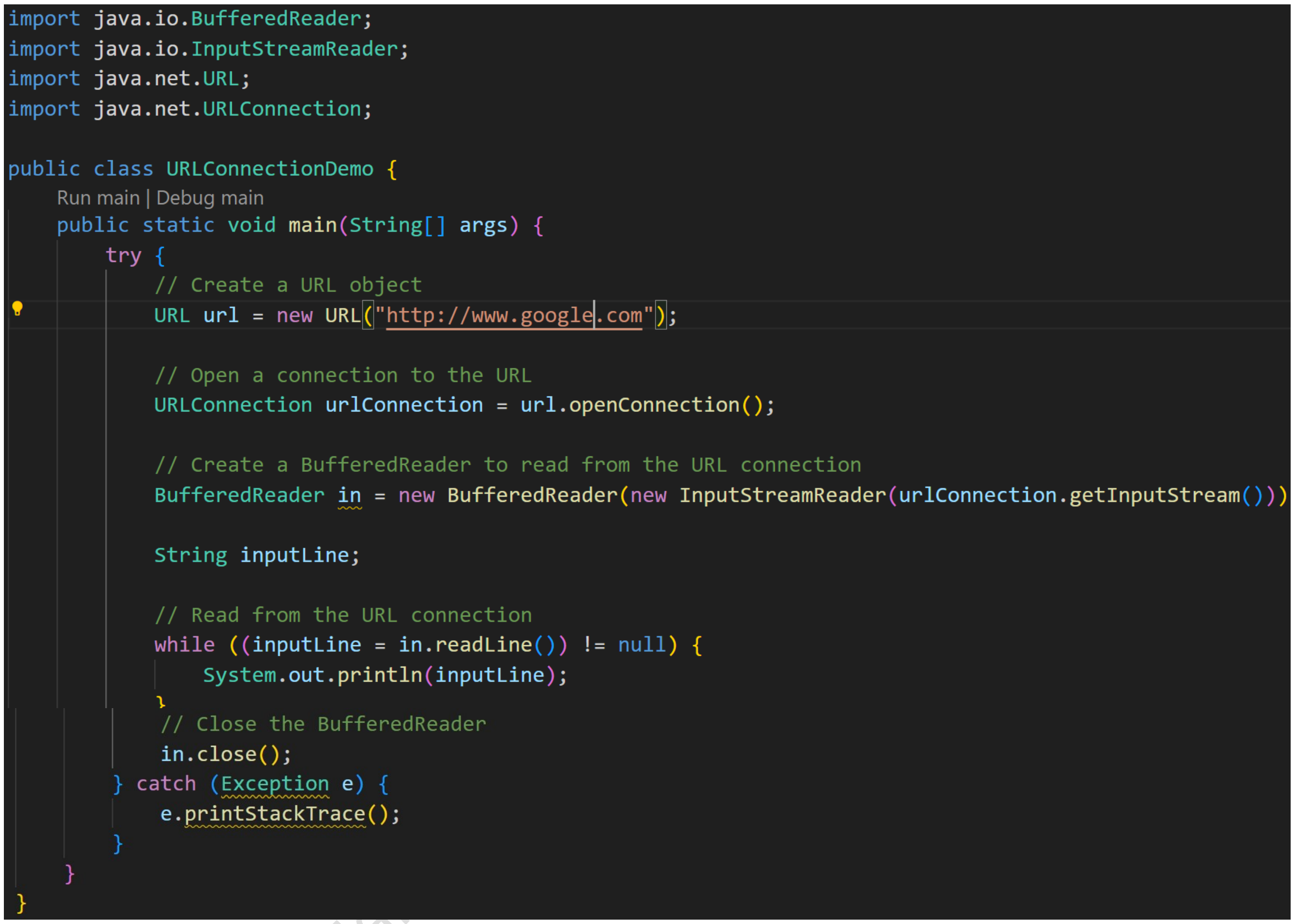
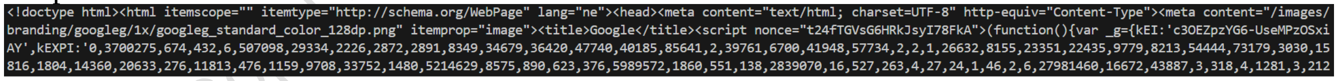
URL Connection
URLConnection is a class in Java that provides a high-level interface for working with resources identified by URLs. It's part of the java.net package and is commonly used to establish connections and interact with remote servers over various network protocols such as HTTP, HTTPS, FTP, etc. URLConnection abstracts the process of connecting to a resource and performing operations like reading from or writing to it. It provides methods for setting request properties, reading response headers, and managing input and output streams for the underlying connection
Here are the basic steps to use the URLConnection class in Java:
1. **Create a URL Object:** Create a URL object by providing the URL of the resource you want to connect to.
2. **Open Connection:** Open a connection to the URL using the `openConnection()` method of the URL object. This method returns an instance of URLConnection.
3. **Configure Connection:** Set any necessary request properties, such as headers or timeouts, using methods like `setRequestProperty()`.
4. **Retrieve Data:** Open an input stream from the URLConnection to read the data from the remote resource.
5. **Read Data:** Read the data from the input stream using standard Java I/O operations.
6. **Close Resources:** Close the input stream and release any resources associated with the URLConnection

//Program to read the content of a web page



```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;

public class URLConnectionDemo {
    Run main | Debug main
    public static void main(String[] args) {
        try {
            // Create a URL object
            URL url = new URL("http://www.google.com");

            // Open a connection to the URL
            URLConnection urlConnection = url.openConnection();

            // Create a BufferedReader to read from the URL connection
            BufferedReader in = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()))

            String inputLine;

            // Read from the URL connection
            while ((inputLine = in.readLine()) != null) {
                System.out.println(inputLine);
            }
            // Close the BufferedReader
            in.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:



```
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ne"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/
branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="t24fTGVsG6HRkJsyI78FkA">(function(){var _g={kEI:'c3OEZpzYG6-UseMPzOSxi
AY',kEXPI:'0,3700275,674,432,6,507098,29334,2226,2872,2891,8349,34679,36420,47740,40185,85641,2,39761,6700,41948,57734,2,2,1,26632,8155,23351,22435,9779,8213,54444,73179,3030,15
816,1804,14360,20633,276,11813,476,1159,9708,33752,1480,5214629,8575,890,623,376,5989572,1860,551,138,2839070,16,527,263,4,27,24,1,46,2,6,27981460,16672,43887,3,318,4,1281,3,212
```

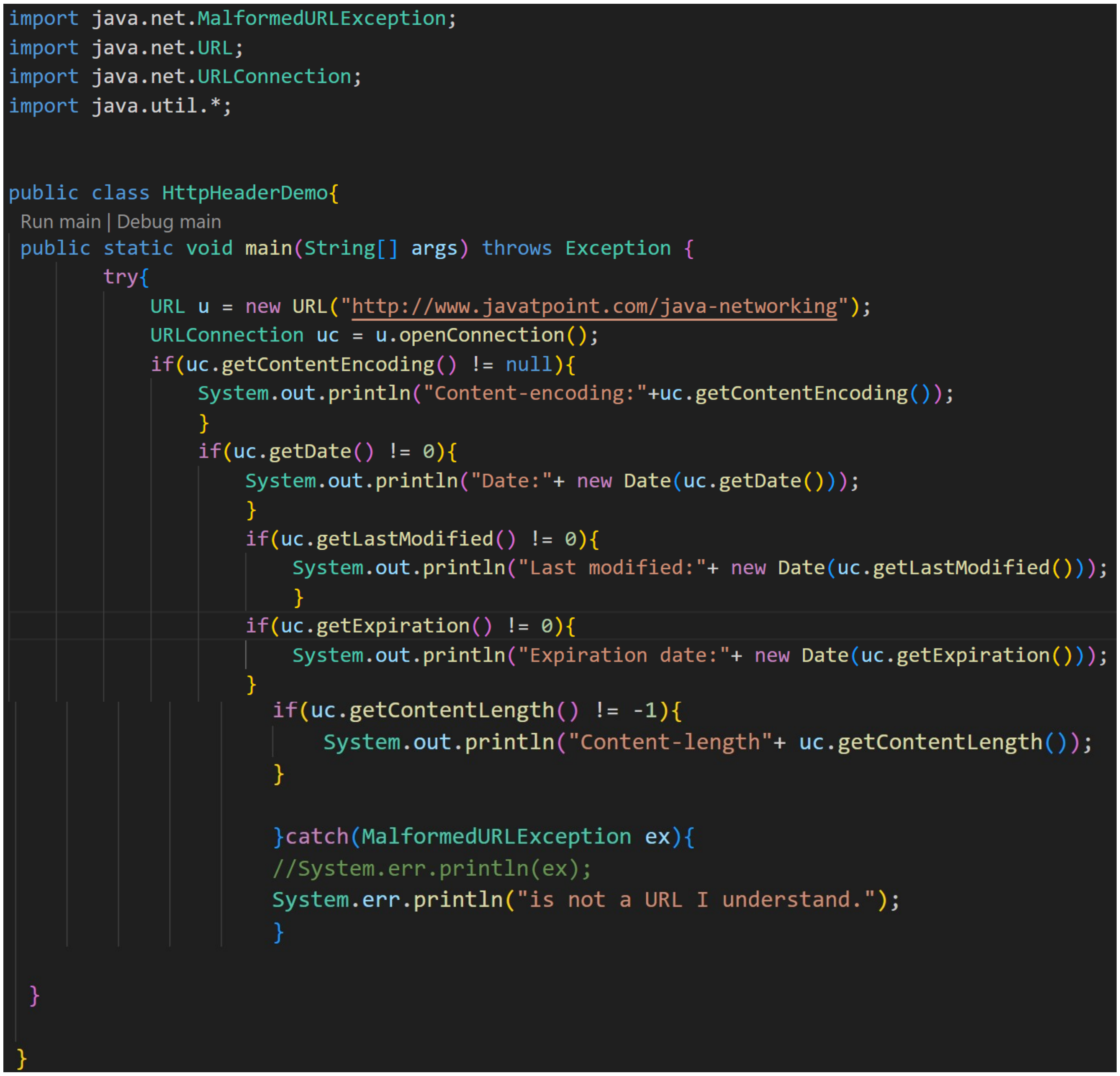
Retrieving specific Header Fields

Mention the methods to retrieve specific MIME Header Fields.

The methods to retrieve specific MIME Header Fields are listed below:

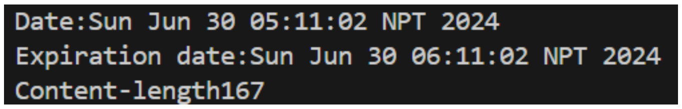
1. **public String getContentType():** This method returns the MIME content type of the data.
2. **public int getContentLength():** This method tells you how many bytes there are in the content.
3. **public String getContentEncoding():** This method return a String that tells you how the content is encoded.
4. **public long getDate():** The getDate() method returns a long that tells you when the document was sent.

5. **public long getExpiration():** Some documents have server-based expiration dates that indicate when the document should be deleted from then cache and reloaded from the server.
6. **public long getLastModified()**: The final date method, getLastModified(), returns the date on which the document was last modified.



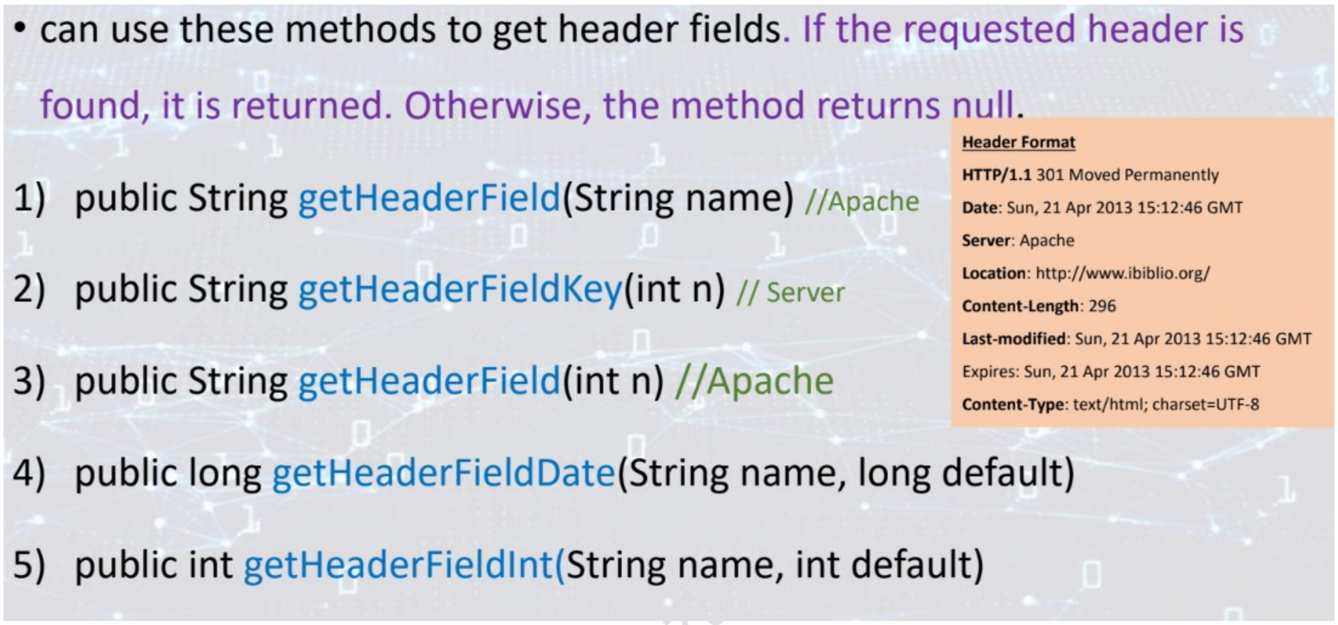
```
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.*;


public class HttpHeaderDemo{
 Run main | Debug main
 public static void main(String[] args) throws Exception {
        try{
            URL u = new URL("http://www.javatpoint.com/java-networking");
            URLConnection uc = u.openConnection();
            if(uc.getContentEncoding() != null){
                System.out.println("Content-encoding:"+uc.getContentEncoding());
                }
                if(uc.getDate() != 0){
                    System.out.println("Date:"+ new Date(uc.getDate()));
                    }
                    if(uc.getLastModified() != 0){
                        System.out.println("Last modified:"+ new Date(uc.getLastModified()));
                        }
                    if(uc.getExpiration() != 0){
                        System.out.println("Expiration date:"+ new Date(uc.getExpiration()));
                    }
                       if(uc.getContentLength() != -1){
                           System.out.println("Content-length"+ uc.getContentLength());
                       }

                       }catch(MalformedURLException ex){
                       //System.err.println(ex);
                       System.err.println("is not a URL I understand.");
                       }

  }

 }
```

Output:



```
Date:Sun Jun 30 05:11:02 NPT 2024
Expiration date:Sun Jun 30 06:11:02 NPT 2024
Content-length167
```

Retriving Arbitrary Header Fields

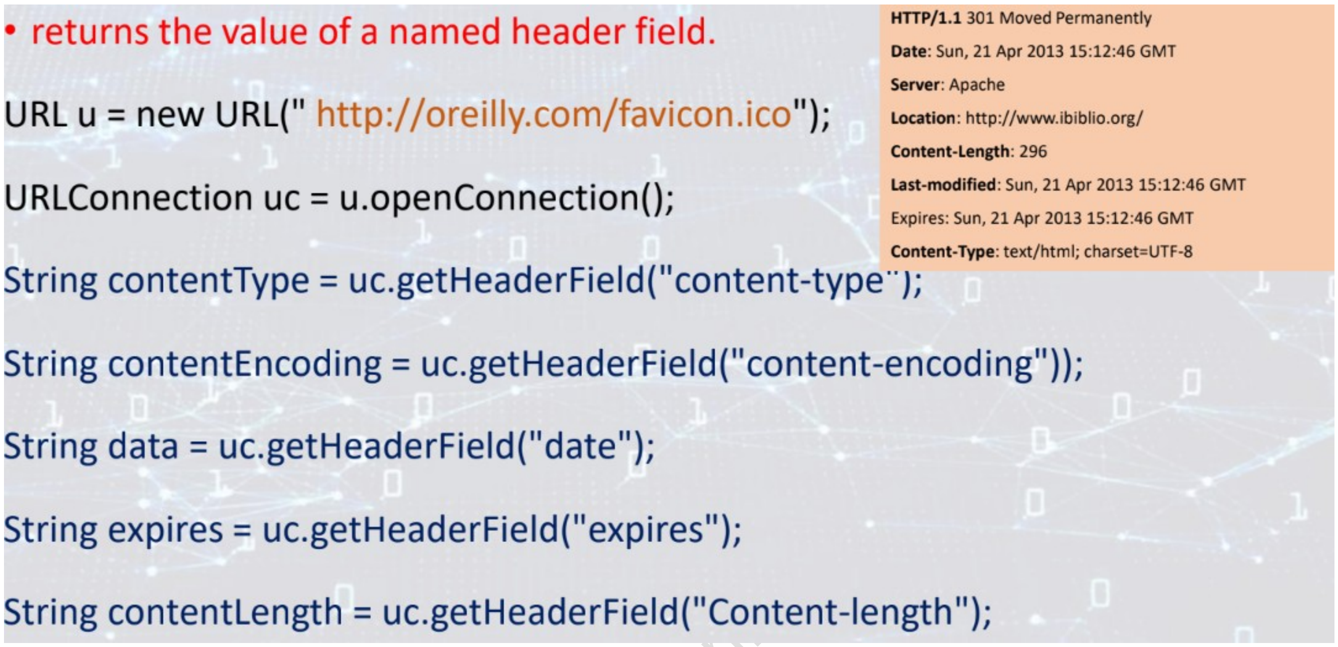


- can use these methods to get header fields. If the requested header is found, it is returned. Otherwise, the method returns null.

1) public String getHeaderField(String name) //Apache
2) public String getHeaderFieldKey(int n) // Server
3) public String getHeaderField(int n) //Apache
4) public long getHeaderFieldDate(String name, long default)
5) public int getHeaderFieldInt(String name, int default)

```
Header Format
HTTP/1.1 301 Moved Permanently
Date: Sun, 21 Apr 2013 15:12:46 GMT
Server: Apache
Location: http://www.ibiblio.org/
Content-Length: 296
Last-modified: Sun, 21 Apr 2013 15:12:46 GMT
Expires: Sun, 21 Apr 2013 15:12:46 GMT
Content-Type: text/html; charset=UTF-8
```

Public String getHeaderField(String name)

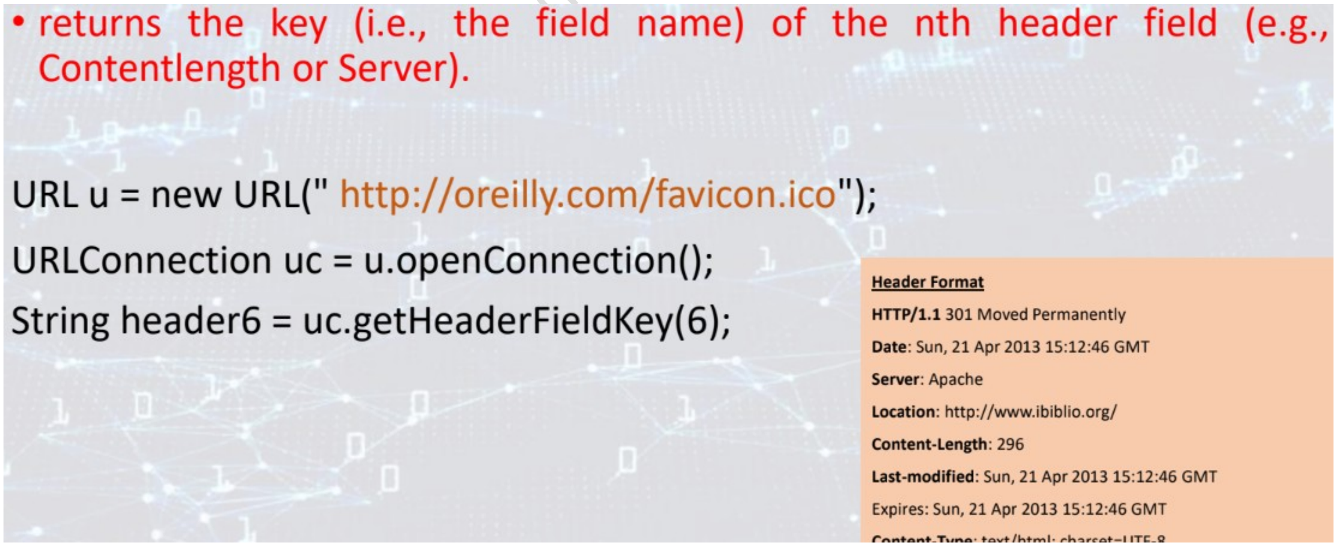


- returns the value of a named header field.

```
URL u = new URL(" http://oreilly.com/favicon.ico");
URLConnection uc = u.openConnection();
String contentType = uc.getHeaderField("content-type");
String contentEncoding = uc.getHeaderField("content-encoding"));
String data = uc.getHeaderField("date");
String expires = uc.getHeaderField("expires");
String contentLength = uc.getHeaderField("Content-length");
```

```
HTTP/1.1 301 Moved Permanently
Date: Sun, 21 Apr 2013 15:12:46 GMT
Server: Apache
Location: http://www.ibiblio.org/
Content-Length: 296
Last-modified: Sun, 21 Apr 2013 15:12:46 GMT
Expires: Sun, 21 Apr 2013 15:12:46 GMT
Content-Type: text/html; charset=UTF-8
```

Public String getHeaderFieldKey(int n)



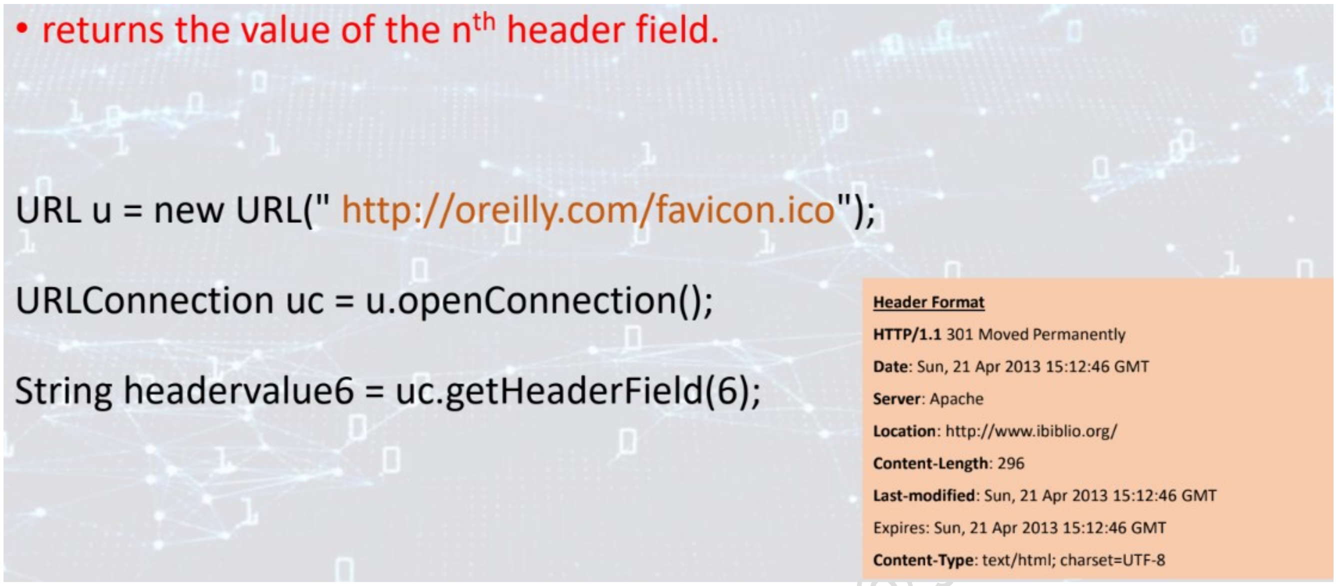
- returns the key (i.e., the field name) of the nth header field (e.g., Contentlength or Server).

```
URL u = new URL(" http://oreilly.com/favicon.ico");
URLConnection uc = u.openConnection();
String header6 = uc.getHeaderFieldKey(6);
```

Header Format

```
HTTP/1.1 301 Moved Permanently
Date: Sun, 21 Apr 2013 15:12:46 GMT
Server: Apache
Location: http://www.ibiblio.org/
Content-Length: 296
Last-modified: Sun, 21 Apr 2013 15:12:46 GMT
Expires: Sun, 21 Apr 2013 15:12:46 GMT
Content-Type: text/html; charset=UTF-8
```

Public String getHeaderField(int n)



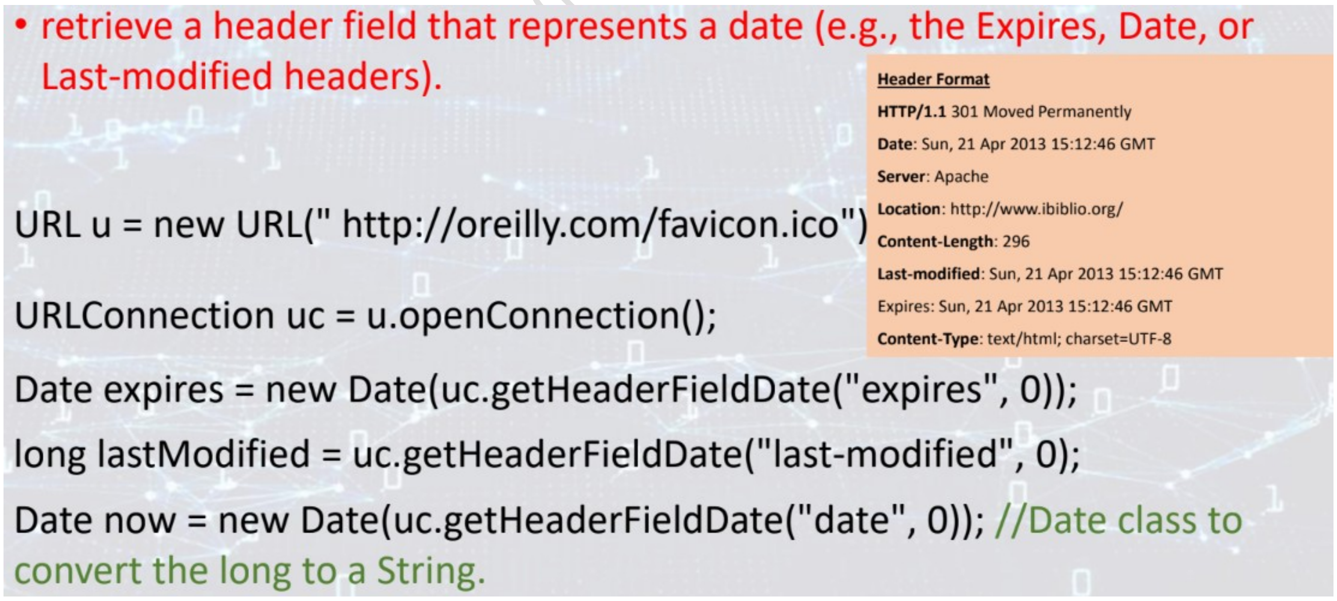
- returns the value of the nth header field.

```
URL u = new URL(" http://oreilly.com/favicon.ico");

URLConnection uc = u.openConnection();

String headervalue6 = uc.getHeaderField(6);
```

Header Format

HTTP/1.1 301 Moved Permanently
Date: Sun, 21 Apr 2013 15:12:46 GMT
Server: Apache
Location: http://www.ibiblio.org/
Content-Length: 296
Last-modified: Sun, 21 Apr 2013 15:12:46 GMT
Expires: Sun, 21 Apr 2013 15:12:46 GMT
Content-Type: text/html; charset=UTF-8

public long getHeaderFieldDate(String name, long default)



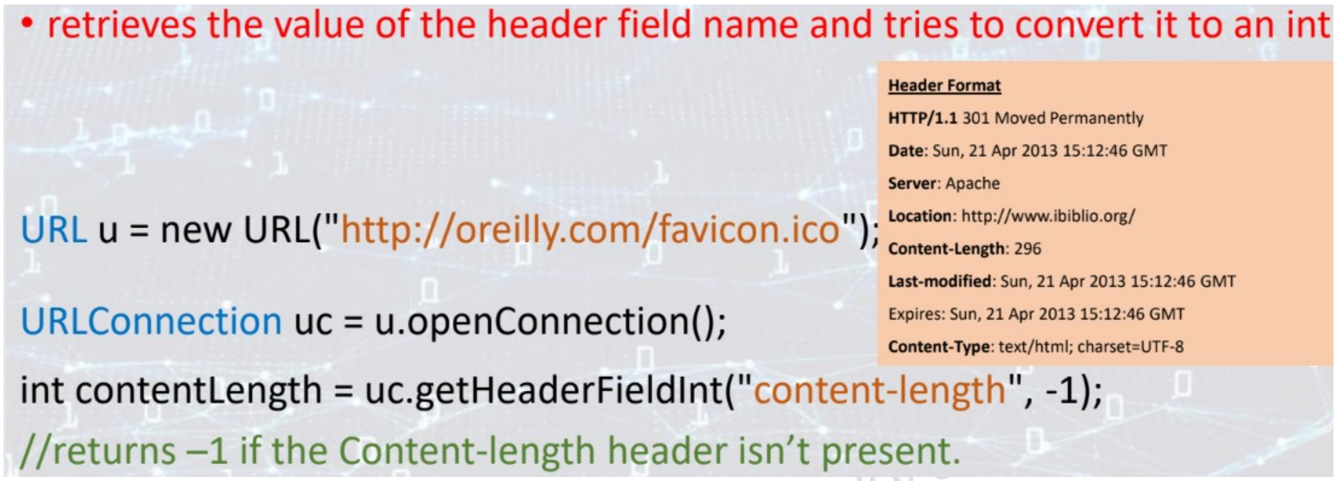
- retrieve a header field that represents a date (e.g., the Expires, Date, or Last-modified headers).

Header Format

HTTP/1.1 301 Moved Permanently
Date: Sun, 21 Apr 2013 15:12:46 GMT
Server: Apache
Location: http://www.ibiblio.org/
Content-Length: 296
Last-modified: Sun, 21 Apr 2013 15:12:46 GMT
Expires: Sun, 21 Apr 2013 15:12:46 GMT
Content-Type: text/html; charset=UTF-8

```
URL u = new URL(" http://oreilly.com/favicon.ico")

URLConnection uc = u.openConnection();

Date expires = new Date(uc.getHeaderFieldDate("expires", 0));
long lastModified = uc.getHeaderFieldDate("last-modified", 0);
Date now = new Date(uc.getHeaderFieldDate("date", 0)); //Date class to
convert the long to a String.
```

public int getHeaderFieldInt(String name, int default)

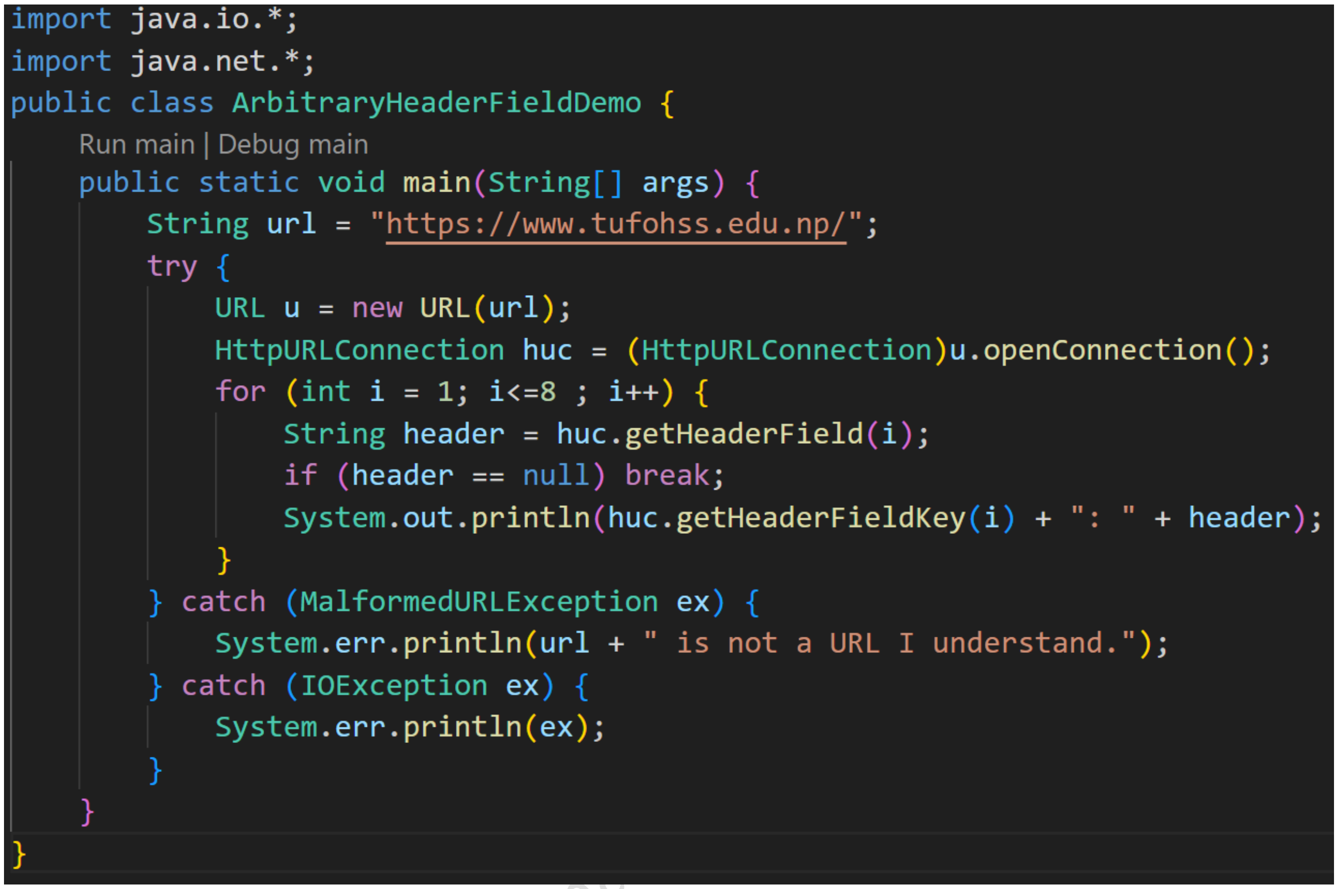


• retrieves the value of the header field name and tries to convert it to an int

```
URL u = new URL("http://oreilly.com/favicon.ico");
URLConnection uc = u.openConnection();
int contentLength = uc.getHeaderFieldInt("content-length", -1);
//returns –1 if the Content-length header isn't present.
```

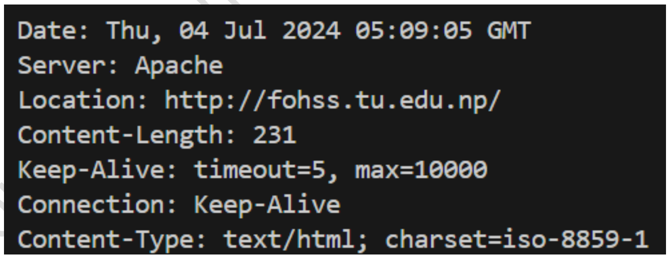
Header Format

```
HTTP/1.1 301 Moved Permanently
Date: Sun, 21 Apr 2013 15:12:46 GMT
Server: Apache
Location: http://www.ibiblio.org/
Content-Length: 296
Last-modified: Sun, 21 Apr 2013 15:12:46 GMT
Expires: Sun, 21 Apr 2013 15:12:46 GMT
Content-Type: text/html; charset=UTF-8
```



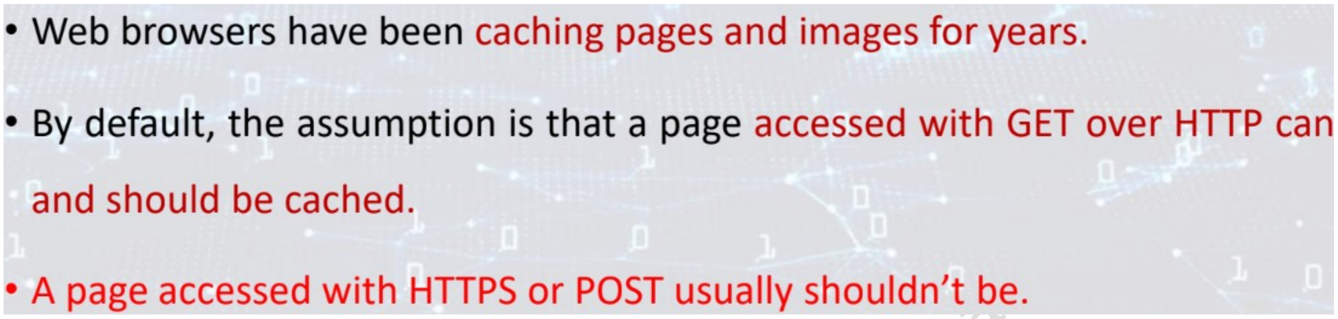
```
import java.io.*;
import java.net.*;
public class ArbitraryHeaderFieldDemo {
    Run main | Debug main
    public static void main(String[] args) {
        String url = "https://www.tufohss.edu.np/";
        try {
            URL u = new URL(url);
            HttpURLConnection huc = (HttpURLConnection)u.openConnection();
            for (int i = 1; i<=8 ; i++) {
                String header = huc.getHeaderField(i);
                if (header == null) break;
                System.out.println(huc.getHeaderFieldKey(i) + ": " + header);
            }
        } catch (MalformedURLException ex) {
            System.err.println(url + " is not a URL I understand.");
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

Output:

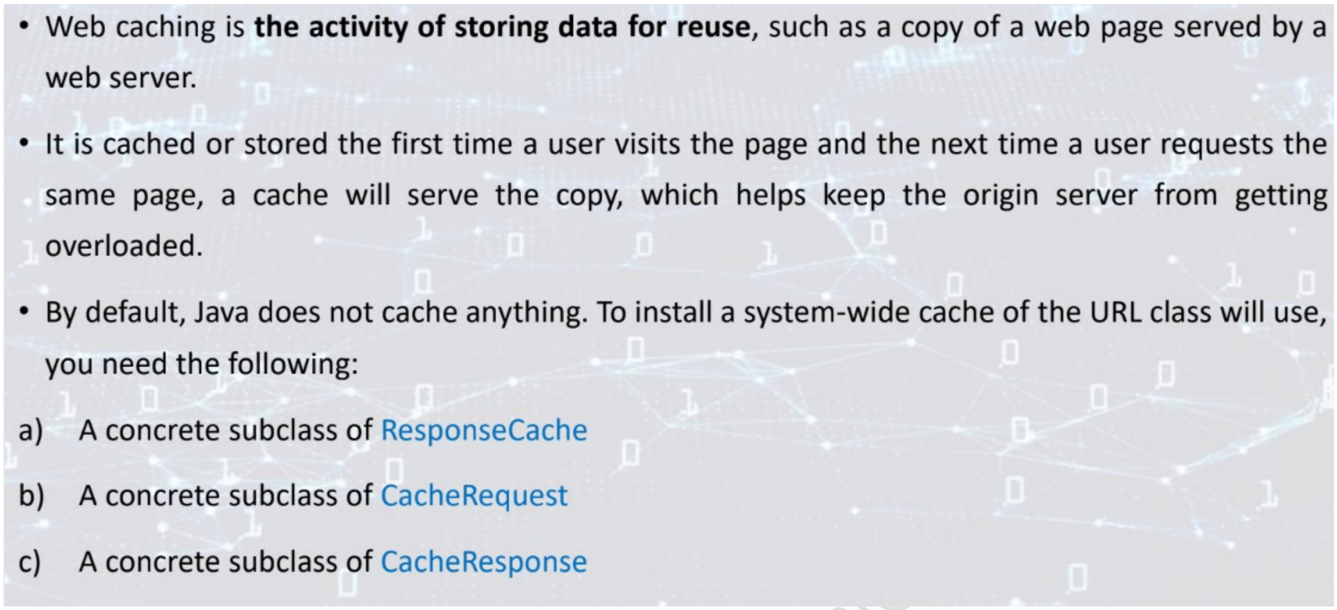


```
Date: Thu, 04 Jul 2024 05:09:05 GMT
Server: Apache
Location: http://fohss.tu.edu.np/
Content-Length: 231
Keep-Alive: timeout=5, max=10000
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

Cache: Web Cache for Java



- Web browsers have been caching pages and images for years.
- By default, the assumption is that a page accessed with GET over HTTP can and should be cached.
- A page accessed with HTTPS or POST usually shouldn't be.



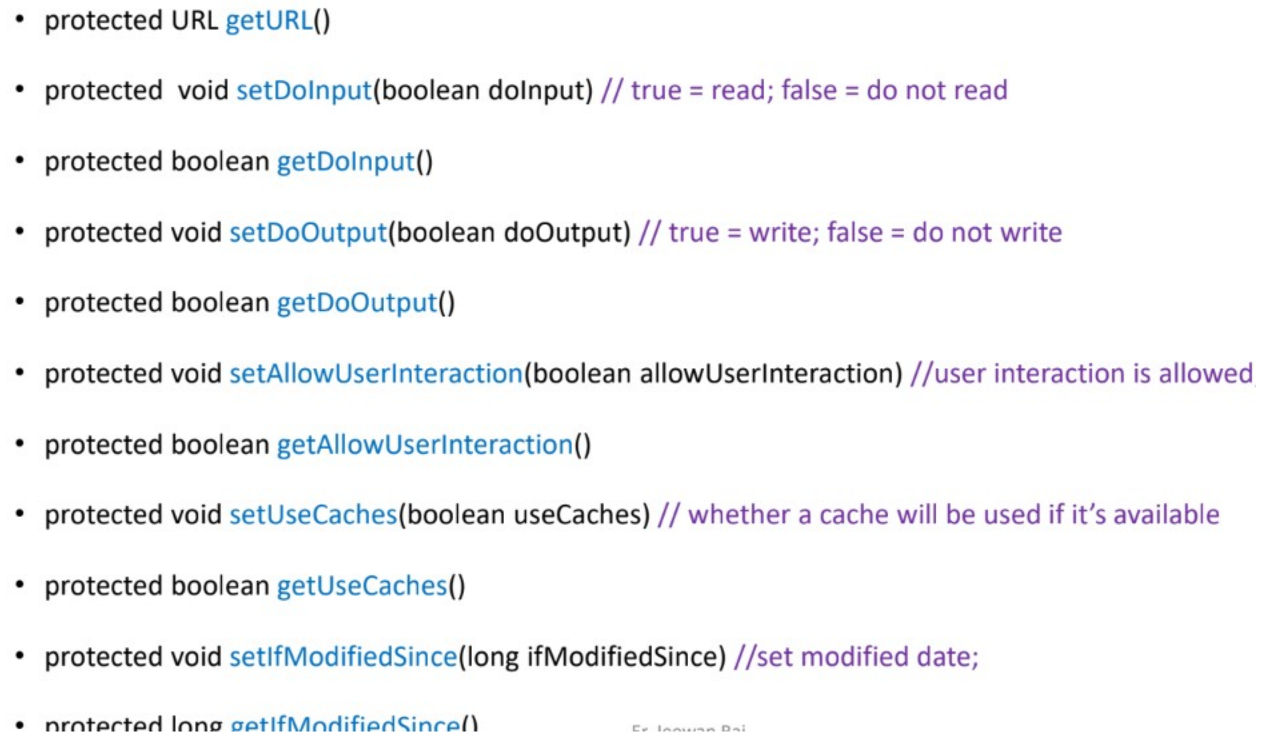
- Web caching is **the activity of storing data for reuse**, such as a copy of a web page served by a web server.
- It is cached or stored the first time a user visits the page and the next time a user requests the same page, a cache will serve the copy, which helps keep the origin server from getting overloaded.
- By default, Java does not cache anything. To install a system-wide cache of the URL class will use, you need the following:

a) A concrete subclass of ResponseCache

b) A concrete subclass of CacheRequest

c) A concrete subclass of CacheResponse

Configuring the connection

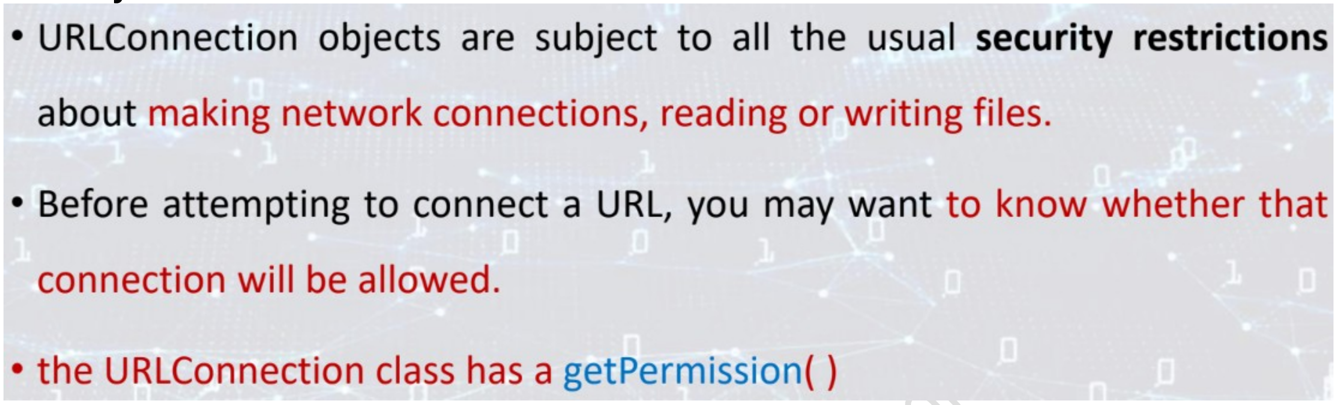


- protected URL getURL()
- protected void setDoInput(boolean doInput) // true = read; false = do not read
- protected boolean getDoInput()
- protected void setDoOutput(boolean doOutput) // true = write; false = do not write
- protected boolean getDoOutput()
- protected void setAllowUserInteraction(boolean allowUserInteraction) //user interaction is allowed
- protected boolean getAllowUserInteraction()
- protected void setUseCaches(boolean useCaches) // whether a cache will be used if it's available
- protected boolean getUseCaches()
- protected void setIfModifiedSince(long ifModifiedSince) //set modified date;
- protected long getIfModifiedSince()

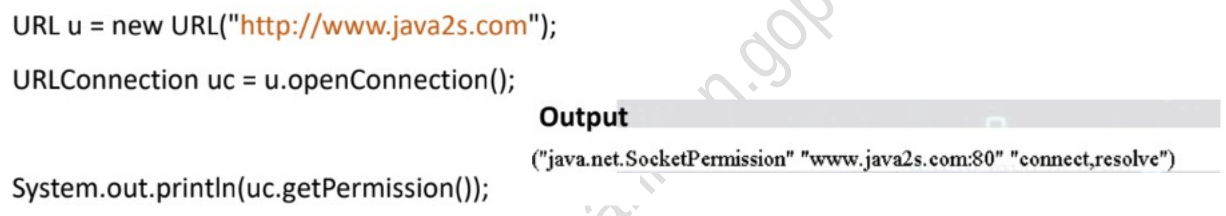
Er. Jeewan Rai

*protected long getIfModifiedSince()

Security consideration for URL Connection



- URLConnection objects are subject to all the usual **security restrictions** about making network connections, reading or writing files.
- Before attempting to connect a URL, you may want to know whether that connection will be allowed.
- the URLConnection class has a getPermission( )

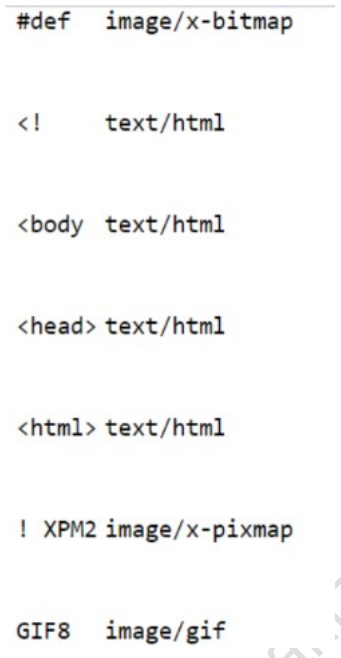


```
URL u = new URL("http://www.java2s.com");
URLConnection uc = u.openConnection();

System.out.println(uc.getPermission());
```

Output

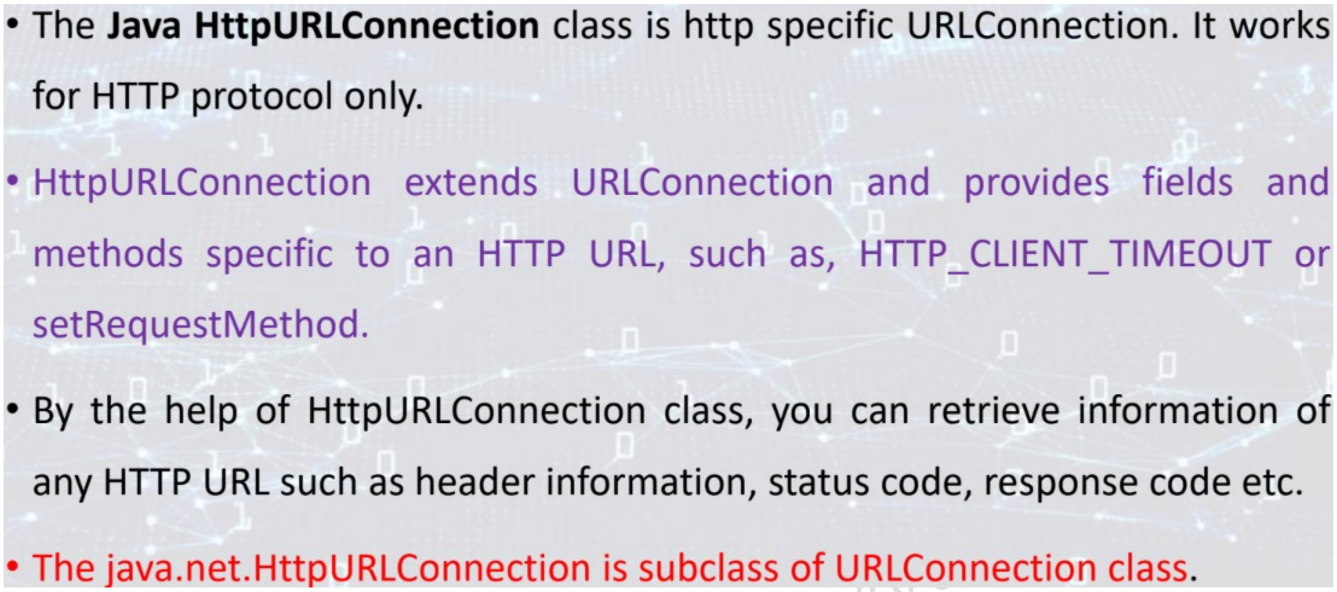
```
("java.net.SocketPermission" "www.java2s.com:80" "connect,resolve")
```

Guessing MIME Media Types
A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes.



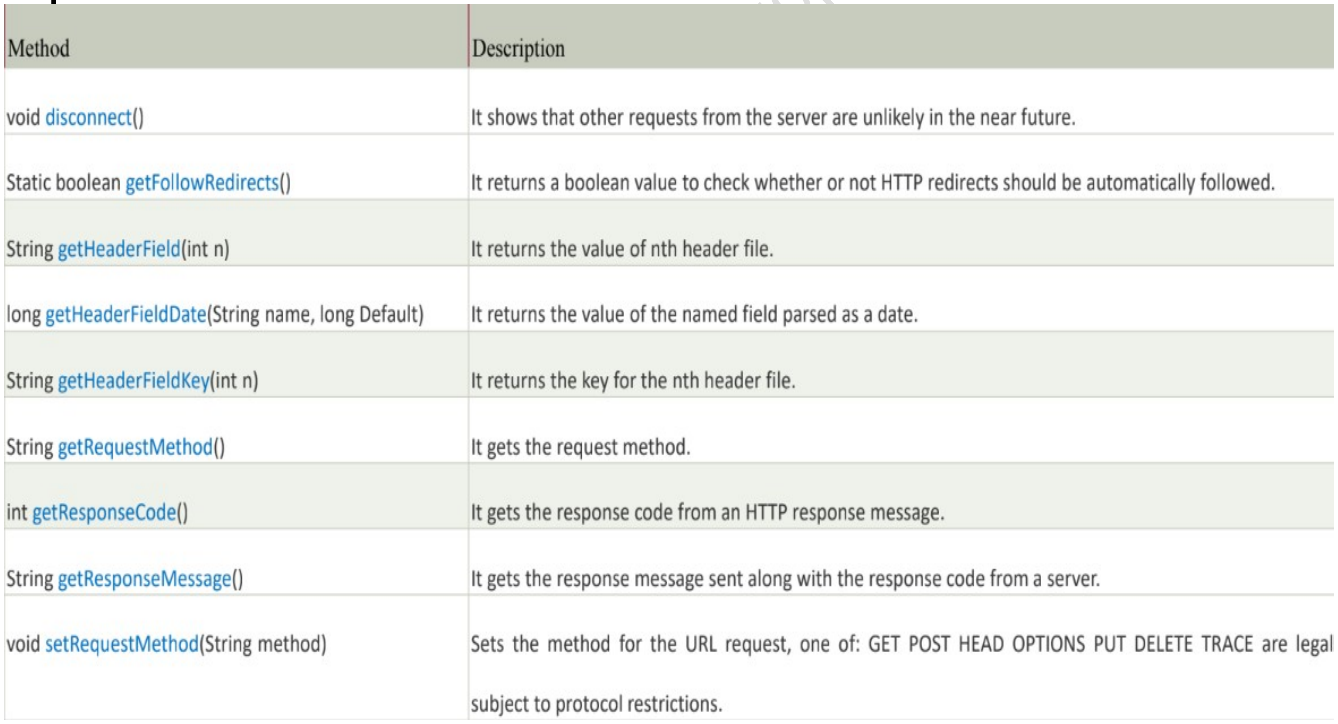
#def	image/x-bitmap
<!	text/html
<body	text/html
<head>	text/html
<html>	text/html
! XPM2	image/x-pixmap
GIF8	image/gif

Http URL Connection



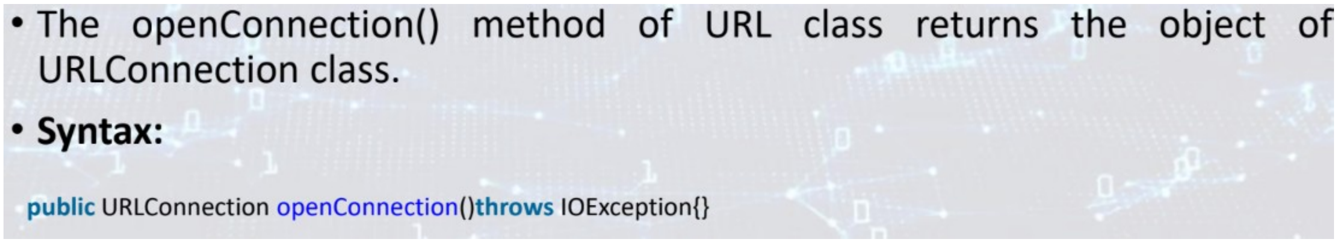
- The **Java HttpURLConnection** class is http specific URLConnection. It works for HTTP protocol only.
- HttpURLConnection extends URLConnection and provides fields and methods specific to an HTTP URL, such as, HTTP_CLIENT_TIMEOUT or setRequestMethod.
- By the help of HttpURLConnection class, you can retrieve information of any HTTP URL such as header information, status code, response code etc.
- The java.net.HttpURLConnection is subclass of URLConnection class.

Http URL Connection Method



Method	Description
void disconnect()	It shows that other requests from the server are unlikely in the near future.
Static boolean getFollowRedirects()	It returns a boolean value to check whether or not HTTP redirects should be automatically followed.
String getHeaderField(int n)	It returns the value of nth header file.
long getHeaderFieldDate(String name, long Default)	It returns the value of the named field parsed as a date.
String getHeaderFieldKey(int n)	It returns the key for the nth header file.
String getRequestMethod()	It gets the request method.
int getResponseCode()	It gets the response code from an HTTP response message.
String getResponseMessage()	It gets the response message sent along with the response code from a server.
void setRequestMethod(String method)	Sets the method for the URL request, one of: GET POST HEAD OPTIONS PUT DELETE TRACE are legal subject to protocol restrictions.

How to get the object of the URL Connection



- The openConnection() method of URL class returns the object of URLConnection class.
- **Syntax:**

```
public URLConnection openConnection()throws IOException{}
```