Concurrent collections

- Collections.synchronized*
- Synchronized vs Concurrent
- Concurrent collections
- Concurrent queues
- Concurrent maps
- Concurrent sets?

Collections.synchronized*

public static <T> Collection<T> synchronizedCollection(Collection<T> c);

public static <T> Set<T> synchronizedSet(Set<T> s);

public static <T> List<T> synchronizedList(List<T> list);

public static <K,V> Map<K,V> synchronizedMap(Map<K,V> m);

public static <T> SortedSet<T> synchronizedSortedSet(SortedSet<T> s);

public static <K,V> SortedMap<K,V> synchronizedSortedMap(SortedMap<K,V> m);

Each of these methods returns a synchronized (thread-safe) Collection backed up by the specified collection.

Synchronized vs Concurrent

Synchronize

The resource which is synchronized can't be modified by multiple threads simultaneously.

Concurrent

Allows multiple threads to access different parts of a collection at a given time.

Concurrent collections

CopyOnWriteArrayList

A thread-safe variant of java.util.ArrayList in which all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array.

CopyOnWriteArraySet

Set wraper for CopyOnWriteArrayList

ConcurrentSkipListSet

A sorted container that can be accessed by multiple threads. This is essentially* the equivalent of TreeSet for concurrent code.

*Skip list is a data structure that allows fast search within an ordered sequence of elements. Fast search is made possible by maintaining a linked hierarchy of subsequences, with each successive subsequence skipping over fewer elements than the previous one

Concurrent Queues

ConcurrentLinkedQueue

An unbounded thread-safe queue based on linked nodes.

Like most other concurrent collection implementations, this class does not permit the use of null elements.

ArrayBlockingQueue

A classic bounded buffer, in which a fixed-sized array holds elements inserted by producers and extracted by consumers.

Once created, the capacity cannot be changed.

LinkedBlockingQueue

An optionally-bounded blocking queue based on linked nodes.

Linked queues typically have higher throughput than array-based queues but less predictable performance in most concurrent applications.

Concurrent maps

ConcurrentHashMap

A hash table supporting full concurrency of retrievals and high expected concurrency for updates.

ConcurrentSkipListMap

A sorted container that can be accessed by multiple threads. This is essentially the equivalent of TreeMap for concurrent code.

Concurrent set?

There is no concurrent set in the library.

If you need it:

- Collections.newSetFromMap(map)
- ConcurrentHashMap.newKeySet()