Collections class fill method

fill(..) method is used to fill the specified object into the list. This method call will replace all the elements in list with the object passed as parameter. This method also throw UnsupportedOperationException too. We will cover that in some time.

Previously we had done several other methods of Collections class such as [addAll(..)](http://data-structure-learning.blogspot.com/2015/05/collections-class-addall-method.html), [swap(..)](http://data-structure-learning.blogspot.com/2015/06/collections-class-swap-method.html), [synchronized](http://data-structure-learning.blogspot.com/2015/06/synchronized-collections-by-collections.html), [binarySearch(..)](http://data-structure-learning.blogspot.com/2015/06/collections-class-binarysearch-method.html), [copy(..)](http://data-structure-learning.blogspot.com/2015/06/collections-class-copy-method.html).

It will take list size to complete this operation so time taken is O(n) where n is size of list or linear time.

We will see the code for this method. Code is take from JavaDoc.

**public** **static** <T> **void** fill(List<? **super** T> list, T obj) {

**int** size = list.size();

**if** (size < ***FILL\_THRESHOLD*** || list **instanceof** RandomAccess) {

**for** (**int** i=0; i<size; i++)

list.set(i, obj);

} **else** {

ListIterator<? **super** T> itr = list.listIterator();

**for** (**int** i=0; i<size; i++) {

itr.next();

itr.set(obj);

}

}

}

List<? **super** T> list – indicates the put principle means that we will set new values. ? super T is wildcard for setting or putting new values into the list.

T obj – is the object to be filled in list.

**if** (size < ***FILL\_THRESHOLD*** || list **instanceof** RandomAccess) {

**for** (**int** i=0; i<size; i++)

list.set(i, obj);

}

If size < FILL\_THRESHOLD or list instanceof RandomAccess then copy them by accessing their index. So for ArrayList, CopyOnWriteArrayList, Stack or Vector then if condition is used as they support RandomAccess.

**else** {

ListIterator<? **super** T> itr = list.listIterator();

**for** (**int** i=0; i<size; i++) {

itr.next();

itr.set(obj);

}

}

If the list does not support RandomAccess then else block is executed. ListIterator<E> is used to set the previously returned value. set(E e) is used to set the previously returned value. We get the returned value by using next() method of ListIterator<E>.

fill(..) can also throw an exception UnsupportedOperationException if the list or ListIterator<E> does not support set(E e) operation.

Below code works fine (without exception)

**public** **static** **void** fillDemo() {

List<String> list = *populateLanguages*();

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

Collections.*fill*(list, "Java");

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

}

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

List<String> languages = **new** ArrayList<String>();

languages.add("Java");

languages.add("JavaScript");

languages.add("C#");

languages.add("Python");

**return** languages;

}

Output:

Before fill operation [Java, JavaScript, C#, Python]

After fill operation [Java, Java, Java, Java]

Below is the example that throws UnsupportedOperationException

**public** **static** **void** fillDemo() {

List<String> list = *populateLanguages*();

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

list=Collections.*unmodifiableList*(list);

Collections.*fill*(list, "Java");

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

}

Output:

Before fill operation [Java, JavaScript, C#, Python]

Exception in thread "main" java.lang.UnsupportedOperationException

at java.util.Collections$UnmodifiableList.set(Unknown Source)

at java.util.Collections.fill(Unknown Source)

at org.collections.CollectionClass.fillDemo(CollectionClass.java:84)

at org.collections.CollectionClass.collectionsAddAllDemo(CollectionClass.java:30)

at org.collections.CollectionClass.main(CollectionClass.java:12)

That’s all on fill(..) method of Collections class.