Why the HashMap isn't thread-safe

The HashMap

- Lacks synchronization.
- There are no guarantees of memory consistency, while iterating.

Map Classes

	Sorted	Blocking	Thread-Safe	Stream Pipeline - Collectors Method
HashMap	No	No	No	`.collect(groupingBy(Function classifier, ...)`
TreeMap	Yes	No	No	`.collect(TreeMap<keyType, valueType>::new, ...`
ConcurrentHashMap	No	No	Yes	`.collect(groupingByConcurrent(Function classifier, ..)`
ConcurrentSkipListMap	Yes	No	Yes	`.collect(ConcurrentSkipListMap<keyType, valueType>::new, ...`
Collections$SynchronizedMap	Yes	Yes	Yes	

Concurrent Classes vs. Synchronized Wrapper Classes

Both concurrent and synchronized collections are thread-safe, and can be used in parallel streams, or in a multi-threaded application.

- **Synchronized collections** are implemented using locks which protect the collection from concurrent access. This means a single lock is used to synchronize access to the entire map.
- **Concurrent collections** are more efficient than synchronized collections, because they use techniques like fine-grained locking, or non-blocking algorithms to enable safe concurrent access without the need for heavy handed locking, meaning synchronized or single access locks.

Concurrent collections are recommended over synchronized collections in most scenarios.