. Bước 1: Tạo Project mới → Tạo JFrameForm có giao diện như sau:



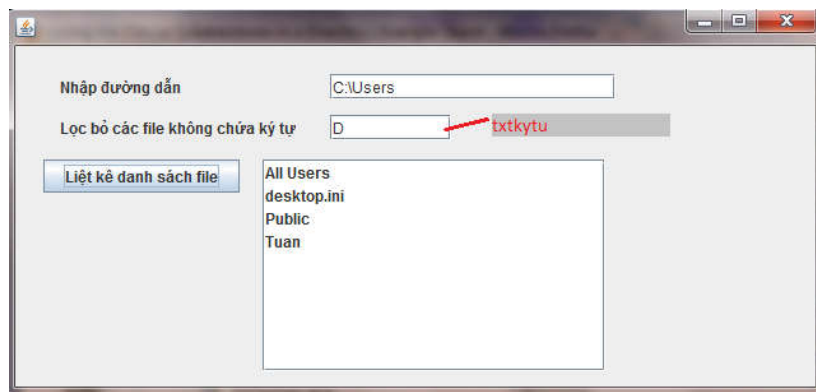
TextField để nhập đường dẫn. Listbox hiển thị tất cả các tập tin và thư mục có trong đường dẫn đến thư mục.

b. Bước 2: Thêm sự kiện cho button Liệt kê danh sách file

```
private void btnlietkeActionPerformed(java.awt.event.ActionEvent evt) {
    File dir=new File(txtpath.getText());
    File dsFile[]=dir.listFiles();
    if(dsFile==null)
    {
        JOptionPane.showMessageDialog(null, "sai duong dan!");
    }else{
        try{
            DefaultListModel dm=new DefaultListModel();
            for(int i=0;i<dsFile.length;i++){
                String filename=dsFile[i].getName();
                dm.addElement(filename);
            }
            list_file.setModel(dm);
        }catch(Exception e)
        {JOptionPane.showMessageDialog(null,e.toString());}
    }
}
```

2. Thêm chức năng lọc tên file cho chương trình:

a. Bước 1: Thêm TextField để nhận ký tự lọc vào giao diện:

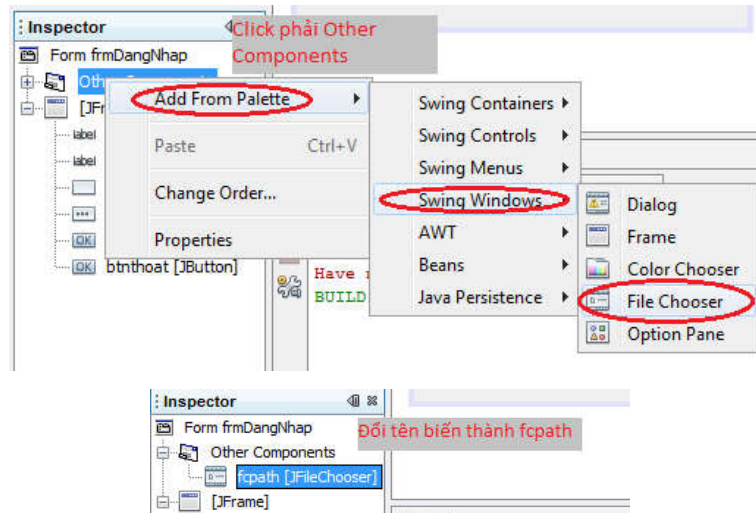


b. Bước 2: Sửa code cho button Liệt kê danh sách file:

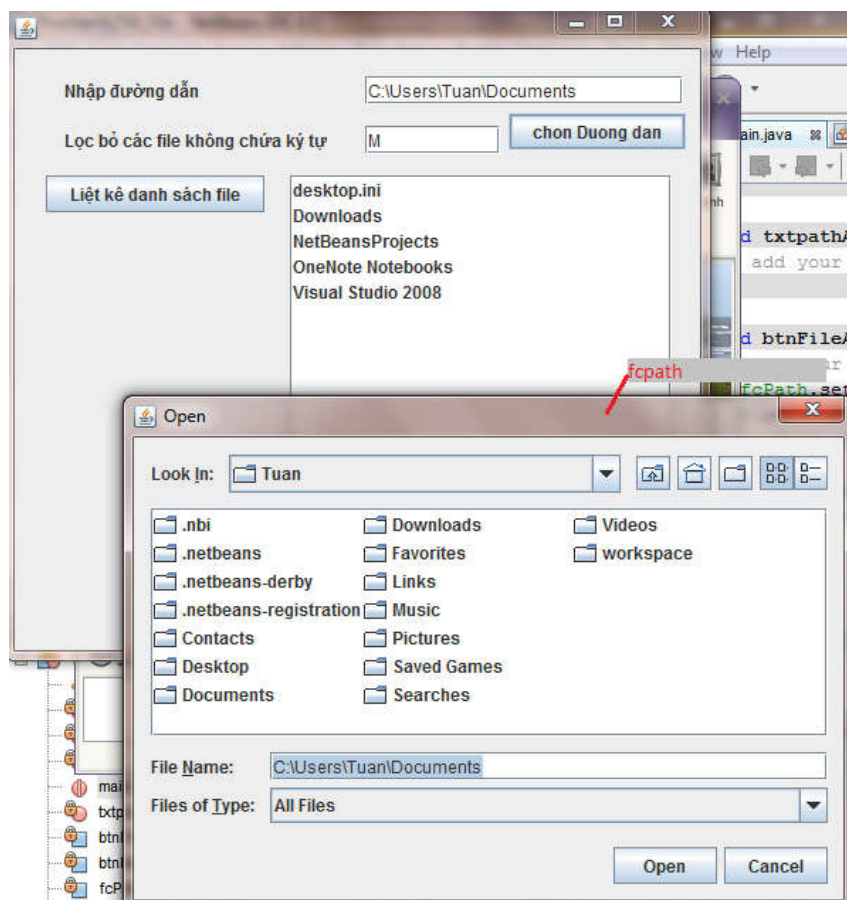
```
private void btnlietkeActionPerformed(java.awt.event.ActionEvent evt) {
    File dir=new File(txtpath.getText());
    File dsFile[]=dir.listFiles();
    if(dsFile==null)
    {
        JOptionPane.showMessageDialog(null, "sai duong dan!");
    }else{
        try{
            DefaultListModel dm=new DefaultListModel();
            for(int i=0;i<dsFile.length;i++){
                String filename=dsFile[i].getName();
                dm.addElement(filename);
            }
            list_file.setModel(dm);
        }catch(Exception e)
        {JOptionPane.showMessageDialog(null,e.toString());}
    }
}
```

3. Thêm chức năng nhập đường dẫn tiện lợi hơn:

a. Bước 1: Thêm JFileChooser vào Form:



b. Bước 2: Chỉnh sửa lại giao diện như sau



c. Bước 3: Thêm sự kiện cho button Chọn đường dẫn:

```
private void btnFileActionPerformed(java.awt.event.ActionEvent evt) {  
    //thiet lap hien thi hop thoai chon duong dan  
    this.fcPath.setVisible(true);  
    //thiet lap che do chon tap tin hay thu muc  
    fcPath.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);  
    //cho hop thoai hien thi len  
    if(this.fcPath.showOpenDialog(this)==JFileChooser.APPROVE_OPTION)  
    {  
        //neu nut open duoc chon  
        try{  
            txtpath.setText(fcPath.getSelectedFile().getCanonicalPath());  
        }catch(IOException e)  
        { JOptionPane.showMessageDialog(this,e); }  
    }  
}
```

2.2 THAO TÁC VỚI FILE

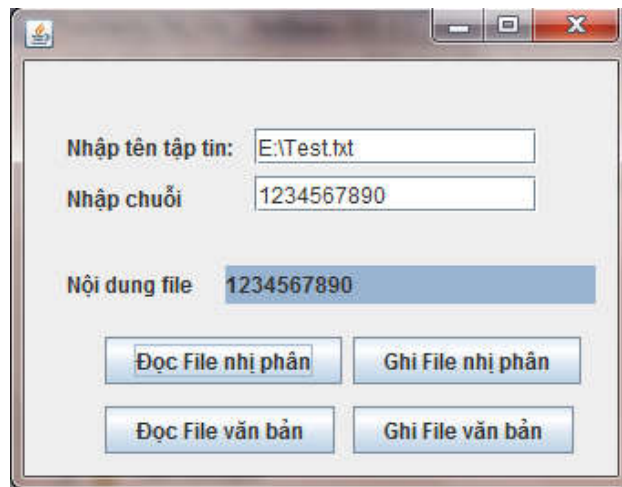
2.2.1 Yêu cầu

Viết chương trình nhập vào tên một tập tin. Viết một dãy số ngẫu nhiên n phần tử vào tập tin này. Sau đó, đọc nội dung tập tin này hiển thị lên màn hình. Yêu cầu: thực hiện ở hai chế độ (đọc/viết nhị phân và đọc viết văn bản) và có sử dụng các luồng đọc/viết có định kiểu và luồng đệm dữ liệu.

2.2.2 Hướng dẫn

Trong bài trên, chúng ta đã làm quen với thao tác trên tên file, trong phần này, chúng ta sẽ cùng đọc và ghi file.

- a. **Bước 1:** Tạo Project mới → Tạo giao diện như hình sau.



b. **Bước 2:** Xử lý sự kiện cho button Ghi File nhị phân:

```
private void btVietFileNhiPhanActionPerformed(java.awt.event.ActionEvent evt) {
    byte a[]=new byte[20];
    File file;
    int i;
    char s[]=txtNhapChuoi.getText().toCharArray();
    for(i=0;i<s.length;i++){
        a[i]=(byte)s[i];
    }

    try {
        file = new File(txtFileName.getText());
        FileOutputStream fo=new FileOutputStream(file);
        fo.write(a);
        fo.close();
    } catch (IOException e) {e.printStackTrace();}
}
```

c. **Bước 3:** Xử lý sự kiện cho button Đọc File nhị phân:

```
private void btDocFileNhiPhanActionPerformed(java.awt.event.ActionEvent evt) {
    byte a[]; //new byte[20];
    File file;
    try{
        file=new File(txtFileName.getText());
        FileInputStream fi=new FileInputStream(file);
        a=new byte[fi.available()];
        fi.read(a);
        fi.close();
        //xuất kết quả lên màn hình
        txtNoiDungFile.setText(new String(a));
    }catch(Exception e){}
}
```

d. **Bước 4:** Xử lý sự kiện cho button Ghi File văn bản:

```
private void btVietFileVanBanActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        FileWriter fw=new FileWriter(new File(txtFileName.getText()));
        fw.write(txtNhapChuoi.getText());
        fw.close();
    }catch(Exception e){e.printStackTrace();}
}
```

e. **Bước 5:** Xử lý sự kiện cho button Đọc File văn bản:

```
private void btDocFileVanBanActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        FileReader fr=new FileReader(new File(txtFileName.getText()));
        StringBuffer sb=new StringBuffer();
        char ca[]=new char[5]; //đọc mỗi lần 5 ký tự
        while(fr.ready()){
            int len=fr.read(ca);
            sb.append(ca,0,len);
        }
        fr.close();
        txtNoiDungFile.setText(sb+"");
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

2.3 BÀI TẬP THÊM

Sinh viên áp dụng các kiến thức đã học để thực hiện các bài tập sau:

2.3.1 Bài tập 1

Viết chương trình tạo ra một số đối tượng phân số ngẫu nhiên. Lưu các đối tượng phân số này xuống tập tin "phanso.dat". Sau đó, đọc các phân số có trong tập tin "phanso.dat" vào chương trình và hiển thị các phân số có mẫu số là số nguyên tố.

2.3.2 Bài tập 2

Viết chương trình nhập tên một tập tin. Tách tập tin thành từng đoạn. Mỗi đoạn lưu vào mảng số nguyên có kích thước tối đa là 100 phần tử. Sau đó, đọc nội dung các mảng này ghi vào tập tin mới có tên là "Mang.txt".

BÀI 3. INETADDRESS

3.1 INETADDRESS

3.1.1 Lớp INETADDRESS

Các thiết bị được kết nối với mạng LAN có địa chỉ vật lý duy nhất. Điều này giúp cho các máy khác trên mạng trong việc truyền các gói tin đến đúng vị trí. Tuy nhiên, địa chỉ này chỉ có ích trong mạng LAN. Một máy không thể xác định được vị trí trên Internet bằng cách sử dụng các địa chỉ vật lý, vì các địa chỉ vật lý không chỉ ra vị trí của máy. Những người lập trình mạng không cần phải quan tâm đến từng chi tiết dữ liệu được định tuyến như thế nào trong một mạng LAN. Hơn nữa, Java không cung cấp khả năng truy xuất tới các giao thức tầng liên kết dữ liệu mức thấp được sử dụng bởi LAN. Việc hỗ trợ như vậy là rất khó khăn. Vì mỗi kiểu giao thức sử dụng một kiểu địa chỉ khác nhau và có các đặc trưng khác nhau, chúng ta cần phải các chương trình khác nhau cho mỗi kiểu giao thức mạng khác nhau. Thay vào đó, Java hỗ trợ giao thức TCP/IP, giao thức này có nhiều vụ liên kết các mạng với nhau. Các thiết bị có một kết nối Internet trực tiếp được cung cấp một định danh duy nhất được gọi là địa chỉ IP. Các địa chỉ IP có thể là tĩnh hoặc động. Các địa chỉ IP được cấp phát động thường được sử dụng khi nhiều thiết bị cần truy cập Internet trong khoảng thời gian nhất định. Một địa chỉ IP chỉ có thể gắn với một máy, nó không thể dùng chung. Địa chỉ này được sử dụng bởi giao thức IP để định tuyến các datagram tới đúng vị trí. Không có địa chỉ, ta không thể liên lạc được với máy đó; vì thế tất cả các máy tính đều phải có một địa chỉ IP duy nhất. Lớp `java.net.InetAddress` biểu diễn một địa chỉ Internet. Nó bao gồm hai trường thông tin: `hostName` (một đối tượng kiểu `String`) và `address` (một số kiểu `int`). Các trường này không phải là trường `public`, vì thế ta không thể truy xuất chúng trực tiếp. Lớp này được sử dụng bởi hầu hết các lớp mạng, bao gồm `Socket`, `ServerSocket`, `URL`, `DatagramSocket`, `DatagramPacket`,...

3.1.2 Tạo đối tượng INETADDRESS

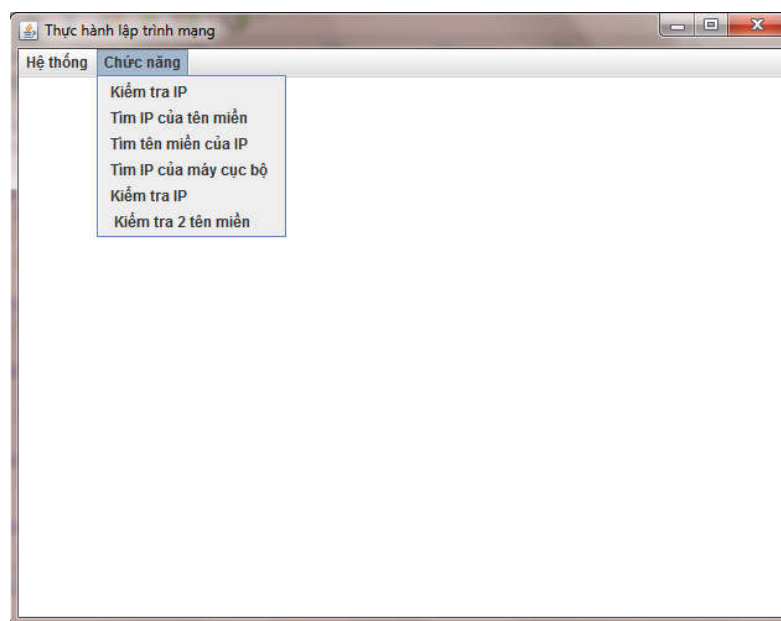
Lớp InetAddress được sử dụng để biểu diễn các địa chỉ IP trong một ứng dụng mạng sử dụng Java. Không giống với các lớp khác, không có các constructor cho lớp InetAddress. Tuy nhiên, lớp InetAddress có ba phương thức tĩnh trả về các đối tượng InetAddress. Các phương thức trong lớp InetAddress:

- public static InetAddress InetAddress.getByName(String hostname)
- public static InetAddress[] InetAddress.getAllByName(String hostname)
- public static InetAddress InetAddress.getLocalHost()

Tất cả các phương thức này đều thực hiện kết nối tới server DNS cục bộ để biết được các thông tin trong đối tượng InetAddress.

3.2 BÀI TẬP THỰC HÀNH

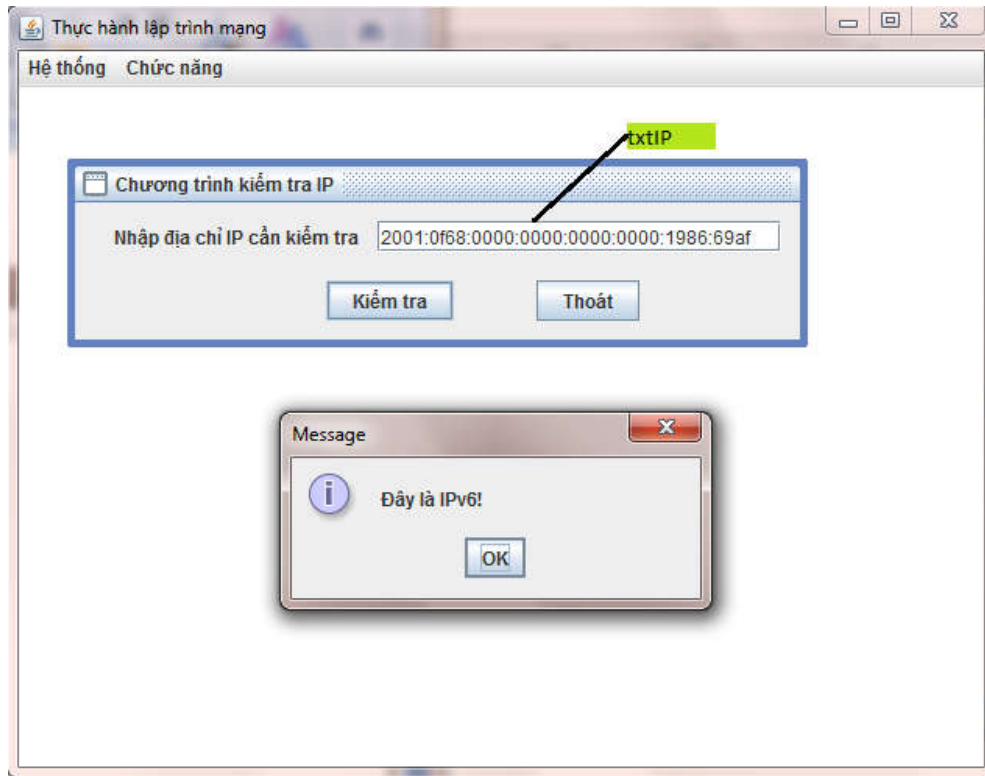
Tạo giao diện như hình sau:



3.2.1 Bài thực hành 1

Viết chương trình kiểm tra một tên miền có địa chỉ IP là phiên bản 4 hay phiên bản 6.

Bước 1: Tạo JFrameForm có giao diện như sau:



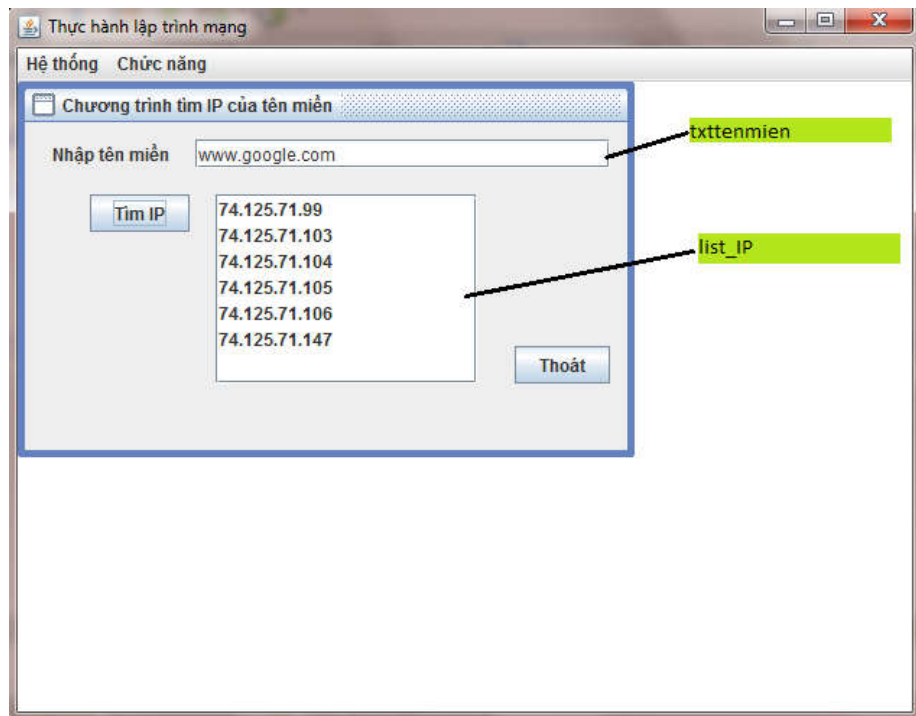
Bước 2: Xử lý sự kiện cho button Kiểm tra

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String IP=txtIP.getText();
    try{
        InetAddress host=InetAddress.getByName(IP);
        if(host!=null)
        {
            if(IP.contains("."))
                JOptionPane.showMessageDialog(null, "Đây là IPv4");
            else
                JOptionPane.showMessageDialog(null, "Đây là IPv6!");
        }else
            JOptionPane.showMessageDialog(null, "Địa chỉ IP của bạn nhập sai!!!");
    }catch(Exception e)
    {
        JOptionPane.showMessageDialog(null, "Địa chỉ IP của bạn nhập sai!!!: \n"+e.toString());
    }
}
```

3.2.2 Bài thực hành 2

Viết chương trình nhập vào địa chỉ tên miền. Cho biết tất cả địa chỉ IP của tên miền tương ứng.

Bước 1: Tạo JFrameForm có giao diện như sau:



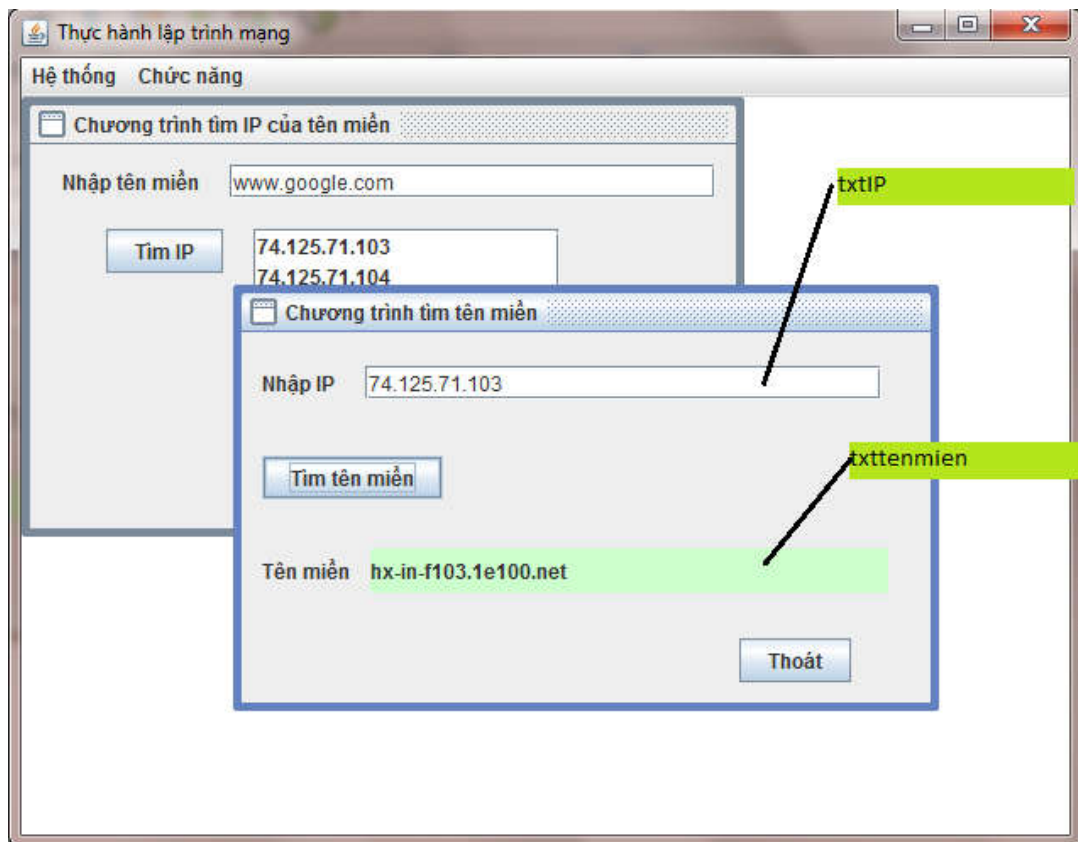
Bước 2: Xử lý sự kiện cho button Tìm IP.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int i,j;
        InetAddress addr[] = InetAddress.getAllByName(txttenmien.getText()); //lấy tất cả các IP của địa chỉ
        DefaultListModel dm=new DefaultListModel();
        for(i=0;i<addr.length;i++){
            byte[] ipAddr = addr[i].getAddress();
            String ipAddrStr = "";
            for ( j=0; j<ipAddr.length; j++){
                if (j > 0) { ipAddrStr += "."; }
                ipAddrStr += ipAddr[j]&0xFF;
            }
            dm.addElement(ipAddrStr);
        }
        list_IP.setModel(dm);
    } catch (UnknownHostException e) {
        JOptionPane.showMessageDialog(null, "Địa chỉ của bạn nhập sai!!!");
    }
}
```

3.2.3 Bài thực hành 3

Viết chương trình nhập vào địa chỉ IP của một máy. Cho biết tên miền tương ứng nếu có.

Bước 1: Tạo JFrameForm có giao diện như sau:



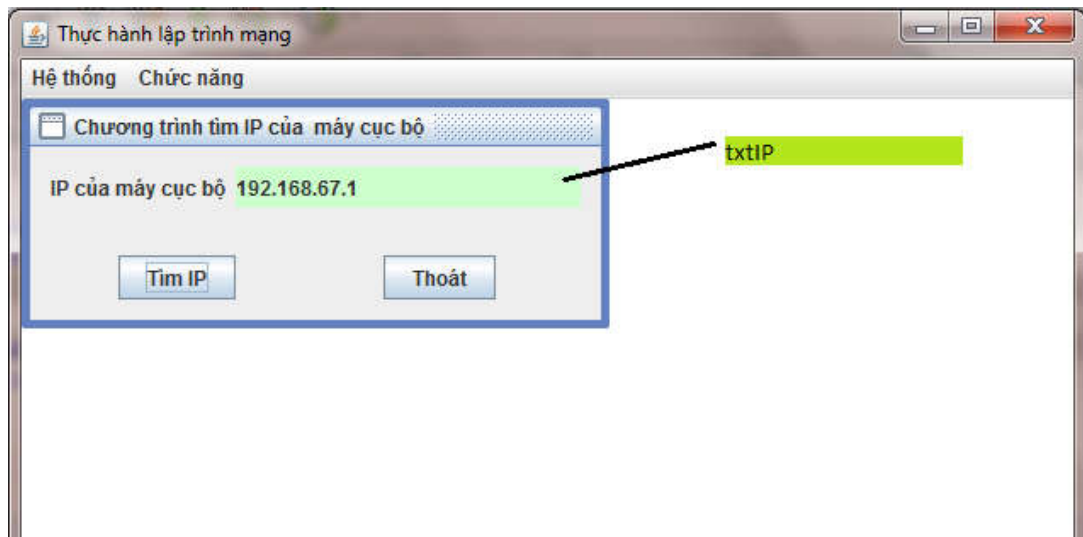
Bước 2: Xử lý sự kiện cho button Tìm tên miền.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        InetAddress addr = InetAddress.getByName(txtIP.getText());
        String hostname = addr.getHostName();
        txttenmien.setText(hostname);
    } catch (UnknownHostException e) {
        JOptionPane.showMessageDialog(null, "Bạn nhập sai tên miền rồi!!!");
    }
}
```

3.2.4 Bài thực hành 4

Viết chương trình cho biết địa chỉ IP của máy cục bộ.

Bước 1: Tạo JFrameForm có giao diện như sau:



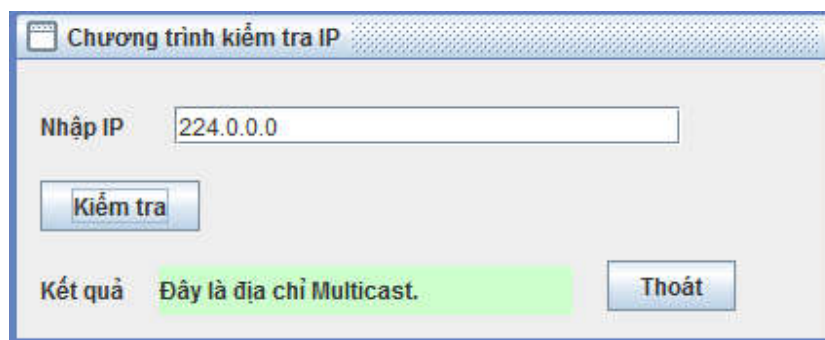
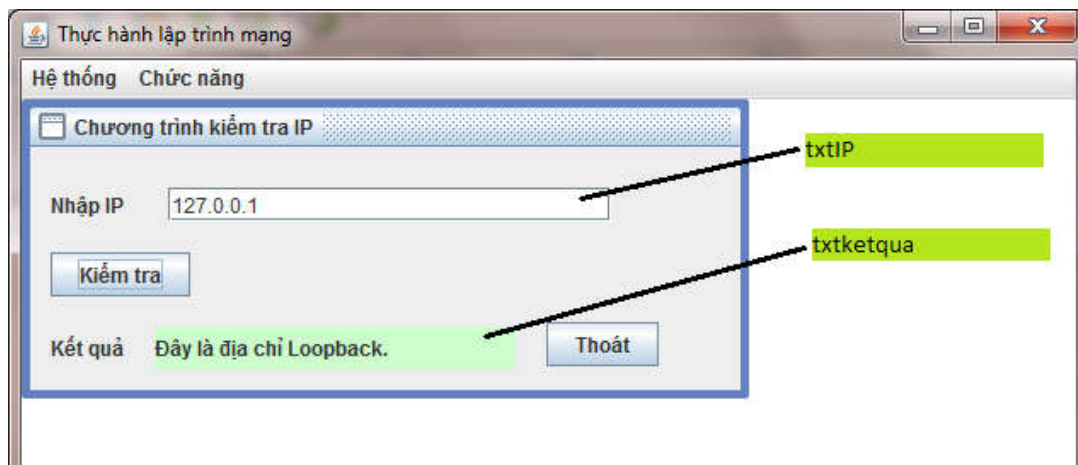
Bước 2: Xử lý sự kiện cho button Tìm IP.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        InetAddress addr = InetAddress.getLocalHost();  
        byte[] ipAddr = addr.getAddress();  
        String ipAddrStr = "";  
        for (int j=0; j<ipAddr.length; j++){  
            if (j > 0) { ipAddrStr += "."; }  
            ipAddrStr += ipAddr[j]&0xFF;  
        }  
        txtIP.setText(ipAddrStr);  
    } catch (UnknownHostException e) {  
        JOptionPane.showMessageDialog(null, "Lỗi!!!");  
    }  
}
```

3.2.5 Bài thực hành 5

Viết chương trình nhập vào một địa chỉ IP, kiểm tra đặc điểm của địa chỉ IP này có phải là một trong các dạng địa chỉ sau: địa chỉ cục bộ, địa chỉ loopback, địa chỉ multicast.

Bước 1: Tạo JFrameForm có giao diện như sau:



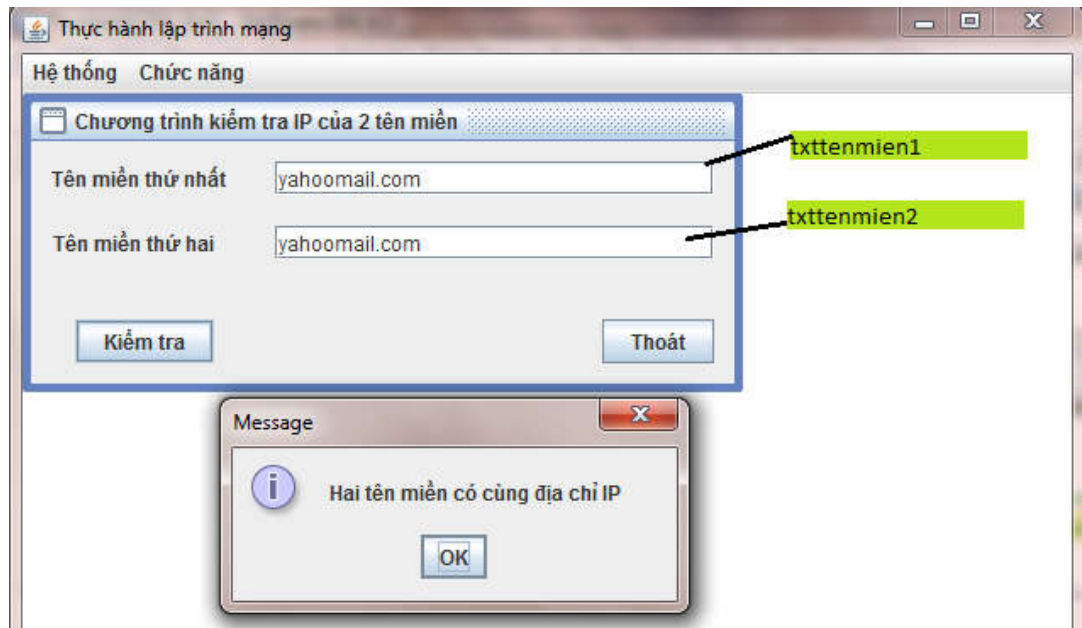
Bước 2: Xử lý sự kiện cho button Kiểm tra.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        InetAddress add=InetAddress.getByName(txtIP.getText());
        InetAddress localhost=InetAddress.getLocalHost();
        if(add.equals(localhost)){
            txtketqua.setText("Đây là địa chỉ Localhost");
        }else{
            if(add.isMulticastAddress())
                txtketqua.setText("Đây là địa chỉ Multicast.");
            else if(add.isLoopbackAddress())
                txtketqua.setText("Đây là địa chỉ Loopback.");
            else txtketqua.setText("Không thấy gì đặc biệt!");
        }
    }catch(UnknownHostException ex){
    }
}
```

3.2.6 Bài thực hành 6

Viết chương trình nhập vào 2 địa chỉ tên miền khác nhau. Kiểm tra xem hai tên miền này có cùng địa chỉ IP không?

Bước 1: Tạo JFrameForm có giao diện như sau:



Bước 2: Xử lý sự kiện cho button Kiểm tra.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    InetAddress add1[], add2[];
    try{
        add1 = InetAddress.getAllByName(txttenmien1.getText());
        add2 = InetAddress.getAllByName(txttenmien2.getText());
        if (Arrays.equals(add1, add2))
            JOptionPane.showMessageDialog(null, "Hai tên miền có cùng địa chỉ IP");
        else
            JOptionPane.showMessageDialog(null, "Hai tên miền không cùng địa chỉ IP");
    } catch (UnknownHostException e) {
        JOptionPane.showMessageDialog(null, e.toString());
    }
}
```

BÀI 4. XỬ LÝ TIẾN TRÌNH

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các khái niệm cơ bản về luồng (Thread), đa nhiệm (Multitasking), đa luồng (MultiThread), cách tạo ra luồng trong Java và ứng dụng vào các bài tập thực tế.

4.1 LUỒNG TRONG JAVA

4.1.1 Giới thiệu Thread

- Thread (luồng) về cơ bản là một tiến trình con (sub-process). Một đơn vị xử lý nhỏ nhất của máy tính có thể thực hiện một công việc riêng biệt. Trong Java, các luồng được quản lý bởi máy ảo Java (JVM).
- Multi-thread (đa luồng) là một tiến trình thực hiện nhiều luồng đồng thời. Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn.

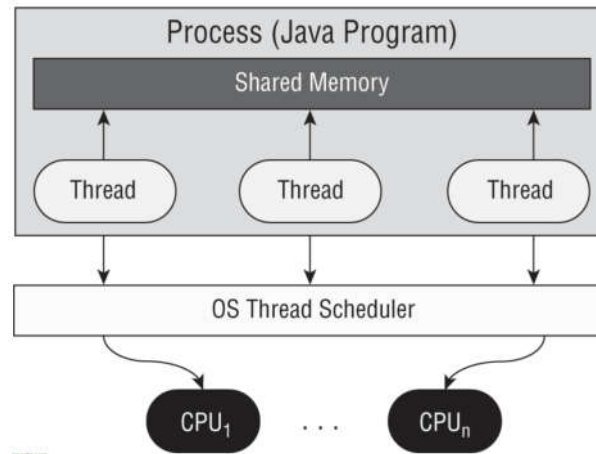
4.1.2 Đa nhiệm (Multitasking)

Multitasking: Là khả năng chạy đồng thời một hoặc nhiều chương trình cùng một lúc trên một hệ điều hành. Hệ điều hành quản lý việc này và sắp xếp lịch phù hợp cho các chương trình đó. Ví dụ, trên hệ điều hành Windows chúng ta có làm việc đồng thời với các chương trình khác nhau như: Microsoft Word, Excel, Media Player, ... Chúng ta sử dụng đa nhiệm để tận dụng tính năng của CPU. Đa nhiệm có thể đạt được bằng hai cách:

- Đa nhiệm dựa trên đơn tiến trình (Process) – Đa tiến trình (Multiprocessing).
- Đa nhiệm dựa trên luồng (Thread) – Đa luồng (MultiThreading).

Đa tiến trình (multiprocessing) và đa luồng (multithreading) cả hai được sử dụng để tạo ra hệ thống đa nhiệm (multitasking). Nhưng chúng ta sử dụng đa luồng nhiều hơn

đa tiến trình bởi vì các luồng chia sẻ một vùng bộ nhớ chung. Chúng không phân bổ vùng bộ nhớ riêng biệt để tiết kiệm bộ nhớ, và chuyển đổi ngữ cảnh giữa các luồng mất ít thời gian hơn tiến trình.



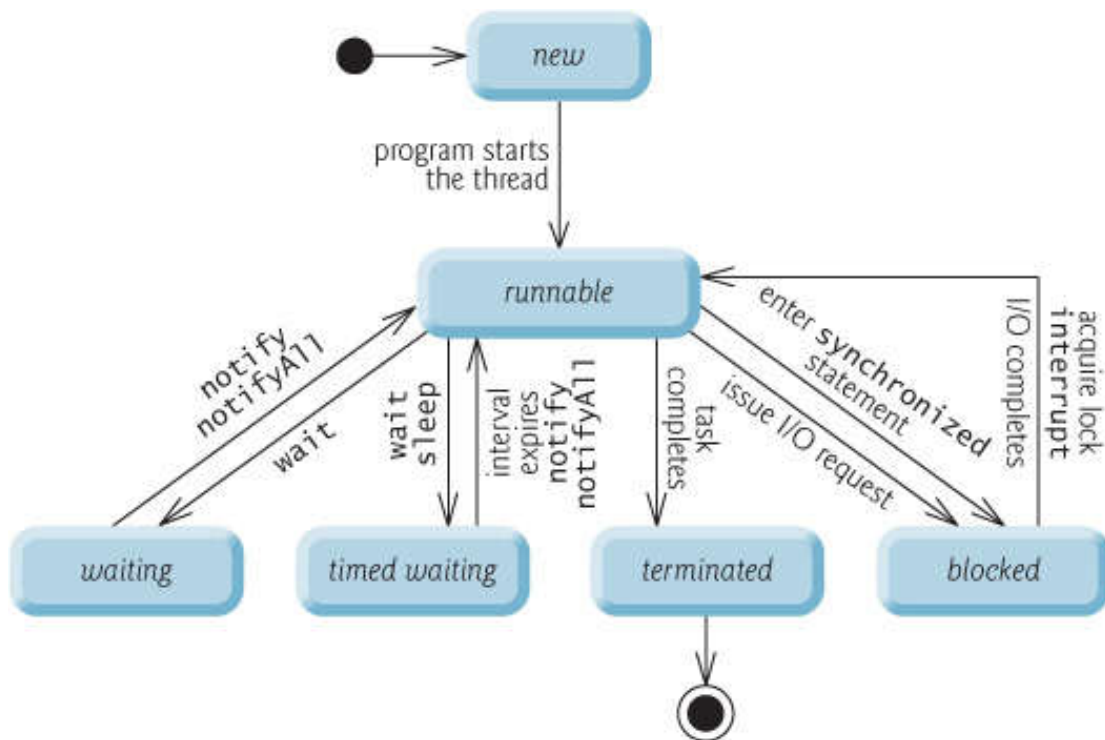
Hình 1: Luồng trong Java

4.1.3 Ưu điểm của đa luồng

- Nó không chặn người sử dụng vì các luồng là độc lập và bạn có thể thực hiện nhiều công việc cùng một lúc.
- Mỗi luồng có thể dùng chung và chia sẻ nguồn tài nguyên trong quá trình chạy, nhưng có thể thực hiện một cách độc lập.
- Luồng là độc lập vì vậy nó không ảnh hưởng đến luồng khác nếu ngoại lệ xảy ra trong một luồng duy nhất.
- Có thể thực hiện nhiều hoạt động với nhau để tiết kiệm thời gian. Ví dụ một ứng dụng có thể được tách thành : luồng chính chạy giao diện người dùng và các luồng phụ nhiệm gửi kết quả xử lý đến luồng chính.

4.1.4 Nhược điểm của đa luồng

- Càng nhiều luồng thì xử lý càng phức tạp.
- Xử lý vấn đề về tranh chấp bộ nhớ, đồng bộ dữ liệu khá phức tạp.
- Cần phát hiện tránh các luồng chết (dead lock), luồng chạy mà không làm gì trong ứng dụng cả.



Hình 2: Vòng đời (các trạng thái) của luồng trong Java

4.2 BÀI TẬP THỰC HÀNH

4.2.1 Bài thực hành 1

Viết chương trình sử dụng `SingleThread` để tạo ra một quả banh chạy trên màn hình.

Bước 1: Tạo ra lớp `Ball.java`

```
import java.awt.*;
import javax.swing.*;

public class Ball {
    private JPanel box;
    private static final int XSIZE=10;
    private static final int YSIZE=10;
    private int x=0;
    private int y=0;
    private int dx=2;
    private int dy=2;

    public Ball(JPanel p) {
        box=p;
    }

    public void draw(){
        Graphics g=box.getGraphics();
        g.fillOval(x, y, XSIZE, YSIZE);
        g.dispose();
    }

    public void move(){
        //xóa hình cũ bằng cách vẽ đè lên
        Graphics g=box.getGraphics();
        g.setXORMode(Color.CYAN);
        g.fillOval(x, y, XSIZE, YSIZE);
        x+=dx;
        y+=dy;
        Dimension d=box.getSize();
        //kiểm tra có đụng các cạnh
        if(x<0){
```

```

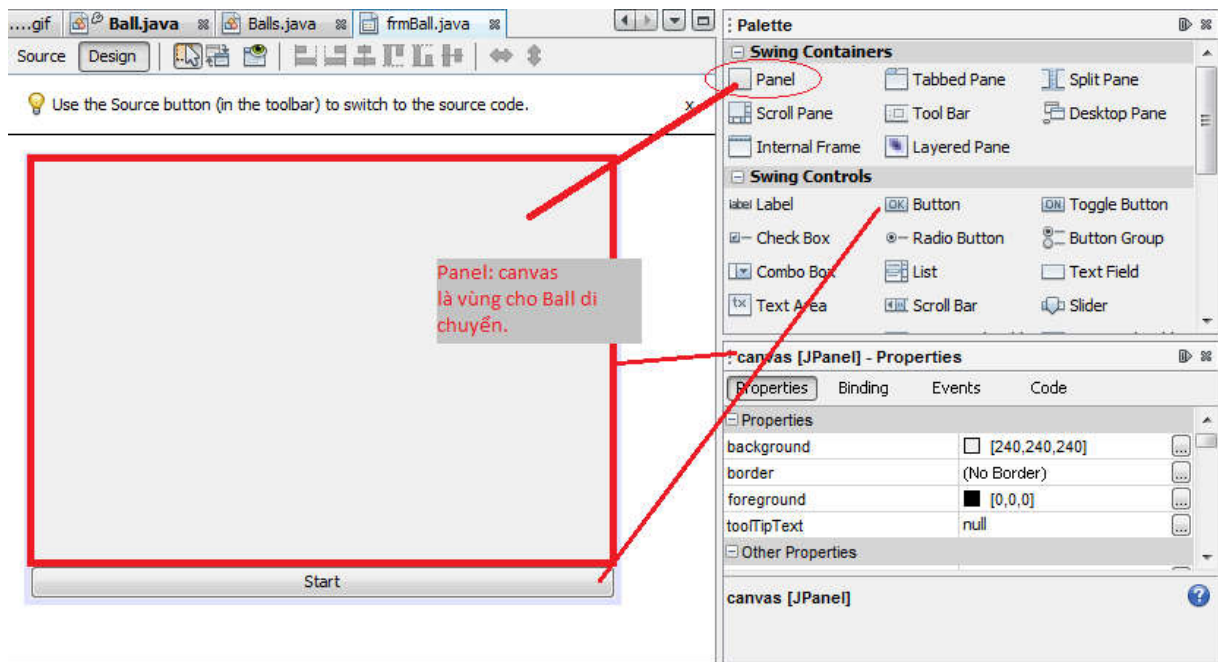
        x=0;
        dx=-dx;
    }
    if(x+XSIZE>=d.getWidth()){
        x=d.width-XSIZE;
        dx=-dx;
    }
    if(y<0){
        y=0;
        dy=-dy;
    }

    if(y+YSIZE>=d.getHeight()){
        y=d.height-YSIZE;
        dy=-dy;
    }
    g.fillOval(x, y, dx, XSIZE);
    g.dispose();
}

public void bounce(){
    draw();
    for(int i=0;i<1000;i++){
        move();
        try{
            Thread.sleep(5);
        }catch(InterruptedException ex){
            JOptionPane.showMessageDialog(null, ex.toString(), "Thông báo
            lỗi", JOptionPane.ERROR_MESSAGE);
        }
    }
}
}

```

Bước 2: Tạo Form có giao diện như sau:



Bước 3: Xử lý sự kiện cho button Start:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    Ball b1=new Ball(canvas);
    b1.bounce();// TODO add your handling code here:
}
```

Bước 4: Chạy thử ứng dụng bằng cách nhấp nhiều lần vào button Start.

4.2.2 Bài thực hành 2

Viết chương trình sử dụng MultiThread để tạo ra nhiều quả banh chạy trên màn hình.

Bước 1: Tạo ra lớp Balls.java

```
import java.awt.*;
import javax.swing.*;

public class Balls extends Thread{
    private JPanel box;
    private static final int XSIZE=10;
    private static final int YSIZE=10;
    private int x=0;
    private int y=0;
    private int dx=2;
    private int dy=2;

    public Balls(JPanel p) {
        box=p;
    }

    public void draw(){
        Graphics g=box.getGraphics();
        g.fillOval(x, y, XSIZE, YSIZE);
        g.dispose();
    }

    public void move(){
        //xoa hinh cu bang cach ve de len
        Graphics g=box.getGraphics();
        g.setXORMode(Color.GREEN);
        g.fillOval(x, y, XSIZE, YSIZE);
        x+=dx;
        y+=dy;
        Dimension d=box.getSize();
        //kiemtra cos dung cac canh
        if(x<0){
            x=0;
            dx=-dx;
        }
        if(x+XSIZE>=d.getWidth()){
            x=d.width-XSIZE;
            dx=-dx;
        }
        if(y<0){
```

```

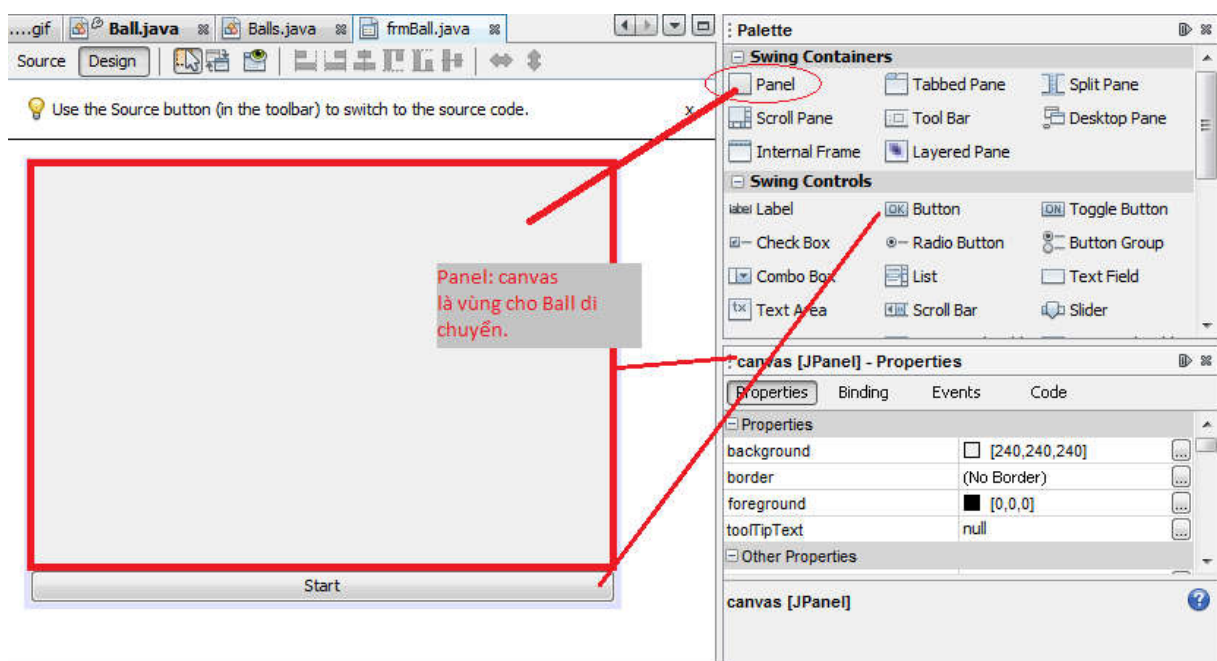
        y=0;
        dy=-dy;
    }

    if(y+YSIZE>=d.getHeight()){
        y=d.height-YSIZE;
        dy=-dy;
    }
    g.fillOval(x, y, dx, XSIZE);
    g.dispose();
}

public void run(){
    draw();
    for(int i=0;i<5000;i++){
        move();
        try{
            sleep(1);
        }catch(InterruptedException ex){
            JOptionPane.showMessageDialog(null, ex.toString(), "Thông báo lỗi", JOptionPane.ERROR_MESSAGE);
        }
    }
}
}

```

Bước 2: Tạo Form có giao diện như sau:



Bước 3: Xử lý sự kiện cho button Start:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    Balls b=new Balls(canvas);  
    b.start();// TODO add your handling code here:  
}
```

Bước 4: Chạy thử ứng dụng bằng cách nhấp nhiều lần vào button Start.

4.3 BÀI TẬP LÀM THÊM

4.3.1 Bài tập 01

Viết chương trình tạo ra một đối tượng FileTWrite kế thừa đối tượng Thread (hoặc cài đặt giao diện Runnable), cho phép viết một dãy số ngẫu nhiên vào tập tin. Đối tượng FileWriter có thuộc tính tên tập tin cần viết. Viết hàm main tạo ra 3 đối tượng viết 3 tập tin chạy ở 3 tiến trình đồng thời riêng biệt.

4.3.2 Bài tập 02

Viết chương trình tạo ra một đối tượng FileReader kế thừa đối tượng Thread (hoặc cài đặt Runnable), cho phép đọc nội dung một tập tin và hiển thị lên màn hình. Đối tượng FileReader có thuộc tính tên tập tin cần mở. Viết hàm main tạo ra 3 đối tượng đọc 3 tập tin chạy ở 3 tiến trình đồng thời riêng biệt.

4.3.3 Bài tập 03

Viết chương trình tạo ra đối tượng đọc tập tin, viết tập tin chạy ở từng tiến trình riêng (có xử lý đồng bộ hóa dữ liệu). Viết hàm main tạo ra đối tượng viết và đọc dữ liệu với tập tin giống nhau. Kiểm tra việc đồng bộ hóa dữ liệu.

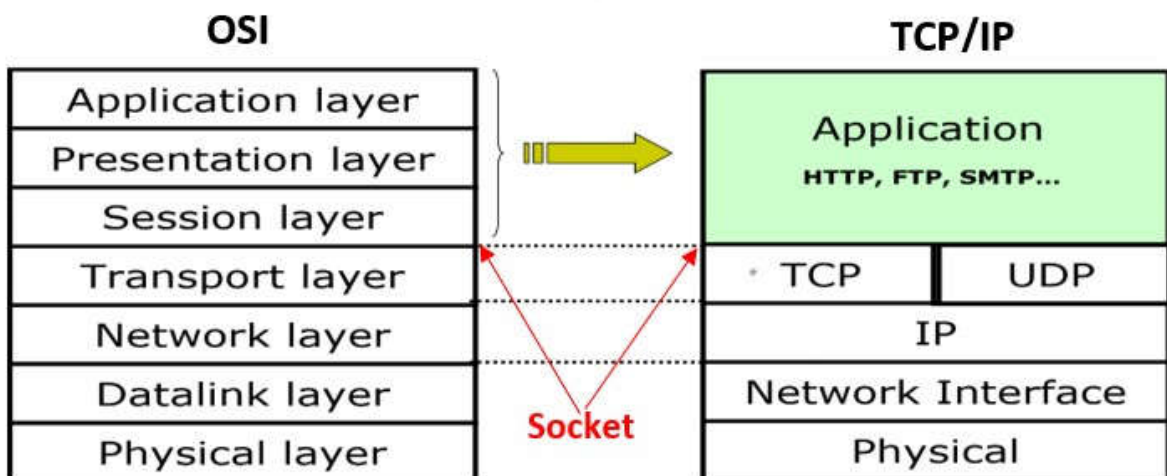
BÀI 5. TCP Socket, FTP

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về TCP Socket, FTP, mô hình kết nối client-server; biết cách khởi tạo, xây dựng các ứng dụng kiểu client-server thông qua 2 giao thức TCP, FTP bằng ngôn ngữ Java.

5.1 TCP SOCKET TRONG JAVA

5.1.1 Socket

Trong hệ thống mạng máy tính tồn tại những mô hình tham chiếu có kiến trúc phân tầng (OSI, TCP/IP...) nhằm hỗ trợ chức năng trao đổi thông tin giữa các ứng dụng ở nhiều máy tính khác nhau.



Hình 1: Socket trong mô hình OSI và TCP/IP

Dữ liệu bên gửi sẽ được đóng gói (Encapsulation) từ tầng trên đến tầng cuối là tầng vật lý (Physical Layer), sau đó nhờ tầng vật lý này chuyển dữ liệu đến tầng vật lý máy bên nhận, bên nhận tiến hành giải mã (decapsulation) gói dữ liệu từ tầng dưới lên tầng trên cùng, là tầng ứng dụng (application layer).

Ở đây, Socket chính là cửa giao tiếp giữa tầng ứng dụng và tầng giao vận (Transport layer). Nói cách khác, Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lí bởi hệ điều hành qua đó các ứng dụng có thể gửi/nhận thông điệp đến/từ các ứng dụng khác. Ở đó, Socket sẽ được ràng buộc với một mã số cổng (Port Number) để giúp tầng giao vận định danh được ứng dụng nhận/gửi thông điệp.

Với ngôn ngữ lập trình Java, chúng ta được cung cấp 3 loại khác nhau của sockets:

- **Stream Socket (TCP):** Tạo luồng dữ liệu hai chiều, đáng tin cậy, có trình tự và không trùng lặp, dữ liệu chỉ được gửi/nhận khi có đã có liên kết. Dùng với Socket Class của java.
- **Datagram Socket (UDP):** Có thể nhận dữ liệu không theo trình tự, trùng lặp. Dùng với DatagramSocket Class.
- **Multicast Socket:** cho phép dữ liệu được gửi đến nhiều bên nhận một lúc. Dùng với DatagramSocket Class.

5.1.2 Lập trình TCP Socket trong Java

Đúng như tính chất của TCP chúng ta cần có liên kết 2 chiều trước khi server và client có thể trao đổi thông điệp với nhau.

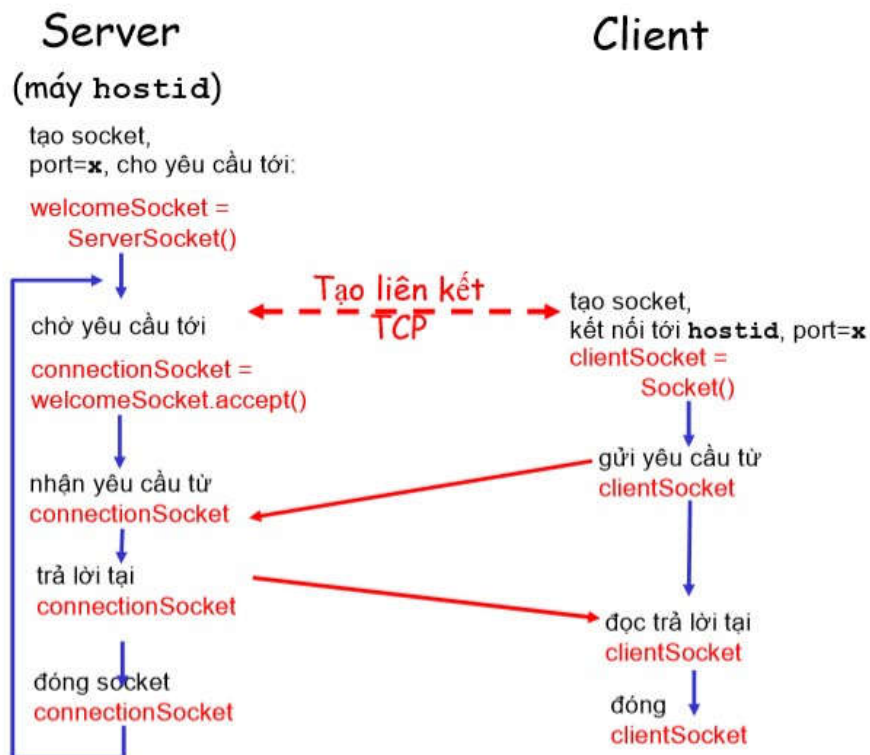
Ban đầu, phía server tạo Socket được ràng buộc với một cổng (port number) để chờ nhận yêu cầu từ phía client.

Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến trình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server.

TCP cung cấp dịch vụ truyền dòng tin cậy và có thứ tự giữa client và server, giữa máy chủ và máy nhận chỉ có 1 địa chỉ IP duy nhất. Thêm vào đó, mỗi thông điệp truyền đi đều có xác nhận trả về.

02/07/2023 10:45:00

1

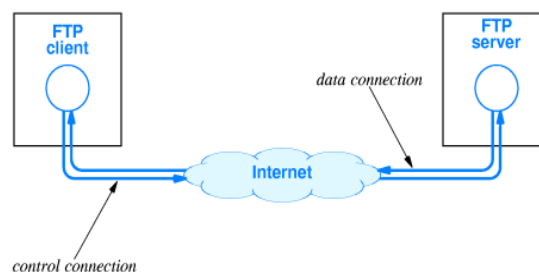


Hình 2: Hoạt động của TCP Socket trong mô hình client-server

5.2 FTP:

5.2.1 Giao thức FTP

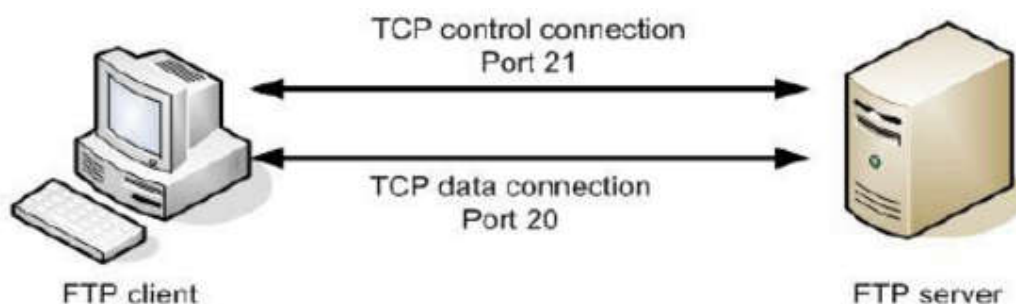
FTP - File Transfer Protocol (Giao thức truyền tải tập tin) được dùng trong việc trao đổi dữ liệu trong mạng thông qua giao thức TCP/IP, thường hoạt động trên 2 cổng là 20 và 21. Với giao thức này, các máy client trong mạng có thể truy cập đến máy chủ FTP để gửi hoặc lấy dữ liệu. Điểm nổi bật là người dùng có thể truy cập vào máy chủ FTP để truyền và nhận dữ liệu dù đang ở xa.



Hình 3: Giao thức FTP

5.2.2 Hoạt động của FTP

Giao thức FTP hoạt động dựa trên mô hình cơ bản của việc truyền và nhận dữ liệu từ máy Client đến máy Server. Quá trình truyền nhận dữ liệu giữa máy Client và Server lại được tạo nên từ 2 tiến trình TCP logic là Control Connection và Data Connection.



Hình 4: FTP gồm 2 đường: kiểm soát và dữ liệu

- **Control Connection:** Đây là phiên làm việc TCP logic đầu tiên được tạo ra khi quá trình truyền dữ liệu bắt đầu. Tuy nhiên, tiến trình này chỉ kiểm soát các thông tin điều khiển đi qua nó, ví dụ như các tập lệnh. Quá trình này sẽ được duy trì trong suốt quá trình phiên làm việc diễn ra.
- **Data Connection:** Khác với tiến trình Control Connection, Data Connection là một kết nối dữ liệu TCP được tạo ra với mục đích chuyên biệt là truyền tải dữ liệu giữa máy Client và máy Server. Kết nối sẽ tự động ngắt khi quá trình truyền tải dữ liệu hoàn tất.

5.2.3 Các phương thức truyền dữ liệu trong FTP

Khi quá trình truyền dữ liệu được thiết lập, dữ liệu sẽ được truyền từ máy Client đến máy Server hoặc có thể ngược lại. Dựa trên việc truyền dữ liệu này, FTP có 3 phương thức truyền tải dữ liệu là stream mode, block mode, và compressed mode.

- **Stream mode:** Phương thức này hoạt động dựa vào tính tin cậy trong việc truyền dữ liệu trên giao thức TCP. Dữ liệu sẽ được truyền đi dưới dạng các byte có cấu

trúc không liên tiếp. Thiết bị gửi chỉ đơn thuần đẩy luồng dữ liệu qua kết nối TCP tới phía nhận mà không có một trường tiêu đề nhất định.

- **Block mode:** Là phương thức truyền dữ liệu mang tính quy chuẩn hơn. Với phương thức này, dữ liệu được chia thành nhiều khối nhỏ và được đóng gói thành các FTP blocks. Mỗi block sẽ chứa thông tin về khối dữ liệu đang được gửi.
- **Compressed mode:** Phương thức truyền sử dụng kỹ thuật nén dữ liệu khá đơn giản là "run-length encoding". Với thuật toán này, các đoạn dữ liệu bị lặp sẽ được phát hiện và loại bỏ để giảm chiều dài của toàn bộ thông điệp khi gửi đi.

5.3 BÀI TẬP THỰC HÀNH

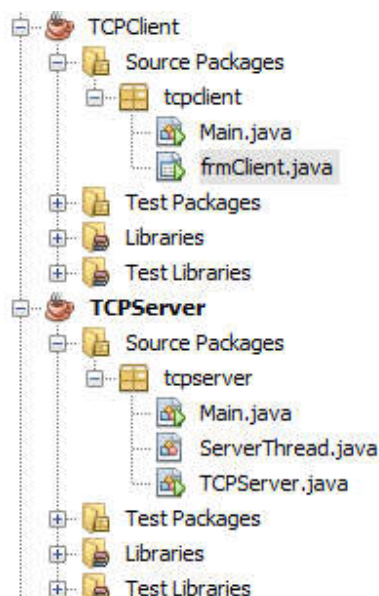
5.3.1 Bài thực hành 01

Viết chương trình giao tiếp giữa client và server sử dụng giao thức TCP, thực hiện các chức năng sau:

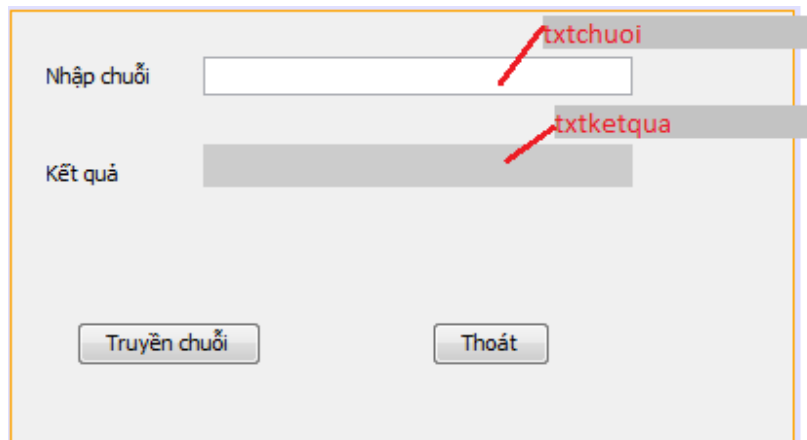
- Client truyền 1 chuỗi lên server, server nhận chuỗi này và chuyển nó thành chữ in hoa sau đó gửi trả kết quả cho client.
- Client nhận kết quả rồi sau đó xuất ra màn hình kết quả vừa nhận.

Yêu cầu: Sử dụng Multithread để server có thể giao tiếp được với nhiều client cùng lúc.

Bước 1: Tạo 2 project mới là TCPServer và TCPClient



Bước 2: Tạo Form frmClient có giao diện như sau:



Thêm khai báo vào đầu lớp frmClient

```
package tcpclient;

import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

/**
 *
 * @author Tuan
 */
public class frmClient extends javax.swing.JFrame {
    private Socket socket=null;
    private PrintWriter out=null;
    private Scanner in=null;
    /** Creates new form frmClient */
    public frmClient() {
        initComponents();
        txtchuoi.requestFocus();
    }
}
```

Xử lý sự kiện cho button Truyền chuỗi.

```

private void btntruyenchuoiActionPerformed(java.awt.event.ActionEvent evt) {
    String chuoi=txtchuoi.getText();//Lấy chuỗi
    String ketqua="";
    try{
        //Socket nhận tham số là địa chỉ Host và port
        socket=new Socket("127.0.0.1",1234);
        out=new PrintWriter(socket.getOutputStream(),true);
        in=new Scanner(socket.getInputStream());
        out.println(chuoi);//truyền chuỗi lên server để xử lý
        ketqua=in.nextLine().trim();//nhận chuỗi kết quả từ server
        txtketqua.setText(ketqua);//Hiển thị chuỗi nhận được từ server lên TextField
        socket.close();//Đóng Socket
    }catch(Exception e)
    {
        try{if(socket!=null) socket.close();}catch(Exception ex){e.printStackTrace();}
        e.printStackTrace();
    }
}

```

Xử lý sự kiện cho button Thoát

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

```

Bước 3: Tạo lớp TCPServer như sau:

```

public class TCPServer {
    static final int PORT = 1234;
    private ServerSocket server=null;
    public TCPServer(){
        try{
            server=new ServerSocket(PORT);
        }catch(Exception e){e.printStackTrace();}
    }
    public void action(){
        Socket socket=null;
        int i=0;
        System.out.println("Serverlistening...");
        try{
            while((socket=server.accept())!=null){
                new ServerThread(socket,"Client#"+i);
                System.out.printf("Thread for Client#%d generating...\n",i++);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    public static void main(String[]args){
        new TCPServer().action();
    }
}

```

Bước 4: Tạo lớp ServerThread với nội dung như sau.

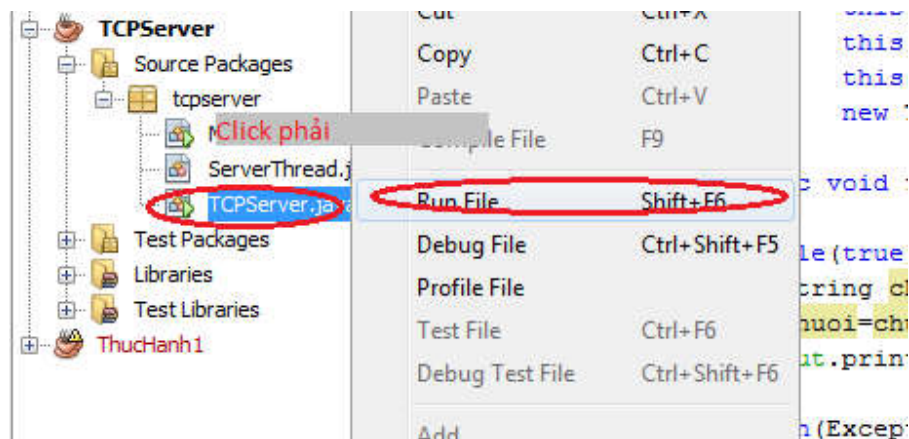
```

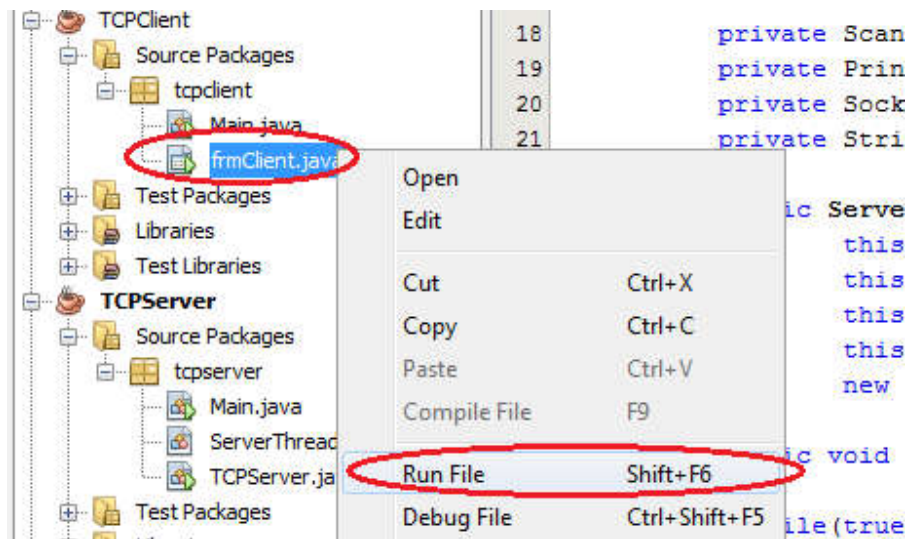
public class ServerThread implements Runnable {
    private Scanner in=null;
    private PrintWriter out=null;
    private Socket socket;
    private String name;

    public ServerThread(Socket socket,String name) throws IOException{
        this.socket=socket;
        this.name=name;
        this.in=new Scanner(this.socket.getInputStream());
        this.out=new PrintWriter(this.socket.getOutputStream(),true);
        new Thread(this).start();
    }
    public void run() {
        try{
            while(true){
                String chuoi=in.nextLine().trim();
                chuoi=chuoi.toUpperCase();
                out.println(chuoi);
            }
        }catch(Exception e){
            System.out.println(name+" has departed");
        }finally{
            try{socket.close();}catch(IOException e){}
        }
    }
}

```

Bước 5: Chạy thử ứng dụng.





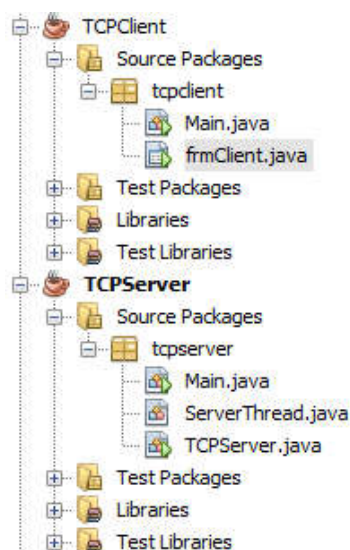
5.3.2 Bài thực hành 02

Viết chương trình giao tiếp giữa client và server sử dụng giao thức TCP, thực hiện các chức năng sau:

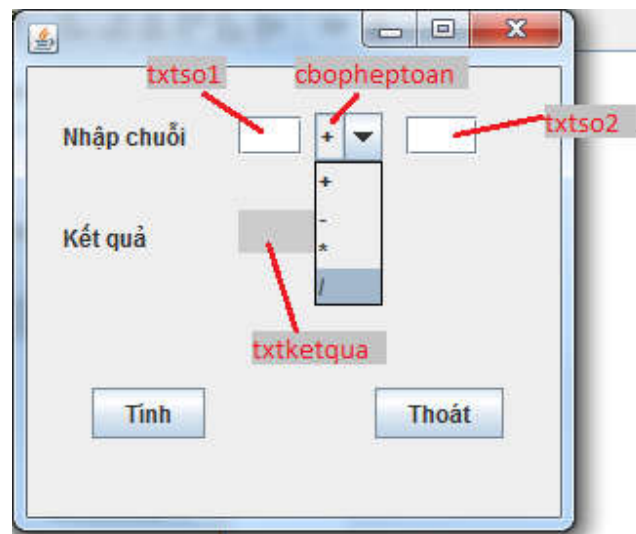
- Client truyền hai số nguyên và phép toán (cộng, trừ, nhân, chia) lên server.
- Server sau khi nhận được thì thực hiện phép toán giữa hai số nguyên và trả kết quả về cho client, client nhận lại kết quả và xuất ra màn hình.

Yêu cầu: Sử dụng Multithread để server có thể giao tiếp được với nhiều client cùng lúc.

Bước 1: Tạo 2 project mới là TCPServer và TCPClient



Bước 2: Tạo Form frmClient có giao diện như sau:



Thêm khai báo vào đầu lớp frmClient

```
package tcpclient;

import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

/**
 *
 * @author Tuan
 */
public class frmClient extends javax.swing.JFrame {
    private Socket socket=null;
    private PrintWriter out=null;
    private Scanner in=null;
    /** Creates new form frmClient */
    public frmClient() {
        initComponents();
        txtchuoi.requestFocus();
    }
}
```

Xử lý sự kiện cho button Tính.

```

private void btntinhActionPerformed(java.awt.event.ActionEvent evt) {
    int so1 =Integer.parseInt(txtso1.getText()); //Lấy số hạng thứ nhất
    int so2 =Integer.parseInt(txtso2.getText()); //Lấy số hạng thứ hai
    String pheptoan=cbopheptoan.getSelectedItem().toString(); //Lấy phép toán
    String chuoi=so1+"@"+pheptoan+"@"+so2; //Ghép thành chuỗi phân cách nhau bởi dấu @
    String ketqua="";
    try{
        //Socket nhận tham số là địa chỉ Host và port
        socket=new Socket("127.0.0.1",1234);
        out=new PrintWriter(socket.getOutputStream(),true);
        in=new Scanner(socket.getInputStream());
        out.println(chuoi); //truyền chuỗi lên server để xử lý
        ketqua=in.nextLine().trim(); //nhận chuỗi kết quả từ server
        txtketqua.setText(ketqua); //Hiển thị chuỗi nhận được từ server lên TextField
        socket.close(); //Đóng Socket
    }catch(Exception e)
    {
        try{if(socket!=null)socket.close();}catch(Exception ex){e.printStackTrace();}
        e.printStackTrace();
    }
}

```

Xử lý sự kiện cho button Thoát

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

```

Bước 3: Tạo lớp TCPServer như sau:

```

public class TCPServer {
    static final int PORT = 1234;
    private ServerSocket server=null;
    public TCPServer() {
        try{
            server=new ServerSocket(PORT);
        }catch(Exception e){e.printStackTrace();}
    }
    public void action(){
        Socket socket=null;
        int i=0;
        System.out.println("Serverlistening...");
        try{
            while((socket=server.accept())!=null){
                new ServerThread(socket,"Client#" +i);
                System.out.printf("Thread for Client%d generating...\n",i++);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    public static void main(String[]args){
        new TCPServer().action();
    }
}

```

Bước 4: Tạo lớp ServerThread với nội dung như sau.

```

import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

/**
 *
 * @author Tuan
 */
public class ServerThread implements Runnable {
    private Scanner in=null;
    private PrintWriter out=null;
    private Socket socket;
    private String name;

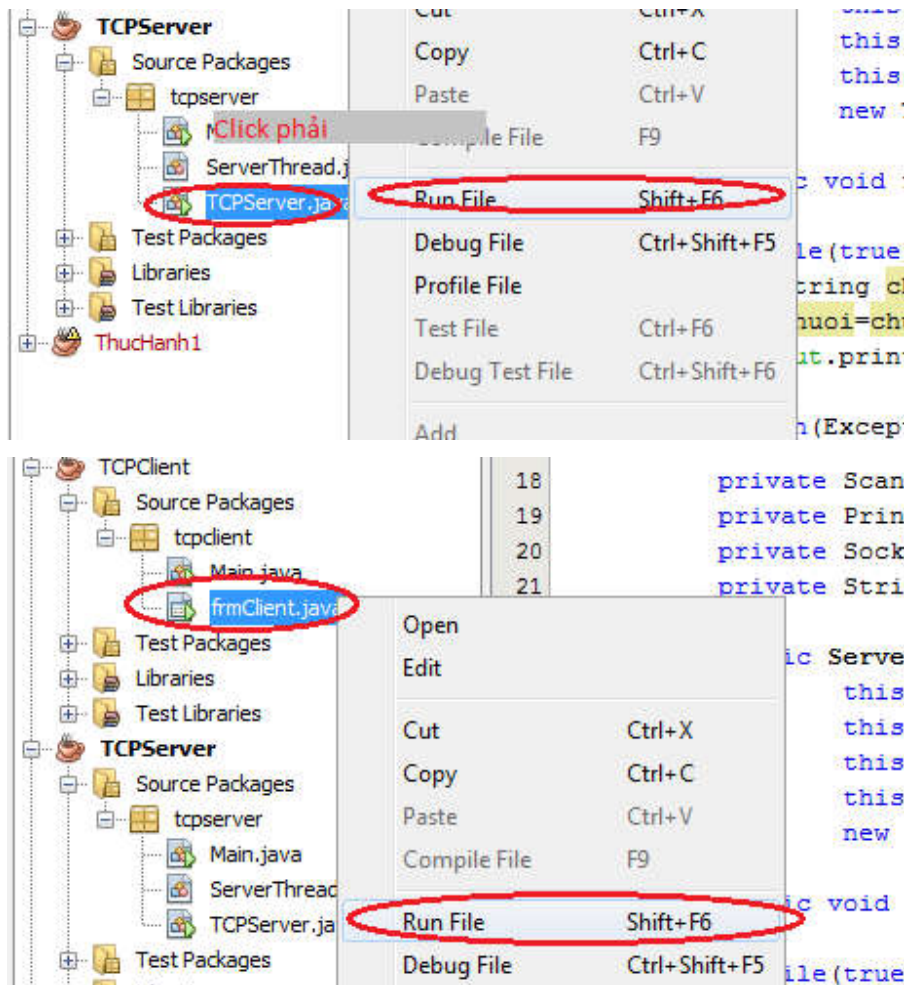
    public ServerThread(Socket socket,String name) throws IOException{
        this.socket=socket;
        this.name=name;
        this.in=new Scanner(this.socket.getInputStream());
        this.out=new PrintWriter(this.socket.getOutputStream(),true);
        new Thread(this).start();
    }

    public void run(){
        try{
            while(true){
                String chuoi=in.nextLine().trim();
                Scanner sc=new Scanner(chuoi);
                sc.useDelimiter("@");
                int so1=sc.nextInt();
                String pheptoan=sc.next();
                int so2=sc.nextInt();

                if(pheptoan.equals("+"))
                    out.println(so1+so2);
                else if(pheptoan.equals("-"))
                    out.println(so1-so2);
                else if(pheptoan.equals("*"))
                    out.println(so1*so2);
                else if(pheptoan.equals("/"))
                    out.println((float)so1/so2);
            }
        }catch(Exception e){
            System.out.println(name+" has departed");
        }finally{
            try{socket.close();}catch(IOException e){}
        }
    }
}

```

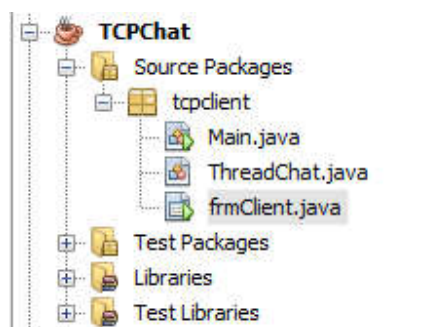
Bước 5: Chạy thử ứng dụng.



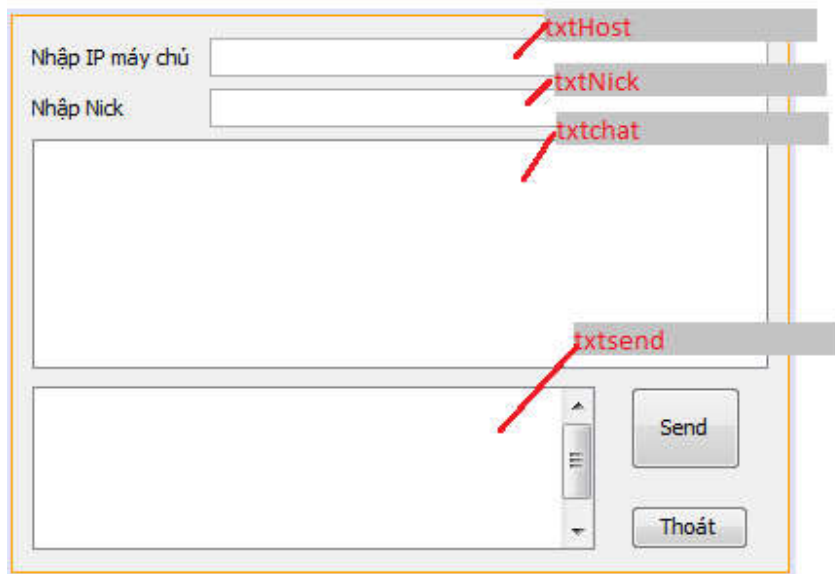
5.3.3 Bài thực hành 03

Viết chương trình cho phép hai máy chat với nhau.

Bước 1: Tạo Project mới TCPChat



Bước 2: Tạo Form frmClient có giao diện như sau:



Thêm các khai báo cho frmClient

```
package tcpclient;

import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

/**
 *
 * @author Tuan
 */
public class frmClient extends javax.swing.JFrame {
    private Socket socket=null;
    private PrintWriter out=null;
    private Scanner in=null;
    /** Creates new form frmClient */
    public frmClient() {
        initComponents();
        txtHost.requestFocus();
        ThreadChat obj=new ThreadChat();
        obj.chat=this;
    }
}
```

Thêm sự kiện cho button Thoát, Viết thêm 1 hàm Hienthi


```
private void btnthoatActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

Thêm sự kiện cho button Thoát

```
public void Hienthi (String str) {
    txtchat.append(str);
}
```

Viết thêm hàm Hienthi để phục vụ cho việc nhận tin nhắn từ máy khác đến.

Thêm sự kiện cho button Send

```
private void btnsendActionPerformed(java.awt.event.ActionEvent evt) {
    String chuoi=txtsend.getText();
    String nick=txtNick.getText();
    String host=txtHost.getText();
    try{
        //Socket nhận tham số là địa chỉ Host và port
        socket=new Socket(host,1234);
        out=new PrintWriter(socket.getOutputStream(),true);
        in=new Scanner(socket.getInputStream());
        out.println(nick+": "+chuoi+"\n");//truyền chuỗi lên server để xử lý
        txtchat.append(nick+": "+chuoi+"\n");
        socket.close();//Đóng Socket
    }catch(Exception e)
    {
        try{if(socket!=null)socket.close();}catch(Exception ex){e.printStackTrace();}
        e.printStackTrace();
    }
}
```

Bước 4: Tạo thêm lớp ThreadChat.java

```
public class ThreadChat implements Runnable {
    private Scanner in=null;
    private Socket socket=null;
    public frmClient chat=null;
    ServerSocket server=null;
    public ThreadChat() {
        try{
            server=new ServerSocket(1234);//Tạo ra Server socket mới nhận port 1234 làm tham số.
        }catch(Exception e){e.printStackTrace();}
        new Thread(this).start();//Khởi động Thread
    }
    public void run(){
        try{
            while(true){
                while((socket=server.accept())!=null){//Nhận kết nối từ máy khác đến
                    this.in=new Scanner(this.socket.getInputStream());
                    String chuoi=in.nextLine().trim();//Nhận chuỗi
                    chat.Hienthi(chuoi+"\n");//Hiển thị chuỗi ra màn hình
                }
            }
        }catch(Exception e){}
        finally{
            try{socket.close();}catch(IOException e){}
        }
    }
}
```

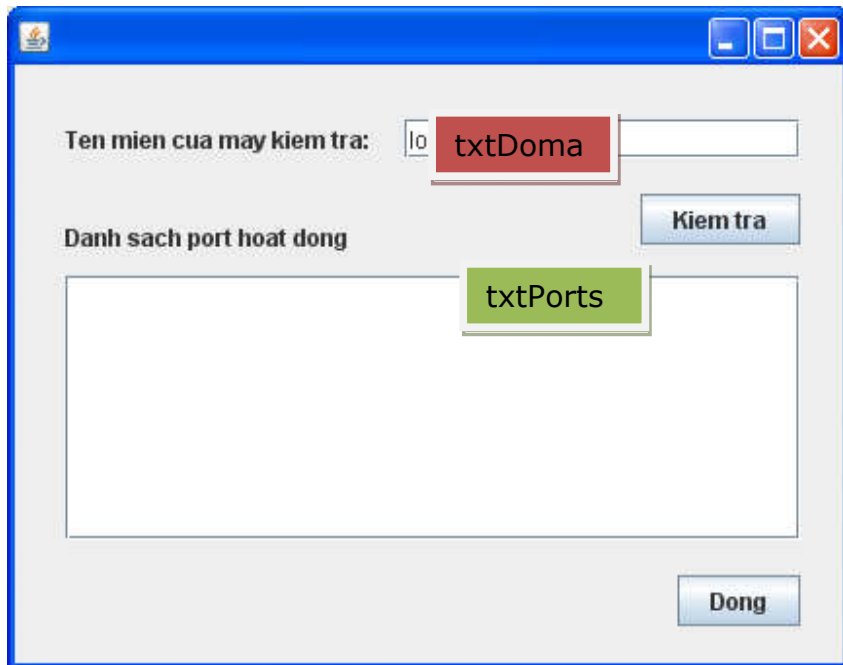
Bước 5: Chạy thử chương trình trên nhiều máy.

- Mở chạy formClient.
- Nhập địa chỉ máy cần chat, nick của mình.
- Mở chạy frmClient trên máy khác và nhập địa chỉ máy cần chat.

5.3.4 Bài thực hành 04

Viết chương trình kiểm tra các cổng từ 1024 đến 65536 (hoặc từ 0 – 1024) của Server bất kỳ, có cổng nào hoạt động không?

Hướng dẫn:



Viết code cho nút kiểm tra:

```
Socket s;  
for(int port=1024;port<65536;port++)  
    try {  
        s = new Socket(this.txtDomain.getText(), port);  
        this.txtPorts.setText(this.txtPorts.getText()+ " \n" + port);  
    } catch (UnknownHostException ex) {  
        JOptionPane.showMessageDialog(this,ex);  
    } catch (IOException e) {  
        //JOptionPane.showMessageDialog(this,e);  
    }  
}
```


5.3.5 Bài thực hành 05

Viết chương trình đọc thông tin Socket tại máy cục bộ và thông tin Socket của máy kết nối trên cổng 80.

(Sinh viên tự thực hiện dựa vào kiến thức đã học)

5.3.6 Bài thực hành 06

Viết chương trình thực hiện mô phỏng giao thức FTP như sau:

- Chương trình client: thực hiện chức năng gửi yêu cầu đường dẫn tên tập tin tới Server sau đó đợi đáp trả từ server dữ liệu liên quan đến tập tin và hiển thị thông tin dữ liệu tập tin lên màn hình và lưu thành tập tin phía máy client nếu tìm thấy ngoài ra hiển thị chuỗi lỗi.
- Chương trình server: có nhiệm vụ lắng nghe kết nối từ client, tìm và mở tập tin liên quan theo yêu cầu của client viết về cho client sau đó đóng luồng lại.

Hướng dẫn

Viết chương trình FTP Server trước.

Bước 1. Viết chương trình quản lý user như ở bài tập phần 1. Cho phép thêm user mới vào cơ sở dữ liệu user.

Bước 2. Viết code cho FTP lắng nghe giao tiếp với client, tuần tự tại mỗi thời điểm xử lý cho 1 client.

```
package ftpservervd;

import java.io.File;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import javax.swing.JOptionPane;
public class Main {
    //khai bao cac hang trong giao thuc giao tiep
    public static final int DANGNHAP=1;
    public static final int KHONGLALENH=0;
```

```
public static final int DANGNHAPKHONGTHANHCONG=0;
public static final int DANGNHAPTHANHCONG=1;
public static final int THOAT=2;
//ham doi chuoai giao tiep thanh hang cho de xu ly
public static int laLenh(String cmd){
    if(cmd.equals("DANGNHAP"))
        return DANGNHAP;
    return KHONGLALENH;
}
//thiet lap port giao tiep cua ung dung, FTP co port la 20 va 21
//vi du chon port 10000
public static final int PORT = 10000;
public static void main(String[] args) {
    //gia su co user, pass, path
    //sinh vien thay 1 user nay bang cach truy xuất cơ sở dữ liệu
    // các user đã tạo ra ở bước 1 do chương trình quản lý user
    String userA="tu";
    String passA="tu";
    String path="C:/";
    ServerSocket s;
    try {
        s = new ServerSocket(PORT);
        while (true) {
            Socket new_s = s.accept();
            //nhận lệnh giao tiếp từ client
            boolean lap=true;
            while(lap)
            {
                String cmd;
                Scanner sc;
                cmd=sc.nextLine();
                //điều phối sự kiện yêu cầu ở phía client
```

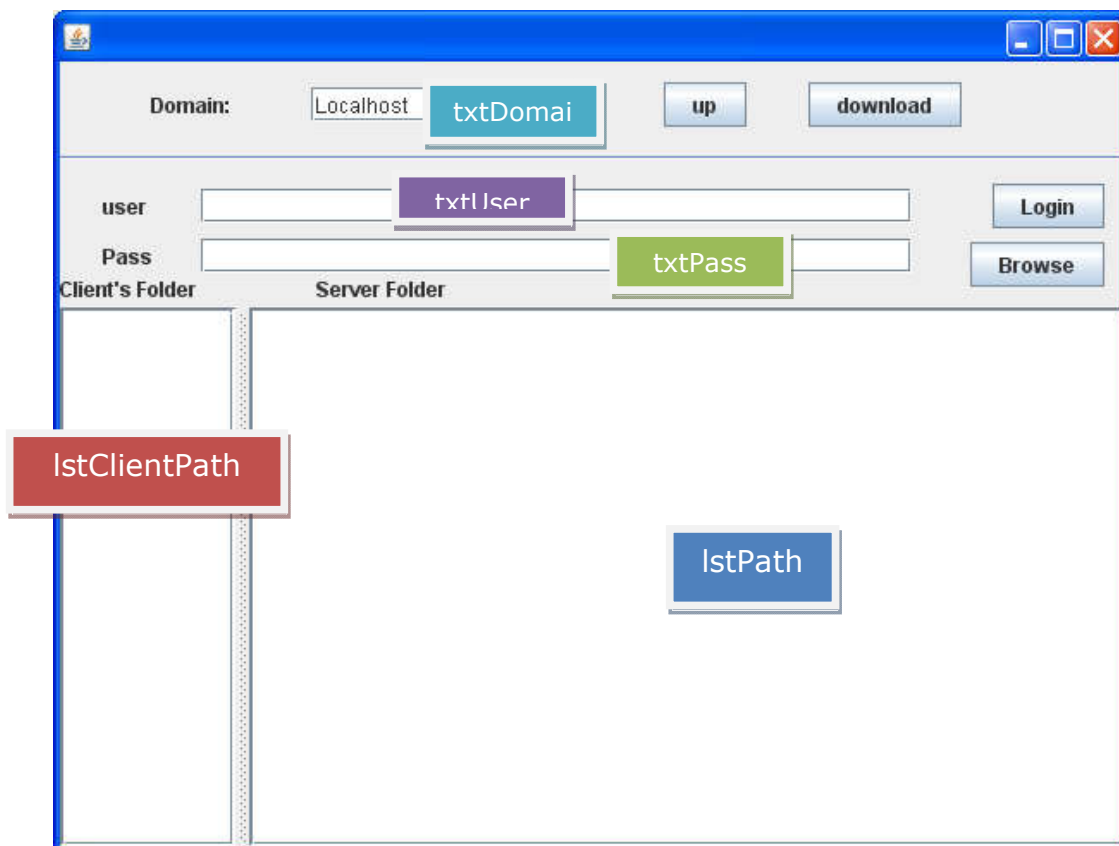
```
switch (laLenh(cmd)){
    case DANGNHAP:
        String user=sc.nextLine();
        String pass=sc.nextLine();
        PrintWriter pw;
            pw=new PrintWriter(new_s.getOutputStream());
        if(user.equals(userA) && pass.equals(passA)){
            pw.println(DANGNHAPTHANHCONG);
            //mo thu muc len goi ve cho client
            File dir=new File(path);
            File dsFile[]=dir.listFiles();
            if(dsFile==null)
            {
                JOptionPane.showMessageDialog(null, "Đường dẫn không
đúng hay không phải thư mục!");
            }else{
                pw.println(dsFile.length);
                for(int i=0;i<dsFile.length;i++)
                    pw.println(dsFile[i].getName());
            }
        }else
        {
            //goi ve khong mo duoc
            pw.println(DANGNHAPKHONGTHANHCONG);
            pw.println("Dang nhap ko thanh cong");
        }
        pw.flush();
        break;
        case THOAT:
            lap=false;
            break;
    }
```

```

    }
    new_s.close();
}
} catch (Exception ex) {
    System.out.println(ex);
}
}
}

```

Bước 3. Tạo giao diện chương trình client truy cập đến server.



Bước 4. Viết code cho nút login.

```

Socket s;
public static final int PORT = 10000;
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //tao socket
    String domain=this.txtDomain.getText();

```

```
try {
    InetAddress ia = InetAddress.getByName(domain);

    try {
        s = new Socket(ia, PORT);
        //goi user, pass len server
        PrintWriter pw;
        //lay du lieu tu form do su dung go vao
        String user=this.txtUser.getText();
        String pass=this.txtPass.getText();
        String cmd="DANGNHAP";
        pw.println(cmd);
        pw.println(user);
        pw.println(pass);
        System.out.println(cmd);
        System.out.println(user);
        System.out.println(pass);
        pw.flush();
        //client doi phan hoi tu server
        Scanner sc=new Scanner(s.getInputStream());
        int cmdR=sc.nextInt();
        if(cmdR==1){
            JOptionPane.showMessageDialog(this, "Dang nhap thanh cong");
            //luu duong dan hien tai ma list hien thi danh tap tin trong folder do.
            DefaultListModel dm=new DefaultListModel();
            int n=sc.nextInt();
            for(int i=0;i<n;i++){
                dm.addElement(sc.nextLine());
            }
            this.lstPath.setModel(dm);
        }else
            JOptionPane.showMessageDialog(this, "Dang nhap khong thanh cong");
```

```
    } catch (IOException ex) {  
        JOptionPane.showMessageDialog(this, ex.toString());  
    }  
  
    } catch (UnknownHostException ex) {  
        JOptionPane.showMessageDialog(this, ex.toString());  
    }  
  
}
```

Bước 5: viết code cho nút Browse

```
String path;  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    fchPath.setVisible(true);  
    //thiet lap  
    fchPath.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);  
    //kiem tra form nay duoc chon nut gi?  
    if(fchPath.showOpenDialog(this)==JFileChooser.APPROVE_OPTION)  
    try{  
        path=fchPath.getSelectedFile().getCanonicalPath();  
        File dir=new File(path);  
        File dsFile[]=dir.listFiles();  
        if(dsFile==null)  
        {  
            JOptionPane.showMessageDialog(null, "sai duong dan!");  
        }else{  
            try{  
                //luu duong dan hien tai ma list hien thi danh tap  
                // tin trong folder do.  
                DefaultListModel dm=new DefaultListModel();  
                for(int i=0;i<dsFile.length;i++){  
                    String filename=dsFile[i].getName();  
                    dm.addElement(filename);  
                }  
            }  
        }  
    }  
}
```

```

    }
    this.lstClientPath.setModel(dm);
    }catch(Exception e)
    {JOptionPane.showMessageDialog(null,e.toString());}
    }

    }catch(Exception e)
    { }
}

```

Bước 6: Viết code cho nút upload

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String fileName=(String) this.lstClientPath.getSelectedValue();
    String cpath=path+"\" + fileName;
    System.out.println(cpath);
    try{
        PrintWriter pw;
        //goi tin hieu lenh
        pw.println("UPLOAD");
        pw.flush();
        System.out.println("Da goi lenh upload len server");
        pw.println(fileName);
        pw.flush();

        System.out.println("Da goi ten tap tin len server");
        BufferedOutputStream bos=new
BufferedOutputStream(s.getOutputStream());
        DataOutputStream dos=new DataOutputStream(bos);
        //mo tap tin ra
        BufferedInputStream bis=new BufferedInputStream (new
FileInputStream(cpath));
        //lap doc noi dung tap tin va goi lieu len server

```

```
byte buf[]=new byte[bis.available()];
//tao bo dem doc het du lieu tu tap tin vao bo dem roi day
//vao luong len server.
bos.write(bis.read(buf));
System.out.println("da goi du lieu tap tin len server");
bos.flush();
//bis.close();
//doi nhan danh sach tap thu cua folder o server voi tinh trang moi
Scanner sc=new Scanner(s.getInputStream());
String cmd=sc.nextLine();
System.out.println("da nhan dap tra tu server");
if(cmd.equals("DANHAN"))
    JOptionPane.showMessageDialog(null, "Đã gửi tập tin thành công");
else
    JOptionPane.showMessageDialog(null, "Gửi tập tin thất bại");
//nhan update
updateFolderServer();

}catch(Exception e){
    System.out.println(e);
}
}
```

Hàm cập nhật danh sách tập tin trong Folder của server sau khi upload

```
public void updateFolderServer(){
    try{

        BufferedInputStream bi;
        Scanner sc=new Scanner(bi);
        DefaultListModel dm=new DefaultListModel();
        //server goi ve so luong tap tin thu muc sau khi upload
        int n=sc.nextInt();
```



```
System.out.println("Da nhan duoc so luong tap tin goi tu server");
//nhan ten tung tap tin thu muc
for(int i=0;i<n;i++)
{
    String filename=sc.nextLine();
    dm.addElement(filename);
}
System.out.println("Da hien thi xong danh sach tap tin");
//hien thi len form
this.lstPath.setModel(dm);
//ve lai giao dien
this.validate();
}catch(Exception e){
    JOptionPane.showMessageDialog(null,"Lỗi",
e.toString(),JOptionPane.ERROR_MESSAGE);
}
}
```

Bước 7: Viết code server đáp trả lại client sau khi nhận tập tin upload thành công.

Định nghĩa thêm hằng phía server.

```
public static final int UPLOAD=3;
```

Cập nhật lại hàm ánh xạ lệnh dạng chuỗi sang số.

```
public static int laLenh(String cmd){
    if(cmd.equals("DANGNHAP"))
        return DANGNHAP;
    if(cmd.equals("UPLOAD"))
        return UPLOAD;
    return KHONGLALENH;
}
```

Bổ sung vào lệnh xử lý switch của server.

```
case UPLOAD:
    System.out.println("Da vao lenh upload");
```

```
String fileName=sc.nextLine();
System.out.println("Da lay ten tap tin");
try{
    String path2;
    //kiem tra chuoai duong dan co dau / cuoi cung hay ko?
    //va gan ten tap tin tu client vao tuong ung
    if (path.lastIndexOf("/")>=path.length()-1)
        path2=path + fileName;
    else
        path2=path + "/" + fileName;
    System.out.println(path2);
    FileOutputStream fos=new FileOutputStream(new File(path2));
    BufferedOutputStream bos=new BufferedOutputStream(fos);
    BufferedInputStream bis;
    bis=new BufferedInputStream(new _s.getInputStream());
    byte buf[]=new byte[bis.available()];
    bos.write(bis.read(buf));
    bos.flush();
    bos.close();
    pw=new PrintWriter(new _s.getOutputStream());
    pw.println("DANHAN");
    pw.flush();
    //yeu cau update lai listbox o server
    //mo thu muc ra va tra ve noi dung thu muc o phia server
    Main.traThuMucClient(path,pw);
}
catch(Exception e)
{
    System.out.println(e);
}

break;
```

Đoạn code gửi danh sách tập tin của thư mục server sau khi upload.

```
static void traThuMucClient(String path, PrintWriter out) {  
    try{  
        File dir=new File(path);  
        File dsFile[];  
        System.out.println("Dang doc tap tin");  
        try{  
            dsFile = dir.listFiles();  
            System.out.println("da la ds tap tin");  
            out.println(dsFile.length);  
            for(int i=0;i<dsFile.length;i++){  
                String filename=dsFile[i].getName();  
                out.println(filename);  
            }  
            out.flush();  
            System.out.println("da goi client");  
        }catch(Exception e)  
            {JOptionPane.showMessageDialog(null,e.toString());}  
  
    } catch(Exception e){  
        System.out.println(e.toString());  
    }  
}
```

Bước 6: Viết code cho nút download

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    //lay tap tin can download  
    String fileName=(String) this.lstPath.getSelectedValue();  
    System.out.println(fileName);  
    try{  
        PrintWriter pw=new PrintWriter(s.getOutputStream());  
        //goi tin hieu lenh  
        pw.println("DOWNLOAD");  
    }
```

```
pw.flush();
System.out.println("Da goi lenh download len server");
pw.println(fileName);
pw.flush();
//doi server goi noi dung tap tin ve
System.out.println("Doi server goi noi dung tap tin ve");
String cpath=path+"\\\" + fileName;
FileOutputStream fos;
fos = new FileOutputStream(new File(cpath));
BufferedOutputStream bos=new BufferedOutputStream(fos);
BufferedInputStream bis=new BufferedInputStream(s.getInputStream());
byte buf[]=new byte[bis.available()];
bos.write(bis.read(buf));
bos.flush();
bos.close();
pw=new PrintWriter(s.getOutputStream());
pw.println("DANHAN");
pw.flush();
//cap nhat lai thu muc client vua download
this.capNhatClientFolder(cpath);
} catch (Exception ex) {
    System.out.println(ex);
}

}

private void capNhatClientFolder(String cpath) {
    //mo thu muc voi duong dan path ra
    File dir=new File(path);
    File dsFile[]=dir.listFiles();
    if(dsFile==null)
    {
        JOptionPane.showMessageDialog(null, " Duong dan sai!");
    }
}
```

```

    }else{
        try{
            //luu duong dan hien tai ma list hien thi danh tap tin trong folder do.
            //path=txtPath.getText();

            DefaultListModel dm=new DefaultListModel();
            for(int i=0;i<dsFile.length;i++){
                String filename=dsFile[i].getName();
                dm.addElement(filename);
            }
            this.lstClientPath.setModel(dm);
            this.validate();
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, e);
        }
    }
}

```

Bổ sung hằng ở server:

```
public static final int DOWNLOAD=4;
```

Cập nhật hàm ánh xạ lệnh dạng chuỗi sang số:

```

public static int laLenh(String cmd){
    if(cmd.equals("DANGNHAP"))
        return DANGNHAP;
    if(cmd.equals("UPLOAD"))
        return UPLOAD;
    if(cmd.equals("DOWNLOAD"))
        return DOWNLOAD;
    return KHONGLALENH;
}

```

Bổ sung trường hợp download cho lệnh switch

```

case DOWNLOAD:
    //lay ten tap tin do client gọi lên

```

```
System.out.println("Da vao lenh download");
String fileNameD=sc.nextLine();
System.out.println("Da lay ten tap tin");
try{
    String cpath;
    //kiem tra chuoai duong dan co dau / cuoi cung hay ko?
    //va gan ten tap tin tu client vao tuong ung
    if (path.lastIndexOf("/")>=path.length()-1)
        cpath=path + fileNameD;
    else
        cpath=path + "/" + fileNameD;
    System.out.println(cpath);
    //mo tap tin ra
    BufferedInputStream bis;
    bis=new BufferedInputStream (new FileInputStream(cpath));
    //lap doc noi dung tap tin va goi lieu len server
    byte buf[]=new byte[bis.available()];
    //tao bo dem doc het du lieu tu tap tin vao bo dem roi day
    //vao luong len server.
    BufferedOutputStream bos;
    bos=new BufferedOutputStream(new_s.getOutputStream());
    bos.write(bis.read(buf));
    System.out.println("da goi du lieu tap tin ve cho client");
    bos.flush();
    //doi nhan danh sach tap thu cua folder o server voi tinh trang moi
    Scanner scRequest=new Scanner(new_s.getInputStream());
    String cmdRequest=scRequest.nextLine();
    System.out.println("da nhan dap tra tu server");
    if(cmdRequest.equals("DANHAN"))
        System.out.println("Đã gửi tập tin thành công");
    else
        System.out.println("Gửi tập tin thất bại");
```

```
}catch(Exception e){  
    System.out.println(e);  
}  
break;
```

BÀI 6. UDP SOCKET

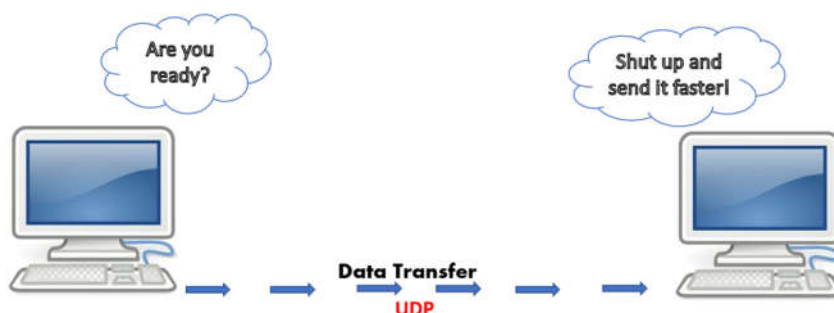
Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về UDP Socket, mô hình kết nối client-server; biết cách khởi tạo, xây dựng các ứng dụng kiểu client-server thông qua giao thức UDP bằng ngôn ngữ Java.

6.1 UDP SOCKET

6.1.1 Lập trình socket với UDP

UDP là viết tắt của User Datagram Protocol. UDP được giới thiệu vào năm 1980 và là một trong những giao thức mạng lâu đời nhất còn được sử dụng. UDP được sử dụng cho những đường truyền yêu cầu tốc độ cao và thời gian ngắn như video playback, game stream, game online, hay được sử dụng để tìm kiếm trong hệ thống tên miền (DNS).

Giao thức này giúp tăng tốc độ truyền tải vì nó không yêu cầu kết nối giữa máy gửi và máy nhận trước khi 2 phía có thể trao đổi thông điệp. Chính điều này UDP cung cấp đường truyền không tin cậy bằng gói dữ liệu/datagrams (group of bytes) trong khi đó TCP cung cấp đường truyền tin cậy bằng dòng byte, vì vậy UDP socket và TCP socket cũng có những khác biệt cơ bản.



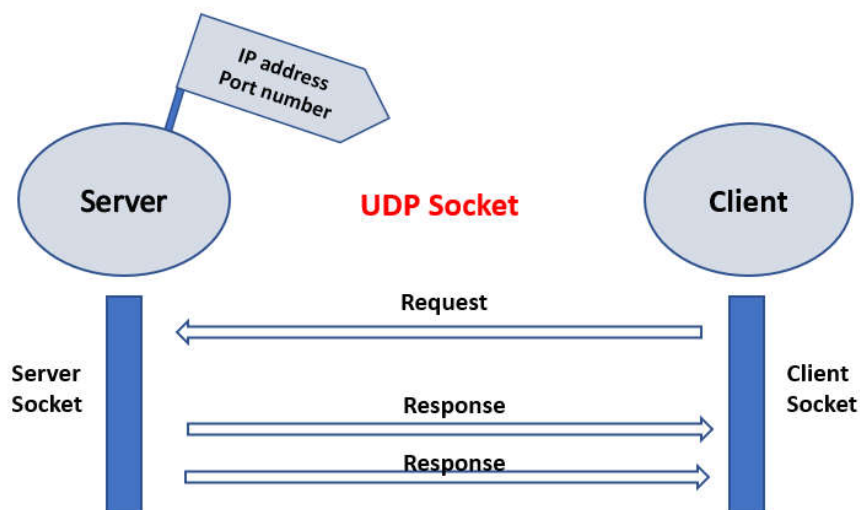
Hình 1: UDP Socket Data Tranfer

6.1.2 Hoạt động của UDP Socket

Với UDP socket, chúng ta không cần thiết lập liên kết "handshaking" giữa client và server trước khi truyền thông điệp, thông điệp được gửi đi một cách độc lập. Bên gửi sẽ chỉ rõ địa chỉ IP và số hiệu cổng (port number) của bên nhận. Sau khi thông điệp được truyền đến bên nhận, bên nhận có thể dựa vào địa chỉ IP và số hiệu cổng tương ứng của bên gửi được gắn trên gói tin để gửi lại response.

Chúng ta có thể thấy rằng việc truyền bằng gói tin có thể làm dữ liệu truyền đến không theo thứ tự hoặc bị mất, thêm nữa UDP không cung cấp chức năng kiểm soát lỗi, mất gói tin nhưng một phần lợi ích của việc truyền bằng gói tin chính là tốc độ và thời gian.

Trước khi lập trình socket UDP chúng ta cần biết quy trình hoạt động của mỗi bên client và server, giữa client và server của UDP socket có ít khác biệt hơn so với TCP socket.



Hình 2: Hoạt động của UDP Socket

Với UDP ta không cần thiết lập liên kết 2 chiều giữa client và server.

- **Về phía client:**

- Cần biết được địa chỉ IP cũng như số hiệu cổng của phía Server
- Mỗi gói tin gửi đi cần có địa chỉ IP, số hiệu cổng của nơi gửi đến, nơi gửi đi
- Gửi gói tin cho server
- Pseudo code cho client:

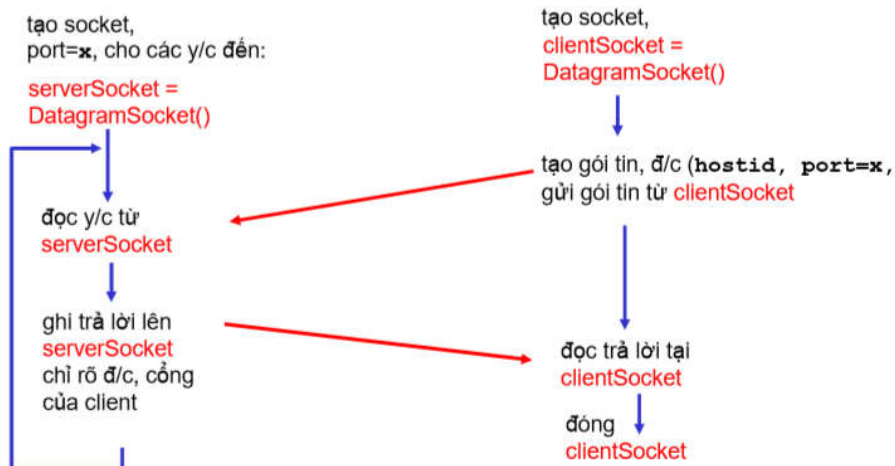
- Tạo socket
- Gửi gói tin đến địa chỉ của server
- Nhận gói tin đã xử lý từ server
- Đóng socket

- **Về phía server:**

- Nhận gói tin, trích xuất địa chỉ IP, số hiệu cổng của client, xử lý và respond gói tin lại cho client.
- Pseudo code cho server:
 - Tạo socket
 - Ràng buộc socket với 1 số hiệu cổng mà client sẽ truyền đến
 - Vòng lặp:
 - + Nhận gói tin từ client
 - + Gửi gói tin đã xử lý cho client
 - Đóng socket

Server (máy *hostid*)

Client



Hình 3: Mô hình client-server sử dụng UDP Socket

6.1.3 So sánh UDP và TCP

UDP cho tốc độ nhanh, không tốn chi phí thiết lập liên kết nên phù hợp cho thông báo dữ liệu thời tiết, phát video trực tuyến, dịch vụ báo giá cổ phiếu (không

dùng cho giao dịch) hoặc cho lưu lương đa luồng,...Trong khi đó, TCP cung cấp dịch vụ truyền dòng byte tin cậy thông qua việc thiết lập liên kết 2 chiều và có kiểm tra lỗi, mất dữ liệu nên phù hợp với việc truyền file, gửi email, trình duyệt web,...

Trước khi lập trình socket ta cần xác định giao thức nào phù hợp, UDP hay TCP và thuộc phía client hay server, phía client thì giả sử phía server đã hoạt động ta chỉ cần kết nối đến, còn phía server thì cần khởi động và chờ kết nối.

6.2 BÀI TẬP THỰC HÀNH

6.2.1 Bài thực hành 01

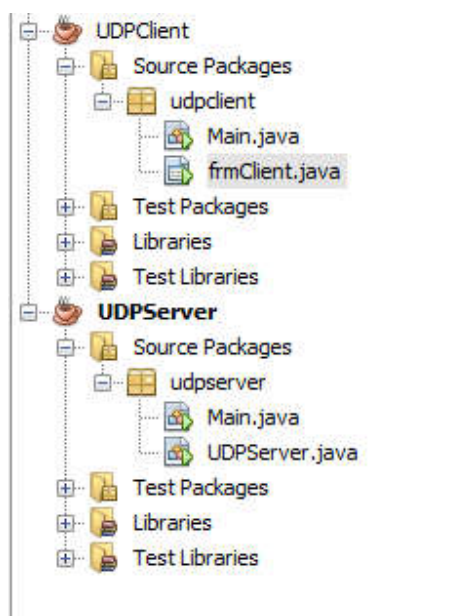
Viết chương trình giao tiếp giữa client và server sử dụng giao thức UDP, thực hiện các chức năng sau:

- Client truyền 1 chuỗi lên server, server nhận chuỗi này và chuyển nó thành chữ in hoa sau đó gửi trả kết quả cho client.
- Client nhận kết quả rồi sau đó xuất ra màn hình kết quả vừa nhận.

Yêu cầu: Sử dụng Multithread để server có thể giao tiếp được với nhiều client cùng lúc.

Hướng dẫn:

Bước 1: Tạo 2 project mới là UDPServer và UDPClient



Bước 2: Tạo Form frmClient có giao diện như sau:

Xử lý sự kiện cho button Truyền chuỗi.

```
private void btnTruyenChuoiActionPerformed(java.awt.event.ActionEvent evt) {
    byte [] sendData;
    DatagramSocket socket;
    try {
        socket = new DatagramSocket();
        String domain=this.txtDomain.getText();
        InetAddress ipServer = InetAddress.getByName(domain);//lưu địa chỉ máy server
        int port = 1234;// Sử dụng Port 1234 để giao tiếp với server
        String stSend = this.txtChuoi.getText();//Lấy dữ liệu cần truyền đi
        sendData = stSend.getBytes();//Chuyển dữ liệu thành dạng byte rồi truyền đi
        //DatagramPacket dùng để lưu dữ liệu
        DatagramPacket sendPacket= new DatagramPacket(sendData,sendData.length,ipServer,port);
        socket.send(sendPacket);
        //Nhận chuỗi kết quả từ server
        byte[] buffer=new byte[65507];//độ lớn tối đa của gói tin 65535-(7 byte header của UDP)
        DatagramPacket receivePacket=new DatagramPacket (buffer,buffer.length);
        socket.receive(receivePacket);//Nhận chuỗi kết quả
        txtKetQua.setText(new String(receivePacket.getData()).trim());//Lấy dữ liệu hiển lên màn hình
        socket.close();//đóng socket
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this,ex.toString());
    }
}
```

Xử lý sự kiện cho button Thoát

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

Bước 3: Tạo lớp UDPServer như sau:

```

package udpserver;
import java.io.*;
import java.net.*;
public class UDPServer {
    static final int PORT = 1234;//Khai báo Port sử dụng
    private DatagramSocket socket = null;//Khai báo DatagramSocket để lưu kết nối
    public UDPServer() {
        try{
            socket=new DatagramSocket(PORT);
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
    public void action(){
        InetAddress host=null;
        int port;
        String chuoi="";//Khai báo biến để lưu chuỗi dữ liệu
        try{
            System.out.println("Server is listening");
            while(true){//vòng lặp chờ
                DatagramPacket packet=receive();//Nhận dữ liệu từ client truyền qua
                host=packet.getAddress();//Lấy thông tin địa chỉ của máy client
                port=packet.getPort();//Lấy thông tin port của máy client
                chuoi=new String(packet.getData()).trim();//Lấy dữ liệu của máy client
                chuoi=chuoi.toUpperCase();//Chuyển thành chữ in hoa
                if(!chuoi.equals(""))
                    send(chuoi,host,port);
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            socket.close();
        }
    }
}

```

```

    private void send(String chuoi,InetAddress host,int port)throws IOException{
        byte[] buffer=chuoi.getBytes();//chuyển chuỗi truyền thành byte
        //Sau đó đưa chuỗi truyền vào gói tin gửi đi
        DatagramPacket packet=new DatagramPacket(buffer,buffer.length,host,port);
        socket.send(packet);
    }
    private DatagramPacket receive()throws IOException{
        byte[] buffer=new byte[65507];//Khai báo mảng byte nhận
        DatagramPacket packet=new DatagramPacket (buffer,buffer.length);
        socket.receive(packet);//Nhận dữ liệu
        return packet;
    }
    public static void main(String []args){
        new UDPServer().action();
    }
}

```

Bước 4: Chạy thử ứng dụng

Chạy file UDPServer.java

Chạy file frmClient.java

6.2.2 Bài thực hành 02

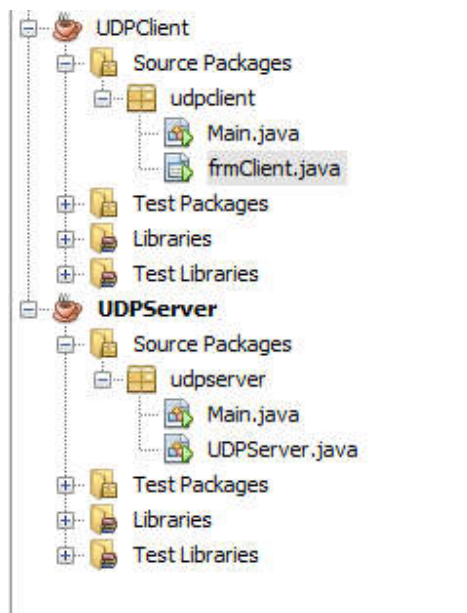
Viết chương trình giao tiếp giữa client và server sử dụng giao thức UDP, thực hiện các chức năng sau:

- Client truyền hai số nguyên và phép toán (cộng, trừ, nhân, chia) lên server.
- Server sau khi nhận được thì thực hiện phép toán giữa hai số nguyên và trả kết quả về cho client, client nhận lại kết quả và xuất ra màn hình.

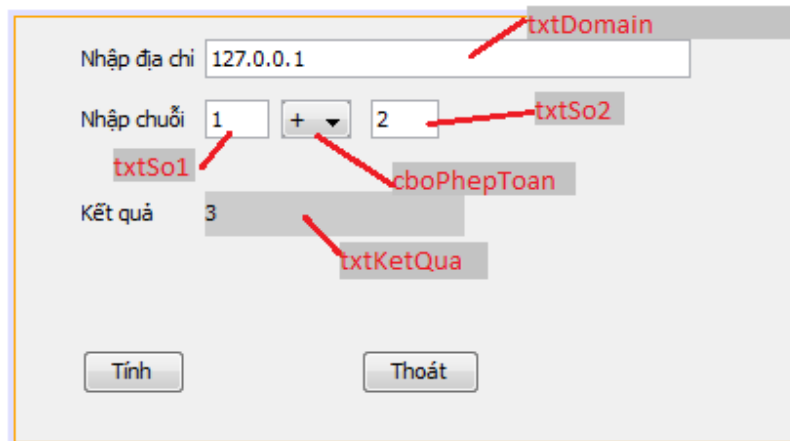
Yêu cầu: Sử dụng Multithread để server có thể giao tiếp được với nhiều client cùng lúc.

Hướng dẫn:

Bước 1: Tạo 2 project mới là UDPServer và UDPClient



Bước 2: Tạo Form frmClient có giao diện như sau:



Xử lý sự kiện cho button Tính.

```
private void btnTinhActionPerformed(java.awt.event.ActionEvent evt) {
    byte [] sendData;
    DatagramSocket socket;
    int so1= Integer.parseInt(txtSo1.getText()); //Lấy số hạng thứ nhất
    int so2= Integer.parseInt(txtSo2.getText()); //Lấy số hạng thứ hai
    String pheptoan= cboPhepToan.getSelectedItem().toString(); //Lấy phép toán
    try {
        socket = new DatagramSocket();
        String domain= this.txtDomain.getText();
        InetAddress ipServer = InetAddress.getByName(domain); //lưu địa chỉ máy server
        int port = 1234; // Sử dụng Port 1234 để giao tiếp với server
        String stSend = so1+"@"+pheptoan+"@"+so2; //Lấy dữ liệu cần truyền đi
        sendData = stSend.getBytes(); //Chuyển dữ liệu thành dạng byte rồi truyền đi
        //DatagramPacket dùng để lưu dữ liệu
        DatagramPacket sendPacket= new DatagramPacket(sendData, sendData.length, ipServer, port);
        socket.send(sendPacket);
        //Nhận chuỗi kết quả từ server
        byte[] buffer= new byte[65507]; //độ lớn tối đa của gói tin 65535-(7 byte header của UDP)
        DatagramPacket receivePacket= new DatagramPacket (buffer, buffer.length);
        socket.receive(receivePacket); //Nhận chuỗi kết quả
        txtKetQua.setText(new String(receivePacket.getData()).trim()); //Lấy dữ liệu hiển lên màn hình
        socket.close(); //đóng socket
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, ex.toString());
    }
}
```

Xử lý sự kiện cho button Thoát

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

Bước 3: Tạo lớp UDPServer như sau:

```

package udpserver;
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class UDPServer {
    static final int PORT = 1234; // Khai báo Port sử dụng
    private DatagramSocket socket = null; // Khai báo DatagramSocket để lưu kết nối
    public UDPServer() {
        try {
            socket = new DatagramSocket(PORT);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void action() {
        InetAddress host = null;
        int port;
        String chuoai = ""; // Khai báo biến để lưu chuỗi dữ liệu
        try {
            System.out.println("Server is listening");
            while (true) { // vòng lặp chờ
                DatagramPacket packet = receive(); // Nhận dữ liệu từ client truyền qua
                host = packet.getAddress(); // Lấy thông tin địa chỉ của máy client
                port = packet.getPort(); // Lấy thông tin port của máy client
                chuoai = new String(packet.getData()).trim(); // Lấy dữ liệu của máy client
                if (!chuoai.equals("")) {
                    Scanner sc = new Scanner(chuoai);
                    sc.useDelimiter("@"); // Cắt chuỗi theo ký tự @
                    int so1 = sc.nextInt(); // Lấy so1 là phần trước chữ @ đầu tiên
                    String pheptoan = sc.next(); // Phép toán là phần trước chữ @ thứ hai
                    int so2 = sc.nextInt(); // so2 là phần trước chữ @ thứ 3
                    if (pheptoan.equals("+")) // Nếu phép toán là phép cộng
                        chuoai = (so1 + so2) + "";
                    else if (pheptoan.equals("-")) // Nếu phép toán là phép trừ

```



```

        chuoai=(so1-so2)+"";
        else if(pheptoa.equals("+"))//Nếu phép toán là phép nhân
            chuoai=(so1*so2)+"";
        else if(pheptoa.equals("/"))//Nếu phép toán là phép chia
            chuoai=((float)so1/so2)+"";
        send(chuoai,host,port);//Truyền chuỗi trả về cho client
    }
}

}catch(Exception e){
    e.printStackTrace();
}finally{
    socket.close();
}

}

private void send(String chuoai,InetAddress host,int port)throws IOException{
    byte[] buffer=chuoai.getBytes();//chuyển chuỗi truyền thành byte
    //Sau đó đưa chuỗi truyền vào gói tin gửi đi
    DatagramPacket packet=new DatagramPacket(buffer,buffer.length,host,port);
    socket.send(packet);
}

private DatagramPacket receive()throws IOException{
    byte[] buffer=new byte[65507];//Khai báo mảng byte nhận
    DatagramPacket packet=new DatagramPacket (buffer,buffer.length);
    socket.receive(packet);//Nhận dữ liệu
    return packet;
}

public static void main(String []args){
    new UDPServer().action();
}

}

```

Bước 4: Chạy thử ứng dụng

Chạy file UDPServer.java

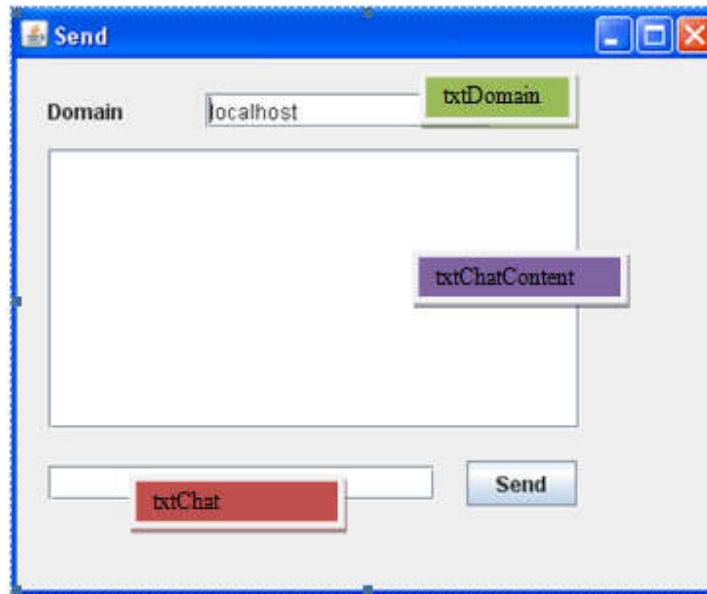
Chạy file frmClient.java

6.2.3 Bài thực hành 03

Viết chương trình cho phép hai máy chat với nhau.

Hướng dẫn:

Bước 1: Thiết kế giao diện: lớp Chat.java



Bước 2: Viết code cho button Send của lớp Chat.java

```
byte []sendData;
boolean ktFinish = false;
DatagramSocket socket;
String strContent="";
try {
    socket = new DatagramSocket();
    String domain=this.txtDomain.getText();
    InetAddress ipServer = InetAddress.getByName(domain);
    int port = Chat.PORT;
    String stSend = this.txtChat.getText();
    sendData = stSend.getBytes();
    DatagramPacket sendPacket;
    sendPaket= new DatagramPacket(sendData, sendData.length,ipServer,port);
    strContent+="\nGui : " + stSend;
    socket.send(sendPacket); //Begin chat
    socket.close();
} catch (Exception ex) {
    JOptionPane.showMessageDialog(this,ex);
}
```

Bước 4: Viết code cho lớp Main.java để tạo ra đối tượng giao diện chat và code xử lý lắng nghe gói dữ liệu chat.

```

Chat app=new Chat();
app.setVisible(true);
//doan nhan du lieu
DatagramSocket socket;
String strContent="";
try {
    byte []buffer = new byte[1024];
    socket = new DatagramSocket(PORT);
    boolean ktFinish=false;
    DatagramPacket receivePacket;
    String stReceive;
    while(ktFinish!=true) {
        receivePacket = new DatagramPacket(buffer,buffer.length);
        socket.receive(receivePacket);
        stReceive=new String(receivePacket.getData(),0,receivePacket.getLength());
        strContent=app.getContentChat();
        strContent+="Nhan : " + stReceive;
        app.setContentChat(strContent);
        if (stReceive.equals("end.")||stReceive.equals("end.")) {
            ktFinish = true; }
    }
}catch(Exception ex){
    JOptionPane.showMessageDialog(null,ex);
}

```

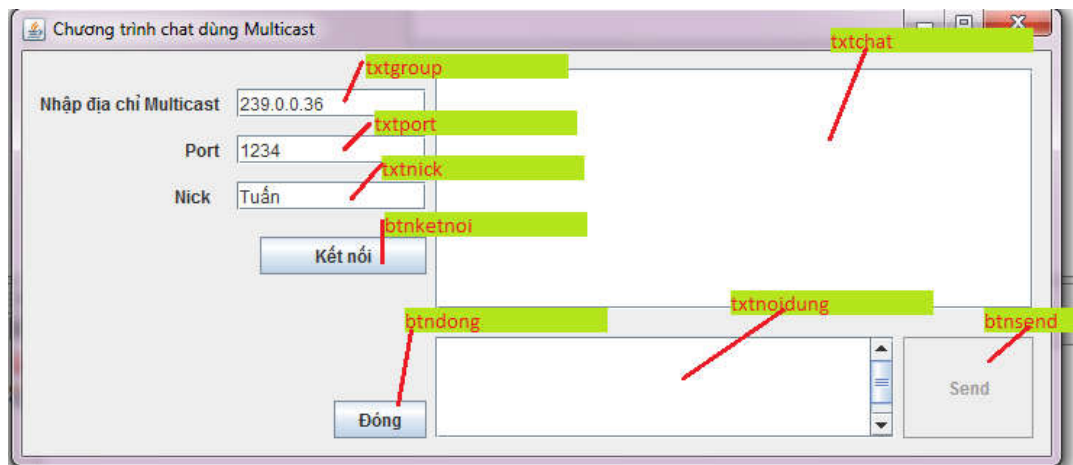
6.2.4 Bài thực hành 04

Viết chương trình cho phép hai máy chat với nhau dùng Multicast.

Hướng dẫn:

B1: Tạo JFrame frmChat

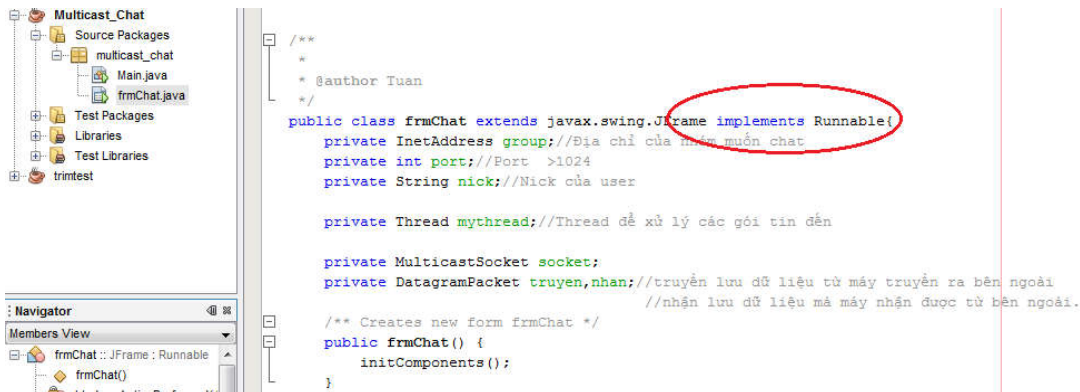
B2: Tạo giao diện như hình sau:



Khai báo các thư viện được sử dụng

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

B3: Khai báo các thuộc tính của Form



B4: Các hàm trong chương trình

```
/** Creates new form frmChat */
public frmChat() { ... }

/**...*/
@SuppressWarnings("unchecked")
Generated Code

private void btnketnoiActionPerformed(java.awt.event.ActionEvent evt) { ... }
public void run() { ... }
private void btndongActionPerformed(java.awt.event.ActionEvent evt) { ... }
private void btnsendActionPerformed(java.awt.event.ActionEvent evt) { ... }
private void formWindowClosing(java.awt.event.WindowEvent evt) { ... }

public static void main(String args[]) { ... }
```

Sự kiện cho nút kết nối

Hàm run này sẽ được gọi khi thread start();

Sự kiện cho nút đóng

Sự kiện cho nút send

Sự kiện cho việc đóng form

B5: Sự kiện cho nút kết nối

```
private void btnketnoiActionPerformed(java.awt.event.ActionEvent evt) {
    if(btnketnoi.getText().equals("Kết nối")){
        //Khi nhấn nút kết nối
        btnketnoi.setText("Ngắt kết nối");
        //chuyển nút kết nối thành nút ngắt kết nối
        txtchat.setEnabled(true);
        txtnoidung.setEnabled(true);
        btnsend.setEnabled(true);
        txtgroup.setEnabled(false);
        txtport.setEnabled(false);
    }
}
```

```
txtnick.setEnabled(false);
try{
    group= InetAddress.getByName(txtgroup.getText());
    if(group.isMulticastAddress()){
        //Kiểm tra xem địa chỉ nhóm có phải địa chỉ multicast hay không
        nick=txtnick.getText();
        port=Integer.parseInt(txtport.getText());
        if(mythread==null){
            //Tạo ra và thiết lập ban đầu cho các đối tượng mạng
            socket=new MulticastSocket(port);
            socket.setTimeToLive(1);
            //Thiết lập đường đi cho gói tin
            socket.joinGroup(group);
            //Đăng ký với router là chương trình máy mình đăng ký vào nhóm group
            truyen=new DatagramPacket(new byte[1],1,group,port);
            nhan=new DatagramPacket(new byte[65507],65507);
            //Tạo ra thread xử lý dữ liệu truyền đến
            mythread=new Thread(this);
            mythread.start();
            //Bắt đầu nhận dữ liệu - Lúc này hàm run sẽ được gọi để thực thi
        }
    }else
        JOptionPane.showMessageDialog(null, "Địa chỉ nhập sai rồi!!");
}catch(Exception e){
    JOptionPane.showMessageDialog(null,e);
}
}
}else{
    //Sự kiện nhấn nút ngắt kết nối
    txtchat.setEnabled(false);
    txtnoidung.setEnabled(false);
    btnsend.setEnabled(false);
    txtgroup.setEnabled(true);
}
```

```

txtport.setEnabled(true);
txtnick.setEnabled(true);
btnketnoi.setText("Kết nối");
    //chuyển nút ngắt kết nối thành nút kết nối
if(mythread!=null){
mythread.interrupt();
    //dừng việc nhận dữ liệu
mythread=null;
try{
    socket.leaveGroup(group);//Ra khỏi group
}catch(IOException e){}
socket.close();
}
}
}

```

B6:Hàm run: là hàm sẽ được gọi khi thread bắt đầu

```

private void btnketnoiActionPerformed(java.awt.event.ActionEvent evt) {...}
public void run() {
try{
    while (!Thread.interrupted()) { //Kiểm tra xem thread có bị ngắt chưa
        nhan.setLength(nhan.getData().length); //thiết lập số byte của buffer
        socket.receive(nhan); //nhận dữ liệu
        String message = new String(nhan.getData(), 0, nhan.getLength(), "UTF8");
        txtchat.append(message + "\n"); //hiển thị dữ liệu nhận được lên màn hình
    }
} catch (IOException e) { //Các thao tác thu dọn bộ nhớ khi có lỗi xảy ra
    if(mythread!=null){
        txtchat.append(e+"\n");
        txtnoidung.setVisible(false);
        this.validate();
        if(mythread!=Thread.currentThread())
            mythread.interrupt();
        mythread=null;
        try{
            socket.leaveGroup(group);
        }catch(IOException ignored){}
        socket.close();
    }
}
}
}

```

B7: Sự kiện cho nút đóng

```
private void btndongActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

B8: Sự kiện cho nút Send

```
private void btnsendActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        byte[] utf=(nick+": "+txtnoidung.getText()).getBytes("UTF8");//Chuyển dữ liệu thành chuỗi byte
        truyen.setData(utf);// gán dữ liệu cho datagrampackage
        truyen.setLength(utf.length);//thiết lập số lượng byte cho buffer
        socket.send(truyen);//bắt đầu truyền dữ liệu đi
        txtnoidung.setText("");//cho nội dung của txtnoidung là rỗng
    }catch(IOException e){//Các xử lý dọn dẹp bộ nhớ khi có lỗi
        if(mythread!=null){
            txtchat.append(e+"\n");
            txtnoidung.setVisible(false);
            this.validate();
            if(mythread!=Thread.currentThread()){
                mythread.interrupt();
            }
            mythread=null;
            try{
                socket.leaveGroup(group);
            }catch(IOException ignored){}
            socket.close();
        }
    }
}
```

B9: Sự kiện cho việc đang đóng form

```
private void formWindowClosing(java.awt.event.WindowEvent evt) {
    if(mythread!=null){// các hàm dọn dẹp bộ nhớ
        mythread.interrupt();
        mythread=null;
        try{
            socket.leaveGroup(group);
        }catch(IOException e){}
        socket.close();
    }
}
```

B10: Kiểm tra bằng cách chạy thử chương trình trên nhiều máy mạng LAN chỉ cần thay nick khác nhau cho từng máy. Những máy nào thiết lập cùng group(cùng địa chỉ Multicast) thì có thể chat được với nhau.

6.3 BÀI TẬP LÀM THÊM

6.3.1 Bài tập 1

Viết chương trình minh hoạt giao thức FTP cho phép hai máy gửi tập tin cho nhau sử dụng UDP Socket.(Sinh viên áp dụng kiến thức đã học để thực hiện)

6.3.2 Bài tập 2

Viết chương trình cho phép gia nhập vào một địa chỉ multicast. Thực hiện chức năng gửi một thông điệp vào địa chỉ này và nhận các thông điệp được gửi tới địa chỉ này. (Sinh viên áp dụng kiến thức đã học để thực hiện)

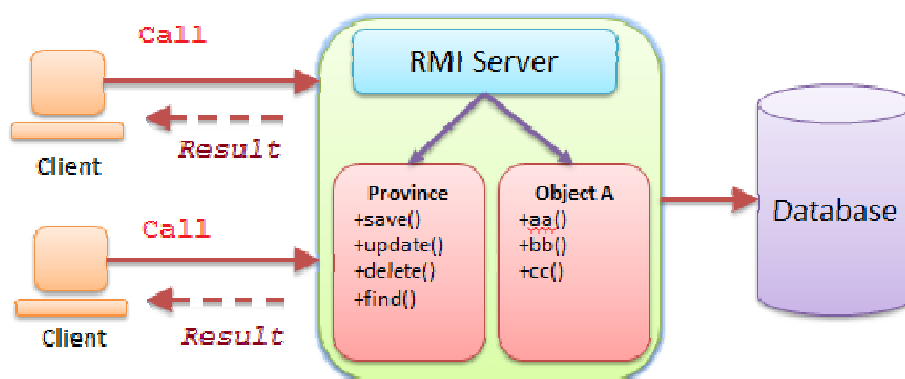
BÀI 7. LẬP TRÌNH RMI

Sau khi thực hành xong bài này, sinh viên sẽ nắm được các kiến thức cơ bản về lập trình đối tượng phân tán (RMI - Remote Method Invocation) trong Java và áp dụng vào các bài tập thực tế.

7.1 JAVA RMI

7.1.1 Khái niệm

RMI - Remote Method Invocation là một kỹ thuật cài đặt các đối tượng phân tán trong Java. RMI là một phần của bộ J2SDK và là hàm thư viện hỗ trợ các lời gọi phương thức từ xa và trả về giá trị cho các ứng dụng tính toán phân tán. Chúng ta giả sử rằng ngôn ngữ Java được sử dụng ở cả hai phía gọi và phía bên phương thức được gọi. Để giải quyết một số vấn đề trong việc truyền thông giữa Client/Server. RMI không gọi trực tiếp mà thông qua lớp trung gian. Lớp này tồn tại ở cả hai phía Client và Server. Lớp ở máy Client gọi là Stub, lớp ở máy Server gọi là Skel (Skeleton).



Hình 1: RMI trong Java

7.1.2 Đặc tính của RMI

RMI là mô hình đối tượng phân tán của Java, RMI giúp cho việc giao tiếp giữa các đối tượng phân tán trong môi trường internet trở nên dễ dàng hơn. RMI là API bậc cao được xây dựng dựa trên lập trình Socket. RMI không những cho phép chúng ta truyền dữ liệu giữa các đối tượng trên các hệ thống máy tính khác nhau, mà còn triệu gọi các phương thức trong các đối tượng ở xa (Remote Object). Việc truyền dữ liệu giữa các máy khác nhau được xử lý một cách trong suốt bởi máy ảo Java (Java Virtual Machine). Tương tự như mô hình Client/Server, RMI vẫn lấy/duy trì khái niệm của Client và Server, tuy nhiên cách tiếp cận (approach) của RMI linh hoạt hơn, mềm dẻo hơn so với một hình Client/Server. Một điều thuận lợi quan trọng nhất của RMI là nó cung cấp cơ chế callbacks, nó cho phép Server triệu gọi các phương thức ở Client.

7.1.3 Kiến trúc RMI



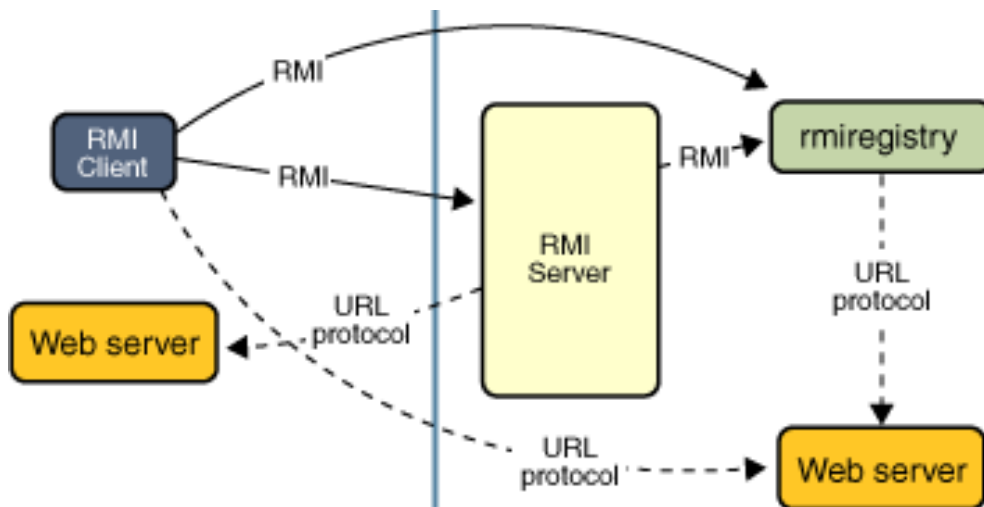
Hình 2: Kiến trúc RMI

Remote Interface: Nên extend từ `java.rmi.remote`. Nó khai báo tất cả các phương thức mà Client có thể triệu gọi. Tất cả các method trong interface này nên throw `RemoteException`

Remote Implementation: Được thực thi từ Remote interface và mở rộng từ UnicastRemoteObject. Triển khai các method được khai báo trong Interface tại đây.

Nó là một Remote Object thực sự. Phát sinh hai lớp trung gian Stub và Skel.

Server Class bao gồm: RMI Registry: Bộ đăng kí này sẽ đăng kí một Remote object với Naming Registry. Giúp các Remote object được chấp nhận khi gọi các method từ xa. Các class thực thi trên server. Client Class: Truy vấn trên tên Remote object trên RMI registry, thông qua stub để gọi các phương thức trên server.



Hình 3: Hoạt động của RMI Server và RMI Client

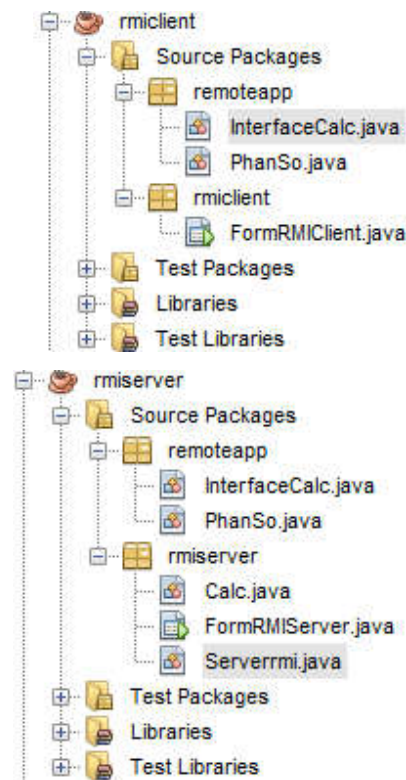
7.2 BÀI TẬP THỰC HÀNH

7.2.1 Bài thực hành 01

Viết chương trình server tạo ra đối tượng thư viện cho phép cộng hai phân số. Một chương trình client tạo ra 2 đối tượng phân số và gọi từ xa đối tượng thư viện ở server để cộng hai phân số này.

Hướng dẫn:

Bước 1: Tạo 2 project rmiclient và rmiserver



Bước 2: Tạo interface InterfaceCalc.java

```
package remoteapp;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface InterfaceCalc extends Remote {
    public int sum(int a, int b) throws RemoteException;
    public PhanSo sum(PhanSo a, PhanSo b) throws RemoteException;
}
```

Bước 3: Tạo lớp PhanSo.java

```
package remoteapp;
import java.io.Serializable;
public class PhanSo implements Serializable{
    int tu;
    int mau;
    public PhanSo(){
        tu=0; mau=1;
    }
    public PhanSo(int t,int m){
        tu=t;
        mau=m;
    }
    public PhanSo sum(PhanSo b){
        PhanSo c=new PhanSo();
        c.setTu(tu*b.getMau()+ b.getTu()*mau);
        c.setMau(mau*b.getMau());
        return c;
    }
    public void setTu(int t){
        tu=t;
    }
    public void setMau(int m){
        mau=m;
    }
    public int getTu(){
        return tu;
    }
    public int getMau(){
        return mau;
    }
}
```

Bước 4: Hiện thực lớp Calc

```

package rmiserver;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import remoteapp.InterfaceCalc;
import remoteapp.PhanSo;

class Calc extends UnicastRemoteObject implements InterfaceCalc {
    public Calc() throws RemoteException
    {
    }
    public int sum(int a,int b) throws RemoteException
    {
        int t=a+b;
        return t;
    }
    public PhanSo sum(PhanSo a, PhanSo b) throws RemoteException
    {
        return a.sum(b);
    }
}

```

Bước 5: Tạo lớp Serverrmi.java

```

package rmiserver;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Serverrmi {
    public static void main(String args[]){
        try {
            Calc c = new Calc();
            Registry r = LocateRegistry.createRegistry(3456);
            r.bind("rmiCalc",c);
        }
        catch (Exception ex)
        {
            System.out.println(ex);
        }
    }
}

```

Bước 6: Tạo FormRMIClient.java

Bước 7: Xử lý sự kiện click trên button + số

```
private void btnCongActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String diaChi=this.txtDiaChi.getText();
        Registry r = LocateRegistry.getRegistry(diaChi,3456);
        InterfaceCalc ic =(InterfaceCalc)r.lookup("rmiCalc");
        int so1=Integer.valueOf(this.txtSo1.getText());
        int so2=Integer.valueOf(this.txtSo2.getText());
        int c=ic.sum(so1, so2);
        this.txtKetQua.setText(String.valueOf(c));
    }
    catch (Exception ex) {
        System.out.println(ex);
    }
}
```

Bước 8: Xử lý sự kiện click trên button + phân số


```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String diaChi=this.txtDiaChi.getText();
        Registry r = LocateRegistry.getRegistry(diaChi,3456);
        InterfaceCalc ic =(InterfaceCalc)r.lookup("rmiCalc");
        PhanSo a=new PhanSo(Integer.valueOf(this.txtTuA.getText()),Integer.valueOf(this.txtMauA.getText()));
        PhanSo b=new PhanSo(Integer.valueOf(this.txtTuB.getText()),Integer.valueOf(this.txtMauB.getText()));
        PhanSo c=ic.sum(a, b);
        this.txtKetQua.setText(String.valueOf(c.getTu())+ "/" + c.getMau());
    }
    catch (Exception ex) {
        System.out.println(ex);
    }
}
```

Bước 9: Xử lý sự kiện click trên button Thoát

```
private void bntThoatActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(1);
}
```

7.2.2 Bài thực hành 02

Viết chương trình Server quản lý thông tin tài khoản người sử dụng ATM. Chương trình có đối tượng quản lý người sử dụng cho phép triệu gọi từ xa bởi các máy ATM.

- Thông tin người sử dụng gồm có: Tên, CMND, mật khẩu, số tiền, lãi xuất, thông tin các giao dịch trên tài khoản.
- Thông tin giao dịch gồm có: ngày giao dịch, số tiền, kiểu giao dịch (rút tiền hoặc gửi tiền).
- Chương trình quản lý đối tượng người sử dụng có các chức năng sau:
 - Thêm một người sử dụng mới vào danh sách để quản lý.
 - Tìm người sử dụng có tên và mật khẩu
 - Cho phép nhập/rút tiền trên tài khoản có tên và mật khẩu (hoặc CMND).

TÀI LIỆU THAM KHẢO

- [1]. Java Network Programming, Fourth Edition (2014) by Elliotte Rusty Harold,
Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol,
CA 95472.
- [2]. Internetworking with TCP/IP, Vol 1 (5th Edition) by Douglas E. Comer
- [3]. TCP/IP Illustrated, Vol. 1: The Protocols (Addison-Wesley Professional
Computing Series) by W. Richard Stevens
- [4]. Computer Networks Subsequent Edition by Andrew S. Tanenbaum
- [5]. Website: <https://codelearn.io/>
- [6]. Website: <https://viblo.asia/>