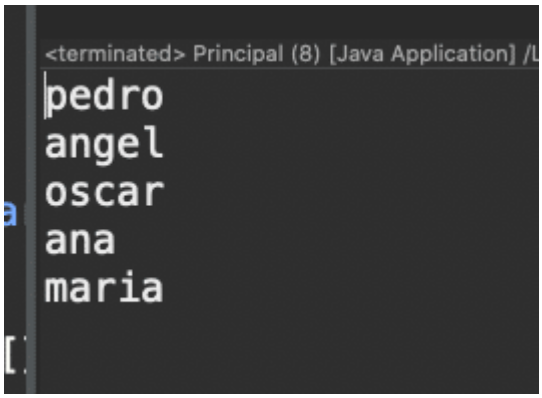
Java Array to Stream ,es una de las operaciones más habituales que nos pueden tocar en el día a día cuando abordamos nuevas capacidades de Java 8 como es el manejo de Streams con código mucho más legacy que puede contener elementos que son simples Arrays de un tipo concreto . ¿Cómo podemos aunar ambas cosas? . La realidad es que la transformación es muy muy sencilla . Vamos a partir de un ejemplo que recorre un array de cadenas.

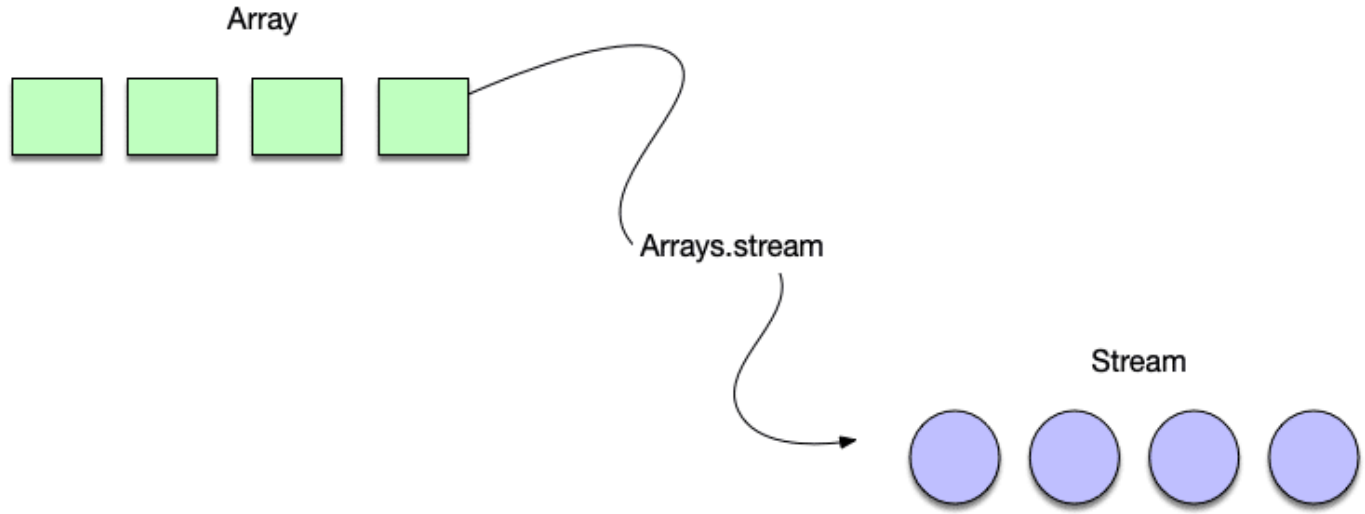
```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		String [] nombres= new String[] { "pedro", "angel",
"oscar","ana","maria"};
		for (String nombre:nombres) {
			System.out.println(nombre);
		}
	}

}
```

Si lo ejecutamos nos mostrará en la pantalla la lista de nombres:



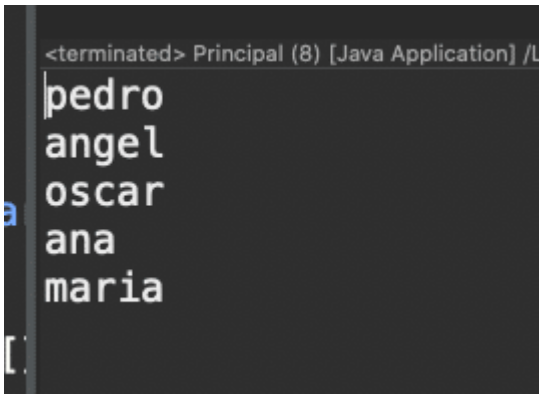
Java Array to Stream

Convertirlo a un Stream y recorrerlos es prácticamente trivial ya que la clase Arrays soporta el método stream que convierte un Array en un Stream de datos y no hay que darle muchas vueltas más:




```
 package com.arquitecturajava;

import java.util.Arrays;

public class Principal {

        public static void main(String[] args) {
                String [] nombres= new String[] { "pedro", "angel",
"oscar","ana","maria"};
                Arrays.stream(nombres).forEach(System.out::println);

        }

}
```

Todo mucho más sencillo y nos permite realizar la conversión de forma prácticamente directa con la simplificación de código que supone al usar un método de referencia el resultado es idéntico:



La ventaja de utilizar un stream es que diseñamos un flujo de trabajo y podemos añadir operaciones intermedias como puede ser filter:

```
package com.arquitecturajava;

import java.util.Arrays;

public class Principal {

	public static void main(String[] args) {
		String [] nombres= new String[] { "pedro", "angel",
"oscar","ana","maria"};
Arrays.stream(nombres).filter(n->n.length()<4).forEach(System.out::pri
ntln);

	}

}
```

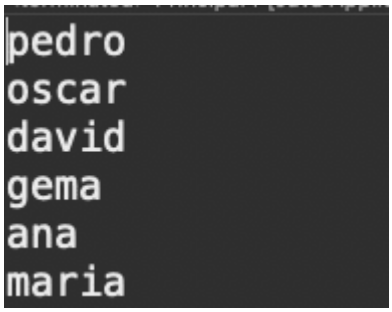
El resultado únicamente imprimirá ana por la consola:

Java Arrays VarArgs y Streams

Otra situación en la que es muy habitual usar Arrays y combinarlos con Streams es cuando usamos métodos que soportan un número variable de argumentos

```
package com.arquitecturajava;

import java.util.Arrays;

public class Principal4 {

	public static void main(String[] args) {
		imprimir ("pedro","oscar","david");
		imprimir ("gema","ana","maria");
	}
	public static void imprimir (String ... nombres) {
		Arrays.stream(nombres).forEach(System.out::println);
	}

}
```

El resultado incluye ambas listas:



Otros artículos relacionados

- Java Stream
- Java Stream Filter
- Java Stream map y estadísticas
- Java Stream Oracle