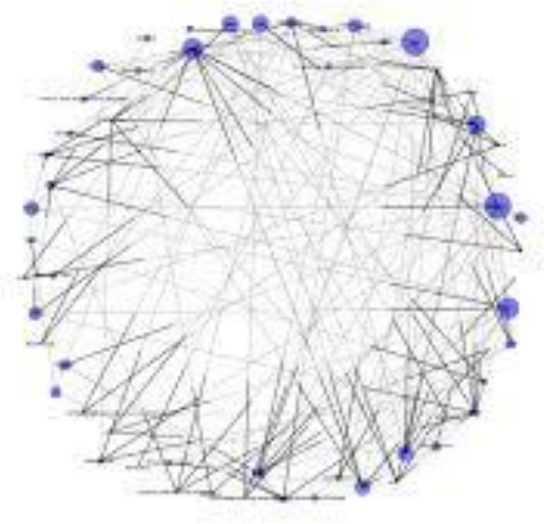
DAM
ACCESO A DATOS



UD1
MANEJO DE FICHEROS EN JAVA
1. CLASE FILE Y PATH

Introducción

- Un **archivo** es un **conjunto de bits almacenados en un soporte digital** que se identifica por un **nombre**, la **carpeta** donde están almacenados, un **tamaño** y un **formato**.

- Los archivos permiten **almacenar información de forma permanente** en los dispositivos.

- Un archivo está formado por un conjunto de **registros**, y los registros están formados por **campos**, que es la unidad que contiene la información en sí.

- La forma de organizar la información en campos y registros depende de su **diseño inicial**, que todo archivo requiere.

Clases asociadas con la gestión de archivos

- Paquete **java.io**:
  - Contiene las clases necesarias para manejar las entradas y salidas en Java.
  - Es necesario importar el paquete java.io para poder usar las clases que contiene.
- Clase **File**:
  - Pertenece a las primeras versiones de Java.
  - Las entidades de la clase **File** permite el manejo de un fichero y la gestión de los mismos.
  - Operaciones:
    - Ver los nombres y otros atributos de un archivo.
    - Crear carpetas,
    - Etc.

Clase File: constructores

FILE-constructores	Uso
File (String ruta+nombreFichero)	new File (“D:\\directorio\\fichero.txt”)
File (String ruta, String nombreFichero)	new File (“directorio”, “fichero.txt”)
File (File ruta, String nombreFichero)	new File (objetoFiledirectorio, “fichero.txt”)

Métodos para crear ficheros/carpetas	Uso
createNewFile()	Crea un fichero asociado al objeto **File**. Devuelve *true* si se pudo crear. Para poder crear un fichero es necesario que no exista previamente. Lanza una excepción de tipo **IOException**. ficheroFile.createNewFile()
mkdir()	*Crea una carpeta en la ruta indicada, siempre que estén creados todas las carpetas padre.* Devuelve *true* si se creo con éxito. carpetaFile.mkdir()
mkdirs()	Crea un directorio incluyendo los directorio no existentes especificados en la ruta. Devuelve *true* si se creo con éxito. carpetaFile.mkdirs()

```java
public static void main(String[] args) {
    //Crea un fichero en la ruta donde está el proyecto
    File miFile = new File("mifichero.txt");
    //Es obligatorio hacer un try/catch al crear un fichero
    try {
        miFile.createNewFile();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Crea un fichero en la carpeta donde está el proyecto

```java
public static void main(String[] args) {
    boolean creado;

    //Crea una carpeta que no existe
    File miCarpeta = new File("d:\\carpeta");
    creado = miCarpeta.mkdir();
    System.out.println("Se ha creado la carpeta " + creado);

    //Crear una ruta. Si las carpetas no existen, las crea
    File miRuta = new File("d:\\carpeta\\cHija1\\cHija2");
    creado = miRuta.mkdirs();
    System.out.println("Se ha creado la ruta " + creado);
}
```

Crea carpetas y rutas

✍ Crea un fichero en la ruta miRuta

Clase File: métodos I

MÉTODO	DESCRIPCIÓN
boolean canRead()	Devuelve *true* si se puede leer el fichero.
boolean canWrite()	Devuelve *true* si se puede escribir en el fichero.
boolean delete()	Elimina el fichero o directorio. Si es un directorio, esté debe estar previamente vacío. Devuelve *true* si la operación se realizó con éxito.
boolean exists()	Devuelve *true* si el fichero o directorio existe.
String getName()	Devuelve el nombre del fichero o directorio si éste existe.
String getAbsolutePath()	Devuelve la ruta absoluta asociada al la instancia de **File**.
String getCanonicalPath()	Devuelve o ruta único absoluto asociada al objeto File. Pueden existir varias rutas absolutas asociadas a una instancia de File, pero solo una es una ruta canónica. Lanza una excepción de tipo **IOException**.
String getPath()	Devuelve la ruta con la que se creó el objeto. Puede ser relativa o no.
String getParent()	Devuelve un *String* con el nombre del directorio padre de la instancia de **File**. Devuelve **null** se non tiene directorio padre.
File getParentFile()	Devuelve un objeto File que contiene al directorio padre de la instancia actual de File. Devuelve null se no tiene directorio padre.

Clase File: métodos II

MÉTODO	DESCRIPCIÓN
boolean isAbsolute()	Devuelve *true* si es una ruta absoluta.
boolean isDirectory()	Devuelve *true* si es un directorio válido.
boolean isFile()	Devuelve *true* si es un fichero válido.
String[] list()	Devuelve un array de *String* con el nombre de los archivos y directorio que contiene el directorio indicado en el objeto **File**. Se no es un directorio devuelve *null*. Se el directorio está vacío devuelve un array vacío.
String[] list(FilenameFilter filtro)	Similar al anterior. Devuelve un array de *String* con el nombre de los archivos y directorio que contiene el directorio indicado en el objeto **File** que cumple con el filtro indicado.
boolean renameTo(File dest)	Cambia el nombre del fichero indicado en el parámetro *dest*. Devuelve *true* si se realizó el cambio correctamente.

File-Ejercicio1

```
public static void main(String[] args) {
    boolean creado = false;
    //Creo un objeto File para la carpeta que ya existe
    File miCarpeta = new File("d:\\unaCarpeta");
    //Creo un objeto File para el fichero nuevo en la carpeta existente
    File miFichero = new File(miCarpeta, "mifichero1.txt");
    try {
        //Creo el fichero en el disco
        creado = miFichero.createNewFile();
    } catch (IOException ex) {
        System.out.println(ex.toString());
    }
    //Muestro resultados relacionados con el objeto File
    System.out.println(creado);
    System.out.println(miFichero.getAbsolutePath());
    System.out.println(miFichero.getName());
    System.out.println(miFichero.isDirectory());
    System.out.println(miCarpeta.isDirectory());
}
```

Muestra distintas informaciones del fichero/carpeta

✍ Pon controles para crear la carpeta o el fichero sí no está creado previamente

File-Ejercicio2

Crea un array de objetos de tipo File y muestra sus nombres

```
public static void main(String[] args) {

    //Crea un objeto File de la carpeta existente
    File miCarpeta = new File("d:\\unaCarpeta");

    /*Almacena en un array de File las carpetas y ficheros
    que contiene miCarpeta*/
    File[] misFiles = miCarpeta.listFiles();

    //Recorrer el array para mostrar los nombres de los objetos
    for (File misFile : misFiles) {
        System.out.println(misFile.getName());
    }

}
```

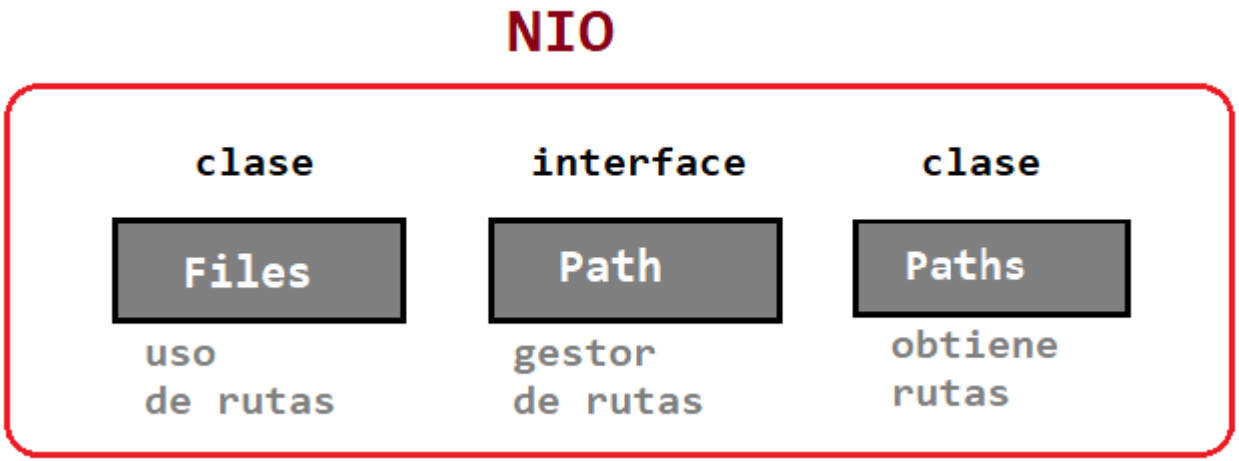
- Realiza la misma operación utilizando el for clásico.
- Para cada objeto muestra si es un fichero o una carpeta

File-Ejercicio3

Conjunto de métodos de File

```
File ruta = new File("D:\\IES\\Teis\\otro1");
File f = new File(ruta, "datos.txt");
System.out.println("1----------------");
System.out.println(f.getAbsolutePath());
System.out.println(f.getParent());
System.out.println(ruta.getAbsolutePath());
System.out.println(ruta.getParent());
System.out.println("2----------------");
if (!f.exists()) {  //se comprueba si el fichero existe o no
    System.out.println("Fichero " + f.getName() + " no existe");
    if (!ruta.exists()) {  //se comprueba si la ruta existe o no
        System.out.println("El directorio " + ruta.getName() + " no existe");
        if (ruta.mkdir()) { //se crea la ruta. Si se ha creado correctamente
            System.out.println("Directorio creado");
            try {
                if (f.createNewFile()) {  //se crea el fichero. Si se ha creado correctamente
                    System.out.println("Fichero " + f.getName() + " creado");
                } else {
                    System.out.println("No se ha podido crear " + f.getName());
                }
            } catch (IOException ex) {
                System.out.println(ex.toString());
            }
        } else {
            System.out.println("No se ha podido crear " + ruta.getName());
        }
    } else {
                 try {
            //si la ruta existe creamos el fichero
            if (f.createNewFile()) {
                System.out.println("Fichero " + f.getName() + " creado");
            } else {
                System.out.println("No se ha podido crear " + f.getName());
            }
        } catch (IOException ex) {
           System.out.println(ex.toString());
        }
    }
} else { //el fichero existe. Mostramos el tamaño
    System.out.println("Fichero " + f.getName() + " existe");
    System.out.println("Tamaño " + f.length() + " bytes");
}
```

Java 8 y la clase New IO II



- A partir de Java 8 tenemos varias herramientas (clases e interface) para trabajar con rutas y ficheros de forma más eficaz.
  - **Files**: clase de utilidad con operaciones básicas sobre ficheros.
  - **Path**: interfaz que se puede utilizar con rutas de ficheros.
  - **Paths**: clase que permite obtener las rutas (Path).

PrintStream).

Path-Ejercicio1

```
public static void main(String[] args) {

    System.out.println("Información de la ruta actual");

    Path relative = Paths.get(".");
    Path absolute = relative.toAbsolutePath().normalize();

    System.out.printf("Ruta relativa: %s\n", relative);

    System.out.printf("Ruta absoluta: %s\n", absolute);

    System.out.printf("Nº carpetas de ruta abs.: %d\n", absolute.getNameCount());

    System.out.printf("Carpeta padre: %s\n", absolute.getParent());

    System.out.printf("Subcarpetas de la ruta (0, 1): %s\n", absolute.subpath(0, 1));
}
```

- Documenta los métodos utilizados
- Modifica la ruta relativa por
  - ..\\
  - ..\\..
- Explica la operación realizada en cada caso

La clase **System** tiene una propiedad estática de la clase **PrintStream** denominada **out**.
Por lo tanto, **printf()** es un método de la clase **PrintStream.**
Conoce su funcionamiento

Path-Ejercicio2

```
public static void main(String[] args) {
    boolean creado = false;

    File miFichero = new File("mifichero1.txt");
    try {
        creado = miFichero.createNewFile();
    } catch (IOException ex) {
        System.out.println(ex.toString());
    }

    Path ruta1 = Paths.get("mifichero.txt");

    System.out.println(ruta1.getFileName());
    System.out.println(ruta1.toAbsolutePath());

    Path ruta2 = ruta.toAbsolutePath();

    for (Path miniruta : ruta2) {
        System.out.println(miniruta);
    }
}
```

✍ Introduce comentarios descriptivos de la operación que se realiza en cada caso

Path-Ejercicio3

```
public static void main(String[] args) {

    Path ruta = Paths.get("mifichero.txt");
    Path rutaDestino = Paths.get("c:\\mifichero.txt");

    try {
        //Copia ficheros de disco
        Files.copy(ruta, rutaDestino);
    } catch (IOException e) {
        System.out.println(e.toString());
    }

}
```

✍ Documenta el método que permite copiar ficheros y añade métodos para eliminar fichero y renombrarlos

Path-Ejercicio4

- En los métodos anteriores provoca excepciones:
  - Haciendo que el fichero no existe
  - Haciendo que la ruta no exista

Después documenta las excepciones que se han producido