UA02. PRIMER ACCESO A DATOS

2.1. Excepciones controladas

Son **excepciones controladas** todas aquellas que representan errores fuera del control del programa. Estas excepciones se controlan en tiempo de ejecución.

Usaremos este tipo de excepción siempre que se espere que el programa puede recuperarse después de lanzarse la excepción. Son de uso obligatorio en cualquier gestión de ficheros.

Este tipo de excepciones son subclases de la clase **Exception**:

- **IOexception:** es una clase de error genérico, la podemos usar siempre que queramos controlar todo tipo de excepción sin saber exactamente cuál se podrá lanzar. Subclase de Exception.
- **FileSystemException:** es una clase que lanza errores cuando una operación con ficheros falla en un fichero o dos.
- **DirectoryNotEmptyException:** nos indicará que la carpeta no está vacía.
- **FileNotFoundException:** nos servirá para indicar que no se ha encontrado el fichero.
- AccesDeniedException: esta excepción es útil para controlar si el acceso al fichero está permitido.
- FileAlreadyExistsException: cuando se crea un fichero y ya existe se lanza esta excepción.
- NoSuchFileException: será útil para controlar si existe el fichero al cual queremos acceder.
- NotDirectoryException: para controlar si existe la carpeta a la cual queremos acceder o crear.

Hay dos maneras de controlar las excepciones:

```
1  private static void excepcionControladaConThrows() throws FileNc
2      File ej = new File("fichero.txt");
3      FileInputStream stream = new FileInputStream(ej);
4  }
5
6  private static void metodoConTryCatch() {
7      File ej = new File("ejemplo.txt");
8      try {
9          FileInputStream stream = new FileInputStream(ej);
10     } catch (FileNotFoundException e) {
11         e.printStackTrace();
12     }
13 }
```

Como podemos ver en el ejemplo, hay dos tipos de excepción: la que se pone en la declaración del método con un **throws** o con un **try-catch** dentro de la implementación del método.

Hay que tener en cuenta que, cuando declaramos el **throws** en el método, quien recibe la función deberá envolver esa llamada dentro de un **try-catch**, es decir, quien haga la llamada al método **excepcionControladaConThrows()** deberá realizarla con un try-catch. Algo más o menos así:

```
1  private static void ejemploLlamada() {
2      try {
3          excepcionControladaConThrows();   // Captura el posible
4      } catch (FileNotFoundException e) {
5          e.printStackTrace();  // Si hay throws, se muestra el er
6      }
7  }
```