El uso de Java Optional Stream , poco a poco va aumentando según todos nosotros vamos actualizando nuestras aplicaciones a Java 8 . Recordemos que los tipos optional actuan como wrappers o envolturas que almacenan un tipo concreto. Es decir Optional<Integer> almacena un entero y Optional<String> almacena un posible valor de cadena. Vamos a construir una sencilla lista de Optional de Strings y operar un poquito con ella. La forma más sencilla de trabajar es usar un bucle for.

```
package com.arquitecturajava;

import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class Principal {

	public static void main(String[] args) {

		List<optional<String>> lista = Arrays.asList(
				Optional.of("hola"),
				Optional.empty(),
				Optional.of("que"),
				Optional.of("tal"));

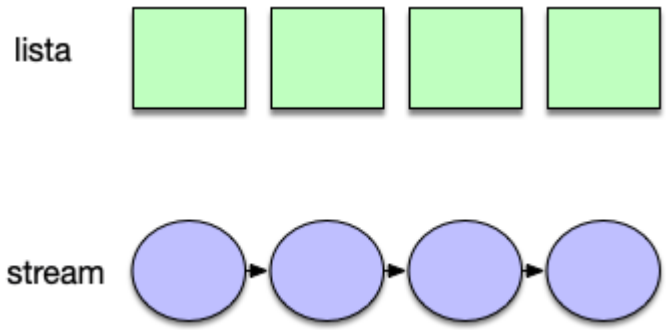
		for(Optional<String> opcion : lista) {
			if(opcion.isPresent()) {
				System.out.println(opcion.get());
			}
		}
	}
```

En esta primera lista tenemos cuatro items uno vacío y tres con contenido por lo tanto si ejecutamos el código y hacemos uso de las ventajas de los tipo optional nos imprimirá 3 elementos por la consola.

hola
que
tal

Java Optional Stream

Acabamos de filtrar el elemento que no tiene contenido. Sin embargo si hacemos un poco caso el código nos daremos cuenta de que es bastante complejo a la hora de trabajar con unos sencillos tipos optionals. ¿Cómo podemos simplificarlo y que sea más elegante? . El primer paso puede ser hacer uso de Streams y comenzar a simplificar las cosas.



Vamos a verlo:

```
package com.arquitecturajava;

import java.util.Arrays;
import java.util.List;
import java.util.Optional;
```

```
public class Principal2 {

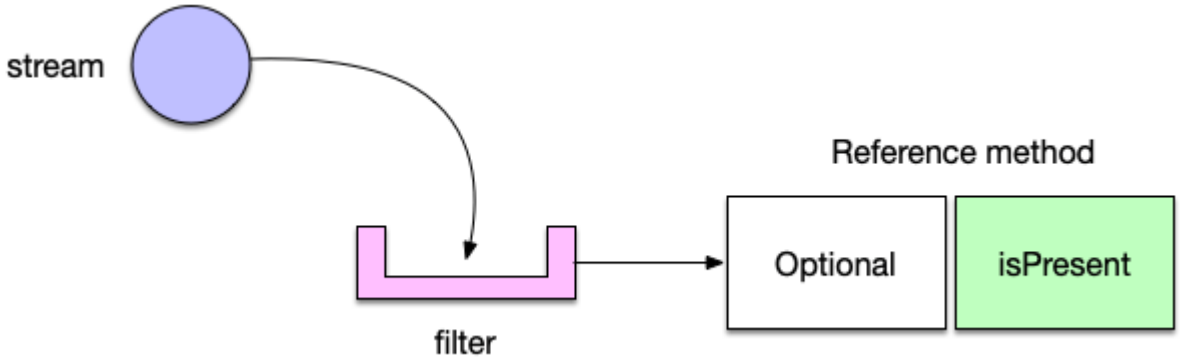
	public static void main(String[] args) {

		List<optional<String>> lista = Arrays.asList(
				Optional.of("hola"),
				Optional.empty(),
				Optional.of("que"),
				Optional.of("tal"));

		lista.stream().forEach((s)-> {
			if (s.isPresent()) {
				System.out.println(s.get());
			}
		});
	}

}
```

Acabamos de convertir una lista en un Stream , usamos el método for each para recorrerla . El resultado es idéntico y la forma de trabajar bastante similar . Ahora bien ya disponemos de un stream y podemos comenzar a usar todos sus métodos. Lo primero que vamos a hacer es usar el método filter y apoyarnos en un method reference, concretamente el método isPresent de la clase optional para ir simplificando la sintaxis.



Veámoslo :

```
package com.arquitecturajava;

import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class Principal3 {

        public static void main(String[] args) {

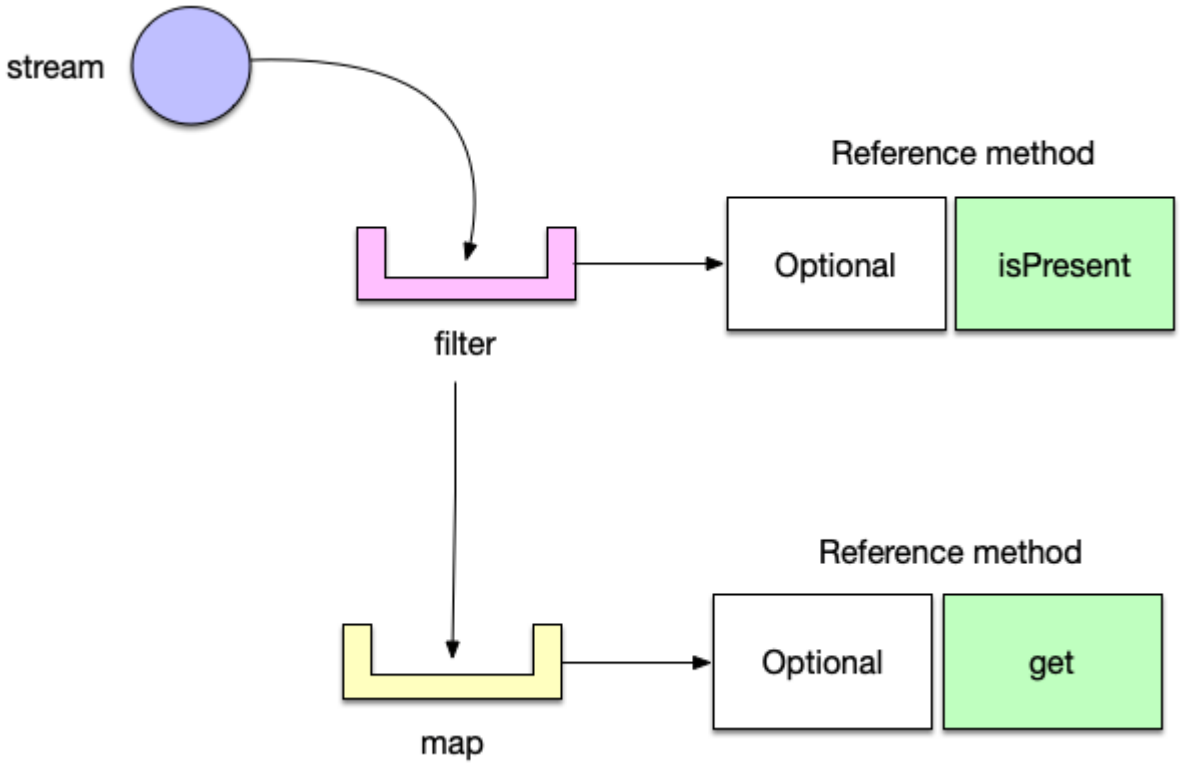
        List<optional<String>> lista = Arrays.asList(
                                Optional.of("hola"),
                                Optional.empty(),
                                Optional.of("que"),
                                Optional.of("tal"));

lista.stream().filter(Optional::isPresent).forEach((s)->{
```

```
                                        System.out.println(s.get());
                });
        }

}
```

Hemos conseguido simplificar un poco la sintaxis. El siguiente paso es usar el método map e invocar al método get de nuestra clase optional



Algo como lo siguiente:

```
package com.arquitecturajava;
```

```
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class Principal4 {

	public static void main(String[] args) {

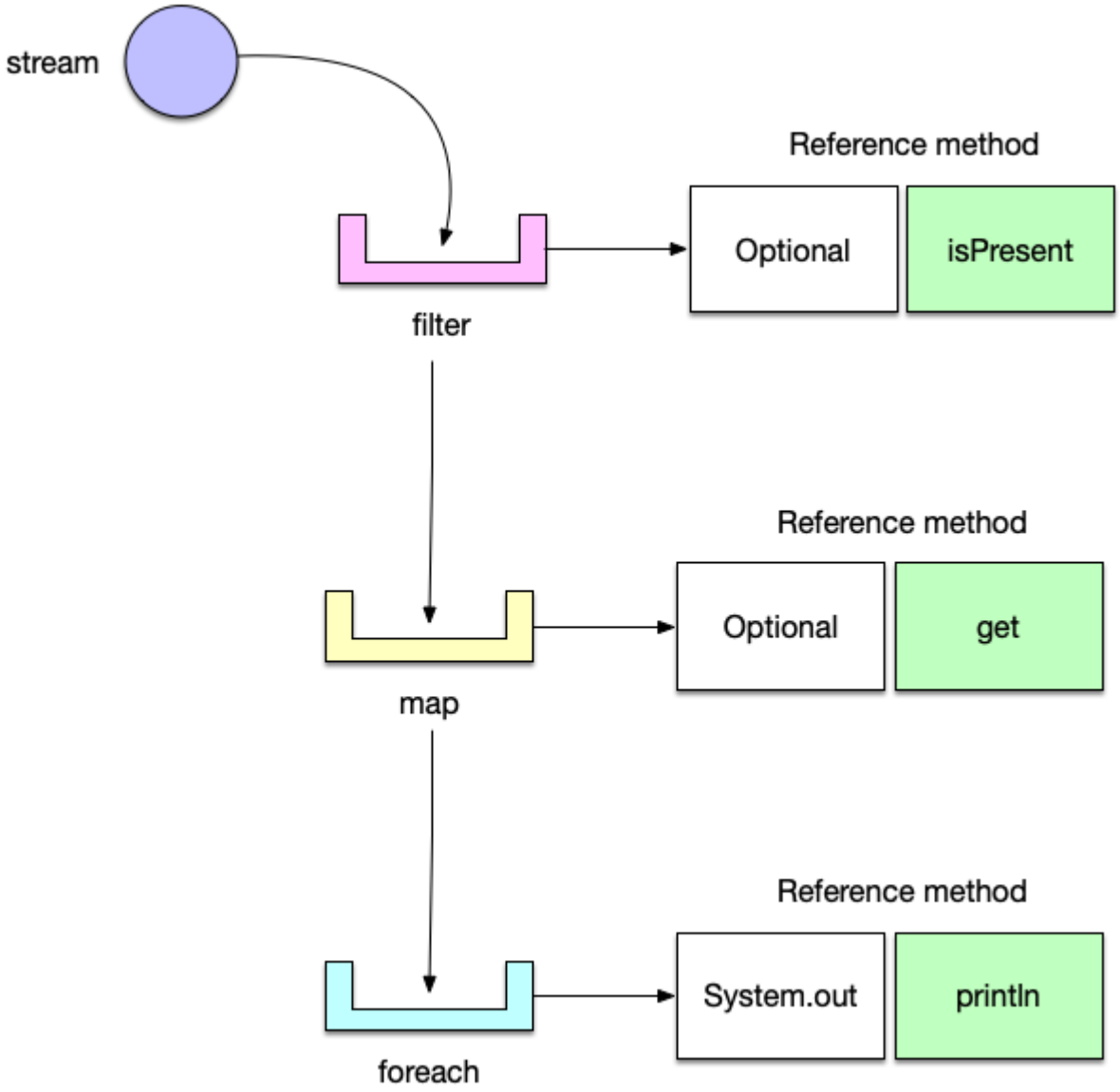
		List<optional<String>> lista = Arrays.asList(
					Optional.of("hola"),
					Optional.empty(),
					Optional.of("que"),
					Optional.of("tal"));
lista.stream().filter(Optional::isPresent).map(Optional::get).forEach(
(s)->{
						System.out.println(s);
		});
	}

}
```

El resultado sigue siendo el mismo:

hola
que
tal

Hemos avanzado bastante con la gestión de Java Optional Stream ahora simplemente nos queda usar el método System.out.println como método de referencia



Así terminamos de compactar la sintaxis.

```
package com.arquitecturajava;
```

```
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class Principal5 {

        public static void main(String[] args) {

                List<optional<String>> lista = Arrays.asList(
                                Optional.of("hola"),
                                Optional.empty(),
                                Optional.of("que"),
                                Optional.of("tal"));

                lista
                .stream()
                .filter(Optional::isPresent)
                .map(Optional::get)
                .forEach(System.out::println);

        }

}
```

La simplificación ha sido fuerte , siendo muy sencillo de entender el código y la combinación de Streams y Optionals

1. Java 8 Lambda Syntax ,simplificando nuestro código
2. Java Optional ifPresent y como utilizarlo
3. Java Stream Collectors y su uso
4. Java Stream String y Java 8

5. JDK Java Optional