Let's say we have created a collection where we have added some important data. We want others to read this data, but they should not be allowed to modify the data in this Collection. The **Collections** class provides certain methods that can be used to make our Collection unmodifiable. These methods return a collection in which if someone tries to add or remove an element, then `UnsupportedOperationException` is thrown.

This feature is particularly useful if our Collection contains some sensitive data. We need to only give read access to our data, but we don't want others to accidentally modify it.

Following is the list of methods available to make Collections unmodifiable:

1. `unmodifiableList(List<? extends T> list)`
2. `unmodifiableSet(Set<? extends T> s)`
3. `unmodifiableMap(Map<? extends K, ? extends V> m)`
4. `unmodifiableCollection(Collection<? extends T> c)`
5. `unmodifiableSortedMap(SortedMap<K,? extends V> m)`
6. `unmodifiableSortedSet(SortedSet<T> s)`

We will not look at examples of each of these methods, as they are essentially the same. We will only look at `unmodifiableList()`.

Making ArrayList unmodifiable

Any List implementation such as an ArrayList or LinkedList can be made unmodifiable by using the `unmodifiableList(List list)` method of the Collections class. If we try to add or remove elements from the returned list, then `UnsupportedOperationException` will be thrown as shown in the below example.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class UnmodifiableArrayList {

    public static void main(String args[]) {
        List<String> list = new ArrayList<String>();
        list.add("India");
        list.add("USA");
        list.add("Russia");

        List<String> unmodifiableList = Collections.unmodifiableList(list);
        // This will throw exception because element can't be added to unmodifiable list.
        unmodifiableList.add("China");
    }
}

```

Run Save Reset

Let's discuss briefly how the `unmodifiableList()` method works. Basically, the Collections class has a static inner class called **UnmodifiableList**. When we call the `unmodifiableList()` method, a new instance of **UnmodifiableList** is returned. This class implements the List interface and overrides the operations like add and remove to throw `UnsupportedOperationException`.

Some snippets of the actual code are shown below for your understanding.

```
    static class UnmodifiableList<E> extends UnmodifiableCollection<E>
                                     implements List<E> {
        private static final long serialVersionUID = -283967356065247728L;

        final List<? extends E> list;

        UnmodifiableList(List<? extends E> list) {
            super(list);
            this.list = list;
        }

        public boolean equals(Object o) {return o == this || list.equals(o);}
        public int hashCode()           {return list.hashCode();}

        public E get(int index) {return list.get(index);}
        public E set(int index, E element) {
            throw new UnsupportedOperationException();
        }
        public void add(int index, E element) {
            throw new UnsupportedOperationException();
        }
        public E remove(int index) {
            throw new UnsupportedOperationException();
        }
        public int indexOf(Object o)            {return list.indexOf(o);}
        public int lastIndexOf(Object o)        {return list.lastIndexOf(o);}
        public boolean addAll(int index, Collection<? extends E> c) {
            throw new UnsupportedOperationException();
```