ArrayList class replaceAll method

This is 18th post in series of ArrayList class. Previously we have seen [ArrayList introduction](http://data-structure-learning.blogspot.com/2015/08/arraylist-class-introduction-and-how-it.html), ArrayList class [constructors](http://data-structure-learning.blogspot.com/2015/08/arraylist-class-constructors.html), [add](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-add-methods.html)() method, [addAll](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-addall-methods.html)() method, [clear](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-clear-method.html)() method, [indexOf](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-indexof-method.html)() method, [contains](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-contains-method.html)() method, forEach() method, get(), isEmpty(), iterator(), lastIndexOf() method, listIterator(), remove(int index), remove(Object o), removeAll(Collection<?> c) and removeIf(Predicate<? super T> E) method.

In this post we will see replaceAll() method. To understand this method you need to know about [lambda operator](http://data-structure-learning.blogspot.com/2015/06/using-functional-interface.html) and [UnaryOperator](http://data-structure-learning.blogspot.com/2015/07/java-lambda-unaryoperator-functional.html) Interface. You also need to know what [Functional Interface](http://data-structure-learning.blogspot.com/2015/06/functional-interfaces-java-8.html) is. I have attached links so it would be useful for you to read about it.

If you are reading this line that I assume that you have gone through above links.

Below is the program that uses UnaryOperator as anonymous inner class and as Lambda Operator.

If the number is even we will replace it by 222 and if it is not then replace it with 111.

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

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.UnaryOperator;

**public** **class** ArrayListReplaceAll {

**public** List<Integer> numbers() {

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

numbers.add(1);

numbers.add(2);

numbers.add(3);

numbers.add(4);

numbers.add(5);

**return** numbers;

}

**public** **void** replaceAllDemoAnonymousInnerClass(List<Integer> numbers) {

numbers.replaceAll(**new** UnaryOperator<Integer>() {

@Override

**public** Integer apply(Integer number) {

**return** number.intValue() % 2 == 0 ? 222 : 111;

}

});

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

}

**public** **void** replaceAllDemoLambdaOperator(List<Integer> numbers) {

numbers.replaceAll(**new** UnaryOperator<Integer>() {

@Override

**public** Integer apply(Integer number) {

**return** number.intValue() % 2 == 0 ? 222 : 111;

}

});

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

}

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

ArrayListReplaceAll replaceAllDemo = **new** ArrayListReplaceAll();

List<Integer> numbers = replaceAllDemo.numbers();

System.***out***.println("UnaryOperator using Anonymous Inner Class");

replaceAllDemo.replaceAllDemoAnonymousInnerClass(numbers);

System.***out***.println("\nUnaryOperator using Lambda Operator");

numbers = replaceAllDemo.numbers();

replaceAllDemo.replaceAllDemoLambdaOperator(numbers);

}

}

Output

UnaryOperator using Anonymous Inner Class

[111, 222, 111, 222, 111]

UnaryOperator using Lambda Operator

[111, 222, 111, 222, 111]

That’s all on replaceAll(UnaryOperator<E> operator) method. In next post we will see retainAll() method. retainAll() method is opposite of removeAll(). retainAll(Collection<?> c) will retain all the elements of this list specified in collection(as parameter).