

# Lecture 01

## Introduction to basic Java programming

CMPU 3027  
Network Programming

# Module Contents

- Use of TCP/IP
- Designing Applications for a distributed environment
- The client/server model and software design
  - Concurrency in clients and servers
  - Socket interface
  - Algorithms and issues in client software design, examples of client software
- Issues in server software design
  - Iterative connectionless servers,
  - Iterative connection-oriented servers,
  - concurrent connection-oriented servers,
- Remote procedure call mechanism
- Emerging networking technologies

# Reading Books

- Java Network Programming, Fourth Edition by Elliotte Rusty Harold
- Comer D.E, Stevens D.L. (2003) Interworking with TCP/IP Volume 3 Client-Server Programming And Applications
- Java: How to Program, 9th Edition by Deitel & Deitel

# Assessment Schedule

Week	Assessment	CA%
6	Theory Test	7.5
12	Theory Test	7.5
1-12	lab	15
14-15	Final Exam	70

# Blended Learning

- Online Class Meeting links
- Recorded Lectures
- Attendance

# Online Class Meeting links

 Networking Programming CMPU3027: 20...     Aneel Rahim  
as Student - View Only

My Home Progress Content Assessment Module Tools Library Help

## Meetings

### Active Meetings

Title	Scheduled At	Actions
Lab Week-1 Network Programming	9/22/2020, 2:00 PM	
Lecture Week-1 Thursday Network Programming	9/24/2020, 2:00 PM	
Lecture Week-1 Friday Network Programming	9/25/2020, 12:00 PM	

- Announcements
- Groups
- Discussions
- Virtual Classroom**
- Checklist
- FAQ

# Online Class Meeting links



Networking Programming CMPU3027: 2...



Aneel Rahim



My Home Progress ▾ Content Assessment ▾ Module Tools ▾ Library Help ▾

## Meetings

### Active Meetings

Title	Scheduled At	Actions
Lab Week-1 Network Programming	9/22/2020, 2:00 PM	⋮
Lecture Week-1 Thursday Network Programming	9/24/2020, 1:00 PM	Launch
Lecture Week-1 Friday Network Programming	9/25/2020, 1:00 PM	Copy External Link
Lab Week-2 Network Programming	9/29/2020, 2:00 PM	Manage Invites

# Recorded Lectures



Networking Programming CMPU3027: 2...



AR

Aneel Rahim



My Home Progress ▾ Content Assessment ▾ Module Tools ▾ Library Help ▾

Lecture Week-3 Friday Network Programming

10/9/2020, 12:00 PM

⋮

Lab Week-4 Network Programming

10/13/2020, 2:00 PM

⋮

Rows per page: 10 ▾ 1-10 of 36 < >

## Recorded Meetings

No recorded meetings yet

# Attendance <https://forms.gle/9zFdfEvxazAAhq6C9>

## Attendance Network Programming

Use this form to document your attendance.

\*Required

Your Name \*

Your answer

Your Student ID \*

Your answer

Indicate if it is Lecture or Lab

Choose

Date of Attendance \*

DD MM YYYY

\_\_ / \_\_ / \_\_

CMPU 3027 Network Programming

# JAVA Installation on Window 8.1

## 1. Check existing java version

**C:\Users>java -version**

java version "1.8.0\_31"

Java(TM) SE Runtime Environment (build 1.8.0\_31-b13)

Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)

## 2. Locate the java folder

**C:\Users>where java**

C:\ProgramData\Oracle\Java\javapath\java.exe

C:\Program Files\Java\jdk1.8.0\_25\bin\java.exe

# JAVA Installation on Window 8.1

## 3. Download JDK

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

## 4. Modify path and classpath variables

Go to control panel, then system and security

**Add to classpath**

C:\Program Files\Java\jdk1.8.0\_25\bin;;

**Add to path**

C:\Program Files\Java\jdk1.8.0\_25\bin;;

- Java SE
- Java EE
- Java ME
- Java SE Subscription
- Java Embedded
- Java Card
- Java TV
- Community
- Java Magazine

Overview Downloads Documentation Community Technologies Training

## Java SE Downloads



Java Platform (JDK) 10

### Java Platform, Standard Edition

#### Java SE 10.0.2

Java SE 10.0.2 is the latest feature release for the Java SE Platform  
[Learn more ▶](#)

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

JDK

[DOWNLOAD ▾](#)

Server JRE

[DOWNLOAD ▾](#)

JRE

[DOWNLOAD ▾](#)

### Java SDKs and Tools

- ⬇ [Java SE](#)
- ⬇ [Java EE and Glassfish](#)
- ⬇ [Java ME](#)
- ⬇ [Java Card](#)
- ⬇ [NetBeans IDE](#)
- ⬇ [Java Mission Control](#)

### Java Resources

- ⬇ [Java APIs](#)
- ⬇ [Technical Articles](#)
- ⬇ [Demos and Videos](#)
- ⬇ [Forums](#)
- ⬇ [Java Magazine](#)
- ⬇ [Developer Training](#)
- ⬇ [Tutorials](#)
- ⬇ [Java.com](#)



Control Panel Home

## View basic information about your computer



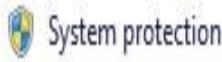
Device Manager

Windows edition



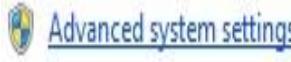
Remote settings

Windows 8.1



System protection

© 2013 Microsoft Corporation. All rights reserved.



Advanced system settings

Get more features with a new edition of Windows



## System

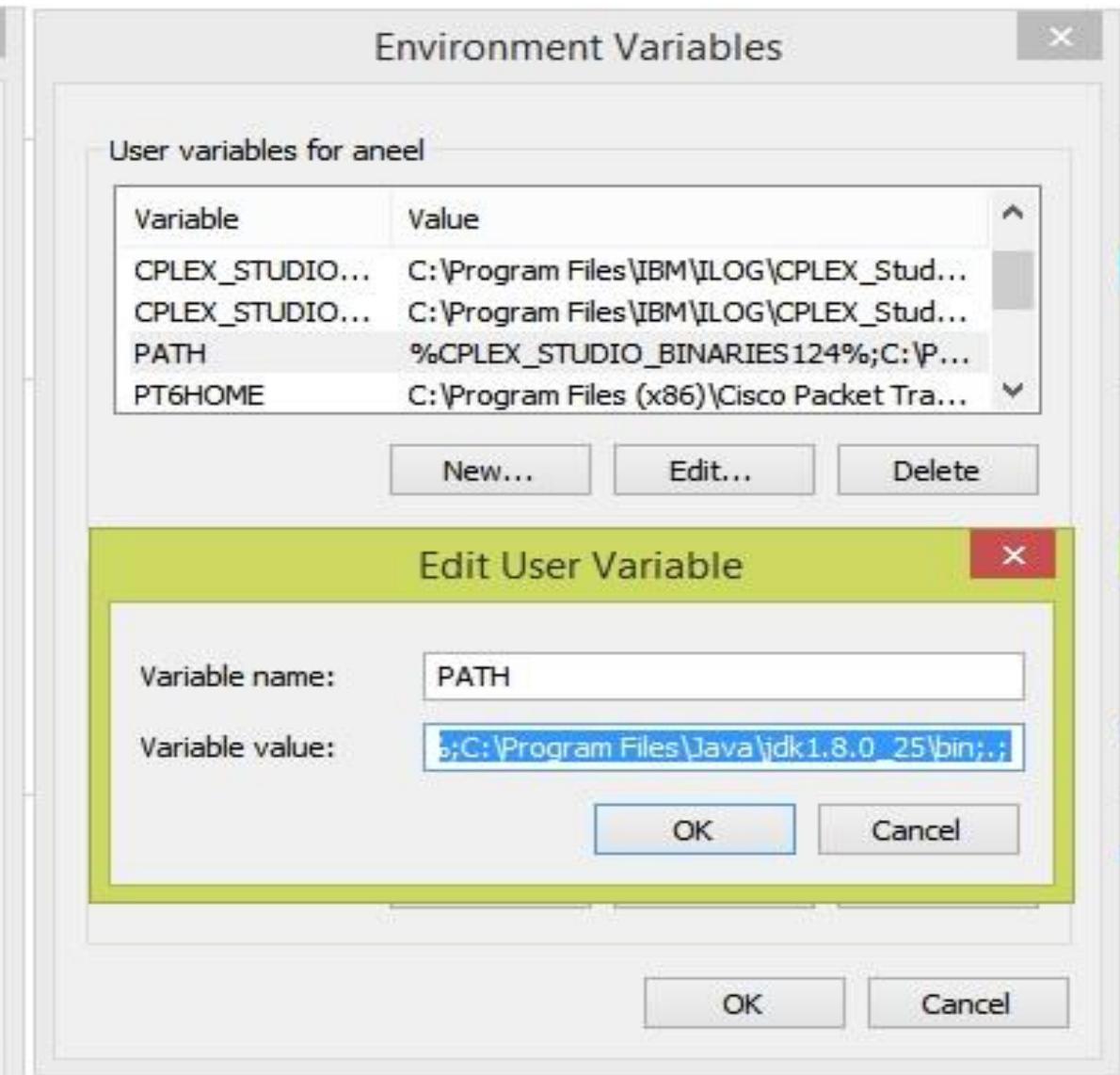
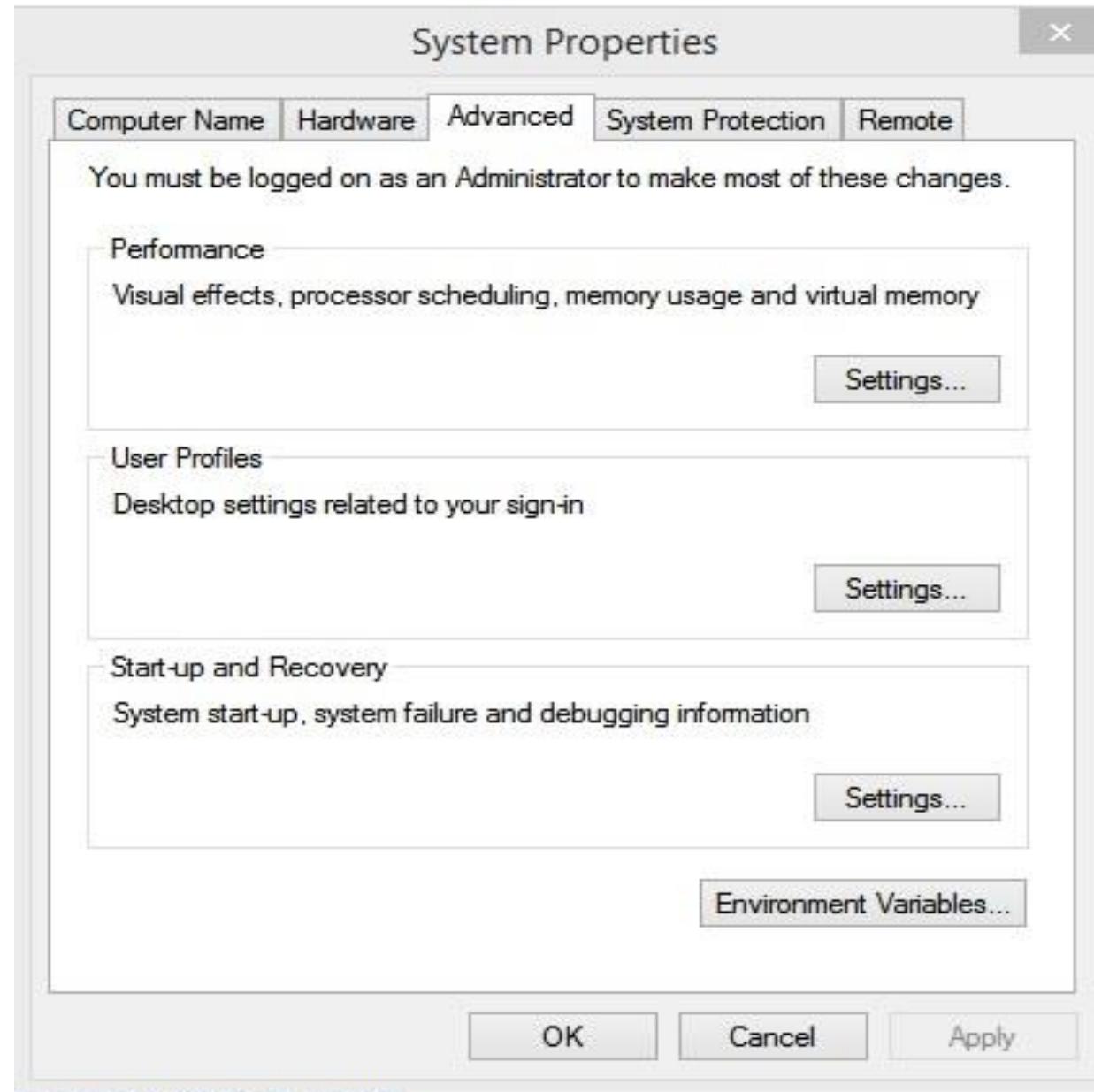
Processor: Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40 GHz

Installed memory (RAM): 6.00 GB (5.88 GB usable)

System type: 64-bit Operating System, x64-based processor

Pen and Touch: No Pen or Touch Input is available for this Display

Support Information



# Compile Java First Program

## 1. Create file with .java extension

HelloWorld.java is created using notepad

## 2. Compile the code

javac HelloWorld.java

## 3. Run the program

java HelloWorld

# 1. Write a program to print Hello World?

```
public class HelloWorld {  
    public static void main( String[] args ) {  
        System.out.println( "Hello World!" );  
    }  
}
```

## 2. Displays the sum of two numbers

```
import java.util.Scanner; // Program that displays the sum of two numbers.

public class Sum {

    public static void main( String[] args ){

        Scanner input = new Scanner( System.in );

        int number1; // first number to add

        int number2; // second number to add

        int sum; // sum of number1 and number2

        System.out.print( "Enter first integer: " ); // prompt

        number1 = input.nextInt(); // read first number from user

        System.out.print( "Enter second integer: " ); // prompt

        number2 = input.nextInt(); // read second number from user

        sum = number1 + number2; // add numbers, then store total in sum

        System.out.printf( "Sum is of two number is = %d\n", sum ); // display sum

    }

}
```

# Rules of Operator Precedence

- Operators in expressions contained within pairs of parentheses are evaluated first.
- Multiplication, division and remainder operations are applied next and have the same level of precedence.
- Addition and subtraction operations are applied last and have same precedence

# Rules of Operator Precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses on the same level (i.e., not nested), they are evaluated left to right.
*, / and %	Multiplication Division Modulus	Evaluated second. If there are several of this type of operator, they are evaluated from left to right.
+ or -	Addition Subtraction	Evaluated last. If there are several of this type of operator, they are evaluated from left to right.

**Fig: Precedence of arithmetic operators**

# Examples of Operator Precedence

Algebra:  $m = \frac{a + b + c + d + e}{5}$

Java: `m = ( a + b + c + d + e ) / 5;`

The following is an example of the equation of a straight line:

Algebra:  $y = mx + b$

Java: `y = m * x + b;`

# Examples of operator precedence

$$y = ax^2 + bx + c$$

Suppose variables a, b, c and x are initialized as follows:

a = 2, b = 3, c = 7 and x = 5.

y = a \* x \* x + b \* x + c;  
6 1 2 4 3 5

# Examples of operator precedence

Step 1.  $y = 2 * 5 * 5 + 3 * 5 + 7;$

$2 * 5$  is **10**

(Leftmost multiplication)

Step 2.  $y = 10 * 5 + 3 * 5 + 7;$

$10 * 5$  is **50**

(Leftmost multiplication)

Step 3.  $y = 50 + 3 * 5 + 7;$

$3 * 5$  is **15**

(Multiplication before addition)

Step 4.  $y = 50 + 15 + 7;$

$50 + 15$  is **65**

(Leftmost addition)

Step 5.  $y = 65 + 7;$

$65 + 7$  is **72**

(Last addition)

Step 6.  $y = 72;$

(Last operation—place **72** into **y**)

# Pseudocode

- Informal language that helps the programmers to develop the algorithms.
- Pseudocode programs are not actually executed on computers.
- Pseudocode is similar to everyday English.

# Algorithms

- Any computing problem can be solved by executing a series of actions in a specified order.
- An algorithm is a procedure for solving a problem in terms of
  - Actions to be executed
  - The order in which these actions are to be executed.

# Flowcharts

- A flowchart is a graphical representation of an algorithm.
- Flowcharts are drawn using certain special-purpose symbols, such as rectangles, diamonds, ovals and small circles;
- These symbols are connected by arrows called flowlines, which indicate the order in which the actions of the algorithm execute

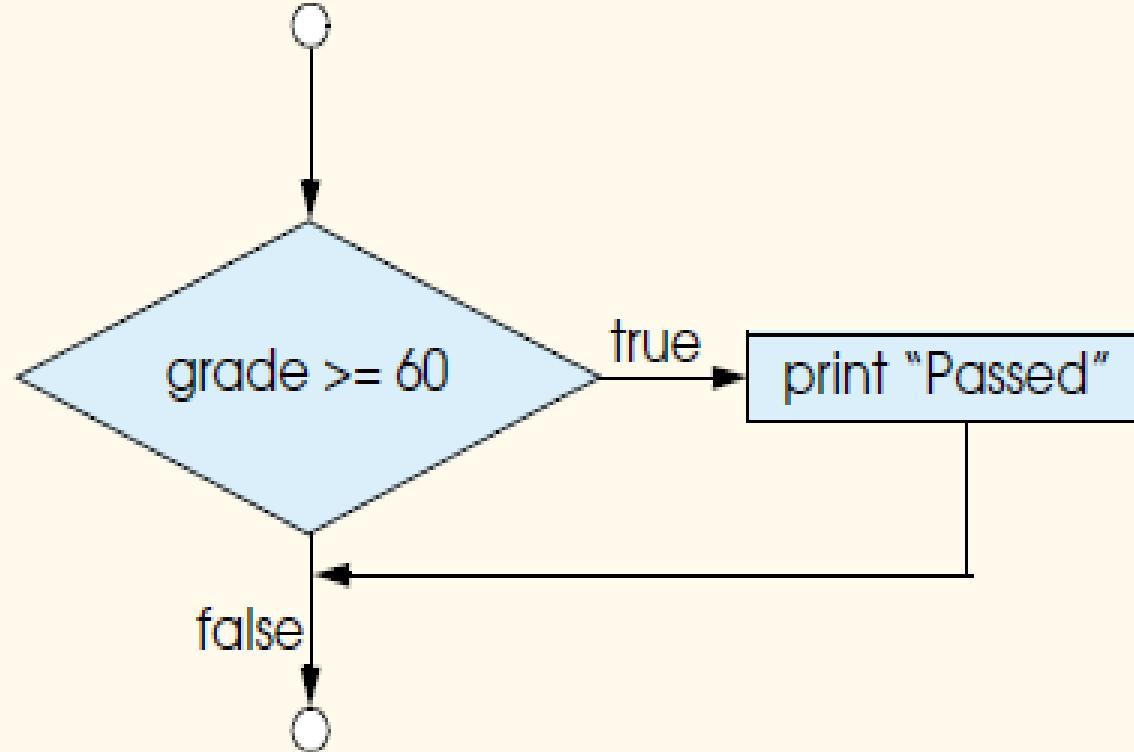
# if statement

A condition is an expression that can be true or false. Program to make a decision based on a condition's value.

*If student's grade is greater than or equal to 60  
Print "Passed"*

```
if ( studentGrade >= 60 )  
    System.out.println( "Passed" );
```

# Flow chart for if statement



### 3. Compare integers using if statements and relational operators

```
import java.util.Scanner; // Compare integers using relational operators
public class Comparison {
    public static void main( String[] args ){
        Scanner input = new Scanner( System.in );
        int number1; // first number to add
        int number2; // second number to add
        int sum; // sum of number1 and number2
        System.out.print( "Enter first integer: " ); // prompt
        number1 = input.nextInt(); // read first number from user
        System.out.print( "Enter second integer: " ); // prompt
        number2 = input.nextInt(); // read second number from user
```

### 3. Compare integers using if statements and relational operators

```
if ( number1 == number2 )
    System.out.printf( "%d == %d\n", number1, number2 );
if ( number1 != number2 )
    System.out.printf( "%d != %d\n", number1, number2 );
if ( number1 < number2 )
    System.out.printf( "%d < %d\n", number1, number2 );
if ( number1 > number2 )
    System.out.printf( "%d > %d\n", number1, number2 );
if ( number1 <= number2 )
    System.out.printf( "%d <= %d\n", number1, number2 );
if ( number1 >= number2 )
    System.out.printf( "%d >= %d\n", number1, number2 );
}
```

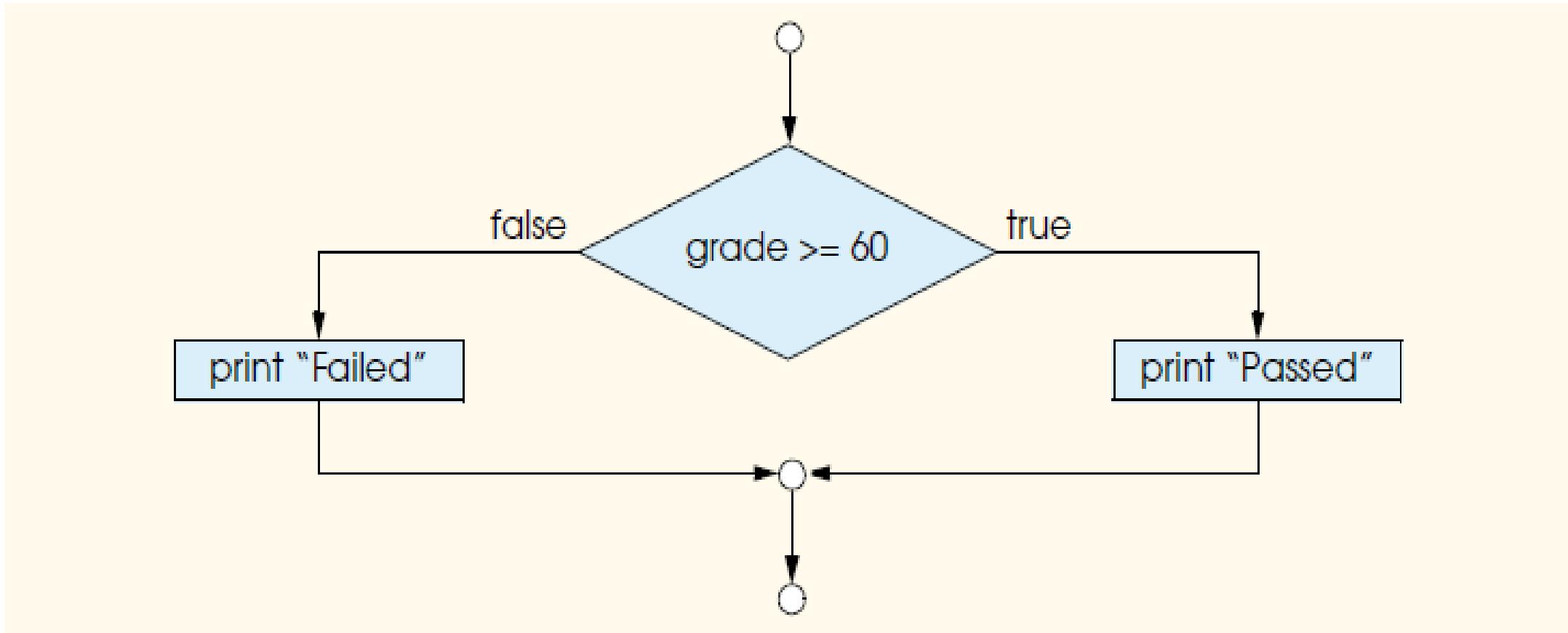
# if else statement

- It allows the programmer to specify that a different action is to be performed when a condition is true from an action when a condition is false.

```
If student's grade is greater than or equal to 60  
    Print "Passed"  
Else  
    Print "Failed"
```

```
if ( grade >= 60 )  
    System.out.println( "Passed" );  
else  
    System.out.println( "Failed" );
```

# Flow chart for if else statement



## 4. Show the use of if and if-else statement

```
import java.util.Scanner; // program uses class Scanner
public class IfElseExample{
    public static void main( String[] args ){
        Scanner input = new Scanner( System.in );
        int grade; // first number to add
        System.out.print( "Enter the student grade: " ); // prompt
        grade = input.nextInt(); // read first number from user

        if ( grade >= 60 )
            System.out.println( "Passed" );
        else
            System.out.println( "Failed" );
    }
}
```

# Repetition

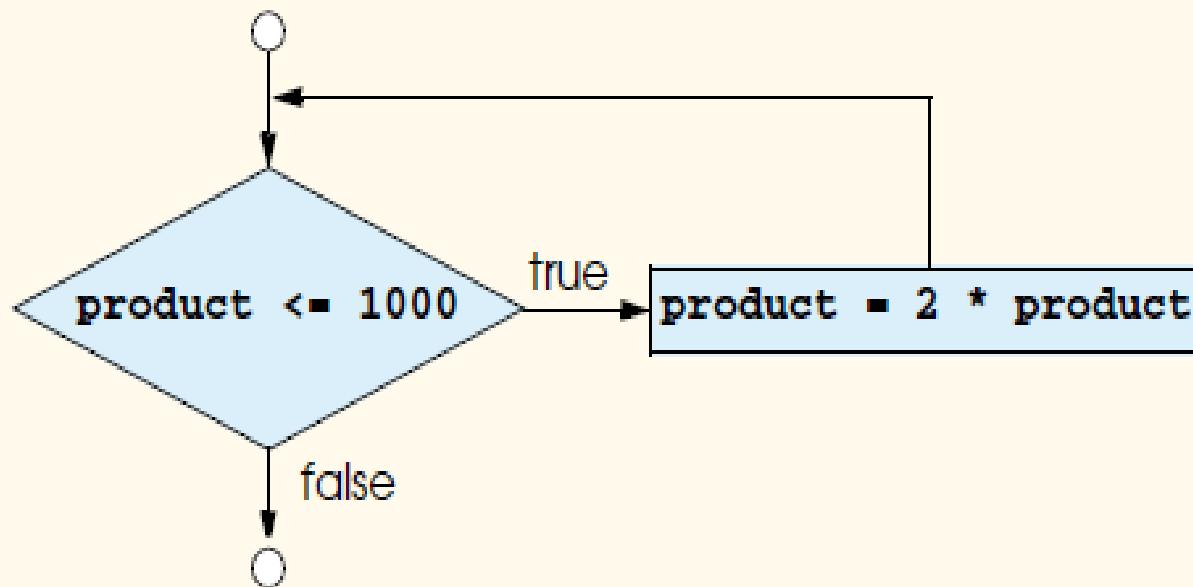
- **While loop**

- A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

```
product = 2

while product <= 1000:
    product = 2 * product
```

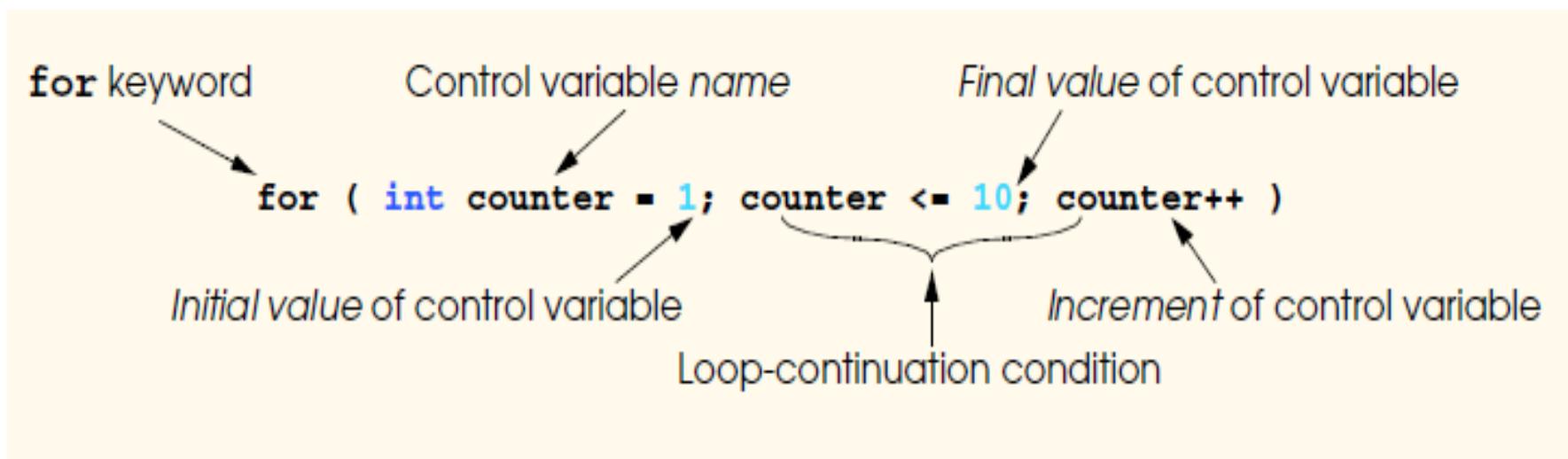
# Flow chart for while loop



## 5. Show the use of while loop statement

```
public class WhileExample {  
    public static void main( String[] args ) {  
        int product = 2;  
        while ( product <= 1000 ) {  
            product = 2 * product;  
            System.out.printf( "The product  
value is = %d \n", product );  
        }  
    }  
}
```

# for loop



# 6. Show the use of for loop statement

```
public class ForExample {  
    public static void main( String[] args ) {  
        int total = 0;  
        for ( int number = 0; number <= 20; number += 2 ) {  
            total += number;  
            System.out.printf( "The Sum is = %d \n", total );  
        }  
    }  
}
```

# Assignment Operators

Assignment operator	Sample expression	Explanation	Assigns
<i>Assume: int c = 3, d = 5, e = 4, f = 6, g = 12;</i>			
<code>+=</code>	<code>c += 7</code>	<code>c = c + 7</code>	10 to c
<code>-=</code>	<code>d -= 4</code>	<code>d = d - 4</code>	1 to d
<code>*=</code>	<code>e *= 5</code>	<code>e = e * 5</code>	20 to e
<code>/=</code>	<code>f /= 3</code>	<code>f = f / 3</code>	2 to f
<code>%=</code>	<code>g %= 9</code>	<code>g = g % 9</code>	3 to g

**Fig. 4.13** | Arithmetic compound assignment operators.

# Increment Operators

Operator	Operator name	Sample expression	Explanation
<code>++</code>	prefix increment	<code>++a</code>	Increment <b>a</b> by 1, then use the new value of <b>a</b> in the expression in which <b>a</b> resides.
<code>++</code>	postfix increment	<code>a++</code>	Use the current value of <b>a</b> in the expression in which <b>a</b> resides, then increment <b>a</b> by 1.
<code>--</code>	prefix decrement	<code>--b</code>	Decrement <b>b</b> by 1, then use the new value of <b>b</b> in the expression in which <b>b</b> resides.
<code>--</code>	postfix decrement	<code>b--</code>	Use the current value of <b>b</b> in the expression in which <b>b</b> resides, then decrement <b>b</b> by 1.

**Fig. 4.14** | Increment and decrement operators.

## 7. Show the use of prefix and postfix

```
public class Increment {  
    public static void main( String[] args ) {  
        int c = 5; // assign 5 to c  
        System.out.println( c );  
        System.out.println( c++ );  
        System.out.println( c );  
        System.out.println(); // skip a line  
  
        c = 5; // demonstrate prefix increment operator  
        System.out.println( c );  
        System.out.println( ++c );  
        System.out.println( c );  
    } // end main  
} // end class Increment
```

# Lecture 02

# TCP/IP Model

CMPU 3027  
Network Programming

# OSI Model

<b>Application</b>	End user application programs e.g. front end application Firefox, email or spreadsheet Back end application file server, database server.
<b>Presentation</b>	Data formatting, Data compression and Data encryption. Convert data into universal format so that other computer can understand. JPEG, GIF
<b>Session</b>	If everyone start talking at once, no one will hear anything. It defines how to start, control and end conversation (session). Manages who can send the data at certain time and for how long.
<b>Transport</b>	It defines how the data is set. Reliable or unreliable (TCP or UDP). It prepares your data for transmission over the network. Your computer communicate with receiving computer to decide how to break your data into small pieces and make sure that no frame will be lost. Application separation on the basis of port number.
<b>Network</b>	It provides logical addressing used the router. It decide the data should go.
<b>Data Link</b>	It deals with physical addressing i.e. MAC address .Switch use this address to forward data to other nodes in same network.
<b>Physical</b>	It deals with electrical signals. Sending 0s and 1s.

# TCP/IP Model

<b>Application</b>	Do useful work like web browsing, email and file transfer.
<b>Transport</b>	Transport the data between client and server applications.
<b>Internet</b>	Route the packets between the networks.
<b>Data Link</b>	Send data within local network
<b>Physical</b>	Specify hardware characteristics. Sending 0s and 1s.

Characteristic	TCP	UDP
Connection	Connection-oriented	Connectionless
Reliability	Higher	Lower
Order of packets	Order restored	Order potentially lost
Data boundaries	Packets are merged	Packets are distinct
Transmission time	Slower than UDP	Faster than TCP
Error checking	Yes	Yes, but no recovery options
Acknowledgement	Yes	No
Weight	Heavy weight requiring more support	Light weight requiring less support

# Use of TCP/IP Model

- History of TCP/IP
- Began with basic services
  - remote login
  - file transfer
  - electronic mail
- Now activity concentrated on distributed applications

# List of TCP/IP Application Layer protocols

- **FTP** file transfer protocol
- **SMTP** simple mail transfer program
- **Telnet** for remote login
- **SSH** for secure login
- **HTTP** for web pages
- **DNS** domain name server

# File Transfer Protocols

- Standard Internet protocol to transfer file from one computer to other on Internet
- Using FTP, a user can upload, download, delete and copy files on a server.
- User name and password is required for FTP server, however some servers provide access without login called anonymous ftp

# Steps for Using FTP on windows

1. Open cmd prompt
2. Write command ftp domain name
3. Enter user/password
4. Run FTP build in commands

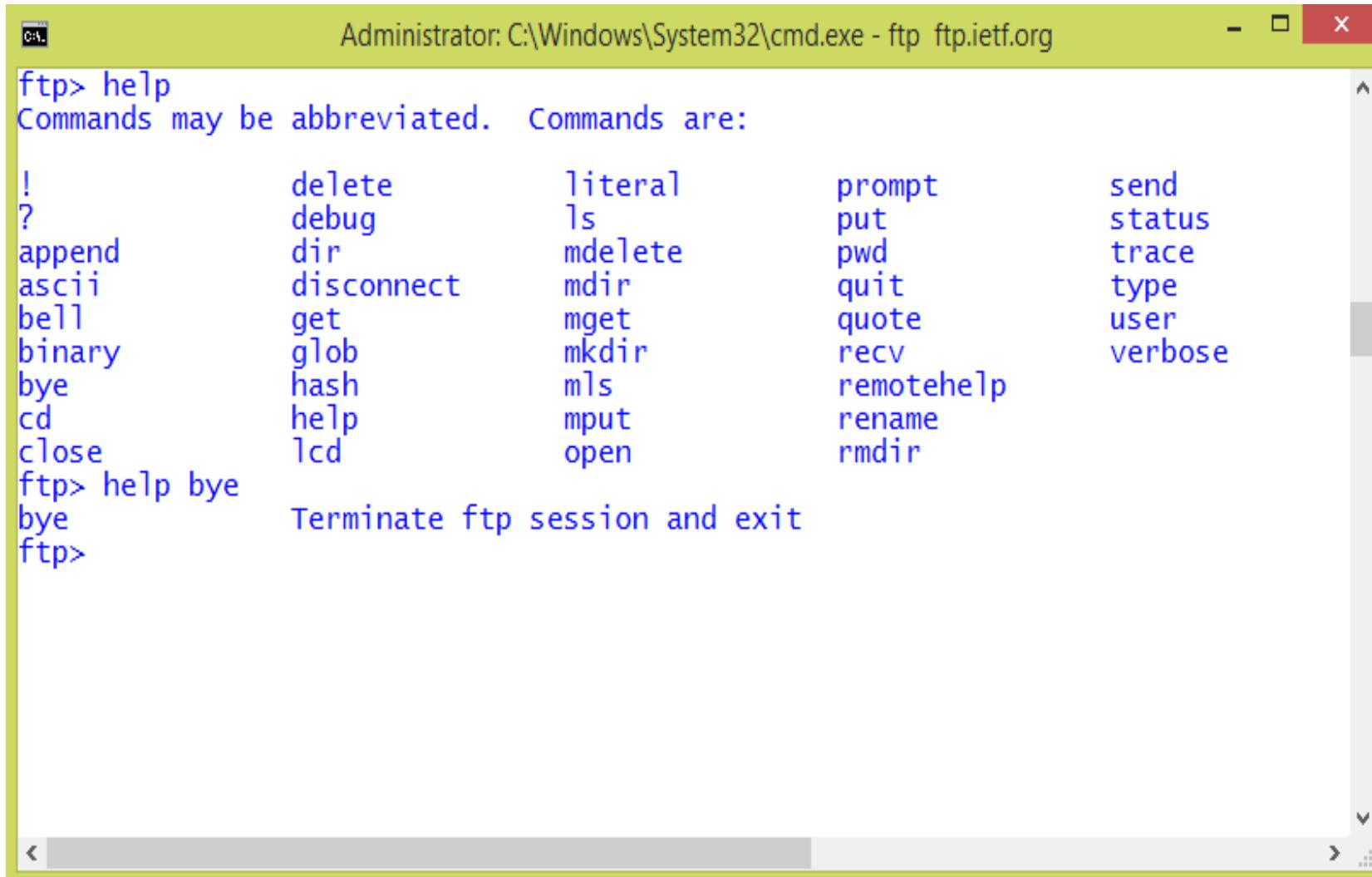
- Example

1. Open cmd prompt
2. ftp ftp.ietf.org
3. username anonymous
4. cd rfc get rfc0978.txt

# Using ftp to download the file

```
Administrator: C:\Windows\System32\cmd.exe - ftp ftp.ietf.org
E:\DIT\Courses\Network-Programming\Lectures\Lecture-02\code>ftp ftp.ietf.org
Connected to ietf.org.
220 FTP server ready
User (ietf.org:(none)): anonymous
331 Anonymous Login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
ftp> cd rfc
250-*=====
* * This directory is maintained by the RFC Editor. If you experience *
* any problems, please report them to rfc-editor@rfc-editor.org. *
* *=====
250 CWD command successful
ftp> get rfc0978.txt
200 PORT command successful
150 Opening ASCII mode data connection for rfc0978.txt (8933 bytes)
226 Transfer complete
ftp: 9223 bytes received in 6.50Seconds 1.42Kbytes/sec.
ftp> -
```

# ftp commands



Administrator: C:\Windows\System32\cmd.exe - ftp ftp.ietf.org

```
ftp> help
Commands may be abbreviated. Commands are:

!          delete      literal      prompt      send
?          debug       ls           put         status
append     dir        mdelete    pwd         trace
ascii      disconnect  mdir        quit        type
bell       get         mget        quote       user
binary     glob        mkdir      recv        verbose
bye        hash        mls         remotehelp
cd         help        mput      rename
close      lcd         open       rmdir
ftp> help bye
bye        Terminate ftp session and exit
ftp>
```

# SMTP (Simple Mail transfer Protocol)

- It is a TCP/IP protocol used in sending and receiving e-mail
- Port number 25
- Port number is not a hardware device but a logical integer value

# Telnet

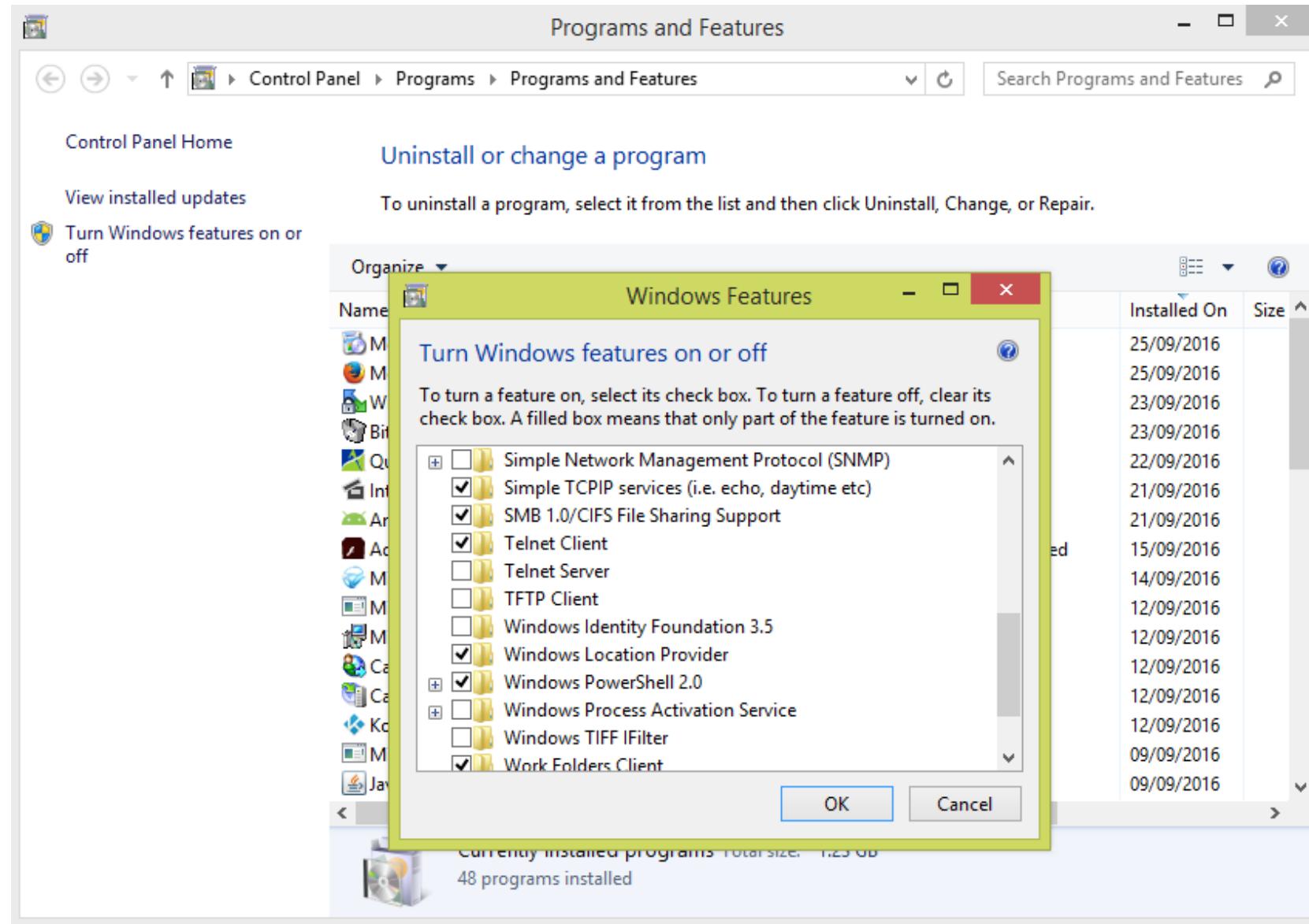
- It is standard application for remote login.
- Unsecure way of managing the devices.
- Port number 23

# Steps for Using Telnet on windows

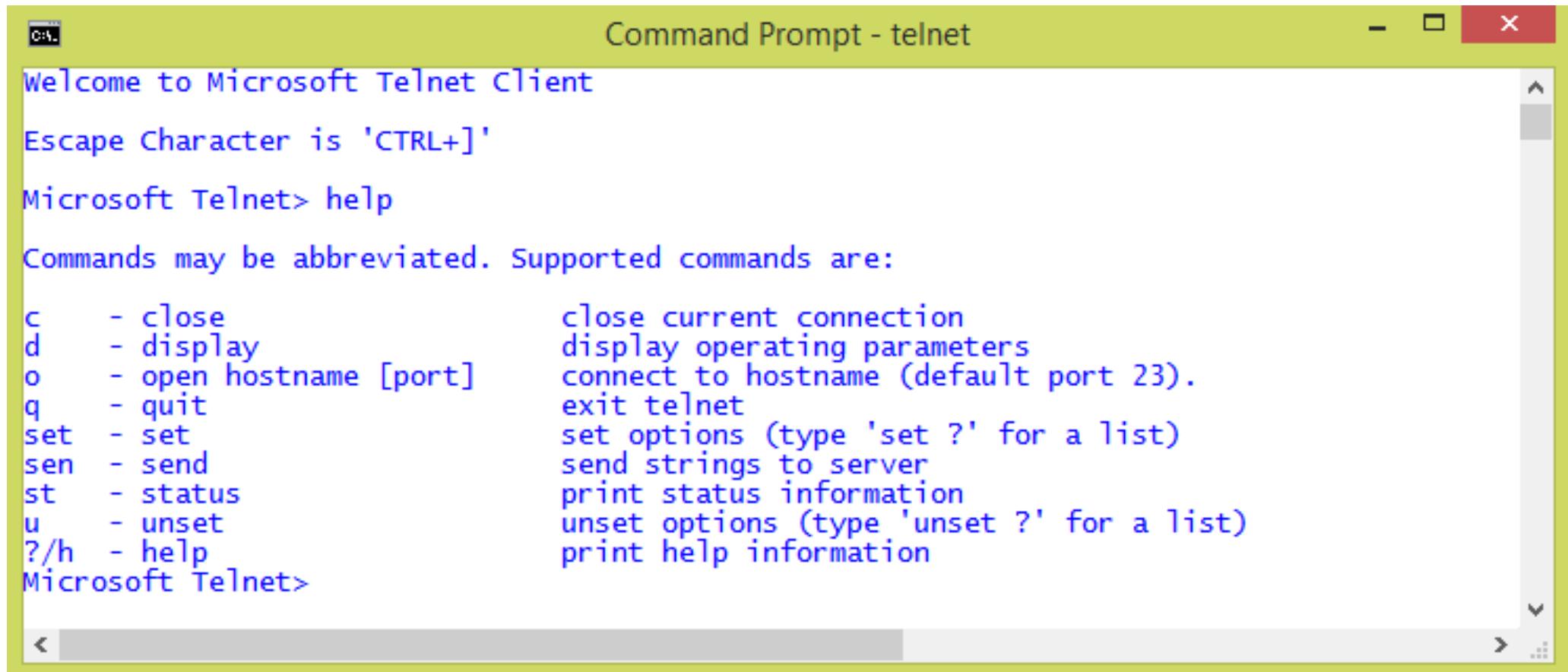
1. Activate telnet client using **"Turn Windows features on or off"**.
2. Open cmd prompt
3. Write **Telnet** and press enter.
4. Enter **help** for commands list
5. Write **open telehack.com 23**

The command **open hostname [port]** will connect to hostname (default port 23)

# Telnet Client in Windows...



# Telnet Client in Windows...



The screenshot shows a Microsoft Telnet Client window titled "Command Prompt - telnet". The window displays the following text:

```
C:\> Welcome to Microsoft Telnet Client
      Escape Character is 'CTRL+['
Microsoft Telnet> help
Commands may be abbreviated. Supported commands are:
c  - close          close current connection
d  - display        display operating parameters
o  - open hostname [port] connect to hostname (default port 23).
q  - quit           exit telnet
set - set           set options (type 'set ?' for a list)
sen - send          send strings to server
st  - status         print status information
u  - unset          unset options (type 'unset ?' for a list)
?/h - help          print help information
Microsoft Telnet>
```

# Telnet Client in Windows...

```
Connected to TELEHACK port 46
It is 2:09 am on Saturday, September 23, 2017 in Mountain View, California, USA.
There are 29 local users. There are 26632 hosts on the network.

Type HELP for a detailed command list.
Type NEWUSER to create an account.

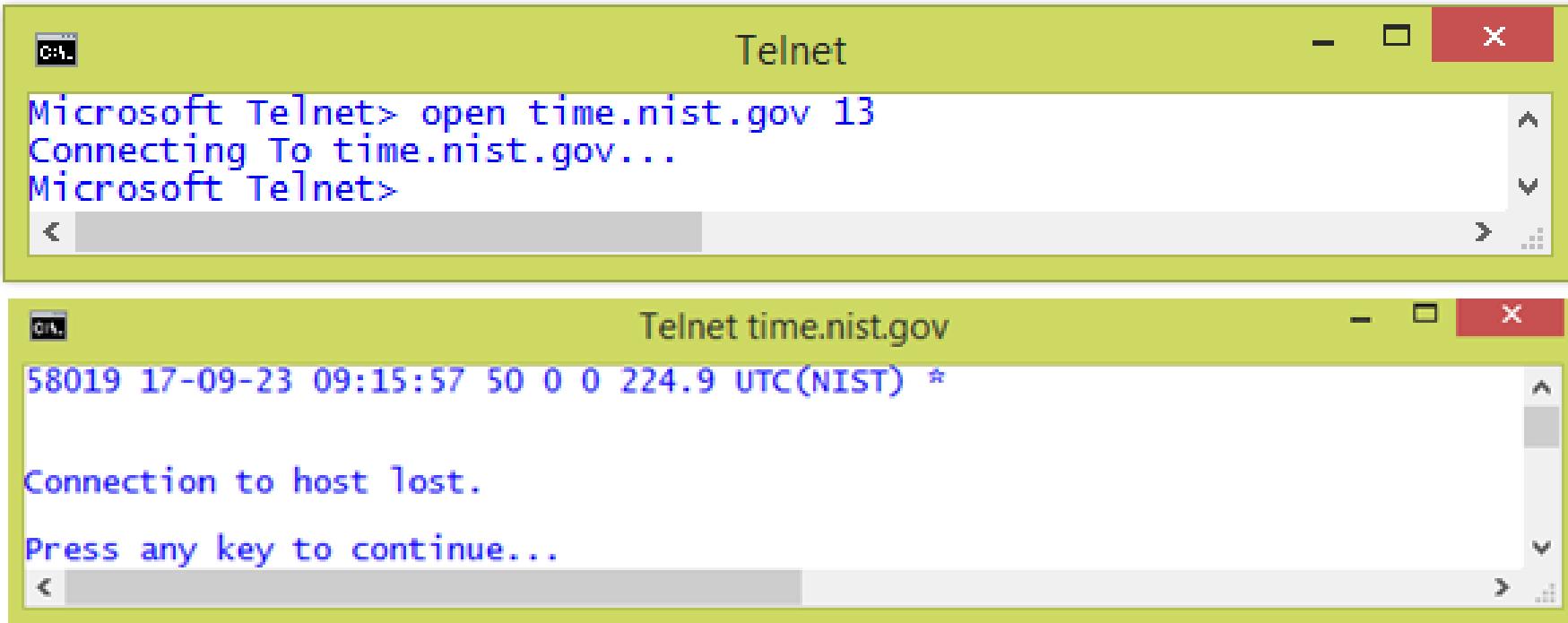
May the command line live forever.

Command, one of the following:
?          a2      advent      areacode    basic      bf
cal        calc     ching       clear       clock     cowsay
date       echo     eliza      factor      figlet    finger
fnord      geoip    help       hosts      ipaddr   joke
login      mac     md5       morse      newuser  notes
octopus    phoon    pig       ping       primes   privacy
qr         rain     rand      rfc       rig      roll
rot13      sleep    starwars  traceroute units   uptime
usenet     users    uumap     uupath   uplot   weather
when      zipcode  zork      zrun

.cal
      August 2017          September 2017         October 2017
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
  1  2  3  4  5           3  4  5  6  7  8  9   1  2  3  4  5  6  7
  6  7  8  9 10 11 12    10 11 12 13 14 15 16  8  9 10 11 12 13 14
13 14 15 16 17 18 19    17 18 19 20 21 22 23 15 16 17 18 19 20 21
20 21 22 23 24 25 26    24 25 26 27 28 29 30  22 23 24 25 26 27 28
27 28 29 30 31

--
```

# Telnet Client talking to time.nist.gov



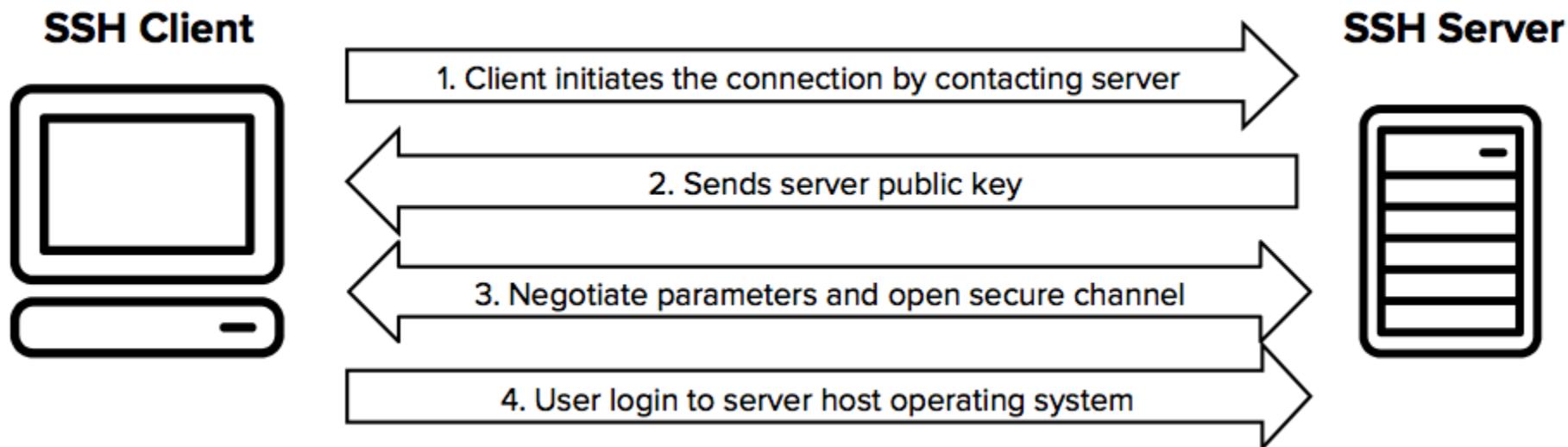
The image displays two overlapping Microsoft Telnet windows. The top window is titled "Telnet" and shows the command "Microsoft Telnet> open time.nist.gov 13" followed by "Connecting To time.nist.gov...". The bottom window is titled "Telnet time.nist.gov" and shows the output "58019 17-09-23 09:15:57 50 0 0 224.9 UTC(NIST) \*". Below this, it displays "Connection to host lost." and "Press any key to continue...". Both windows have standard Windows-style title bars and scrollbars.

```
Microsoft Telnet> open time.nist.gov 13
Connecting To time.nist.gov...
Microsoft Telnet>

58019 17-09-23 09:15:57 50 0 0 224.9 UTC(NIST) *
Connection to host lost.
Press any key to continue...
```

# SSH (Secure Shell)

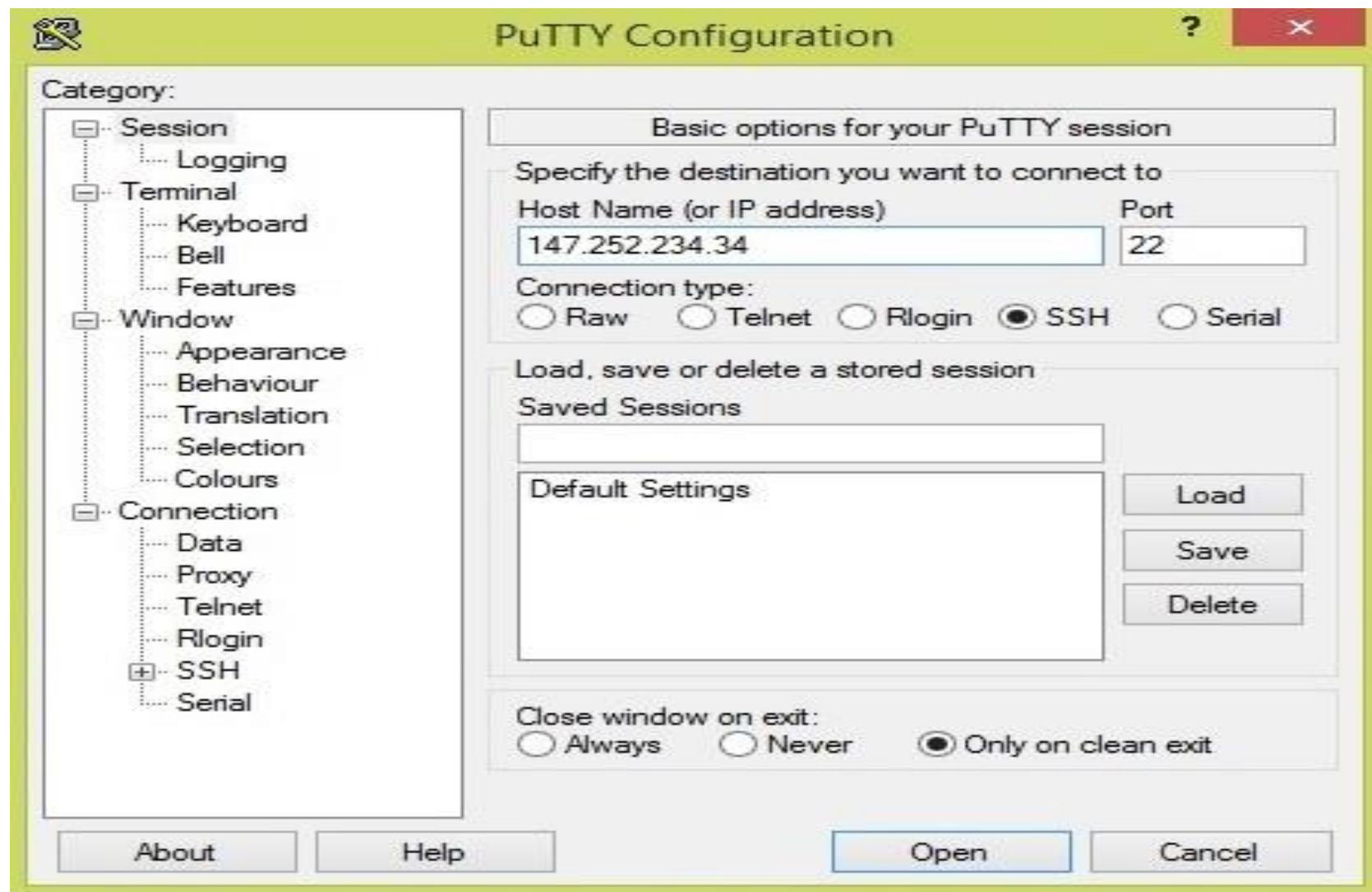
- It is also used for remote login and use port 22.
- Secure way of managing the devices.



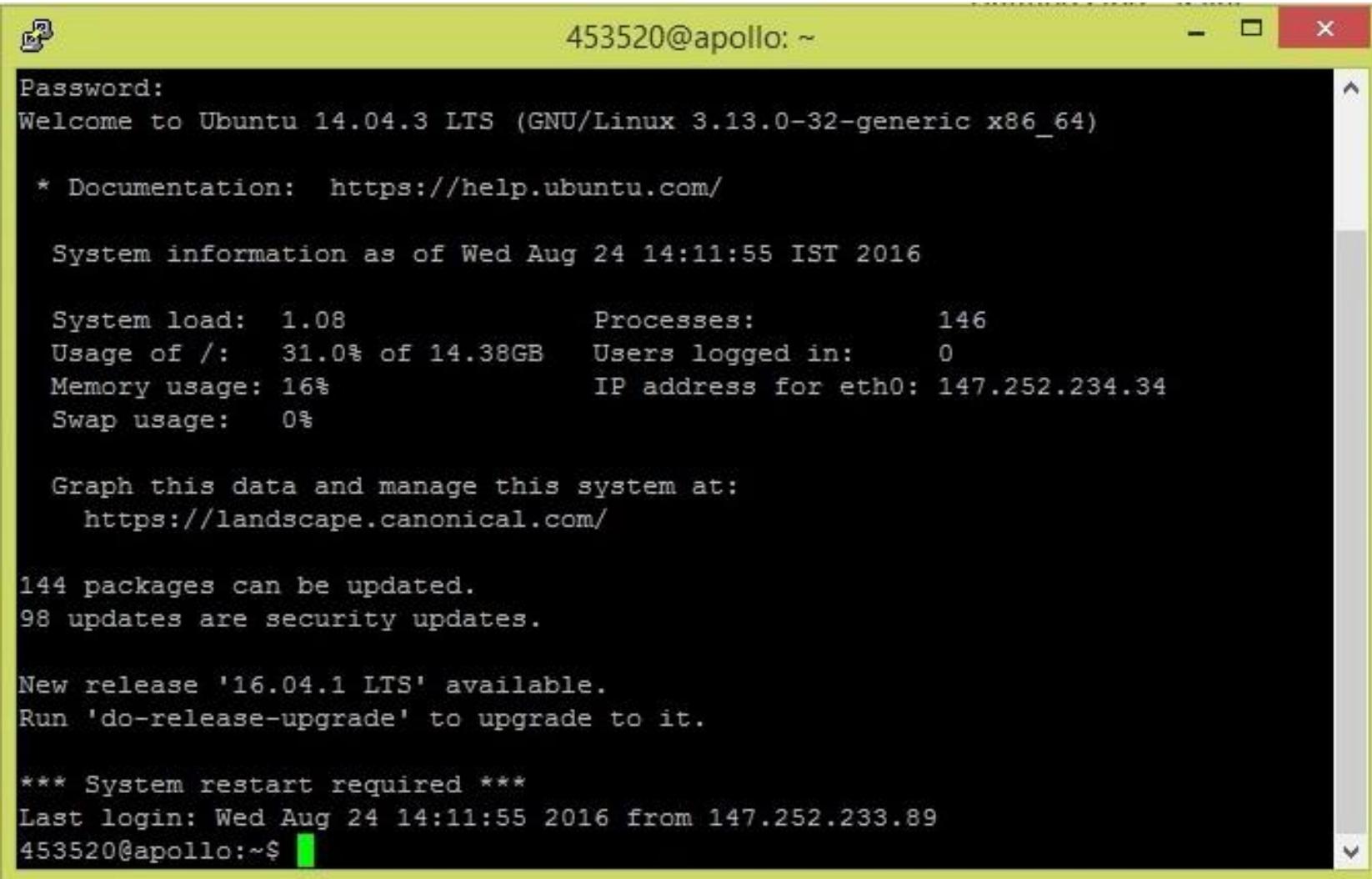
# Steps for Using SSH on windows to access Linux server

1. Open putty and enter the hostname 147.252.234.34
2. Enter user/password (its same user name and password that you normally use to login the lab computers)
3. Test few commands like date, cal

# Putty using SSH

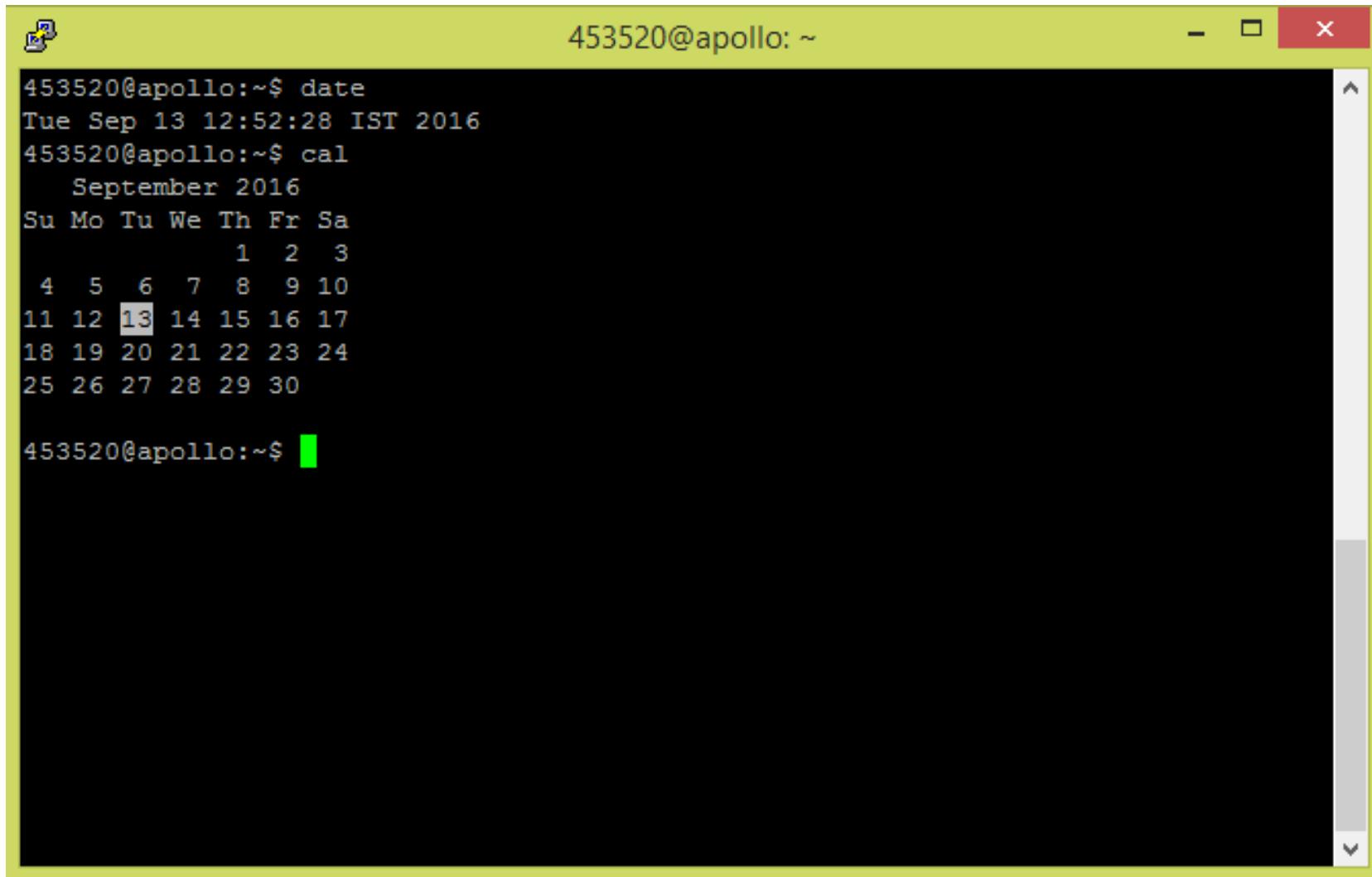


# Putty using SSH



A screenshot of a Putty terminal window titled "453520@apollo: ~". The window displays a Linux command-line interface. The session starts with a password prompt, followed by a welcome message for Ubuntu 14.04.3 LTS. It then shows system information as of Wednesday, August 24, 2016, at 14:11:55 IST. The system load is 1.08, processes are 146, and there are 0 users logged in. Memory usage is 16%, and the IP address for eth0 is 147.252.234.34. Swap usage is 0%. A link to a system management interface is provided: "Graph this data and manage this system at: https://landscape.canonical.com/". It also indicates that 144 packages can be updated, and 98 of those are security updates. A new release, '16.04.1 LTS', is available, and it suggests running 'do-release-upgrade' to upgrade to it. A warning message states "\*\*\* System restart required \*\*\*". The last login information is shown as "Last login: Wed Aug 24 14:11:55 2016 from 147.252.233.89". The prompt at the bottom is "453520@apollo:~\$".

# Putty using SSH



A screenshot of a Putty terminal window. The title bar reads "453520@apollo: ~". The window displays a Linux command-line session:

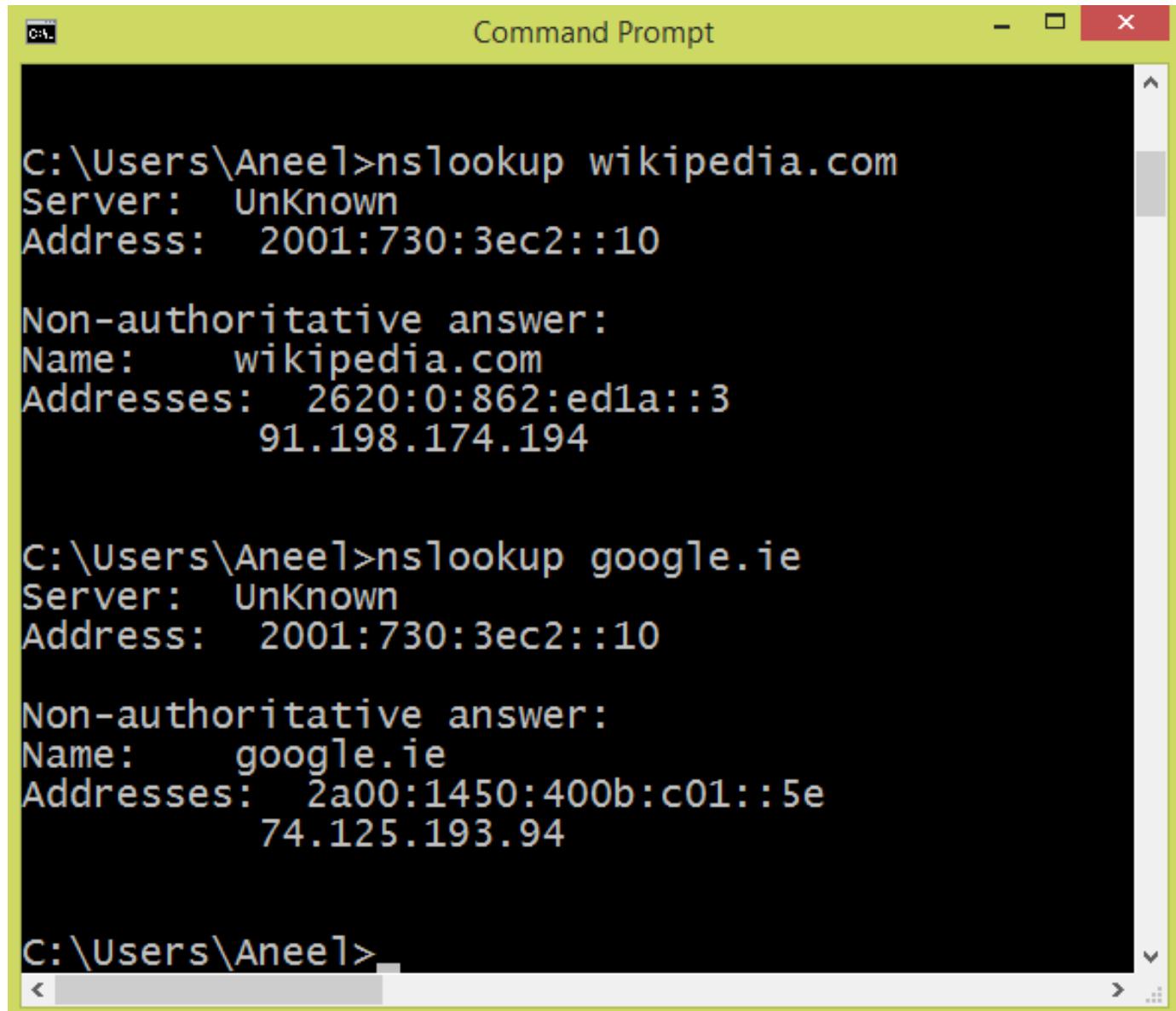
```
453520@apollo:~$ date
Tue Sep 13 12:52:28 IST 2016
453520@apollo:~$ cal
September 2016
Su Mo Tu We Th Fr Sa
        1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30

453520@apollo:~$
```

# DNS (Domain Name System)

- DNS is like a phone book
- It contains domain names and translate them into IP address.
- IP address of www.dit.ie is 147.252.25.70

# nslookup example



Command Prompt

```
C:\Users\Aneel>nslookup wikipedia.com
Server: UnKnown
Address: 2001:730:3ec2::10

Non-authoritative answer:
Name: wikipedia.com
Addresses: 2620:0:862:ed1a::3
           91.198.174.194

C:\Users\Aneel>nslookup google.ie
Server: UnKnown
Address: 2001:730:3ec2::10

Non-authoritative answer:
Name: google.ie
Addresses: 2a00:1450:400b:c01::5e
           74.125.193.94

C:\Users\Aneel>
```

# Http (Hypertext transfer protocol)

- TCP/IP based communication protocol,
- It is used to deliver data (HTML files, image files, query results, etc.)

on the World Wide Web.

- Port number 80.

# Writing Java Programs

1. Convert the URL to IP?
2. Find the IP of local Machine?
3. Convert IP to URL?
4. Find the address is IPv4 or IPv6?
5. Check the status of web server running or no?

# Writing Java Programs.....

6. Find the characteristics of IP?
7. Compare dit.ie and math.dit.ie has same IP?
8. Test TCP/IP application layer protocols for www.dit.ie ?
9. Download the web page from dit.ie?
10. Check the given IP is spam or not ?

# 1. Convert the URL to IP?

```
// It converts the URL string to IP  
import java.net.*;  
  
public class IP {  
    public static void main (String[] args) {  
        try {  
            InetAddress address = InetAddress.getByName(args[0]);  
            System.out.println(address);  
        }  
        catch (Exception ex) {  
            System.out.println(" Can't find the IP ");  
        }    }    }  
}
```

## 2. Find the IP of local Machine?

```
// It find the IP address of local machine  
import java.net.*;  
  
public class MyIP {  
    public static void main (String[] args) {  
        try {  
            InetAddress address = InetAddress.getLocalHost();  
            System.out.println(address);  
        }  
        catch (Exception ex) {  
            System.out.println("Could not find the address.");  
        }  
    }  
}
```

### 3. Convert IP to URL?

```
import java.net.*;

public class ReverseIP {

    public static void main (String[] args) {

        try {

            InetAddress ia = InetAddress.getByName("157.55.39.29");
            System.out.println(ia.getCanonicalHostName());
        }

        catch (Exception ex) {
            System.out.println(" Can't find the URL ");
        }
    }
}
```

## 4. Find IP4 and IP6?

```
// It find the address ip4 or ip6
import java.net.*;
public class IPTest {
    public static void main (String[] args) {
        try {
            InetAddress ia = InetAddress.getByName(args[0]);
            byte[] address = ia.getAddress();
            if (address.length == 4)
                System.out.println("IP4 Address");
            else if (address.length == 16)
                System.out.println("IP6 Address");
        }
        catch (Exception ex) {
            System.out.println(" Can't find the IP ");
        }
    }
}
```

# 5. Check Status of Web Server?

```
import java.net.*;
public class WebsiteTest {
    public static void main (String[] args) {
        try {
            InetAddress ia = InetAddress.getByName("www.java2s.com");
            boolean reachable = ia.isReachable(2000);
            if(reachable)
                System.out.println("Website Server is Online");
            else
                System.out.println("Website Server is Down...");}
        catch (Exception ex) {
            System.out.println(" Can't find the IP ");
        }
    }
}
```

# IP address Ranges

- Loopback IP address (127.0.0.0 to 127.255.255.255)
- Multicast IP address (224.0.0.0 to 239.255.255.255)
- Private IP address (192.168.0.0 to 192.168.255.255)  
(10.0.0.0 to 10.255.255.255)  
(172.16.0.0 to 172.31.255.255)

# 6. Find Characteristics of IP?

```
import java.net.*;

public class IPCharacteristics {
    public static void main(String[] args) {
        try {
            InetAddress address = InetAddress.getByName(args[0]);
            if (address.isSiteLocalAddress())
                System.out.println(address + " is a local network address.");
            if (address.isLoopbackAddress())
                System.out.println(address + " is loopback address.");
            if (address.isMulticastAddress())
                System.out.println(address + " is a multicast address.");
            else
                System.out.println(address + " is a unicast address.");
        }
        catch (UnknownHostException ex) {
            System.err.println("Could not resolve " + args[0]);
        }
    }
}
```

# 7. Compare IP of www.dit.ie and www.maths.dit.ie?

```
import java.net.*;

public class IPCompare {
    public static void main (String args[]) {
        try {
            InetAddress dit = InetAddress.getByName("www.dit.ie");
            InetAddress comp = InetAddress.getByName("www.maths.dit.ie");
            System.out.println(dit);
            System.out.println(comp);
            if (dit.equals(comp))
                System.out.println("www.dit.ie is the same IP as www.maths.dit.ie");
            else
                System.out.println("www.dit.ie is not same IP as www.maths.dit.ie");
        }
        catch (UnknownHostException ex) {
            System.out.println(ex);
        }
    }
}
```

## 8. Test TCP/IP application layer protocols for www.dit.ie ?

```
import java.net.*;
class TestProtocol {
    public static void main(String [] args ) {
        testProtocol("http://www.dit.ie");
        testProtocol("https://www.dit.ie");
        testProtocol("ftp://dit.ie");
        testProtocol("mailto:aneel.rahim@dit.ie");
    }
    public static void testProtocol(String url) {
        try {
            URL u = new URL(url);
            System.out.println(u.getProtocol() + " is supported");
        }
        catch (Exception ex) {
            System.out.println(url + " is not supported");
        }
    }
}
```

## 9. Download the web page from dit.ie?

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

class WebPage {
    public static void main(String [] args ) {
        try {
            URL u = new URL("http://www.dit.ie");
            InputStream in = u.openStream();
            int c;
            while ((c = in.read()) != -1)
                System.out.write(c);
            in.close();
        }
        catch (Exception ex) {
            System.err.println(ex);
        }
    }
}
```

# 10. IP Spam Checker

```
import java.net.*;

public class SpamCheck {

    public static void main(String[] args) throws Exception {
        String IP = "2.0.0.127"; // ip is 127.0.0.2 we reverse it
        String website = ".sbl.spamhaus.org"; //www.spamhaus.org/sbl/
        String query;

        try {
            query = IP + website;
            InetAddress.getByName(query);
            System.out.println( IP + " is a known spammer.");
        }

        catch (Exception e) {
            System.out.println(IP + " appears legitimate");
        }
    }
}
```

# Lecture 03

# Client Server Model

CMPU 3027  
Network Programming

# Client Server Model

- It allows two applications to establish communications and exchange data.
- This is called peer-to-peer communications.
- The applications can be on the same machine or different machines.
- One side starts execution and waits for the other side to contact it
- The other side then initiates the communication
- The application which waits for an incoming communication request is called a server.
- The application which initiates peer-to-peer communications is called a client.

# Client Server Model.....

- The server receives the client's request, performs the necessary computation and returns the result to the client.
- The server provides access to services or data on the server system
- While some services or data can be offered to anyone, some must be kept secure

# Security of Client Server Model

Servers must contain code that handles the issues of

- Authentication: Verify the Client identity.
- Authorization: Is client permitted to access the service. Student and staff are authorized different role for web course
- Data security: The protection of data from unauthorized access.

# Socket

- Socket is a connection between two hosts.
- Sockets allow the programmer to treat a network connection as just like reading and writing the data on files.
- Sockets hide the programmer from low-level details of the network, such as error detection, packet sizes.

# Socket Seven Operations

1. Connect to a remote machine
2. Send data
3. Receive data
4. Close a connection
5. Bind to a port
6. Listen for incoming data
7. Accept connections from remote machines on the bound port

# Standard Vs. Non-standard Server Software

- Standard application services are those defined by TCP/IP
- They are assigned well-known, widely recognised protocol port identifiers
- Examples are TELNET, FPT, SMTP
- Other services are called non-standard application services

# Types of Client Server

- Client-server software can be of two types
  - Connectionless
  - Connection-oriented
- These correspond to the two major transport protocols in TCP/IP
  - Clients using UDP are connectionless
  - Client using TCP are connection-oriented

# TCP

- Provides the level of reliability needed to communicate across an internet
- It checks that data arrives and automatically retransmits missing segments
- It computes a checksum over the data to ensure error-free transmission
- It uses sequence numbers to ensure data arrives in the right order and eliminates duplicate packets.

# UDP

- Provides Best Effort delivery only
- Does not guarantee delivery
- Client requests may be lost, delayed, duplicated or delivered out of order
- Server response may be lost, delayed, duplicated or delivered out of order

# Steps for Creating a Simple Server

- Step 1: Create a ServerSocket

```
ServerSocket server = new ServerSocket( portNumber, queueLength );
```

- Step 2: Wait for a Connection

```
Socket connection = server.accept();
```

# Steps for Creating a Simple Server...

- Step 3: Get the Socket's I/O Streams
  - Get the OutputStream and InputStream objects that enable the server to communicate with the client by sending and receiving bytes.

```
ObjectInputStream input = new ObjectInputStream( connection.getInputStream() );
```

```
ObjectOutputStream output = new ObjectOutputStream( connection.getOutputStream() );
```

# Steps for Creating a Simple Server...

- Step 4: Perform the Processing
  - The server and the client communicate via the OutputStream and InputStream objects.
- Step 5: Close the Connection
  - When the transmission is complete, the server closes the connection

# Steps for Creating a Simple Client...

- Step 1: Create a Socket to Connect to the Server

```
Socket connection = new Socket( serverAddress, port );
```

- Step 2: Get the Socket's I/O Streams

- Get the OutputStream and InputStream objects that enable the client to communicate with the server by sending and receiving bytes.

# Steps for Creating a Simple Client

- Step 3: Perform the Processing
  - The server and the client communicate via the OutputStream and InputStream objects
- Step 4: : Close the Connection
  - When the transmission is complete, the client closes the connection

# Writing Java Programs

1. Construct a Date by talking to time.nist.gov.
2. Constructing sockets without connecting.
3. Store Socket Address.
4. Write a program to get socket information
5. Write Network Based English to Irish translator?

# Writing Java Programs

6. Simple client to connect to a whois server.
7. Write port scanner program for TCP ports.
8. Write port scanner program for UDP ports.
9. Write a simple PoxyClient.
10. Write a program to adjust IP traffic for VOIP applications.

# 1. Construct a Date by talking to time.nist.gov

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

public class DayTimeClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("time.nist.gov", 13);
            socket.setSoTimeout(15000);
            InputStream in = socket.getInputStream();
            int c;
            while ((c = in.read()) != -1)
                System.out.print((char)c);
            socket.close();
        }
        catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

## 2. Constructing sockets without connecting

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

public class SocketAdd {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket();
            SocketAddress address = new InetSocketAddress("time.nist.gov", 13);
            socket.connect(address);
            InputStream in = socket.getInputStream();
            int c;
            while ((c = in.read()) != -1)
                System.out.print((char)c);
            socket.close();
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

### 3. Store Socket address

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

public class SocketStore {

    public static void main(String[] args) {
        try {
            Socket socket = new Socket("www.yahoo.com", 80);
            System.out.println("Successfully connected to yahoo");
            System.out.println("Saving the Socket Address");
            SocketAddress yahoo = socket.getRemoteSocketAddress();
            System.out.println("Closing the Socket");
            socket.close();
            Socket socket2 = new Socket();
            socket2.connect(yahoo);
            System.out.println("Again connected to yahoo using socket Address");
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

## 4. Program to print Socket Information

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

public class SocketInfo {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("www.dit.ie", 80);
            System.out.println("Connected to " +
                socket.getInetAddress() + " on port " + socket.getPort());
            System.out.println("From " + socket.getLocalAddress() + " on
port " + socket.getLocalPort());
        }
        catch (Exception ex) {
            System.err.println(ex);
        } } }
```

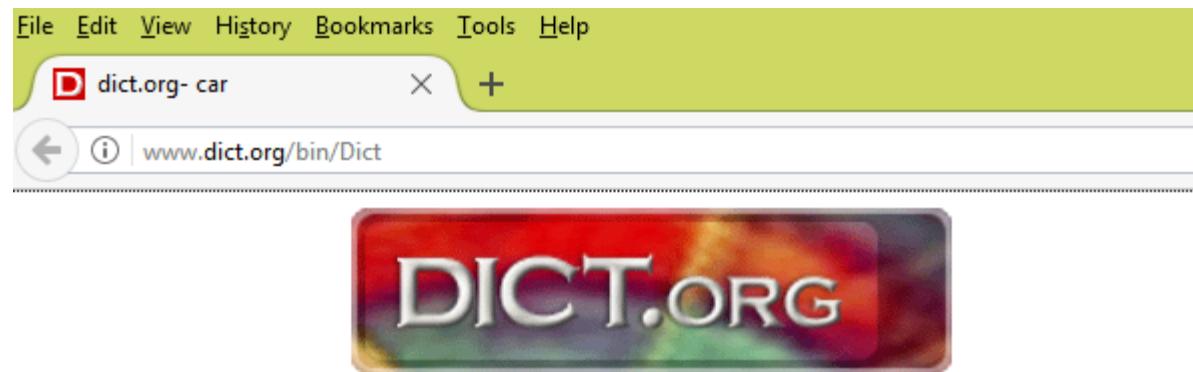
# 5. A network-based English-to-Irish Translator

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

public class DictClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("dict.org", 2628);
            socket.setSoTimeout(15000);
            OutputStream out = socket.getOutputStream();
            Writer writer = new OutputStreamWriter(out, "UTF-8");
            InputStream in = socket.getInputStream();
            Reader reader = new InputStreamReader(in, "UTF-8");
            writer.write("define fd-eng-iri " + args[0] + "\r\n");
            writer.flush();
        }
    }
}
```

## 5. A network-based English-to-Irish Translator

```
int c;  
  
while((c= reader.read()) != -1)  
    System.out.print((char)c);  
  
writer.write("quit\r\n");  
writer.flush();  
socket.close();  
  
}  
  
catch (Exception ex) {  
    System.err.println(ex);  
}  } }
```



Search for:

Search type:

Database:

[Database copyright information](#)

[Server information](#)

[Wiki: Resources, links, and other information](#)

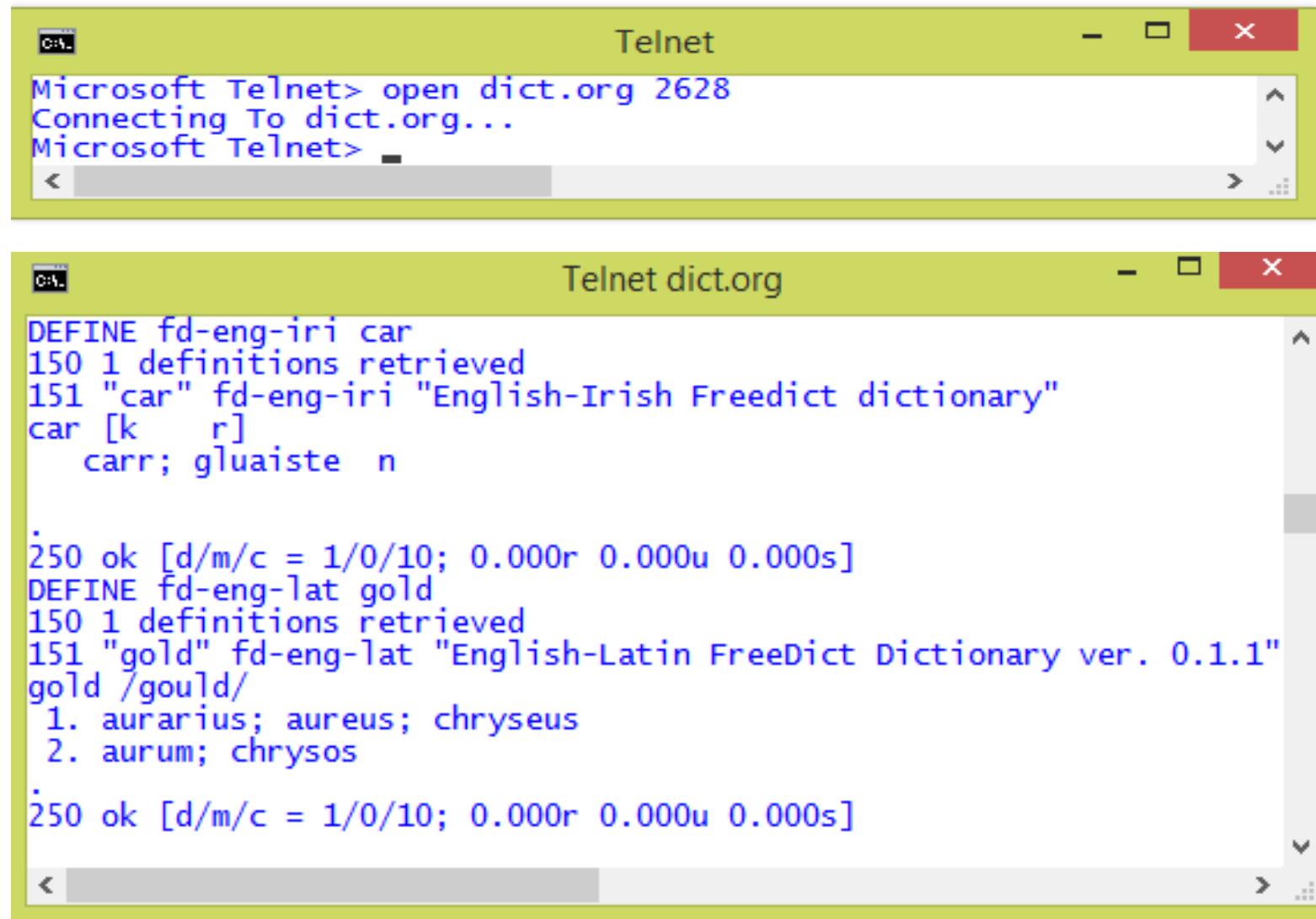
---

1 definition found for car

From English-Irish Freedict dictionary:

car [ka:r]  
carr; gluaisteún

# Telnet Client in Windows...



The image shows two separate windows, both titled "Telnet". The top window is titled "Telnet" and shows the command "open dict.org 2628" being entered, followed by the message "Connecting To dict.org...". The bottom window is titled "Telnet dict.org" and displays the results of a search for the word "car" and "gold" in various dictionaries. The output includes definitions from the English-Irish Freedict dictionary and the English-Latin FreeDict Dictionary.

```
Microsoft Telnet> open dict.org 2628
Connecting To dict.org...
Microsoft Telnet>

Telnet dict.org
DEFINE fd-eng-iri car
150 1 definitions retrieved
151 "car" fd-eng-iri "English-Irish Freedict dictionary"
car [k r]
    carr; gluaiste n

.
250 ok [d/m/c = 1/0/10; 0.000r 0.000u 0.000s]
DEFINE fd-eng-lat gold
150 1 definitions retrieved
151 "gold" fd-eng-lat "English-Latin FreeDict Dictionary ver. 0.1.1"
gold /gould/
    1. aurarius; aureus; chryseus
    2. aurum; chrysos
.
250 ok [d/m/c = 1/0/10; 0.000r 0.000u 0.000s]
```

```
220 pan.alephnull.com dictd 1.12.1/rf on Linux 4.4.0-1-amd64 <auth.mime> <62713521.25371.1506159074@pan.alephnull.com>
show DB
110 72 databases present
gcide "The Collaborative International Dictionary of English v.0.48"
wn "WordNet (r) 3.0 (2006)"
moby-thesaurus "Moby Thesaurus II by Grady Ward, 1.0"
elements "The Elements (07Nov00)"
vera "V.E.R.A. -- Virtual Entity of Relevant Acronyms (September 2014)"
jargon "The Jargon File (version 4.4.7, 29 Dec 2003)"
foldoc "The Free On-line Dictionary of Computing (18 March 2015)"
easton "Easton's 1897 Bible Dictionary"
hitchcock "Hitchcock's Bible Names Dictionary (late 1800's)"
bouvier "Bouvier's Law Dictionary, Revised 6th Ed (1856)"
devil "The Devil's Dictionary (1881-1906)"
world02 "CIA World Factbook 2002"
gaz2k-counties "U.S. Gazetteer Counties (2000)"
gaz2k-places "U.S. Gazetteer Places (2000)"
gaz2k-zips "U.S. Gazetteer Zip Code Tabulation Areas (2000)"
fd-tur-eng "Turkish-English FreeDict Dictionary ver. 0.2.1"
fd-por-deu "Portuguese-German FreeDict Dictionary ver. 0.1.1"
fd-nld-eng "Dutch-English Freedict Dictionary ver. 0.1.3"
fd-eng-ara "English-Arabic FreeDict Dictionary ver. 0.6.2"
fd-spa-eng "Spanish-English FreeDict Dictionary ver. 0.1.1"
fd-eng-hun "English-Hungarian FreeDict Dictionary ver. 0.1"
fd-ita-eng "Italian-English FreeDict Dictionary ver. 0.1.1"
fd-wel-eng "Welsh-English Freedict dictionary"
fd-eng-nld "English-Dutch FreeDict Dictionary ver. 0.1.1"
fd-fra-eng "French-English FreeDict Dictionary ver. 0.3.4"
fd-tur-deu "Turkish-German FreeDict Dictionary ver. 0.1.1"
fd-swe-eng "Swedish-English FreeDict Dictionary ver. 0.1.1"
fd-nld-fra "Nederlands-French FreeDict Dictionary ver. 0.1.1"
fd-eng-swa "English-Swahili xFried/FreeDict Dictionary"
fd-deu-nld "German-Dutch FreeDict Dictionary ver. 0.1.1"
fd-fra-deu "French-German FreeDict Dictionary ver. 0.1.1"
fd-eng-cro "English-Croatian Freedict Dictionary"
fd-eng-ita "English-Italian FreeDict Dictionary ver. 0.1.1"
fd-eng-lat "English-Latin FreeDict Dictionary ver. 0.1.1"
fd-lat-eng "Latin-English FreeDict Dictionary ver. 0.1.1"
fd-fra-nld "French-Dutch FreeDict Dictionary ver. 0.1.2"
fd-ita-deu "Italian-German FreeDict Dictionary ver. 0.1.1"
fd-eng-hin "English-Hindi FreeDict Dictionary ver. 1.5.1"
fd-deu-eng "German-English FreeDict Dictionary ver. 0.3.4"
fd-por-eng "Portuguese-English FreeDict Dictionary ver. 0.1.1"
fd-lat-deu "Latin - German FreeDict dictionary ver. 0.4"
fd-jpn-deu "Japanese-German FreeDict Dictionary ver. 0.1.1"
fd-eng-deu "English-German FreeDict Dictionary ver. 0.3.6"
fd-eng-scr "English-Serbo-Croat Freedict dictionary"
```

# 6. Simple client to connect to a whois server

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

public class Whois {
    public static void main(String[] args) {
        try {
            int c;
            Socket s = new Socket("whois.internic.net", 43);
            InputStream in = s.getInputStream();
            OutputStream out = s.getOutputStream();
            String str = "google.com"+ "\n";
            out.write(str.getBytes());
            while ((c = in.read()) != -1)
                System.out.print((char)c);
            s.close();
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

# Using whois in Windows

https://docs.microsoft.com/en-us/sysinternals/downloads/whois

Microsoft Technologies Documentation Resources

Sysinternals Learn Downloads Community

Home / Downloads

Filter

- Insight for Active Directory
- AdRestore
- PipeList
- PsFile
- PsPing
- ShareEnum
- TCPView
- Whois**
- > Process Utilities
- > Security Utilities
- > System Information

## Whois v1.14

07/04/2016 • 1 minutes to read • Contributors 

**By Mark Russinovich**

Published: July 4, 2016

 [Download Whois \(158 KB\)](#)

## Introduction

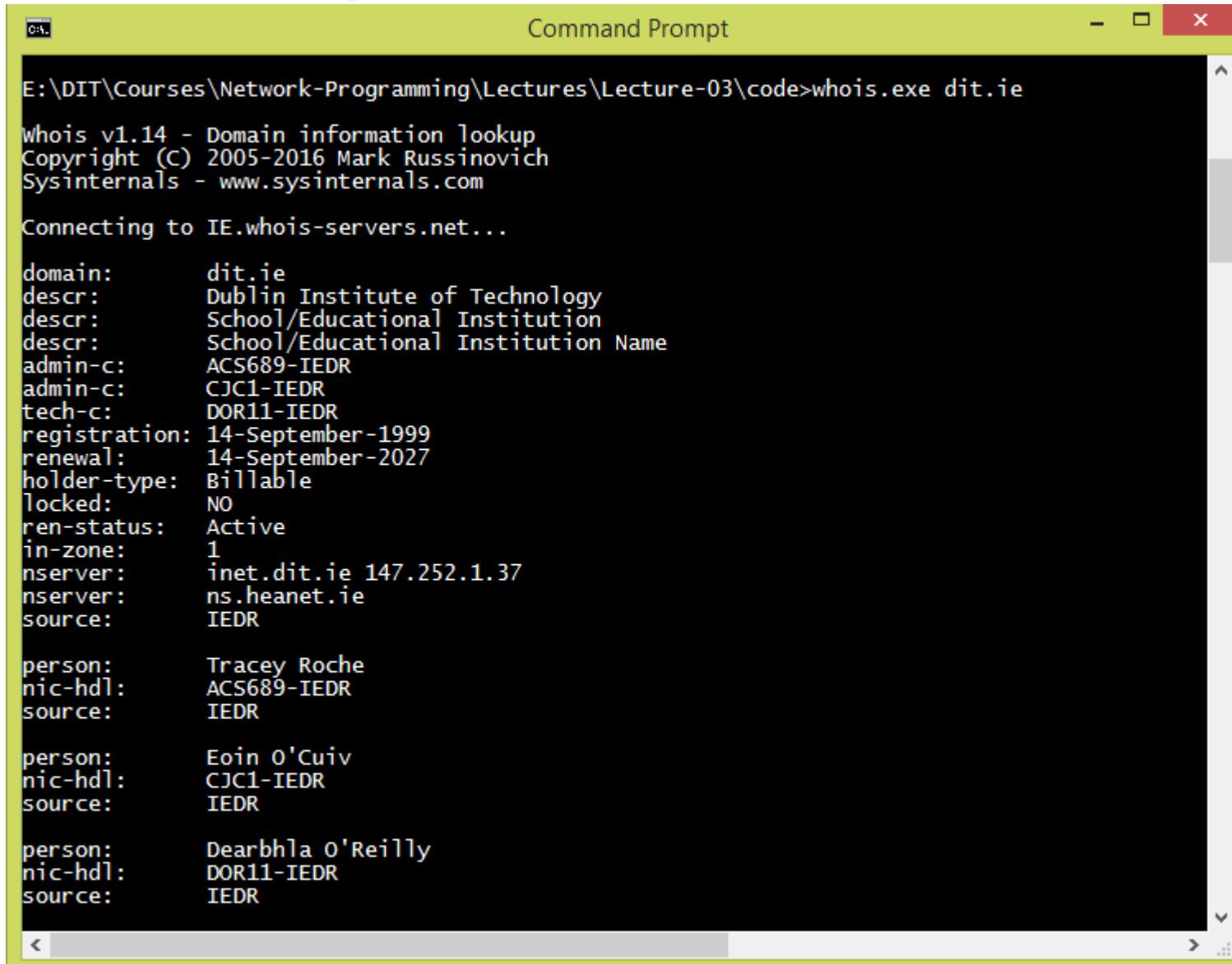
Whois performs the registration record for the domain name or IP address that you specify.

## Usage

**Usage: whois [-v] domainname [whois.server]**

Download PDF

# Using whois in Windows



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The window has a yellow header bar with the title and standard window controls. The main area of the window displays the output of the "whois" command for the domain "dit.ie". The output is as follows:

```
E:\DIT\Courses\Network-Programming\Lectures\Lecture-03\code>whois.exe dit.ie
whois v1.14 - Domain information lookup
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Connecting to IE.whois-servers.net...

domain: dit.ie
descr: Dublin Institute of Technology
descr: School/Educational Institution
descr: School/Educational Institution Name
admin-c: ACS689-IEDR
admin-c: CJC1-IEDR
tech-c: DOR11-IEDR
registration: 14-September-1999
renewal: 14-September-2027
holder-type: Billable
locked: NO
ren-status: Active
in-zone: 1
nserver: inet.dit.ie 147.252.1.37
nserver: ns.heanet.ie
source: IEDR

person: Tracey Roche
nic-hdl: ACS689-IEDR
source: IEDR

person: Eoin O'Cuiv
nic-hdl: CJC1-IEDR
source: IEDR

person: Dearbhla O'Reilly
nic-hdl: DOR11-IEDR
source: IEDR
```

## 7. TCP Port Scanner

```
import java.net.*;
public class TCPPortScanner {
    public static void main(String args[]) {
        for(int port=1; port<=65535; port++) {
            try {
                Socket socket = new Socket("127.0.0.1",port);
                System.out.println("Port in use: " + port );
                socket.close();
            }
            catch (Exception e) {
                System.out.println("Port not in use: " + port );
            }
        }
    }
}
```

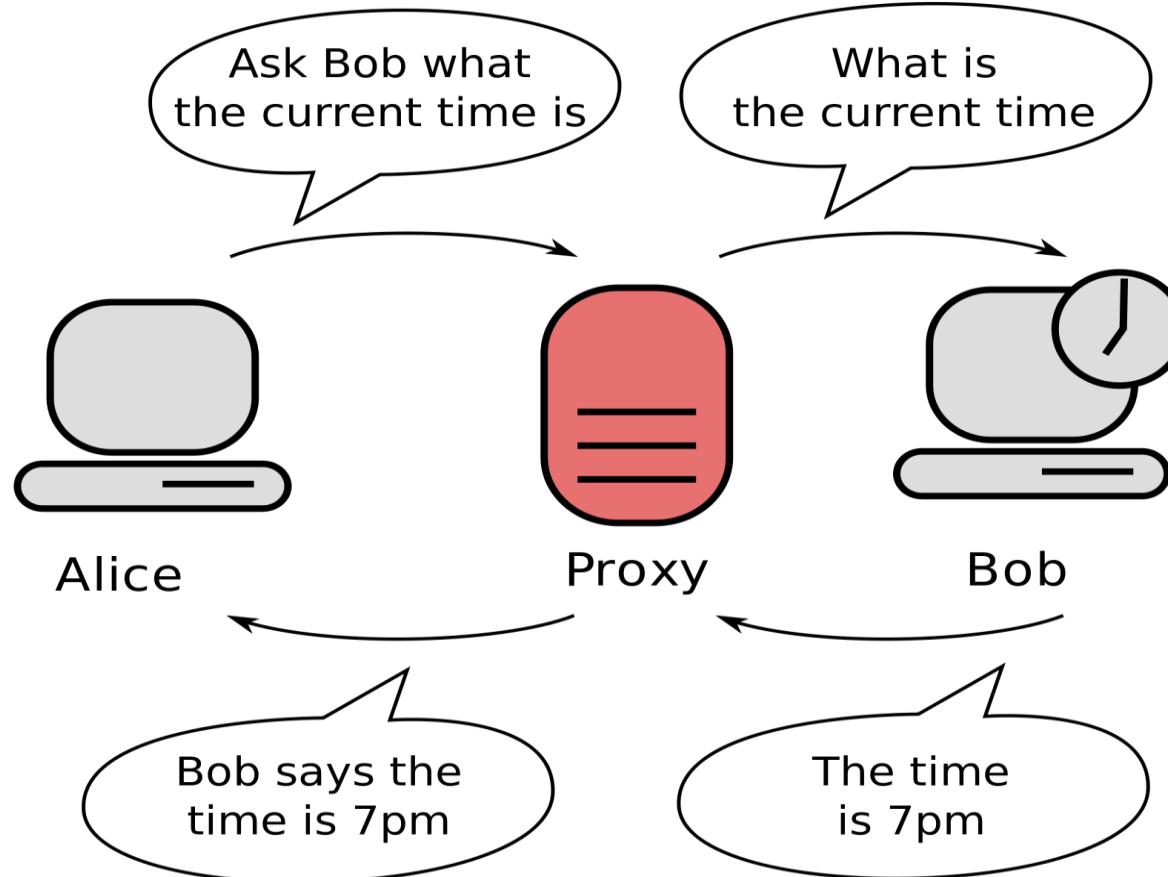
## 8. UDP Port Scanner

```
import java.net.*;
public class UDPPortScanner {
    public static void main(String[] args) {
        for (int port = 1; port <= 5000; port++) {
            try {
                // the next line will fail and drop into the catch block if
                // there is already a server running on port i
                DatagramSocket server = new DatagramSocket(port);
                server.close();
            }
            catch (SocketException ex) {
                System.out.println("There is a server on port " + port + ".");
            }
        }
    }
}
```

# Proxy server

- A proxy server is a server that sits between a client application, such as a Web browser, and a real server.
- A client connects to the proxy server, requesting some service, such as a file or web page available from a different server and the proxy server fulfil the request.
- The proxy server can be used to protect the client privacy.

# Proxy server



## 9. Write a simple PoxyClient.

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

public class ProxyClient {

    public static void main(String[] args) {

        try {

            SocketAddress proxyAddress = new
            InetSocketAddress("119.23.204.52",1080); // setup proxy

            Proxy proxy = new Proxy(Proxy.Type.SOCKS, proxyAddress);

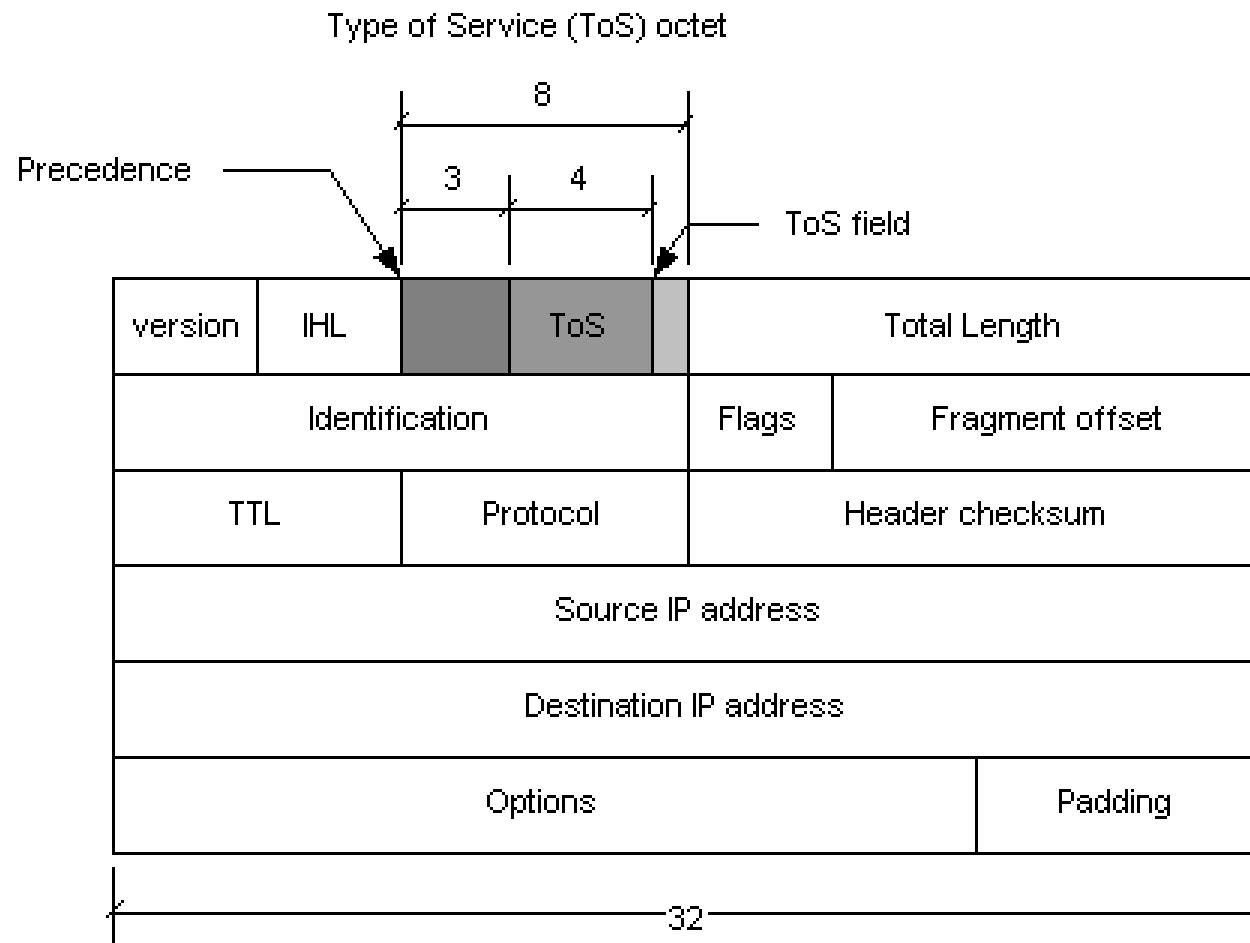
            Socket s = new Socket(proxy); // setup socket

            SocketAddress remote = new InetSocketAddress("www.google.ie", 8080);
            s.connect(remote); //connect

            System.out.println("Successfully connected using Proxy Server");}

        catch (Exception ex) {
            System.err.println(ex);
        }
    }
}
```

# IPv4 Header



# Type of service (TOS) in RFC 791

September 1981

Internet Protocol  
Specification

Type of Service: 8 bits

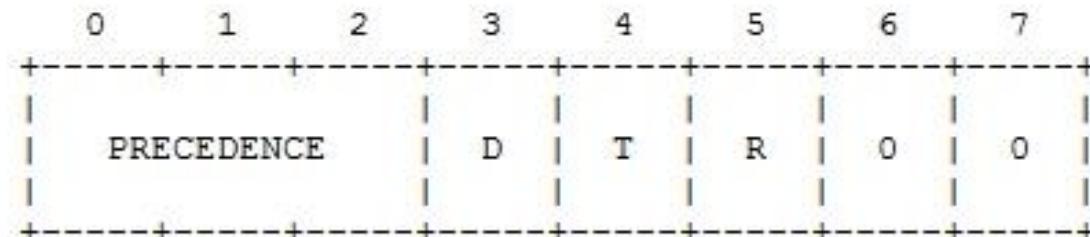
Bits 0-2: Precedence.

Bit 3: 0 = Normal Delay, 1 = Low Delay.

Bits 4: 0 = Normal Throughput, 1 = High Throughput.

Bits 5: 0 = Normal Reliability, 1 = High Reliability.

Bit 6-7: Reserved for Future Use.



## 10. Write a program to adjust IP traffic for VOIP applications.

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

public class IPTraffic {
    public static void main(String[] args) {
        int IPTOS_LOWCOST = 2;
        int IPTOS_RELIABILITY = 4;
        int IPTOS_THROUGHPUT = 8;
        int IPTOS_LOWDELAY = 16;
        try {
            Socket s1 = new Socket("www.yahoo.com", 80);
            s1.setTrafficClass(IPTOS_LOWCOST);
            Socket s2 = new Socket("www.yahoo.com", 80);
            s2.setTrafficClass(IPTOS_RELIABILITY);
            Socket s3 = new Socket("www.yahoo.com", 80);
            s3.setTrafficClass(IPTOS_THROUGHPUT | IPTOS_LOWDELAY);
            System.out.println(s3.getTrafficClass());
        } catch (Exception ex) {
            System.err.println(ex);
        }
    }
}
```

# Lecture 04

# Server Side Programming

CMPU 3027  
Network Programming

# Four Basic types of Servers

<b>iterative connectionless</b>	<b>iterative connection-oriented</b>
<b>concurrent connectionless</b>	<b>concurrent connection-oriented</b>

# Iterative Servers

- Servers which handle one request at a time are called Iterative servers
- Iterative servers are easier to build and understand.
- They may have poor performance.

# Concurrent Servers

- Servers which handle more than one request at a time are called Concurrent servers
- Concurrent servers can deliver better performance than Iterative servers
- They require more programming effort as compare to Iterative servers

# Connection-Oriented Servers

- A connection-oriented server uses the TCP transport protocol
- TCP handles errors, packet loss, and out-of-order delivery automatically for the application
- When the server connects to a client, TCP will deliver and receive all requests or responses or else inform the server that the connection has broken
- This simplifies the programming effort

# Connection less Servers

- A connection-less server uses the UDP transport protocol.
- Client or server must take responsibility for reliable delivery.
- UDP facilitates the multicast and broadcast communication

# Writing Java Programs

1. Modified the IP Spam checker program?
2. Write Simple TCP Client Server Program?
3. Write Echo Client and Server program?
4. Write TCP Daytime Client and Server?
5. Write Simple UDP Client Server Program?

# Writing Java Programs

6. Write Ftp Client program?
7. Write Smtplib Client Program?
8. Modify Smtplib Client Program?
9. Write TCP Daytime Client Program using apache common net?
10. Write UDP Daytime Client Program using apache common net?

# IP Blacklist

- <https://www.spamhaus.org/lookup/>
- Check below IP online and also with java program
  - 127.0.0.2
  - 91.205.40.0

<https://www.spamhaus.org/lookup/>



Home	SBL	XBL	PBL	DBL	DROP	ROKSO			
Blocklist Removal Center					About Spamhaus   FAQs   News Blog				

## Blocklist Removal Center

### IP Address Lookup

This Lookup tool is **only** for IP Addresses - do not enter domains or email addresses. If you do not know what an IP address is, or what IP to look up, please contact your Internet Service Provider and ask them to help you.

**IP Address Lookup Tool.** This lookup tool checks to see if the **IP Address** you enter is currently listed in the live Spamhaus IP blocklists: **SBL, XBL and PBL**.

Enter an **IP Address**

### Associated Documents

- ▶ How Blocklists Work
- ▶ What is an "IP Address"?

If your IP address is listed on one of our IP blocklists; SBL, XBL or PBL (collectively known as the 'Zen' blocklist), this lookup tool will tell you which one and will give you a link to information on what to do.

<https://www.spamhaus.org/lookup/>

The screenshot shows the official website for The SPAMHAUS Project. At the top, there's a navigation bar with links for Home, SBL, XBL, PBL, DBL, DROP, and ROKSO. Below the navigation is a yellow banner with the text "Blocklist Removal Center" on the left and "About Spamhaus | FAQs | News Blog" on the right, along with a small RSS feed icon. The main content area features a section titled "Blocklist Removal Center". On the left, a sidebar titled "Blocklist Documents" lists links to SBL, XBL, PBL, DBL FAQs, and instructions on how blocklists work. The main content area displays search results for the IP address 127.0.0.2 across three different blocklists: SBL, PBL, and XBL.

## Blocklist Removal Center

### Blocklist Documents

- ▶ SBL FAQs
- ▶ XBL FAQs
- ▶ PBL FAQs
- ▶ DBL FAQs
- ▶ How Blocklists Work
- ▶ Lookup another address

### Blocklist Lookup Results

**127.0.0.2 is listed in the SBL**, in the following records:

- [SBL2](#)

**127.0.0.2 is listed in the PBL**, in the following records:

- [PBL000003](#)

**127.0.0.2 is listed in the XBL**, because it appears in:

- [CBL](#)

This lookup tool is for manual (non-automated) lookups only. Any perceived use of automated tools to access this web lookup system will result in firewalls or other countermeasures.

# 1. Modified the IP Spam checker program

```
import java.net.*;

public class SpamCheck {

    public static void main(String[] args) {
        String IP = args[0];// User Input IP, we will Reverse it
        String website = "sbl.spamhaus.org"; //www.spamhaus.org/sbl/
        try {
            InetAddress address = InetAddress.getByName(IP);
            byte[] quad = address.getAddress();
            String query = website;
            for (byte octet : quad) {
                int unsignedByte = octet < 0 ? octet + 256 : octet;
                query = unsignedByte + "." + query;
            }
            System.out.println(query);
            InetAddress.getByName(query);
            System.out.println(IP + " is a known spammer.");
        } catch (Exception e) {
            System.out.println(IP + " appears legitimate");
        }
    }
}
```

## 2. Simple TCP Server

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

public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket server = new ServerSocket( 5000, 10 ); // create ServerSocket
            System.out.println("Server is Running on port 5000");
            Socket connection = server.accept();
            ObjectOutputStream out = new ObjectOutputStream( connection.getOutputStream() );
            ObjectInputStream in = new ObjectInputStream( connection.getInputStream() );
            String message = "Connection successful";
            out.writeObject(message);
            message = ( String ) in.readObject();
            System.out.println ("Data from Client: " + message);
            connection.close();
        } catch (Exception ex) {
            System.err.println(ex);
        }
    }
}
```

## 2. Simple TCP Client

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

public class Client {
    public static void main(String[] args) {
        try {
            Socket client = new Socket("localhost", 5000);
            System.out.println("Client is connected to Server");
            ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());
            ObjectInputStream in = new ObjectInputStream(client.getInputStream());
            String message = (String) in.readObject();
            System.out.println("Data from Server: " + message);
            message = "Hello";
            out.writeObject(message);
            client.close(); }
        catch (Exception ex) {
            System.err.println(ex);
        } } }
```

### 3. Echo Server

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

public class EchoServer {

    public static void main(String[] args) {

        try {

            ServerSocket server = new ServerSocket( 5000, 10 ); // create ServerSocket
            System.out.println("Server is Running on port 5000");
            Socket connection = server.accept();
            ObjectOutputStream out = new ObjectOutputStream( connection.getOutputStream() );
            ObjectInputStream in = new ObjectInputStream( connection.getInputStream() );
            String message = "Connection successful";

            while (true) {
                message = ( String ) in.readObject();
                System.out.println ("Data from Client: " + message);
                out.writeObject(message); }

        catch (Exception ex) {
            System.err.println(ex);
        }
    }
}
```

### 3. Echo Client

```
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class EchoClient {

    public static void main(String[] args) {

        try {

            Socket client = new Socket( "localhost",5000 );

            System.out.println("Client is connected to Server");

            ObjectOutputStream out = new ObjectOutputStream( client.getOutputStream() );
            ObjectInputStream in = new ObjectInputStream( client.getInputStream() );
            Scanner input = new Scanner( System.in );

            while (true) {

                System.out.println("Enter your Message");

                String message = input.nextLine();
                out.writeObject(message);

                message = ( String ) in.readObject();

                System.out.println ("Data from Server: " + message);}}
```

**catch** (Exception ex) {  
    System.err.println(ex);  
}

## 4. DayTimeServer

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
    public static void main(String[] args) {
        try {
            ServerSocket server = new ServerSocket(4000);
            Socket connection = server.accept();
            ObjectOutputStream out = new
ObjectOutputStream(connection.getOutputStream());
            Date now = new Date();
            out.writeObject(now);
            connection.close();
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

## 4. DayTimeClient

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

public class DayTimeClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 4000);
            socket.setSoTimeout(15000);
            ObjectInputStream in = new
            ObjectInputStream(socket.getInputStream());
            System.out.println(in.readObject());
            socket.close();
        }
        catch (Exception ex) {
            System.err.println(ex);
        }
    }
}
```

# 5. UDPServer

```
import java.net.*;
public class UDPServer {
    public static void main(String[] args) {
        try {
            DatagramSocket server = new DatagramSocket(5000);
            byte data[] = new byte[ 100 ];
            DatagramPacket receivePacket = new DatagramPacket( data, data.length );
            System.out.println("Server is Running");
            server.receive(receivePacket);
            System.out.println(new String (receivePacket.getData()));
            server.close();
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

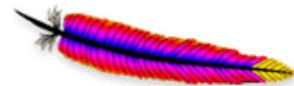
## 5. UDPClient

```
import java.net.*;
public class UDPClient {
    public static void main(String[] args) {
        try {
            DatagramSocket client = new DatagramSocket();
            String message = "Hello Server";
            byte data [] = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket( data,
                data.length, InetAddress.getLocalHost(), 5000);
            System.out.println("Sending Data to Server");
            client.send(sendPacket);
            client.close();
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

# Apache Commons Net

- Library provide support for client side programming for Internet protocols.
- Download it [https://commons.apache.org/proper/commons-net/download\\_net.cgi](https://commons.apache.org/proper/commons-net/download_net.cgi)  
or from the Brightspace.
- Add **commons-net-3.5.jar** to the classpath.

<https://commons.apache.org/proper/commons-net/>



**Apache Commons™**  
http://commons.apache.org/

commons  
net™

Apache Commons Net™ Last Published: 01 May 2016 | Version: 3.5 ApacheCon Apache Commons

**DOCUMENTATION**

**Overview**

Migration How-to  
FAQ  
**Download**

Javadoc 3.5 (Java 1.6)  
Javadoc 1.4.1 (Java 1.3)

**DEVELOPMENT**

Coding Specifications  
Mailing lists  
Issue Tracking  
SVN repository

**PROJECT DOCUMENTATION**

Project Information

**About**

Summary  
Team  
Source Code Management  
Issue Management  
Mailing Lists  
Dependency Information

## Apache Commons Net

Apache Commons Net™ library implements the client side of many basic Internet protocols. The purpose of the library is to provide fundamental protocol access, not higher-level abstractions. Therefore, some of the design violates object-oriented design principles. Our philosophy is to make the global functionality of a protocol accessible (e.g., TFTP send file and receive file) when possible, but also provide access to the fundamental protocols where applicable so that the programmer may construct his own custom implementations (e.g, the TFTP packet classes and the TFTP packet send and receive methods are exposed).

## Features

Supported protocols include:

- FTP/FTPS
- FTP over HTTP (experimental)
- NNTP
- SMTP(S)
- POP3(S)
- IMAP(S)
- Telnet
- TFTP
- Finger
- Whois
- rexec/rcmd/rlogin
- Time (rdate) and Daytime
- Echo
- Discard

## 6. Ftp Client

```
import java.net.*;
import java.io.*;
import org.apache.commons.net.ftp.FTPClient;
class FtpClient{

    public static void main(String[] args) {
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect("ftp.ietf.org");
            if (ftpClient.login("anonymous", ""))
                System.out.println("FTP Successfully logged in!");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return;
    }

    String fileName = "rfc0978.txt";
    FileOutputStream fos = new FileOutputStream(fileName);
}
```

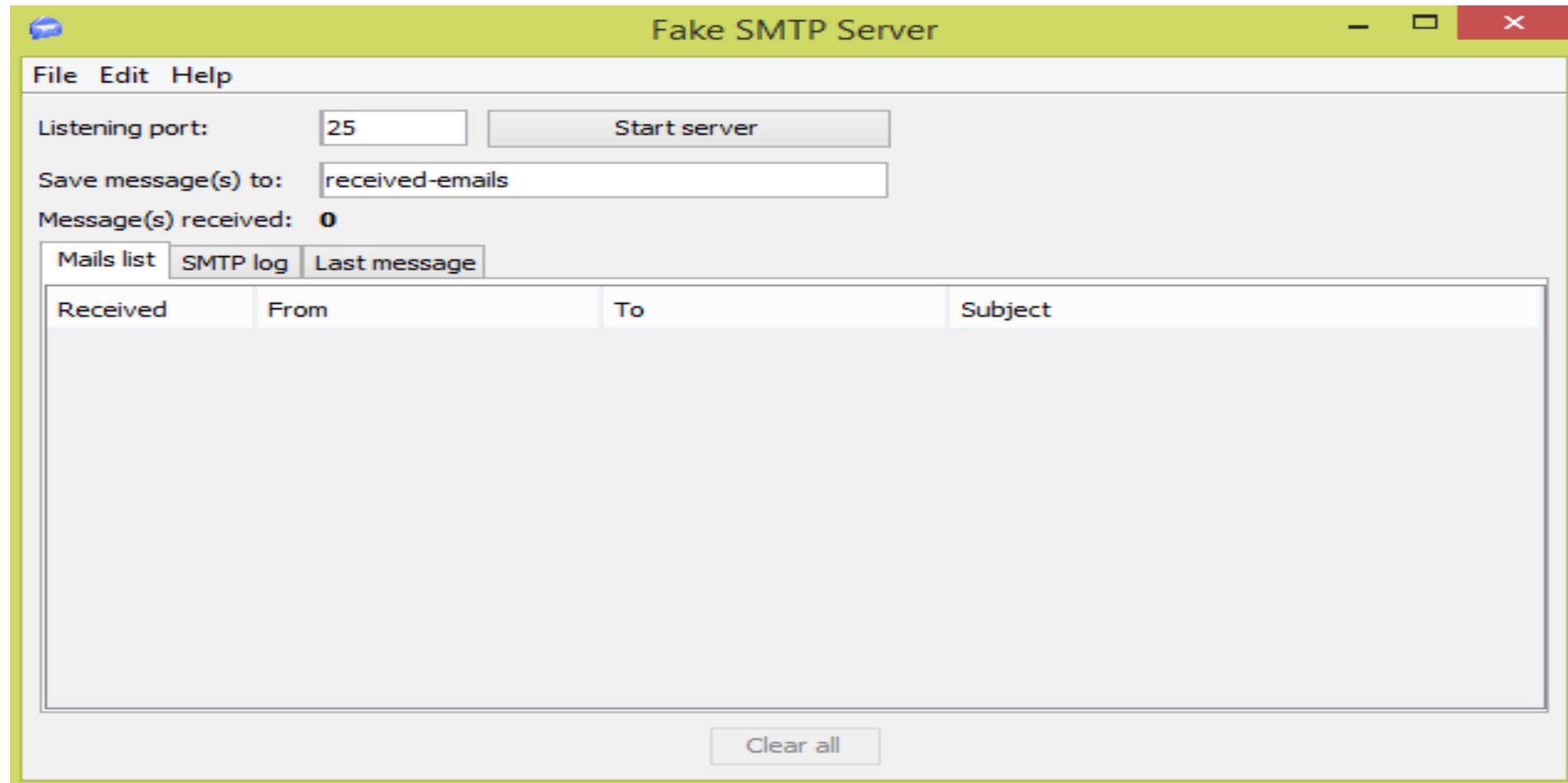
# 6. Ftp Client

```
System.out.println("FTP Changing Working Directory");
ftpClient.changeWorkingDirectory("/rfc");
if (ftpClient.retrieveFile(fileName, fos))
    System.out.println("File downloaded successfully !");
else
    System.out.println("File downloading failed !");
ftpClient.logout();
}
catch (Exception e) {
    System.out.println(e);
}
}
```

# FakeSMTP

- It is a Free SMTP Server with GUI for testing emails in applications easily.
- It is written in Java.
- Download it <https://nilhcem.github.io/FakeSMTP/download.html> or from the Brightspace.

# FakeSMTP



## 7. SMTP Client

```
import java.net.*;
import java.io.*;
import org.apache.commons.net.smtp.SMTPClient;
class SmtpClient{
    public static void main(String[] args) {
        try {
            SMTPClient client = new SMTPClient();
            client.connect("127.0.0.1");
            client.sendSimpleMessage("aneel.rahim@dit.ie",
                "aneel.rahim@dit.ie",
                "Hi! I am testing SMTP working." );
            client.disconnect();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

## 8. SMTP Header

```
import java.net.*;
import java.io.*;
import org.apache.commons.net.smtp.SMTPClient;
import org.apache.commons.net.smtp.SimpleSMTPHeader;
class SmtpHeader{
    public static void main(String[] args) {
        String sender = "aneel.rahim@dit.ie";
        String recipient = "aneel.rahim@dit.ie";
        String subject = "SMTP Testing";
        try {
            SMTPClient client = new SMTPClient();
            client.connect("127.0.0.1");
            client.setSender(sender);
            client.addRecipient(recipient);
        }
    }
}
```

## 8. SMTP Header

```
Writer writer = client.sendMessageData();  
SimpleSMTPHeader header = new  
SimpleSMTPHeader(sender, recipient, subject);  
writer.write(header.toString());  
writer.write("Hi! I am testing SMTP Header");  
writer.close();  
client.completePendingCommand();  
client.disconnect();  
}  
catch (Exception e) {  
    System.out.println(e);  
} } }
```

## 9. TCP Daytime Client

```
import java.net.*;
import java.io.*;
import org.apache.commons.net.daytime.DaytimeTCPClient;
class MyTCPClient{
    public static void main(String[] args) {
        try {
            DaytimeTCPClient client = new DaytimeTCPClient();
            client.connect("time-a.nist.gov");
            System.out.println("Connected Successfully to NIST Server");
            System.out.println("Getting the Time from Server");
            System.out.println(client.getTime());
            client.disconnect();
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

# 10. UDP Daytime Client

```
import java.net.*;
import java.io.*;
import java.net.InetAddress;
import org.apache.commons.net.daytime.DaytimeUDPClient;

class MyUDPClient{

    public static void main(String[] args) {

        try {

            DaytimeUDPClient client = new DaytimeUDPClient();
            InetAddress address = InetAddress.getByName("time.nist.gov");
            client.setDefaultTimeout(60000);
            client.open();
            System.out.println("Getting the Time from UDP Server");
            System.out.println(client.getTime(address,13));
        }

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

# Lecture 05

## Concurrency in clients and servers

CMPU 3027  
Network Programming

# Some items Remaining from previous lecture

- Lab Solution
- Telnet Client using Java

# Lab Solution: Daytime client for UDP

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

public class UDPClient {

    public static void main(String[] args) {
        int PORT = 13;
        String HOSTNAME = "time.nist.gov";
        try {
            DatagramSocket socket = new DatagramSocket();
            socket.setSoTimeout(10000);
            InetAddress host = InetAddress.getByName(HOSTNAME);
            DatagramPacket request = new DatagramPacket(new byte[1], 1, host, PORT);
            DatagramPacket response = new DatagramPacket(new byte[1024], 1024);
```

# Lab Solution: Daytime client for UDP

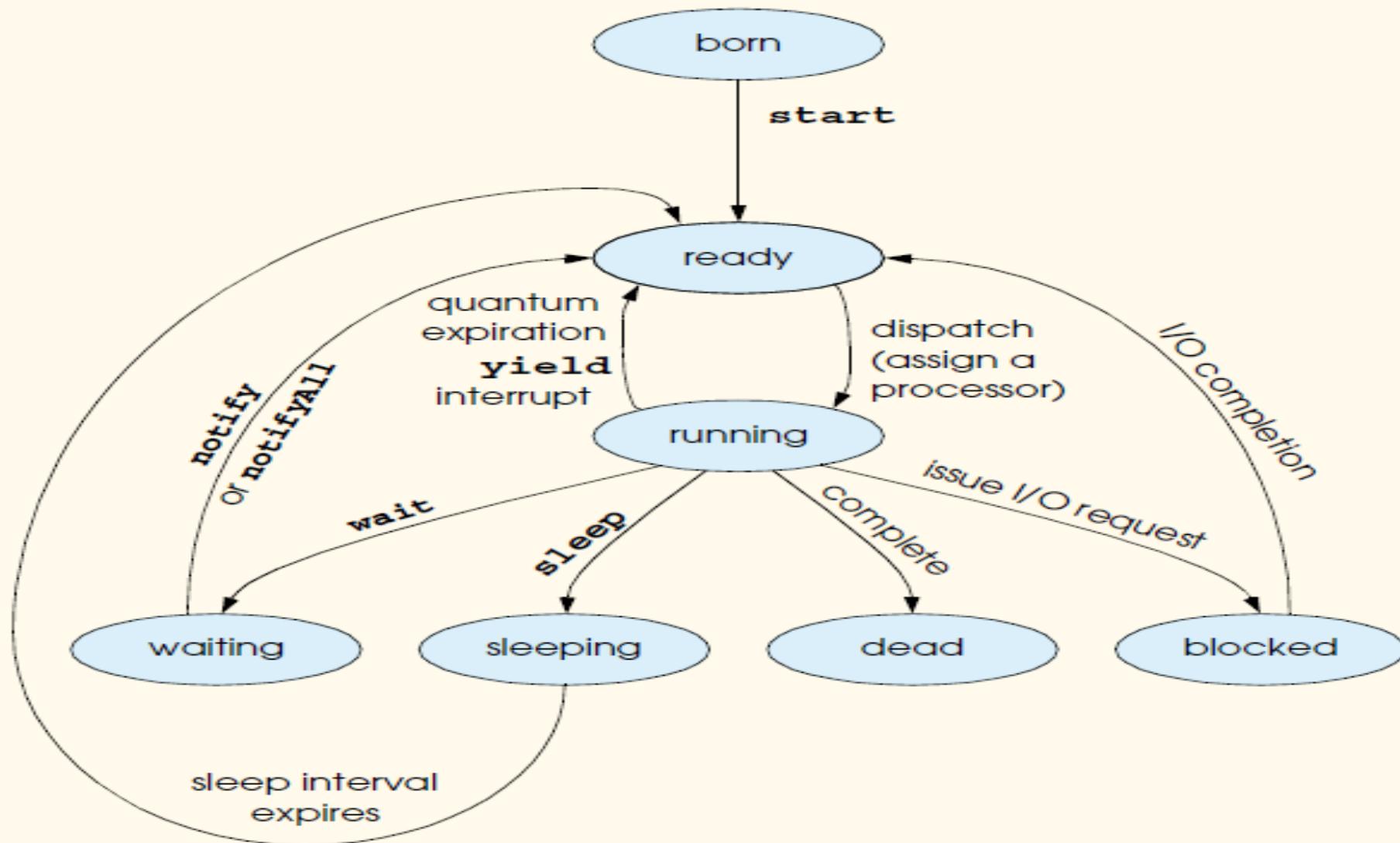
```
socket.send(request);  
socket.receive(response);  
String result = new String(response.getData(), 0,  
response.getLength(),"US-ASCII");  
System.out.println(result);}  
catch (Exception e) {  
    System.out.println(e);  
}
```

# Telnet Client

```
import java.io.*;
import org.apache.commons.net.telnet.TelnetClient;
public class Telnetclient {
    public static void main(String[] args) {
        TelnetClient telnet = new TelnetClient();
        try {
            telnet.connect("telehack.com",23);
            InputStream input = telnet.getInputStream();
            int c;
            while ((c = input.read()) != -1)
                System.out.print((char)c);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

# Process and Threads

- A process is a program in execution.
- A process may be divided into a number of independent units known as threads.
- Threads are light-weight processes within a process.
- A process is a collection of one or more threads and associated system resources



Thread Life Cycle

# Two ways of Creating Threads

- Extending the Thread Class
- Implementing the Runnable Interface

# Steps for creating thread by extending the Thread Class

1. Create a class by extending the Thread class and override the run() method:

```
class MyThread extends Thread {  
    public void run() {  
        // thread body of execution}  
}
```

2. Create a thread object:

```
MyThread thr1 = new MyThread();
```

3. Start Execution of created thread:

```
thr1.start();
```

# Creating Thread by extending Thread class

```
class MyThread extends Thread {  
    public void run() {  
        System.out.println("thread is running");  
    }  
    public static void main(String [] args) {  
        MyThread t = new MyThread();  
        t.start();  
        System.out.println("Main function is ended");  
    }  
}
```

# Steps for creating thread by implementing Thread Class

1. Create a class that implements the interface Runnable and override run() method:

```
class MyThread implements Runnable {  
    public void run() {  
        // thread body of execution}}
```

2. Creating Object:

```
MyThread myObject = new MyThread();
```

3. Creating Thread Object:

```
Thread thr1 = new Thread(myObject);
```

4. Start Execution:

```
thr1.start();
```

# Creating Thread by implementing Runnable interface

```
class MyThread2 implements Runnable {  
    public void run() {  
        System.out.println("thread is running");  
    }  
    public static void main(String [] args) {  
        Thread t = new Thread(new MyThread2());  
        t.start();  
        System.out.println("Main function is ended");  
    }  
}
```

# Writing Java Programs

1. Write a program to create multiple threads?
2. Write a program to create multiple threads to calculate math function?
3. Find the status of thread?
4. Write a program, to show thread join function?
5. Multiple threads to perform concurrent operations
6. Setting Thread Priority

# 1. Write a program to create multiple threads

```
public class ThreadTest {  
    public static void main( String args[] ) {  
        PrintThread thread1, thread2, thread3, thread4;  
        thread1 = new PrintThread( "thread1" );  
        thread2 = new PrintThread( "thread2" );  
        thread3 = new PrintThread( "thread3" );  
        thread4 = new PrintThread( "thread4" );  
        thread1.start();  
        thread2.start();  
        thread3.start();  
        thread4.start();  
    }    }
```

# 1. Write a program to create multiple threads

```
class PrintThread extends Thread {  
    public PrintThread( String name ) {  
        super( name );  
        System.out.println( getName() + " Constructor" );  
    }  
    public void run() {  
        System.out.println( getName() + " is Running" );  
    }  
}
```

## 2. Multiple threads to calculate math function

```
class MathThread {  
    public static void main(String [] args ) {  
        MathPow mp = new MathPow();  
        MathSqrt ms = new MathSqrt();  
        MathFloor mf = new MathFloor();  
        mp.start();  
        ms.start();  
        mf.start();  
    }  
}
```

## 2. Multiple threads to calculate math function

```
class MathPow extends Thread {  
    public void run() {  
        System.out.println("Calculating Power " + Math.pow(2,3) );}  
  
class MathSqrt extends Thread {  
    public void run() {  
        System.out.println("Calculating Sqrt " + Math.sqrt(900) );}  
  
class MathFloor extends Thread {  
    public void run() {  
        System.out.println("Calculating Floor " + Math.floor(9.4) );}
```

### 3. Find the status of thread?

```
class ThreadStatus extends Thread {  
    public void run() {  
        System.out.println("I am in Run Function and Thread Status "  
        + Thread.currentThread().isAlive());  
    }  
    public static void main(String [] args ) throws Exception {  
        ThreadStatus t = new ThreadStatus();  
        t.start();  
        t.join();  
        System.out.println("I am in Main Function and Thread Status "  
        + t.isAlive());  
    }    }
```

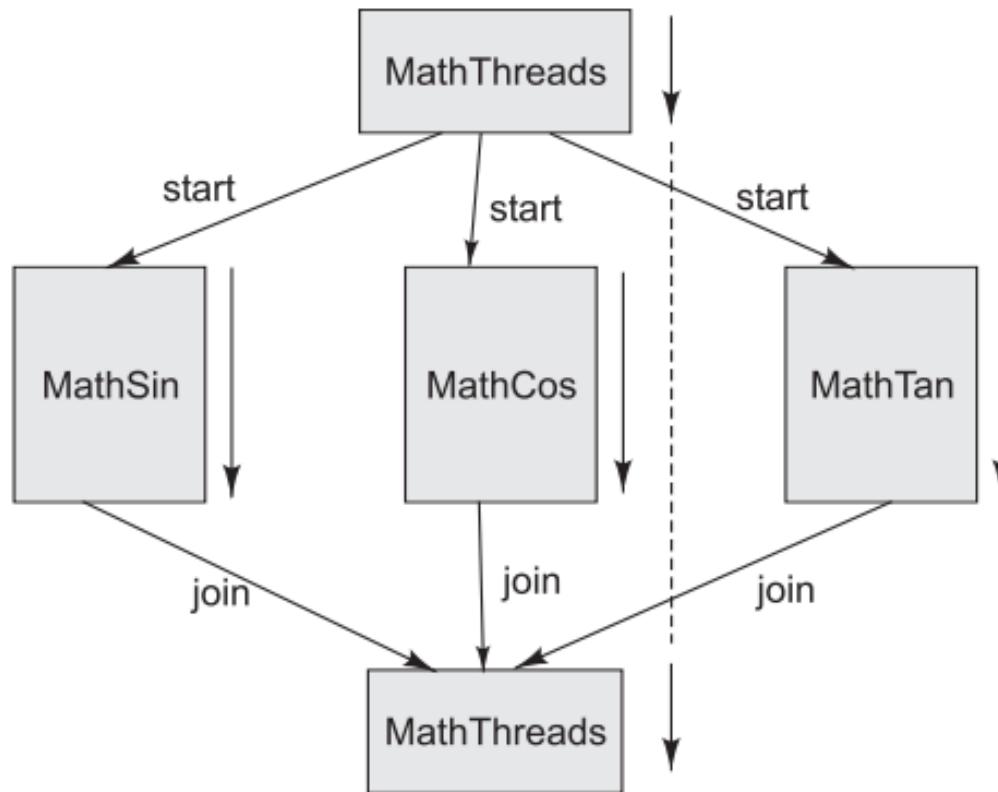
## 4. Write a program to show thread join function?

```
class ThreadJoin extends Thread {  
    public void run() {  
        System.out.println("Thread started:::"  
            +Thread.currentThread().getName());  
    }  
    public static void main(String [] args) throws Exception {  
        ThreadJoin thread1 = new ThreadJoin();  
        ThreadJoin thread2 = new ThreadJoin();  
        ThreadJoin thread3 = new ThreadJoin();  
        thread1.start();  
        thread1.join();  
        System.out.println("Thread-0 status:::" +thread1.isAlive());  
    }  
}
```

## 4. Write a program to show thread join function?

```
thread2.start();
thread2.join();
System.out.println("Thread-1 status:::" +thread2.isAlive());
thread3.start();
thread3.join();
System.out.println("Thread-2 status:::" +thread3.isAlive());
System.out.println("All threads are dead, exiting main thread");
}
```

## 5. Multiple threads to perform concurrent operations



# 5. Multiple threads to perform concurrent operations

```
import java.lang.Math;

class MathSin extends Thread {
    public double deg;
    public double res;
    public MathSin(int degree) {
        deg = degree;
    }
    public void run() {
        System.out.println("Execute sin" + deg);
        double Deg2Rad = Math.toRadians(deg);
        res = Math.sin(Deg2Rad);
        System.out.println("Exit MathSin. Res =" + res);
    }
}
```

```
class MathCos extends Thread {
    public double deg;
    public double res;
    public MathCos(int degree) {
        deg = degree;
    }
    public void run() {
        System.out.println("Execute cos" + deg);
        double Deg2Rad = Math.toRadians(deg);
        res = Math.cos(Deg2Rad);
        System.out.println("Exit MathCos. Res =" + res);
    }
}
```

## 5. Multiple threads to perform concurrent operations

```
class MathTan extends Thread {  
  
    public double deg;  
  
    public double res;  
  
    public MathTan(int degree) {  
  
        deg = degree;  
  
    }  
  
    public void run() {  
  
        System.out.println("Execute tan" + deg);  
        double Deg2Rad = Math.toRadians(deg);  
        res = Math.tan(Deg2Rad);  
        System.out.println("Exit MathTan. Res ="  
        +res);  
    }  
}
```

```
class MathFunThread {  
  
    public static void main(String args[]) {  
  
        MathSin st = new MathSin(45);  
        MathCos ct = new MathCos(60);  
        MathTan tt = new MathTan(30);  
        st.start();  
        ct.start();  
        tt.start();  
  
        try { // wait for completion of threads  
            st.join();  
            ct.join(); //wait for MathCos object  
            tt.join();  
            double z = st.res + ct.res + tt.res;  
            System.out.println("Sum of sin cos tan = " +z);  
        } catch(Exception e) {  
            System.out.println(e);  
        }  
    }  
}
```

# 6. Setting Thread Priority

```
class A extends Thread {  
  
    public void run() {  
  
        System.out.println("ThreadA Function");  
  
        for(int i = 1; i <= 4000; i++)  
  
            System.out.println("\t ThreadA: i= " + i);  
  
        System.out.println("Exit from A"); } }
```

```
class B extends Thread {  
  
    public void run() {  
  
        System.out.println("ThreadB Function");  
  
        for(int j = 1; j <= 4000; j++)  
  
            System.out.println("\t ThreadB: j= " + j);  
  
        System.out.println("Exit from B"); } }
```

```
class C extends Thread {  
  
    public void run() {  
  
        System.out.println("ThreadC Function");  
  
        for(int k = 1; k <= 4000; k++)  
  
            System.out.println("\t ThreadC: k= " + k);  
  
        System.out.println("Exit from C"); } }
```

# 6. Setting Thread Priority

```
public class ThreadPriority {  
    public static void main(String args[]) {  
        A threadA = new A();  
        B threadB = new B();  
        C threadC = new C();  
        threadC.setPriority(Thread.MAX_PRIORITY);  
        threadB.setPriority(threadA.getPriority() + 1);  
        threadA.setPriority(Thread.MIN_PRIORITY);  
        System.out.println("Started Thread A");  
        threadA.start();  
        System.out.println("Started Thread B");  
        threadB.start();  
        System.out.println("Started Thread C");  
        threadC.start();  
        System.out.println("End of main thread");  
    }  
}
```

# Lecture 06

## Concurrency in clients and servers...

CMPU 3027  
Network Programming

# 1. How to interrupt a Thread

```
1 class ThreadInterrupted extends Thread{  
2     public void run(){  
3         int counter =0;  
4         while(!Thread.interrupted())  
5         {  
6             System.out.println(counter++);  
7         }  
8     }  
9     public static void main(String args[]) throws Exception {  
10        ThreadInterrupted t1=new ThreadInterrupted();  
11        t1.start();  
12        System.in.read();  
13        t1.interrupt();  
14    }  
15 }
```

## 2.Thread Synchronization

```
1 class Counter{  
2     int count = 0;  
3     public synchronized void add(){  
4         this.count++;  
5         System.err.println(Thread.currentThread().getName() + "updated the counter to "+this.count);  
6     }  
7 }  
8 public class MyThread3 extends Thread{  
9     private Counter counter;  
10    public MyThread3(Counter counter){  
11        this.counter = counter;  
12    }  
13    public void run() {  
14        for(int i=0; i<5; i++)  
15            counter.add();  
16    }  
17    public static void main(String[] args){  
18        Counter counter = new Counter();  
19        Thread threadA = new MyThread3(counter);  
20        Thread threadB = new MyThread3(counter);  
21        threadA.start();  
22        threadB.start();  
23    }  
24 }
```

### 3.Single threaded TCP/IP Client/Server

```
1 import java.net.*;
2 import java.io.*;
3 public class Server {
4     public static void main(String[] args) {
5         ServerSocket server = null;
6         try {
7             server = new ServerSocket( 5000, 10 ); // create ServerSocket
8             System.out.println("Server is Running on port 5000");
9         }
10        catch (Exception ex) {
11            System.err.println(ex);
12        }
13
14        while (true){
15            try {
16                System.out.println("Waiting for Client to Connect");
17                Socket client = server.accept();
18                ConnectionThread t = new ConnectionThread(client);
19                t.start();
20            }
21            catch (Exception ex) {
22                System.err.println(ex);
23            } } } }
```

### 3. Single threaded TCP/IP Client/Server

```
25  class ConnectionThread extends Thread {  
26      private Socket connection;  
27      public ConnectionThread (Socket con){  
28          connection = con;  
29      }  
30      public void run() {  
31          try {  
32              ObjectOutputStream out = new ObjectOutputStream( connection.getOutputStream() );  
33              ObjectInputStream in = new ObjectInputStream( connection.getInputStream() );  
34              String message = "Connection successful";  
35              out.writeObject(message);  
36              message = ( String ) in.readObject();  
37              System.out.println ("Data from Client: " + message);  
38              connection.close();  
39          }  
40          catch (Exception ex) {  
41              System.err.println(ex);  
42          } } }
```

### 3. Single threaded TCP/IP Client/Server

```
1 import java.net.*;
2 import java.io.*;
3 public class Client {
4     public static void main(String[] args) {
5         try {
6             Socket client = new Socket( "localhost",5000 );
7             System.out.println("Client is connected to Server");
8             ObjectOutputStream out = new ObjectOutputStream( client.getOutputStream() );
9             ObjectInputStream in = new ObjectInputStream( client.getInputStream() );
10            String message = ( String ) in.readObject();
11            System.out.println ("Data from Server: " + message);
12            message = "Hello";
13            out.writeObject(message);
14            client.close();
15        }
16        catch (Exception ex) {
17            System.err.println(ex);
18        }
19    }
20 }
```

# Client Software Design

- Applications that act as clients are conceptually simpler than applications that act as servers for several reasons.
  - Most client software does not explicitly handle concurrent interactions with multiple servers.
  - Client software does not usually require special privilege because it does not usually access privileged protocol ports.
  - Most client software does not need to enforce protections.

# Programs for Lab practice

- Multithreaded TCP/IP client server
- Multithreaded UDP client server
- Not included for quiz and exam

## 4. Multithreaded TCP/IP Client/Server

```
1 import java.net.*;
2 import java.io.*;
3 import java.util.*;
4 public class MultiServer {
5     public static void main(String[] args) {
6         ServerSocket server = null;
7         try {
8             server = new ServerSocket( 5000, 10 ); // create ServerSocket
9             System.out.println("Server is Running on port 5000");
10        }
11    catch (Exception ex) {
12        System.err.println(ex);
13    }
14    try {
15        System.out.println("Waiting for Client to Connect");
16        Socket client = server.accept();
17        Thread r = new ReadingThread(client);
18        Thread w = new WritingThread(client);
19        w.start();
20        r.start();
21    }
22    catch (Exception ex) {
23        System.err.println(ex);
24    }
}
```

## 4. Multithreaded TCP/IP Client/Server

```
26  class ReadingThread extends Thread {  
27      private Socket connection;  
28      public ReadingThread (Socket con){  
29          connection = con;  
30      }  
31      public void run() {  
32          try {  
33              ObjectInputStream in = new ObjectInputStream( connection.getInputStream() );  
34              String message;  
35              while (true){  
36                  message = ( String ) in.readObject();  
37                  System.out.println ("Server>>>" + message);  
38              }  
39          }  
40          catch (Exception ex) {  
41              System.err.println(ex);  
42          }  
43      }  
}
```

## 4. Multithreaded TCP/IP Client/Server

```
45  class WritingThread extends Thread {  
46  
47      private Socket connection;  
48      public WritingThread (Socket con){  
49          connection = con;  
50      }  
51      public void run() {  
52          try {  
53              ObjectOutputStream out = new ObjectOutputStream( connection.getOutputStream() ) ;  
54              String message;  
55              Scanner s = new Scanner(System.in);  
56              while (true){  
57                  System.out.print("Server>>>");  
58                  message = s.nextLine();  
59                  out.writeObject(message);  
60              }  
61          }  
62          catch (Exception ex) {  
63              System.err.println(ex);  
64          }  
      }  
  }
```

## 4. Multithreaded TCP/IP Client/Server

```
1 import java.net.*;
2 import java.io.*;
3 import java.util.*;
4 public class MultiClient {
5     public static void main(String[] args) {
6         Socket client = null;
7         try {
8             client = new Socket( "localhost",5000 );
9             System.out.println("Client is connected to Server");
10        }
11        catch (Exception ex) {
12            System.err.println(ex);
13        }
14
15        try {
16            Thread r = new ReadingThread(client);
17            Thread w = new WritingThread(client);
18            w.start();
19            r.start();
20        }
21        catch (Exception ex) {
22            System.err.println(ex);
23        }
    } }
```

## 4. Multithreaded TCP/IP Client/Server

```
25 class ReadingThread extends Thread {  
26     private Socket connection;  
27     public ReadingThread (Socket con){  
28         connection = con;  
29     }  
30     public void run() {  
31         try {  
32             ObjectInputStream in = new ObjectInputStream( connection.getInputStream() );  
33             String message;  
34             while (true){  
35                 message = ( String ) in.readObject();  
36                 System.out.println ("Client>>>" + message);  
37             }  
38         }  
39         catch (Exception ex) {  
40             System.err.println(ex);  
41         }  
42     }  
43 }
```

## 4. Multithreaded TCP/IP Client/Server

```
45  class WritingThread extends Thread {  
46      private Socket connection;  
47      public WritingThread (Socket con){  
48          connection = con;  
49      }  
50      public void run() {  
51          try {  
52              ObjectOutputStream out = new ObjectOutputStream( connection.getOutputStream() );  
53              String message;  
54              Scanner s = new Scanner(System.in);  
55              while (true){  
56                  System.out.print("Client>>>");  
57                  message = s.nextLine();  
58                  out.writeObject(message);  
59              }  
60          catch (Exception ex) {  
61              System.err.println(ex);  
62          }  
63      }  
64  }
```

## 5. Multithreaded UDP Client/Server

```
1 import java.net.*;
2 import java.io.*;
3 import java.util.*;
4 public class MultiUDPServer {
5     public static void main(String[] args) {
6         DatagramSocket server = null;
7         try {
8             server = new DatagramSocket(5001);
9             System.out.println("Server is Running on port 5001");
10        }
11    catch (Exception ex) {
12        System.err.println(ex);
13    }
14    try {
15        Thread r = new ReadingThread(server);
16        Thread w = new WritingThread(server);
17        r.start();
18        w.start();
19    }
20    catch (Exception ex) {
21        System.err.println(ex);
22    }
23 }
```

## 5. Multithreaded UDP Client/Server

```
24 class ReadingThread extends Thread {  
25     private DatagramSocket readingserver;  
26     public static int clientport;  
27     public ReadingThread (DatagramSocket server){  
28         readingserver = server;  
29     }  
30     public void run() {  
31         byte data[] = new byte[ 100 ];  
32         DatagramPacket receivePacket = new DatagramPacket( data, data.length );  
33         try {  
34             readingserver.receive(receivePacket);  
35             clientport = receivePacket.getPort();  
36             System.out.println(new String (receivePacket.getData()));  
37             while (true){  
38                 readingserver.receive(receivePacket);  
39                 System.out.println(new String (receivePacket.getData()));  
40             }  
41         }  
42         catch (Exception ex) {  
43             System.err.println(ex);  
44         } } }
```

## 5. Multithreaded UDP Client/Server

```
46  class WritingThread extends Thread {  
47  
48      private DatagramSocket writingserver;  
49      public WritingThread (DatagramSocket server){  
50          writingserver= server;  
51      }  
52      public void run() {  
53          try {  
54              String message = null;  
55              Scanner s = new Scanner(System.in);  
56              byte data [] ;  
57              while (true){  
58                  message = s.nextLine();  
59                  data = message.getBytes();  
60                  DatagramPacket sendPacket = new DatagramPacket( data, data.length,  
61                                  InetAddress.getLocalHost(),ReadingThread.clientport);  
62                  System.out.println("Sending Data to Client");  
63                  writingserver.send(sendPacket);  
64              }  
65          }  
66          catch (Exception ex) {  
67              System.err.println(ex);  
68      }  } }
```

## 5. Multithreaded UDP Client/Server

```
1 import java.net.*;
2 import java.io.*;
3 import java.util.*;
4 public class MultiUDPClient {
5     public static void main(String[] args) {
6         DatagramSocket client = null;
7         try {
8             client = new DatagramSocket();
9         }
10        catch (Exception ex) {
11            System.err.println(ex);
12        }
13        try {
14            Thread r = new ReadingThread(client);
15            Thread w = new WritingThread(client);
16            w.start();
17            r.start();
18        }
19        catch (Exception ex) {
20            System.err.println(ex);
21        }
    }
```

## 5. Multithreaded UDP Client/Server

```
23  class ReadingThread extends Thread {  
24      private DatagramSocket readingserver;  
25      public ReadingThread (DatagramSocket server){  
26          readingserver = server;  
27      }  
28      public void run() {  
29          byte data[] = new byte[ 100 ];  
30          DatagramPacket receivePacket = new DatagramPacket( data, data.length );  
31          try {  
32              while (true){  
33                  readingserver.receive(receivePacket);  
34                  System.out.println(new String (receivePacket.getData()));  
35              }  
36          }  
37          catch (Exception ex) {  
38              System.err.println(ex);  
39          } } }
```

## 5. Multithreaded UDP Client/Server

```
41  class WritingThread extends Thread {  
42      private DatagramSocket writingserver;  
43      public WritingThread (DatagramSocket server){  
44          writingserver= server;  
45      }  
46      public void run() {  
47          try {  
48              String message;  
49              Scanner s = new Scanner(System.in);  
50              byte data [] ;  
51              while (true){  
52                  message = s.nextLine();  
53                  data = message.getBytes();  
54                  DatagramPacket sendPacket = new DatagramPacket( data, data.length,  
55                      InetAddress.getLocalHost(),5001);  
56                  System.out.println("Sending Data to Server");  
57                  writingserver.send(sendPacket);  
58              }  
59          }  
60          catch (Exception ex) {  
61              System.err.println(ex);  
62          }  
63      }  
64  }
```

# Lecture 07

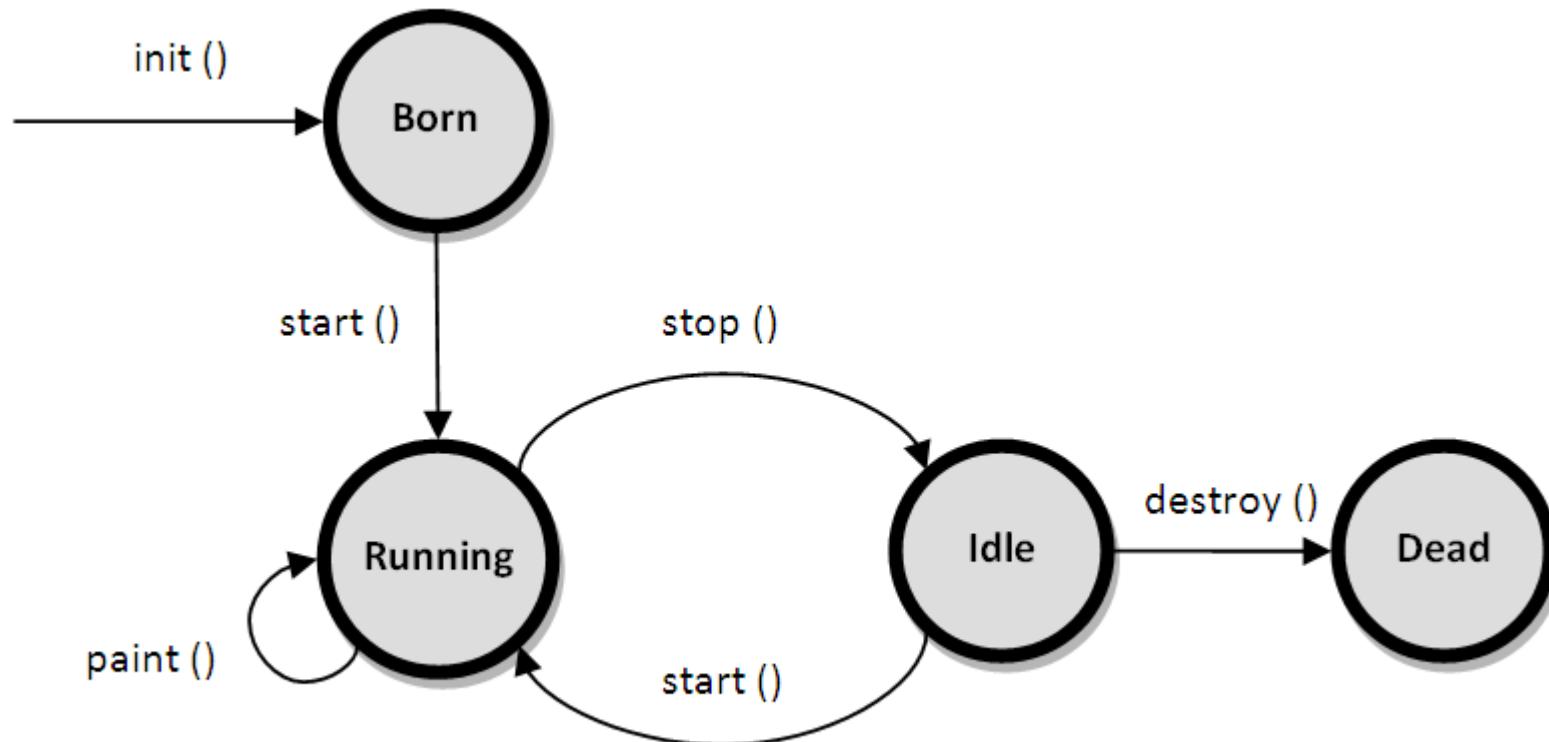
# Client side programming

CMPU 3027  
Network Programming

# Java Applets

- A Java applet is a special kind of Java application that is designed to work within a web browser.
- Applets are not permitted to perform file or certain memory operations on the client machine. For example trying to delete files on the client machine.
- Java applets are usually compact(4kB to 10kB), so that they can be easily downloaded across the web.

# Applets life cycle



# Running Java Applets

- How to compile java code
  - `javac WelcomeApplet.java`
- How to run java applet
  - `appletViewer WelcomeApplet.html`

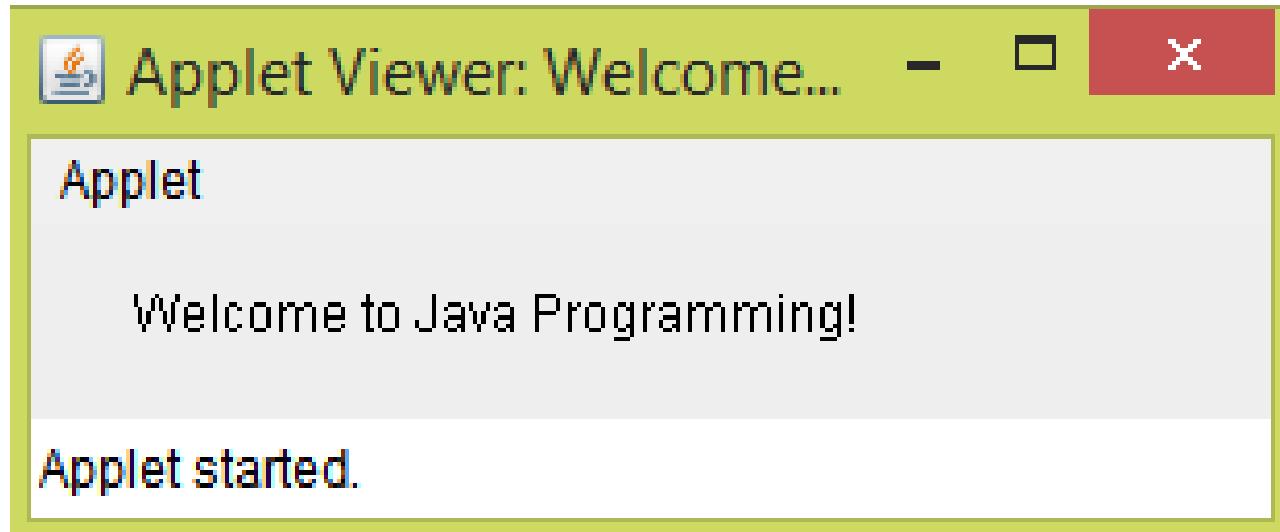
# 1. WelcomeApplet.java

```
1 import java.awt.Graphics;
2 import javax.swing.JApplet;
3 public class WelcomeApplet extends JApplet {
4     public void paint( Graphics g ){
5         super.paint( g );
6         g.drawString( "Welcome to Java Programming!", 25, 25 );
7     }
8 }
```

# 1. WelcomeApplet.html

```
1 <html>
2   <applet code = "WelcomeApplet.class" width = "300" height = "45">
3     </applet>
4   </html>
```

# 1. Output of WelcomeApplet.html



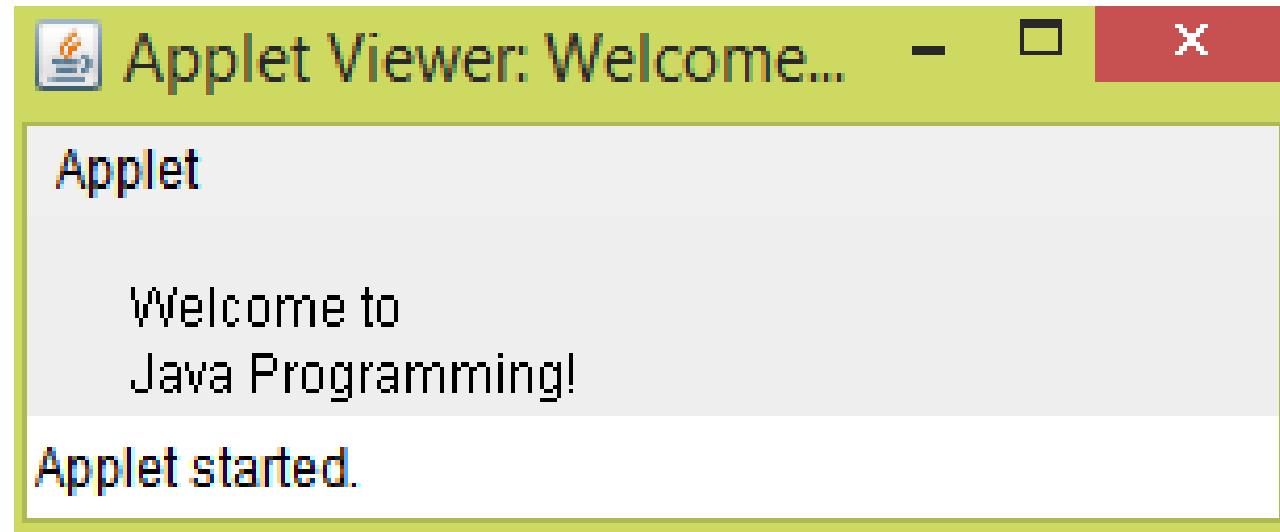
## 2. Print Multiple String using Applet

```
1 import java.awt.Graphics;
2 import javax.swing.JApplet;
3 public class WelcomeApplet2 extends JApplet {
4     public void paint( Graphics g ){
5         super.paint( g );
6         g.drawString( "Welcome to", 25, 25 );
7         g.drawString( "Java Programming!", 25, 40 );
8     }
9 }
```

## 2. WelcomeApplet2.html

```
1 <html>
2   <applet code = "WelcomeApplet2.class" width = "300" height = "45">
3     </applet>
4   </html>
```

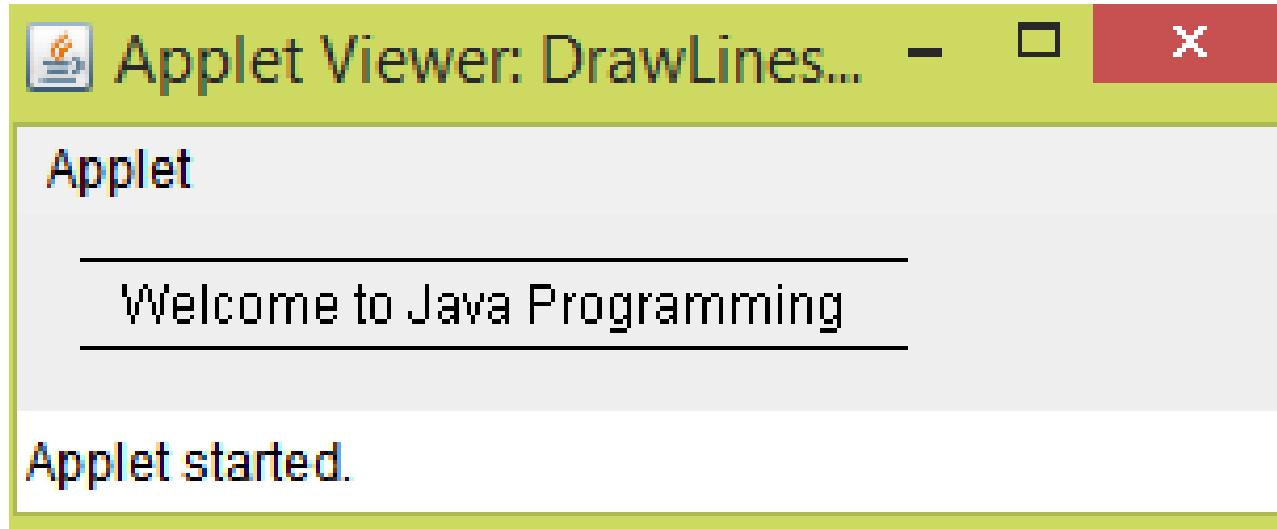
## 2. Output of WelcomeApplet2.html



### 3. Draw Line using Applet

```
1 import java.awt.Graphics;
2 import javax.swing.JApplet;
3 public class DrawLines extends JApplet {
4     public void paint( Graphics g ){
5         super.paint( g );
6         g.drawLine( 15, 10, 210, 10 );
7         g.drawLine( 15, 30, 210, 30 );
8         g.drawString( "Welcome to Java Programming", 25, 25 );
9     }
10 }
```

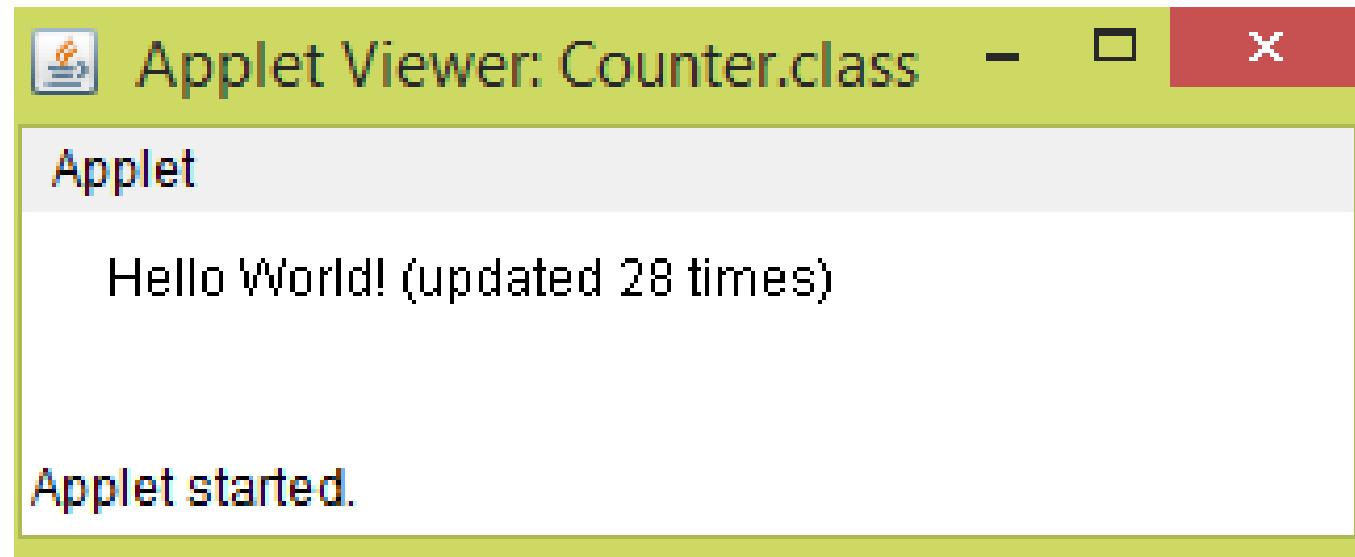
### 3. Output of DrawLines.html



## 4. Paint function Counter

```
1 import java.awt.Graphics;
2 import java.applet.Applet;
3 public class Counter extends Applet {
4     private int count = 0;
5     public void paint(Graphics g){
6         count++;
7         g.drawString("Hello World! (updated " + count + " times)", 20, 20);
8     }
9 }
```

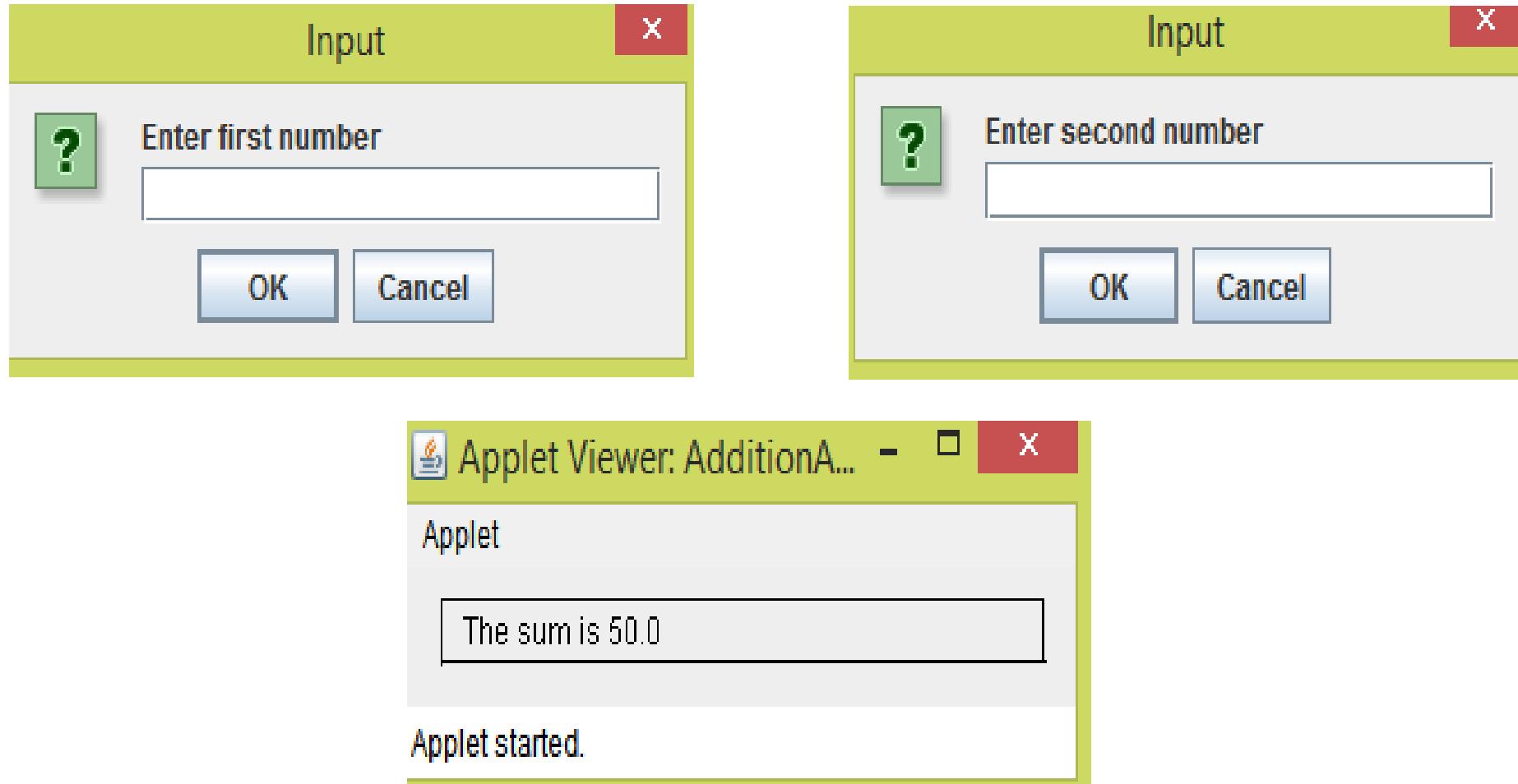
## 4. Output of Counter.html



# 5. Add two number using Java Applets

```
1 import java.awt.Graphics;
2 import javax.swing.*;
3 public class AdditionApplet extends JApplet {
4     double sum;
5     public void init(){
6         String firstNumber; // first string entered by user
7         String secondNumber; // second string entered by user
8         double number1; // first number to add
9         double number2; // second number to add
10        firstNumber = JOptionPane.showInputDialog("Enter first number");
11        secondNumber = JOptionPane.showInputDialog("Enter second number");
12        number1 = Double.parseDouble( firstNumber );
13        number2 = Double.parseDouble( secondNumber );
14        sum = number1 + number2;
15    }
16    public void paint( Graphics g ){
17        super.paint( g );
18        g.drawRect( 15, 10, 270, 20 );
19        g.drawString( "The sum is " + sum, 25, 25 );
20    }
21 }
```

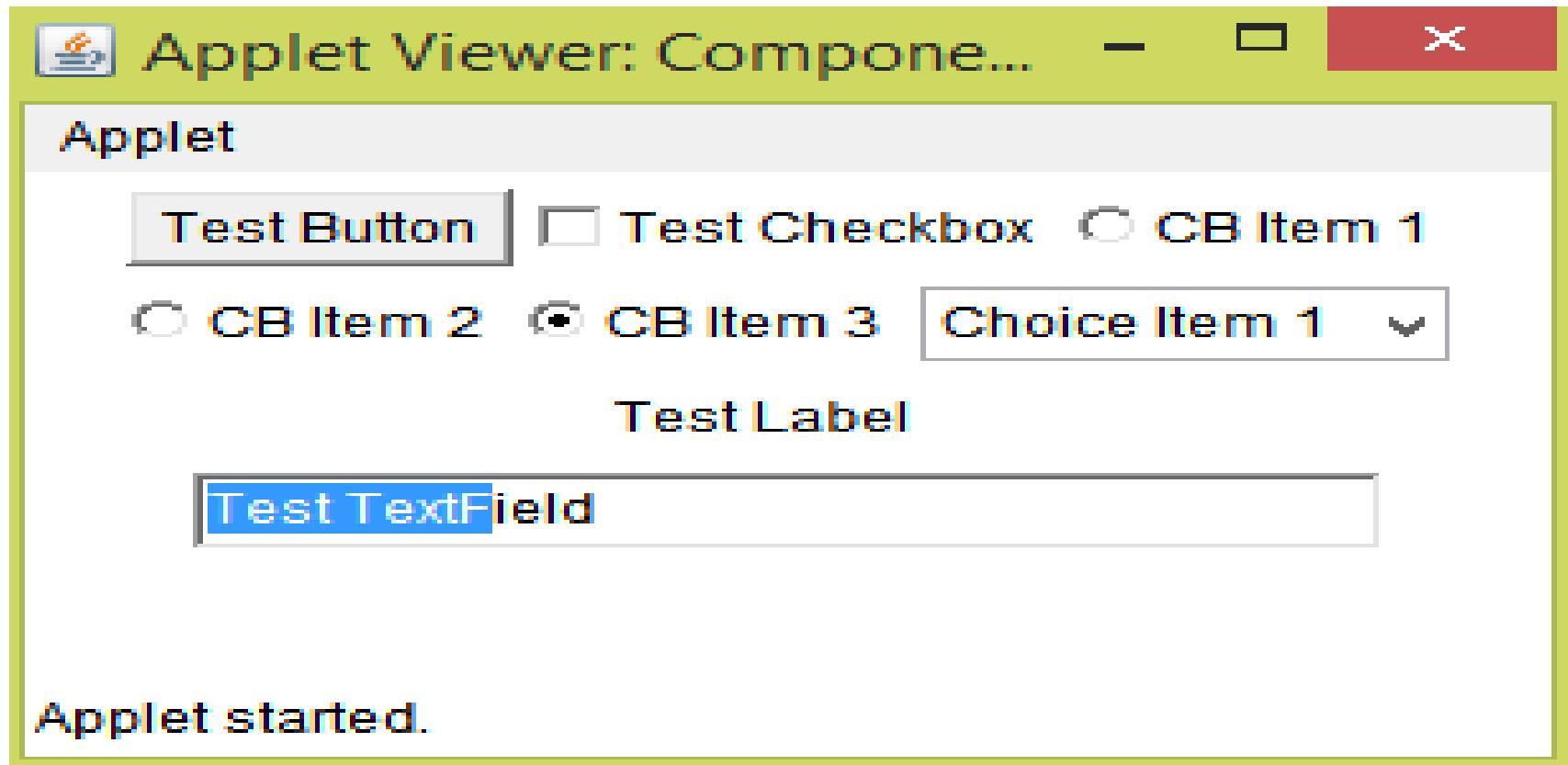
# 5. Output of AdditionApplet.html



# 6. Create button, checkbox, label and text field

```
1 import java.applet.Applet;
2 import java.awt.*;
3 public class ComponentApplet extends Applet {
4     public void init(){
5         Button b = new Button("Test Button");
6         add(b);
7         Checkbox cb = new Checkbox("Test Checkbox");
8         add(cb);
9         CheckboxGroup cbg = new CheckboxGroup();
10        add(new Checkbox("CB Item 1", cbg, false));
11        add(new Checkbox("CB Item 2", cbg, false));
12        add(new Checkbox("CB Item 3", cbg, true));
13        Choice choice = new Choice();
14        choice.addItem("Choice Item 1");
15        choice.addItem("Choice Item 2");
16        choice.addItem("Choice Item 3");
17        add(choice);
18        Label l = new Label("Test Label");
19        add(l);
20        TextField t = new TextField("Test TextField", 30);
21        add(t);
22    }
23 }
```

# 6. ComponentApplet.html



## 7. Draw Colour Rectangle

```
1 import java.applet.Applet;
2 import java.awt.*;
3 public class ShowColor extends Applet {
4     public void paint(Graphics g){
5         super.paint( g );
6         g.setColor( new Color( 255, 0, 0 ) );
7         g.fillRect( 25, 25, 100, 20 ); // x=25,y=25,100 wide,20 tall
8         g.drawString( "Red Rectangle " , 130, 40 );
9         g.setColor( new Color( 0.0f, 1.0f, 0.0f ) );
10        g.fillRect( 25, 50, 100, 20 );
11        g.drawString( "Green Rectangle " , 130, 65 );
12        g.setColor( Color.blue );
13        g.fillRect( 25, 75, 100, 20 );
14        g.drawString( "Blue Rectangle " , 130, 90 );
15    }
}
```

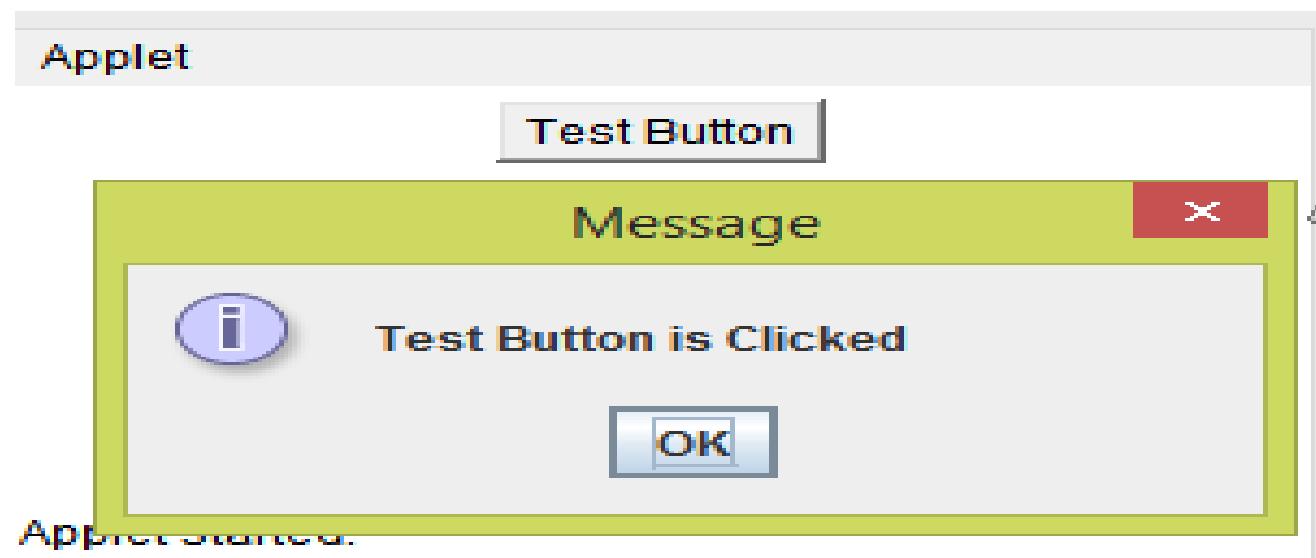
# 7. ShowColor.html



## 8. Create button with click event

```
1 import java.applet.Applet;
2 import java.awt.*;
3 import java.awt.event.*;
4 import javax.swing.*;
5 public class ButtonTest extends Applet implements ActionListener {
6     public void init() {
7         Button b = new Button("Test Button");
8         add(b);
9         b.addActionListener(this);
10    }
11    public void actionPerformed(ActionEvent e) {
12        JOptionPane.showMessageDialog( null, "Test Button is Clicked" );
13    }
}
```

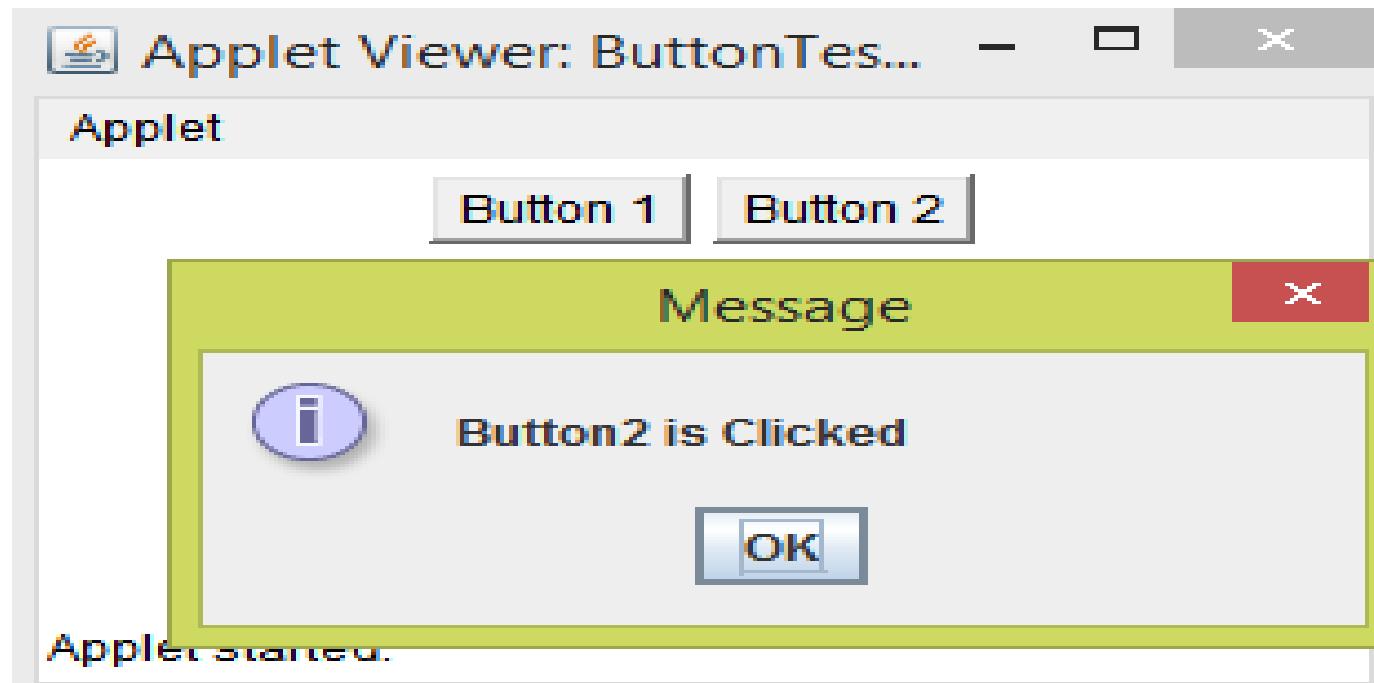
# 8. ButtonTest.html



# 9. Create Multiple button with click event

```
1 import java.applet.Applet;
2 import java.awt.*;
3 import java.awt.event.*;
4 import javax.swing.*;
5 public class ButtonTest2 extends Applet implements ActionListener {
6     Button b1,b2;
7     public void init(){
8         b1 = new Button("Button 1");
9         b2 = new Button("Button 2");
10        add(b1);
11        add(b2);
12        b1.addActionListener(this);
13        b2.addActionListener(this);
14    }
15    public void actionPerformed(ActionEvent e) {
16        if ( e.getSource() == b1)
17            JOptionPane.showMessageDialog( null, "Button1 is Clicked" );
18        if ( e.getSource() == b2)
19            JOptionPane.showMessageDialog( null, "Button2 is Clicked" );
20    }
}
```

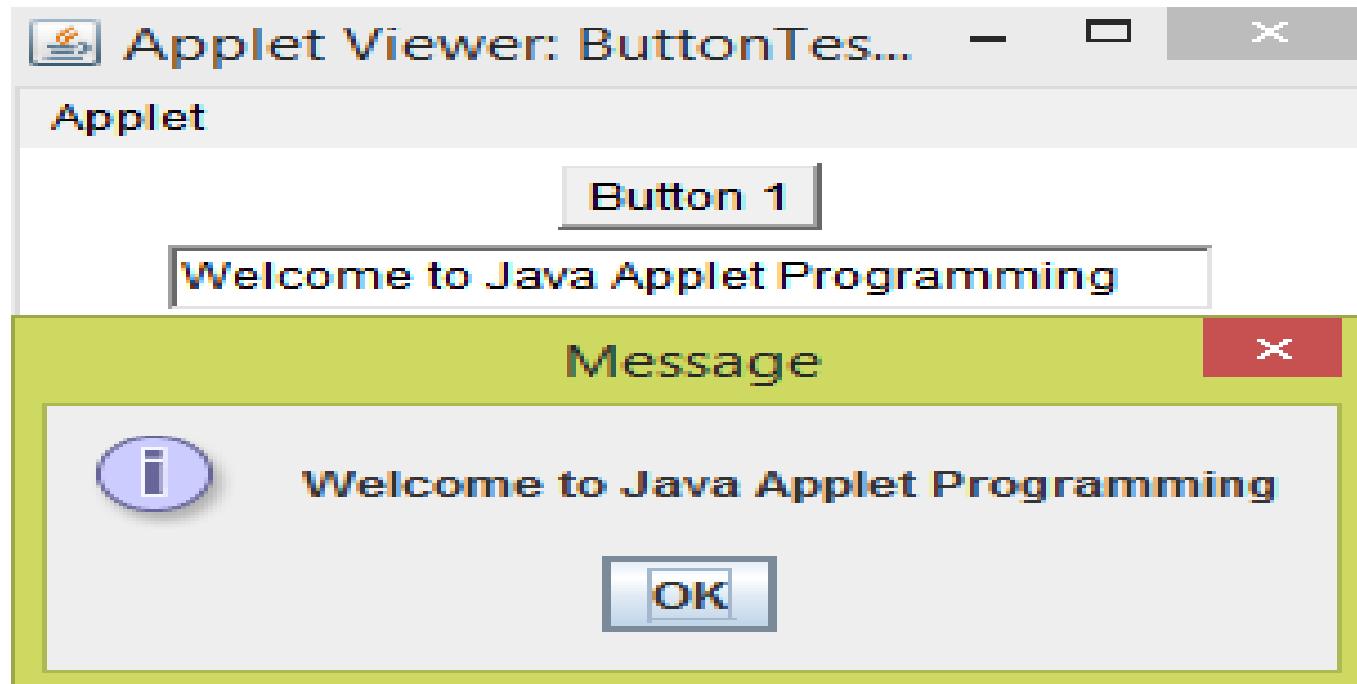
# 9. ButtonTest2.html



# 10. Create Text Field button

```
1 import java.applet.Applet;
2 import java.awt.*;
3 import java.awt.event.*;
4 import javax.swing.*;
5 public class ButtonTest3 extends Applet implements ActionListener {
6     Button b1;
7     TextField t1;
8     public void init(){
9         b1 = new Button("Button 1");
10        t1 = new TextField(30);
11        add(b1);
12        add(t1);
13        b1.addActionListener(this);
14    }
15    public void actionPerformed(ActionEvent e) {
16        String message = t1.getText();
17        JOptionPane.showMessageDialog( null, message );
18    }
}
```

# 10. ButtonTest3.html



# 11. Show Image

```
1 import java.awt.*;
2 import java.applet.Applet;
3 public class ImageTest extends Applet {
4     public void paint(Graphics g) {
5         Image myImage = this.getImage( this.getDocumentBase(), "dit.gif" );
6         g.drawImage( myImage, 10, 10, this);
7     }
8 }
```

# 11 .ImageTest.html



# 12. Play Audio File

```
1 import java.awt.*;
2 import java.applet.Applet;
3 import java.applet.AudioClip;
4 public class AudioTest extends Applet {
5     public void paint(Graphics g) {
6         AudioClip clip = this.getAudioClip( this.getDocumentBase() , "Alarm.wav" );
7         clip.play();
8     }
9 }
```

# 13. TCP Client Server with Graphical Interface....

```
1 import java.applet.Applet;
2 import java.awt.*;
3 import java.awt.event.*;
4 import javax.swing.*;
5 import java.net.*;
6 import java.io.*;
7 public class Server extends JFrame implements ActionListener {
8     JButton b1;
9     JTextField t1;
10    ServerSocket server = null;
11    Socket connection;
12    ObjectOutputStream out;
13    public Server(){
14        super( "Server" );
15        Container container = getContentPane();
16        container.setLayout( new FlowLayout() );
17        b1 = new JButton("Send to Client");
18        t1 = new JTextField(30);
19        container.add(b1);
20        container.add(t1);
21        setSize( 400, 300 );
22        setVisible( true );
23        b1.addActionListener(this);
24        try {
25            server = new ServerSocket( 4000, 10 ); // create ServerSocket
26            System.out.println("Server is Running on port 4000");
27            connection = server.accept();
28            out = new ObjectOutputStream( connection.getOutputStream() );
29        }
30        catch (Exception ex) {
31            System.err.println(ex);
32        }
33    }
34}
```

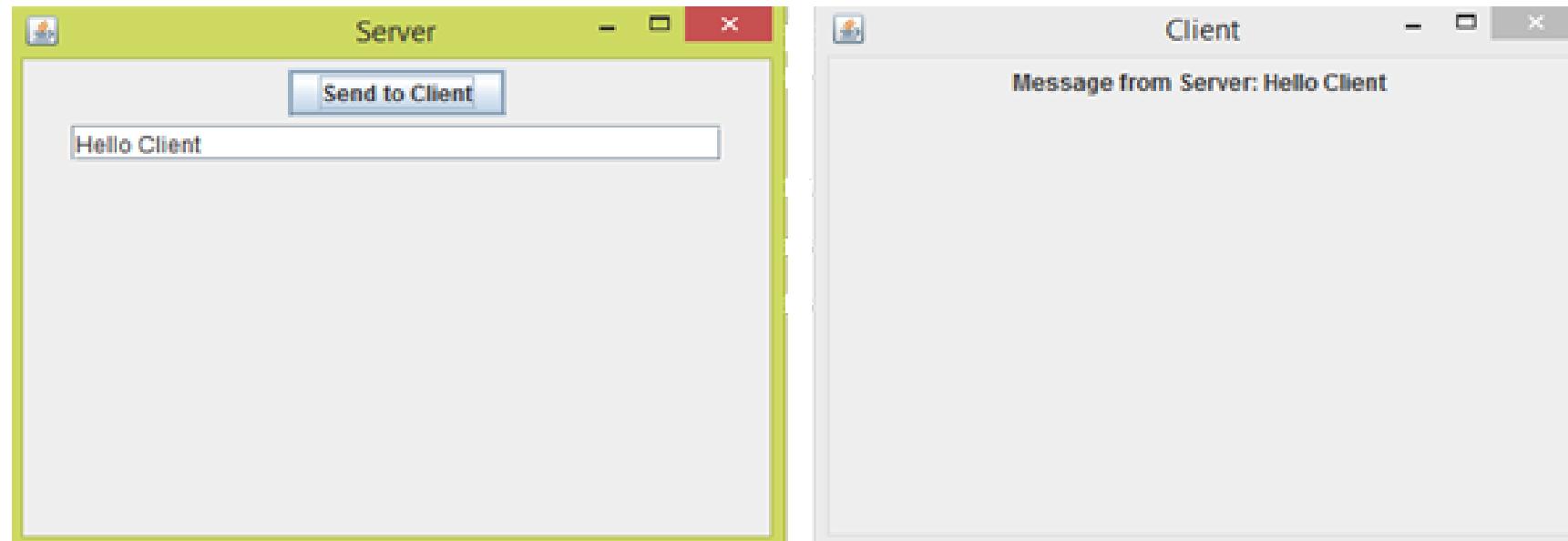
# 13. TCP Client Server with Graphical Interface....

```
34     public void actionPerformed(ActionEvent e) {  
35         try {  
36             String message = t1.getText();  
37             out.writeObject(message);  
38             System.out.println("Data send to client");  
39         }  
40         catch (Exception ex) {  
41             System.err.println(ex);  
42         }  
43     public static void main( String args[] ) {  
44         Server application = new Server();  
45         application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE );  
46     } }
```

# 13. TCP Client Server with Graphical Interface....

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4 import java.net.*;
5 import java.io.*;
6 public class Client extends JFrame {
7     public Client (){
8         super( "Client" );
9         JLabel label = new JLabel("Server Data will print here");
10        Container container = getContentPane();
11        container.setLayout( new FlowLayout() );
12        container.add(label);
13        setSize( 400, 300 );
14        setVisible( true );
15        try {
16            Socket client = new Socket( "localhost",4000 );
17            System.out.println("Client is connected to Server");
18            ObjectInputStream in = new ObjectInputStream( client.getInputStream() );
19            while (true) {
20                String message = ( String ) in.readObject();
21                label.setText( "Message from Server: "+ message );
22            }
23        catch (Exception ex) {
24            System.err.println(ex);
25        }
26        public static void main( String args[] ) {
27            Client application = new Client();
28            application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE );
29        }
30    }
```

# Output of Client/Server Program



# Lecture 08

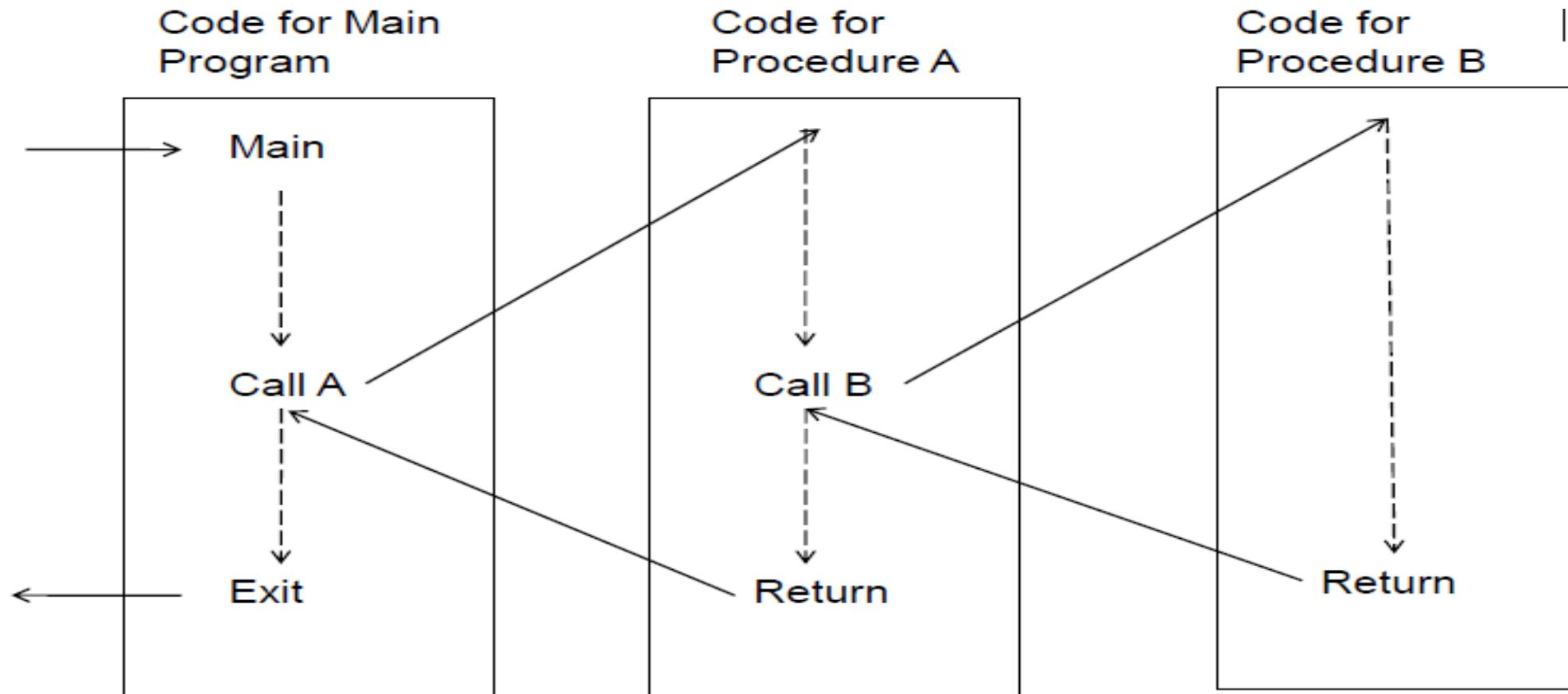
## RPC and RMI

CMPU 3027  
Network Programming

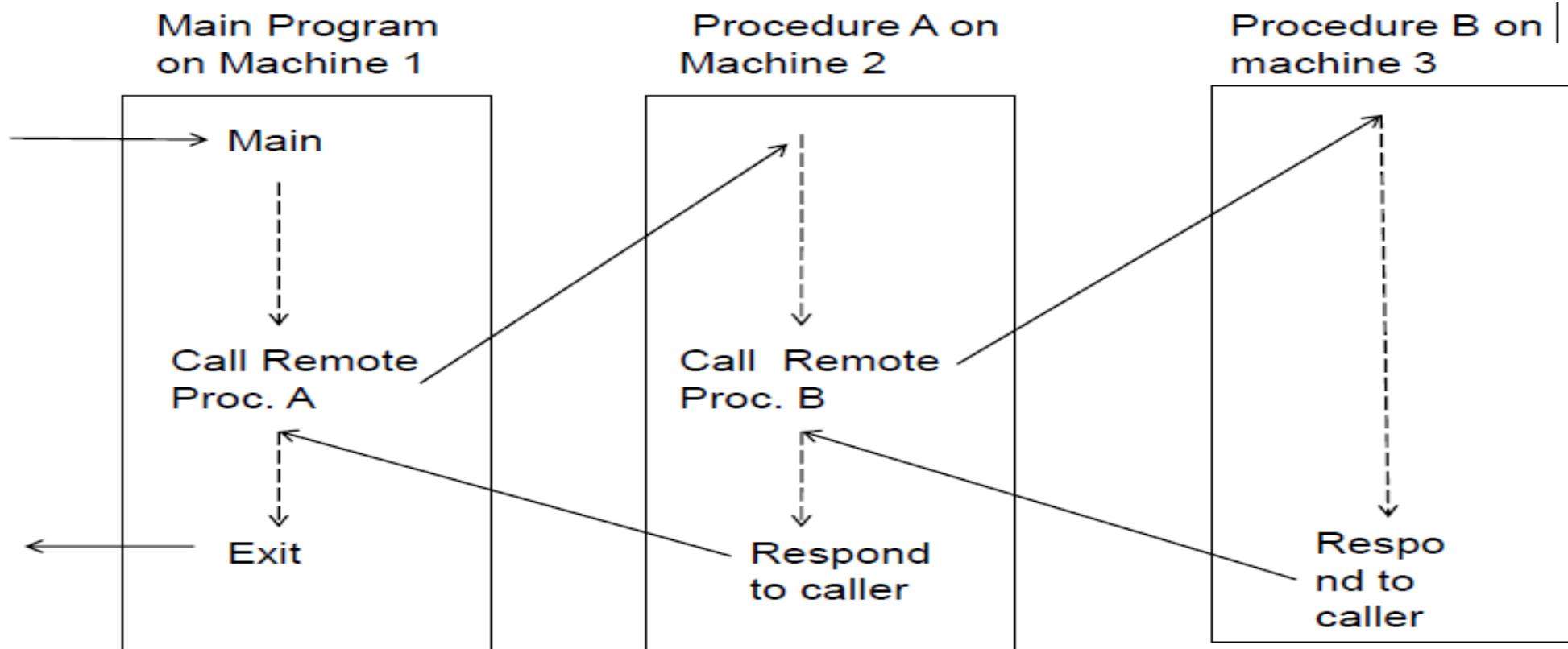
# Remote procedure calls (RPC)

- It allows a procedural program(i.e., a program written in C) to call a function residing on another computer as conveniently as if that function were part of the same program running on the same computer.
- RPC performs all the networking and marshaling of data (i.e., packaging of function arguments and return values for transmission over a network).

# Flow of control during normal function Call



# Flow of control during remote procedure call



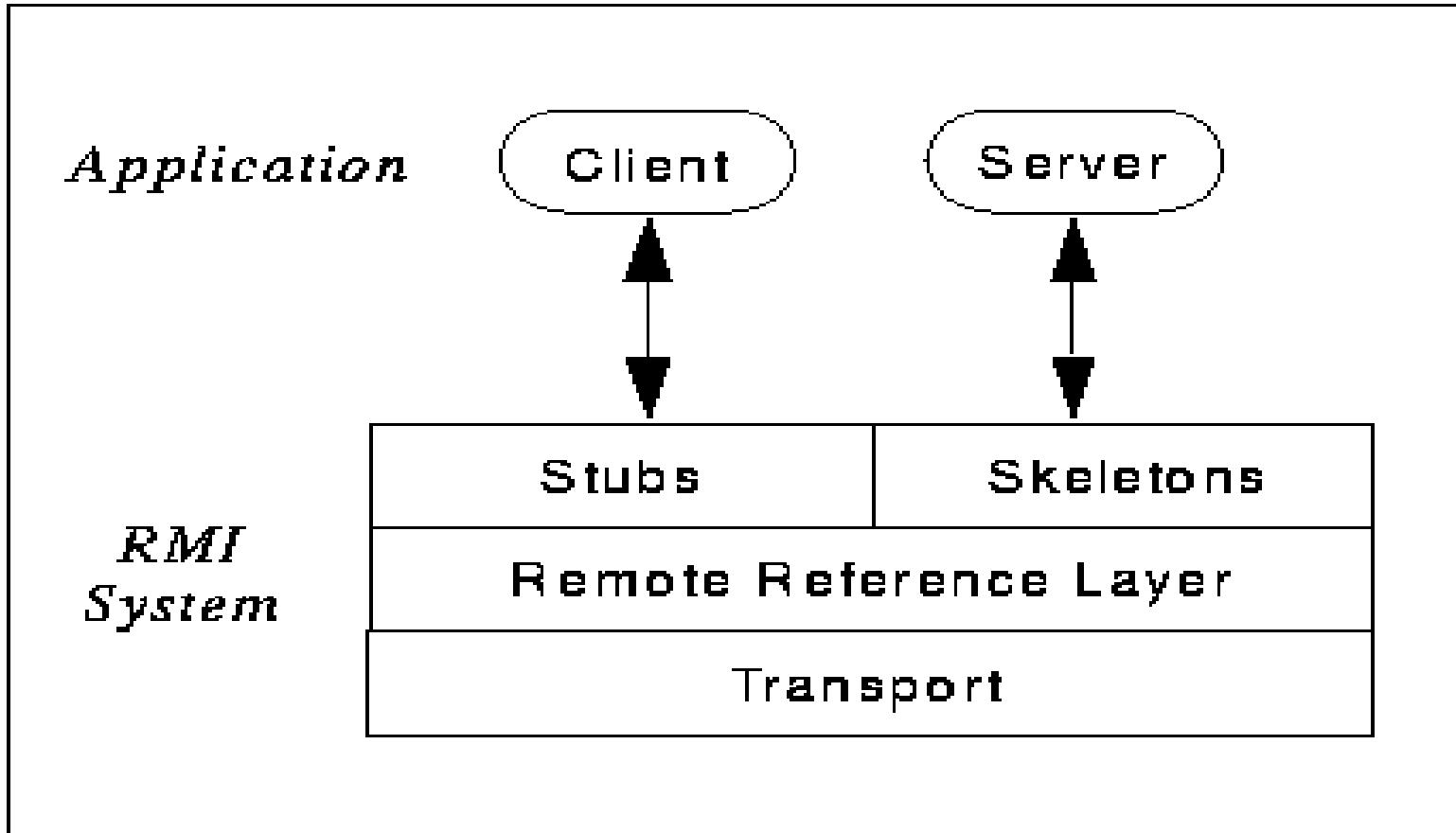
# Disadvantages of RPC

- It supports a limited set of simple data types.
- RPC is not suitable for passing and returning Java objects.
- It requires the programmer to learn a special interface definition language (IDL) to describe the functions that can be invoked remotely.

# Remote Method Invocation (RMI)

- RMI is Java's implementation of RPC for Java-object-to-Java-object distributed Communication.
- RMI does not require the programmer to learn an IDL.

# RMI Architecture



RMI architecture

# RMI Layers

- The RMI system consists of three layers
  - Stub/skeleton layer
  - Remote reference layer
  - Transport layer

# RMI Layers....

- **Stub/skeleton layer:** client-side stubs (proxies) and server-side skeletons
- **Stubs:** Marshals (writes and transmits) the parameters to the remote server and unmarshals (reads) the return value
- **Skeletons:** unmarshals (reads) the parameters for the remote method and Marshals (writes and transmits) the result (return value or exception) to the caller.

# RMI Layers....

- The purpose of the "marshalling/unmarshalling" process is to transfer data between the RMI system.
- **Marshalling** is the process of converting the data or the objects into a byte stream.
- **Unmarshalling** is the reverse of marshalling and convert the byte stream to objects.

# RMI Layers....

- **Remote reference layer:** It determine whether the server is a single object or is a replicated object that require communications with multiple locations.
- **Transport layer:** It is responsible for connection set up and management. It also track the remote object.

# 1. Simple RMI program to print Hello World

- **Hello.java** - a remote interface
- **Server.java** - a remote object implementation that implements the remote interface
- **Client.java** - a simple client that invokes a method of the remote interface
- **Remote interface:** A remote interface extends the interface `java.rmi.Remote` and declares a set of remote methods. Each remote method must throws `RemoteException`.
- **Remote Object:** A remote object is an instance of a class that implements a remote interface.

# 1. Simple RMI program to print Hello World

## Remote Interface Hello.java

```
1 import java.rmi.*;
2 public interface Hello extends Remote {
3     String sayHello() throws RemoteException;
4 }
```

# Server Skeleton

- Create and export a remote object
- Register the remote object with a Java RMI registry
- Java RMI registry:
  - The server "registers" its services in the registry.
  - The client does not need to know the location of individual servers, and does a lookup on the RMI Registry for the service it needs.

# 1. Simple RMI program to print Hello World

## HelloServer.java

```
1 import java.rmi.RemoteException;
2 import java.rmi.server.UnicastRemoteObject;
3 import java.rmi.registry.LocateRegistry;
4 import java.rmi.registry.Registry;
5 public class HelloServer implements Hello {
6     public String sayHello() {
7         return "Hello, world!";
8     }
9     public static void main(String args[]) {
10        try {
11            HelloServer obj = new HelloServer();
12            Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
13            // Bind the remote object's stub in the registry
14            Registry registry = LocateRegistry.getRegistry();
15            registry.bind("Hello", stub);
16            System.out.println("Server is Running");
17        }
18        catch (Exception e) {
19            System.out.println(e);
20        }
21    }
22}
```

# 1. Simple RMI program to print Hello World

## HelloClient.java

```
1 import java.rmi.registry.*;
2 public class HelloClient {
3     public static void main(String[] args) {
4         try {
5             Registry registry = LocateRegistry.getRegistry("localhost");
6             Hello stub = (Hello) registry.lookup("Hello");
7             String message = stub.sayHello();
8             System.out.println(message);
9         }
10        catch (Exception e) {
11            System.out.println(e);
12        }
13    }
14 }
```

# How to run the program using cmd

- Compile the code
  - `javac Hello.java HelloServer.java HelloClient.java`
- Start the Java RMI registry
  - `start rmiregistry`
- Run the server and client
  - `java HelloServer`
  - `java HelloClient`

## 2. RMI program to perform arithmetic operation

### Remote Interface Calculator.java

```
1 import java.rmi.*;
2 public interface Calculator extends Remote {
3     int Add (int a, int b) throws RemoteException;
4     int Subtract (int a, int b) throws RemoteException;
5     int Multiply(int a, int b) throws RemoteException;
6 }
```

## 2. RMI program to perform arithmetic operation

### CalculatorServer.java

```
1 import java.rmi.RemoteException;
2 import java.rmi.server.UnicastRemoteObject;
3 import java.rmi.registry.LocateRegistry;
4 import java.rmi.registry.Registry;
5 public class CalculatorServer implements Calculator {
6     public int Add( int a, int b) {
7         return a+b;};
8     public int Subtract( int a, int b) {
9         return a-b;};
10    public int Multiply( int a, int b) {
11        return a*b;};
12    public static void main(String args[]) {
13        try {
14            CalculatorServer obj = new CalculatorServer();
15            Calculator stub = (Calculator) UnicastRemoteObject.exportObject(obj, 0);
16            // Bind the remote object's stub in the registry
17            Registry registry = LocateRegistry.getRegistry();
18            registry.bind("Calculator", stub);
19            System.out.println("Server is Running");
20        }
21        catch (Exception e) {
22            System.out.println(e);
23        }
24    }
25 }
```

## 2. RMI program to perform arithmetic operation

### CalculatorClient.java

```
1 import java.rmi.registry.*;
2 public class CalculatorClient {
3     public static void main(String[] args) {
4         try {
5             Registry registry = LocateRegistry.getRegistry("localhost");
6             Calculator stub = (Calculator) registry.lookup("Calculator");
7             System.out.println("Addition " + stub.Add(10,5));
8             System.out.println("Subtraction " + stub.Subtract(10,5));
9             System.out.println("Multiply " + stub.Multiply(10,5));
10        }
11        catch (Exception e) {
12            System.out.println(e);
13        }
14    }
15 }
```

### 3. Simple chat program between Client and Server

#### Remote Interface Chat.java

```
1 import java.rmi.*;
2 public interface Chat extends Remote {
3     public void sendMessage(String text) throws RemoteException;
4     public String getMessage() throws RemoteException;
5 }
```

### 3. Simple chat program between Client and Server

#### ChatServer.java

```
1 import java.rmi.RemoteException;
2 import java.rmi.server.UnicastRemoteObject;
3 import java.rmi.registry.LocateRegistry;
4 import java.rmi.registry.Registry;
5 import java.util.*;
6 public class ChatServer implements Chat {
7     public void sendMessage(String s) throws RemoteException {
8         System.out.println("Client says " + s);
9     }
10    public String getMessage() throws RemoteException {
11        Scanner input = new Scanner(System.in);
12        System.out.println("Enter the Server message");
13        String message = input.nextLine();
14        return message;
15    }
16    public static void main(String args[]) {
17        try {
18            ChatServer obj = new ChatServer();
19            Chat stub = (Chat) UnicastRemoteObject.exportObject(obj, 0);
20            Registry registry = LocateRegistry.getRegistry();
21            registry.bind("Chat", stub);
22            System.out.println("Server is Running");
23        }
24        catch (Exception e) {
25            System.out.println(e);
26        }
27    }
28}
```

### 3. Simple chat program between Client and Server

#### ChatClient.java

```
1 import java.rmi.registry.*;
2 import java.util.*;
3 public class ChatClient {
4     public static void main(String[] args) {
5         try {
6             Registry registry = LocateRegistry.getRegistry("localhost");
7             Chat stub = (Chat) registry.lookup("Chat");
8             Scanner input = new Scanner(System.in);
9             while (true) {
10                 System.out.println("Enter the Client message");
11                 String message = input.nextLine();
12                 stub.sendMessage(message);
13                 message = stub.getMessage();
14                 System.out.println(message);
15             }
16         } catch (Exception e) {
17             System.out.println(e);
18         }
19     }
20 }
```

# Programs for Lab practice

- Creating Client Applets for HelloWorld, Calculator, Chat Client Server.
- Add permission in `java.policy` file in `C:\Program Files\Java\jdk1.8.0_25\jre\lib\security`
  - `grant codeBase "file:E:/DIT/Courses/Network-Programming/Lectures/Lecture-08/code/*" { permission java.security.AllPermission;};`
- Not included for quiz and exam

## 4. Creating Applet for “Simple RMI program to print Hello World”

### HelloApplet.java

- The code for HelloServer and Hello interface remains same
  - HelloServer.java (slide 15)
  - Hello.java (slide 13)
- Modify the code HelloClient.java (slide 16) by adding the applet.

## 4. Creating Applet for “Simple RMI program to print Hello World”

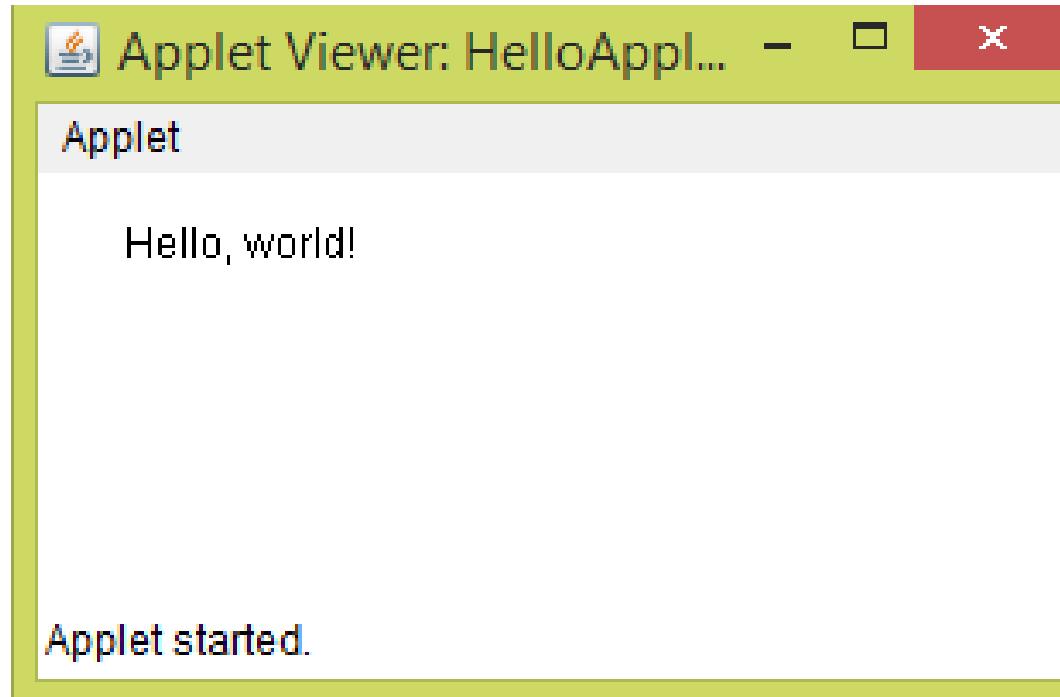
### HelloApplet.java

```
1 import java.rmi.registry.*;
2 import java.applet.Applet;
3 import java.awt.Graphics;
4 import javax.swing.*;
5 public class HelloApplet extends Applet {
6     String message = "";
7     public void init() {
8         try {
9             Registry registry = LocateRegistry.getRegistry("localhost");
10            Hello stub = (Hello) registry.lookup("Hello");
11            message = stub.sayHello();
12        }
13        catch (Exception e) {
14            System.out.println(e);
15        }
16        public void paint(Graphics g) {
17            g.drawString(message, 25, 25);
18        }
}
```

## 4. HelloApplet.html

```
1  <html>
2  <applet code="HelloApplet.class" width = "300" height = "125">
3  </applet>
4  </html>
```

## 4. Output of HelloApplet.html



## 5. Creating Applet for “RMI program to perform arithmetic operation”

### CalculatorApplet.java

- The code for CalculatorServer and Calculator interface remains same
  - CalculatorServer.java (slide 19)
  - Calculator.java (slide 18)
- Modify the code CalculatorClient.java ( slide 20) by adding the applet.

## 5. Creating Applet for “RMI program to perform arithmetic operation”

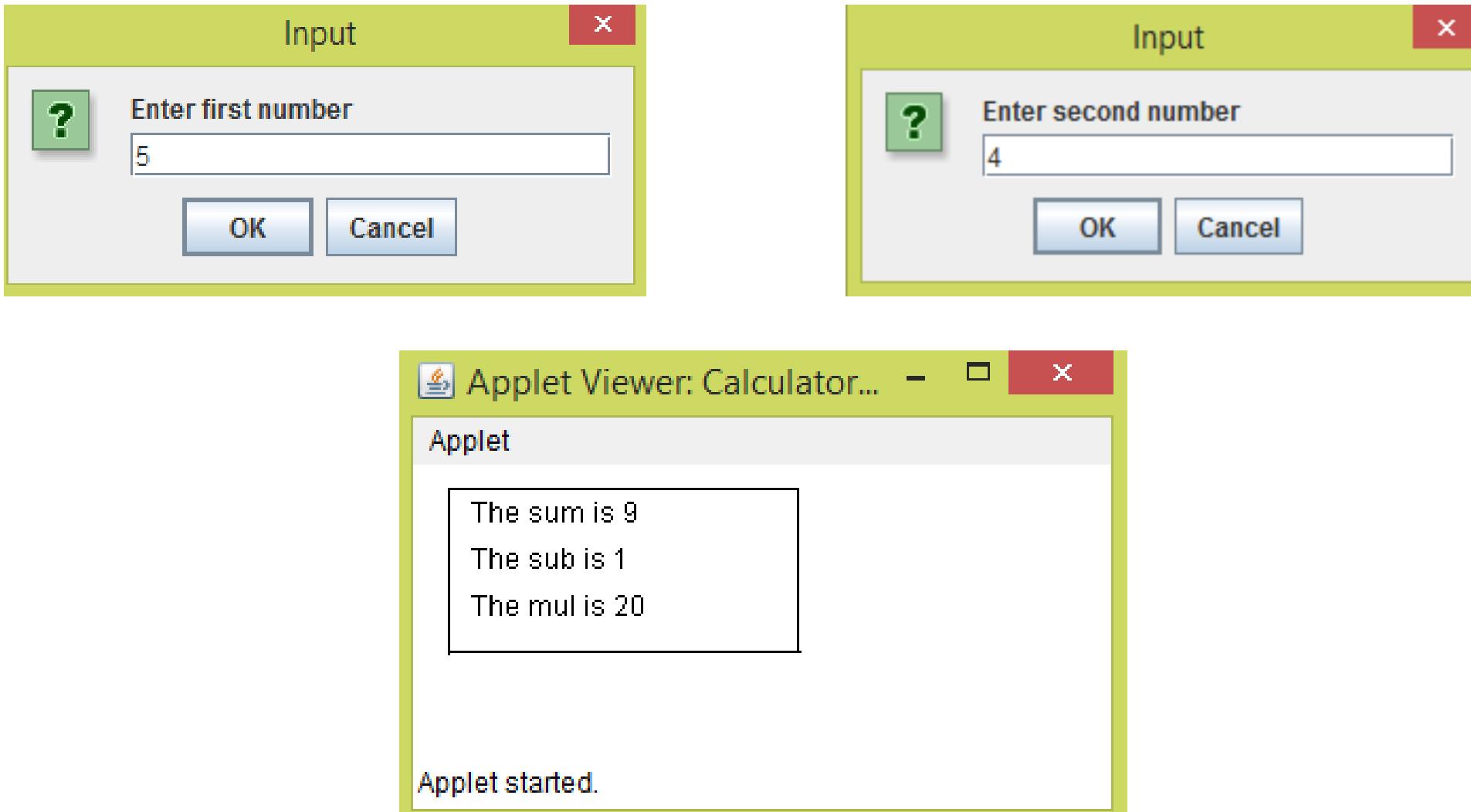
### CalculatorApplet.java

```
1 import java.rmi.registry.*;
2 import java.awt.Graphics;
3 import javax.swing.*;
4 public class CalculatorApplet extends JApplet {
5     int sum,sub,mul;
6     public void init() {
7         try {
8             Registry registry = LocateRegistry.getRegistry("localhost");
9             Calculator stub = (Calculator) registry.lookup("Calculator");
10            String firstNumber; // first string entered by user
11            String secondNumber; // second string entered by user
12            int number1; // first number to add,sub and mul
13            int number2; // second number to add,sub and mul
14            firstNumber = JOptionPane.showInputDialog("Enter first number");
15            secondNumber = JOptionPane.showInputDialog("Enter second number");
16            number1 = Integer.parseInt(firstNumber );
17            number2 = Integer.parseInt(secondNumber );
18            sum = stub.Add(number1,number2);
19            sub = stub.Subtract(number1,number2);
20            mul = stub.Multiply(number1,number2);
21        }
22        catch (Exception e) {
23            System.out.println(e);
24        }
25        public void paint( Graphics g ){
26            g.drawRect( 15, 10, 150, 70 );
27            g.drawString( "The sum is " + sum, 25, 25 );
28            g.drawString( "The sub is " + sub, 25, 45 );
29            g.drawString( "The mul is " + mul, 25, 65 );
30        }
    }
```

# 5. CalculatorApplet.html

```
1 <html>
2   <applet code="CalculatorApplet.class" width = "300" height = "125">
3     </applet>
4   </html>
```

## 5. Output of CalculatorApplet.html



## 6. Creating Applet for “Simple chat program between Client and Server”

### ChatApplet.java

- The code for ChatServer and Chat interface remains same
  - ChatServer.java (slide 22)
  - Chat.java (slide 21)
- Modify the code ChatClient.java ( slide 23) by adding the applet.

## 6. Creating Applet for “Simple chat program between Client and Server”

### ChatApplet.java

```
1 import java.rmi.registry.*;
2 import java.applet.Applet;
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 public class ChatApplet extends Applet implements ActionListener{
7     Chat stub;
8     TextField text;
9     Label label;
10    public void init() {
11        try {
12            Button button = new Button("Send Data to Server");
13            text = new TextField(30);
14            label = new Label("Server Data will print here");
15            Registry registry = LocateRegistry.getRegistry("localhost");
16            stub = (Chat) registry.lookup("Chat");
17            add(button);
18            add(text);
19            add(label);
20            button.addActionListener(this);
21        catch (Exception e) {System.out.println(e);}
22        public void actionPerformed(ActionEvent event) {
23            try {
24                stub.sendMessage(text.getText());
25                label.setText(stub.getMessage());
26            catch (Exception e) {
27                System.out.println(e);
28            }
29        }
30    }
31}
```

# 6. ChatApplet.html

```
1  <html>
2   <applet code="ChatApplet.class" width = "300" height = "125">
3   </applet>
4 </html>
```

## 6. Output of ChatApplet.html

C:\Windows\System32\cmd.exe - java ChatServer  
E:\DIT\Courses\Network-Programming\Lectures\Lecture-08\code>start rmiregistry.exe  
E:\DIT\Courses\Network-Programming\Lectures\Lecture-08\code>java ChatServer  
Server is Running  
Client says Hello how are you  
Enter the Server message  
I am fine

Applet Viewer: ChatAppl... - □ ×  
Applet  
Send Data to Server  
Hello how are you  
I am fine  
Applet started.

# Lecture 09

# Server Software Design

CMPU 3027  
Network Programming

# Statefull and Stateless server

- State information: Information maintained by the server about the status of ongoing interactions with clients.
- Servers that do not keep any state information are called stateless servers,
- otherwise, called stateful servers.

# Example of a File Server

- Server program awaits request from a client
  - Extract data from a specified file
  - Store data in a specified file
- Server carries out request and replies to client

# Example of Stateless File Server

- **If Stateless each read request must contain**
  - complete filename
  - position in file from where data is to be read
  - number of bytes of data to be read
- **Each write request must contain**
  - complete filename
  - position in file from where data is to be stored
  - data to be written

# Example of Stateful File Server

Handle	File Name	Current Position
1	Program1.c	0
2	lecture.doc	120
3	sample.txt	25
4	sample2.c	1000

Table of State Information

# Example of Stateful File Server

- If the server maintains state information it creates a state table
- When client requests server to open a file, server
  - adds entry to state table containing filename
  - adds a handle, an integer to identify file
  - adds a record pointer initialised to zero
  - sends handle back to client

# Example of Stateful File Server

- When client wants data
  - It sends short request including the handle
  - Server looks up handle and record pointer in state table
  - Server updates pointer
- When client is finished it informs server which deletes entry in table

# Iterative Connection- Oriented Server Algorithm (TCP server)

1. **ServerSocket** creates the socket on specified port and bind it to the local IP address.
2. **Accept** function call obtains the next incoming connection request and returns a socket descriptor for the new connection
3. **readObject** is used to read the client requests
4. **writeObject** is used to send replies
5. **Close** is used to release the socket when the server finishes with a connection

## Iterative Connectionless Server Algorithm (UDP Server)

1. Create a **datagram socket** and bind to the well-known address for the service offered
2. Repeatedly **read** next request from client, formulate response, **send** reply back to client

# Concurrent Connection-Oriented Server Algorithm (TCP server with Multithreading)

1. **ServerSocket** creates the socket on specified port and bind it to the local IP address.
2. Repeatedly call **accept** to receive next request from a client and create new thread to handle response
3. Thread - Receive the **connection request**
4. Thread - **read** client requests, **send** replies
5. Thread - **Close** connection and exit

## Concurrent Connection less Server Algorithm (UDP server with Multithreading)

1. Create a **datagram socket** and bind to the well-known address for the service offered
2. Repeatedly **read** next request from client, create new **thread** to handle response

# Java Servlets

- A servlet extends the functionality of a server.
- Servlets are effective for developing Web-based solutions.
- It provide secure access to a Web site and interact with databases on behalf of a client.

# Java Servlets....

- The servlets provide the communication between clients and servers via the HTTP protocol.
- A client sends an HTTP request to the servlet. The servlet container receives the request and does its processing, which may include interacting with a database.
- The servlet returns its results to the client—normally in the form of an HTML or XML.

# Tomcat Setting

- Download Tomcat 9.0.0.M13
  - <http://tomcat.apache.org/download-90.cgi>
- Setting Environmental Variables
  - JAVA\_HOME
    - C:\Program Files\Java\jdk1.8.0\_25\bin
  - JRE\_HOME
    - C:\Program Files\Java\jre1.8.0\_25
  - CLASSPATH
    - C:\tomcat9\lib\servlet-api.jar;.;

# Tomcat Setting...

- Open window cmd
  - `cd c:\tomcat9\bin`
- Start the tomcat
  - `startup`
- Stop the tomcat
  - `shutdown`
- Open the browser and type
  - <http://localhost:8080/>

# Tomcat is Running

The screenshot shows a web browser window with the following details:

- Address Bar:** Shows "Apache Tomcat/9.0.0.M13" and "localhost:8080".
- Toolbar:** Includes standard links like File, Edit, View, History, Bookmarks, Tools, and Help.
- Header:** Shows the Apache logo and the text "Apache Tomcat/9.0.0.M13" next to a close button.
- Search Bar:** Contains a search icon and the word "Search".
- Navigation Buttons:** Back, Forward, Stop, and Refresh.
- Page Content:**
  - Header:** "Apache Tomcat/9.0.0.M13"
  - Logo:** The Apache feather logo.
  - Text:** "The Apache Software Foundation" and "http://www.apache.org/".
  - Message:** "If you're seeing this, you've successfully installed Tomcat. Congratulations!"
  - Image:** A cartoon cat icon with a TM symbol.
  - Text:** "Recommended Reading:" followed by three links:
    - [Security Considerations HOW-TO](#)
    - [Manager Application HOW-TO](#)
    - [Clustering/Session Replication HOW-TO](#)
  - Buttons:** "Server Status", "Manager App", and "Host Manager".
  - Footer:** "Developer Quick Start" section with links:
    - [Tomcat Setup](#)
    - [First Web Application](#)
    - [Realms & AAA](#)
    - [JDBC Data Sources](#)
    - [Examples](#)
    - [Servlet Specifications](#)
    - [Tomcat Versions](#)

# 1. Write a program to print “Hello World” using Java Servlet.

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4 public class HelloWorld extends HttpServlet {
5     public void doGet(HttpServletRequest request, HttpServletResponse response)
6         throws ServletException, IOException {
7     response.setContentType("text/html");
8     PrintWriter out = response.getWriter();
9     out.println("<html>");
10    out.println("<body>");
11    out.println("<h1>Hello World!</h1>");
12    out.println("</body>");
13    out.println("</html>");
14 }
15 }
```

# How to run the java servlet

1. Compile program
  - javac HelloWorld.java
2. Copy HelloWorld.class file into C:\tomcat9\webapps\ROOT\WEB-INF\classes
  - copy HelloWorld.class c:\tomcat9\webapps\ROOT\WEB-INF\classes\
3. Create following entries in web.xml file located at C:\tomcat9\webapps\ROOT\WEB-INF

```
<servlet>
  <servlet-name>HelloWorld</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>HelloWorld</servlet-name>
  <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

# HelloWorld.html



**Click the button to invoke the servlet**

[Get HTML Document](#)

```
1 <html>
2   <body>
3     <h1>Click the button to invoke the servlet</h1>
4     <form action="http://localhost:8080>HelloWorld" method="get">
5       <input type="submit" value="Get HTML Document">
6     </form>
7   </body>
8 </html>
```

# Output of program through web browser



**Hello World!**

## 2. Write a program to pass parameter to server using Java Servlet.

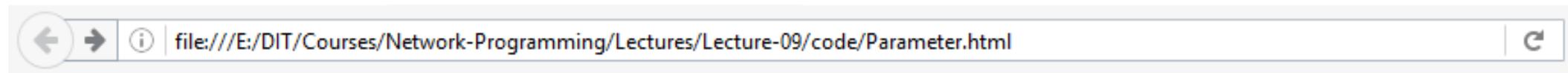
```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4 public class Parameter extends HttpServlet {
5     public void doGet(HttpServletRequest request, HttpServletResponse response)
6         throws ServletException, IOException {
7         response.setContentType("text/html");
8         PrintWriter out = response.getWriter();
9         out.println("<html>");
10        out.println("<body>");
11        out.println("<h1> Hello \t" + request.getParameter("firstname") );
12        out.println( "Welcome to Servlets! </h1>" );
13        out.println("</body>");
14        out.println("</html>");
15    }
}
```

# How to run the java servlet

1. Compile program
  - javac Parameter.java
2. Copy Parameter.class file into C:\tomcat9\webapps\ROOT\WEB-INF\classes
  - copy Parameter.class c:\tomcat9\webapps\ROOT\WEB-INF\classes\
3. Create following entries in web.xml file located at C:\tomcat9\webapps\ROOT\WEB-INF

```
<servlet>
  <servlet-name> Parameter </servlet-name>
  <servlet-class> Parameter </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name> Parameter </servlet-name>
  <url-pattern>/Parameter </url-pattern>
</servlet-mapping>
```

# Parameter.html



Type your first name and press the Submit button

 submit

```
1 <html>
2   <body>
3     <h1>Type your first name and press the Submit button</h1>
4     <form action="http://localhost:8080/Parameter" method=get>
5       <input type = "text" name = "firstname" />
6       <input type="submit" value="submit">
7     </form>
8   </body>
9 </html>
```

# Output of program through web browser



**Hello aneel Welcome to Servlets!**

### 3. Write a program to pass parameter to Servlet using post method

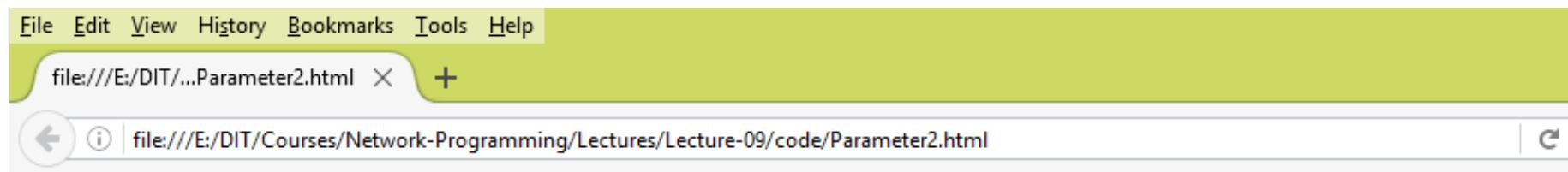
```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4 public class Parameter2 extends HttpServlet {
5     public void doPost(HttpServletRequest request, HttpServletResponse response)
6         throws ServletException, IOException {
7         response.setContentType("text/html");
8         PrintWriter out = response.getWriter();
9         out.println("<html>");
10        out.println("<body>");
11        out.println("<h1> Hello \t" + request.getParameter("firstname"));
12        out.println( "Welcome to Servlets! </h1>" );
13        out.println("</body>");
14        out.println("</html>");
15    }
}
```

# How to run the java servlet

1. Compile program
  - `javac Parameter2.java`
2. Copy `Parameter.class` file into `C:\tomcat9\webapps\ROOT\WEB-INF\classes`
  - `copy Parameter2.class c:\tomcat9\webapps\ROOT\WEB-INF\classes\`
3. Create following entries in `web.xml` file located at `C:\tomcat9\webapps\ROOT\WEB-INF`

```
<servlet>
  <servlet-name> Parameter2</servlet-name>
  <servlet-class> Parameter2</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name> Parameter2 </servlet-name>
  <url-pattern>/Parameter2 </url-pattern>
</servlet-mapping>
```

# Parameter2.html



Type your first name and press the Submit button

 submit

```
1 <html>
2   <body>
3     <h1>Type your first name and press the Submit button</h1>
4     <form action="http://localhost:8080/Parameter2" method=post>
5       <input type = "text" name = "firstname" />
6       <input type="submit" value="submit">
7     </form>
8   </body>
9 </html>
```

# Output of program through web browser

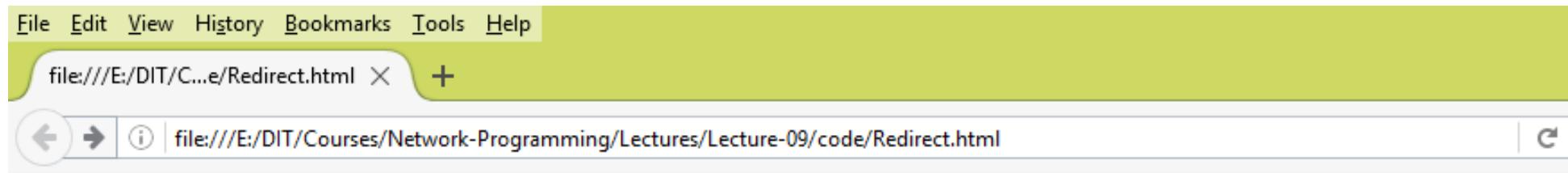


**Hello aneel Welcome to Servlets!**

#### 4. Write a program to redirect from one page to another.

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4 import java.util.*;
5 public class Redirect extends HttpServlet {
6     public void doGet(HttpServletRequest request, HttpServletResponse response)
7         throws ServletException, IOException {
8         String location = request.getParameter( "page" );
9         if ( location != null )
10            if ( location.equals( "dit" ) )
11                response.sendRedirect( "http://www.dit.ie" );
12            else
13                if ( location.equals( "HelloWorld" ) )
14                    response.sendRedirect( "HelloWorld" );
15    }
}
```

# Redirect.html



The screenshot shows a web browser window with the following details:

- Menu Bar:** File Edit View History Bookmarks Tools Help
- Title Bar:** file:///E:/DIT/C...e/Redirect.html X +
- Address Bar:** file:///E:/DIT/Courses/Network-Programming/Lectures/Lecture-09/code/Redirect.html
- Content Area:** Click a link to be redirected to the appropriate page
- Links:** [www.dit.ie](#) and [Hello World](#)

**HTML Source Code:**

```
1 <html>
2   <body>
3     <p>Click a link to be redirected to the appropriate page</p>
4     <p><a href = "http://localhost:8080/Redirect?page=dit"> www.dit.ie</a><br/>
5       <a href = "http://localhost:8080/Redirect?page=HelloWorld"> Hello World</a>
6     </p>
7   </body>
8 </html>
```

# Output of program through web browser

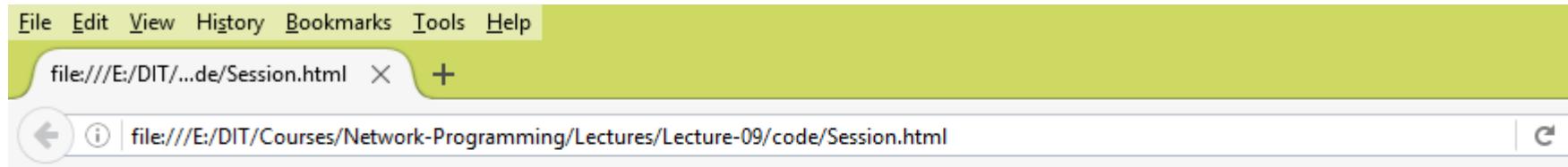


**Hello World!**

## 5. Write a program to display the session information.

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4 import java.util.*;
5 public class Session extends HttpServlet {
6     public void doGet(HttpServletRequest request, HttpServletResponse response)
7         throws ServletException, IOException {
8         response.setContentType("text/html");
9         PrintWriter out = response.getWriter();
10        HttpSession session = request.getSession(true);
11        // print session info
12        Date created = new Date(session.getCreationTime());
13        Date accessed = new Date(session.getLastAccessedTime());
14        out.println("Session ID " + session.getId());
15        out.println("Session Created: " + created);
16        out.println("Session Last Accessed: " + accessed);
17    }
}
```

# Session.html

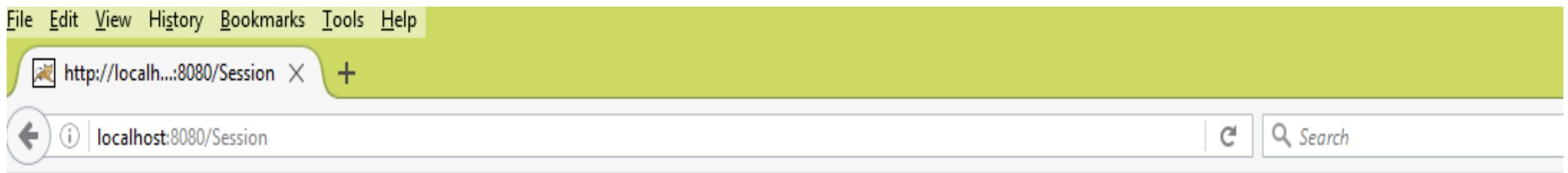


**Click the button to get servlet session information**

**Get Session Info**

```
1 <html>
2   <body>
3     <h1>Click the button to get servlet session information</h1>
4     <form action="http://localhost:8080/Session" method=get>
5       <input type="submit" value="Get Session Info">
6     </form>
7   </body>
8 </html>
```

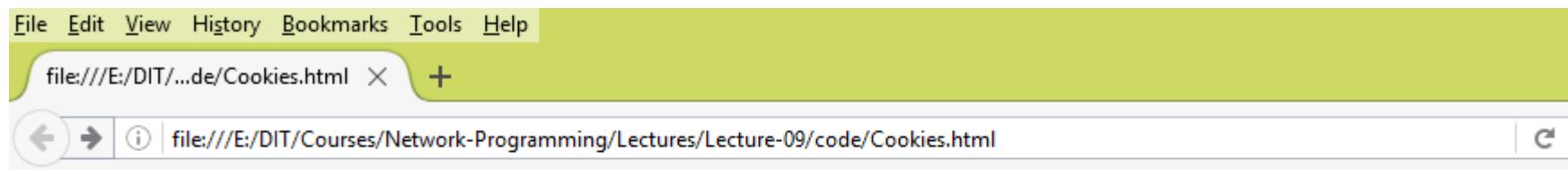
# Output of program through web browser



## 6. Write a program to create cookies.

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4 public class Cookies extends HttpServlet {
5     public void doPost(HttpServletRequest request, HttpServletResponse response)
6         throws ServletException, IOException {
7         response.setContentType("text/html");
8         PrintWriter out = response.getWriter();
9         String language = request.getParameter( "language" );
10        Cookie cookie = new Cookie( "Book", language );
11        cookie.setMaxAge(60*60);
12        response.addCookie( cookie );
13        out.println("<html>");
14        out.println("<body>");
15        out.println( "<h1> Cookie is created with name Book and value is " );
16        out.println(language+"</h1>" );
17        out.println("</body>");
18        out.println("</html>");
19    }
}
```

# Session.html



## Select a programming language:

- C
  - Java
  - Python
- 

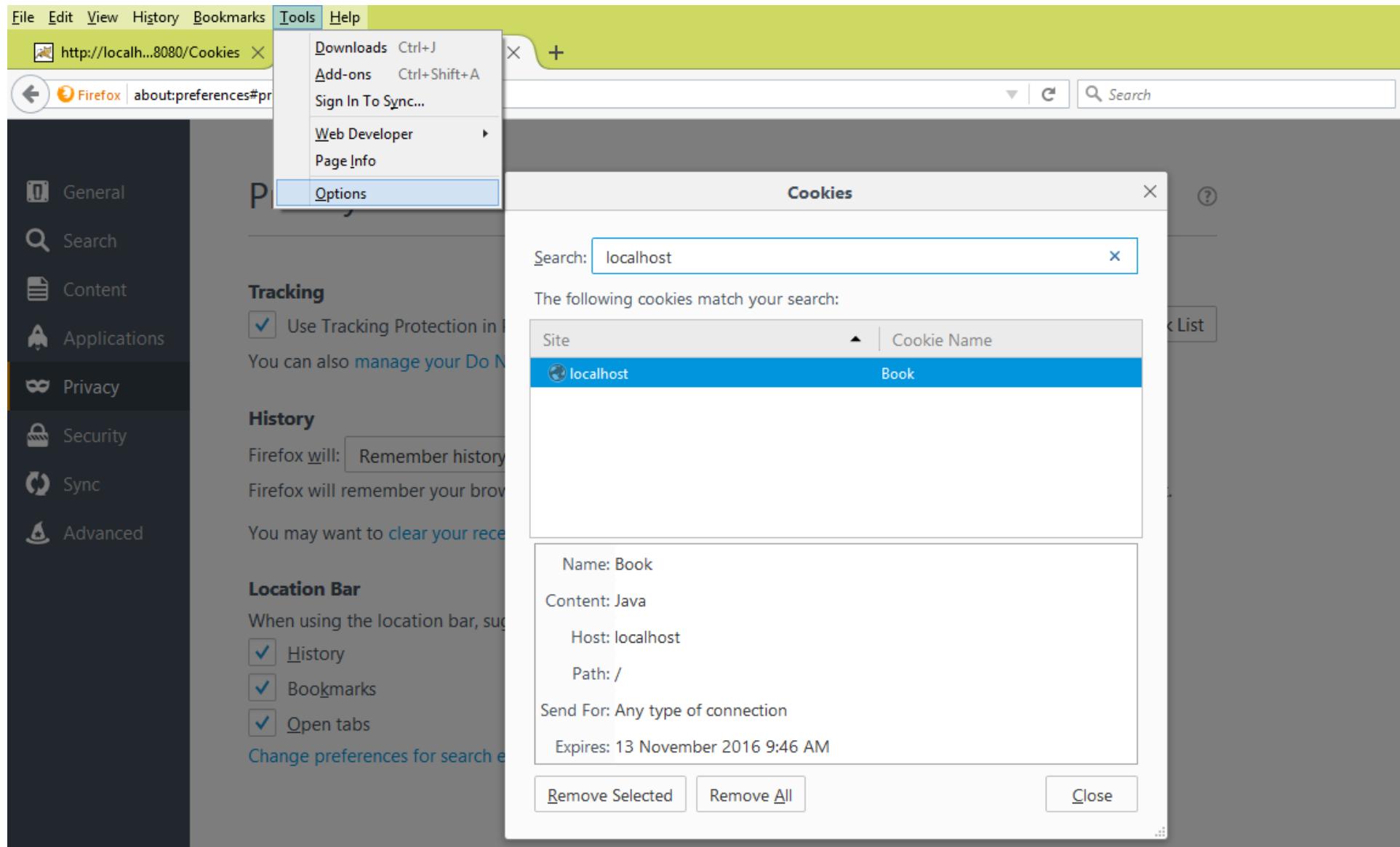
```
1 <html>
2   <body>
3     <h1>Select a programming language:</h1>
4     <form action="http://localhost:8080/Cookies" method="post">
5       <input type = "radio" name = "language" value = "C" />C <br />
6       <input type = "radio" name = "language" value = "Java" />Java <br />
7       <input type = "radio" name = "language" value = "Python" />Python <br />
8       <input type="submit" value="submit">
9     </form>
10   </body>
11 </html>
```

# Output of program through web browser



**Cookie is created with name Book and value is Java**

# Cookies in Firefox



# Lecture 10

## Multicast Sockets & Secure Sockets

CMPU 3027  
Network Programming

# Multicast

- Multicast means sending data from one host to many different hosts, but not to everyone.
- Data only goes to clients who joined the particular multicast group.
- IP range is 224.0.0.0 to 239.255.255.255

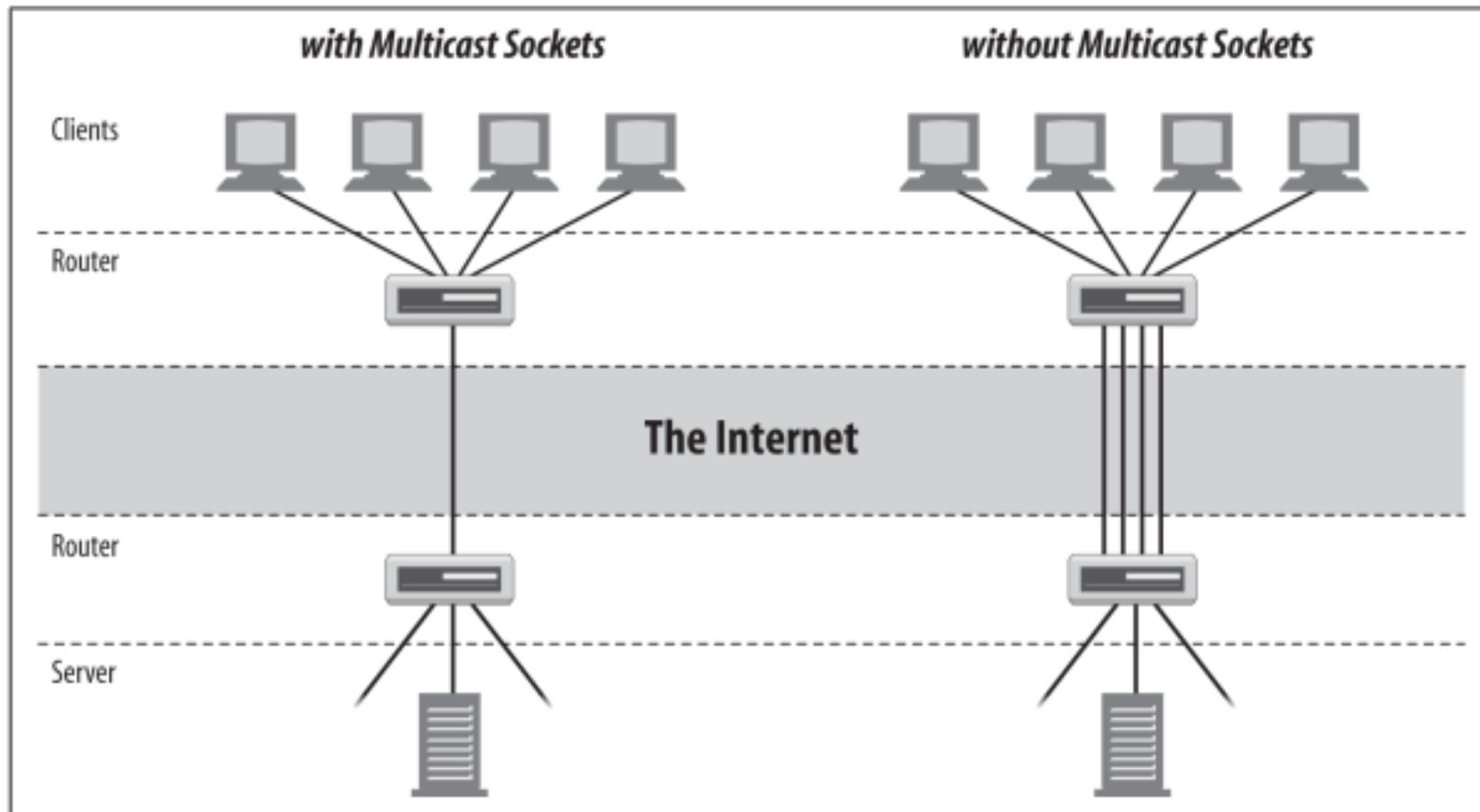
# Multicast Group

- A multicast group is a set of Internet hosts that share a multicast address.
- Any data sent to the multicast address is relayed to all the members of the group.
- Membership in a multicast group is open; hosts can enter or leave the group at any time.
-

# Communicating with a Multicast Group

- When a host wants to send data to a multicast group, it puts that data in multicast datagrams, which are nothing more than UDP datagrams.
- Multicast data is sent via UDP, which, though unreliable.
- The biggest restriction on multicasting is the availability of special multicast routers.

# Communicating with a Multicast Group



# Communicating with a Multicast Group

- Multicast Socket is used for multicast group communication, it can perform four key operations:
  1. Join a multicast group.
  2. Send data to the members of the group.
  3. Receive data from the group.
  4. Leave the multicast group.

# Step 1. Joining Multicast Group

```
1 import java.net.*;
2 import java.io.*;
3 public class JoinMulticastGroup {
4     public static void main(String[] argv) throws Exception {
5         try {
6             String groupName = "224.2.2.2";
7             int port = 4000;
8             MulticastSocket msocket = new MulticastSocket(port);
9             System.out.println("Creating MulticastSocket at 4000");
10            InetAddress group = InetAddress.getByName(groupName);
11            System.out.println("Joining MulticastGroup at 224.2.2.2");
12            msocket.joinGroup(group);
13        }
14        catch (Exception ex) {
15            System.err.println(ex);
16        }
    }
```

## Step 2. Sending Data to Multicast Group

```
1 import java.net.*;
2 import java.io.*;
3 public class SendData {
4     public static void main(String[] argv) throws Exception {
5         try {
6             String groupName = "224.2.2.2";
7             int port = 4000;
8             MulticastSocket msocket = new MulticastSocket(port);
9             System.out.println("Creating MulticastSocket at 4000");
10            InetSocketAddress group = InetSocketAddress.getByName(groupName);
11            System.out.println("Joining MulticastGroup at 224.2.2.2");
12            msocket.joinGroup(group);
13            byte[] data = new byte[1024];
14            DatagramPacket dp = new DatagramPacket(data, data.length, group, port);
15            System.out.println("Sending Data To Multicast");
16            msocket.send(dp);
17        }
18        catch (Exception ex) {
19            System.err.println(ex);
20        }
21    }
22 }
```

## Step 3. Receiving Data from Multicast Group

```
1 import java.net.*;
2 import java.io.*;
3 public class ReceiveData {
4     public static void main(String[] argv) throws Exception {
5         try {
6             String groupName = "224.2.2.2";
7             int port = 4000;
8             MulticastSocket msocket = new MulticastSocket(port);
9             System.out.println("Creating MulticastSocket at 4000");
10            InetAddress group = InetAddress.getByName(groupName);
11            System.out.println("Joining MulticastGroup at 224.2.2.2");
12            msocket.joinGroup(group);
13            byte[] data = new byte[1024];
14            DatagramPacket dp = new DatagramPacket(data, data.length, group, port);
15            System.out.println("Waiting for Data from MulticastGroup....");
16            msocket.receive(dp);
17            System.out.println("Received Data from MulticastGroup");
18        }
19        catch (Exception ex) {
20            System.err.println(ex);
21        }
    }
```

## Step 4. Leaving the Multicast Group

```
1 import java.net.*;
2 import java.io.*;
3 public class Receiver {
4     public static void main( String[] args ) {
5         try {
6             MulticastSocket ms = new MulticastSocket(4000);
7             InetAddress group = InetAddress.getByName("224.2.2.2");
8             System.out.println("Joining the MulticastGroup");
9             ms.joinGroup(group);
10            byte[] buffer = new byte[100];
11            DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
12            System.out.println("Waiting for Server Message");
13            ms.receive(dp);
14            System.out.println("Getting Message from Server " + new String (dp.getData()));
15            System.out.println("Leaving the MulticastGroup");
16            ms.leaveGroup(group);
17            ms.close();
18        }
19        catch (Exception ex) {
20            System.err.println(ex);
21        }
    }
```

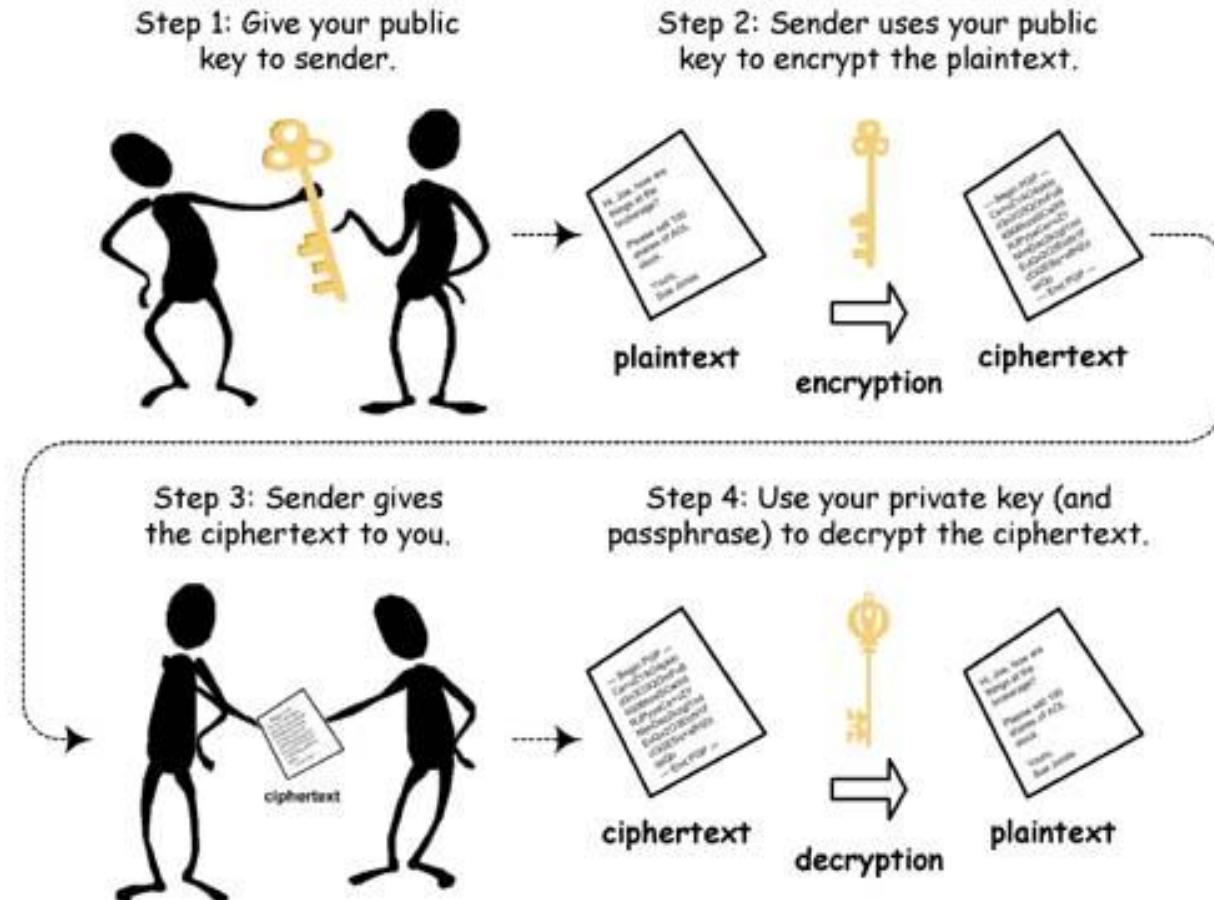
## Step 4. Leaving the Multicast Group

```
1 import java.net.*;
2 import java.io.*;
3 import java.util.Scanner;
4 public class Sender {
5     public static void main( String[] args ){
6         try {
7             InetAddress ia = InetAddress.getByName("224.2.2.2");
8             int port = 4000;
9             MulticastSocket ms = new MulticastSocket(5000);
10            Scanner input = new Scanner( System.in );
11            System.out.println("Enter message for Clients ");
12            String message = input.nextLine();
13            byte[] data= message.getBytes();
14            DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
15            System.out.println("Sending message for Clients ");
16            ms.send(dp);
17        }
18        catch (Exception ex) {
19            System.err.println(ex);
20        }
    }
```

# Secure Sockets

- Data that travels across a network can easily be accessed by someone who is not the intended recipient.
- When the data includes private information, such as passwords and credit card numbers, steps must be taken to make the data secure.
- SSL uses a combination of cryptographic processes to provide secure communication over a network

# Client Server using Public Key Encryption



## 5. Simple SSL Client Sending Data

```
1 import java.net.*;
2 import java.io.*;
3 import javax.net.ssl.SSLSocketFactory;
4 public class SSLSimpleClient {
5     public static void main(String[] args) {
6         try {
7             SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault();
8             Socket socket = factory.createSocket("www.usps.com", 443);
9             System.out.println("SSL Socket created with www.usps.com port 443");
10            Writer out = new OutputStreamWriter(socket.getOutputStream(), "US-ASCII");
11            System.out.println("Sending Credit Card Details...");
12            out.write("Name: Aneel Rahim\r\n");
13            out.write("Product-ID: 67X-89\r\n");
14            out.write("Address: DIT kevin Street Dublin Ireland\r\n");
15            out.write("Card number: 4000-1234-5678-9017\r\n");
16            out.write("Expires: 08/05\r\n");
17            out.flush();
18        }
19        catch (Exception ex) {
20            System.err.println(ex);
21        }
22    }
23}
```

## 6. Simple SSL Client Receiving Data

```
1 import java.net.*;
2 import java.io.*;
3 import javax.net.ssl.SSLSocketFactory;
4 public class SSLSimpleClient2 {
5     public static void main(String[] args) {
6         try {
7             SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault();
8             Socket socket = factory.createSocket("www.usps.com", 443); // default https port
9             Writer out = new OutputStreamWriter(socket.getOutputStream(), "UTF-8");
10            out.write("GET http://www.usps.com/ HTTP/1.1 \r\n");
11            out.write("Host:www.usps.com\r\n");
12            out.write("\r\n");
13            out.flush();
14            out.flush();
15            BufferedReader in = new BufferedReader(
16                new InputStreamReader(socket.getInputStream()));
17            String s;
18            while (!(s = in.readLine()).equals("")) {
19                System.out.println(s);
20            }
21            catch (Exception ex) {
22                System.err.println(ex);
23            }
24        }
25    }
26}
```

## 7. Create SSL Client Server program.

```
1 import java.net.*;
2 import java.io.*;
3 import javax.net.ssl.*;
4 public class SSLSocketServer {
5     public static void main(String[] args) {
6         try {
7             SSLServerSocketFactory factory = (SSLServerSocketFactory)
8                 SSLServerSocketFactory.getDefault();
9             ServerSocket server = factory.createServerSocket(5432);
10            while (true) {
11                Socket client = server.accept();
12                ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());
13                out.writeObject("Hi");
14                out.close();
15                client.close();
16            }
17        } catch (Exception ex) {
18            System.err.println(ex);
19        }
    }
```

## 7. Create SSL Client Server program....

```
1 import java.net.*;
2 import java.io.*;
3 import javax.net.ssl.*;
4 public class SSLSocketClient {
5     public static void main(String[] args) {
6         try {
7             SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault();
8             Socket client = factory.createSocket("localhost", 5432);
9             ObjectInputStream in = new ObjectInputStream(client.getInputStream());
10            String message = (String) in.readObject();
11            System.out.println ("Data from Server: " + message);
12            client.close();
13        }
14        catch (Exception ex) {
15            System.err.println(ex);
16        }
17    }
18 }
```

# How to compile SSL client server program

Simple Method to compile the code

1. keytool -genkey -keystore SSLStore.jks -alias SSLCertificate
  
2. java -Djavax.net.ssl.keyStore=SSLStore.jks -  
    Djavax.net.ssl.keyStorePassword=aneel123 SSLSocketServer
  
3. java -Djavax.net.ssl.trustStore=SSLStore.jks -  
    Djavax.net.ssl.trustStorePassword=aneel123 SSLSocketClient

# How to compile SSL client server program

```
C:\Windows\System32\cmd.exe
E:\DIT\Courses\Network-Programming\Lectures\Lecture-10\code>keytool -genkey -keystore SSLStore -alias SSLCertificate
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: aneel rahim
What is the name of your organizational unit?
[Unknown]: school of computing
What is the name of your organization?
[Unknown]: dit
What is the name of your City or Locality?
[Unknown]: dublin
What is the name of your State or Province?
[Unknown]: leinster
What is the two-letter country code for this unit?
[Unknown]: ie
Is CN=aneel rahim, OU=school of computing, O=dit, L=dublin, ST=leinster, C=ie correct?
[no]: yes

Enter key password for <SSLCertificate>
  (RETURN if same as keystore password):
E:\DIT\Courses\Network-Programming\Lectures\Lecture-10\code>
```

# How to compile SSL client server program

## Second Method to compile the code

1. keytool -genkey -keystore server-keystore.jks -alias server-key -keyalg RSA -keysize 512
2. keytool -export -alias server-key -keystore server-keystore.jks -rfc -file server.cer
3. keytool -genkey -alias client-key -keystore client-truststore.jks
4. keytool -import -alias server-key -file server.cer -keystore client-truststore.jks
  
5. java -Djavax.net.ssl.keyStore=server-keystore.jks -Djavax.net.ssl.keyStorePassword=server  
SSLocketServer
6. java -Djavax.net.ssl.trustStore=client-trustStore.jks -Djavax.net.ssl.trustStorePassword=client  
SSLocketClient

## 8. Print SSL Session information

```
1 import java.net.*;
2 import java.io.*;
3 import javax.net.ssl.*;
4 public class SSLSessionServer {
5     public static void main(String[] args) {
6         System.setProperty("javax.net.ssl.keyStore", "server-keystore.jks");
7         System.setProperty("javax.net.ssl.keyStorePassword", "server");
8         try {
9             SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault();
10            ServerSocket server = factory.createServerSocket(5432);
11            Socket client = server.accept();
12            SSLSession session = ((SSLSocket) client).getSession();
13            System.out.println("Peer host is " + session.getPeerHost());
14            System.out.println("Cipher is " + session.getCipherSuite());
15            System.out.println("Protocol is " + session.getProtocol());
16            System.out.println("ID is " + session.getId());
17            System.out.println("Session created in " + session.getCreationTime());
18            System.out.println("Session accessed in " + session.getLastAccessedTime());
19            ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());
20            out.writeObject("Hi");
21            out.close();
22            client.close();
23        }
24        catch (Exception ex) {
25            System.err.println(ex);
26        }
    }
```

## 8. Print SSL Session information

```
1 import java.net.*;
2 import java.io.*;
3 import javax.net.ssl.*;
4 public class SSLSessionClient {
5     public static void main(String[] args) {
6         System.setProperty("javax.net.ssl.trustStore", "client-trustStore.jks");
7         try {
8             SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault();
9             Socket client = factory.createSocket("localhost", 5432);
10            SSLSession session = ((SSLSocket) client).getSession();
11            System.out.println("Peer host is " + session.getPeerHost());
12            System.out.println("Cipher is " + session.getCipherSuite());
13            System.out.println("Protocol is " + session.getProtocol());
14            System.out.println("ID is " + session.getId());
15            System.out.println("Session created in " + session.getCreationTime());
16            System.out.println("Session accessed in " + session.getLastAccessedTime());
17            ObjectInputStream in = new ObjectInputStream(client.getInputStream());
18            String message = (String) in.readObject();
19            System.out.println ("Data from Server: " + message);
20            client.close();
21        }
22        catch (Exception ex) {
23            System.err.println(ex);
24        }
25    }
26}
```