Unit-3 (URLs and URIs)

URL(Uniform Resource Locator)

It is string of characters but it refers to just the **address.**

It is the most used way to locate resources on the web.

E.g. Address:

House No., Ward No,

Kathmandu, Nepal.

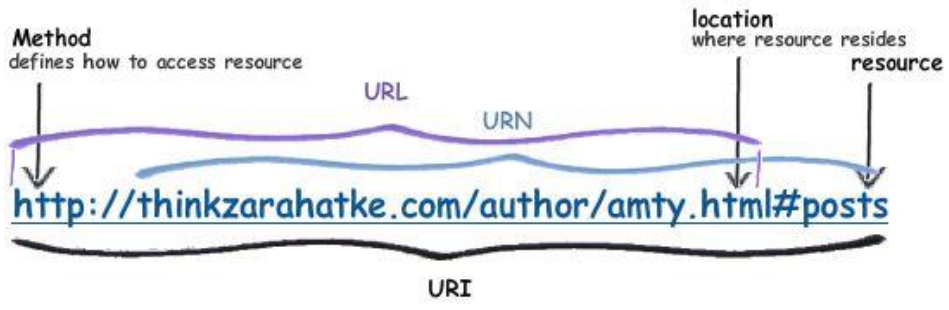
The **URL** Class is the simplest way for a java program to **locate** and **retrieve** information form the network.

It consists of multiple parts including **protocols, IP address, path and port** to locate the information and the mechanism for retrieving it.

URL uses protocol such as **http://** and **ftp://** to identify the resource

The syntax of a URL is:

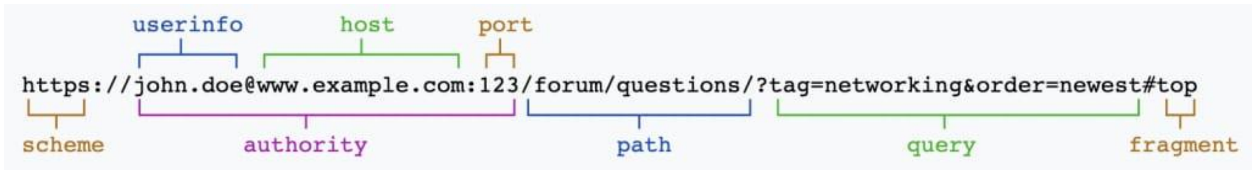
protocol://userInfo@host:port/path?query#fragment


Method
defines how to access resource
URL
URN
location
where resource resides
resource
http://thinkzarahatke.com/author/amty.html#posts
URI

Absolute URL: With absolute URL, you put the exact address on the page that you're linking too. For example, a tag with an absolute URL would look like <a href = https://yourwebsite.com/yourpage.html>

Relative URL: A relative URL does not have a full address. Instead, it tells the browser to assume that the page is on the same website. So, to get to yourpage.html, your tag would look like <a href = "yourpage.html">

Component of URL


userinfo
host
port
https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top
scheme
authority
path
query
fragment

1. The **Scheme**, which is the protocol that you're using to interact.
2. The **Authority**, which is the target you're accessing. This breaks down into **userinfo**, **host**, and **port**.
3. The **Path**, which is the resource you're requesting on the host.
4. The **Query**, which are the parameters being used within the web application.
5. The **Fragment**, which is the target to jump to within a given page.

Example:

https://samriddhicollege.edu.np/wp-content/uploads/2019/09/Networking_Programming-Syllabus.zip

Base URL:

https://samriddhicollege.edu.np/wp-content/uploads/2019/09/

Relative URL:

Networking_Programming-Syllabus.zip

Resolved URL:

https://samriddhicollege.edu.np/wp-content/uploads/2019/09/Networking_Programming-Syllabus.zip

Lab: Write a java program to find baseurl,relativeurl and resolvedurl from given url https://samriddhicollege.edu.np/wp-content/uploads/2019/09/Networking_Programming-Syllabus.zip

Example:

```
public static void main(String[] args) throws MalformedURLException {

        String baseurl="https://samriddhicollege.edu.np/wp-
content/uploads/2019/09/";
        String relativeUrl="Networking_Programming-Syllabus.zip";
        URL baseUrl=new URL(baseurl);
        URL resolvedRelativeUrl=new URL(baseUrl, relativeUrl);
        System.out.println("BaseUrl:"+baseurl);
        System.out.println("Relative Url:"+relativeUrl);
        System.out.println("Resolved Relative Url:"+resolvedRelativeUrl);
        }
```

```
Output:
BaseUrl:https://samriddhicollege.edu.np/wp-content/uploads/2019/09/

Relative Url:Networking_Programming-Syllabus.zip

Resolved Relative Url:https://samriddhicollege.edu.np/wp-
content/uploads/2019/09/Networking_Programming-Syllabus.zip
```

The URL Class:

The java.net.URL class is an abstraction of a Uniform Resource Locator such as http:// www.samriddhicollege.edu.np/ or ftp://ftp.redhat.com/pub/.

Creating New URLs

Unlike the **InetAddress** objects, you can construct instances of **java.net.URL**. The constructors differ in the information they require:

```
public URL(String url) throws MalformedURLException
```

(Creates an instance of a URL from the String representation.)

```
URL url1 = new URL("https://samriddhicollege.edu.np/bca-college-in-kathmandu-
valley/");
```

```
public URL(String protocol, String hostname, String file)
throws MalformedURLException
```

(Creates an instance of a URL from the given protocol name, host name, and file name.)

```
 URL urlexample=new URL("Https", "samriddhicollege.edu.np", "/bca-college-in-
kathmandu-valley/");
```

```
public URL(String protocol, String host, int port, String file)
```

```
throws MalformedURLException
```
(Creates an instance of a URL from the given protocol, host, port number, and file.)

```
URL urlexample=new URL("Https", "samriddhicollege.edu.np",443, "/bca-college-in-kathmandu-valley/");
```

```
public URL(URL base, String relative) throws MalformedURLException
```
(*Creates an instance of a URL by parsing the given spec within a specified context.)*

```
URL u1=new URL("https://www.samriddhicollge.edu.np");
  URL u2=new URL(u1, "bca-college-in-kathmandu-valley");
```

All these constructors throw a **MalformedURLException** if you try to create a URL for an unsupported protocol or if the URL is syntactically incorrect.

Lab: write a java program to spit different component of url from given url:
(https://www.example.com:8080/path/to/resource?key1=value1&key2=value2#section2)

Example:
```
    public static void main(String[] args) throws MalformedURLException {
URL url1 = new
URL("https://www.example.com:8080/path/to/resource?key1=value1&key2=value2#sectio
n2");
        System.out.println(url1.toString());
        System.out.println();
        System.out.println(
            "Different components of the URL1-");
       System.out.println("Protocol:- " + url1.getProtocol());
        System.out.println("Hostname:- " + url1.getHost());
System.out.println("Default port:- "+ url1.getDefaultPort());
        // Retrieving the query part of URL
        System.out.println("Query:- " + url1.getQuery());
        // Retrieving the path of URL
        System.out.println("Path:- " + url1.getPath());
        // Retrieving the file name
        System.out.println("File:- " + url1.getFile());
        // Retrieving the reference
        System.out.println("Reference:- " + url1.getRef());
        }
```
Output:
Different components of the URL1-
Protocol:- https
Hostname:- www.example.com
Default port:- 443

Query:- key1=value1&key2=value2
Path:- /path/to/resource
File:- /path/to/resource?key1=value1&key2=value2
Reference:- section2

Retrieving Data from a URL:

The URL class has several methods that retrieve data from a URL

```
public InputStream openStream() throws IOException
public URLConnection openConnection() throws IOException
public URLConnection openConnection(Proxy proxy) throws IOException
public Object getContent() throws IOException
public Object getContent(Class[] classes) throws IOException
```

openStream():

- The most basic and most commonly used of these methods is **openStream()**, which returns an **InputStream** from which you can read the data.
- The data you get from this **InputStream** is the raw (i.e., uninterpreted) content the URL references: ASCII if you're reading an ASCII text file, raw HTML if you're reading an HTML file, binary image data if you're reading an image file, and so forth.
- It does not include any of the HTTP headers or any other protocol-related information.

OpenStream Example:

```
import java.io.*;
import java.net.*;
public class App {
    public static void main(String[] args) throws Exception  {
        try {
            URL u = new URL("https://www.ncell.com.np/en/about/career

");
            InputStream in = u.openStream();
            int c;
            while ((c = in.read()) != -1)
             System.out.write(c);
             in.close();
           } catch (IOException ex) {
            System.err.println(ex);
           }

    }

}
```

```
        BufferedReader in = new BufferedReader(
    new InputStreamReader(u.openStream()));

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

openConnection():

- The URLConnection gives you access to everything sent by the server: in addition to the document itself in its raw form (e.g., HTML, plain text, binary image data).
- URLConnection lets you access the HTTP headers as well as the raw HTML

OpenConnection Example:

```
import java.io.*;
import java.net.*;
public class App {
    public static void main(String[] args) throws Exception  {
        StringBuilder content = new StringBuilder();
          try
          {
            URL u = new URL("https://www.ncell.com.np/en/about/career

");
            URLConnection uc = u.openConnection();
            InputStream in = uc.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(in));
            String line;
            while ((line = bufferedReader.readLine()) != null)
            {
              content.append(line + "\n");
            }
            bufferedReader.close();
          }
          catch(Exception e)
          {
            e.printStackTrace();
          }
          System.out.println(content);
    }
}
```

getContent():

getContent() is a method that is often used to fetch and process the content of a specified URL. This method is typically part of a larger library or API for handling HTTP connections..

GetContentExample:

```
import java.net.*;
public class App {
    public static void main(String[] args) throws Exception  {
        URL url1=new URL("https://samriddhicollege.edu.np/contact-us/");
        System.out.println(" Given Url is : "+url1);
        System.out.println(" The content of given url is: "+url1.getContent());
    }

}
Output:
Given Url is : https://www.ncell.com.np/en/about/career
 The content of given url is:
sun.net.www.protocol.http.HttpURLConnection$HttpInputStream@ca263c2
```

Splitting a URL into Pieces:

URLs are composed of five pieces:

- The scheme, also known as the protocol
- The authority
- The path
- The fragment identifier, also known as the section or ref

- The query string

For example, in the URL **http://www.ibiblio.org/javafaq/books/jnp/index.html?isbn=1565922069#toc**, the **scheme** is http, the **authority** is www.ibiblio.org, the **path** is /javafaq/books/jnp/index.html, the **fragment** identifier is toc, and the **query string** is isbn=1565922069. However, not all URLs have all these pieces. For instance, the URL http://www.faqs.org/rfcs/rfc3986.html has a scheme, an authority, and a path, but no fragment identifier or query string.
The **authority** may further be divided into the user info, the host, and the port. For example, in the URL http://admin@www.blackstar.com:8080/, the authority is **ad min@www.blackstar.com:8080**. This has the **user info** admin, the **host** www.black- star.com, and the **port** 8080.

Read-only access to these parts of a URL is provided by nine public methods: **get File(), getHost(), getPort(), getProtocol(), getRef(), getQuery(), getPath(), getUserInfo(), and getAuthority().**
Example:

```
import java.net.*;
public class App {
    public static void main(String[] args) throws Exception {
       String url="ftp://mp3:mp3@138.247.121.61:21000/c%3a/";
       try {
        URL u = new URL(url);
        System.out.println("The URL is " + u);
        System.out.println("The scheme is " + u.getProtocol());
        System.out.println("The user info is " + u.getUserInfo());
        String host = u.getHost();
        if (host != null) {
        int atSign = host.indexOf('@');
        if (atSign != -1) host = host.substring(atSign+1);
        System.out.println("The host is " + host);
        } else {
```

```
        System.out.println("The host is null.");
        }
        System.out.println("The port is " + u.getPort());
        System.out.println("The path is " + u.getPath());
        System.out.println("The ref is " + u.getRef());
        System.out.println("The query string is " + u.getQuery());
        } catch (MalformedURLException ex) {
        System.err.println(url + " is not a URL I understand.");
        }
        System.out.println();
    }
}
```

Output:

```
The URL is ftp://mp3:mp3@138.247.121.61:21000/c%3a/
The scheme is ftp
The user info is mp3:mp3
The host is 138.247.121.61
The port is 21000
The path is /c%3a/
The ref is null
The query string is null
```

Equality and Comparison

- The URL class contains the usual **equals()** and **hashCode**() methods.
- Two URLs are considered equal if and only if both URLs point to the same resource on the same host, port, and path, with the same fragment identifier and query string.

Example: Creates URL objects for **http://www.ibiblio.org/** and **http://ibiblio.org/** and tells you if they're the same using the **equals()** method.

```
import java.net.*;
public class App {
    public static void main(String[] args) throws Exception {
        try {
            URL www = new URL ("http://www.ibiblio.org/");
            URL ibiblio = new URL("http://ibiblio.org/");
            if (ibiblio.equals(www)) {
            System.out.println(ibiblio + " is the same as " + www);
            } else {
            System.out.println(ibiblio + " is not the same as " + www);
            }
            } catch (MalformedURLException ex) {
            System.err.println(ex);
            }

    }
}
```

Output:

http://ibiblio.org/ is the same as http://www.ibiblio.org/

The URL class also has a **sameFile()** method that checks whether two URLs point to the same resource:

```
public boolean sameFile(URL other)
```

Example:

```
import java.net.*;
public class App {
    public static void main(String[] args) throws Exception {
        URL u1 = new URL("http://www.ncsa.uiuc.edu/HTMLPrimer.html#GS");
        URL u2 = new URL("http://www.ncsa.uiuc.edu/HTMLPrimer.html#HD");
        if (u1.sameFile(u2)) {
        System.out.println(u1 + " is the same file as \n" + u2);
        } else {
        System.out.println(u1 + " is not the same file as \n" + u2);
        }

    }
}
```

Output:

```
http://www.ncsa.uiuc.edu/HTMLPrimer.html#GS is the same file as
http://www.ncsa.uiuc.edu/HTMLPrimer.html#HD
```

Conversion

URL has three methods that convert **an instance to another form**: toString(), toExternalForm(), and toURI(). Like all good classes, java.net.URL has a toString() method. The String produced by toString() is always an **absolute URL**, such as **http://www.cafeaulait.org/javatuto-rial.html**.

```
public String toExternalForm()
```

The toExternalForm() method returns a human-readable String representing the URL. It is identical to the toString() method.

```
 public URI toURI() throws URISyntaxException
```

URI class provides much more accurate, specification-conformant behavior than the URL class

URI Class:

- A URI is a generalization of a URL that includes not only **Uniform Resource Locators** but also **Uniform Resource Names** (URNs). Most URIs used in practice are URLs,
- In Java, URIs are represented by the **java.net.URI** class. This class differs from the java.net.URL class.

URI Constructor:

URI(String str) : Constructs a URI object by parsing the specified string
URI(String scheme, String ssp, String fragment): ssp(`schemeSpecificPart)` Constructs a URI from the given components. A component may be left undefined by passing null. Initially the result string is empty.
URI(String scheme, String userInfo, String host, int port, String path, String query, String fragment)
URI(String scheme, String host, String path, String fragment)
URI(String scheme, String authority, String path, String query, String fragment)

Constructing a URI

```
  public URI(String uri) throws URISyntaxException
  public URI(String scheme, String schemeSpecificPart, String fragment)
    throws URISyntaxException
  public URI(String scheme, String host, String path, String fragment)
    throws URISyntaxException
  public URI(String scheme, String authority, String path, String query,
    String fragment) throws URISyntaxException
  public URI(String scheme, String userInfo, String host, int port,
    String path, String query, String fragment) throws URISyntaxException
```

Schema: string representing schema
UserInfo: userinfo of URI

Host:host component of URI
Port: listening port number
Path: path of URI
Query: querystring part of URI
Fragment: optional Fragment of URI

1. constructor creates a new URI object from any convenient string. For example:

```
    URI voice = new URI("tel:+1-800-9988-9938");
    URI web = new URI("http://www.xml.com/pub/a/2003/09/17/stax.html#id=_hbc");
    URI book = new URI("urn:isbn:1-565-92870-9");
```

```
2. URI absolute = new URI("http", "//www.ibiblio.org" , null);
```

```
3. URI today= new URI("http", "www.ibiblio.org", "/javafaq/index.html", "today");
```

This produce **URI http://www.ibiblio.org/javafaq/index.html#today.**

4. `URI today = new URI("http", "www.ibiblio.org", "/javafaq/index.html",`

```
 "referrer=cnet&date=2014-02-23", "today");
5. URI styles = new URI("ftp", "anonymous:elharo@ibiblio.org",

    "ftp.oreilly.com", 21, "/pub/stylesheet", null, null);
```

The Parts of the URI

Methods:

create() : creates a new URI object.
toURL() : Constructs a URL from this URI.
getScheme() : Returns the scheme component of the URI
getRawSchemeSpecificPart() : Returns the raw scheme specific component of the URI
getSchemeSpecificPart() : Returns the decoded scheme specific component of the URI
getUserInfo() : Returns the user info component of the URI in decoded form, or null if it is undefined.

```
public String getScheme()
public String getSchemeSpecificPart()
public String getRawSchemeSpecificPart()
public String getFragment()
public String getRawFragment()
```

Example:

Lab:Write a java program to get different parts of URI in given URI= http://www.xml.com/pub/a/2003/09/17/stax.html#id=_hbc

```java
import java.net.*;
public class App
    {
    public static void main(String[] args) throws Exception {
        String str = "http://www.xml.com/pub/a/2003/09/17/stax.html#id=_hbc";
        URI uri = URI.create(str);
        System.out.println(uri.normalize().toString());
        System.out.println("Scheme = " + uri.getScheme());
        System.out.println("Schemespecificpart = "+ uri.getSchemeSpecificPart());
         System.out.println("Raw User Info = " + uri.getFragment());
         System.out.println("User Info = " + uri.getUserInfo());
        System.out.println("Authority = " + uri.getAuthority());
        System.out.println("Host = " + uri.getHost());
        System.out.println("Path = " + uri.getPath());
        System.out.println("Port = " + uri.getPort());
        System.out.println("Query = " + uri.getQuery());

    }
}
```

```
Output:
http://www.xml.com/pub/a/2003/09/17/stax.html#id=_hbc
Scheme = http
Raw Scheme = //www.xml.com/pub/a/2003/09/17/stax.html
Scheme-specific part = //www.xml.com/pub/a/2003/09/17/stax.html
Raw User Info = id=_hbc
User Info = id=_hbc
Raw User Info = null
User Info = null
Authority = www.xml.com
Raw Authority = www.xml.com
Host = www.xml.com
Path = /pub/a/2003/09/17/stax.html
Port = -1
Query = null
```

A URI that has a scheme is an absolute URI. A URI without a scheme is relative. The isAbsolute() method returns true if the URI is absolute, false if it's relative:

Resolving Relative URIs

The URI class has three methods for converting back and forth between relative and absolute **URIs: public URI resolve(URI uri) p**
Public URI resolve(String uri)
public URI relativize(URI uri)

The resolve() methods compare the uri argument to this URI and use it to construct a new URI object that wraps an absolute URI. For example, consider these three lines of code:
URI absolute = new URI("http://www.example.com/");
URI relative = new URI("images/logo.png");
URI resolved = absolute.resolve(relative);

Example:

```
import java.net.*;

public class App
    {
    public static void main(String[] args) throws Exception {

        String uribase="https://samriddhicollege.edu.np";
        String urirelative="/student-clubs/";
        URI uriBase=new URI(uribase);
        //create() method
        URI uri=URI.create(uribase);
        System.out.println("Base URI="+uriBase.toString());
        URI uriRelative=new URI(urirelative);
        System.out.println("Relative URI="+uriRelative.toString());
        //resolve method
        URI uriResolved=uriBase.resolve(uriRelative);
        System.out.println("Resolved URI="+uriResolved.toString());

    }
}
```

Output:

```
Base URI=https://samriddhicollege.edu.np
Relative URI=/student-clubs/
Resolved URI=https://samriddhicollege.edu.np/student-clubs/
```

URL Encoder and Decoder:
It is necessary when we are dealing with data transmission over the network. Data transmitted between client and server makes more secure.
URL Encoder:

- **Java.net.URLEncoder** class
- Encoding makes the form of URL more **reliable** and **secure**
- Where the user request is triggered by a **get** method, the form parameters and their values are appended at the end of URL after a '?' sign.
- This process replaces unsafe ASCII characters with a '%' followed by two hexadecimal digits.

Example
bca@network programming=>bca%40network+programming

Rules where encoding a string

- Alphanumeric characters and certain special characters such as '*','_','-' and '.' Remains unchanged
- Spaces are converted into '+' signs.
- All other characters are encoded by one or more bytes using the encoding schema specified.
  Example:

bca@samriddhi college=>bca%40samriddhi+college

Method: encode()

a) **Syntax:public static String encode(String s)**
   Parameters:
   s:String to be encoded
```
String url=URLEncoder.encode("https://samriddhicollege.edu.np/bca-college-
in-kathmandu-valley/");
```

b) **Syntax:public static String encode(String s, String enc)**
```
Parameters:
s:string to be encoded
Enc: encoding to be used e.g. UTF-8
String url=URLEncoder.encode("https://samriddhicollege.edu.np/bca-college-
in-kathmandu-valley/","UTF-8");
```

URL Decoder

- **Java.net.URLDecoder** class
- URL decoding is the reverse process of URL encoding.
- It converts the encoded characters back to their original form.

Example:
Bca%40network+programming=>bca@network programming

Steps while decoding the strings

- Alphanumeric characters and certain special characters such as '*','_','-' and '.' Remains unchanged
- + signs are converted into spaces.
- All other characters are decoded using the encoding scheme specified. The string of the form %xy, is converted to the character whose encoding would have resulted in this three character representation.

Example:
Bca%40samriddhi+college=>bca@samriddhi college

Decode()

a) **Syntax**; public static String decode(String s)
Parameters:
s:encoded string to be decoded
URLDecoder.decode("Hello+world");

b) **Syntax**:public static String decode(String s, String enc)
Parameters:
s:string to be decoded
enc:encoding to be used e.g. UTF-8

URLDecoder.decode("Hello+world","UTF-8")

Lab: Write a java program to perform url encoding and decoding.

Example:

```
            String url1=URLEncoder.encode("https://samriddhicollege.edu.np/bca-
college-in-kathmandu-valley/","UTF-8");//dencode
            System.out.println(url1);
           String deurl= URLDecoder.decode(url1);//decode
           System.out.println(deurl);
```

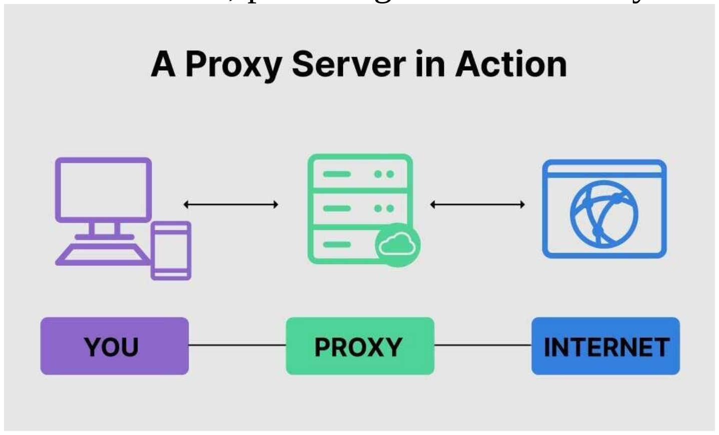
Proxies:

- Proxy means **"in place of"**, representing or **"in place of "** or **"on behalf of"**
- A real world example can be a cheque or credit card is a proxy for what is in our bank account.
- Proxy pattern does-**"Controls and manage access to the object they are protecting".**

Benefits of Using a Proxy Server for Network Programming

- It provides an extra layer of security, as the proxy server can filter out malicious traffic and protect the computers from potential attacks.

- Proxies can restrict access to specific websites or content, helping to control what users can and cannot view.
- Additionally, the proxy server can be used to hide the IP address. This can help protect personal information and browsing from being tracked.
- Proxies can filter out malicious content and block access to harmful websites, providing an additional layer of security.



System Properties:

- For basic operations, set a few system properties to **point to the addresses of your local proxy servers.**
- If you are using a pure HTTP proxy, set **http.proxyHost** to the **domain name** or the **IP address** of your proxy server and **http.proxyPort** to the **port of the proxy server** (the default is 80).

```
System.setProperty("http.proxyHost", "192.168.254.254");
System.setProperty("http.proxyPort", "9000");
System.setProperty("http.nonProxyHosts", java.oreilly.com|xml.oreilly.com");
```

Proxy Class

- The **proxy class** allows **more fine-grained control** of proxy server from with in a java program
- Specifically, it allows you to choose different proxy servers for different remote hosts
- The proxies themselves are represented by instances of the **java.net.Proxy** class.

There are three kinds of proxies

- **Proxy.Type.DIRECT**
This type indicates a direct connection to the target server without using any proxy.
When you don't want to route traffic through a proxy server and prefer a straightforward connection.
- **Proxy.Type.HTTP**
An HTTP proxy is used for HTTP requests. It can handle HTTP traffic and can cache web content, perform content filtering, and provide anonymity by masking the original IP address.
- **Proxy.Type.SOCKS**
A SOCKS proxy routes network packets between a client and server through a proxy server. It can handle various types of traffic, including HTTP, FTP, and SMTP.

Example:

```
 SocketAddress address = new InetSocketAddress("proxy.example.com", 80);
 Proxy proxy = new Proxy(Proxy.Type.HTTP, address);
```

The ProxySelector Class

Each running virtual machine has a single **java.net.ProxySelector** object it uses to **locate the proxy server** for different connections.
To change the Proxy Selector, pass the new selector to the static **ProxySelector.setDefault()** method, like so:

```
ProxySelector selector = new LocalProxySelector();//return list of proxies
ProxySelector.setDefault(selector);
```

Communicating with Server-Side Programs Through GET

- The URL class makes it easy for Java applets and applications to communicate with server side programs such as CGIs, servlets, PHP pages, and others that use the GET method.
- For example, consider this HTML form for the local search engine on my Café site. You can see that it uses the GET method. The program that processes the form is accessed via the URL http://www.google.com/search. It has four separate name-value pairs, three of which have default values:

```
<form name="search" action="MyServlet" method="get">
    <input name="q" />
    <input type="hidden" value="cafeconleche.org" name="domains" />
    <input type="hidden" name="sitesearch" value="cafeconleche.org" />
    <input type="hidden" name="sitesearch2" value="cafeconleche.org"
/>
    <br />
    <input type="image" height="22" width="55"
    src="images/search_blue.gif" alt="search" border="0"
    name="search-image" />
   </form>
```

Accessing Password-Protected Sites

- Many popular sites, such as The Wall Street Journal, require a username and password for access. Some sites, such as the W3C member pages, implement this correctly through HTTP authentication.
- Java's URL class can access sites that use HTTP authentication
- Java does not provide support for sites that use nonstandard, cookie-based authentication

THE AUTHENTICATOR CLASS

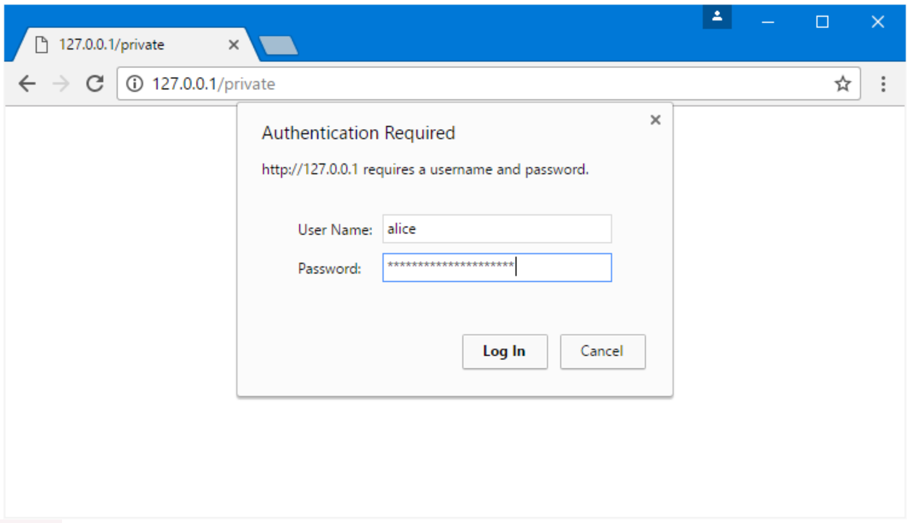
The java authenticator class is used to perform authentication for network connection. The java authenticator class is a built-in class in java. The authenticator class is used in an application where authentication is required to visit some URLs for the users. An authenticator class performs authentication by prompting the user for credential information like username and password.

The java.net package includes an Authenticator class you can use to provide a username and password for sites that protect themselves using HTTP authentication:

```
public abstract class Authenticator extends Object // Java 1.2
```

Since Authenticator is an abstract class, you must subclass it. Different subclasses may retrieve the information in different ways. For example, a character mode program might just ask the user to type the username and password on System.in.

Screenshot of Authentication Dialog.



Authentication Required

http://127.0.0.1 requires a username and password.

User Name: alice

Password: ********************

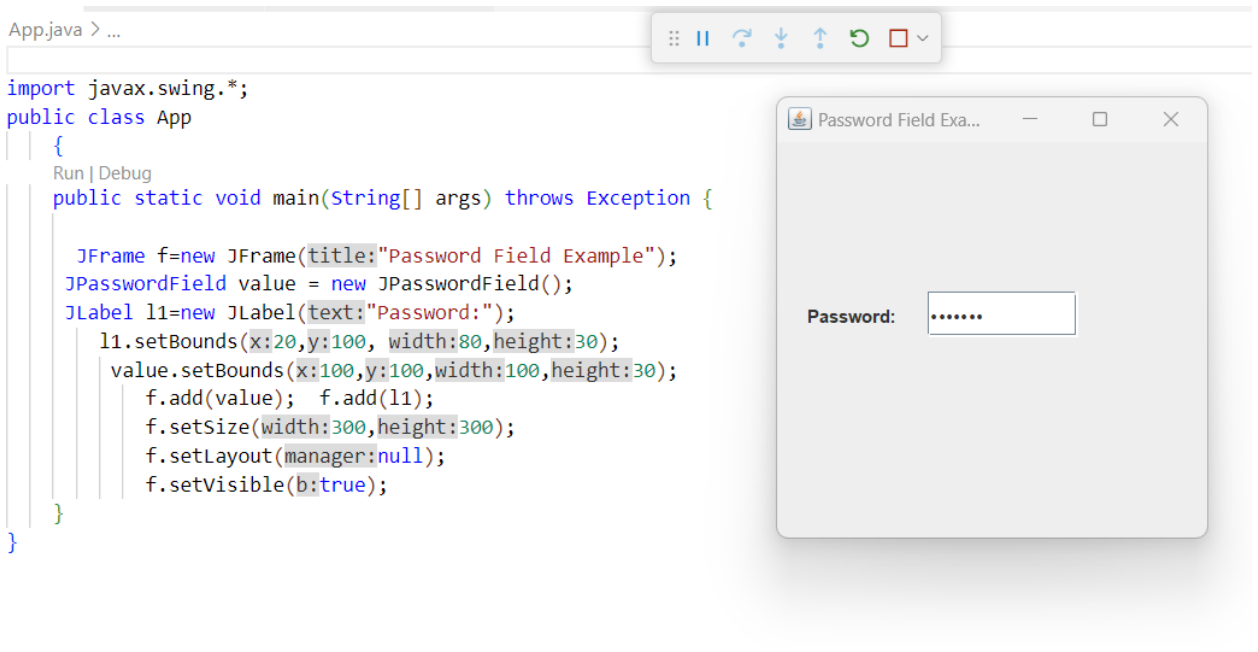
To make the URL class use the subclass, install it as the default authenticator by passing it to the static Authenticator.setDefault() method:

```
public static void setDefault(Authenticator a)
Example:
import java.io.*;
import java.net.*;
public class App
    {
    public static void main(String[] args) throws Exception {

      String data;
      try {
      //create object of authenticator class
      MyAuthenticatorclass obj =new MyAuthenticatorclass();
      Authenticator.setDefault(new MyAuthenticatorclass());
      URL url = new URL("https://www.educba.com/");
      // reads data from the url in html form
      BufferedReader br = new BufferedReader(new
InputStreamReader(url.openStream()));
      System.out.println("The requesting URL is : "+url.getHost());
      obj.getPasswordAuthentication() ;//validation
      while ((data = br.readLine()) != null) {
      System.out.println(data);
      }
       //403 Status Code
      br.close();
      } catch (MalformedURLException e) {
      System.out.println("Malformed URL Exception : " + e);
      } catch (IOException e) {
          System.out.println("IO Exception: " + e);
          }



        }
}
class MyAuthenticatorclass extends Authenticator
{
        protected PasswordAuthentication getPasswordAuthentication()
        {
        String username = "suneel", password = "suneel123";
        this.print();
        return new PasswordAuthentication(username, password.toCharArray());
        }
        void print()
        {
        int hostname = getRequestingPort();
        System.out.println("The request Port number :" + hostname);
        }
}
```

The JPasswordField Class

One useful tool for asking users for their passwords in a more or less secure fashion is the **JPasswordField** component from Swing:

public class **JPasswordField** extends **JTextField**



App.java > ...

```
import javax.swing.*;
public class App
    {
    Run | Debug
    public static void main(String[] args) throws Exception {

      JFrame f=new JFrame(title:"Password Field Example");
     JPasswordField value = new JPasswordField();
     JLabel l1=new JLabel(text:"Password:");
        l1.setBounds(x:20,y:100, width:80,height:30);
         value.setBounds(x:100,y:100,width:100,height:30);
            f.add(value);  f.add(l1);
            f.setSize(width:300,height:300);
            f.setLayout(manager:null);
            f.setVisible(b:true);
    }
}
```