Method References

Lambda expressions are used to create anonymous methods. But sometimes, lambda just calls an existing method.

Let us take example of it.

* We will create an anonymous function for filtering null values from List of String.
* If String is NOT NULL then we will convert it to uppercase and print it.

**public** **static** List<String> populateNames(){

String[] nameArr = { "Harry", **null**, "Sansa", "Ned", **null**, "Roose",

"Walder", "Jaqen", **null**, "Joffery", "Robert", **null**,

"Catleyn", "Peter", "Jack", **null**, "George", "Martin",

"Frey", "Brandon" };

List<String> names=**new** ArrayList<String>(Arrays.*asList*(nameArr));

**return** names;

}

**public** **static** **void** toUpperCaseString(List<String> names){

names.stream()

.filter(name -> name!=**null**)

.map(str -> str.toUpperCase())

.forEach(str -> System.***out***.println(str));

}

Now the anonymous function is the Predicate in filter method. This is filter(name -> name!=**null**) anonymous function.

Now we will convert that filtered value to upper case. This is map(str -> str.toUpperCase()) is calling an existing method.

So when we call the String class’s toUpperCase() method we are just calling an existing method. There is no problem with telling “How to call the method” rather than calling method just by name. So we can replace lambda in map(..).

So **map(str -> str.toUpperCase())** becomes **map(String::toUpperCase)** as method reference.

Let us take a different example for Integer sort using method reference.

**public** **static** **void** intsSort(){

**int**[] age = { 16, 22, 22, 25, 24, 16, 23, 19, 19, 27, 20, 20, 12, 13, 18, 15, 20, 26, 29, 22 };

List<Integer> list=**new** ArrayList<Integer>();

**for** (**int** i = 0; i < age.length; i++) {

list.add(age[i]);

}

System.***out***.println("Before Sort "+list);

list.sort(Integer::*compare*);

System.***out***.println("After Sort "+list);

}

Output:

Before Sort [16, 22, 22, 25, 24, 16, 23, 19, 19, 27, 20, 20, 12, 13, 18, 15, 20, 26, 29, 22]

After Sort [12, 13, 15, 16, 16, 18, 19, 19, 20, 20, 20, 22, 22, 22, 23, 24, 25, 26, 27, 29]

Let us now see how to create method references.

|  |  |
| --- | --- |
| Lambda | (args) -> ClassName.staticMethod(args) |
| Method Reference | ClassName::staticMethod |
| Lambda | (arg0, args) ->arg0.instanceMethod(args) |
| Method Reference | ClassName::instanceMethod |
| Lambda | args() -> instance.instanceMethod(args) |
| Method Reference | instance::instanceMethod |

That’s all on Method References. We will look into them again once we start with Stream API of Java 8.

Read about Constructor references, Lambda Operator and 43 different Functional Interfaces.