## A small Java program that demonstrates how to use the `URL` and `URLConnection` classes to fetch data from a URL..

This program opens a URL, establishes a connection, and retrieves the response from the server.

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

public class URLConnectionDemo {
    public static void main(String[] args) {
        try {
            // Specify the URL
            URL url = new URL("http://www.example.com");

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

            // Set some default timeouts (optional)
            urlConnection.setConnectTimeout(5000);  // 5 seconds
            urlConnection.setReadTimeout(5000);     // 5 seconds

            // Open an input stream to read data from the URL
            BufferedReader in = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
            String inputLine;

            // Read and print the content from the URL
            while ((inputLine = in.readLine()) != null) {
                System.out.println(inputLine);
            }
            in.close();
        } catch (MalformedURLException e) {
            System.err.println("Invalid URL: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("Error reading from URL: " + e.getMessage());
        }
    }
}
```

## Explanation:

1. **Creating a URL Object**:
   `URL url = new URL("http://www.example.com");`
   This creates a `URL` object that points to the URL you want to fetch content from.
2. **Opening a Connection**:
   `URLConnection urlConnection = url.openConnection();`
   This opens a connection to the URL and returns a `URLConnection` object that allows you to interact with the resource.
3. **Reading from the URL**:
   A `BufferedReader` is used to read the data from the input stream of the connection (`urlConnection.getInputStream()`). The program reads and prints each line from the URL until there's no more content.
4. **Handling Exceptions**:
   The `MalformedURLException` is thrown if the URL is invalid, and `IOException` is thrown in case of network-related issues or if an error occurs while reading the content.

## Example Output:

This would print the HTML content of the URL (`http://www.example.com` in this case) to the console.

Let me know if you need any more details!