Tarea Individual 19 - UrlConnection

Desarrollar una aplicación en Java que abra una URL proporcionada como argumento de línea de comandos, establezca una conexión con la URL y lea su contenido línea por línea utilizando la clase URLConnection. El contenido debe ser impreso en la consola. El propósito es conocer cómo usar URLConnection para realizar peticiones HTTP y trabajar con flujos de entrada en Java.

Instrucciones

1. Crear un programa en Java que haga lo siguiente:
    1. Obtener una URL a partir de un argumento de línea de comandos (args[0]).
    2. Crear un objeto URL a partir del argumento y manejar la excepción MalformedURLException si la URL es inválida.
    3. Abrir una conexión a la URL utilizando openConnection() de la clase URL.
    4. Obtener el flujo de entrada desde la conexión utilizando getInputStream() de la clase URLConnection.
    5. Leer el contenido de la URL línea por línea con BufferedReader y InputStreamReader.
    6. Imprimir cada línea del contenido de la URL en la consola.
    7. Manejar las excepciones MalformedURLException y IOException adecuadam ente.
    8. Cerrar el flujo de entrada después de haber leído todo el contenido.
2. Consideraciones adicionales:
    - El programa debe poder manejar URLs de diferentes tipos (HTTP, HTTPS, etc.) sin fallos.
    - El manejo adecuado de excepciones es esencial para asegurar que el programa no termine inesperadamente.
    - Se recomienda que el programa permita recibir la URL desde la línea de comandos para que el usuario pueda elegir la URL a procesar.

Voy a detallar a qué parte de las instrucciones corresponde cada sección del código corregido:

Instrucción 1.1: Obtener una URL a partir de un argumento de línea de comandos (args[0]):

```
if (args.length != 1) {
   System.out.println("Por favor, ingrese una URL como argumento.");
   return;
}
```

```
String urlString = args[0];
```

Este bloque verifica que se haya proporcionado un argumento desde la línea de comandos y lo asigna a la variable urlString.

Instrucción 1.2: Crear un objeto URL a partir del argumento y manejar la excepción MalformedURLException si la URL es inválida:

```
try {
   URL url = new URL(urlString);
```

Este bloque intenta crear un objeto URL a partir del argumento. Si el formato es inválido, la excepción MalformedURLException será capturada más adelante.

Instrucción 1.3: Abrir una conexión a la URL utilizando openConnection() de la clase URL:

```
URLConnection connection = url.openConnection();
```

Se utiliza el método openConnection() de la clase URL para abrir la conexión con la URL proporcionada.

Instrucción 1.4: Obtener el flujo de entrada desde la conexión utilizando getInputStream() de la clase URLConnection:

```
try (BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
```

Aquí se utiliza getInputStream() para obtener un flujo de entrada desde la conexión y se envuelve en un BufferedReader para leer el contenido de manera eficiente.

Instrucción 1.5: Leer el contenido de la URL línea por línea con BufferedReader y InputStreamReader:

```
String line;
while ((line = reader.readLine()) != null) {
   System.out.println(line);
}
```

Este bloque lee el contenido línea por línea utilizando el método readLine() y lo imprime en la consola.

Instrucción 1.6: Imprimir cada línea del contenido de la URL en la consola:
System.out.println(line);

Cada línea leída se imprime en la consola.

Instrucción 1.7: Manejar las excepciones MalformedURLException y IOException adecuadamente:

```
} catch (MalformedURLException e) {
   System.out.println("URL mal formada: " + urlString);
} catch (IOException e) {
   System.out.println("Error al conectar o leer la URL: " + urlString);
}
```

Estos bloques capturan las excepciones MalformedURLException y IOException, mostrando mensajes de error claros.

Instrucción 1.8: Cerrar el flujo de entrada después de haber leído todo el contenido:

```
try (BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
```

El flujo se cierra automáticamente gracias al uso de try-with-resources.

Consideraciones adicionales:

1. **Manejar URLs de diferentes tipos (HTTP, HTTPS, etc.):**

```
if (connection instanceof HttpURLConnection) {
    HttpURLConnection httpConnection = (HttpURLConnection) connection;
    if (httpConnection.getResponseCode() != 200) {
        System.out.println("Error al acceder a la URL. Código de respuesta: "
                + httpConnection.getResponseCode());
        return;
    }
} else {
    System.out.println("La conexión no es de tipo HTTP. No se puede procesar.");
    return;
}
```

   Este bloque verifica si la conexión es de tipo HttpURLConnection y maneja URLs de diferentes tipos.

2. **Manejo adecuado de excepciones:** Ya se ha detallado en la instrucción 1.7.

14. **Permitir recibir la URL desde la línea de comandos:** Ya se ha cubierto en la instrucción 1.1.

RESULTADO:

El argumento que le pase fue la dirección a mi cuenta de github.