JAVA - URL PROCESSING

URL stands for Uniform Resource Locator and represents a resource on the World Wide Web, such as a Web page or FTP directory.

This section shows you how to write Java programs that communicate with a URL. A URL can be broken down into parts, as follows:

```
protocol://host:port/path?query#ref
```

Examples of protocols include HTTP, HTTPS, FTP, and File. The path is also referred to as the filename, and the host is also called the authority.

The following is a URL to a Web page whose protocol is HTTP:

```
http://www.amrood.com/index.htm?language=en#j2se
```

Notice that this URL does not specify a port, in which case the default port for the protocol is used. With HTTP, the default port is 80.

URL Class Methods:

The **java.net.URL** class represents a URL and has complete set of methods to manipulate URL in Java.

The URL class has several constructors for creating URLs, including the following:

SN	Methods with Description
1	**public URL*Stringprotocol, Stringhost, intport, Stringfile* throws MalformedURLException.** Creates a URL by putting together the given parts.
2	**public URL*Stringprotocol, Stringhost, Stringfile* throws MalformedURLException** Identical to the previous constructor, except that the default port for the given protocol is used.
3	**public URL*Stringurl* throws MalformedURLException** Creates a URL from the given String
4	**public URL*URLcontext, Stringurl* throws MalformedURLException** Creates a URL by parsing the together the URL and String arguments

The URL class contains many methods for accessing the various parts of the URL being represented. Some of the methods in the URL class include the following:

SN	Methods with Description
1	**public String getPath** Returns the path of the URL.
2	**public String getQuery** Returns the query part of the URL.
3	**public String getAuthority** Returns the authority of the URL.
4	**public int getPort** Returns the port of the URL.

5 **public int getDefaultPort**
Returns the default port for the protocol of the URL.

6 **public String getProtocol**
Returns the protocol of the URL.

7 **public String getHost**
Returns the host of the URL.

8 **public String getHost**
Returns the host of the URL.

9 **public String getFile**
Returns the filename of the URL.

10 **public String getRef**
Returns the reference part of the URL.

11 **public URLConnection openConnection throws IOException**
Opens a connection to the URL, allowing a client to communicate with the resource.

Example:

The following URLDemo program demonstrates the various parts of a URL. A URL is entered on the command line, and the URLDemo program outputs each part of the given URL.

```
// File Name : URLDemo.java

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

public class URLDemo
{
   public static void main(String [] args)
   {
      try
      {
         URL url = new URL("http://www.amrood.com/index.htm?language=en#j2se");

         System.out.println("URL is " + url.toString());
         System.out.println("protocol is " + url.getProtocol());
         System.out.println("authority is " + url.getAuthority());
         System.out.println("file name is " + url.getFile());
         System.out.println("host is " + url.getHost());
         System.out.println("path is " + url.getPath());
         System.out.println("port is " + url.getPort());
         System.out.println("default port is " + url.getDefaultPort());
         System.out.println("query is " + url.getQuery());
         System.out.println("ref is " + url.getRef());
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

A sample run of the this program would produce the following result:

```
URL is http://www.amrood.com/index.htm?language=en#j2se
protocol is http
authority is www.amrood.com
file name is /index.htm?language=en
host is www.amrood.com
path is /index.htm
port is -1
default port is 80
```

```
query is language=en
ref is j2se
```

URLConnections Class Methods:

The openConnection method returns a **java.net.URLConnection**, an abstract class whose subclasses represent the various types of URL connections.

For example:

- If you connect to a URL whose protocol is HTTP, the openConnection method returns an HttpURLConnection object.
- If you connect to a URL that represents a JAR file, the openConnection method returns a JarURLConnection object.
- etc...

The URLConnection class has many methods for setting or determining information about the connection, including the following:

SN	Methods with Description
1	**Object getContent** Retrieves the contents of this URL connection.
2	**Object getContent*Class*[]*classes*** Retrieves the contents of this URL connection.
3	**String getContentEncoding** Returns the value of the content-encoding header field.
4	**int getContentLength** Returns the value of the content-length header field.
5	**String getContentType** Returns the value of the content-type header field.
6	**int getLastModified** Returns the value of the last-modified header field.
7	**long getExpiration** Returns the value of the expires header field.
8	**long getIfModifiedSince** Returns the value of this object's ifModifiedSince field.
9	**public void setDoInput*booleaninput*** Passes in true to denote that the connection will be used for input. The default value is true because clients typically read from a URLConnection.
10	**public void setDoOutput*booleanoutput*** Passes in true to denote that the connection will be used for output. The default value is false because many types of URLs do not support being written to.
11	**public InputStream getInputStream throws IOException** Returns the input stream of the URL connection for reading from the resource.
12	**public OutputStream getOutputStream throws IOException** Returns the output stream of the URL connection for writing to the resource
13	**public URL getURL** Returns the URL that this URLConnection object is connected to

Example:

The following URLConnectionDemo program connects to a URL entered from the command line.

If the URL represents an HTTP resource, the connection is cast to HttpURLConnection, and the data in the resource is read one line at a time.

```
// File Name : URLConnDemo.java

import java.net.*;
import java.io.*;
public class URLConnDemo
{
   public static void main(String [] args)
   {
      try
      {
         URL url = new URL("http://www.amrood.com");
         URLConnection urlConnection = url.openConnection();
         HttpURLConnection connection = null;
         if(urlConnection instanceof HttpURLConnection)
         {
            connection = (HttpURLConnection) urlConnection;
         }
         else
         {
            System.out.println("Please enter an HTTP URL.");
            return;
         }
         BufferedReader in = new BufferedReader(
         new InputStreamReader(connection.getInputStream()));
         String urlString = "";
         String current;
         while((current = in.readLine()) != null)
         {
            urlString += current;
         }
         System.out.println(urlString);
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

A sample run of the this program would produce the following result:

```
$ java URLConnDemo

    a complete HTML content of home page of amrood.com.....
```