**Synchronized Collection**

Synchronized method is used **to lock an object for any shared resource**. When a thread invokes a synchronized method, it automatically acquires the lock for that object and releases it when the thread completes its task.

The **synchronizedCollection()** method of Java Collections class is used to get a synchronized (thread-safe) collection backed by the specified collection.

Syntax

Following is the declaration of **synchronizedCollection()** method:

1. **public** **static** <T> Collection<T> synchronizedCollection(Collection<T> c)

Parameter

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Required/Optional** |
| c | It is the collection to be "wrapped" in a synchronized collection. | Required |

Returns

The **synchronizedCollection()** method returns a synchronized view of the specified collection.

Example 1

**import** java.util.\*;

**public** **class** CollectionsSynchronizedCollectionExample1 {

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

        //Create vector object

          Vector<String> vector = **new** Vector<String>();

          //Add values in the vector

          vector.add("A");

          vector.add("B");

          vector.add("C");

          vector.add("D");

          vector.add("E");

          //Create a synchronized view

          Collection<String> coll = Collections.synchronizedCollection(vector);

          System.out.println("Synchronized view is :"+coll);

          }

}

**Output:**

Synchronized view is :[A, B, C, D, E]

Example 2

**import** java.util.\*;

**public** **class** CollectionsSynchronizedCollectionExample2 {

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

        //Create a list with items

          List<Integer> list = Arrays.asList(44, 55, 99, 77, 88, 66);

          //Create a synchronized view

          Collection<Integer> coll = Collections.synchronizedCollection(list);

          System.out.println("Synchronized view is :"+coll);

          }

}

**Output:**

Synchronized view is :[44, 55, 99, 77, 88, 66]

Example 3

**import** java.util.\*;

**public** **class** CollectionsSynchronizedCollectionExample3 {

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

        Collection<String> obj = **new** ArrayList<>();

          obj.add("KL"); // not thread safe

          Collection<String> synObj = Collections.synchronizedCollection(obj);

          synObj.add("University"); // thread safe

          //Below iteration will result in non-deterministic behavior

          Iterator<String> ite1 = synObj.iterator();

**while** (ite1.hasNext()) {

              String s = ite1.next();

              System.out.println(s);

          }

          //Below code is the right way to iterate synchronized collection

**synchronized**(synObj) {

              Iterator<String> ite2 = synObj.iterator();

**while** (ite2.hasNext()) {

                  String s = ite2.next();

                  System.out.println(s);

              }

          }

     }

}

**Output:**

**KL**

**University**

**KL**

**University**