TÀI LIỆU THỰC TẬP LẬP TRÌNH MẠNG: LAB 11

DANH MỤC THUẬT NGỮ TIẾNG ANH

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

DANH MỤC CHỮ VIẾT TẮT

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

LỜI NÓI ĐẦU

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

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

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

Xin trân trọng cảm ơn!

Nhóm tác giả

LAB 11. LẬP TRÌNH MULTICAST, URL, MAIL [4]

A. MỤC TIÊU

Xây dựng ứng dụng Client/Server sử dụng lớp MulticastSocket để kết nối và trao đổi thông tin.

- Thực hiện được các thao tác lập trình java giao tiếp với URL, URLConnection sử dụng lớp URL, URLConnection nằm trong gói java.net.
- Thực hiện được kỹ thuật gửi email qua gmail sử dụng mail API

B. NỘI DUNG

- Lập trình các ứng dụng multicast.
- Lập trình với URL, URLConnection
- Gửi thư với mail API.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

- Viết được các ứng dụng muilticast: Chat, gửi ảnh đến một nhóm người dùng.
- Xử lý liên quan đến biểu diễn tài nguyên URL trên web.
- Viết được ứng dụng gửi thư hoàn chỉnh.

D. YÊU CẦU PHẦN CỨNG - PHẦN MỀM

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

E. HƯỚNG DẪN

Bài 1.

Viết chương trình chat multicast, giữa Client và Server sử dụng lớp MulticastSocket.

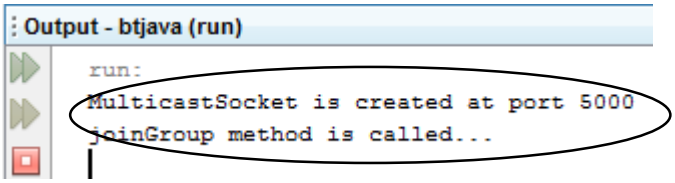
Hướng dẫn:

Bước 1: Tạo class MulticastChatServer

```
import java.net.*;
public class MuticastChatServer {
   public static void main(String args[]) throws Exception {
        int portnumber = 5000;
        if (args.length >= 1) {
            portnumber = Integer.parseInt(args[0]);
        }
        // Create a MulticastSocket
 MulticastSocket ServerMulticastSocket = new MulticastSocket(portnumber);
    System.out.println("MulticastSocket is created at: " + portnumber);
        // Determine the IP address of a host, given the host name
        InetAddress group = InetAddress.getByName("225.4.5.6");
        ServerMulticastSocket.joinGroup(group);
        System.out.println("joinGroup method is called...");
        boolean infinite = true;
```

```
        while (infinite) {
            byte buf[] = new byte[1024];
            DatagramPacket data = new DatagramPacket(buf, buf.length);
            ServerMulticastSocket.receive(data);
            String msg = new String(data.getData()).trim();
            System.out.println("Message received from Client = " + msg);
        }
        ServerMulticastSocket.close();
    }
}
```

Kết quả:

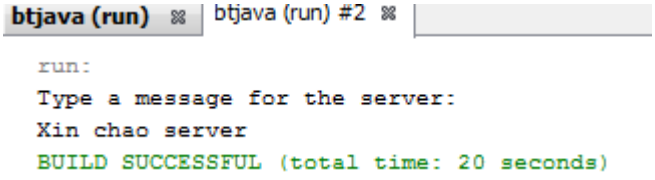


Bước 2. Tạo class MulticastChatClient

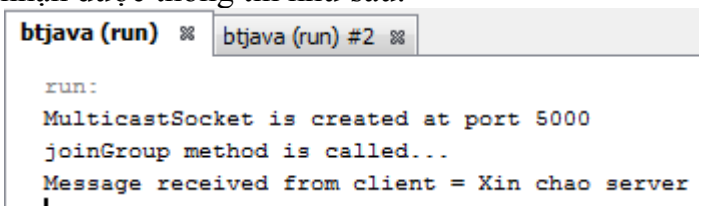
```
import java.net.*;
import java.io.*;
public class MulticastChatClient {
    public static void main(String args[]) throws Exception {
        int portnumber = 5000;
        if (args.length >= 1) {
            portnumber = Integer.parseInt(args[0]);
        }
        // Create a MulticastSocket
     MulticastSocket chatMulticastSocket = new
MulticastSocket(portnumber);
      InetAddress group = InetAddress.getByName("225.4.5.6");
        // Joins a multicast group
        chatMulticastSocket.joinGroup(group);
        String msg = "";
        System.out.println("Type a message for the Server:");
        BufferedReader br =
        new BufferedReader(new InputStreamReader(System.in));
        msg = br.readLine();
        // Send the message to Multicast address
        DatagramPacket data = new DatagramPacket(msg.getBytes(), 0,
        msg.length(), group, portnumber);
        chatMulticastSocket.send(data);
        chatMulticastSocket.close();
```

```
    }
}
```

Chạy chương trình phía Client:



Khi đó phía Server nhận được thông tin như sau:



Bài 2:

Viết chương trình broad cast thời gian và ngày tháng tới nhiều máy khách.

Hướng dẫn:

Bước 1: Tạo class BroadcastServer

```
public class BroadcastServer{
public static final int PORT = 1200;
public static void main(String args[]) throws Exception{
                  MulticastSocket socket;
                  DatagramPacket packet;
                  InetAddress address;
                address = InetAddress.getByName();
                  socket = new MulticastSocket();
// join a Multicast group and send the group salutations
                  socket.joinGroup(address);
                  byte[] data = null;
                        for(;;){
                          Thread.sleep(1000);
                    System.out.println("Sending ");
                    String str = (new Date()).toString();
                    data = str.getBytes();
             packet = new DatagramPacket(data,str.length(),address,PORT);
                           // Sends the packet
                    socket.send(packet);
                  }  // for
               }  // main
} // class BroadcastServer
```

Bước 2: Tạo class BroadcastClient

```
             public class BroadcastClient{
```

```
 public static final int PORT = 1200;
 public static void main(String args[]) throws Exception{
                  MulticastSocket socket;
                  DatagramPacket packet;
                  InetAddress address;
      address = InetAddress.getByName(args[0]);
      socket = new MulticastSocket(BroadcastServer.PORT);
      //join a Multicast group and send the group salutations
                  socket.joinGroup(address);
                  byte[] data = null;
                  packet = new DatagramPacket(data,data.length);
                  for(;;){
                    // receive the packets
                    socket.receive(packet);
                    String str = new String(packet.getData());
             System.out.println(" Time signal recieved from"+
                      packet.getAddress() + " Time is : " +str);
                  } // for
               }  // main
} // class Broadcast
```

Bài 3: Viết chương trình multicast ảnh.

Bước 1: Tạo class MulticastImageSender

```
 public class MulticastImageSender {
  private static final int TIMEOUT = 3000;
  private static final int MAXFILELEN = 65000; // File must fit in
single datagram
  public static void main(String[] args) throws IOException,
InterruptedException {
    if (args.length < 4)  // Test for correct # of args
      throw new IllegalArgumentException(
        "Parameter(s):  <Multicast Address> <Port> <TTL> <Image File>
[<Image File>...]");
    InetAddress multicastAddress = InetAddress.getByName(args[0]);
    int destPort = Integer.parseInt(args[1]);  // Destination port of
multicast packets
    int TTL = Integer.parseInt(args[2]);
    // Create a UDP multicast socket with any available local port
    MulticastSocket socket = new MulticastSocket();
    socket.setTimeToLive(TTL);  // Set the TTL
    for (int i=3; i < args.length; i++)
```

```
    {
      RandomAccessFile file = new RandomAccessFile(args[i], "r");
      if (file.length() > MAXFILELEN)
        throw new IOException("File too big");
      byte [] fileBuffer = new byte[(int) file.length()];
      file.read(fileBuffer);
      file.close();
      // Create a datagram to send
      DatagramPacket sendPacket = new DatagramPacket(fileBuffer,
        fileBuffer.length, multicastAddress, destPort);
      socket.send(sendPacket);  // Send the echo string
      System.out.println("Sent " + args[i] + " to " +
                  sendPacket.getAddress().getHostAddress() +
                 " on port " + sendPacket.getPort());
      Thread.sleep(TIMEOUT);
    }
    socket.close();
  }
}
```

Bước 2: Tạo class MulticastImageReceiver

```
public class MulticastImageReceiver extends JFrame {
  private JLabel picture;     // Label to contain image
  public MulticastImageReceiver() {
    super("Multicast Image Receiver");  // Set the window title
    setSize(300, 300);    // Set the window size
    picture = new JLabel("No image", SwingConstants.CENTER);
    JScrollPane scrollPane = new JScrollPane(picture);
    getContentPane().add(scrollPane, "Center");
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent evt) {
        System.exit(0);
      }
    });
  }
  public JLabel getPicture() {
    return picture;
  }
  public static void main(String[] args) throws IOException {
    if (args.length != 2) // Test for correct # of args
      throw   new   IllegalArgumentException("Parameter(s):     <Multicast
```

```
Address> <Port>");
    final InetAddress multicastAddress = InetAddress.getByName(args[0]);
    if (!multicastAddress.isMulticastAddress())
      throw new IllegalArgumentException("Not a multicast address");
    int port = Integer.parseInt(args[1]);
MulticastImageReceiver multicastImageReceiver = new
MulticastImageReceiver();
    multicastImageReceiver.setVisible(true);
    new  Thread(new  MulticastImageReceiverThread(multicastImageReceiver,
multicastAddress, port,"No Image")).start();
  }
}
class MulticastImageReceiverThread implements Runnable {
  private static final int MAXFILELEN = 65000; //
  private InetAddress multicastAddress;       // Sender multicast address
  private int port;                            // Sender port
  Runnable updateImage;
  String imageText;                            // Label text
  byte[] image = new byte[MAXFILELEN];         // Bytes of image
  boolean imageValid = false;
  public MulticastImageReceiverThread(final MulticastImageReceiver frame,
      InetAddress multicastAddress, int port, String initialImageText) {
    this.multicastAddress = multicastAddress;
    this.port = port;
    this.imageText = initialImageText;
    updateImage = new Runnable() {
      public void run() {
        JLabel picture = frame.getPicture();
        picture.setText(imageText);
        if (imageValid) {
          ImageIcon newImage = new ImageIcon(image);
          picture.setIcon(newImage);
          picture.setPreferredSize(new Dimension(newImage.getIconWidth(),
                                                 newImage.getIconHeight()));
        } else
          picture.setIcon(null);
        picture.revalidate();
      }
    };
  }
```

```
public void changeImage() {
 try {
      SwingUtilities.invokeAndWait(updateImage);
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
  public void run() {
    DatagramPacket recvPacket = new DatagramPacket(image, MAXFILELEN);
    MulticastSocket socket;
     try {
      socket = new MulticastSocket(port);
      socket.joinGroup(multicastAddress);  // Join the multicast group
    } catch (IOException e) {
      imageText = "Problem with multicast socket";
      imageValid = false;
      changeImage();
      return;
    }
    for (;;) {
      try {
        socket.receive(recvPacket);  // Receive the image
      } catch (IOException e) {
        break;   // Assume exception due to file closing
      }
      imageText = "";
      imageValid = true;
      changeImage();
      recvPacket.setLength(MAXFILELEN);   // You have to reset this!!!
    }
  }
}
```

Bài 4 .

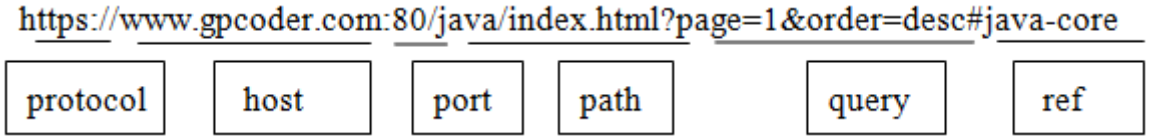
Xây dựng chương trình multicast theo yêu cầu sau. Chức năng chương trình:

- Tham gia vào group của multicast
- Gửi dữ liệu đến địa chỉ multicast
- Nhận dữ liệu từ multicast
- Hiển thị lên màn hình
- Chỉ cần các client tham gia vào group của địa chỉ multicast này, khi có dữ liệu được gửi vào đó tất cả client đều nhận được.

Gợi ý:
- Tham gia vào group của địa chỉ multicast
- Gửi dữ liệu đến địa chỉ multicast (Lúc này các client muốn nhận được thì phải tham gia vào group của multicast đó)

1. Lớp URL

Biểu diễn một tài nguyên trên Web. Một URL được phân chia thành các phần như sau:



Hình 14. Các thành phần của URL

Các phương thức để truy cập các phần khác nhau của URL:

- **public String** getPath() : trả về path của URL.
- **public int** getPort() : trả về port của URL.
- **public String** getProtocol() : trả về protocol của URL.
- **public String** getHost() : trả về host của URL.
- **public String** getRef() : trả về phần reference của URL.
- **public URLConnection** openConnection(): mở một kết nối tới URL, cho phép một Client giao tiếp với tài nguyên.

Lớp URLConnection

- Lớp trừu tượng URLConnection là lớp **super class** của tất cả các lớp biểu diễn cho kết nối tương tác (2 chiều) giữa ứng dụng và một URL.
- Tạo một URLConnection từ một URL thực hiện qua bước sau :
    1. Xây dựng một đối tượng URL.
    2. Gọi phương thức openConnection() của đối tượng URL để tìm kiếm một đối tượng URLConnection cho URL đó.
    3. Cấu hình đối tượng URL.
    4. Đọc các trường header.
    5. Nhận một luồng nhập và đọc dữ liệu.
    6. Nhận một luồng xuất và ghi dữ liệu.
    7. Đóng liên kết

Bài 5. Phân tích các phần của một địa chỉ URL.

```
import java.net.URL;
public class TestURL
{
    public static void main(String[] args) throws Exception
    {
        URL url = new
URL("https://www.packtpub.com:80/books/content/support");
```

```
        displayURL(url);
    }
    private static void displayURL(URL url)
    {
        System.out.println("URL: " + url);
        System.out.printf(" Protocol: %-32s Host: %-32s\n",
        url.getProtocol(),url.getHost());
        System.out.printf(" Port: %-32d Path: %-32s\n",
        url.getPort(),url.getPath());
        System.out.printf(" Authority: %-32s Query: %-32s\n",
        url.getAuthority(),url.getQuery());
        System.out.println(" User Info: " + url.getUserInfo());
    }
}
```

Bài 6. Lấy dữ liệu từ 1 URL

```
public static void main(String[] args)
{
    try
    {
        DownloadResource dlr = new DownloadResource();
        URL url=new
URL("http://cdn2.tstatic.net/tribunnews/foto/bank/images/Shaila-
Sabt.jpg");
        String destinationFilePath="c:/a/girl.jpg";
        long bytes=dlr.download(url, destinationFilePath);
        System.out.printf("%d bytes downloaded",bytes);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
public long download(URL url, String destinationFilePath)throws Exception
{
    long bytes=0;
    FileOutputStream fos= new FileOutputStream(destinationFilePath);
    int len=512;
    InputStream is=url.openStream();
    byte[] buffer=new byte[512];
    while(is.available()!=0)
```

```
    {
        len=is.read(buffer);
        bytes+=len;
        fos.write(buffer,0,len);
    }
    fos.close();
    return bytes;
}
```

Bài 7. Đọc thông tin, nội dung trực tiếp từ một URL.

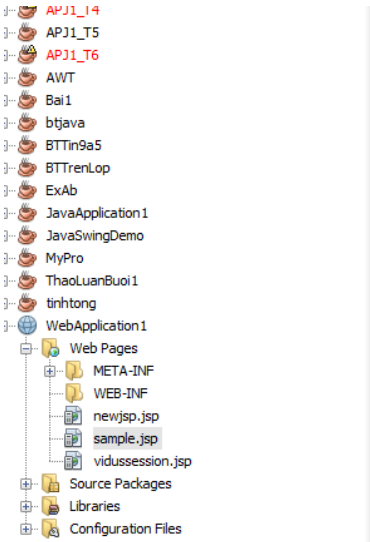
Hướng dẫn: Tạo URL, gọi phương thức openStream() để lấy một luồng từ đó có thể đọc nội dung của URL.

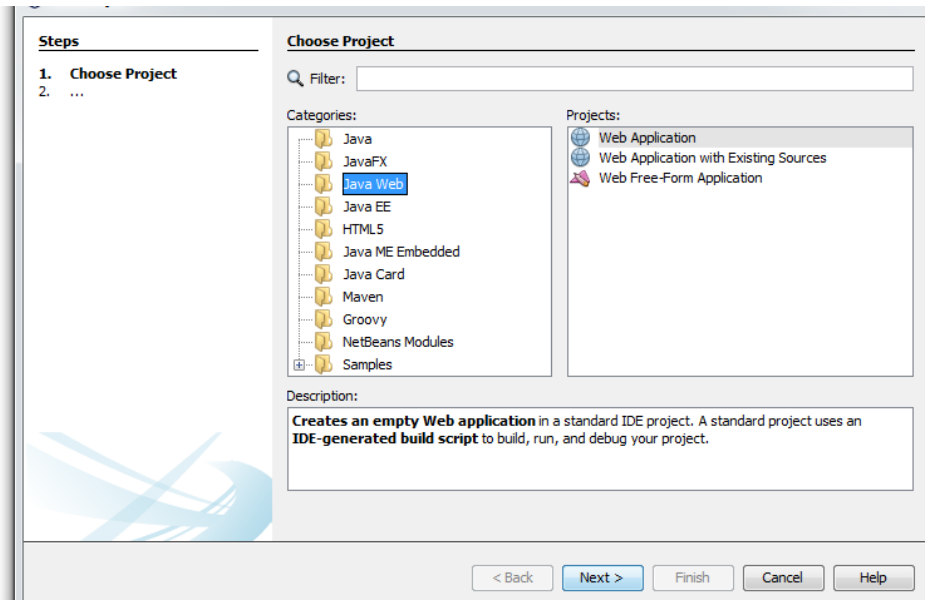
```
import java.net.*;
import java.io.*;
public class URLReader
{
    public static void main(String[] args) throws Exception
    {
        URL oracle = new URL("http://www.oracle.com/");
        BufferedReader in = new BufferedReader(
        new InputStreamReader(oracle.openStream()));
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }
}
```

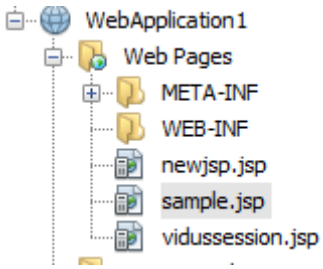
Bài 8. Viết chương trình gửi request đến một trang web xử lý sau đó nhận kết quả.

Hướng dẫn:

Bước 1: Vào NETBEAN tạo file sample.jsp có nội dung sau







Nội dung File sample.jsp

```
<html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%
            String us=request.getParameter("user");
            String psw=request.getParameter("password");
            if(us.equals(psw))
                out.println("Welcome "+us);
            else
            out.println("Something wrong with your username & password");
         %>
    </body>
</html>
```

Bước 2: Mở trình duyệt thử với address sau

localhost:8080/WebApplication1/sample.jsp?user=teo&password=teo

Bước 3: Viết code dùng URLConnection gửi dữ liệu đến web Server

```
import java.io.OutputStream;
import java.net.URL;
import java.net.URLConnection;
import java.util.Scanner;
public class SendoutputSample
{
    public static void main(String[] args) throws Exception
    {
        String spec="http://localhost:8080/sample.jsp";
        URL url=new URL(spec);
        URLConnection urlcon=url.openConnection();
        urlcon.setDoInput(true);
        urlcon.setDoOutput(true);
        urlcon.connect();
        try(OutputStream os = urlcon.getOutputStream())
        {
            os.write("user=teo&password=teo".getBytes());
            os.flush();
        }
         try(Scanner in=new Scanner(urlcon.getInputStream()))
        {
            if (in.hasNextLine())
            {
                String line=in.nextLine();
                System.out.println(line);
            }
        }
    }
}
```

Thực thi, quan sát kết quả

Bước 4: Thay password trong dòng code gạch chân trên với password bằng một giá trị gì đó khác "teo". Quan sát kết quả.

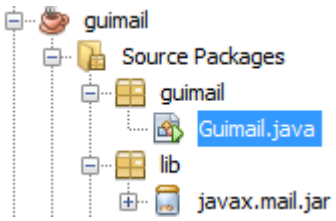
Bài 9. Viết chương trình gửi mail qua địa chỉ gmail cho trước.

Hướng dẫn:

Bước 1: Email phải kích hoạt tính năng POP:

- Để kích hoạt POP; Sau khi đăng nhập vào tài khoản gmail -> chọn Setting -> chọn Forwarding and POP/IMAP
- Chọn chức năng pop cho tất cả thư

Bước 2: **import** thư viện javax.mail.jar

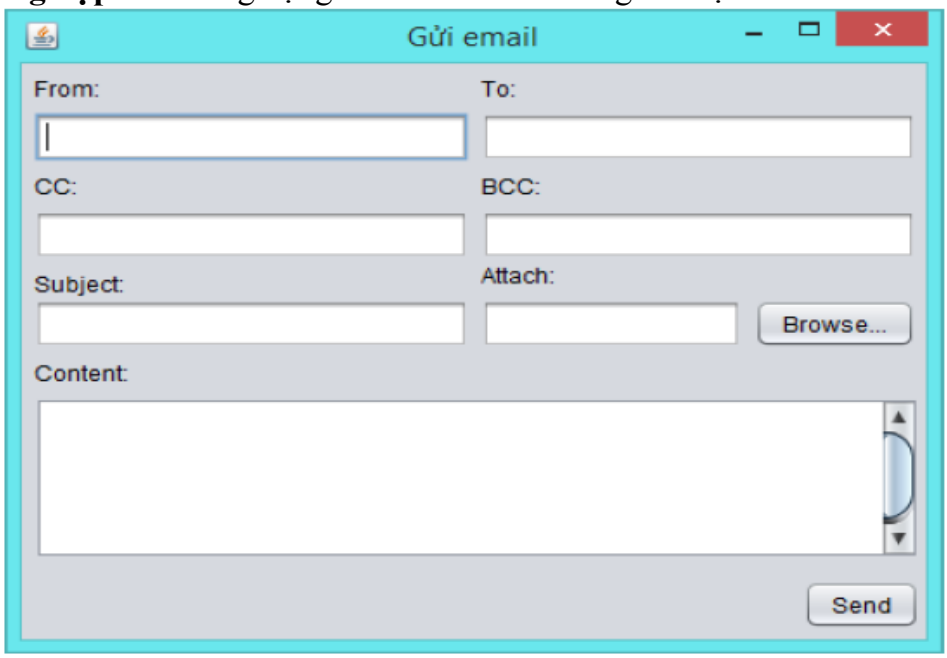


Bước 3: Tạo phương thức sendmail với thuộc tính truyền vào tên email, password của email gửi, tên email nhận, nội dung gửi, tiêu đề.

```
import java.util.Properties;
import javax.mail.*
public class Guimail
{
    public static void sendmail(String userName,String password,String
email2,String content,String title) throws AddressException,
MessagingException
    {
     //Tạo đối tượng để thiết lập các thuộc tính cho việc gửi mail
        Properties props = new Properties();
        props.put("mail.smtp.auth", true);
        props.put("mail.smtp.starttls.enable", true);
        props.put("mail.smtp.host", "smtp.gmail.com");
        // port mail thường là 587 hoặc 456
        props.put("mail.smtp.port", "587");
         // Tạo đối tượng Session (phiên làm việc với gmail)
    Session session = Session.getDefaultInstance(props, new
Authenticator()
        {
         protected PasswordAuthentication
getPuserNameasswordAuthentication()
            {
                //để gửi mail qua SMTP cần có tài khoản hợp lệ của google
                return new PasswordAuthentication(userName, password);
            }
        });
                // Tạo đối tượng chứa thông điệp cần gửi mail
        Message message=new MimeMessage(session);
                // Truyền địa chỉ muốn gửi thông điệp
    message.addRecipient(Message.RecipientType.TO, new
InternetAddress(email2));
        message.setSubject(content);
        message.setContent(title,"text/html");
```

```
        Transport.send(message,userName,password);
    }
    public static void main(String[] args) throws MessagingException
    {
       try
        {
            String username1="congquyen2503";
            String pass="quyen2345";
            String username2="luongthaohieu@gmail.com";
            String content="test xem duoc khong ;";
            String title="mail from "+username1;
            sendmail(username1,pass,username2,content,title);
            System.out.println("gui thanh cong");
        }
        catch(Exception e)
        {
            System.out.println("gui that bai :"+e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Bài 10. Tổng hợp: Viết ứng dụng mail hoàn chỉnh có giao diện như sau:



Gợi ý:

Tạo giao diện, thực hiện chức năng gửi mail.

Chú ý hai thư viện gửi mail: mail.jar và activation.jar

TÀI LIỆU THAM KHẢO

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

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

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

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

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

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

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

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

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

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

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

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

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