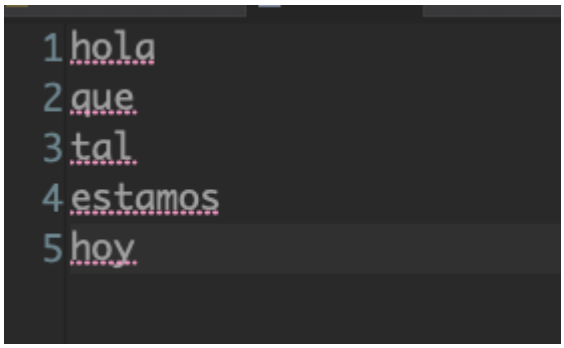
File to String Java 8 es una consulta muy habitual sobre como queremos de forma rápida leer un fichero de texto y convertirlo en una cadena con Java Esto siempre ha sido un concepto muy complejo en todas las versiones del lenguaje ya que Java IO se basa en el concepto de decoradores que aporta una gran flexibilidad a cambio de incrementar la complejidad del código. Veamos como se leía un fichero en Java con una versión muy clásica.


```
package com.arquitecturajava;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class Principal1 {

	public static void main(String[] args) {
		try {
			BufferedReader lector = new BufferedReader(new
FileReader("fichero/hola.txt"));
			StringBuilder cadena = new StringBuilder();
			String line = null;
			while ((line = lector.readLine()) != null) {
```

```
                cadena.append(line);
            }
            lector.close();

            String contenido = cadena.toString();
            System.out.println(contenido);
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
```

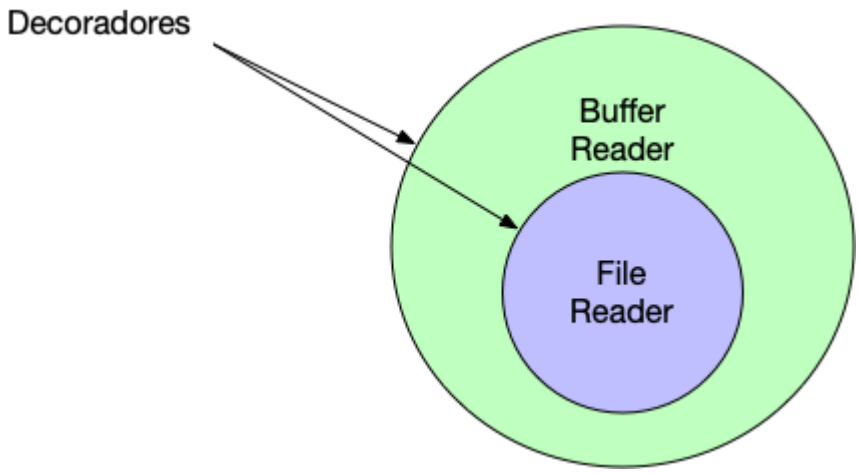
File To String , Java Clásico

Cuando uno empieza a trabajar en Java es difícil entender porque leer un fichero puede ser tan complejo . Todo el mundo espera un código más cercado a una o dos lineas y nos encontramos con 30 lineas de código para conseguir imprimir un fichero en la consola . En este caso tenemos un fichero que contiene:



¿Porque es el API tan complejo? . Bueno básicamente porque usa un concepto de que

denomina decorador y que añade una gran flexibilidad a la hora de leer todo tipo de ficheros en un grupo de lineas reducido , es decir por ejemplo leer un fichero encriptado tendría una longitud similar .



A cambio se paga un precio y es que leer un fichero trivial puede suponer un mayor esfuerzo del esperado.

File To String Java 8

Para solventar este problema las diversas versiones de Java han ido solventándolo pero es Java 8 la que consigue una mayor reducción del código dejándolo en prácticamente nada.

```
package com.arquitecturajava;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class Principal2 {

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

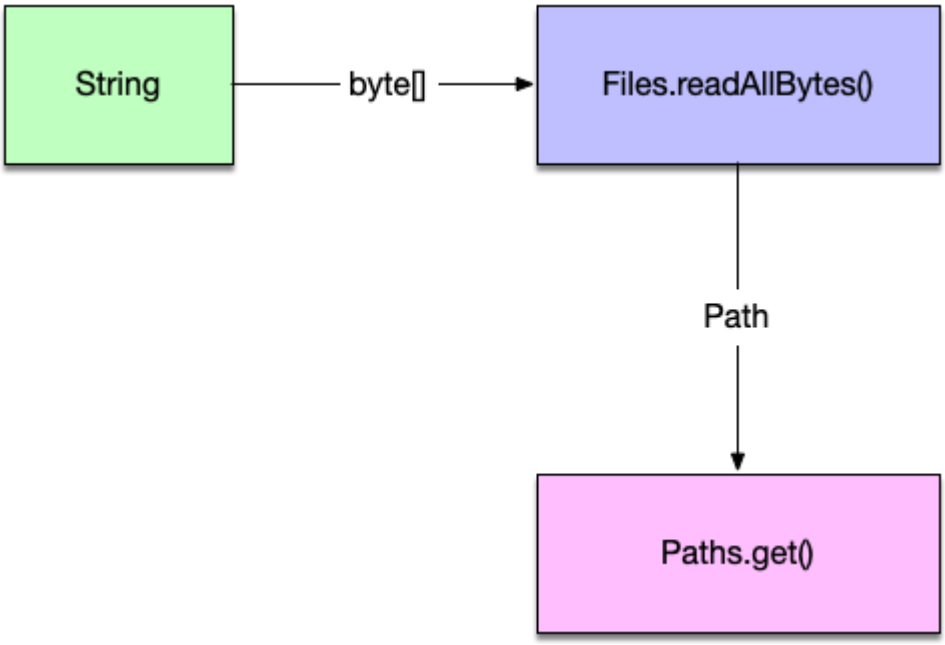
```
                        String contenido = new
String(Files.readAllBytes(Paths.get("fichero/hola.txt")));
                        System.out.println(contenido);
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

}
```

Oferta Cursos 70% Descuento

- Programación Orientada a Objeto
- Java APIs
- Desarrollo Web Java
- Java 8 Stream y Lambdas

Clases y relaciones

En este caso usamos una combinación de dos clases relativamente nuevas una la clase Paths que nos devuelve la ruta a un fichero concreto y otra la clase Files que tiene un método para leer todos los bytes a partir de un Path determinado. De esta forma en muy poco código conseguimos el resultado que buscamos.



Otros artículos relacionados

1. Java Files Walk y recursividad
2. Java Stream File
3. Java 8