Before we discuss in detail about fail safe iterator and fail fast iterator in addition to  their comparison , we should understand the term *Concurrent Modification* .

**What is Concurrent Modification ?**  
  
When one or more thread is iterating over the collection, in between, one thread changes the structure of the collection (either adding the element to the collection or by deleting the element in the collection or by updating the value at particular position in the collection) is known as Concurrent Modification

**Difference between Fail Fast iterator and Fail Safe iterator**  
  
**Fail fast Iterator**

Fail fast iterator while iterating through the collection , instantly throws Concurrent Modification Exception if there is structural modification  of the collection . Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Fail-fast iterator can throw ConcurrentModificationException in two scenarios :

1. *Single Threaded Environment*  
      
   After the creation of the iterator , structure is modified at any time by any method other than iterator's own remove method.
2. *Multiple Threaded Environment*   
     
    If one thread is modifying the structure of the collection while other thread is iterating over it .

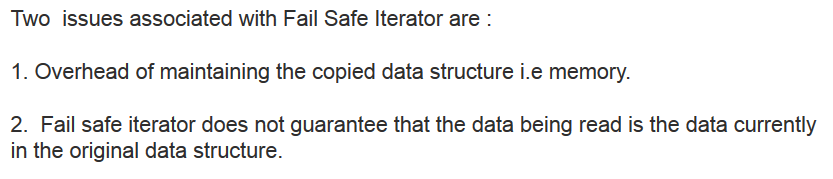
According to  [Oracle docs](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) , **the fail-fast behavior of an iterator cannot be guaranteed** as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw ConcurrentModificationException on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: **the fail-fast behavior of iterators should be used only to detect bugs.**

**How  Fail  Fast Iterator  come to know that the internal structure is modified ?**

Iterator read internal data structure (object array) directly . The internal data structure(i.e object array) should not be modified while iterating through the collection. To ensure this it maintains an internal  flag *"mods" .*Iterator checks the *"mods" flag* whenever it gets the next value (using hasNext() method and next() method). Value of *mods* flag changes whenever there is an structural modification. Thus indicating iterator to throw ConcurrentModificationException.

**Fail Safe Iterator :**

First of all, there is no term as fail-safe given in many places as Java SE specifications does not use this term. I am using this term to demonstrate the difference between Fail Fast and Non-Fail Fast Iterator. These iterators make a copy of the internal collection (object array) and iterates over the copied collection. Any structural modification done to the iterator affects the copied collection, **not original collection**. So, original collection remains structurally **unchanged**.



fail safe iterator is ordinarily too costly, but may be more efficient than alternatives when traversal operations vastly outnumber mutations, and is useful when you cannot or don’t want to synchronize traversals, yet need to preclude interference among concurrent threads. The "snapshot" style iterator method uses a reference to the state of the array at the point that the iterator was created. This **array never changes during the lifetime of the iterator, so interference is impossible and the iterator is guaranteed not to throw ConcurrentModificationException**.The iterator will not reflect additions, removals, or changes to the list since the iterator was created. Element-changing operations on iterators themselves (remove(), set(), and add()) are not supported. These methods throw UnsupportedOperationException.

