PRÁCTICA 2

2.1 Objectivo

El objetivo de esta práctica es la introducción al acceso a ficheros mediante flujos de datos, a utilizar métodos para la entrada/salida de datos formateada y a introducir métodos para trabajar con el sistema de archivos del ordenador.

2.2 Flujos de acceso a archivos (escritura)

En esta práctica trabajaremos con flujos de datos ”reales”, creados por nosotros mismos. Introduciremos la clase *FileOutputStream*, una clase heredera de *OutputStream*, destinada a la escritura de archivos en disco.

Para crear un objeto de tipo *FileOutputStream* necesitaremos indicar al constructor de la clase el nombre del archivo que queremos crear/escribir. De esta forma se creará un objeto que¸ a todos los efectos, se comportará como un objeto de tipo *OutputStream*, es decir, podemos considerarlo como un “tubo” a través del cual escribimos bytes de uno en uno.

Existen diferentes formas de crear un fichero para poder escribir en él. En el ejemplo siguiente vemos la forma más directa:

```
FileOutputStream fos=new FileOutputStream(“d:\\temp\\datos.dat”);
```

Notad la doble barra invertida para especificar la ruta. Es la forma que tiene Java de tratar los códigos de escape en las cadenas de texto.

Ejercicio 1

Ejercicio 1.1

Escribe un programa que abra un fichero llamado “texto.dat” y que escriba en él el texto introducido por teclado. Para ello crea un objeto de tipo *FileOutputStream* tal como se ha descrito antes.

Ejercicio 1.2

Escribe un programa que abra un fichero llamado “nums.dat” y escriba en él los números enteros (utiliza una variable de tipo *int*) del 1 al 10. Para ello crea un objeto de tipo *FileOutputStream* tal como se ha descrito antes.

2.3 Flujos de acceso a archivos (lectura)

Ahora introduciremos la clase *FileInputStream*, una clase heredera de *InputStream* destinada a la lectura de archivos en disco.

Para crear un objeto de tipo *FileInputStream* necesitaremos indicarle al constructor de la clase el nombre del archivo que queremos leer. Esto creará un objeto que, a todos los efectos, se comportará como un objeto de tipo *InputStream*, es decir, podremos considerarlo un “tubo” del cual extraemos bytes de uno en uno. No nos importa el hecho de que en el otro extremo del “tubo” haya un archivo del que vienen los datos. Podremos trabajar con el “tubo” usando las técnicas vistas en el caso de *InputStream*.

Una forma, la más directa, de abrir un archivo para leer su contenido es la siguiente:

```
FileInputStream fis=new FileInputStream(“d:\\temp\\datos.dat”);
```

En este ejemplo se supone que tenemos el fichero *datos.dat* en la ruta indicada, *d:\temp*.

Hay que tener en cuenta que aunque el objeto *fis* sea del tipo *FileInputStream*, podemos trabajar con él como si fuera un objeto de tipo *InputStream* debido a la herencia.

Ejercicio 2

Ejercicio 2.1

Haz un programa que permita leer el contenido del fichero “texto.dat” creado en el primer ejercicio y sacarlo por la salida estándar (la pantalla). ¿Consigues reproducir por pantalla el contenido del fichero?

Ejercicio 2.2

Haz ahora un programa que permita leer contenido del fichero “nums.dat” creado en el primer ejercicio y sacarlo por la salida estándar (la pantalla). ¿Consigues reproducir por pantalla el contenido del fichero? ¿A qué crees que es debido?

Ejercicio 2.3

Ahora modifica el programa anterior de forma que intente abrir un fichero inexistente. ¿Qué ocurre?

Al no existir el fichero se produce una excepción en Java. Todos los métodos *read()*, *write()*, etc. que trabajan con flujos lanzan excepciones del tipo *java.io.IOException*. Estas excepciones deben ser capturadas.

En general las partes de los programas que trabajen con flujos deben estar dentro de una cláusula *try...catch*, tal como se ve a continuación:

```
try{
InputStream is = ... ;
while(...){
/* Leemos y procesamos los datos */
}
} catch (IOException ioe){
System.err.println("Error al abrir el flujo tal y tal...");
ioe.printStackTrace();
}
```

En programas pequeños en los que no queramos complicarnos con estructuras de este tipo, podemos tomar el camino fácil de mandar la excepción "hacia arriba". Por ejemplo:

```
public static void main(String args[]) throws IOException {
/* Programa creado por un programador vago */
```

Ejercicio 2.4

Modifica el código del ejercicio 2.3 de forma que capture la excepción e informe por la salida estándar del hecho.