UT2 Ficheros

Acceso a Datos

Persistencia de datos en ficheros

Son la forma más elemental de almacenamiento de información. en última instancia todos los sistemas almacenan en ficheros

Hasta los años 80 fueron lo que se usaba, después llegaron las BBDD relacionales.

En este tema vamos a ver cómo programar en JAVA la lectura, escritura, modificación y borrado de datos en ficheros.

Tipos de ficheros y codificación

Tipos

Texto: caracteres, espacios, números, retornos de carro…

Binarios: Cualquiera que no sea texto.

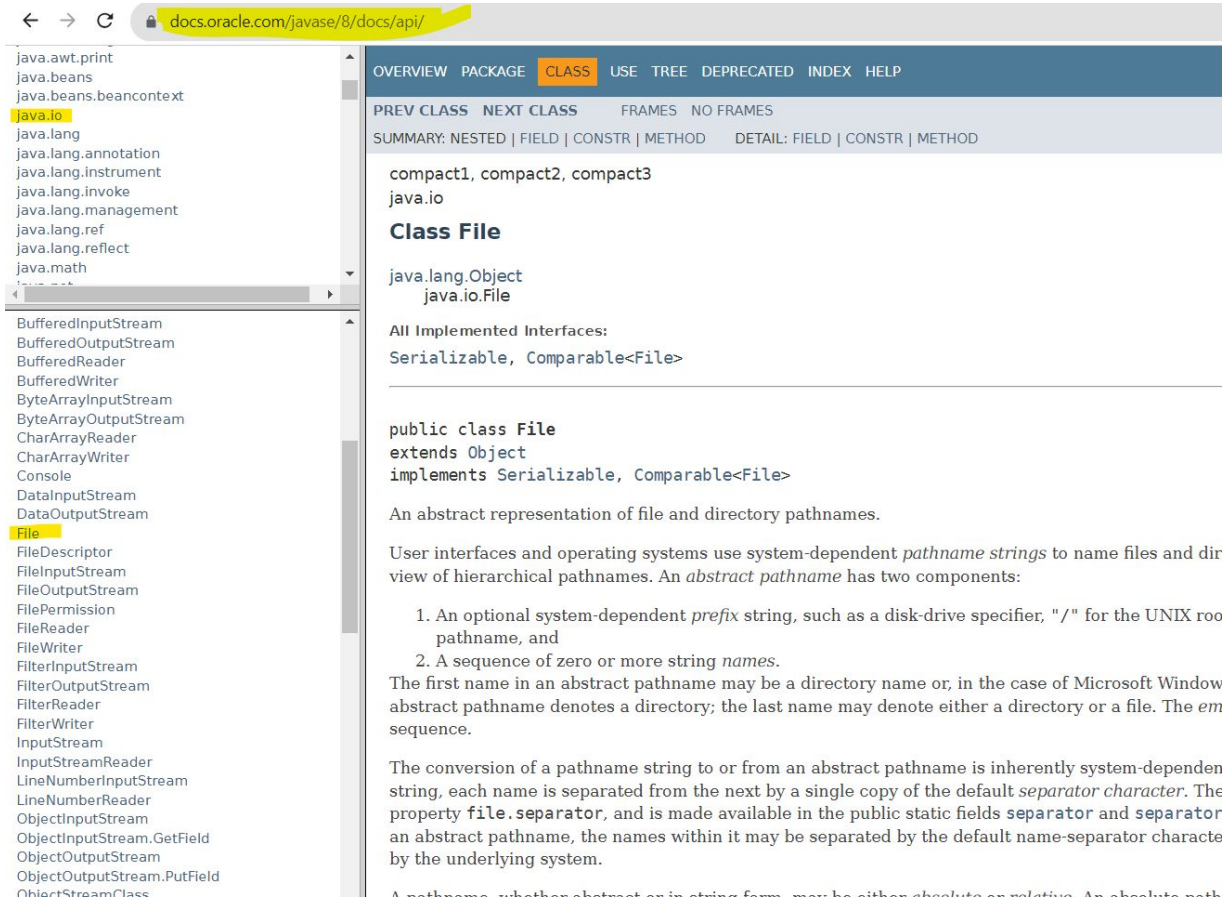
Codificación: Un fichero se almacena en memoria como una secuencia de bytes y dependiendo de la codificación, esos bytes serán distintos

La más utilizada y la que debemos usar es **UTF-8**

Clase File Java

Consultar: https://docs.oracle.com/javase/8/docs/api/

Pinchamos en java.io, clase File



Métodos de la clase File

Categoría	Modificador/tipo	Método(s)	Funcionalidad
Constructor		`File(String ruta)`	Crea objeto `File` para la ruta indicada, que puede corresponder a un directorio o a un fichero.

Consulta de propiedades	`boolean`	`canRead()`
		`canWrite()`
		`canExecute()`
			Comprueban si el programa tiene diversos tipos de permisos sobre el fichero o directorio, tales como de lectura, escritura y ejecución (si se trata de un fichero). Para un directorio, `canExecute()` significa que se puede establecer como directorio actual.

	`boolean`	`exists()`	Comprueba si el fichero o directorio existe.
	`boolean`	`isDirectory()` `isFile()`	Comprueban si se trata de un directorio o de un fichero.
	`long`	`length()`	Devuelve longitud del fichero.
	`File`	`getParent()` `getParentFile()`	Devuelven el directorio padre.
	`String`	`getName()`	Devuelve nombre del fichero.
Enumeración	`String[]`	`list()`	Devuelve un *array* con los nombres de los directorios y ficheros dentro del directorio.
	`File[]`	`listFiles()`	Devuelve un *array* con los directorios y ficheros dentro del directorio.
Creación, borrado y renombrado	`boolean`	`createNewFile()`	Crea nuevo fichero.
	`static File`	`createTempFile()`	Crea nuevo fichero temporal y devuelve objeto de tipo `File` asociado, para poder trabajar con él.
	`boolean`	`delete()`	Borra fichero o directorio.
	`boolean`	`renameTo()`	Renombra fichero o directorio.
	`boolean`	`mkdir()`	Crea un directorio.
Otras	`java.nio.file.Path`	`toPath()`	Devuelve un objeto que permite acceder a información y funcionalidad adicional

Ejemplo de programa que muestra el contenido de un directorio

```
import java.io.File;
public class ListaDirectorio {
   public static void main(String[] args) {
       //String ruta=".";
       //String ruta="Documents"; //no existe
       String ruta="C://Users//vanma//FCTs Las Encinas/";
       //String ruta="C://Users//vanma//Documents/FCTs Las Encinas/fp-fcts-anexo5-autorizacion-excepc-periodos.doc"; //es un fichero
       if(args.length>=1) ruta=args[0];
       File fichero=new File(ruta);
       if(!fichero.exists()){
           System.out.println("No existe el fichero o directorio("+ruta+").");
       }
       else{
           if(fichero.isFile()){
               System.out.println(ruta+" es un fichero");
           }
           else{
               System.out.println(ruta+" es un directorio: Contenidos: ");
               File [] directorio=fichero.listFiles();
               for(File f:directorio){
                   String texto=f.isDirectory() ? "/":f.isFile() ? "_":"?";
                   System.out.println("("+texto+")"+f.getName());
               }
           }
       }
   }
}
```

Actividad 1

Modifica el programa anterior para que además muestre:

1. Tamaño, si es un fichero. Usa `length()`
2. Permisos,
    - si tiene de lectura que muestre un r,
    - w para escritura
    - y x para ejecución.
3. Fecha de la última modificación dándole un formato. Usa `SimpleDateFormat`

Actividad 2

¿Serías capaz de crear un nuevo directorio en una ruta?

Cambiarle el nombre.

Borrarlo.

Actividad 3: Entrega en el Aula Virtual

Vamos a pedirle al usuario que introduzca la ruta del fichero o el directorio y que elija con un menú si quiere:

- Mostrar info (como en la Actividad 1)
- Crear un nuevo directorio.
- Renombrar un directorio. Debe pedirle al usuario el nuevo nombre.
- Borrar un directorio.

Acceso a ficheros

Ficheros de texto o binarios, acceso secuencial o aleatorio, se accede de la misma forma:

Usando un **puntero** y una zona de **memoria** que se llama `buffer`

El puntero apunta a una zona de memoria o bien al final del fichero: **EOF (EndOfFile)**

Operaciones sobre ficheros

Apertura. Antes de hacer nada con un fichero, hay que abrirlo. Esto se hace al crear una instancia de una clase que se utilizará para operar con él.

Lectura. Mediante el método `read ()`. Consiste en leer contenidos del fichero para volcarlos a memoria y poder trabajar con ellos. El puntero se sitúa justo después del último carácter leído.

Salto. Mediante el método `skip()`. Consiste en hacer avanzar el puntero un número determinado de bytes o caracteres hacia delante.

Escritura. Mediante el método `write()`. Consiste en escribir contenidos de memoria en un lugar determinado del fichero. El puntero se sitúa justo después del último carácter escrito.

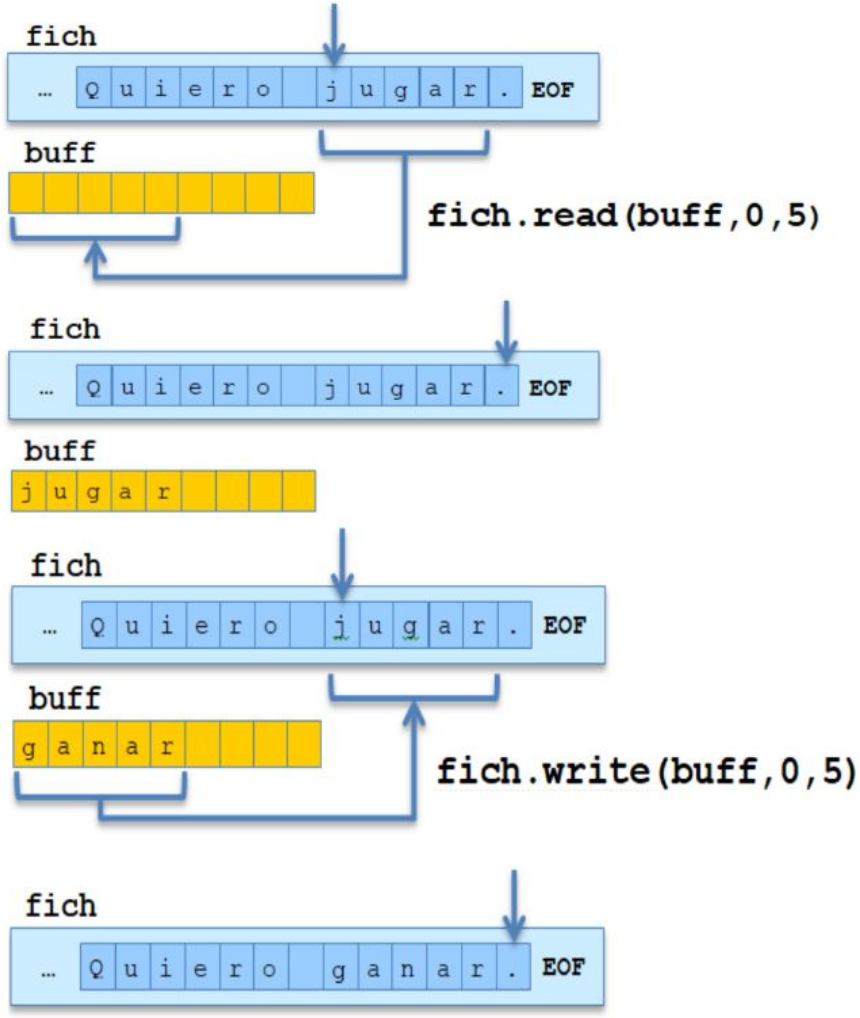
Cierre. Mediante el método `close ()`. Para terminar, hay que cerrar el fichero.

Lectura

Indicar el **buffer** en el que se guardaran los datos y el tamaño en **bytes** que se va a leer, si no especificamos se llena hasta que se llena el buffer.

Escritura

Indicar el **buffer** y el tamaño en **bytes**, desde el buffer se transfieren al fichero en la posición que indica el puntero.



Clases básicas en entrada y salida

	Fuente de datos	Lectura	Escritura
Flujo binario	Ficheros	`FileInputStream`	`FileOutputStream`
	Memoria (`byte[]`)	`ByteArrayInputStream`	`ByteArrayOutputStream`
	Tuberías	`PipedInputStream`	`PipedOutputStream`
Flujo de texto	Ficheros	`FileReader`	`FileWriter`
	Memoria (`char[]`)	`CharArrayReader`	`CharArrayWriter`
	Memoria (`String`)	`StringReader`	`StringWriter`
	Tuberías	`PipedReader`	`PipedWriter`

Buffering

Permite acelerar los procesos de lectura, ya que en cada acceso los últimos datos ya estarán en buffer. Ocurre lo mismo con la escritura, se pasan al buffer y después al fichero.

Flujo sin buffering	Flujo con buffering
`new FileInputStream("f.bin")`	`new BufferedInputStream(new FileInputStream("f.bin"))`
`new FileOutputStream("f.bin")`	`new BufferedOutputStream(new FileOutputStream("f.bin"))`
`new FileReader("f.txt")`	`new BufferedReader(new FileReader("f.txt"))`
`new FileWriter("f.txt")`	`new BufferedWriter(new FileWriter("f.txt"))`

Métodos de lectura y escritura para buffering en ficheros de texto

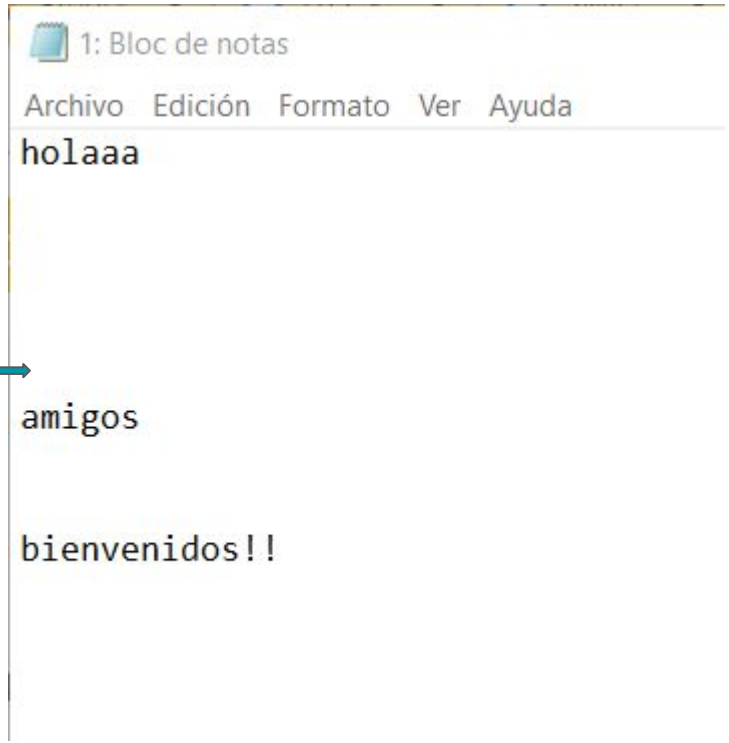
Clase	Método	Funcionalidad
`BufferedReader`	`String readLine()`	Lee hasta el final de la línea actual.
`BufferedWriter`	`void newLine()`	Escribe un separador de líneas. El separador de líneas puede depender del sistema operativo, y suele ser distinto en Linux y en Windows. El método `readLine()` de `BufferedReader` tiene en cuenta estas particularidades.

Ejemplo de lectura de líneas de un fichero de texto

Vamos a crear un archivo de texto en una ruta, por ejemplo:

`C:/temp/1.txt`

En ese archivo .txt escribimos algo de texto.



Ejemplo de lectura de líneas de un fichero de texto

```
import java.io.*;
public class LecturaFicheroTexto { //para leer texto de un fichero
    public static void main(String[] args) {
       File f=new File("C:/temp/1.txt");
       try{
           FileReader fr=new FileReader(f);
           BufferedReader br= new BufferedReader(fr);
           String linea="";
           while((linea=br.readLine())!=null)
               System.out.println(linea);
       }
       catch (FileNotFoundException e){
           e.printStackTrace();
       }
       catch (IOException e){
           e.printStackTrace();
       }
       catch(Exception e){
           e.printStackTrace();
       }

    }
}
```

Actividad 4: Entrega en el Aula Virtual

Modifica el código anterior `LecturaFicheroTexto` para que busque un texto en el fichero.

Debe pedirle al usuario el fichero y el texto a buscar por teclado.

Debe leer línea a línea el texto y usar el método apropiado de clase String de Java para encontrar las ocurrencias.

Ejemplo de escritura en un fichero

```
import java.io.*;
public class EscrituraFicheroTexto { //escribe un texto en un fichero de texto
   public static void main (String [] args){
        File f =new File("C:/temp/f2.txt");
        FileWriter fw=null;
        PrintWriter pw=null;
        try{
            fw=new FileWriter(f);
            pw=new PrintWriter(fw);

            pw.println("AAAAA");
            pw.println("");
            pw.println("BBBBBB");

        }catch(IOException e){
            e.printStackTrace();

        }
        finally{
            try{fw.close();
            pw.close();}
            catch(IOException e){ e.printStackTrace();}
        }
   }
}
//en el de lectura también deberíamos cerrar el fichero!!
```

Ejemplo de Lectura Escritura

copia el contenido de un fichero existente en otro que crea

```
import java.io.*;

   public class LeeEscribe {
       public static void main(String[] args) {
           try (FileReader inputStream =
                       new FileReader("C:/temp/entrada.txt");
                FileWriter outputStream =
                       new FileWriter("C:/temp/salida.txt")) {
               int c;
               while ((c = inputStream.read()) != -1) {
                   outputStream.write(c);
               }
           }
           catch(FileNotFoundException fnfEx) {
               System.out.println("El archivo de entrada no existe.");
           }
           catch(IOException ioEx) {
               System.out.println("El archivo de salida no se pudo abrir.");
           }
       }
   }
```

Ejemplo de Lectura Escritura con buffer

```
import java.io.*;
public class LeeEscribeBuffer {
   public static void main(String[] args) {
       try (BufferedReader inputStream =
                   new BufferedReader(
                           new FileReader("C:/temp/entrada.txt" ));
            PrintWriter outputStream =
                   new PrintWriter(
                           new BufferedWriter (
                                   new FileWriter("C:/temp/salida.txt" )))) {

           String linea;
           while ((linea= inputStream.readLine()) !=  null) {
               outputStream.println(linea);
           }
       }
       catch(FileNotFoundException fnfEx) {
           System.out.println("El archivo de entrada no existe." );
       }
       catch(IOException ioEx) {
           System.out.println("El archivo de salida no se pudo abrir." );
       }
```

Ejemplo de lectura y escritura en un fichero

Abrimos un fichero de texto,

lo leemos y buscamos espacios en blanco que eliminar si son al principio de línea

o sustituir por un espacio único si es dentro de línea,

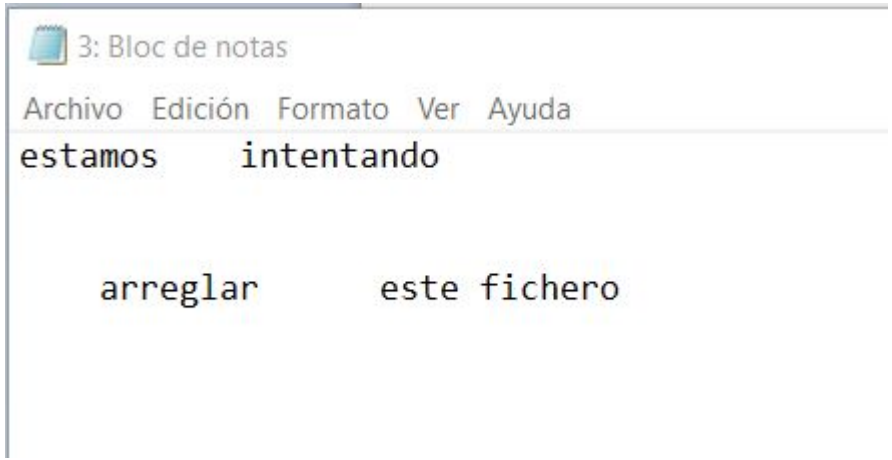
cambiamos la primera letra de línea a mayúsculas.

Se crea un archivo temporal para guardar el nuevo archivo modificado.

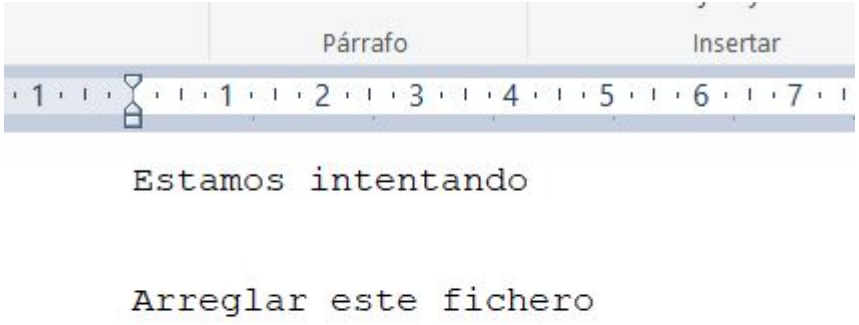
Ejemplo de lectura y escritura en un fichero

```
import java.io.File;
import java.io. BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.io.*;

public class ArreglarFichero {
   public static void main (String[] args){
       String nomFichero="C:/temp/3.txt";
       File f=new File(nomFichero);
       if(!f.exists()){
           System.out.println("El fichero "+nomFichero+" no existe");
           return;
       }
       try(BufferedReader bfr=new BufferedReader(new FileReader(f))){

           File temp=File.createTempFile(nomFichero,"");
           System.out.println("Creamos un fichero temporal
"+temp.getAbsolutePath());

           FileWriter wr=new FileWriter(temp);
           BufferedWriter bfw=new BufferedWriter(wr);

           String linea =bfr.readLine();
           while (linea!=null){
               boolean princLinea=true, espacios=false,primerLetra=false;
               for(int i=0;i<linea.length();i++){
                   char c=linea.charAt(i);
                   if(Character.isWhitespace(c)){
                       if(!espacios && !princLinea){
                           bfw.write(c);
                       }
                       espacios=true;
                   }else if(Character.isAlphabetic(c)){
                       if(!primerLetra){
                           bfw.write(Character.toUpperCase(c));
                           primerLetra=true;
                       }
                       else{
                           bfw.write(c);
                           espacios=false;
                           princLinea=false;
                       }
                   }
               }
               bfw.newLine();
               linea=bfr.readLine();
           }
           bfw.close();

       }
       catch (IOException e){
           System.out.println(e.getMessage());
       }
       catch(Exception e){
           e.printStackTrace();
       }
   }
}
```

Ejemplo de lectura y escritura en un fichero



Los archivos temporales se guardan en una carpeta temporal: C:\Users\vanma\AppData\Local\Temp

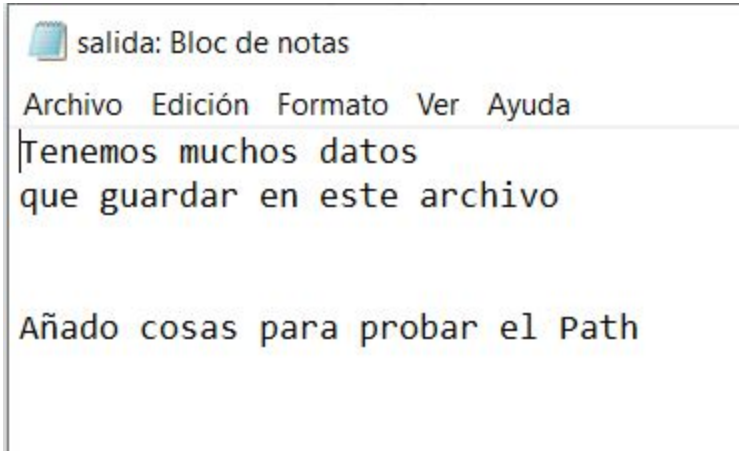


Gestión de ficheros con java nio Path

```
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Path;

import java.io.*;
import java.nio.file.Paths;
import java.util.List;

public class LeeLineas {

   public static void main(String[] args) {
        Path entrada = Paths.get("C:/temp/entrada.txt");
        Path salida = Paths.get("C:/temp/salida.txt");
        //Lista de cadenas para leer las lineas
        List<String> fileLista;
        try {
//Leemos de una vez el archivo de caracteres con java.nio
            fileLista = Files.readAllLines(entrada, Charset.forName("UTF-8"));
//Escribimos una vez el archivo de caracteres con java.nio
            Files.write(salida, fileLista,Charset.forName("UTF-8"));
        } catch (IOException e) {
            System.err.format("IOException: %s%n", e);
        }
   }
}
```

File vs Path

En el fichero salida se copian las líneas de entrada.txt



Práctica 1

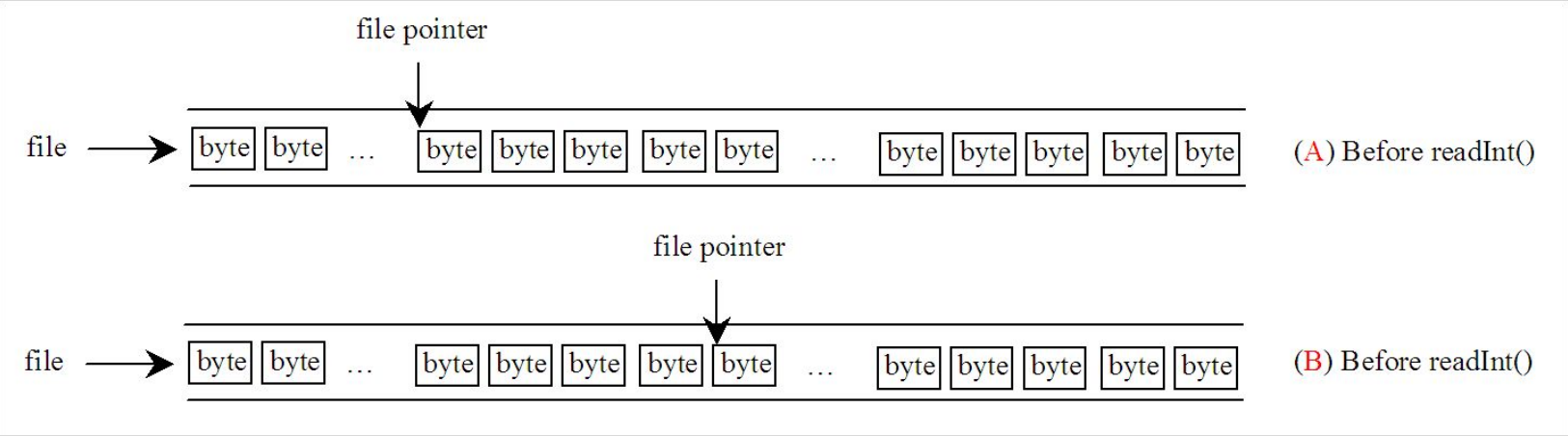
Realiza los ejercicios del pdf que encontrarás en el Aula Virtual.

Operaciones con ficheros de acceso aleatorio

Usaremos la clase RandomAccessFile

Podremos realizar operaciones de lectura y escritura sobre el fichero.

Se puede poner el cursor donde lo necesitemos usando seek()



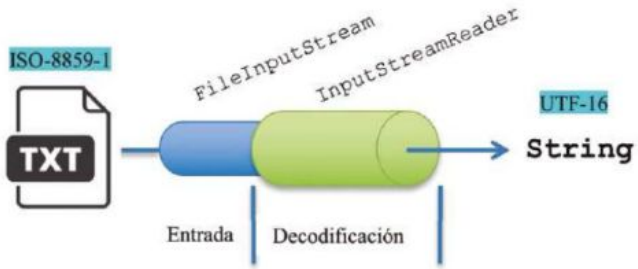
Método	Funcionalidad
`RandomAccessFile(File file, String mode)` `RandomAccessFile(String name, String mode)`	Constructor. Abre el fichero en el modo indicado, si se dispone de permisos suficientes. *r:* modo de solo lectura. *rw:* modo de lectura y escritura. *rwd, rws:* Como *rw* pero con escritura síncrona. Esto significa que todas las operaciones de escritura (de datos con *rwd* y de datos y metadatos con *rws*) deben haberse completado cuando termina la función. Esto puede hacer que la llamada a la función tarde más, pero asegura que no se pierde información crítica ante una caída del sistema.
`void close()`	Cierra el fichero. Es conveniente hacerlo siempre al final.
`void seek(long pos)`	Posiciona el puntero en la posición indicada.
`int skipBytes(int n)`	Intenta avanzar el puntero el número de *bytes* indicado. Se devuelve el número de *bytes* que se ha avanzado. Podría ser menor que el solicitado, si se alcanza el fin del fichero.

`int read()`
`int read(byte[] buffer)`
`int read(byte[] buffer, int offset, int longitud)`
	Lee del fichero. Según la variante, un *byte* o hasta llenar el *buffer,* o el número de *bytes* indicados, que se copiarán en la posición indicada *(offset)* del *buffer*. Devuelve el número de *bytes* leídos, o -1 si no se pudo leer nada porque el puntero estaba al final del fichero.

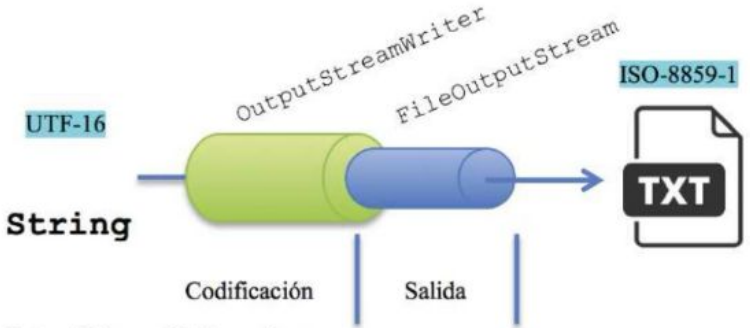
`void readFully(byte[] buffer)` `readFully(byte[] buffer, int offset, int longitud)`	Como `int read(byte[] b)`, pero si no se puede leer hasta llenar el *buffer,* o el número de *bytes* indicado, porque se llega al final del fichero, se lanza la excepción `IOException`. Útil cuando se sabe que se podrá leer hasta llenar el *buffer* o el número de *bytes* indicados. En cualquier caso, la eventualidad de que no se pueda completar la lectura se puede gestionar capturando la excepción.
`String readLine()`	Lee hasta el final de la línea de texto actual.

`void write(int b)`
`void write(byte[] buffer)`
`void write(byte[] buffer, int offset, int longitud)`
	Escribe en el fichero. Según la variante, un *byte,* o todos los contenidos del *buffer,* o el número de *bytes* indicados a partir de la posición indicada *(offset)*. Si alcanza el fin del fichero, siguen escribiendo.

Entrada salida con recodificación de texto




```
InputStreamReader fisr =
new InputStreamReader(new FileInputStream("f.8859_1.txt "), "ISO-8859-1 ");
```




```
OutputStreamWriter fosw =
new OutputStreamWriter(new FileOutputStream("f.8859_1.txt "), "ISO-8859-1 ");
```