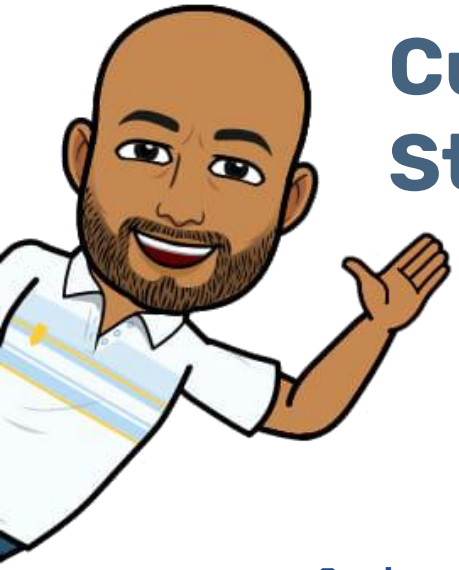


Curso de Java Standard

Ing. Octavio Robleto

octavio.robleto@gmail.com

https://octaviorobleto.com

Introducción

Muchas veces nos vamos a encontrar con objetos que al ser instanciados necesitan ser cerrados al finalizar su uso, uno de esos objetos son los archivos, al tratar de leer los archivos el sistema toma el objeto lo carga en memoria y no libera el recurso hasta que nosotros lo indiquemos.

Ya hemos visto 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 tratar de cerrar el recurso en el bloque del **finally** ya que no importa si ocurrió un problema o siguió como se esperaba, el recurso hay que liberarlo.

Pero como podemos apreciar el código se hace un poco engorroso o difícil de interpretar.

```
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();
    }
}
```

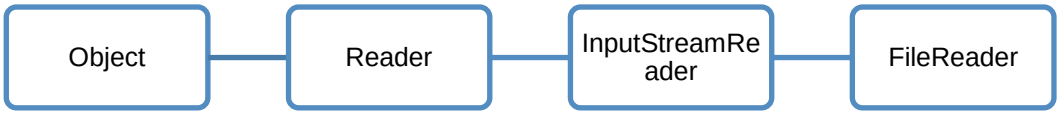
try with resources

Java 7 incorpora la sentencia 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** ya que por medio de ella Java detecta que posee el método **close** y así invocarlo al terminar de utilizar el recurso.

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

```
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;
}
```