Synchronized Collections by Collections class

HashSet, HashMap, TreeSet, TreeMap, ArrayList, LinkedList and several other classes are not synchronized. Means they cannot be used in multi-threading environment. So what we can do is either use concurrent collections that are defined in java.util.concurrent package or use the synchronized collections returned by Collections class.

Do not confuse Collection interface with Collections class. Watch for character ‘s’. Learn difference between [Collection and Collections](http://data-structure-learning.blogspot.com/2015/05/difference-between-collection-and.html).

In this post we will see how to create the Synchronized collections through Collections class. There are several different methods which work for different Collection<E>.

|  |  |
| --- | --- |
| **Method (All methods are static)** | **Usage** |
| synchronizedCollection(Collection<T> c) | Returns the synchronized Collection backed by specified Collection. |
| synchronizedSet(Set<T> s) | Returns the synchronized Set backed by specified Set. |
| synchronizedSortedSet(SortedSet<T> s) | Returns the synchronized SortedSet backed by specified SortedSet. |
| synchronizedNavigableSet(NavigableSet<T> s) | Returns the synchronized NavigableSet backed by specified NavigableSet. |
| synchronizedList(List<T> list) | Returns the synchronized List backed by specified List. |
| synchronizedMap(Map<K,V> m) | Returns the synchronized Map backed by specified Map. |
| synchronizedSortedMap(SortedMap<K,V> m) | Returns the synchronized SortedMap backed by specified SortedMap. |
| synchronizedNavigableMap(NavigableMap<K,V> m) | Returns the synchronized NavigableMap backed by specified NavigableMap. |

There is one catch in using all this synchronized collections. The catch is iterator returned by this synchronized collections must be synchronized explicity.

Look at the below example.

**package** org.collections;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.List;

**public** **class** CollectionsSync {

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

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

list.add("Monday");

list.add("Tuesday");

list.add("Wednesday");

list.add("Thursday");

list.add("Friday");

list.add("Saturday");

list.add("Sunday");

List<String> syncList = Collections.*synchronizedList*(list);

/\*\*

\* It is the user's task to manually synchronize the iterator.

\* \*/

**synchronized** (syncList) {

Iterator<String> iter = syncList.iterator();

**while** (iter.hasNext()) {

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

}

}

}

}

Remember it is very important that the iterator returned must be synchronized explicitly. Else there might be non-deterministic behavior in iteration.

We will see different methods of Collections class in next post.