educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 1 - Java Analyst

Módulo 8

educación IT
CODE YOUR FUTURE

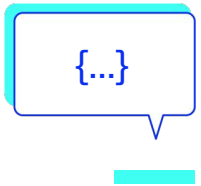
Try with resources

Introducción

Muchas veces, será necesario **cerrar objetos instanciados cuyo uso ha finalizado**. Por ejemplo: los archivos. Al tratar de leer un archivo, el sistema toma el objeto, lo carga en memoria, y no libera el recurso hasta que se lo indiquemos.

Ya vimos que el objeto **`FileReader`** tiene la excepción verificada **`FileNotFoundException`**, pero el método close de la misma clase también posee una excepción verificada **`IOException`** que indica que puede no cerrar el recurso por algún problema (interrupción de la comunicación, etc.).

Siempre vamos a intentar cerrar el recurso en el bloque del **`finally`**, ya que no importa si ocurrió un problema o siguió como se esperaba, se debe liberar el recurso. Pero, como podemos apreciar en la siguiente pantalla, el código se hace un poco engorroso o difícil de interpretar.



```java
FileReader archivo = null;
try {
    archivo = new FileReader("C:/archivo.txt");
    //algoritmo para leer el archivo
} catch (FileNotFoundException e) {
    e.printStackTrace();
}finally {
    try {
        archivo.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

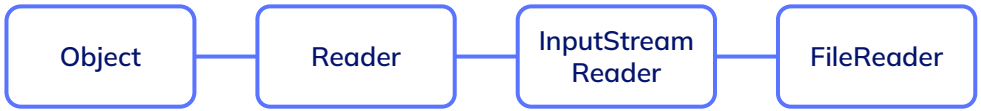
try with resources

Java 7 incorpora la sentencia denominada **`try-with-resources`** con el objetivo de **cerrar los recursos de forma automática en la sentencia `try-catch-finally` y hacer más simple el código.** Para ello, la clase del objeto a trabajar debe tener implementada la interfaz **`AutoCloseable`**. Por medio de ella, Java detecta que posee el método close y lo invoca al terminar de utilizar el recurso.

La mayoría de las clases relacionadas con entrada y salida implementan la interfaz **`AutoCloseable`**.

> **Por ejemplo:** las relacionadas con el sistema de ficheros y flujos de red como **`InputStream`**, también las relacionadas con la conexión de base de datos mediante **JDBC** con **`Connection`**.

```java
try (FileReader archivo = new FileReader("C:/archivo.txt")) {
    // algoritmo para leer el archivo
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```




```
public abstract class Reader implements Readable, Closeable {
```

```
public interface Closeable extends AutoCloseable {
```

```
public interface AutoCloseable {
    void close() throws Exception;
}
```

¡Sigamos trabajando!