ArrayList class forEach method

In previous posts we saw ArrayList [Introduction](http://data-structure-learning.blogspot.com/2015/08/arraylist-class-introduction-and-how-it.html) and 3 different [constructors](http://data-structure-learning.blogspot.com/2015/08/arraylist-class-constructors.html) of ArrayList class. We also saw overloaded version of [add](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-add-methods.html)(), [addAll](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-addall-methods.html)() method. We also saw [clear](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-clear-method.html)() method which sets all elements of ArrayList to null. We also covered [indexOf](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-indexof-method.html)() method that returns index of object and [contains](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-contains-method.html)() method that returns true of object exists in list, false otherwise.

In this post we will see a method called as forEach(). This is a new method as it is introduced in Java 8. I will explain in short here. But if you want to learn in details then refer following posts. I am listing them in order.

1. Lambda Calculus
2. Imperative and Declarative
3. Behavior Parameterization Part 1
4. Behavior Parameterization Part 2
5. Functional Interfaces
6. Using Functional Interfaces
7. Functional Interface without Arguments
8. Functional Interface with Arguments
9. Iteration Problems prior to Java 8

If you read all the above posts then you will know why is this method introduced and how we can we use it.

As we know ArrayList class implements List interface. List interface extends Collection interface and Collection interface extends Iterable interface. Below diagram will help you visualize the hierarchy.

Now ArrayList class has a method called as forEach which accept [Consumer](http://data-structure-learning.blogspot.com/2015/07/java-lambda-consumer-functional.html) as argument. Consumer is a functional interface in Java 8. It performs operation on an accepted value and returns void.

forEach() method is also defined in Iterable Interface. Yes, we can define a method in interface in Java 8, and they are called default methods.

So, if you write like this List<String> names = **new** ArrayList<String>(); then this will call forEach method in Iterable interface. Refer the diagram above.

And if you write like this ArrayList<String> names = **new** ArrayList<String>(); then this call with call forEach() method in ArrayList class.

The thing about forEach() method is that it handles iteration internally. You do not need to specify loop or iterator to iterate the collection. It is handled internally by forEach() method. It is a great way to eliminate a boiler plate code.

Let us print all the elements in List<String> using forEach() method.

**package** org.example.collections.list.arraylist;

**import** java.util.ArrayList;

**import** java.util.function.Consumer;

**public** **class** ArrayListForEachDemo {

**public** ArrayList<String> arrayListAdd() {

ArrayList<String> names = **new** ArrayList<String>();

names.add("Ned");

names.add("Catelyn");

names.add("Rob");

names.add("Arya");

names.add("Sansa");

**return** names;

}

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

ArrayListForEachDemo forEachDemo = **new** ArrayListForEachDemo();

ArrayList<String> names = forEachDemo.arrayListAdd();

forEachDemo.arrayListForEach(names);

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

forEachDemo.arrayListForEachLambdaOperator(names);

}

**public** **void** arrayListForEach(ArrayList<String> names) {

names.forEach(**new** Consumer<String>() {

@Override

**public** **void** accept(String str) {

System.***out***.print(str + " ");

}

});

}

**public** **void** arrayListForEachLambdaOperator(ArrayList<String> names) {

names.forEach(name -> System.***out***.print(name + " "));

}

}

That’s all on forEach() method. I have written several posts on [Functional Interfaces](http://data-structure-learning.blogspot.com/p/functional-programming-in-java.html), [Lambda Calculus](http://data-structure-learning.blogspot.com/p/functional-programming-in-java.html) and [java.util.function](http://data-structure-learning.blogspot.com/p/javautilfunction-package.html) package.

java.util.function package contains 43 different Functional Interfaces.

In next post we will see the get() method. It returns the element at index.