**Fail Fast Vs Fail Safe Iterators in Java**

"Fail fast" means: it may fail ... and the failure condition is checked aggressively so that the failure condition is detected before damage can be done. The fail-fast iterators are typically implemented using a counter (or something) on the list object.

When the list is updated, the counter is incremented. When an Iterator is created, the current value of the counter is embedded in the Iterator object. When an Iterator operation is performed, the method compares the two counter values and throws a CME if they are different.

The only difference is fail-safe iterator doesn't throw any Exception, contrary to fail-fast Iterator, if Collection is modified structurally while one thread is iterating over it. This is 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.

**fail-fast Iterators in Java**

Difference between fail-safe vs fail-fast iterator in javaAs 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 throws ConcurrentModificationException.

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. to read more about Iterator see my post What is Iterator in Java.

**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.

Fail Safe : In this case the iterator will make a copy of the internal data structure and iterates over the copied data structure. Thus any modifications done to the internal data structure will not effect the iterator. In this case we dont find ConcurrentModificationException. But in this approach we can find the following two issues:

a. The overhead of copying the data structure (time and memory)

b. The fail safe iterator doesnt guarantees the data being read is the data currently in the data structure with the created collection.