**ACTIVIDAD 4**

**Enunciado**

Realizar un programa con un método privado de nombre *Visualizar()*, que tenga un parámetro que sea un objeto URI y que visualice el resultado de aplicar diferentes métodos a un objeto URI.

Se usarán las siguientes URIs:

* http://docs.oracle.com
* http://docs.oracle.com/javase/7
* http://docs.oracle.com/javase/7/docs/api/java/net/URL.html

El nombre de la clase principal será Actividad4.

El resultado esperado es el siguiente:

URI: http://docs.oracle.com

Host: docs.oracle.com

Port: -1

Path:

Scheme: http

Authority: docs.oracle.com

URI: http://docs.oracle.com/javase/7

Host: docs.oracle.com

Port: -1

Path: /javase/7

Scheme: http

Authority: docs.oracle.com

URI: http://docs.oracle.com/javase/7/docs/api/java/net/URL.html

Host: docs.oracle.com

Port: -1

Path: /javase/7/docs/api/java/net/URL.html

Scheme: http

Authority: docs.oracle.com

**Solución**

**package paquete2;**

**import** java.net.MalformedURLException;

**import** java.net.URL;

**public** **class** Actividad4 {

**public** **static** **void** main(String[] args) {

**try** {

// Crear objetos URL utilizando diferentes constructores

URL url1 = **new** ~~URL~~("http://docs.oracle.com");

URL url2 = **new** ~~URL~~("http", "docs.oracle.com", "/javase/7");

URL url3 = **new** ~~URL~~("http", "docs.oracle.com", 80, "/javase/7");

URL base = **new** ~~URL~~("http://docs.oracle.com");

URL url4 = **new** ~~URL~~(base, "/javase/7/docs/api/java/net/URL.html");

// Visualizar diferentes objetos URL

System.***out***.println("Constructor simple para una URL:");

*Visualizar*(url1);

System.***out***.println("Constructor para protocolo, host y directorio:");

*Visualizar*(url2);

System.***out***.println("Constructor para protocolo, host, puerto y directorio:");

*Visualizar*(url3);

System.***out***.println("Constructor para un objeto URL y su directorio:");

*Visualizar*(url4);

} **catch** (MalformedURLException e) {

System.***out***.println("URL mal formada: " + e.getMessage());

}

}

**public** **static** **void** Visualizar(URL url) {

// Mostrar los diferentes componentes de la URL

System.***out***.println("\tMétodo toString():" + url.toString());

System.***out***.println("\tMétodo getProtocol():" + url.getProtocol());

System.***out***.println("\tMétodo getHost():" + url.getHost());

System.***out***.println("\tMétodo getPort():" + url.getPort());

System.***out***.println("\tMétodo getFile():" + url.getFile());

System.***out***.println("\tMétodo getUserInfo():" + url.getUserInfo());

System.***out***.println("\tMétodo getPath():" + url.getPath());

System.***out***.println("\tMétodo getAuthority():" + url.getAuthority());

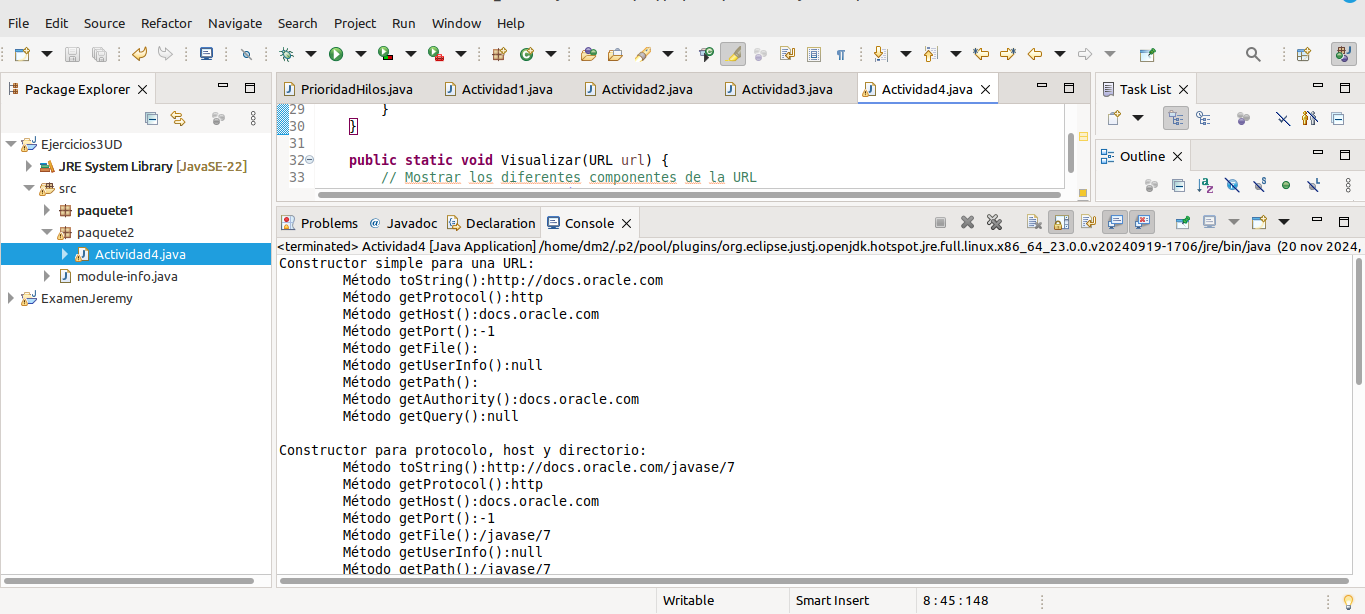
System.***out***.println("\tMétodo getQuery():" + url.getQuery());

System.***out***.println();

}

**}**

**Resolución de problemas**

****

**ACTIVIDAD 5**

**Enunciado**

Realizar un programa con un método privado llamado VisualizarConexión(), que use el método *newHttpClient()* de la clase Java HttpClient para realizar la conexión a la web *www.vitoria-gasteiz.com*, que use el método *newBuilder()* de la clase HttpRequest para crear una solicitud con la URI proporcionada ([www.vitoria-gasteiz](http://www.vitoria-gasteiz/).com) y que cree un objeto de la clase HttpResponse mediante el siguiente código:

HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

El manejo de la respuesta requiere lo siguiente:

1. Imprimir en consola una **primera línea** en forma de cadena del objeto HttpResponse. El método toString() del objeto HttpResponse devuelve una cadena que describe el objeto, incluyendo detalles cómo el código de estado y los encabezados de la respuesta.
2. Imprimir en consola una **segunda línea** con el código de estado HTTP de la respuesta. El método statusCode() del objeto HttpResponse devuelve un entero que representa el código de estado HTTP (por ejemplo, 200 para "OK", 404 para "Not Found").
3. Imprimir en consola una **tercera línea** con el valor del encabezado Content-Type de la respuesta. El método headers() del objeto HttpResponse devuelve un objeto HttpHeaders, que proporciona acceso a los encabezados de la respuesta. El método firstValue("Content-Type") devuelve un Optional<String> que contiene el primer valor del encabezado Content-Type, si está presente. El método orElse("N/A") se utiliza para proprocionar un valor predeterminado ("N/A") en caso de que el encabezado Content-type no esté presente. Una posible solución para la impresión de esta línea en la consola sería la siguiente:

System.out.println("\tMétodo getContentType():" + response.headers().firstValue("Content-Type").orElse("N/A"));

El nombre de la clase principal será Actividad5.

El resultado esperado es el siguiente:

Conexión con www.vitoria-gasteiz.com

==========================

Método toString():(GET http://www.vitoria-gasteiz.com) 200

Método getStatusCode():200

Método getContentType():text/html; charset=UTF-8

**Solución**

**package paquete2;**

**import** java.net.URL;

**import** java.net.URLConnection;

**import** java.io.IOException;

**import** java.util.List;

**import** java.util.Map;

**public** **class** Actividad5 {

**public** **static** **void** main(String[] args) {

**try** {

// Establecer la conexión con la URL

URL url = **new** ~~URL~~("http://www.vitoria-gasteiz.org");

URLConnection conexion = url.openConnection();

// Visualizar la información de la conexión

System.***out***.println("Conexión con www.vitoria-gasteiz.org");

System.***out***.println("==========================");

*VisualizarConexion*(conexion);

} **catch** (IOException e) {

System.***out***.println("Error al establecer conexión con la URL: " + e.getMessage());

}

}

**private** **static** **void** VisualizarConexion(URLConnection conexion) **throws** IOException {

// Asegúrate de conectar antes de recuperar datos

conexion.connect();

// Mostrar información básica

System.***out***.println("\tMétodo toString():" + conexion.toString());

System.***out***.println("\tMétodo Fecha():" + conexion.getDate());

System.***out***.println("\tMétodo getContentType():" + conexion.getContentType());

System.***out***.println();

// Mostrar campos de cabecera con getHeaderField

System.***out***.println("Campos Cabecera con getHeaderField:");

System.***out***.println("====================================");

**for** (**int** i = 0;; i++) {

String headerName = conexion.getHeaderFieldKey(i);

String headerValue = conexion.getHeaderField(i);

**if** (headerName == **null** && headerValue == **null**) {

**break**;

}

**if** (headerName == **null**) {

System.***out***.println("\t\tLinea " + (i + 1) + ":" + headerValue);

**continue**;

}

System.***out***.println("\t\t" + headerName + ":" + headerValue);

}

System.***out***.println();

// Mostrar campos de cabecera con getHeaderFields

System.***out***.println("Campos Cabecera con getHeaderFields:");

System.***out***.println("====================================");

Map<String, List<String>> headerFields = conexion.getHeaderFields();

**for** (Map.Entry<String, List<String>> entry : headerFields.entrySet()) {

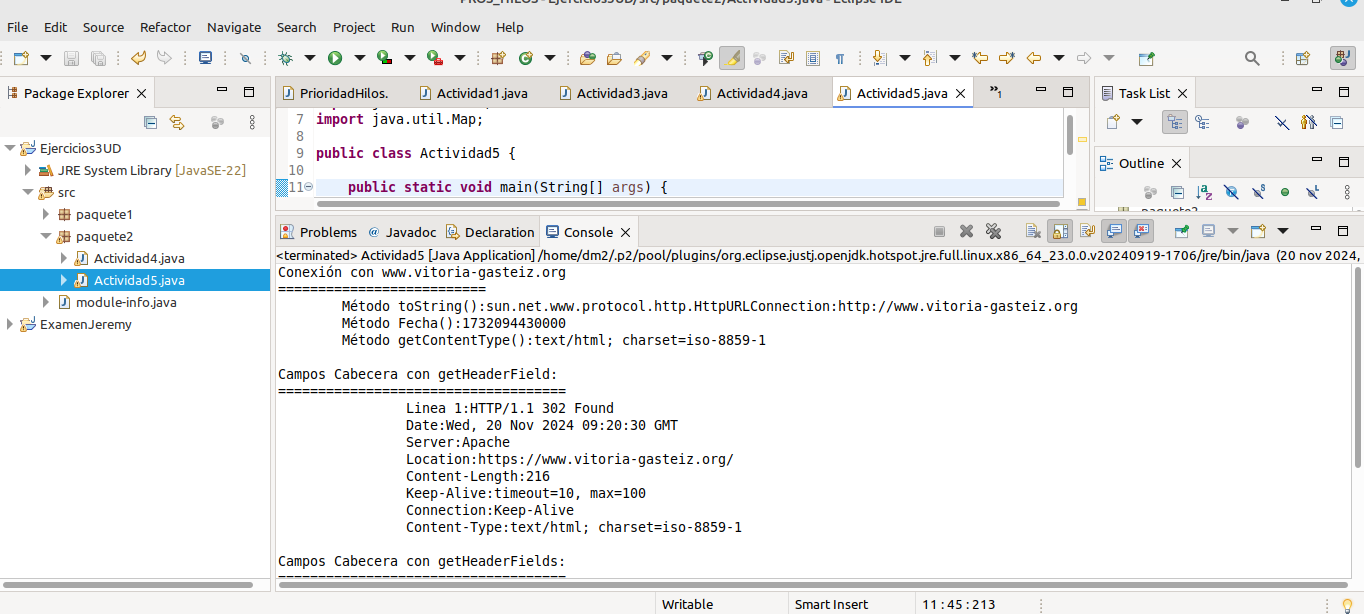
System.***out***.println(entry.getKey() + ":\t" + entry.getValue());

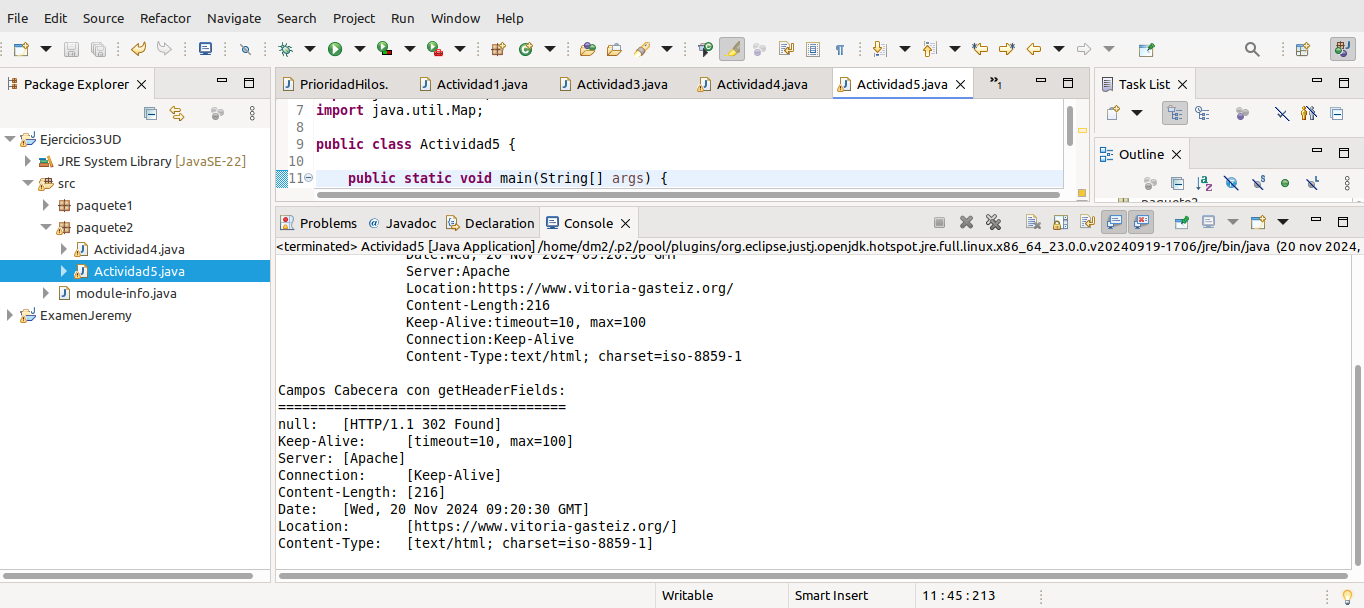
}

}

**}**

**Resolución de problemas**

****

****

**CRITERIOS DE CORRECCIÓN**

1. Nivel de ajuste a la nomenclatura de objetos definida en el enunciado de la actividad y nivel de ajuste al almacenamiento de la información requerida en las subcarpetas definidas para la actividad y unidad didáctica correspondientes (15%).
2. Hacer lo que se indica en el enunciado (55%).
3. Claridad del código Java (15%).
4. Documentación del código Java (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida).   
   Dicha explicación se incluirá al principio de los archivos de código fuente Java mediante líneas de comentarios.   
   La primera línea de comentarios sólo incluirá lo siguiente:  
    // Resolución de problemas.
6. Variables no usadas (-15%).
7. Librerías no usadas (-15%).