ArrayList class lastIndexOf method

This is 12th 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](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-foreach-method.html)() method, [get](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-get-method.html)(), [isEmpty](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-isempty-method.html)() and [iterator](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-iterator-method.html)() method.

In this post we will see lastIndexOf() method. We saw [indexOf](http://data-structure-learning.blogspot.com/2015/09/arraylist-class-indexof-method.html)() method and it returns index of the object if it exists and -1 if object does not exists.

lastIndexOf() is similar to indexOf() method as lastIndexOf() method starts to search for object in reverse way. So it returns the highest index of specified element if found else it returns -1.

Below is the code taken from JDK.

**public** **int** lastIndexOf(Object o) {

**if** (o == **null**) {

**for** (**int** i = size-1; i >= 0; i--)

**if** (elementData[i]==**null**)

**return** i;

} **else** {

**for** (**int** i = size-1; i >= 0; i--)

**if** (o.equals(elementData[i]))

**return** i;

}

**return** -1;

}

The **for loop** used is **reverse**. It traverses from end to beginning of List. As soon as it finds the element it returns the index. If it is not able to find it returns -1.

Below is the code that demonstrate the lastIndexOf() method

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

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ArrayListlastIndexOf {

/\*\*

\* This method is used to fill Integer values

\* to List using add() method.

\* \*/

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

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

numbers.add(10);

numbers.add(12);

numbers.add(5);

numbers.add(10);

numbers.add(7);

**return** numbers;

}

/\*\*

\* This method takes List and candidate as input.

\* It searches for the candidate in List and returns

\* the index.

\*

\* **@param** numbers

\* **@param** number

\* \*/

**public** **void** lastIndexofDemo(List<Integer> numbers, **int** number){

**int** index = numbers.lastIndexOf(number);

System.***out***.println("Index for "+number+" is:: "+index);

}

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

ArrayListlastIndexOf lastIndexOf = **new** ArrayListlastIndexOf();

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

//Prints 3

lastIndexOf.lastIndexofDemo(numbers, 10);

//Prints 4

lastIndexOf.lastIndexofDemo(numbers, 7);

//Not Found

//Prints -1

lastIndexOf.lastIndexofDemo(numbers, 17);

}

}

Output

Index for 10 is:: 3

Index for 7 is:: 4

Index for 17 is:: -1

That’s all on lastIndexOf() method. In next post we will see ListIterator() method. It is better version of iterator() because it allows several different methods like add(), hasNext(), hasPrevious(), next(), nextIndex(), previous(), previousIndex(), remove() and set() method.