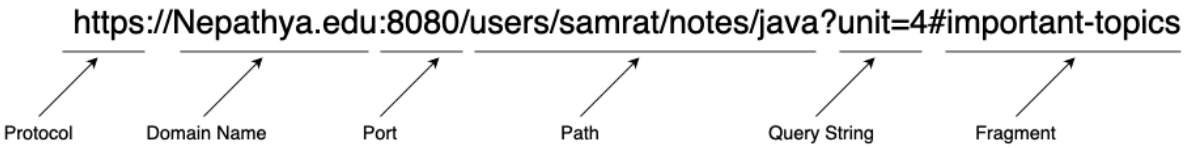
1. **What is URL? Give an example of a URL that shows each components of URL. Diagrammatic representation is more preferred.**
   A URL (Uniform Resource Locator) is the address used to access resources on the internet. It tells the browser where to find a web page, file, or service, and how to communicate with the server (using HTTP, HTTPS, FTP). It is like the home address of a specific webpage or file on the web.



   Protocol:Defines the method of communication (e.g., https).
   Domain Name:The website's name or IP address (e.g., Nepathya.edu).
   Port (optional):The port number on the server used for communication (e.g., 8080).
   Path:The location of a specific page or file on the server (e.g., /users/samrat/notes/java).
   Query String (optional):Passes data to the server, usually after a ? symbol (e.g., ?unit=4).
   Fragment (optional):Points to a section within the page, indicated by # (e.g., #important-topics).

2. **URLs vs URIs with examples for each. Can all URLs be URIs? Why?**

URI: It is a broader term that includes both URLs and URNs (Uniform Resource Names). A URI identifies a resource, but it doesn't necessarily indicate how to locate it. Example (specifies a local file): file:///C:/Documents/notes.txt

URL: A type of URI that not only identifies a resource but also provides the means to locate it (e.g., through a protocol like HTTP, FTP, etc.). Example (provides the address and location): https://Nepathya.edu:8080/users/samrat/notes/java?unit=4#important-topics

Yes, all URLs are considered URIs, but not all URIs are URLs. That's because a URL is a type of URI — it identifies a resource and also tells you how to access it, like through a web address. But some URIs, like URNs, only identify a resource without giving any information about where it's located or how to reach it. So while every URL qualifies as a URI, the reverse isn't always true.

3. Write a JAVA program for each of the following:

To illustrate important methods of URL class.

```
import java.net.*;

public class URLDemo {

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

 try {

// Creating a URL object

URL url = new URL(""https://samrat.com/index.html"");

      // Displaying URL components
       System.out.println("Protocol: " + url.getProtocol());
       System.out.println("Host: " + url.getHost());
       System.out.println("Port: " + url.getPort());
       System.out.println("File: " + url.getFile());
       System.out.println("Path: " + url.getPath());
       System.out.println("Query: " + url.getQuery());

   } catch (MalformedURLException e) {
      e.printStackTrace();
   }
}


}
```

To retrieve data from a URL. Use different URL than used in the program that I demoed in the class.

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

public class URLDataRetrieval {

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

 // Creating a URL object and retrieving data URL url = new
URL("https://www.example.com");

BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));
String inputLine;

  // Reading and printing data from the URL
   while ((inputLine = reader.readLine()) != null) {
      System.out.println(inputLine);
   }

   // Close the reader
   reader.close();
}


}
```

To check equality of URLs. Describe use case for equals() and sameFile() methods.

```
import java.net.*;

public class CheckURLEquality {

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

 try {

 // Creating two URL objects URL url1 = new URL("https://www.example.com"); URL url2 = new URL("https://www.example.com");

      // Using equals() method to check if the URLs are equal
      System.out.println("Using equals(): " + url1.equals(url2)); // Checks if the URLs are logically equal

      // Using sameFile() method to check if they refer to the same file
      System.out.println("Using sameFile(): " + url1.sameFile(url2)); // Checks if they point to the same file

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

To demo encoding and decoding of special characters using URLEncoder class.

```
import java.net.*;

public class URLEncoderDemo {

public static void main(String[] args) {

try {

// Encoding a string with special characters String originalString = "Namaste from Nepal!"; String encodedString = URLEncoder.encode(originalString, "UTF-8");
```

```
System.out.println("Encoded: " + encodedString);
//Output:encoded:Namaste+from+Nepal%21

      // Decoding the encoded string
       String decodedString = URLDecoder.decode(encodedString, "UTF-8");
       System.out.println("Decoded: " + decodedString);//Output:Decoded: Namaste from
Nepal!

   } catch (UnsupportedEncodingException e) {
      e.printStackTrace();
   }
}


}
```

To access password protected site using default authentication.

```
import java.net.;

 import java.io.;

public class ProtectedSiteAccess {

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

 // Set default authenticator for all connections

 Authenticator.setDefault(new Authenticator() {

 protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication("admin", "1234".toCharArray());

}

 });

   // Access a protected resource that requires HTTP Basic Auth
    URL protectedUrl = new URL("https://httpbin.org/basic-auth/admin/1234");
    BufferedReader br = new BufferedReader(new
InputStreamReader(protectedUrl.openStream()));
```

```
    String responseLine;
    System.out.println("=== Response from Protected Site ===");
    while ((responseLine = br.readLine()) != null) {
        System.out.println(responseLine);
    }
    br.close();
}


}
```

To access password protected site asking user to promt for username and password.

```
import java.net.;

import java.io.;

import java.util.Scanner;

public class PasswordPromptSite {

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

Scanner sc = new Scanner(System.in);

   System.out.print("Enter username: ");
    String username = sc.nextLine();
    System.out.print("Enter password: ");
    String password = sc.nextLine();
```

```
    URL url = new URL("https://example.com/protected-resource");
    URLConnection connection = url.openConnection();

    // Create the Basic Authentication header
    String auth = username + ":" + password;
    String encodedAuth = new
String(java.util.Base64.getEncoder().encode(auth.getBytes()));
    connection.setRequestProperty("Authorization", "Basic " + encodedAuth);

    // Read and display the response from the protected site
    BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
    String line;
    while ((line = reader.readLine()) != null) {
      System.out.println(line);
    }
}


}
```