Constructor References

In previous post we saw Method References. In this post we will see Constructor References.

We can create Constructor Reference using ClassName and new keyword as ClassName::new. This fits for the no-args constructor.

We can use Supplier Functional Interface for creating an object of Person class.

Supplier<Person> p = Person::**new**;

Person person =p.get();

The above piece of code is equivalent to:

Supplier<Person> p = () -> **new** Person();

Person person = p.get();

Let us now try for different examples with different parameters.

Let us see how to use Constructor Reference when we have parameter to constructor.

Function<String, Person> p = Person::**new**;

Person person = p.get("John Doe");

Let see Lambda Operator for same.

Function<String, Person> p = (name) -> **new** Person(name);

Person person = p.apply("John Doe");

So Function's get method will return new instance of Person with name.

Now, let us say that you want to create multiple instance of Person with name as parameter. We can leverage the concept of Function interface.

**public** **class** App {

**public** **static** **void** main(String[] args) {

List<String> names = **new** ArrayList<>(Arrays.*asList*("Harry", "John",

"Kip", "Gunther"));

List<Person> person = *nameMapper*(names, Person::**new**);

System.***out***.println(person);

}

**public** **static** List<Person> nameMapper(List<String> names, Function<String, Person> func) {

List<Person> person = **new** ArrayList<Person>();

**for** (String name : names) {

person.add(func.apply(name));

}

**return** person;

}

}

Now one step ahead. Let’s say you have 2 parameters for Person constructor as Name and Age. Here, we can leverage the concept of BiFunction interface.

BiFunction<Integer, String, Person> p = Person::**new**;

Person person = p.apply(23, "John Doe");

Lambda equivalent would be as below:

BiFunction<Integer, String, Person> p = (age, name) -> **new** Person(age, name);

Person person = p.apply(23, "John Doe");

That’s all on Constructor Reference. Read about [Method references](http://data-structure-learning.blogspot.com/2015/08/method-references.html), Lambda Operator and 43 different Functional Interfaces.