UNIT 3
URLs AND URIs

LH - 5HRS

PREPARED BY:

ER. SHARAT MAHARJAN

NETWORK PROGRAMMING

PRIME COLLEGE

CONTENTS (LH - 5HRS)

3.1 URIs, URLs and Relative URLs

3.2 The URL Class: Creating New URLs, Retrieving Data from a URL, Splitting a URL into Pieces, Equality & Comparison and Conversion

3.3 The URI Class: Constructing a URI, The Parts of the URI, Resolving

3.4 x-www-form-urlencoded: URL Encoder and URL Decoder

3.5 Proxies: System Properties, The ProxyClass and The ProxySelector Class

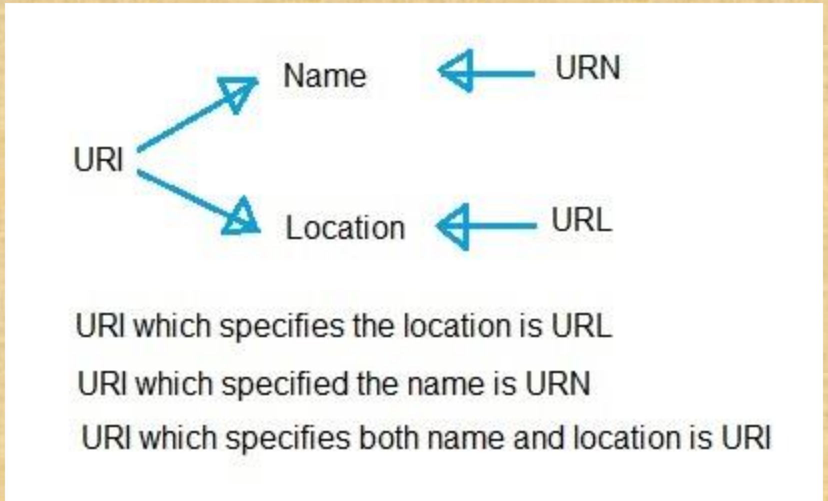
3.6 Communicating with Server-Side Programs Through GET

3.7 Accessing Password-Protected Sites: The Authenticator Class, The PasswordAuthentication Class and The JPasswordField Class

3.1 URIs, URLs and Relative URLs

1. URIs

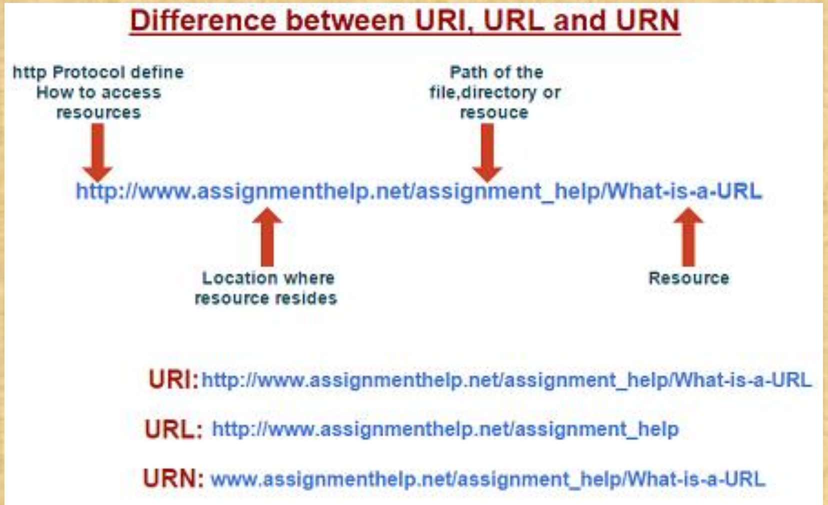
- A Uniform Resource Identifier (URI) is a **string of characters** in a **particular syntax** that **identifies a resource**.
- The resource identified may be a **file** on a server; but it may also be an **email address**, a **news message**, a **book**, a **person's name**, an **Internet host**, the **current stock price of Oracle**, or something else.



URI which specifies the location is URL

URI which specified the name is URN

URI which specifies both name and location is URI


Difference between URI, URL and URN
http Protocol define How to access resources
Path of the file,directory or resouce
http://www.assignmenthelp.net/assignment_help/What-is-a-URL
Location where resource resides
Resource

URI: http://www.assignmenthelp.net/assignment_help/What-is-a-URL

URL: http://www.assignmenthelp.net/assignment_help

URN: www.assignmenthelp.net/assignment_help/What-is-a-URL

- **Syntax**

scheme:scheme-specific-part:fragment

- For instance, the URI http://www.ietf.org/rfc/rfc3986.txt has the **scheme http**, **the authority**(**scheme-specific-part**) **www.ietf.org**, and the **path /rfc/rfc3986.txt** . This URI does not have a query part.
- The URI http://www.powells.com/cgibin/biblio?inkey=62-1565928709-0 has the **scheme http**, the **authority www.powells.com**, the **path /cgi-bin/biblio**, and the **query inkey=62-1565928709-0**.
- The scheme part is composed of lowercase letters, digits, and the plus sign, period, and hyphen.

2. URLs

- A URL is a URI that, as well as **identifying a resource**, provides a specific **network location for the resource** that a **client can use to retrieve a representation of that resource**.
- A **generic URI** may **tell you what a resource** is, but **not actually tell you where or how to get that resource**.
- In Java, it's the difference between the **java.net.URI class that only identifies resources and the java.net.URL class that can both identify and retrieve resources**.
- The syntax of a URL is:

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

- http://www.cafeaulait.org/javafaq.html#xtocid1902914
- In a URL, the protocol part can be file, ftp, http, etc.

3. Relative URLs

- A URL may inherit the **protocol, hostname, and path of its parent document**. **URLs that aren't complete** but inherit pieces from their parent are called **relative URLs**.
- A **completely specified URL** is called an **absolute URL**.

Eg: http://www.ibiblio.org/javafaq/javatutorial.html

    <a href="javafaq.html">

The browser **cuts javatutorial.html** off the end of **http://www.ibiblio.org/javafaq/javatutorial.html** to get **http://www.ibiblio.org/javafaq/**. Then it **attaches javafaq.html onto the end of http://www.ibiblio.org/javafaq/** to get **http://www.ibiblio.org/javafaq/javafaq.html**. Finally, it loads that document.

- If the relative link begins with a /, then it is relative to the document root instead of relative to the current file.
- Thus, if we click on link http://www.ibiblio.org/javafaq/javatutorial.html

  <a href="/projects/ipv6/">

The browser would **throw away /javafaq/javatutorial.html** and **attach /projects/ipv6/ to the end of http://www.ibiblio.org** to **get http://www.ibiblio.org/projects/ipv6/.**

3.2 The URL Class

The URL Class:

1. Creating New URLs
2. Retrieving Data from a URL
3. Splitting a URL into Pieces
4. Equality and Comparison
5. Conversion

- The java.net.URL class is an abstraction of a Uniform Resource Locator such as http://www.lolcats.com/ or ftp://ftp.redhat.com/pub/.
- It extends java.lang.Object, and it is a final class that cannot be subclassed.

1. Creating New URLs

Instances of java.net.URL. The constructors differ in the information they

require:

a. **public URL(String url)** throws MalformedURLException

b. **public URL(String protocol, String hostname, String file)** throws MalformedURLException

c. **public URL(String protocol, String host, int port, String file)** throws MalformedURLException

d. **public URL(URL base, String relative)** throws MalformedURLException

a. Constructing a URL from a string

The simplest URL constructor just takes an absolute URL in string form as its single argument:

public URL(String url) throws MalformedURLException

Example:

```
try {
URL u = new URL("http://www.audubon.org/");
} catch (MalformedURLException ex) {
System.err.println(ex);
}
```

Example 1: Which protocols does a virtual machine support?

```
import java.net.*;
public class ProtocolTester {
public static void main(String[] args) {
// hypertext transfer protocol
testProtocol("http://www.adc.org");
// secure http
testProtocol("https://www.amazon.com/exec
/obidos/order2/");
// file transfer protocol
testProtocol("ftp://ibiblio.org/pub/languages
/java/javafaq/");
// Simple Mail Transfer Protocol
testProtocol("mailto:elharo@ibiblio.org");
// telnet
testProtocol("telnet://dibner.poly.edu/");}
private static void testProtocol(String url) {
try {  //throws when protocol is not supported
URL u = new URL(url);
System.out.println(u.getProtocol() + " is
supported");
} catch (MalformedURLException ex) {
String protocol =
url.substring(0,url.indexOf(':'));
System.out.println(protocol + " is not
supported");
}}}
```

b. Constructing a URL from its component parts

You can also build a URL by specifying the protocol, the hostname, and the file:

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

```
try {
URL u = new URL("http", "www.eff.org", "/blueribbon.html");
} catch (MalformedURLException ex)
{
System.out.println("shouldn't happen; all VMs recognize http");
}
```

c. public URL(String protocol, String host, int port, String file) throws MalformedURLException

```
try {
URL u = new URL("http", "www.eff.org", 80, "/blueribbon.html");
} catch (MalformedURLException ex)
{
System.out.println("shouldn't happen; all VMs recognize http");
}
```

d. Constructing relative URLs

public URL(URL base, String relative) throws MalformedURLException

The constructor computes the new URL as http://
www.ibiblio.org/javafaq/mailinglists.html.

For example:

```
try {
URL u1 = new URL("http://www.ibiblio.org/javafaq/index.html");
URL u2 = new URL (u1, "mailinglists.html");
} catch (MalformedURLException ex) {
System.err.println(ex);
}
```

2. Retrieving Data from a URL

a. **public InputStream openStream()** throws IOException

b. **public URLConnection openConnection()** throws IOException

c. **public URLConnection openConnection(Proxy proxy)** throws IOException

d. **public Object getContent()** throws IOException

e. **public Object getContent(Class[] classes)** throws IOException

a. public final InputStream openStream() throws IOException

```
try {
URL u = new URL("http://www.lolcats.com");
InputStream in = u.openStream();
int c;
while ((c = in.read()) != -1) System.out.write((char)c);
in.close();
} catch (IOException ex) {
System.err.println(ex);
}
```

Example 2: Download a web page

```
import java.io.*;
import java.net.*;
public class SourceViewer {
public static void main (String[] args) {
try {
// Open the URL for reading
URL u = new URL("http://google.com");
// buffer the input to increase performance
BufferedReader in_data = new BufferedReader(new InputStreamReader(u.openStream()));
String entry;
while ((entry = in_data.readLine()) != null) {
System.out.println(entry);}
} catch (MalformedURLException ex) {
System.err.println("URL is not a parseable URL");
} catch (IOException ex) {
System.err.println(ex);}}}
```

d. public final Object getContent() throws IOException

```
URL u = new URL("https://www.oreilly.com");
Object o = u.getContent();
```

Example 3: Download an object

```
import java.io.*;
import java.net.*;
public class ContentGetter {
public static void main (String[] args) {
try {
URL u = new URL(“https://www.oreilly.com”);
Object o = u.getContent();
System.out.println("I got a " + o.getClass().getName());
} catch (MalformedURLException ex) {
System.err.println(args[0] + " is not a parseable URL");
} catch (IOException ex) {
System.err.println(ex);
}}}
```

3. Splitting a URL into Pieces

URLs are composed of **five pieces**:

• The **scheme**, also known as the protocol

• The **authority** (userInfo + host = admin@www.abc.com)

• The **path**

• The **fragment identifier**, link to other places on the same webpage or section

• The **query string**

For example, given the URL http://www.ibiblio.org/books/jnp/index-network-dev-java-programming-language.html.gz?=156522069#toc,

- the **scheme is http**,
- the **authority** is www.ibiblio.org,
- the **path** is /books/jnp/index-network-dev-java-programming-language.html.gz
- the **fragment identifier is toc**, and
- the **query string** is 156522069.

Example 4. The parts of a URL

```
import java.net.*;
public class URLSplitter {
public static void main(String args[]) {
try {
URL u = new
URL("https://www.prime.edu.np/bca/");
System.out.println("The URL is " + u);
System.out.println("The scheme is " +
u.getProtocol());
System.out.println("The user info is " +
u.getUserInfo());
System.out.println("The host is" + u.getHost());
System.out.println("The port is " +
u.getPort());
System.out.println("The path is " +
u.getPath());
System.out.println("The ref is " + u.getRef());
//fragment
System.out.println("The query string is "
+u.getQuery());
} catch (MalformedURLException ex) {
System.err.println("It is not a URL I
understand.");
}
System.out.println();}}
```

4. 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**.
- The **equals() method** actually tries to **resolve the host with DNS**, for example, it can tell that

✓**http://www.ibiblio.org/ and http://ibiblio.org/ are the same.**

- On the other hand, **equals()** does not go so far as to actually compare the resources identified by two URLs.

<u>**For example**</u>,

✓**http://www.oreilly.com/ is not equal to http://www.oreilly.com/index.html;**

✓and **http://www.oreilly.com:80 is not equal to http://www.oreilly.com/.**

- The **hashCode() method returns an int** that is used when URL objects are used as keys in hash tables.

Example 5: Are http://www.ibiblio.org and http://ibiblio.org the same?

```
import java.net.*;
public class URLEquality {
public static void main (String[] args) {
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);
}}}
```

5. Conversion

- URL has three methods that convert an instance to another form:

1. toString()
2. toExternalForm()
3. toURI()

- java.net.URL has a toString() method. Print statements call toString() implicitly. public String toExternalForm()
- The toExternalForm() method converts a URL object to a string that can be used in an HTML. The toExternalForm() method returns a human-readable String representing the URL. It is identical to the toString() method. In fact, **all the toString() method does is return toExternalForm().**
- Finally, the **toURI() method** converts a **URL object to an equivalent URI object:**

public URI toURI() throws URISyntaxException

3.3 The URI Class

The URI Class

1. Constructing a URI
2. The Parts of the URI
3. Resolving Relative URIs
4. Equality and Comparison
5. String Representations

1. Constructing a URI

a. **public URI(String uri)** throws URISyntaxException

b. **public URI(String scheme, String schemeSpecificPart, String fragment)** throws URISyntaxException

c. **public URI(String scheme, String host, String path, String fragment)** throws URISyntaxException

d. **public URI(String scheme, String authority, String path, String query, String fragment)** throws URISyntaxException

e. **public URI(String scheme, String userInfo, String host, int port, String path, String query, String fragment)** throws URISyntaxException

- The **first 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("isbn:1-565-92870-9");

- If the string **argument does not follow URI syntax rules**—for example, if the URI begins with a colon—this constructor **throws a URISyntaxException**.
- The **second constructor** that takes a scheme specific part is **mostly used for nonhierarchical URIs**. The scheme is the URI's protocol, such as http, urn, tel, and so forth of ASCII letters and digits and the three characters +, -, and .. It must begin with a letter. Passing null for this argument omits the scheme, thus creating a relative URI.
- For example:

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

✓URI relative = new URI(null, "/javafaq/index.shtml", "today");

- The **third constructor** is used for hierarchical URIs such as http and ftp URLs. The host and path together (separated by a /) form the scheme-specific part for this URI.

<u>For example:</u>

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

This produces the URI http://www.ibiblio.org/javafaq/index.html#today.

- The **fourth constructor** is basically the same as the third, with the addition of a query string. For example:

✓URI today = new URI("http", "www.ibiblio.org", "/javafaq/index.html“, "referrer=cnet&date=2014-02-23", "today");

- The **fifth constructor** is the master hierarchical URI constructor that the previous two invoke. It divides the authority into separate user info, host, and port parts, each of which has its own syntax rules.

For example:

✓URI styles = new URI("ftp", "anonymous:elharo@ibiblio.org“, "ftp.oreilly.com", 21, "/pub/stylesheet", null, null);

- However, the resulting URI still has to follow all the usual rules for URIs; and again null can be passed for any argument to omit it from the result.

2. The Parts of the URI

- A URI reference has up to **three parts**: a scheme, a scheme-specific part, and a fragment identifier. The general format is:

scheme:scheme-specific-part:fragment

The URI class has **getter methods** that **return these three parts** of each URI object. The **getRawFoo()** methods **return the encoded forms** of the parts of the URI, while the equivalent **getFoo() methods first decode any percent escaped characters and then return the decoded part**:

public String getScheme()

public String getSchemeSpecificPart()

public String getRawSchemeSpecificPart()

public String getFragment()

public String getRawFragment()

- 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**:

✓**public boolean isAbsolute()**

- The **isOpaque() method** returns **false if the URI is hierarchical**, **true if it's not hierarchical**— that is, if it's opaque:

✓**public boolean isOpaque()**

- If the **URI is opaque, all you can get is the scheme, scheme-specific part, and fragment identifier. However, if the URI is hierarchical**, there are **getter methods for all the different parts of a hierarchical URI**:

✓public String getAuthority()

✓public String getFragment()

✓public String getHost()

✓public String getPath()

✓public String getPort()

✓public String getQuery()

✓public String getUserInfo()

- These above methods all return the **decoded parts**; in other words, percent escapes, such as %3C, are changed into the characters they represent, such as <. **If you want the raw, encoded parts of the URI**, there **are five parallel getRaw_Foo_()** methods:

✓public String getRawAuthority()

✓public String getRawFragment()

✓public String getRawPath()

✓public String getRawQuery()

✓public String getRawUserInfo()

Example 6: The parts of a URI

```
import java.net.*;
public class URISplitter {
public static void main(String args[]) {
try {
URI u = new URI("https://www.prime.edu.np/");
System.out.println("The URI is " + u);
if (u.isOpaque()) {//doesn't describe path to
resource
System.out.println("This is an opaque URI.");
System.out.println("The scheme is " +
u.getScheme());
System.out.println("The scheme specific part is "+
u.getSchemeSpecificPart());
System.out.println("The fragment ID is " +
u.getFragment());
}
else {
System.out.println("This is a hierarchical URI.");
System.out.println("The scheme is " +
u.getScheme());
System.out.println("The host is " + u.getHost());
System.out.println("The user info is " +
u.getUserInfo());
System.out.println("The port is " + u.getPort());
System.out.println("The path is " + u.getPath());
System.out.println("The query string is " +
u.getQuery());
System.out.println("The fragment ID is " +
u.getFragment());
}
} catch (URISyntaxException ex) {
System.err.println("Does not seem to be a URI.");
}}}
```

3. Resolving Relative URIs

- The URI class has three methods for converting back and forth between relative and absolute URIs:

public URI resolve(URI uri)

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);
```

- After they've executed, **resolved** contains the absolute URI

http://www.example.com/images/logo.png.

- the resolve() method resolves as much of the URI as it can and returns a new relative URI object as a result.
- For example, take these three statements:

URI top = new URI("javafaq/books/");

URI resolved = top.resolve("jnp3/examples/07/index.html");

- After they've executed, **resolved** now contains the relative URI **javafaq/books/jnp3/examples/07/index.html** with no scheme or authority.
- The **relativize() method** creates a new URI object from the uri argument that is relative to the invoking URI. The argument is not changed. For example:

URI absolute = new URI("http://www.example.com/images/logo.png");

URI top = new URI("http://www.example.com/");

URI relative = top.relativize(absolute);

- The URI object relative now contains the relative URI **images/logo.png.**

4. Equality and Comparison

The **hashCode()** (integer) method is consistent with **equals(Object o)** (boolean). Equal URIs do have the same hash code and unequal URIs are fairly unlikely to share the same hash code.

compareTo(Object o) method compares two URIs.

5. String Representations

- Two methods convert URI objects to strings, **toString()** and **toASCIIString():**

public String toString()

public String toASCIIString()

- The **toString() method returns an decoded string form of the URI**.
- The **toASCIIString() method returns an encoded string form of the URI**.

3.4 x-www-form-urlencoded

- In the case of x-www-form-urlencoded, an alphanumeric and reserved character is url encoded e.g. space is replaced by %20.
- Characters used in URLs must come from a fixed subset of ASCII, specifically:
  - The capital letters A–Z
  - The lowercase letters a–z
  - The digits 0–9
  - The punctuation characters - _ . ! ~ * ' (and ,)

1. URLEncoder

2. URLDecoder

1. URLEncoder

- To URL encode a string, pass the string and the character set name to the URLEncoder.encode() method. For example:

String encoded = URLEncoder.encode("This*string*has*asterisks", "UTF-8");

- URLEncoder.encode() returns a copy of the input string with a few changes. Any **nonalphanumeric characters are converted into % sequences (except the space, underscore, hyphen, period, and asterisk characters)**. It also encodes all non-ASCII characters. The **space is converted into a plus sign.**

Example 7: x-www-form-urlencoded strings

```java
import java.io.*;
import java.net.*;
public class Example7 {
public static void main(String[] args) {
try {
System.out.println(URLEncoder.encode("This string has spaces","UTF-8"));
System.out.println(URLEncoder.encode("This*string*has*asterisks","UTF-8"));
System.out.println(URLEncoder.encode("This%string%has%percent%signs","UTF-8"));
System.out.println(URLEncoder.encode("This+string+has+pluses","UTF-8"));
System.out.println(URLEncoder.encode("This/string/has/slashes","UTF-8"));
System.out.println(URLEncoder.encode("This\"string\"has\"quote\"marks","UTF-8"));
System.out.println(URLEncoder.encode("This:string:has:colons","UTF-8"));
System.out.println(URLEncoder.encode("This~string~has~tildes","UTF-8"));
System.out.println(URLEncoder.encode("This(string)has(parentheses)","UTF-8"));
System.out.println(URLEncoder.encode("This.string.has.periods","UTF-8"));
System.out.println(URLEncoder.encode("This=string=has=equals=signs","UTF-8"));
System.out.println(URLEncoder.encode("This&string&has&ampersands","UTF-8"));
System.out.println(URLEncoder.encode("Thiséstringéhasénon-ASCII characters", "UTF-8"));
} catch (UnsupportedEncodingException ex) {
System.out.println("Encoding not supported.");
}}}
```

2. URLDecoder

- The corresponding URLDecoder class has a static decode() method that decodes strings encoded in x-www-form-url-encoded format. That is, it converts all plus signs to spaces and all percent escapes to their corresponding character:
- public static String decode(String s, String encoding) throws UnsupportedEncodingException
- We can pass an entire URL to it rather than splitting it into pieces first. For example:

```
try {
System.out.println(URLDecoder.decode("This+string+has+spaces","UTF-8"));
System.out.println(URLDecoder.decode("This%25string%25has%25percent%25signs","UTF-8"));
}catch (UnsupportedEncodingException ex) {
System.out.println("Encoding not supported.");}
```

3.5 Proxies

- The **local client sends a request to the Proxy Server**, and then the **Proxy Server sends a request to the remote Server**. The **remote server sends the result to the proxy, and the proxy returns to the client.**

Why use a proxy?

1. Security considerations: Use proxy to shield client side details from remote server.

2. Access control: The proxy can have a filtering function to block sites that do not want the client to visit.

3. Performance considerations: When multiple clients request a unified resource, proxy can cache that resource instead of requesting the remote server every time.

- Java programmers can use the URL class to do most of the communication with the proxy server or protocol.

<u>System Properties</u>

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

- If you use http proxy, use http.proxyHost to set the proxy host and http.proxyPort to set the proxy port. Hosts that do not want to be proxied can use http.nonProxyHosts.

- The Proxy class allows more fine-grained control of proxy servers from within a Java program. Specifically, it allows you to choose different proxy servers for different remote hosts.
- There are still only three kinds of proxies, HTTP, SOCKS, and direct connections (no proxy at all), represented by three constants in the Proxy.Type enum:

✓**Proxy.Type.DIRECT**

✓**Proxy.Type.HTTP**

✓**Proxy.Type.SOCKS**

ProxySelector Class

There is only one ProxySelector instance per JVM, used to locate different proxy servers for different connections.

The default ps only detects some system properties (such as those mentioned above) and URL protocol and then decides how to connect to the remote server.

3.6 Communicating with Server-Side Programs Through GET

The HTML for the form looks like this:

```
<form class="center mb1em" action="search" method="GET">
<input style="*vertical-align:middle;" size="45" name="q" value="" class="qN">
<input style="*vertical-align:middle; *padding-top:1px;" value="Search"
class="btn" type="submit">
<a href="search?type=advanced"><span class="advN">advanced</span></a>
</form>
```

There are only **two input fields** in this form: the **Submit button and a text field named q**. Thus, to submit a search request, you just need to **append q=searchTerm to http://www.dmoz.org/search.**

For example, to search for "**java**", We would open a connection to the URL **http://www.dmoz.org/search/?q=java** and read the resulting input stream.

3.7 Accessing Password-Protected Sites

- Many popular sites require a username and password for access. Some sites, such as the W3C member pages, implement this through HTTP authentication.
- Java's URL class can access sites that use HTTP authentication.

The Authenticator Class:

- 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

The PasswordAuthentication Class

- PasswordAuthentication is a very simple final class that supports two read-only properties: **username and password**.
- The **username is a String**. The **password is a char array** so that the password can be erased when it's no longer needed.
- Both username and password are set in the constructor:

public PasswordAuthentication(String userName, char[] password)

Each is accessed via a getter method:

- **public String getUserName()**
- **public char[] getPassword()**

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

- This lightweight component behaves almost exactly like a text field.
- JPasswordField also stores the passwords as a char array so that when we are done with the password. It provides the getPassword() method to return this:

public char[] getPassword()

THANK YOU FOR YOUR ATTENTION