

UD16.- Persistencia de datos. Ficheros

1. Acceso al sistema de archivos.
2. Ficheros secuenciales.
 - A. Organización y acceso.
 - B. Operaciones sobre ficheros secuenciales.
 - C. Acceso a un fichero secuencial.
3. Jerarquía de flujos.
4. Ficheros secuenciales en Java.
 - A. Ficheros con Flujos de bytes.
 - B. Ficheros de texto. Flujos de caracteres.
 - a. Crear fichero de texto.
 - b. Leer fichero de texto carácter a carácter.
 - c. Leer fichero de texto línea a línea.

1. Sistema de archivos. Clase File.

```
File carpeta1=new File("c:/datos");
File archivo1=new File(carpeta1,"bd.txt");
```

```
String ruta="documentos/manuales/2014/java.doc";
ruta=ruta.replace('/',File.separatorChar);
```

El nombre de fichero es su ruta+nombre+.+extensión.

En un programa nunca se usan rutas absolutas. Se usan rutas relativas dentro del proyecto.

1. Sistema de archivos. Clase File

<code>String toString()</code>	Para obtener la cadena descriptiva del objeto
<code>boolean exists()</code>	Devuelve <code>true</code> si existe la carpeta o archivo.
<code>boolean canRead()</code>	Devuelve <code>true</code> si el archivo se puede leer
<code>boolean canWrite()</code>	Devuelve <code>true</code> si el archivo se puede escribir
<code>boolean isHidden()</code>	Devuelve <code>true</code> si el objeto File es oculto
<code>boolean isAbsolute()</code>	Devuelve <code>true</code> si la ruta indicada en el objeto File es absoluta.
<code>boolean equals(File f2)</code>	Compara <code>f2</code> con el objeto <code>File</code> y devuelve verdadero si son iguales
<code>int compareTo(File f2)</code>	Compara basado en el orden alfabético del texto (sólo funciona bien si ambos archivos son de texto) <code>f2</code> con el objeto <code>File</code> y devuelve cero si son iguales, un entero negativo si el orden de <code>f2</code> es mayor y positivo si es menor
<code>String getAbsolutePath()</code>	Devuelve una cadena con la ruta absoluta al objeto File.

1. Sistema de archivos. Clase File

<code>File getFileAbsolute()</code>	Como la anterior pero el resultado es un objeto File
<code>String getName()</code>	Devuelve el nombre del objeto File.
<code>String getParent()</code>	Devuelve el nombre de su carpeta superior si la hay y si no <code>null</code>
<code>File getParentFile()</code>	Como la anterior pero la respuesta se obtiene en forma de objeto File.
<code>boolean setReadOnly()</code>	Activa el atributo de sólo lectura en la carpeta o archivo.
<code>URL toURL()</code>	
<code>throws MalformedURLException</code>	Convierte el archivo a su notación URL correspondiente
<code>URI toURI()</code>	Convierte el archivo a su notación URI correspondiente

1. Sistema de archivos. Clase File

Métodos de carpetas

boolean isDirectory()	Devuelve true si el objeto File es una carpeta y false si es un archivo o si no existe.
boolean mkdir()	Intenta crear una carpeta y devuelve true si fue posible hacerlo
boolean mkdirs()	Usa el objeto para crear una carpeta con la ruta creada para el objeto y si hace falta crea toda la estructura de carpetas necesaria para crearla.
boolean delete()	Borra la carpeta y devuelve true si puedo hacerlo
String[] list()	Devuelve la lista de archivos de la carpeta representada en el objeto File.
static File[] listRoots()	Devuelve un array de objetos File, donde cada objeto del array representa la carpeta raíz de una unidad de disco.
File[] listfiles()	Igual que la anterior, pero el resultado es un array de objetos File.

1. Sistema de archivos. Clase File

Métodos de archivos

<code>long length()</code>	Devuelve el tamaño del archivo en bytes (en el caso del texto devuelve los caracteres del archivo)
<code>boolean createNewFile() throws IOException</code>	Crea un nuevo archivo basado en la ruta dada al objeto <code>File</code> . Hay que capturar la excepción <code>IOException</code> que ocurriría si hubo error crítico al crear el archivo. Devuelve <code>true</code> si se hizo la creación del archivo vacío y <code>false</code> si ya había otro archivo con ese nombre.
<code>static File createTempFile(String prefijo, String sufijo) throws IOException</code>	Crea un objeto <code>File</code> de tipo archivo temporal con el prefijo y sufijo indicados. Se creará en la carpeta de archivos temporales por defecto del sistema. El <code>prefijo</code> y el <code>sufijo</code> deben de tener al menos tres caracteres (el sufijo suele ser la extensión), de otro modo se produce una excepción del tipo <code>IllegalArgumentException</code> . Requiere capturar la excepción <code>IOException</code> que se produce ante cualquier fallo en la creación del archivo
<code>static File createTempFile(String prefijo, String sufijo, File directorio)</code>	Igual que el anterior, pero utiliza el directorio indicado.
<code>void deleteOnExit()</code>	Borra el archivo cuando finaliza la ejecución del programa

2.A. Ficheros secuenciales

Organización de ficheros: cómo se guardan los registros en un fichero.

Acceso a un fichero: para acceder a la información de un fichero se debe leer el registro de un fichero. A un fichero con organización secuencial solo puede accederse secuencialmente. Se lee registro a registro, tal y como están almacenados, para acceder a un registro hay que leer todos los anteriores.

2.B. Ficheros secuenciales

Operaciones sobre un fichero:

- **Abrir fichero:** indicar al SO ruta y nombre del archivo, y solicitarle los recursos necesarios para operar con él.

○ Modalidad de apertura fichero secuencial:

- Para lectura: se van a *Leer* registros del fichero. **Entrada**.



- Para escritura: se va a crear el fichero, si existe, se elimina y se crea de nuevo vacío. Se van a *Escribir* registros en el fichero.

Salida.



- Para adición: se van a *Escribir* registros en el fichero añadiéndolos al final del mismo. **Salida**



2.B. Ficheros secuenciales

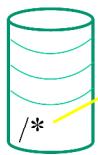
Operaciones sobre un fichero:

- **Consultar** un fichero: conseguir los datos de algún/os registro/s de un fichero, en un fichero secuencial sólo puede realizarse mediante sucesivas lecturas de sus registros. Uno a uno.
- **Actualización** de un fichero: cambiar los datos de un fichero, es decir, insertar nuevo/s registro/s, modificar o eliminar algún/os registros. En un fichero secuencial sólo se puede añadir registros al final del mismo.
- **Mantenimiento** de un fichero o **CRUD**: consultar y actualizar un fichero.
- **Clasificar** un fichero: recolocar los registros del fichero para que estén ordenados según el criterio indicado.
- **Cerrar** un fichero: liberar los recursos que el SO nos ha cedido para el uso del fichero.

Programación Java Nuria Fuentes Pérez



2.C. Acceso a un fichero secuencial



Marca de fin de
fichero (ff)

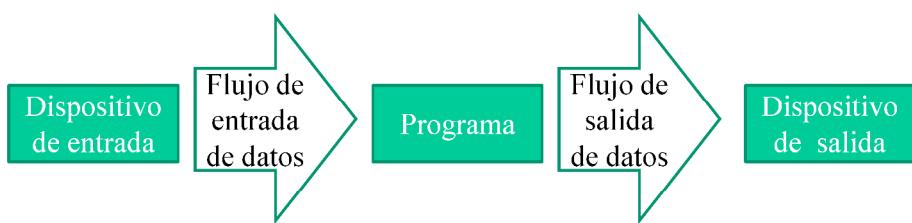
- ff(nFichero): devuelve verdadero si en la última operación de lectura se leyó la marca de fin de fichero.
- ff(nFichero): devuelve verdadero si el puntero del fichero está al final de fichero.

```
Abrir nFichero para lectura
Leer nFichero,nRegistro
Mientras no ff(nFichero) hacer
    // tratamiento del registro
    Leer nFichero, nRegistro
Fin Mientras
Cerrar nFichero
```

```
Abrir nFichero para lectura
Mientras no ff(nFichero) hacer
    Leer nFichero, nRegistro
    // tratamiento del registro
Fin Mientras
Cerrar nFichero
```

3. Jerarquía de flujos de datos

- ❖ Intercambio de datos entre el programa y el exterior
 - ✓ Diferentes dispositivos (fichero, pantalla, red, ...)
 - ✓ Diversidad de formas de comunicación
 - Modo de acceso: secuencial, aleatorio
 - Información intercambiada: binaria, caracteres, líneas
- ❖ Flujo (Stream)
 - ✓ Abstracción de cualquier fuente y/o destino de datos



3. Jerarquía de flujos de datos

Los flujos se implementan en las clases del paquete [java.io](#)

InputStream/OutputStream

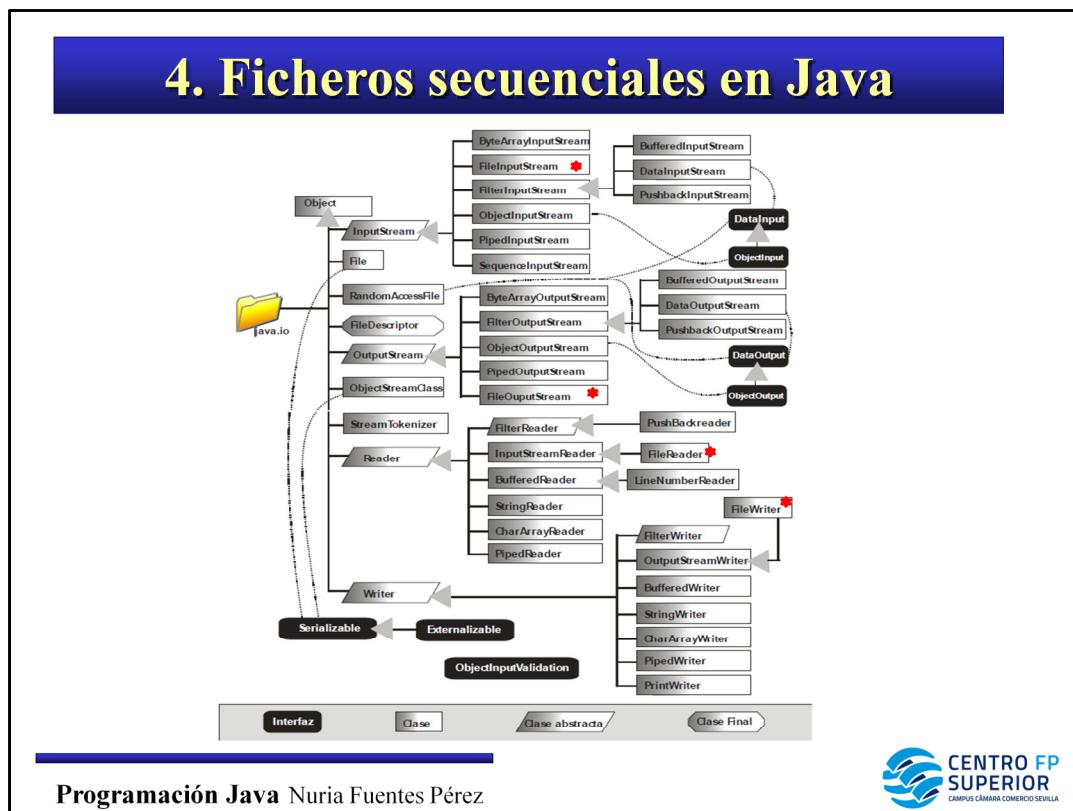
Estas dos clases **abstractas**, definen las funciones básicas de lectura y escritura de una secuencia de **bytes** pura (sin estructurar). Estos flujos de bits, no representan ni textos ni objetos, sino datos binarios puros. Tienen muchas subclases; de hecho casi todas las clases preparadas para la lectura y la escritura, derivan de estas.

Los métodos más importantes son **read (leer)** y **write (escribir)**, que sirven para leer un byte del dispositivo de entrada o escribir un byte respectivamente.

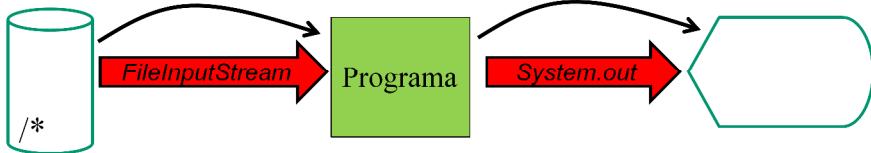
Reader/Writer

Clases abstractas que definen las funciones básicas de escritura y lectura basada en texto Unicode. Se dice que estas clases pertenecen a la jerarquía de lectura/escritura orientada a **caracteres**, mientras que las anteriores pertenecen a la jerarquía orientada a bytes.

4. Ficheros secuenciales en Java



4.A. Lectura de un fichero bytes.



Constructores

Constructor y descripción

`FileInputStream(File file)`

Crea FileInputStreamabriendo una conexión a un archivo real, el archivo nombrado por el File objeto fileen el sistema de archivos.

`FileInputStream(FileDescriptor fdObj)`

Crea un FileInputStreamutilizando el descriptor de archivo fdObj, que representa una conexión existente a un archivo real en el sistema de archivos.

`FileInputStream(String name)`

Crea FileInputStreamabriendo una conexión a un archivo real, el archivo nombrado por el nombre de la ruta name en el sistema de archivos.

EXCEPCIONES??

Fin de fichero
(ff)???

Programación Java Nuria Fuentes Pérez



4.A. Lectura de un fichero bytes.

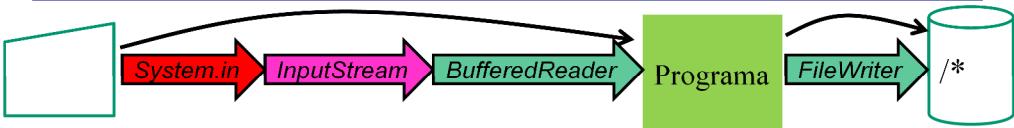
```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

class LeerFichero {
    public static void main(String[] args) {
        //Entorno:
        File f;
        int byt;
        FileInputStream fPrueba = null;
        //Algoritmo:
        try {
            f = new File("prueba.txt");
            fPrueba = new FileInputStream(f);
            byt = fPrueba.read();
            while (byt != -1) {
                System.out.print((char) byt);
                byt = fPrueba.read();
            }
        } catch (FileNotFoundException fnfe) {
            System.out.println("El fichero prueba.txt no existe");
        } catch (IOException ioe) {
            System.err.println("Error de E/S al cerrar");
        } finally {
            if (fPrueba != null) {
                try {
                    fPrueba.close();
                } catch (IOException ioe) {
                    System.err.println("Error de E/S al cerrar");
                }
            }
        }
    }
}
```

Programación Java Nuria Fuentes Pérez



4.B.a Creación de un fichero de texto



Constructores

Constructor y descripción

`FileWriter(File file)`

Construye un objeto FileWriter dado un objeto File.

`FileWriter(File file, boolean append)`

Construye un objeto FileWriter dado un objeto File.

`FileWriter(FileDescriptor fd)`

Construye un objeto FileWriter asociado con un descriptor de archivo.

`FileWriter(String fileName)`

Construye un objeto FileWriter dado un nombre de archivo.

`FileWriter(String fileName, boolean append)`

Construye un objeto FileWriter dado un nombre de archivo con un valor booleano que indica si se deben agregar o no los datos escritos.

EXCEPCIONES??

Programación Java Nuria Fuentes Pérez



4.B.a. Creación de un fichero de texto

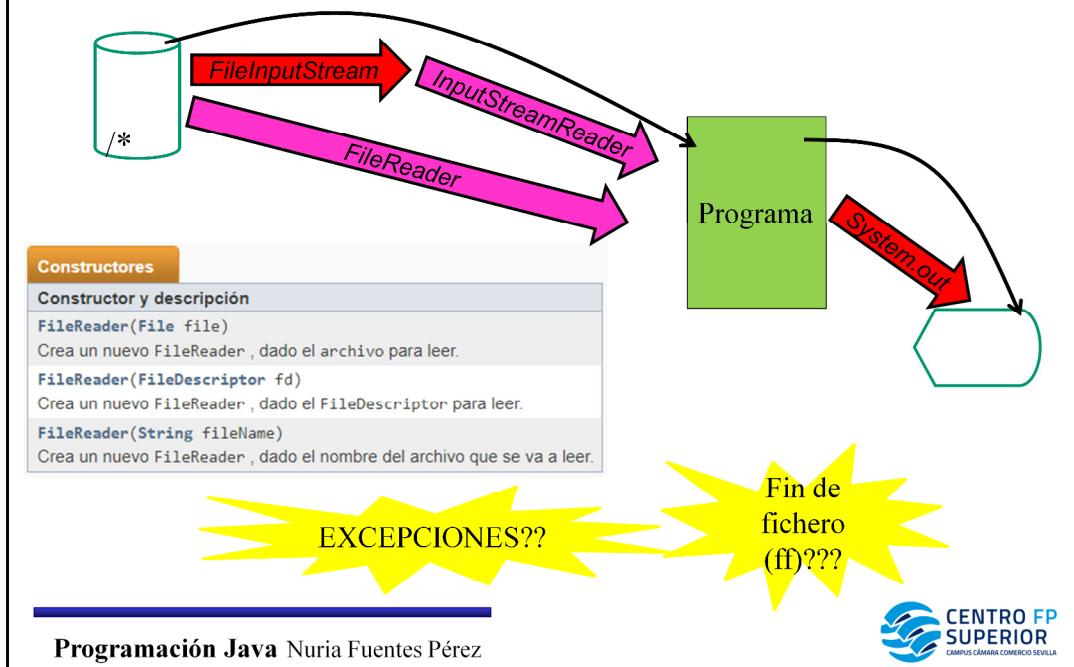
```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

class EscribirFicheroTexto {
    public static void main(String[] args) {
        //Entorno:
        File f;
        FileWriter fSalida;
        BufferedReader teclado;
        String linea;
        //Algoritmo:
        f = new File("salida.txt");//ruta por defecto
        fSalida = null;
        try {
            fSalida = new FileWriter(f);
            teclado = new BufferedReader(new InputStreamReader(System.in));
            linea = teclado.readLine();
            while (linea.length() > 0) {
                fSalida.write(linea + "\n");
                linea = teclado.readLine();
            }//Fin Mientras
        } catch (IOException ioe) {
            System.err.println("Error de E/S");
        } finally {
            if (fSalida != null) {
                try {
                    fSalida.close();
                } catch (IOException ioe) {
                    System.err.println("Error de E/S al cerrar");
                }//Fin Si
            }//Fin Try
        }//Fin Programa
    }
}
```

Programación Java Nuria Fuentes Pérez



4.B.b. Lectura carácter a carácter fichero de texto



4.B.b. Lectura carácter a carácter fichero de texto

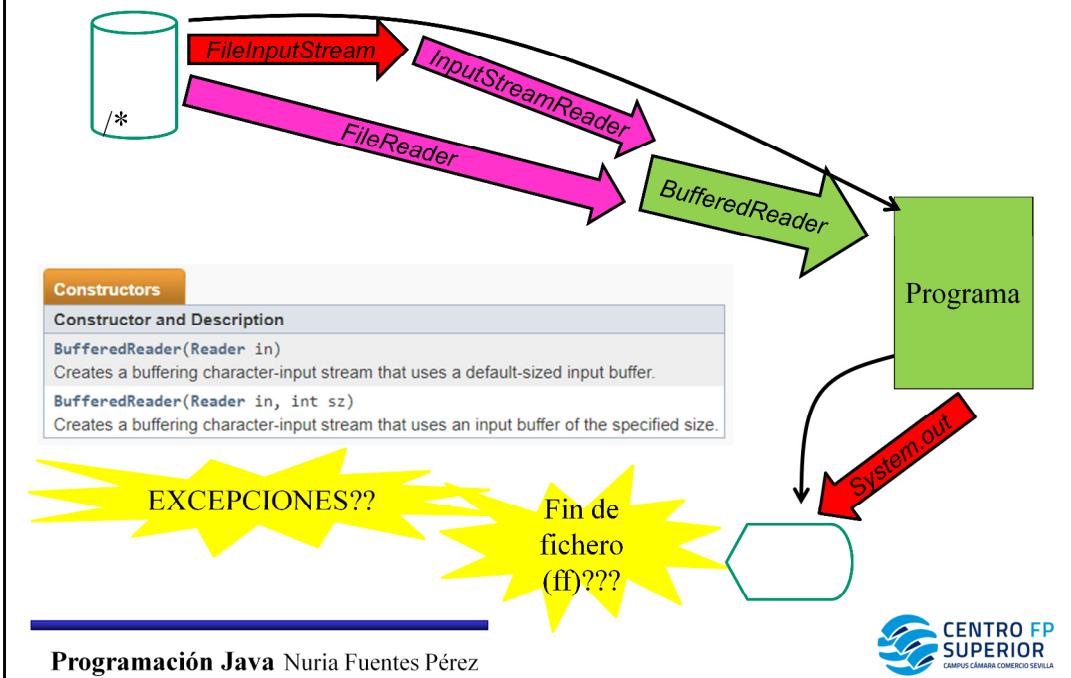
```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class LeerFicheroCar {
    public static void main(String[] args) {
        //Entorno:
        File f;
        FileReader fSalida;
        int byt;
        //Algoritmo:
        fSalida = null;
        try {
            f = new File("salida.txt");
            fSalida = new FileReader(f);
            byt = fSalida.read();
            while (byt != -1) {
                System.out.print((char) byt);
                byt = fSalida.read();
            }
        } catch (FileNotFoundException fnfe) {
            System.out.println("No existe el fichero salida.txt");
        } catch (IOException ioe) {
            System.out.println("Error de E/S al leer");
        } finally {
            if (fSalida != null) {
                try {
                    fSalida.close();
                } catch (IOException ioe) {
                    System.out.println("Error de E/S al cerrar");
                }
            }
        }//Fin Try
        }//Fin Si
    }//Fin Programa
}
```

Programación Java Nuria Fuentes Pérez



4.B.c. Lectura línea a línea de fichero de texto



4.B.c. Lectura línea a línea de fichero de texto

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class LeerFicherolinea {
    public static void main(String[] args) {
        //Entorno:
        String linea;
        BufferedReader fSalida;
        //Algoritmo:
        fSalida = null;
        try {
            fSalida = new BufferedReader(new FileReader("salida.txt"));
            linea = fSalida.readLine();
            while (linea != null) {
                System.out.println(linea);
                linea = fSalida.readLine();
            }//Fin Mientras
        } catch (FileNotFoundException fnfe) {
            System.out.println("No existe salida.txt");
        } catch (IOException ioe) {
            System.out.println("Error de E/S al leer");
        } finally {
            if (fSalida != null) {
                try {
                    fSalida.close();
                } catch (IOException ioe) {
                    System.out.println("Error de E/S al cerrar fichero");
                }//Fin Try
            }//Fin Si
        }//Fin Try
    }//Fin Programa
}
```

Programación Java Nuria Fuentes Pérez

