El concepto de Java Diamond Operator llega a Java en la versión 7. Se le denomina operador diamante por la forma que tiene el operador “<>” y permite simplificar el manejo de los genéricos. Muchas veces nos olvidamos de usarla ya que estamos acostumbrados la sintaxis básica y es difícil cambiar los hábitos. Vamos a ver un ejemplo sencillo de uso del operador diamante. Supongamos que tenemos el siguiente código en Java.

CURSO SPRING BOOT
GRATIS
APUNTATE!!

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class Principal {

	public static void main(String[] args) {

		List&amp;lt;String&amp;gt; lista= new
ArrayList&amp;lt;String&amp;gt;();
		lista.add("hola");
		lista.add("adios");
		lista.forEach(System.out::println);

	}
```

```
}
```

Todo funciona correctamente , ahora bien no era necesario especificar tanto. Nos es suficiente con definir el tipo de dato en la referencia no hace falta ubicarlo en el constructor.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class Principal {

	public static void main(String[] args) {

		List&amp;lt;String&amp;gt; lista= new
ArrayList&amp;lt;&amp;gt;();
		lista.add("hola");
		lista.add("adios");
		lista.forEach(System.out::println);

	}

}
```

De esta forma conseguimos simplificar el manejo de genericos. En un primer lugar nos parece que la ganancia es poca.

Java Diamond Operator

Sin embargo usar java diamond operator puede simplificar la construcción de tipos genéricos mucho mas complejos como por ejemplo:

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Principal2 {

	public static void main(String[] args) {

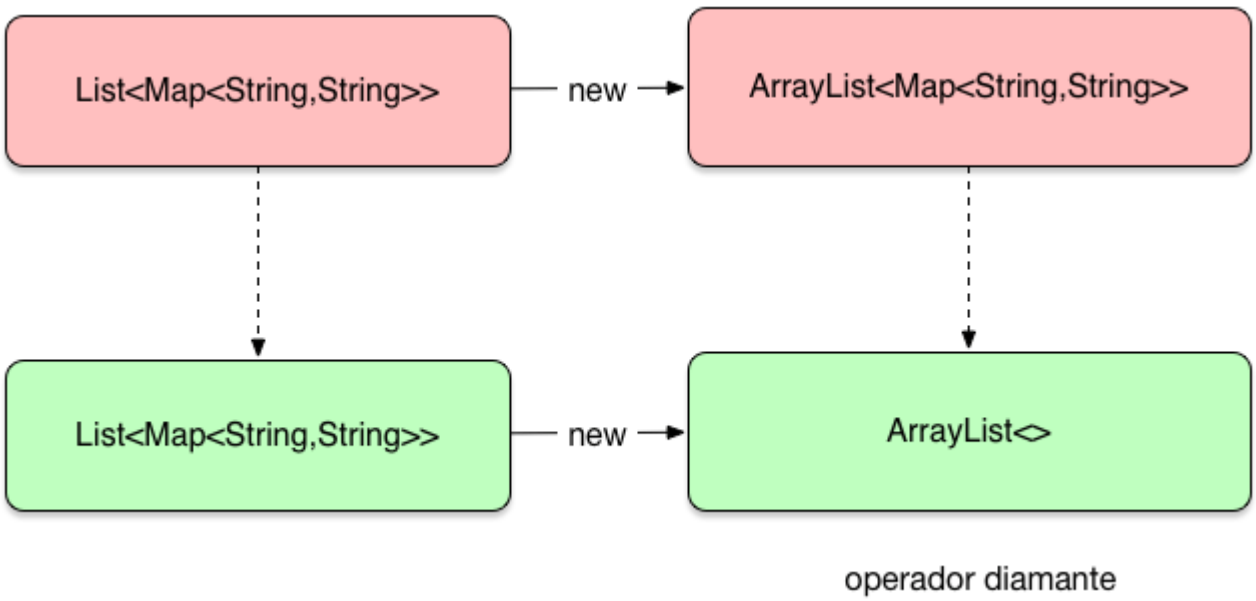
		List&amp;lt;Map&amp;lt;String, String&amp;gt;&amp;gt;
lista= new
ArrayList&amp;lt;Map&amp;lt;String,String&amp;gt;&amp;gt;();
		Map&amp;lt;String,String&amp;gt; mapa= new
HashMap&amp;lt;String,String&amp;gt;();
		mapa.put("clave1", "valor1");
		lista.add(mapa);

		lista.forEach(System.out::println);

	}

}
```

En este caso tenemos una lista que incluye un mapa y todo esta lleno de anotaciones genéricas. En este ejemplo el operador diamante nos puede ayudar bastante.



Veámoslo en código;

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Principal3 {

        public static void main(String[] args) {

                List&amp;lt;Map&amp;lt;String, String&amp;gt;&amp;gt;
lista= new ArrayList&amp;lt;&amp;gt;();
                Map&amp;lt;String,String&amp;gt; mapa= new
```

```
HashMap&amp;lt;&amp;gt;();
                mapa.put("clave1", "valor1");
                lista.add(mapa);

                lista.forEach(System.out::println);

        }

}
```

Acostumbremonos a usar más Java Diamond Operator para clarificar las estructura de genéricos que utilizamos en nuestro código.

CURSO SPRING BOOT
GRATIS
APUNTATE!!

Otros artículos relacionados

1. Utilizando Java Custom Generics
2. Java Generic Methods
3. Java Generic Repository y JPA