Ejercicios de Flujos de E/S y Red

Índice

1 Leer un fichero de texto....2
2 Copia de ficheros (*)....2
3 Lectura de una URL....2
4 Foro de Internet (*)....3
5 Gestión de productos (*)....3
6 Javamail (*)....4

1. Leer un fichero de texto

Vamos a realizar un programa que lea un fichero de texto ASCII, y lo vaya mostrando por pantalla. El esqueleto del programa se encuentra en el fichero `Ej1.java`. Se pide:

a) ¿Qué tipo de flujo de datos utilizaremos para leer el fichero? Añadir al programa la creación del flujo de datos adecuado (variable `in`), compilar y comprobar su correcto funcionamiento.

b) Podemos utilizar un flujo de procesamiento llamado `BufferedReader` que mantendrá un *buffer* de los caracteres leídos y nos permitirá leer el fichero línea a línea además de utilizar los métodos de lectura a más bajo nivel que estamos usando en el ejemplo. Consultar la documentación de la clase `BufferedReader` y aplicar la transformación sobre el flujo de entrada (ahora `in` deberá ser un objeto `BufferedReader`). Compilar y comprobar que sigue funcionando correctamente el método de lectura implementado.

c) Ahora vamos a cambiar la forma de leer el fichero y lo vamos a hacer línea a línea aprovechando el `BufferedReader`. ¿Qué método de este objeto nos permite leer líneas de la entrada? ¿Qué nos devolverá este método cuando se haya llegado al final el fichero? Implementar un bucle que vaya leyendo estas líneas y las vaya imprimiendo, hasta llegar al final del fichero. Compilar y comprobar que sigue funcionando de la misma forma.

2. Copia de ficheros (*)

En el fichero `Ej2.java` tenemos un programa que realizará una copia de ficheros en Java. Se pide:

a) Intentar compilar el programa y ver que errores obtenemos. Añadir la captura de las excepciones necesarias para que el programa compile (no capturar `Exception` en general).

b) Probar el programa copiando un fichero de texto. Comprobar que la copia se ha hecho correctamente.

c) Ahora probar a copiar un fichero ejecutable (binario) con nuestro programa. Intentar ejecutar la copia y comprobar que falla. ¿Por qué ocurre esto? ¿Cómo podemos solucionarlo? Cambiar el tipo de flujo de datos para solucionar este problema y comprobar que la nueva versión realiza un copia correcta del ejecutable .

3. Lectura de una URL

El fichero `Ej3.java` es un programa que tomará una URL como parámetro, accederá a ella y leerá su contenido mostrándolo por pantalla. Deberemos añadir código para:

a) Crear un objeto URL para la `url` especificada en el método `creaURL`, capturando las posibles excepciones que se pueden producir si está mal formada y mostrando el mensaje de error correspondiente por la salida de error. Compilar y comprobar que ocurre al pasar URLs correctas e incorrectas.

b) Abrir un flujo de entrada desde la URL indicada en el método `leeURL`. Deberemos obtener un `InputStream`de la URL, y convertirlo a un objeto `BufferedReader`, aplicando las transformaciones intermedias necesarias, para poder leer de la URL los caracteres línea a línea. Comprobar que lee correctamente algunas URLs conocidas. Descomentar el bloque de código que realiza la lectura de la URL.

4. Foro de Internet (*)

Vamos a realizar una aplicación que nos permita acceder a un foro en Internet. Partiremos de la plantilla que tenemos en el fichero `Ej4.java`. Se pide:

a) Introducir en el método `leeMensajes` el código necesario para leer los mensajes publicados en el foro. Para ello accederemos a la URL definida en la constante `URL_LISTA` y leeremos el contenido que nos proporcione mediante un objeto `DataInputStream`. Esta información nos la devolverá como una serie de cadenas (UTF) cada una de las cuales correspondientes a uno de los mensajes.

```
<mensaje1:UTF>
<mensaje2:UTF>
...
<mensajeN:UTF>
```

Leeremos mensajes hasta llegar al final del flujo (cuando se produzca una excepción `EOFException`), e imprimiremos en la consola cada mensaje leído de la siguiente forma:

```
DataInputStream dis = new DataInputStream(in);
try {
  while(true) {
    String msg = dis.readUTF();
    System.out.println(msg);
  }
} catch(EOFException e) {}
```

b) Ahora vamos a permitir enviar un mensaje al foro. Para ello introduciremos el código necesario en el método `enviaMensaje`, en el que utilizaremos un objeto `DataOutputStream` para enviar los datos a la URL definida en `URL_ENVIAR`. Deberemos enviar nuestro *nick* y el mensaje de la siguiente forma:

```
DataOutputStream dos = new DataOutputStream(out);
dos.writeUTF(nick);
dos.writeUTF(msg);
```

5. Gestión de productos (*)

Vamos a hacer una aplicación para gestionar una lista de productos que vende nuestra empresa. Escribiremos la información de estos productos en un fichero, para almacenarlos de forma persistente. Se pide:

a) Introducir el código necesario en el método `almacenar` de la clase `GestionProductos` para guardar la información de los productos en el fichero definido en la constante `FICHERO_DATOS`. Guardaremos esta información codificada en un fichero binario. Deberemos codificar los datos de cada producto (titulo, autor, precio y disponibilidad) utilizando un objeto `DataOutputStream`.

b) Introducir en el método `recuperar` el código para cargar la información de este fichero. Para hacer esto deberemos realizar el procedimiento inverso, utilizando un objeto `DataInputStream` para leer los datos de los productos almacenados. Leeremos productos hasta llegar al final del fichero, cuando esto ocurra se producirá una excepción del tipo `EOFException` que podremos utilizar como criterio de parada.

c) Modificar el código anterior para, en lugar de codificar manualmente los datos en el fichero, utilizar la serialización de objetos para almacenar y recuperar objetos `Producto` del fichero.

6. Javamail (*)

En el proyecto `jhd-sesion06-javamail` tenemos una serie de ejemplos de uso de esta librería. Vamos a probar cada uno de ellos.

a) El ejemplo más sencillo es `EnvioSimple`. Vamos a abrir el código con el editor de Eclipse y observar su contenido. Modificar el ejemplo de forma que nos envíe un mensaje a nuestra cuenta de e-mail. Como servidor de correo saliente podemos dejar el que viene, o bien usar cualquier otro que no requiera autentificación. Comprobar que el correo llega correctamente.

b) Vamos ahora a a ver como segundo ejemplo el fichero `EnvioAdjunto`. En él se envía un correo con una imagen como adjunto. Modificar la dirección del destinatario para que nos envíe el correo a nuestra cuenta, y comprobar que lo envía correctamente.

c) Modificar el ejemplo anterior de forma que el contenido del mensaje esté escrito en HTML y la imagen se ponga como cabecera.

d) Por último, probaremos un ejemplo de envío con un servidor de correo que utilice SMTP con SSL y requiera autentificación, como es el caso de Gmail. Para poder probarlo necesitaremos una cuenta de esas características. Modificar el código de `EnvioAutentificacion` para adaptarlo a la configuración de nuestro servidor de correo saliente, y probar que funciona correctamente.