# Top 5 Concurrent Collections from JDK 5 and 6 Java Programmer Should Know

Several new Collection classes are added in Java 5 and Java 6 specially concurrent alternatives of standard synchronized ArrayList, Hashtable and synchronized HashMap collection classes. Many Java programmer still not familiar with these new collection classes from `java.util.concurrent` package and misses a whole new set of functionality which can be utilized to build more scalable and high performance Java application. In this Java tutorial we will some of useful collection classes e.g. ConcurrentHashMap, BlockingQueue which provides some of the very useful functionalities to build concurrent Java application. By the way this is not a comprehensive article explaining each feature of all these concurrent collections, Instead I will just try to list out why they are there, which Collection class they replace or provides alternative for. Idea is to keep it short and simple while highlighting key points of those useful `java.util.concurrent` collections.

**1. ConcurrentHashMap**

ConcurrentHashMap is undoubtedly most popular collection class introduced in Java 5 and most of us are already using it. `ConcurrentHashMap` provides a concurrent alternative of Hashtable or Synchronized Map classes with aim to support higher level of concurrency by implementing fined grained locking. Multiple reader can access the Map concurrently while a portion of Map gets locked for write operation depends upon concurrency level of Map. `ConcurrentHashMap` provides better scalability than there synchronized counter part. Iterator of `ConcurrentHashMap` are fail-safe iterators which doesn't throw `ConcurrencModificationException` thus eliminates another requirement of locking during iteration which result in further scalability and performance.

**2. CopyOnWriteArrayList and CopyOnWriteArraySet**

`CopyOnWriteArrayList` is a concurrent alternative of synchronized List. `CopyOnWriteArrayList` provides better concurrency than synchronized List by allowing multiple concurrent reader and replacing the whole list on write operation. Yes, write operation is costly on `CopyOnWriteArrayList` but it performs better when there are multiple reader and requirement of iteration is more than writing. Since `CopyOnWriteArrayList` Iterator also don't throw `ConcurrencModificationException` it eliminates need to lock the collection during iteration. Remember both `ConcurrentHashMap` and `CopyOnWriteArrayList` doesn't provides same level of locking as Synchronized Collection and achieves thread-safety by there locking and mutability strategy. So they perform better if requirements suits there nature. Similarly, `CopyOnWriteArraySet` is a concurrent replacement to Synchronized Set. See What is CopyOnWriteArrayList in Java for more details

**3. BlockingQueue**

BlockingQueue is also one of better known collection class in Java 5. `BlockingQueue` makes it easy to implement producer-consumer design pattern by providing inbuilt blocking support for `put()` and `take()` method. `put()` method will block if Queue is full while `take()` method will block if Queue is empty. Java 5 API provides two concrete implementation of `BlockingQueue` in form of ArrayBlockingQueue and LinkedBlockingQueue, both of them implement FIFO ordering of element. `ArrayBlockingQueue` is backed by Array and its bounded in nature while `LinkedBlockingQueue` is optionally bounded. Consider using `BlockingQueue` to solve producer Consumer problem in Java instead of writing your won wait-notify code. Java 5 also provides `PriorityBlockingQueue`, another implementation of `BlockingQueue` which is ordered on priority and useful if you want to process elements on order other than FIFO.

Interview Questions
core java interview question (161)
data structure and algorithm (45)
Coding Interview Question (32)
SQL Interview Questions (24)
thread interview questions (20)
database interview questions (18)
servlet interview questions (17)
collections interview questions (15)
spring interview questions (9)
Programming interview question (4)
hibernate interview questions (4)

Translate this blog
Select Language
Powered by Google Translate

**4. Deque and BlockingDeque**

Deque interface is added in Java 6 and it extends `Queue` interface to support insertion and removal from both end of `Queue` referred as `head` and `tail`. Java6 also provides concurrent implementation of `Deque` like `ArrayDeque` and `LinkedBlockingDeque`. Deque Can be used efficiently to increase parallelism in program by allowing set of worker thread to help each other by taking some of work load from other thread by utilizing `Deque` double end consumption property. So if all Thread has there own set of task `Queue` and they are consuming from head; helper thread can also share some work load via consumption from tail.

**5. ConcurrentSkipListMap and ConcurrentSkipListSet**

Just like ConcurrentHashMap provides a concurrent alternative of synchronized HashMap. `ConcurrentSkipListMap` and `ConcurrentSkipListSet` provide concurrent alternative for synchronized version of `SortedMap` and `SortedSet`. For example instead of using TreeMap or TreeSet wrapped inside synchronized Collection, You can consider using ConcurrentSkipListMap or ConcurrentSkipListSet from java.util.concurrent package. They also implement `NavigableMap` and `NavigableSet` to add additional navigation method we have seen in our post How to use NavigableMap in Java.

That's all on this list of concurrent Collection classes from Java 5 and 6. They are added on `java.util.concurrent` package as concurrent alternative of there synchronized counterpart. It's good idea to learn these Collection classes along with other popular classes from Java Collection Framework.

Related **Java Collection Tutorials** from Javarevisited Blog

How to sort ArrayList in ascending and descending order in Java

Top 10 Java Collection Interview Questions – Answered

Difference between ArrayList and Vector in Java

Difference between LinkedList and ArrayList in Java

Difference between TreeSet, HashSet and LinkedHashSet in Java