Lecture 03
Custom Networking
Part 1

Networking Basics
Working with URL

Reference: tutorial-2015/networking/index.html

Why should we study this lecture?

- Nowadays, distributed applications are popular. People need large applications, running based on a computer network (local area networks-LANs- or wide area network-WAN), including many sites working concurrently. Do you want to create such applications?
- How do we develop Java network applications?

Contents

- Networking Basics
- Working with URLs

1- Networking Basics

- Some definitions related to networking
- Client-Server Model

Definitions

- **Platform**: hardware + operating system.
- **Client**: an application running in a computer (such as browser) can receive data from another (server).
- **Server**: an application running in a computer (such as IIS-Windows Internet Information Service) can supply data to others (clients).
- **IP address** (internet protocol): unsigned integer helps identifying a network element(computer, router,…).
- **IPv4**: 4-byte IP address, such as 192.143.5.1
- **IPv6**: 16-byte IP address
- **Port**: unsigned 2-byte integer helps operating system differentiating a network communicating process.
- **Protocol**: Rules for packaging data of a network communication because client and server can be working in different platform. Two common basic protocols are TCP and UDP

Definitions

- **TCP**: (*Transmission Control Protocol*) is a connection-based protocol (only one connecting line only) that provides a reliable flow of data between two computers based on the acknowledge mechanism.
- **UDP**: (*User Datagram Protocol*) is a protocol that sends independent packets of data, called datagrams, from one computer to another with no guarantees about arrival (many connecting lines can be used, acknowledge mechanism is not used). Many firewalls and routers have been configured not to allow UDP packets. Ask your system administrator if UDP is permitted.
- **Serialization**: a process that converts object's state (values in fields of the object) to a byte stream.
- **De- serialization**: a process that splits data in a byte stream then set data to fields of an object.

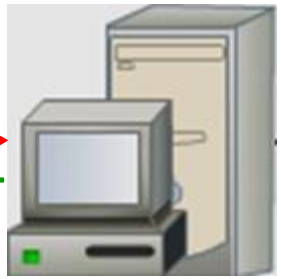
Client-Server Model

Step 1: Client sends a request to server

Step 2: Server analyzes the request then process it

Request can be
(file.html, file.txt
Script file -.asp, .aspx. .php, .jsp….
Execute a method of running object

Client



Response:
File.html, .txt,…
Result of processing

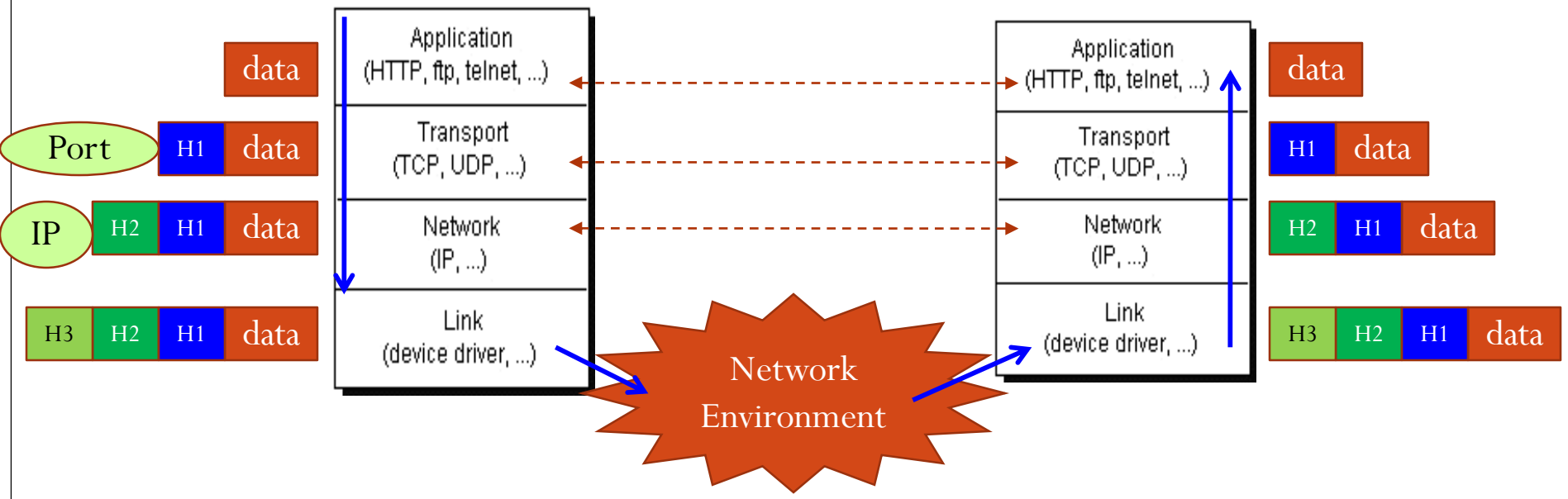
Client (can be a browser)

Server (can be a container, web container or Application container)

Step 3: Server sends response to client

Client-Server Model: Anatomy

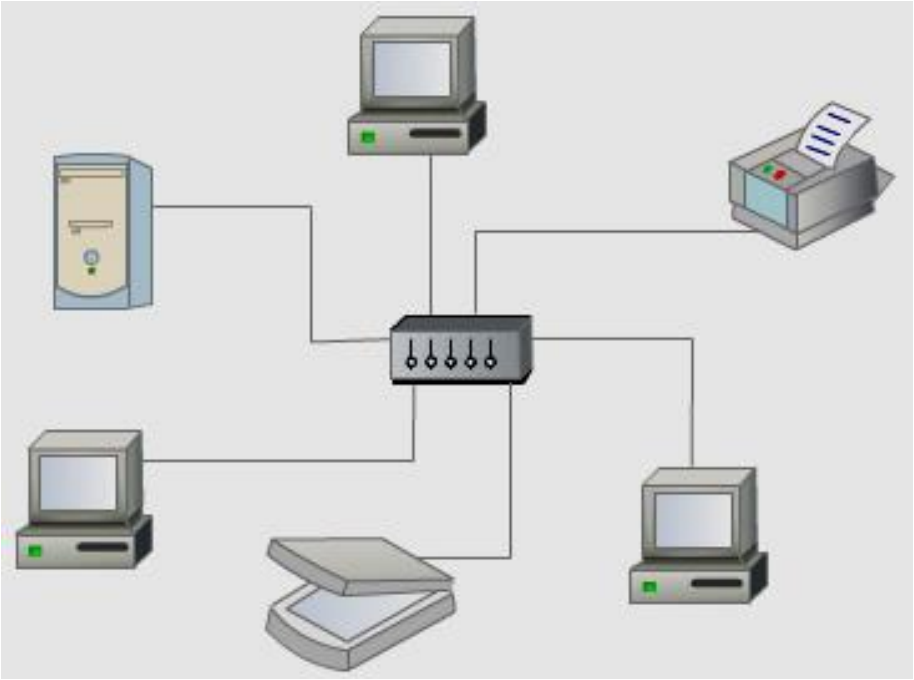
- Computers running on the Internet communicate to each other:



A package is attached an appropriate header (H-identifiable data) when it is transferred to each layer. A layer is an applications or a function library of network managing system

Client-Server Model: Anatomy…

- How to distinguish a computer in a network?



IP:152.3.21.121 or Hostname

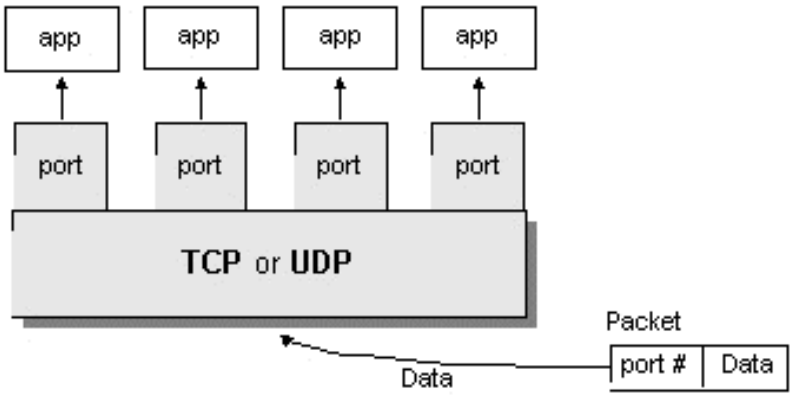
Personal computer IP: 127.0.0.1

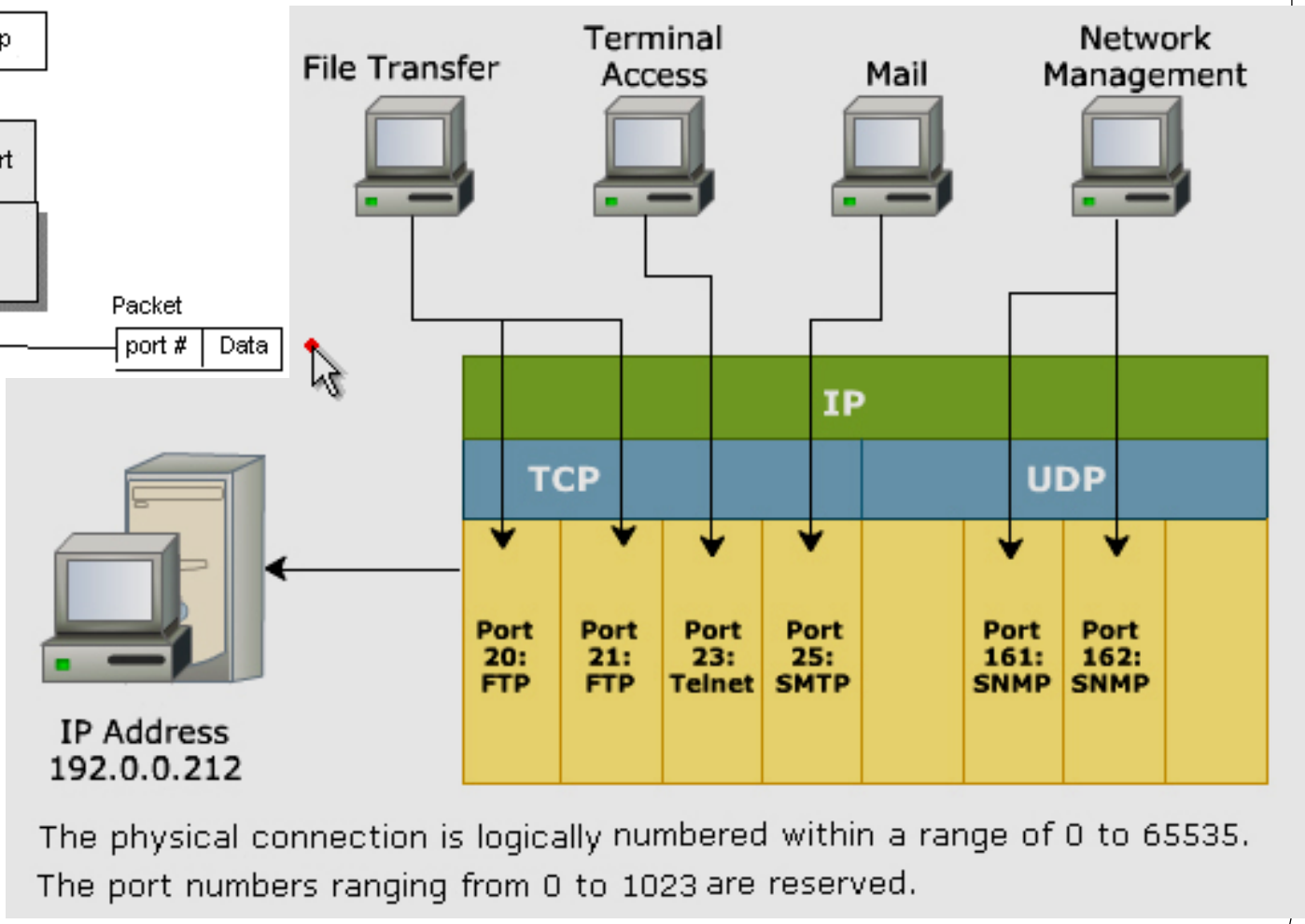
An IP address is either a 32-bit or 128-bit unsigned number used by IP, a lower-level protocol on which protocols like UDP and TCP are built. The IP address architecture is defined by *RFC 790:*

Client-Server Model: Anatomy…

How to distinguish a network-communicating process in a computer?

PORT





The physical connection is logically numbered within a range of 0 to 65535.

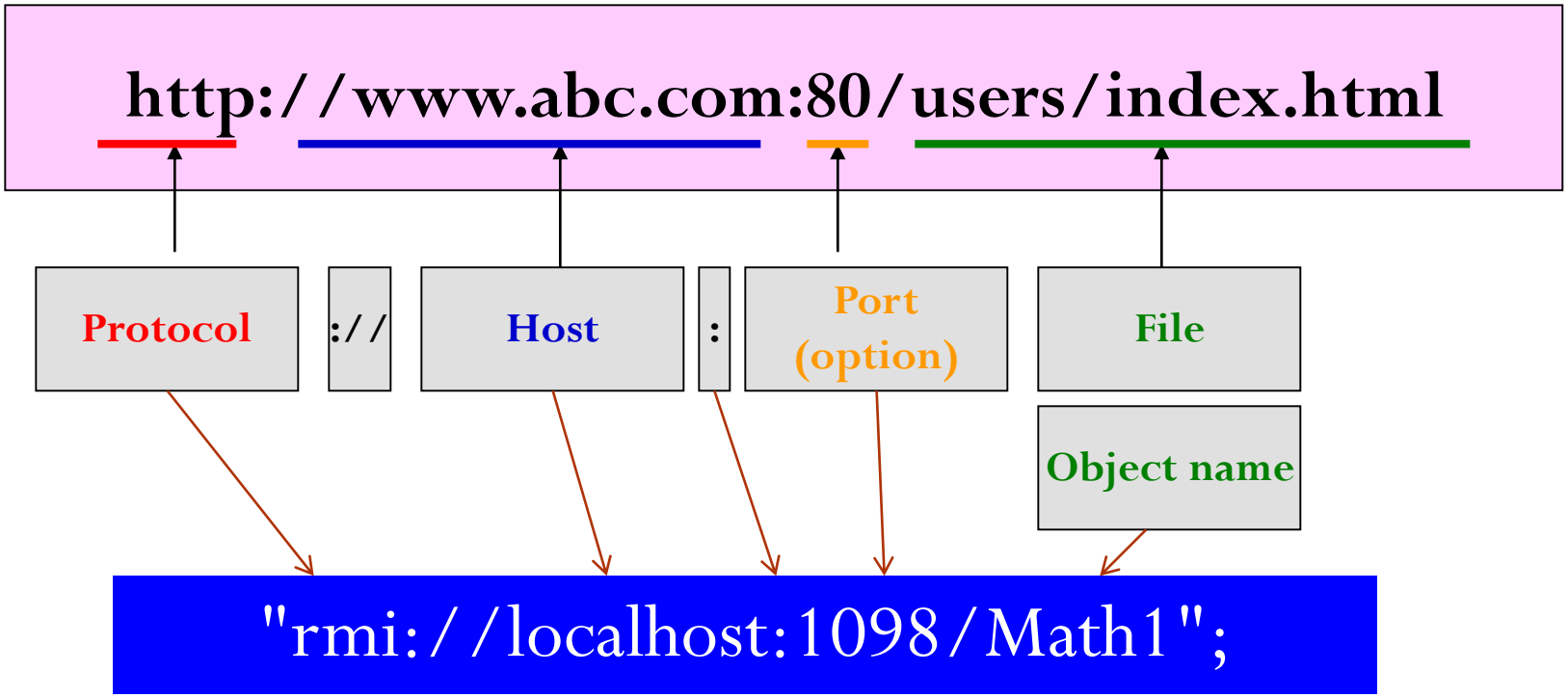
The port numbers ranging from 0 to 1023 are reserved.

Client-Server Model: Anatomy…

How to specify a resource in internet/network?

URL: Uniform Resource Locator

URN: Uniform Resource Name. It involves URL and pathname

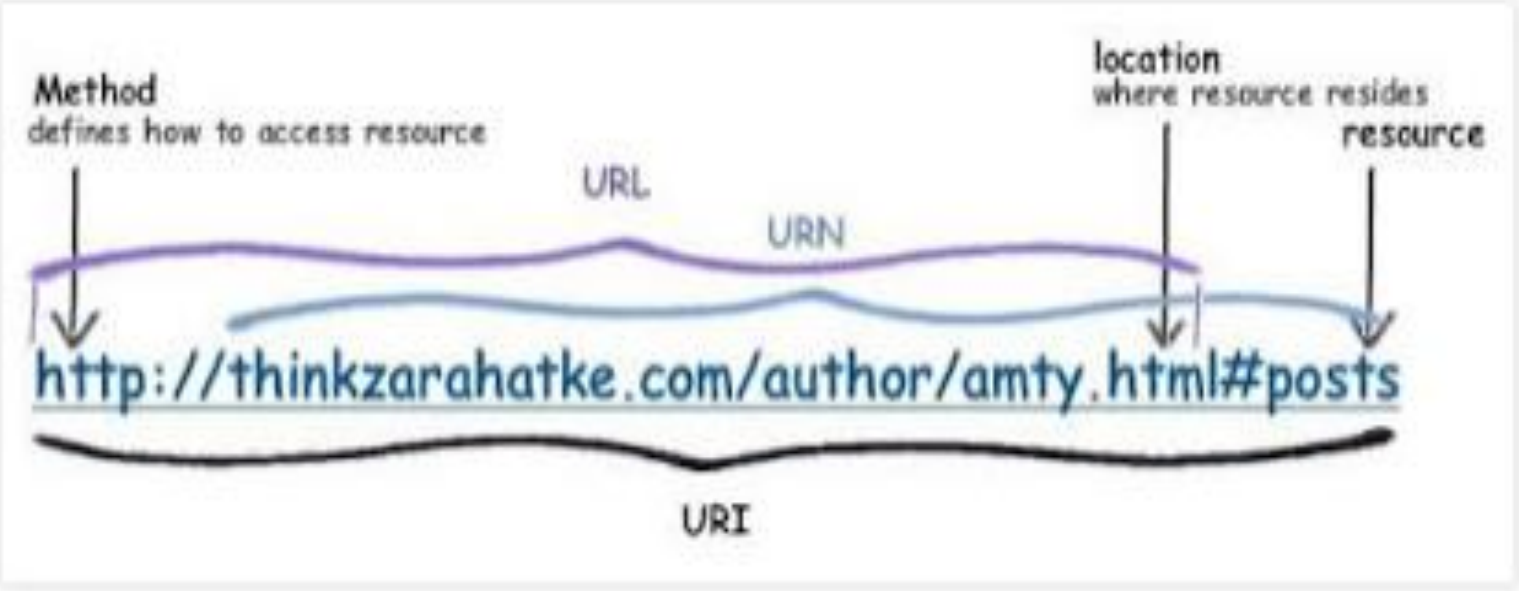


Client-Server Model: Anatomy…

How to specify a resource in internet/network?

URL: Uniform Resource Locator

URN: Uniform Resource Name. It involves URL and pathname



2- Working With URL

- The package ***java.net***
- The class ***java.net.URL***
- Demonstrations for using the URL and URLConnection classes to get contents from urls.

The java.net package

- It contains basic APIs for connecting computer networks.
- Reference: docs-Java8/api/java/net/package-tree.html
- Common used classes:
  - java.net.**URL** (implements java.io.Serializable)
  - java.net.**URLConnection** (abstract class)
    - java.net.**HttpURLConnection**
    - java.net.**JarURLConnection**
  - java.net.**URLDecoder**
  - java.net.**URLEncoder**
  - java.net.**URLStreamHandler**
  - java.net.**ServerSocket** (implements java.io.Closeable)
  - java.net.**Socket** (implements java.io.Closeable)
- This session will introduce the URL only.

The URL Class

- A URL takes the form of a string that describes how to find a resource on the Internet. URLs have two main components: the protocol needed to access the resource and the location of the resource.
- public final class **URL** extends Object implements Serializable
- **Constructors:**

  URL(**String** spec) Creates a URL object from the String representation.

  URL(**String** protocol, **String** host, int port, **String** file)Creates a URL object from the specified protocol, host, port number, and file.

  URL(**String** protocol, **String** host, int port, **String** file, **URLStreamHandler** handler)Creates a URL object from the specified protocol, host, port number, file, and handler.

  URL(**String** protocol, **String** host, **String** file)Creates a URL from the specified protocol name, host name, and file name.

  URL(**URL** context, **String** spec)Creates a URL by parsing the given spec within a specified context.

  URL(**URL** context, **String** spec, **URLStreamHandler** handler)Creates a URL by parsing the given spec with the specified handler within a specified context.

Demo 1: Parse a URL

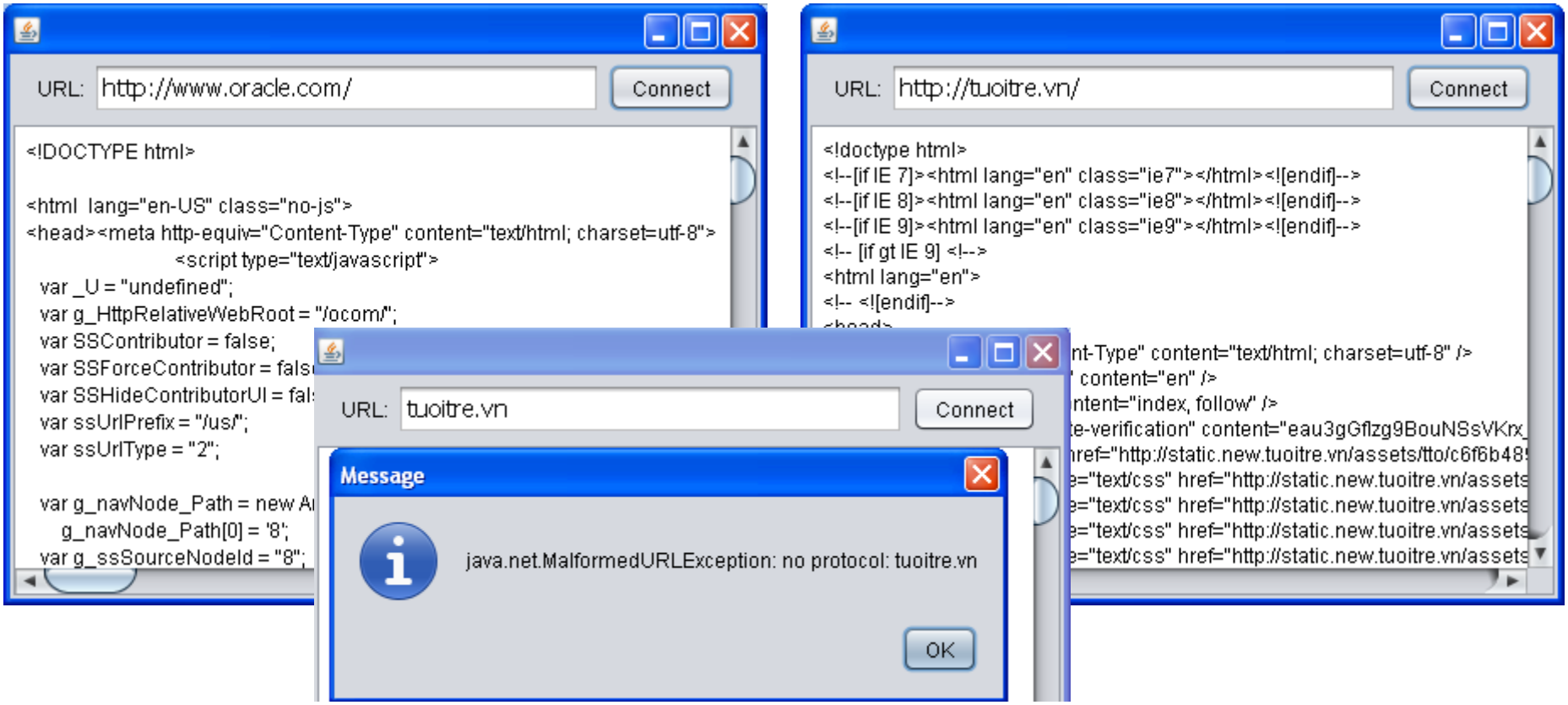
- This program will get components in a URL and no connection is carried out.

```
import java.io.IOException;
import java.net.URL;


public class ParseURL {

    public static void main(String [] args)
    {
        try
        {
            URL url = new URL("https://docs.oracle.com:80/javase/tutorial/networking/overview/index.html?name=networking#DOWLOADING");
            System.out.println("URL is: " + url.toString());// Trả về chuỗi url đầy đủ
            System.out.println("protocol is: "  + url.getProtocol());//Trả về giao thức của URL đó
            System.out.println("authority is: "  + url.getAuthority());//Trả về thành phần chính của URL đó
            System.out.println("file name is: " + url.getFile());//Trả về tên file của URL đó
            System.out.println("host is: " + url.getHost());//Trả về host của URL đó
            System.out.println("path is: " + url.getPath());//Trả về đường dẫn của URL đó
            System.out.println("port is: " + url.getPort());//Trả về cổng của URL đó
            System.out.println("default port is: " + url.getDefaultPort());//Trả về port mặc định cho protocol của URL đó
            System.out.println("query is: " + url.getQuery());//Trả về thành phần truy vấn của URL đó
            System.out.println("ref is: " + url.getRef());//Trả về thành phần tham chiếu của URL đó
        }catch( IOException e)
        {
            System.out.println("Can not define url");
        }
    }

}
```

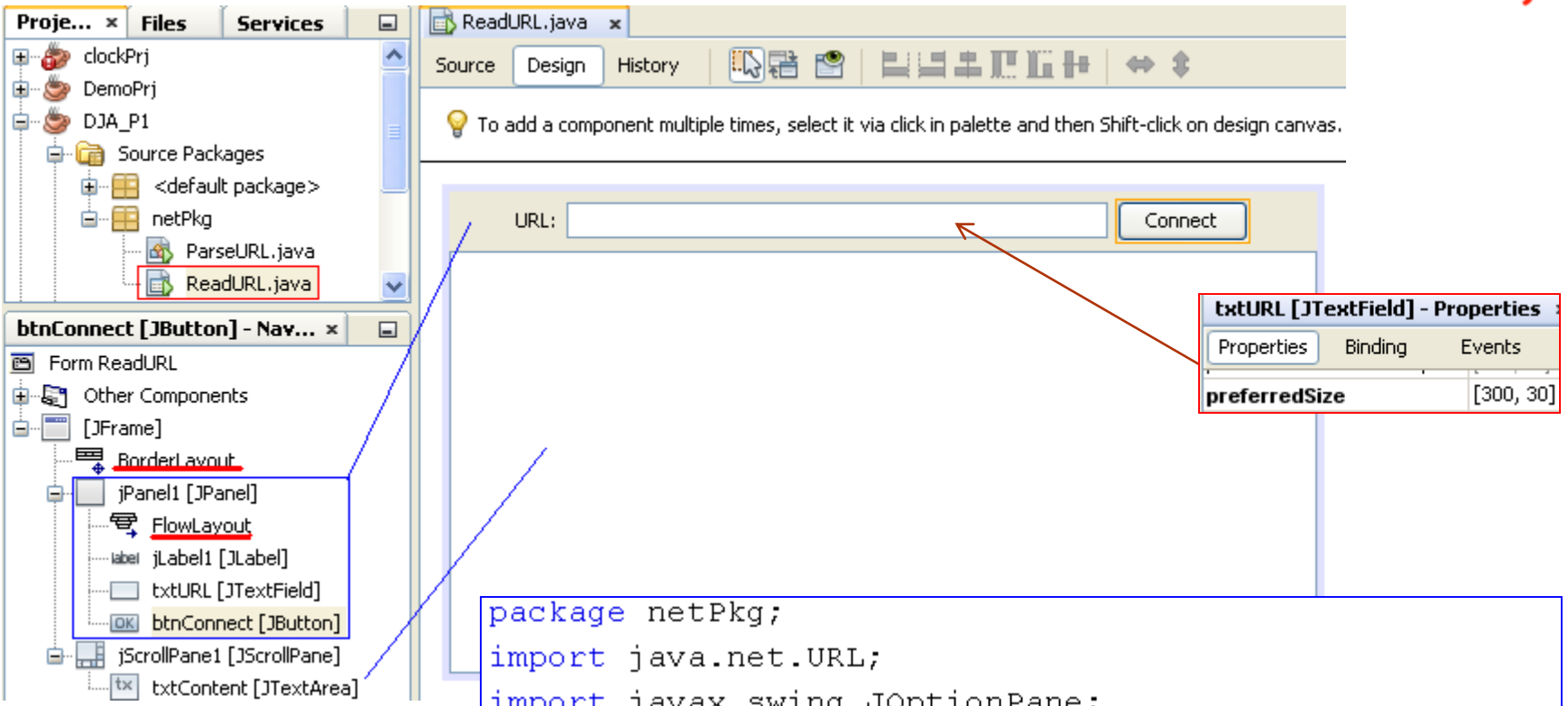
```
Output - JavaApplication1 (run)
URL is: https://docs.oracle.com:80/javase/tutorial/networking/overview/index.html?name=networking#DOWLOADING
protocol is: https
authority is: docs.oracle.com:80
file name is: /javase/tutorial/networking/overview/index.html?name=networking
host is: docs.oracle.com
path is: /javase/tutorial/networking/overview/index.html
port is: 80
default port is: 443
query is: name=networking
ref is: DOWLOADING
BUILD SUCCESSFUL (total time: 0 seconds)
```

Demo 2: Read a URL

In the following program, if user enters a URL then clicks the button **Connect**, the content of this URL will be shown. If inputted URL string is malformed, a message will be shown.



Demo 2: Read a URL...



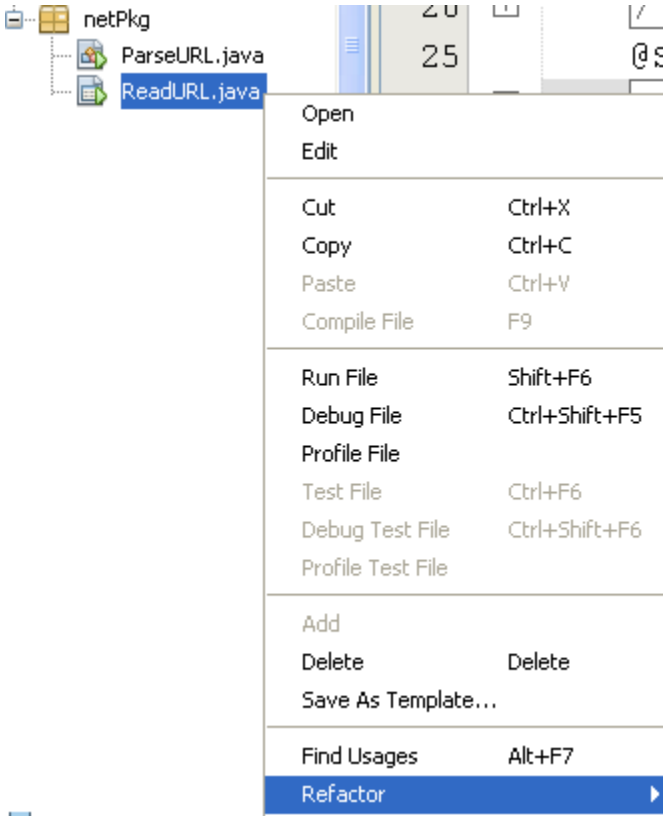

```
package netPkg;
import java.net.URL;
import javax.swing.JOptionPane;
import java.awt.Cursor;// for changing cursor
import java.io.*; // for reading data stream
public class ReadURL extends javax.swing.JFrame {
    /** Creates new form GetURLContents ...3 lines */
    public ReadURL() {
        initComponents();
        this.setSize(450, 300);
    }
```

Demo 2: Read a URL…

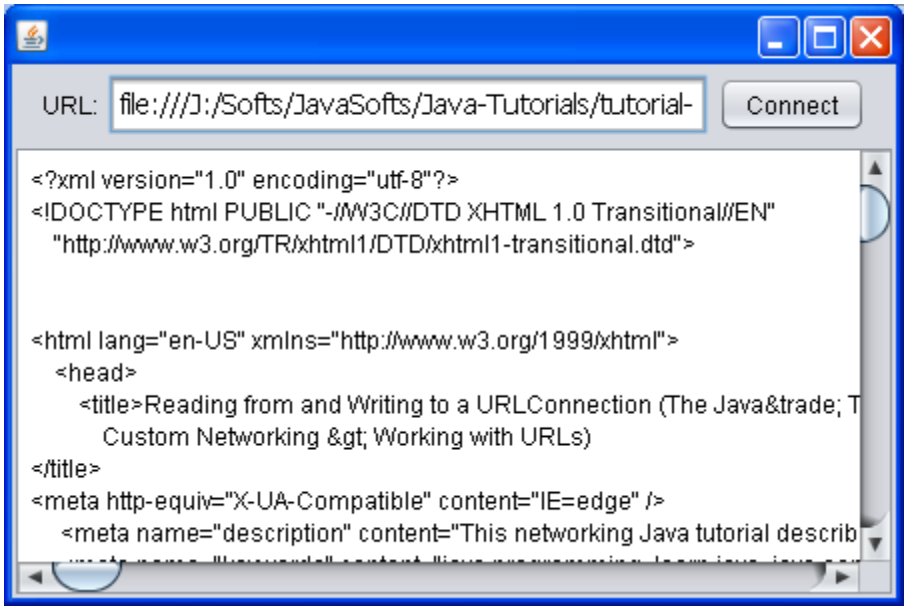
```
private String readContent (String urlString) throws Exception {
    String content="";
    // with directive throws, try catch can be missed
    // try {
    URL url= new URL (urlString);
    BufferedReader in = new BufferedReader(
            new InputStreamReader(url.openStream()));
    String inputLine;
    while ((inputLine = in.readLine()) != null)
        content += inputLine + "\n";
    in.close();
    // }
    return content;
}
```

```
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {
    // Exception may be thrown when the method readContent is called
    // Use thy catch
    try{
        this.setCursor(new Cursor(Cursor.WAIT_CURSOR));
        this.txtContent.setText(readContent(txtURL.getText()));
        this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
    }
    catch (Exception e){
        JOptionPane.showMessageDialog(this, e);
    }
}
```

Demo 3: Using URLConnection



Use the function REFRACTOR of NetBeans to copy and rename the class ReadURL to ReadURLConnection, modify code to gain the similar result as following:



Demo 3: Using URLConnection

```
package netPkg;
import java.net.URL;
import java.net.URLConnection;
import java.awt.Cursor;
import javax.swing.JOptionPane;
import java.io.*;
public class ReadURLConnection extends javax.swing.JFrame {
    /** Creates new form GetURLContents ...3 lines */
    public ReadURLConnection() {
        initComponents();
        this.setSize(450, 300);
    }
```

Demo 3: Using URLConnection

```
private String readContent (String urlString) throws Exception {
        String content="";
        URL url= new URL (urlString);
        URLConnection con = url.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(
                                        con.getInputStream()));
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            content += inputLine + "\n";
        in.close();
        return content;
    }
```

This code is modified from those in the previous demo.

```
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        this.setCursor(new Cursor(Cursor.WAIT_CURSOR));
        this.txtContent.setText(readContent(txtURL.getText()));
        this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
    }
    catch (Exception e){
        JOptionPane.showMessageDialog(this, e);
    }
}
```

This code is not different from those in the previous demo.

Summary

- IP and Port
- TCP, UDP Protocols
- Sockets and Ports
- Client Sockets/ Server Sockets in Java
- Object Streams and Serialization
- Remote Control using Object Streams
- Remote Method Invocation
  - Remote interface, Class for Server object
  - Server Program, Client Program

Thank You