Why Netowk?
- communication
- Access to a large amount of dynamic data
- dynamic data
- ability to distribute data

uses
- to communicate with other programming running in outer OS (or System) which may be written in other language beside java.

Overview of Networing
- TCP
    Transmission Control Protocol
    Reliable communication (error Control)
-UDP
    User Datagram Protocol
    Faster, unreliable communications (no error Control)

- Information is transferred in packets
ports are virtual network connections (0 to 65535)
    <1024 reserved.

- Reserved port numbers and services

ftp 20,21
Telnet 23
SMTP 25
HTTP 80
POP3 110
IMAP 143
SNMP 161

Making a Connection
- Sender must know:
    - Address of remote machine
    - Port number
    - Type and structure of data to be transferred
- Remote machine can determine:
    - Senders address and port

Networking Classes
- java.net.*; package

TCP Classes	UDP Classes
URL	DatagramPacket

URLConnection	DatabramSocket
Socket	MulticastSocket
ServerSocket	
InetAddress	

InetAddress Class

- Abstract class
- Obtaining an InetAddress Object:
    - InetAddress getLocalHost()
    - InetAddress getByName(String n)
    - InetAddress[ ] getAllByName(String n)

URL Class

- Common Constructors:
    - URL(String spec)
    - URL(String protocol, String host, int port, String file)
    - URL(String protocol, String host, String file)
- Common Methods:
    - Object getContent()
    - String getFile()
    - String getHost()
    - Int getPort()
    - String getProtocol()
    - URLConnection openConnection()
    - InputString openStream()
    - String toString()

Example

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

public class ReadURL
{
   public static void main(String[] args)
   {
      String str;

      try{
         URL url = new URL("http://www.google.com/");
         InputStream is = url.openStream();
         InputStreamReader isr = new InputStreamReader(is);
         BufferedReader br = new BufferedReader(isr);
         while((str = br.readLine()) != null)
```

```
            System.out.println(str);
         br.close();
      } catch(IOException e){
         System.out.println(e);
      }
   }
}
```

Example 2

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

public class ReadURL2
{
   public static void main(String[] args)
   {
      String str;

      try{
         URL url = new URL("http://www.google.com/");
         URLConnection uc = url.openConnection();
         InputStream is = uc.getInputStream();
         InputStreamReader isr = new InputStreamReader(is);
         BufferedReader br = new BufferedReader(isr);
         while((str = br.readLine()) != null)
            System.out.println(str);
         br.close();
      } catch(IOException e){
         System.out.println(e);
      }
   }
}
```

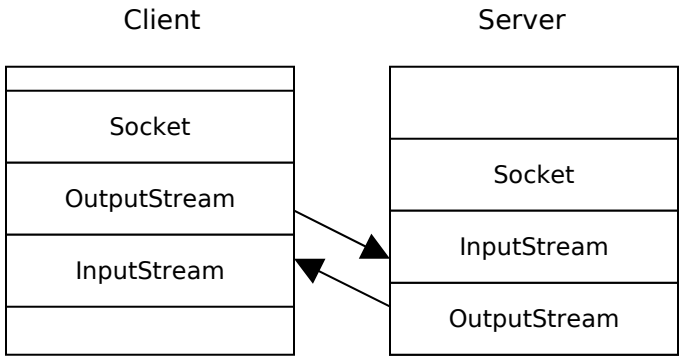
Example 3

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

public class ReadURL3
{
   public static void main(String[] args)
   {
      String str;
      Properties props = System.getProperties();
      props.setProperty("proxySet","true");
```

```
        props.setProperty("ProxyPort","3128");
        props.setProperty("proxyHost","proxy1.wlink.com.np");


        try{
            URL url = new URL("http://www.google.com/");
            URLConnection uc = url.openConnection();
            InputStream is = uc.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            while((str = br.readLine()) != null)
                System.out.println(str);
            br.close();
        } catch(IOException e){
            System.out.println(e);
        }
    }
}
```

Overview of Sockets

- Uses TCP/IP
- Sockets support bi-directional communication with ports
- Connection stays open
- The ServerSocket class waits for a connection
- The Socket class sends and receives packages



Creating a Server

- Create ServerSocket object to listen for connections
- Accept a connection
- Open InputStream
- Open OutputStream
- Send and receive information
- Close streams and connections

Client Side application

- Create Socket object
- Open InputStream
- Open OutputStream
- Send and receive information
- Close streams and connection.