LECTURA DE FICHEROS PLANOS

Ficheros de datos

- Un fichero permite almacenar datos en memoria secundaria para utilizarlos en el futuro
- Un fichero puede ser de entrada (input) o salida (output) dependiendo de si se utiliza para leer o escribir datos
- En Java existen dos tipos de ficheros: ficheros binarios y ficheros de texto
- La clase **PrintWriter** se utiliza para escribir en ficheros de texto. La clase **File** se utiliza para leer ficheros de texto.

Lectura de Ficheros

- Para leer de un fichero de texto se utiliza la **clase File** y la **clase Scanner**.

- Para leer los datos almacenados en el fichero es necesario utilizar un bucle, por lo que utilizaremos una sentencia **while** hasta llegar al final del fichero (EOF)

- Utilizaremos el método **hasNext()** de la clase Scanner para **detectar el final del fichero**.

Classe ‘File’

Els objectes de la classe File representen rutes del Sistema de Fitxers.

No representa el contingut !!!

Que es pot fer amb aquesta classe?

- **Funcions de manipulació i consulta de la pròpia estructura jeràrquica** (creació, eliminació, obtenció de la ubicació, etc. de fitxers o carpetes)
- **Funcions de manipulació i consulta de les característiques particulars dels elements** (noms, mida o capacitat, etc.)
- **Funcions de manipulació i consulta d’atributs específics de cada sistema operatiu** i que, per tant, només serà funcional si el sistema operatiu amfitrió suporta també la funcionalitat. Ens referim, per exemple, als permisos d’escriptura, d’execució, atributs d’ocultació, etc.

Lectura de Ficheros

- Para leer de un fichero de texto se utiliza la **clase File** y la **clase Scanner**.

- Para leer los datos almacenados en el fichero es necesario utilizar un bucle, por lo que utilizaremos una sentencia **while** hasta llegar al final del fichero (EOF)

- Utilizaremos el método **hasNext()** de la clase Scanner para **detectar el final del fichero**.

Lectura de Ficheros

Para leer la información almacenada en un fichero de texto:

1. Abrir el fichero con un objeto de la clase File y vincularlo con un objeto de la clase Scanner

2. Leer el fichero

3. Cerrar el fichero

LecturaFicheros.java

```java
package ficherosPlanos;
import java.io.File;
import java.util.Scanner;

public class LecturaFicheros {

    public static void main(String[] args) {

        // Fichero del que queremos leer
        File fichero = new File("prueba.txt");
        Scanner s = null;

        try {
            // Leemos el contenido del fichero
            System.out.println("... Leemos el contenido del fichero ...");
            s = new Scanner(fichero);

            // Leemos linea a linea el fichero
            while (s.hasNextLine()) {
                String linea = s.nextLine();    // Guardamos la linea en un String
                System.out.println(linea);      // Imprimimos la linea
            }

        } catch (Exception ex) {
            System.out.println("Mensaje: " + ex.getMessage());
        } finally {
            // Cerramos el fichero tanto si la lectura ha sido correcta o no
            try {
                if (s != null)
                    s.close();
            } catch (Exception ex2) {
                System.out.println("Mensaje 2: " + ex2.getMessage());
            }
        }
    }
}
```

Escritura de Ficheros

Para escribir en un fichero de texto utilizaremos dos clases:

FileWriter y **PrintWriter**.

La clase FileWriter permite tener acceso al fichero en modo escritura.

La clase **FileWriter** proporciona el **método write()** para escribir cadenas de caracteres aunque lo normal es utilizar esta clase junto con la clase PrintWriter para facilitar la escritura.

La clase **PrintWriter** permite **escribir** caracteres en el fichero **de la misma forma que en la pantalla**.

Interfaz Serializable

La **serialización** es la transformación de un objeto en una secuencia de bytes que pueden ser posteriormente leídos para reconstruir el objeto original.

El objeto serializado pueda guardarse en un fichero o puede enviarse por red para reconstruirlo en otro lugar.

Puede crearse en un sistema Windows y enviarlo. por ejemplo, a otro sistema que utilice Linux.

Interfaz Serializable

Para escribir objetos en un fichero binario en Java se utiliza la **clase ObjectOutputStream** derivada de OutputStream.

Un objeto ObjectOutputStream se crea a partir de un objeto FileOutputStream asociado al fichero.

El constructor de la clase es:
ObjectOutputStream(OutputStream nombre);
Lanza una excepción **IOException**.
Por ejemplo, las instrucciones para crear el fichero personas.dat para escritura de objetos serían éstas:

```
FileOutputStream fos = new FileOutputStream ("/ficheros/personas.dat");
ObjectOutputStream salida = new ObjectOutputStream (fos);
```

Interfaz Serializable

Para leer los objetos contenidos en un fichero binario que han sido almacenados mediante ObjectOutputStream se utiliza la **clase ObjectInputStream** derivada de InputStream.

Un objeto ObjectInputStream se crea a partir de un objeto FileInputStream asociado al fichero.
El constructor de la clase es:
ObjectInputStream(InputStream nombre);
Lanza una excepción **IOException**.
Por Ejemplo, las instrucciones para crear el objeto ObjectInputStream para lectura de objetos del fichero personas.dat serían éstas:

```
FileInputStream fis = new FileInputStream ("/ficheros/personas.dat");
ObjectInputStream entrada = new ObjectInputStream (fis);
```

La clase proporciona el **método readObject()** que devuelve el objeto del fichero (tipo Object).
Es necesario hacer un **casting** para guardarlo en una variable del tipo adecuado.