CLASES PARA GESTIÓN DE FLUJOS DE DATOS DESDE/HACIA FICHEROS

En java podemos utilizar dos tipos de ficheros:

- **Modo Texto**: en este caso los datos son almacenados usando código ASCII y por tanto son plenamente visibles usando cualquier editor.
- **Modo Binario**: en este caso los datos son almacenados en notación hexadecimal y por tanto se ocupa un editor binario para reconocerlos sin embargo un archivo binario es más compacto que un archivo texto.

En ambos casos el acceso a los mismos se puede realizar de forma secuencial o aleatoria.

Ficheros de texto

Los ficheros de texto normalmente se generan con un editor, almacenan caracteres alfanuméricos en un formato estándar (ASCII, UNICODE, UTF8, etc.). Utilizan las clases **FileReader** para leer caracteres y **FileWriter** para escribir los caracteres en el fichero.

Cuando trabajamos con ficheros, cada vez que leemos o escribimos en uno debemos hacerlo dentro de un manejador de excepciones **try-catch**. Al usar la clase **FileReader** se puede generar la excepción ***FileNotFoundException*** (porque el nombre del fichero no existe o no es válido) y al usar la clase **FileWriter** la excepción ***IOException*** (el disco está lleno o protegido contra escritura).

Escritura en Ficheros de Texto

Para escribir cadenas de caracteres en un fichero el paquete java.io proporciona las clases **`FileWriter`**, **`BufferedWriter`** y **`PrinterWriter`**.

La clase **FileWriter** proporciona los siguientes métodos para escritura:

Método	Acción
void write(int c)	Escribe un carácter
void write(char[] buf)	Escribe un array de caracteres
void write(char[] buf, int desplazamiento, int n)	Escribe n caracteres de datos en la matriz buf y comenzando por buf [desplazamiento]
void write(String str)	Escribe una cadena de caracteres
append(char c)	Añade un carácter a un fichero

El primer paso para poder escribir en un fichero de texto, es crear un objeto **`FileWtiter`** que nos permita tener acceso al fichero en modo escritura. Para ello utilizaremos unos de los siguientes **constructores**:

```
FileWriter(String path);
```

```
FileWriter fw = new FileWriter("Nombres.txt");
```

```
FileWriter(File fichero);
```

```
File f = new File("Nombres.txt");
FileWriter fw = new FileWriter(f);
```

`FileWriter(String path, boolean append);`

```
FileWriter fw = new FileWriter("Nombres.txt", true);
```

`FileWriter(File fichero, boolean append);`

```
File f = new File("("Nombres.txt");
FileWriter fw = new FileWriter(f, true);
```

Se puede construir un objeto **`FileWriter`** proporcionando la ruta del fichero directamente o a partir de un objeto `File` existente

El parámetro `append` permite indicar si los datos que se van a escribir se añadirán a los existentes (`true`) o sobreescribirán a éstos (`false`). Si se utiliza uno de los dos primeros constructores, los datos escritos en el fichero sustituirán a los existentes, es equivalente a `append false`.

La clase **`FileWriter`** proporciona el método `write()` que permite escribir en el fichero la cadena de caracteres pasada como parámetro, aunque también se puede utilizar la clase estándar de escritura `PrintWriter` para realizar esta operación.

El segundo paso es crear un objeto **`PrintWriter`** **que** nos permite que la escritura en un fichero se realice de la misma forma que la escritura en pantalla. La diferencia está en que en el caso de la pantalla el objeto **`PrintWriter`** está asociado a **`System.out`**, mientras que para un fichero de texto habrá que construir el objeto **`PrintWriter`** a partir del objeto **`FileWriter`**, o bien utilizar un método de la clase **FileWriter.**

```
FileWriter fw = new FileWriter("datos.txt");
PrintWriter salida = new PrintWriter(fw);
Salida.println("Hola");
```

Desde la versión 5 de Java, también es posible crear un objeto **`PrintWriter`** a partir de un objeto **`File`** o incluso de la ruta del fichero, por lo que las instrucciones anteriores podrían reducirse a una.

```
PrintWriter salida = new PrintWriter("datos.txt");
```

Una vez creado el objeto `PrintWriter`, podemos utilizar los métodos `print()`, `println()` y `printf()` para escribir en el fichero.

Ejemplo: El programa almacenará unos nombres de un en un fichero en disco utilizando **PrintWriter**.

```
package ejemplos03EscrituraTexto;
import java.io.File;
import java.io.FileWriter;
import java.io.FilterWriter;
import java.io.IOException;
import java.io.PrintWriter;

// Ejemplo de utilización de las clase FileWriter y PrintWriter

public class Ejemplo01PrintWriter {
	public static void main(String[] args) throws IOException {
		//array de nombres
```

```
            String [] nombres = {"Maria", "Ana", "Santiago", "Jorge",
"Iciar", "Isabel", "Pedro", "Manuel"};

            //     File f = new File("Nombres.txt");
            //     FileWriter fw = new FileWriter(f);

            FileWriter fw = new FileWriter("Nombres.txt");

             // la siguiente sentencia añade los nombres a los ficheros
      //    FileWriter fw = new FileWriter("Nombres.txt", true);
      //    PrintWriter salida = new PrintWriter(fw);
            // las dos sentencias anteriores son equivalentes a la
siguiente
            PrintWriter salida = new PrintWriter("Nombres.txt");

            for (int i = 0; i < nombres.length; i++) {
                  salida.println(nombres[i]);
            }
            salida.flush();
            salida.close();
      }
}
```

La llamada al método **`flush()`** garantiza que todos los datos enviados a través del buffer de salida han sido escritos en el fichero y el método **`close()`** cierra la conexión con el fichero y libera los recursos utilizados por ésta.

El siguiente ejemplo escribe caracteres en un fichero de nombre FichTexto01.txt (si no existe lo crea). Los caracteres se escriben uno a uno y se obtienen de un String. Utilizando el método **write().**

```
package ejemplos03EscrituraTexto;
import java.io.*;
/*
 * escribe caracteres en un fichero de nombre FichTexto01.txt (si no
existe lo crea).
 *  se escribe el array completo en una operación de escritura
 */

public class Ejemplo02FileWriter {
      public static void main(final String[] args) throws IOException {

            File f= new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros\\FichTexto0111.txt
");
            FileWriter fw = new FileWriter(f); // crea el fichero de
salida

            String cadena = "Esto es una prueba de FileWriter método
write escribiendo caracter a caracter";
```

```
            char[] cad = cadena.toCharArray(); // convierte un String en
un array de caracteres

            fw.write("HOla"); // escribe la palabra Hola en el fichero
            for(int i=0; i< cad.length; i++){
                  fw.write(cad[i]); // se escribe un caracter
                  fw.write("\n"); // se produce un salto de línea
            }
            fw.append('*'); // añade un * al final
            fw.write(cadena);
            fw.append('*'); // añade un * al final
            fw.write(cad);
            fw.close(); // cierra el fichero
            System.out.println("Programa Finalizado");
      }
}
```

En lugar de escribir los caracteres uno a uno, también se pueden escribir todo el array: ***fw.write(cad)***.

```
package ejemplos03EscrituraTexto;

import java.io.*;

public class Ejemplo02FileWriterB {
      /*
       * escribe caracteres en un fichero de nombre FichTexto02.txt (si
no existe lo crea).
       *  se escribe el array completo en una operación de escritura
       */

      public static void main(final String[] args) throws IOException {

            File f= new File("FichTexto02.txt");
            FileWriter fw = new FileWriter(f); // crea el fichero de
salida

            String cadena = "Esto es una prueba de FileWriter método
write escribiendo todo"
                        + "el array en una unica operacion";
            char[] cad = cadena.toCharArray(); // convierte un String en
un array de caracteres

            fw.write(cad); // se escribe el array
            fw.append('*'); // añade un * al final
            fw.close(); // cierra el fichero
            System.out.println("Programa Finalizado");
      }
}
```

El siguiente ejemplo escribe cadenas de caracteres que se obtienen de un array de String, las cadenas se graban una a continuación de la otra sin saltos de línea.

```
package ejemplos03EscrituraTexto;
import java.io.*;
/*
 * Escribe cadenas de caracteres que se obtiene de un array de String,
 * las cadenas se graban una a continuación de la otra sin saltos de
línea
 */
public class Ejemplo03FileWriter {
	public static void main(final String[] args) {

		// creamos un array de string
		String provincias[] = {"La Coruña", "Lugo", "Orense",
"Pontevedra", "Guipúzcoa",
				"Vizcaya", "Alava"};

		try{
			File f= new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros\\Provincias.txt");
			FileWriter fw = new FileWriter(f); // crea el fichero
de salida
			for (int i=0; i< provincias.length; i++)
				fw.write(provincias[i]);
			fw.close();
		}catch(IOException ioe){
			System.out.println("Disco lleno o protegido");
			ioe.printStackTrace();
		}finally{
			System.out.println("programa finalizado");
		}
	}
}
```

Ejemplo de cómo utilizar el desplazamiento y el número de caracteres en el método **write().**

```
package ejemplos03EscrituraTexto;
import java.io.*;

/*
 * Escribe cadenas de caracteres que se obtiene de un array de String,
las cadenas se graban desde un
 *  desplazamiento  de 10 caracteres
 */
public class Ejemplo04FileWriter {
	public static void main(final String[] args) {

		String cadena = "Probando el desplazamiento y el numero de
caracteres a escribir";
```

```
            try{
                  File f= new File("Desplazamiento.txt");
                  FileWriter fw = new FileWriter(f); // crea el fichero
de salida

                  //desde la posición 3 escribe 10 caracteres
                  fw.write(cadena, 3, 10);

                  fw.close();
            }catch(IOException ioe){
                  System.out.println("Disco lleno o protegido");
            }finally{
                  System.out.println("programa finalizado");
            }
      }
}
```

La clase **BufferedWriter** añade un buffer para realizar una escritura eficiente de caracteres. Para construir un **BufferedWriter** necesitamos la clase **FileWriter**.

```
BufferedWriter fichero = new BufferedWriter(new FileWriter
(NombreFichero));
```

El siguiente ejemplo escribe 10 filas de caracteres en un fichero de texto y después de escribir cada fila salta una línea con el método **newLine().**

```
package ejemplos03EscrituraTexto;
import java.io.*;
/*
 * El siguiente ejemplo escribe 10 filas de caracteres en un fichero
de texto y después de escribir
 * cada fila salta una línea con el método newLine().
 */
public class Ejemplo05FileWriter {
      public static void main(final String[] args) {
            try{
                  FileWriter fw = new FileWriter("Filas.txt");
                  //se utiliza cuando queremos escribir lineas
                  BufferedWriter bw = new BufferedWriter(fw);

                  for(int i=0; i<11; i++){
                        bw.write("Fila número: " +i);
                        bw.newLine();
                  }
                  bw.close();
            }catch(FileNotFoundException fne){
                  System.out.println("No se encuentra el fichero");
                  fne.printStackTrace();
            }catch(IOException ioe){
                  System.out.println("Error de L/E");
```

```
            }
        }
}
```

Lectura de un Fichero de Texto

Para recuperar cadenas de caracteres de un fichero el paquete java.io proporciona las clases **`FileReader`** y **`BufferedReader`**.

El primer paso para recuperar información de un fichero de texto, es crear un objeto `FileReader` asociado al mismo. Un objeto `FileReader` representa un fichero de texto "abierto" para la lectura de datos. Este objeto es capaz de adaptar la información recuperada del fichero a las características de una aplicación Java, transformado los butes almacenados en el mismo en caracteres Unicode.

Constructores

```
FileReader(String path); // construye un objeto
proporcionándole directamente la ruta

FileReader(File fichero); // construye un objeto a partir de
un objeto File existente
```

Ejemplo: crea un objeto `FileReader` haciendo referencia al fichero `datos.txt`

```
File f = new File("datos.txt);

FileReader fr = new FileReader(f);
```

La clase `FileReader` proporciona el método `Read()` para la lectura de la información almacenada en un fichero. Se utiliza poco, porque recupera la información como byte y posteriormente hay que convertirla en `String`, por lo que se utiliza más la clase `BufferedReader` para leer, utilizando el objeto `FileReader` como puente para crear un objeto de este tipo.

El segundo paso consiste en crear un BufferedReader, para lo que se utiliza el constructor:

```
BufferedReader(Reader entrada);
```

Para leer información del disco el procedimiento es igual que cuando lo hacemos desde el teclado, sólo que en este caso el objeto Reader no es System.in, sino que será el objeto FileReader asociado al fichero.

```
BufferedReader br = new BufferedReader(fr);
```

Una vez creado el objeto se puede utilizar el método `readLine()` para leer líneas de texto del fichero de forma similar a como leemos del teclado. En el caso de los ficheros, como pueden estar formados por más de una línea, será necesario utilizar un bucle `while` para recuperar todas las líneas de texto del mismo de forma secuencial.

```
package ejemplos04ALecturaTexto;

import java.io.*;

//ejemplo que lee el fichero nombres01 linea a linea
public class Ej01FileReaderLineas {
```

```
	public static void main(String[] args) {
		File f;
		FileReader fr;
		BufferedReader br = null;
		try{
			f = new File("Nombres01.txt");
		//	if(f.exists()){
				fr = new FileReader(f);
				br = new BufferedReader(fr);
				String nombre; // variable donde se recupera la
informacion
				while((nombre = br.readLine())!= null){
					System.out.println(nombre);
				}
		//	}else
			//	System.out.println(("El fichero no existe"));
		}catch(FileNotFoundException fn){
			System.out.println("No se encuentra el fichero");
		}catch(IOException ioe){
			System.out.println("Error de L/E");
		}finally{
			try {
				br.close();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
				System.out.println("Fin del programa");
			}
		}
	}
```

El método `readLine()` apunta a la siguiente línea de texto después de recuperar la línea actual. Cuando no existen más líneas para leer, la llamada a `readLine()` devuelve `null`.

Si lo que se quiere es leer carácter a carácter en lugar de línea a línea se utiliza el método `read()` en lugar de `readLine()`. El método `read()` devuelve un entero que representa el código Unicode del carácter leído, siendo el resultado `-1` si no hay más caracteres para leer.

```
package ejemplos04ALecturaTexto;

import java.io.*;

public class Ej02FileReaderCaracter {

	public static void main(String[] args) {
		File f;
		FileReader fr = null;
		int caracter;
```

```
            try{
                  f = new File("Nombres01.txt");
                  if(f.exists()){
                        fr = new FileReader(f);
                        //leemos un caracter y sino es el final lo
escribimos el final del fichero -1
                        while((caracter = fr.read()) != -1 ){
                              System.out.println((char)caracter);
                        }
                  }else
                        System.out.println(("El fichero no existe"));
            }catch(FileNotFoundException fn){
                  System.out.println("No se encuentra el fichero");
            }catch(IOException ioe){
                  System.out.println("Error de L/E");
            }finally{
                  try{
                        fr.close();
                  }catch(IOException ioe){
                        System.out.println("Error al cerrar el fichero");
                  }
            }
      }
}
```

Los métodos que proporciona la clase **FileReader** devuelven el número de caracteres leídos o -1 si se ha llegado al final del fichero. Son los siguientes:

Método	Acción
int read()	Lee un carácter y lo devuelve
int read(char[] buf)	Lee hasta buf.length caracteres de datos de una matriz de caracteres (buf). Los caracteres leídos del fichero se van almacenando en buf.
int read(char[] buf, int desplazamiento, int n)	Lee hasta n caracteres de datos en la matriz buf y comenzando por buf [desplazamiento] y devuelve el número de caracteres leídos.

Es posible realizar la lectura de los datos almacenados en un fichero de texto utilizando la clase Scanner, para ello se pasará el objeto `File` asociado al fichero al constructor de `Scanner`, y después se utilizarán los métodos de esta clase.

```
package ejemplos04ALecturaTexto;

import java.io.*;
import java.util.Scanner;

public class Ej03FileReaderArrayPalabras {
      public static void main(String[] args) {
            File f = new File("Nombres01.txt");
            FileReader fr = null;
```

```
            Scanner sc = null;
            try{

                        fr = new FileReader(f);
                        sc = new Scanner(fr);
                        //mientras no encuentre el final sigue leyendo
                        while(sc.hasNext()){
                              System.out.print(sc.next());
                        }

            }catch(FileNotFoundException fn){
                  System.out.println("No se encuentra el fichero");
            }finally{
                  try{
                        sc.close();
                        fr.close();
                  }catch(IOException ioe){
                        System.out.println("Error al cerrar el fichero");
                  }
            }
      }
}
```