// Reading from a URL

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

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

Using URLConnection to handle HTTP requests

GET vs POST

GET is to retrieve data from resource
POST submits data to be processed

GET submits a query string as a name/value pair.

Example: www.myserver.com/myform.asp?name=val1&address=val2

GET properties:

- cached, stay in browser history, can be bookmarked, have length restrictions (2048 chars)
- data is visible in the URL
- only ASCII characters allowed

POST requests are sent as part of the HTTP message body

POST properties:

- not cached, not part of browser history, cannot be bookmarked
- binary data allowed
- reload will cause data to be resubmitted

HTTP GET METHOD

```
String url = "http://example.com";
String charset = "UTF-8";
String param1 = "value1";
String param2 = "value2";
String p1 = URLEncoder.encode(param1, charset);
String p2 = URLEncoder.encode(param2, charset);
String query = String.format("param1=%s&param2=%s", p1, p2);

URLConnection connection = new URL(url + "?" + query).openConnection();
connection.setRequestProperty("Accept-Charset", charset);
InputStream response = connection.getInputStream();
// . . .

// if no headers need to be set, invoking openStream is sufficient
InputStream response = new URL(url).openStream();
```

Writing to a URLConnection – HTTP POST METHOD

The following steps would be required:

- Create a URL and retrieve the URLConnection object
- Set output capability (Trigger POST)
- Open a connection and obtain output stream
- Write to output stream

An example program that invokes a Servlet that reverses a string. The Servlet requires that the input is of the form: string=string_to_reverse

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

public class Reverse {
        public static void main(String[] args) throws Exception {
          if (args.length != 2) {
                  System.err.println("Usage:  java Reverse "
                 + "http://<location of your servlet/script>"
                 + " string_to_reverse");

         System.exit(1);
        }

       String stringToReverse = URLEncoder.encode(args[1], "UTF-8");
       URL url = new URL(args[0]);
       URLConnection connection = url.openConnection();
       connection.setDoOutput(true);
       OutputStreamWriter out = new OutputStreamWriter(
                connection.getOutputStream());
       out.write("string=" + stringToReverse);
       out.close();
       BufferedReader in = new BufferedReader(
                new InputStreamReader(
                   connection.getInputStream()));

        String decodedString;
        while ((decodedString = in.readLine()) != null) {
                    System.out.println(decodedString);         }
        in.close();     } }
```

// Using URLConnection to download a file using ftp

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

/**
 * This program demonstrates how to download a file from FTP server
 * using FTP URL.
 * @author www.codejava.net
 *
 */
public class FtpUrlDownload {
    private static final int BUFFER_SIZE = 4096;

    public static void main(String[] args) {
        String ftpUrl = "Error! Hyperlink reference not valid.";
        String host = "www.yourserver.com";
        String user = "tom";
        String pass = "secret";
        String filePath = "/project/2012/Project.zip";
        String savePath = "E:/Download/Project.zip";

        ftpUrl = String.format(ftpUrl, user, pass, host, filePath);
        System.out.println("URL: " + ftpUrl);

        try {
            URL url = new URL(ftpUrl);
            URLConnection conn = url.openConnection();
            InputStream inputStream = conn.getInputStream();

            FileOutputStream outputStream = new FileOutputStream(savePath);

            byte[] buffer = new byte[BUFFER_SIZE];
            int bytesRead = -1;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }

            outputStream.close();
            inputStream.close();

            System.out.println("File downloaded");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Sockets

Ports:
TCP and UDP use ports to map incoming data to a particular process

Networking classes:
TCP: URL, URLConnection, Socket, ServerSocket
UDP: DatagramPacket, DatagramSocket, MulticastSocket

Endpoint of a two-way communication link between two network programs.
An endpoint is defined by an IP address and a port number.
Every TCP connection can be uniquely defined by its endpoints.

Sockets represent a connection between a client and server program.
Java provides the Socket and ServerSocket classes in java.net

ServerSocket listens to and accepts connections to clients.

```
/*
Client code that demonstrates basic use of sockets.  The program sends a user-
input string to a server which echoes it back.

“taranis” is assumed to be the hostname on which the server is running and 7 is
the port it’s listening on.
*/

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

public class EchoClient {
  public static void main(String[] args) throws IOException {
        Socket echoSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;
        try {
                echoSocket = new Socket("taranis", 7);
                out = new PrintWriter(echoSocket.getOutputStream(), true);
                in = new BufferedReader(new InputStreamReader(
                                                echoSocket.getInputStream()));
            } catch (UnknownHostException e) {
                    System.err.println("Don't know about host: taranis.");
                     System.exit(1);         }
        catch (IOException e) {
                System.err.println("Couldn't get I/O for "
                   + "the connection to: taranis.");
        System.exit(1);         }

        BufferedReader stdIn = new BufferedReader(
                                        new InputStreamReader(System.in));
        String userInput;
        while ((userInput = stdIn.readLine()) != null) {
            out.println(userInput);
            System.out.println("echo: " + in.readLine());    }
        out.close();
        in.close();
        stdIn.close();
        echoSocket.close();     } }
```

```
// Download from
//http://docs.oracle.com/javase/tutorial/networking/sockets/examples/KnockKnockClient.java


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

public class KnockKnockClient {
    public static void main(String[] args) throws IOException {

        Socket kkSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            kkSocket = new Socket("taranis", 4444);
            out = new PrintWriter(kkSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(kkSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: taranis.");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection to: taranis.");
            System.exit(1);
        }

        BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
        String fromServer;
        String fromUser;

        while ((fromServer = in.readLine()) != null) {
            System.out.println("Server: " + fromServer);
            if (fromServer.equals("Bye."))
                break;

            fromUser = stdIn.readLine();
         if (fromUser != null) {
                System.out.println("Client: " + fromUser);
                out.println(fromUser);
         }
        }

        out.close();
        in.close();
        stdIn.close();
        kkSocket.close();
    }
}
```

```java
// download from
//http://docs.oracle.com/javase/tutorial/networking/sockets/examples/KnockKnockSer
ver.java

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

public class KnockKnockServer {
    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(4444);
        } catch (IOException e) {
            System.err.println("Could not listen on port: 4444.");
            System.exit(1);
        }

        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException e) {
            System.err.println("Accept failed.");
            System.exit(1);
        }

        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(
                  new InputStreamReader(
                  clientSocket.getInputStream()));
        String inputLine, outputLine;
        KnockKnockProtocol kkp = new KnockKnockProtocol();

        outputLine = kkp.processInput(null);
        out.println(outputLine);

        while ((inputLine = in.readLine()) != null) {
             outputLine = kkp.processInput(inputLine);
             out.println(outputLine);
             if (outputLine.equals("Bye."))
                break;
        }
        out.close();
        in.close();
        clientSocket.close();
        serverSocket.close();
    }
}
```

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

public class KnockKnockProtocol {
    private static final int WAITING = 0;
    private static final int SENTKNOCKKNOCK = 1;
    private static final int SENTCLUE = 2;
    private static final int ANOTHER = 3;

    private static final int NUMJOKES = 5;

    private int state = WAITING;
    private int currentJoke = 0;

    private String[] clues = { "Turnip", "Little Old Lady", "Atch", "Who", "Who" };
    private String[] answers = { "Turnip the heat, it's cold in here!",
                                 "I didn't know you could yodel!",
                                 "Bless you!",
                                 "Is there an owl in here?",
                                 "Is there an echo in here?" };

    public String processInput(String theInput) {
        String theOutput = null;

        if (state == WAITING) {
            theOutput = "Knock! Knock!";
            state = SENTKNOCKKNOCK;
        } else if (state == SENTKNOCKKNOCK) {
            if (theInput.equalsIgnoreCase("Who's there?")) {
                theOutput = clues[currentJoke];
                state = SENTCLUE;
            } else {
                theOutput = "You're supposed to say \"Who's there?\"! " +
                 "Try again. Knock! Knock!";
            }
        } else if (state == SENTCLUE) {
            if (theInput.equalsIgnoreCase(clues[currentJoke] + " who?")) {
                theOutput = answers[currentJoke] + " Want another? (y/n)";
                state = ANOTHER;
            } else {
                theOutput = "You're supposed to say \"" +
                 clues[currentJoke] +
                 " who?\"" +
                 "! Try again. Knock! Knock!";
                state = SENTKNOCKKNOCK;
            }
        } else if (state == ANOTHER) {
            if (theInput.equalsIgnoreCase("y")) {
                theOutput = "Knock! Knock!";
                if (currentJoke == (NUMJOKES - 1))
                    currentJoke = 0;
                else
                    currentJoke++;
                state = SENTKNOCKKNOCK;
            } else {
                theOutput = "Bye.";
                state = WAITING;
            }
        }
        return theOutput;
    }}
```

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

public class KKMultiServer {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = null;
        boolean listening = true;

        try {
            serverSocket = new ServerSocket(4444);
        } catch (IOException e) {
            System.err.println("Could not listen on port: 4444.");
            System.exit(-1);
        }

        while (listening)
         new KKMultiServerThread(serverSocket.accept()).start();

        serverSocket.close();
    }}


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

public class KKMultiServerThread extends Thread {
    private Socket socket = null;

    public KKMultiServerThread(Socket socket) {
     super("KKMultiServerThread");
     this.socket = socket;
    }

    public void run() {

     try {
         PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
         BufferedReader in = new BufferedReader(
                         new InputStreamReader(
                         socket.getInputStream()));

         String inputLine, outputLine;
         KnockKnockProtocol kkp = new KnockKnockProtocol();
         outputLine = kkp.processInput(null);
         out.println(outputLine);

         while ((inputLine = in.readLine()) != null) {
       outputLine = kkp.processInput(inputLine);
       out.println(outputLine);
       if (outputLine.equals("Bye"))
           break;
         }
         out.close();
         in.close();
         socket.close();

     } catch (IOException e) {
         e.printStackTrace();
     }
    }
}
```

Datagrams:

A *datagram* is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed.

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

public class QuoteClient {
    public static void main(String[] args) throws IOException {

        if (args.length != 1) {
             System.out.println("Usage: java QuoteClient <hostname>");
             return;
        }

            // get a datagram socket
        DatagramSocket socket = new DatagramSocket();

            // send request
        byte[] buf = new byte[256];
        InetAddress address = InetAddress.getByName(args[0]);
        DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 4445);
        socket.send(packet);

            // get response
        packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);

         // display response
        String received = new String(packet.getData(), 0, packet.getLength());
        System.out.println("Quote of the Moment: " + received);

        socket.close();
    }
}
```

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

public class QuoteServerThread extends Thread {

    protected DatagramSocket socket = null;
    protected BufferedReader in = null;
    protected boolean moreQuotes = true;

    public QuoteServerThread() throws IOException {
     this("QuoteServerThread");
    }

    public QuoteServerThread(String name) throws IOException {
        super(name);
        socket = new DatagramSocket(4445);

        try {
            in = new BufferedReader(new FileReader("one-liners.txt"));
        } catch (FileNotFoundException e) {
            System.err.println("Could not open quote file. Serving time instead.");
        }}

    public void run() {

        while (moreQuotes) {
            try {
                byte[] buf = new byte[256];

                // receive request
                DatagramPacket packet = new DatagramPacket(buf, buf.length);
                socket.receive(packet);

                // figure out response
                String dString = null;
                if (in == null)
                    dString = new Date().toString();
                else
                    dString = getNextQuote();

                buf = dString.getBytes();

       // send the response to the client at "address" and "port"
                InetAddress address = packet.getAddress();
                int port = packet.getPort();
                packet = new DatagramPacket(buf, buf.length, address, port);
                socket.send(packet);
            } catch (IOException e) {
                e.printStackTrace();
       moreQuotes = false;
            }
         }
        socket.close();
    }

    protected String getNextQuote() {
        String returnValue = null;
        try {
            if ((returnValue = in.readLine()) == null) {
                in.close();
```

```
            moreQuotes = false;
                      returnValue = "No more quotes. Goodbye.";
                  }
              } catch (IOException e) {
                  returnValue = "IOException occurred in server.";
              }
              return returnValue;
          }
      }
```