What is fail safe and fail fast Iterator in Java?

Java Collections supports two types of Iterator, fail-safe and fail fast. The main distinction between a fail-fast and fail-safe Iterator is whether or not the underlying collection can be modified while it begins iterated. If you have used Collection like ArrayList then you know that when you iterate over them, no other thread should modify the collection. If the Iterator detects any structural change after iteration has begun e.g adding or removing a new element then it throws `ConcurrentModificationException`, this is known as fail-fast behavior and these iterators are called *fail-fast iterator* because they fail as soon as they detect any modification.

Though it's not necessary that iterator will throw this exception when multiple threads modified it simultaneously. it can happen even with the single thread when you try to remove elements by using ArrayList's remove() method instead of Iterator's remove method, as discussed in my earlier post, **2 ways to remove objects from ArrayList**.

Most of the Collection classes from Java 1.4 e.g. `Vector`, `ArrayList`, `HashMap`, HashSet has fail-fast iterators. The other type of iterator was introduced in Java 1.5 when concurrent collection classes
e.g. `ConcurrentHashMap`, `CopyOnWriteArrayList`, and `CopyOnWriteArraySet` was introduced.

This iterator uses a view of the original collection for doing iteration and that's why they don't throw `ConcurrentModificationException` even when the original collection was modified after iteration has begun. This means you could iterate and work with stale value, but this is the cost you need to pay for a fail-safe iterator and this feature is clearly documented

Difference between Fail Safe and Fail Fast Iterator in Java

In order to best understand the difference between these two iterators, you need to try out examples with both traditional collections like ArrayList and concurrent collections like `CopyOnWriteArrayList`. Nevertheless, let's first see some key differences one at a time :

1) fail-fast Iterator throws ConcurrentModfiicationException as soon as they detect any structural change in the collection during iteration, basically which changes the modCount variable hold by Iterator. While fail-fast iterator doesn't throw CME.

You can also see Core Java Volume 1 - Fundamentals by Cay S. Horstmann to learn more about how to use Iterator and the properties of different types of iterators in Java.

2) Fail-fast iterator traverse over original collection class while fail-safe iterator traverse over a copy or view of the original collection. That's why they don't detect any change on original collection classes and this also means that you could operate with stale value.

3) Iterators from Java 1.4 Collection classes like ArrayList, HashSet and Vector are fail-fast while Iterators returned by concurrent collection classes like CopyOnWriteArrayList or CopyOnWriteArraySet are fail-safe.

4) Iterator returned by synchronized Collections are fail-fast while iterator returned by concurrent collections are fail-safe in Java.

5) Fail fast iterator works in live data but become invalid when data is modified while fail-safe iterator are weekly consistent.

When to use fail fast and fail-safe Iterator

Use fail-safe iterator when you are not bothered about Collection to be modified during iteration, as fail-fast iterator will not allow that. Unfortunately, you can't choose to fail safe or fail-fast iterator, it depends on which Collection class you are using.

Most of the JDK 1.4 Collections e.g. `HashSet`, `Vector`, `ArrayList` has fail-fast Iterator and only Concurrent Collections introduced in JDK 1.5
e.g. `CopyOnWriteArrayList` and `CopyOnWriteArraySet` supports fail-safe Iteration.

Also, if you want to remove elements during iteration please use the iterator's `remove()` method and don't use the remove method provided by Collection classes e.g. ArrayList or HashSet because that will result in `ConcurrentModificationException`.