**Difference between fail-fast Iterator vs fail-safe Iterator in Java**

fail-fast Iterators in Java

As name suggest **fail-fast Iterators** fail as soon as they realized that *structure of Collection has been changed since iteration has begun*. Structural changes means adding, removing or updating any element from collection while one thread is Iterating over that collection. fail-fast behavior is implemented by keeping a modification count and if iteration thread realizes the change in modification count it throwsConcurrentModificationException.

Java doc says this is not a guaranteed behavior instead its done of "best effort basis", So application programming can not  rely on this behavior. Also since multiple threads are involved while updating and checking modification count and this check  is done without synchronization, there is a chance that Iteration thread still sees a stale value and might not be able to detect any change done by parallel threads. Iterators returned by most of JDK1.4 collection are fail-fast including Vector, ArrayList, HashSet etc

fail-safe Iterator in java

Contrary to fail-fast Iterator, **fail-safe iterator** doesn't throw any Exception if Collection is modified structurally

while one thread is Iterating over it because they work on clone of Collection instead of original collection and that’s why they are called as fail-safe iterator. Iterator of CopyOnWriteArrayList is an example of fail-safe Iterator also iterator written by ConcurrentHashMap keySet is also fail-safe iterator and never throw ConcurrentModificationException in Java.