23.10 Concurrent Collections

In Chapter 16, we introduced various collections from the Java Collections API. We also mentioned that you can obtain *synchronized* versions of those collections to allow only one thread at a time to access a collection that might be shared among several threads. The collections from the `java.util.concurrent` package are specifically designed and optimized for sharing collections among multiple threads.

8

Figure 23.22 lists the many concurrent collections in package `java.util.concurrent.` The entries for `ConcurrentHashMap` and `LinkedBlockingQueue` are shown in **bold** because these are by far the most frequently used concurrent collections. Like the collections introduced in Chapter 16, the concurrent collections were enhanced in Java SE 8 to support lambdas. However, rather than providing methods to support streams, the concurrent collections provide their own implementations of various stream-like operations— e.g., `ConcurrentHashMap` has methods `forEach`, `reduce` and `search`—that are designed and optimized for concurrent collections that are shared among threads. For more information on the concurrent collections, visit

Collection	Description
ArrayBlockingQueue	A fixed-size queue that supports the producer/consumer relationship—possibly with many producers and consumers.
ConcurrentHashMap	**A hash-based map (similar to the HashMap introduced in Chapter 16) that allows an arbitrary number of reader threads and a limited number of writer threads. This and the LinkedBlockingQueue are by far the most frequently used concurrent collections.**
ConcurrentLinkedDeque	A concurrent linked-list implementation of a double-ended queue.
ConcurrentLinkedQueue	A concurrent linked-list implementation of a queue that can grow dynamically.
ConcurrentSkipListMap	A concurrent map that is sorted by its keys.
ConcurrentSkipListSet	A sorted concurrent set.
CopyOnWriteArrayList	A thread-safe ArrayList. Each operation that modifies the collection first creates a new copy of the contents. Used when the collection is traversed much more frequently than the collection's contents are modified.
CopyOnWriteArraySet	A set that's implemented using CopyOnWriteArrayList.
DelayQueue	A variable-size queue containing Delayed objects. An object can be removed only after its delay has expired.
LinkedBlockingDeque	A double-ended blocking queue implemented as a linked list that can optionally be fixed in size.

LinkedBlockingQueue	**A blocking queue implemented as a linked list that can optionally be fixed in size. This and the ConcurrentHashMap are by far the most frequently used concurrent collections.**
LinkedTransferQueue	A linked-list implementation of interface TransferQueue. Each producer has the option of waiting for a consumer to take an element being inserted (via method transfer) or simply placing the element into the queue (via method put). Also provides overloaded method tryTransfer to immediately transfer an element to a waiting consumer or to do so within a specified timeout period. If the transfer cannot be completed, the element is not placed in the queue. Typically used in applications that pass messages between threads.
PriorityBlockingQueue	A variable-length priority-based blocking queue (like a PriorityQueue).
SynchronousQueue	[For experts.] A blocking queue implementation that does not have an internal capacity. Each insert operation by one thread must wait for a remove operation from another thread and vice versa.

Fig. 23.22

Concurrent collections summary (package java.util.concurrent).