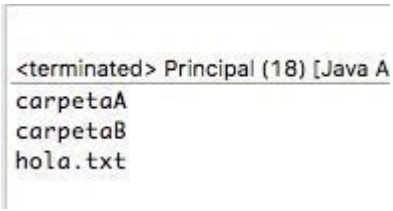
El uso de Java List Directory es muy común . Probablemente será una de las búsquedas de google más habituales en el mundo Java. Recordemos que el api de Java IO esta soportado desde la primera versión de Java por lo tanto el código para realizar esta operación es un clásico. Supongamos que disponemos de una carpeta con los siguiente ficheros y queremos listarlos por la consola.

Para realizar un listado de forma clásica usaremos el siguiente bloque de código:

```
package com.arquitecturajava;

import java.io.File;

public class Principal {

	public Principal() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {

		File micarpeta= new File("micarpeta");

		File[] listaFicheros=micarpeta.listFiles();

		for (int i=0;i&amp;amp;lt;listaFicheros.length;i++) {
```

```
System.out.println(listaFicheros[i].getName());

                }

        }

}
```

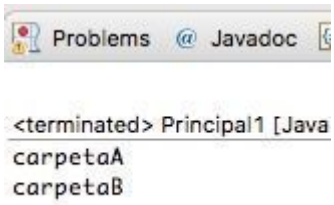
El resultado se imprime en la consola:



De la misma forma podemos hacer algo similar y quedarnos únicamente con el listado de carpetas apoyándonos en el método isDirectory de la clase File.

```
                File micarpeta= new File("micarpeta");
                File[] listaFicheros=micarpeta.listFiles();

                for (int i=0;i&amp;lt;listaFicheros.length;i++) {

                        if (listaFicheros[i].isDirectory()) {

System.out.println(listaFicheros[i].getName());
                        }
                }
```

El resultado solo muestra las carpetas:



Java List Directory Streams

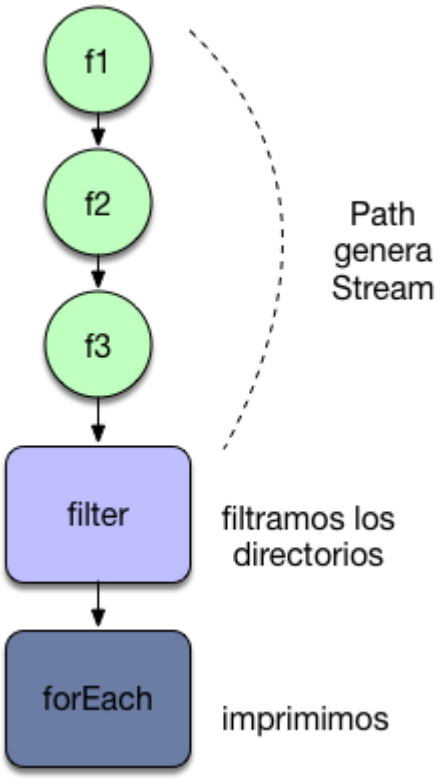
El código es razonable. Ahora bien ¿Se podría hacer de una forma más elegante utilizando Java8? .La verdad es que sí , sin ninguna duda. Vamos a verlo:

```
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) throws IOException {

Files.list(Paths.get("micarpeta")).forEach(System.out::println);
        }
}
```

El concepto de Java List Directory se simplifica claramente y el resultado es idéntico.

```
<terminated> Principal (18) [Java A
carpetaA
carpetaB
hola.txt
```

Acabamos de usar un Stream de Java 8 para recorrer un directorio.De la misma forma podríamos utilizar el método filter y un predicado para filtrar los elementos del directorio y mostrar únicamente las carpeta.



Veamoslo:

```
package com.arquitecturajava;

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

```
import java.nio.file.Paths;

public class Principal2 {

        public Principal2() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws IOException {

Files.list(Paths.get("micarpeta")).filter(Files::isDirectory).forEach(
System.out::println);
                }

}
```

Con estos ejemplos podemos ver como Java 8 simplifica de forma clara alguna de las operaciones más clásicas de Java IO.

Otros artículos relacionados:

1. Java Stream Filter y Predicates
2. Java 8 Lambda Expressions (I)
3. Java Lambda reduce y wrappers
4. Java Stream String y Java 8
5. Java Streams Oracle