Most of the collections discussed up until now, such as ArrayList, LinkedList, HashSet, HashMap, etc., are not thread-safe. If two parallel threads modify any of these collections parallelly, the user can get stale data or `ConcurrentModificationException`.

We can use thread-safe alternatives such as CopyOnWriteArrayList, ConcurrentHashMap, etc., but what if we don't want to use these alternatives? What if we have already created an ArrayList, and now we want to make it thread-safe.

The **Collections** class provides us with the following methods that can be used to make our existing collection thread-safe.

1. `synchronizedCollection(Collection<T> c)`
2. `synchronizedList(List<T> list)`
3. `synchronizedMap(Map<K,V> m)`
4. `synchronizedSet(Set<T> s)`
5. `synchronizedSortedMap(SortedMap<K,V> m)`
6. `synchronizedSortedSet(SortedSet<T> s)`

Making an ArrayList thread-safe

To make an ArrayList thread-safe we can use the `synchronizedList()` method. Let's see how this method works internally. The **Collections** class contains a static inner class called **SynchronizedList**. The `synchronizedList()` method is called when the object of this class is returned. If you look at the implementation of this class below, then you can see that all the methods have been synchronized.

Since all the methods are synchronized, this makes it very slow. So, we should always try to use the thread-safe implementations instead of making a collection thread-safe using this method.

```java
    static class SynchronizedList<E>
        extends SynchronizedCollection<E>
        implements List<E> {
        private static final long serialVersionUID = -7754090372962971524L;

        final List<E> list;

        SynchronizedList(List<E> list) {
            super(list);
            this.list = list;
        }
        SynchronizedList(List<E> list, Object mutex) {
            super(list, mutex);
            this.list = list;
        }

        public boolean equals(Object o) {
            if (this == o)
                return true;
            synchronized (mutex) {return list.equals(o);}
        }
        public int hashCode() {
            synchronized (mutex) {return list.hashCode();}
        }

        public E get(int index) {
            synchronized (mutex) {return list.get(index);}
        }
```