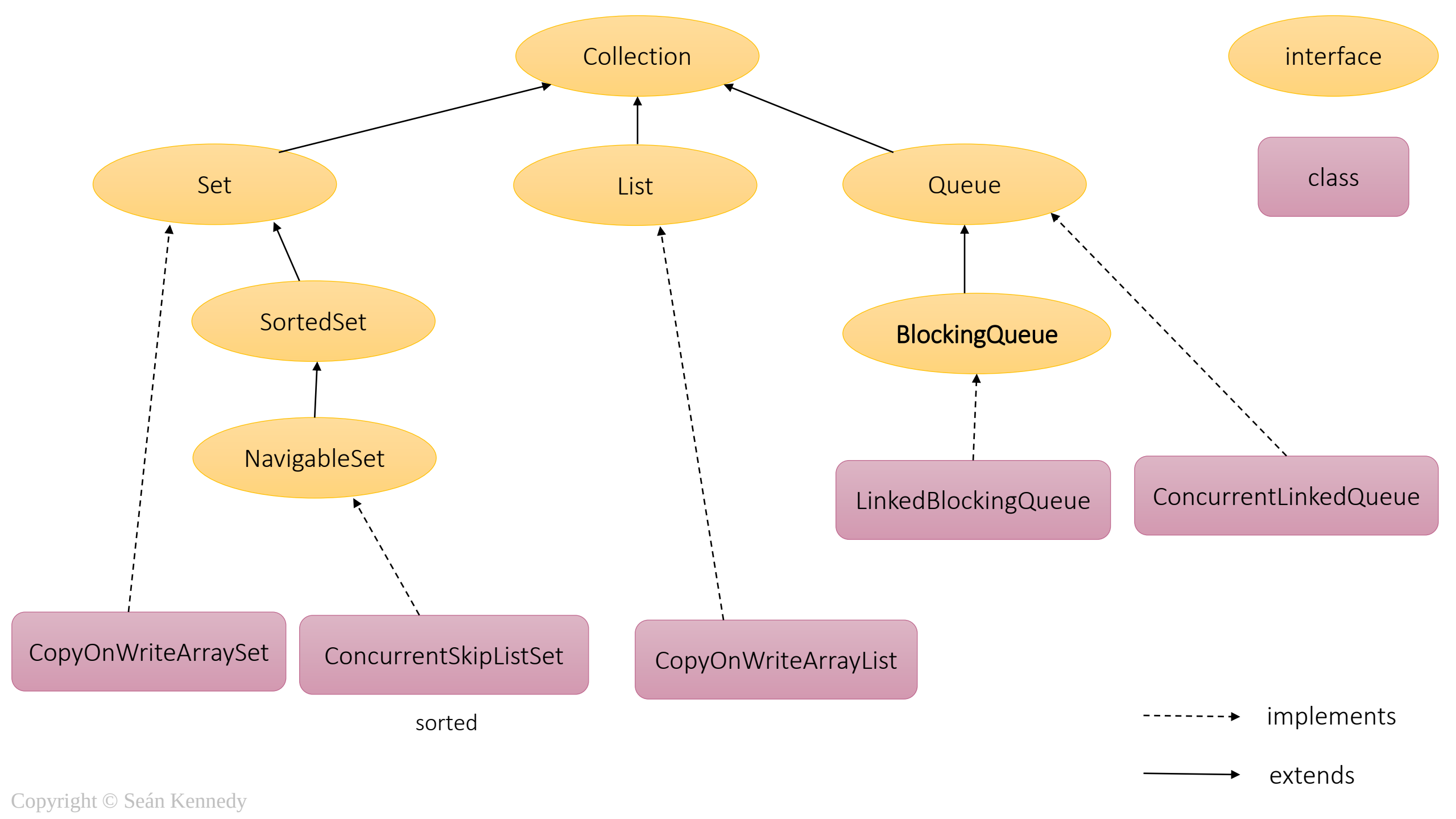
Concurrency

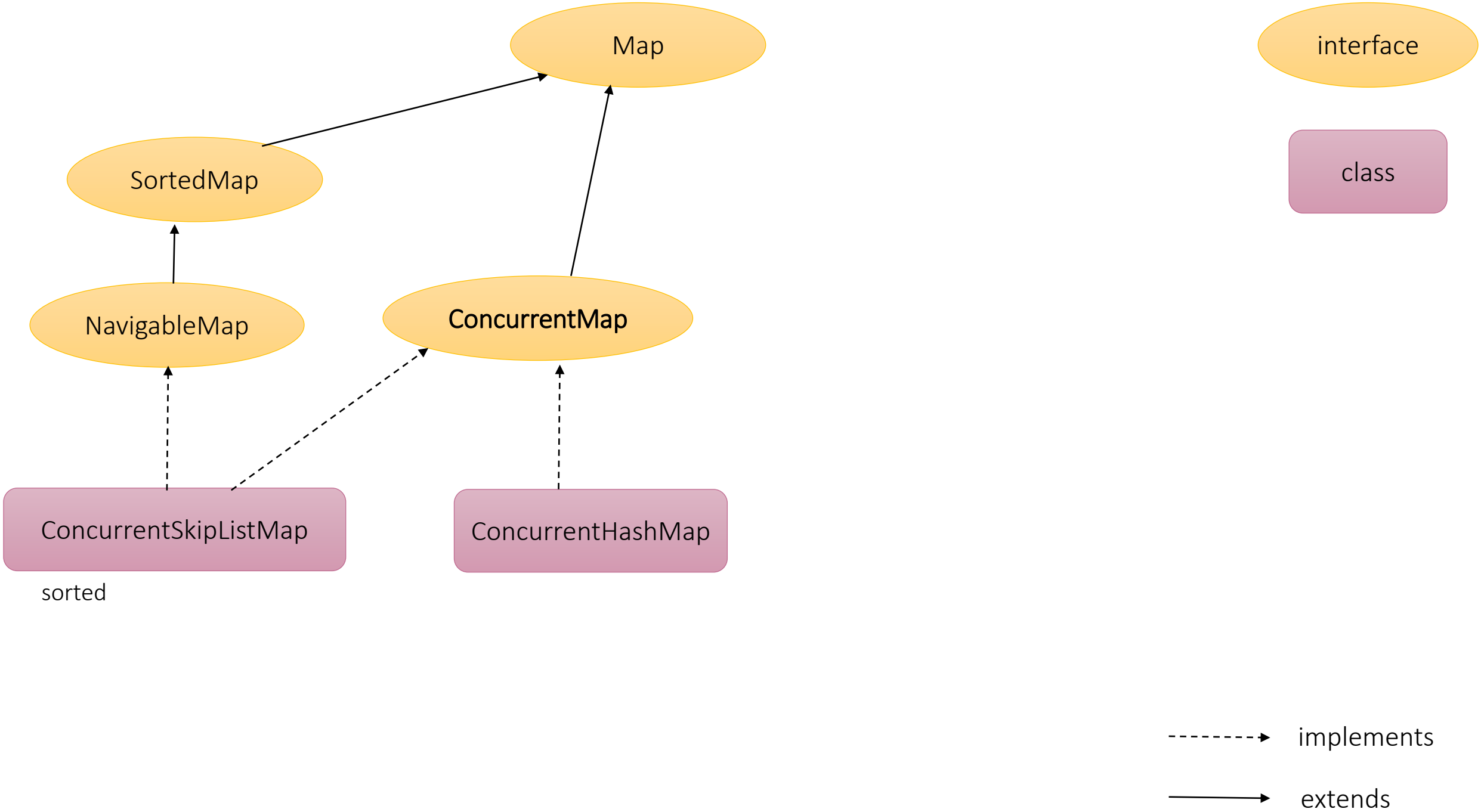
Concurrent Collections

Concurrent collections

- The problem with regular collections
- Concurrent collections overview
- SkipList collections
- CopyOnWrite collections
- Blocking queues
- *synchronized* collections

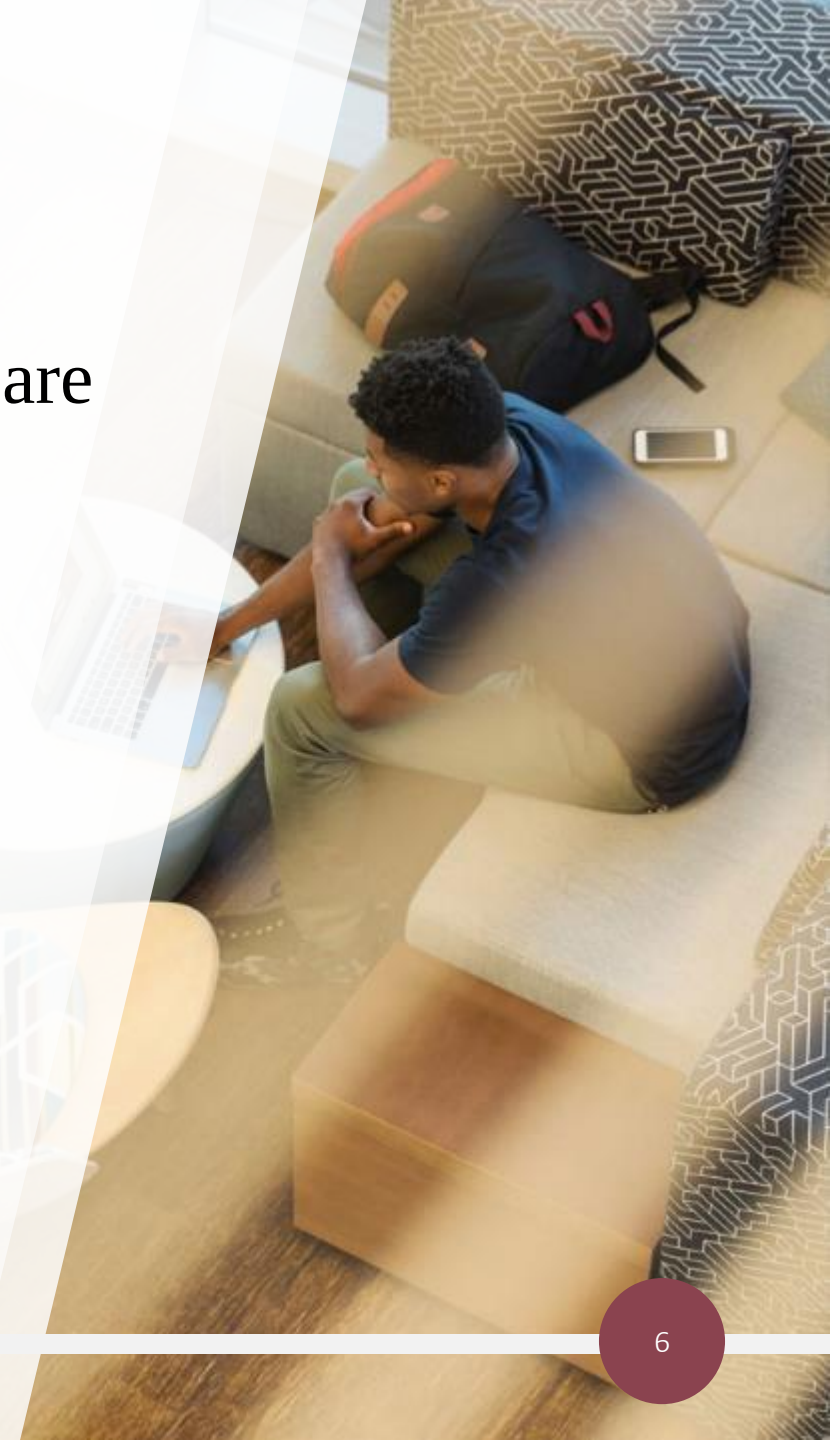
- TheProblem.java





SkipList Collections

- *ConcurrentSkipListSet* and *ConcurrentSkipListMap* are sorted concurrent collections.
- Sorted by natural order.
- *ConcurrentSkipListSet* ⇔ *TreeSet*
- *ConcurrentSkipListMap* ⇔ *TreeMap*



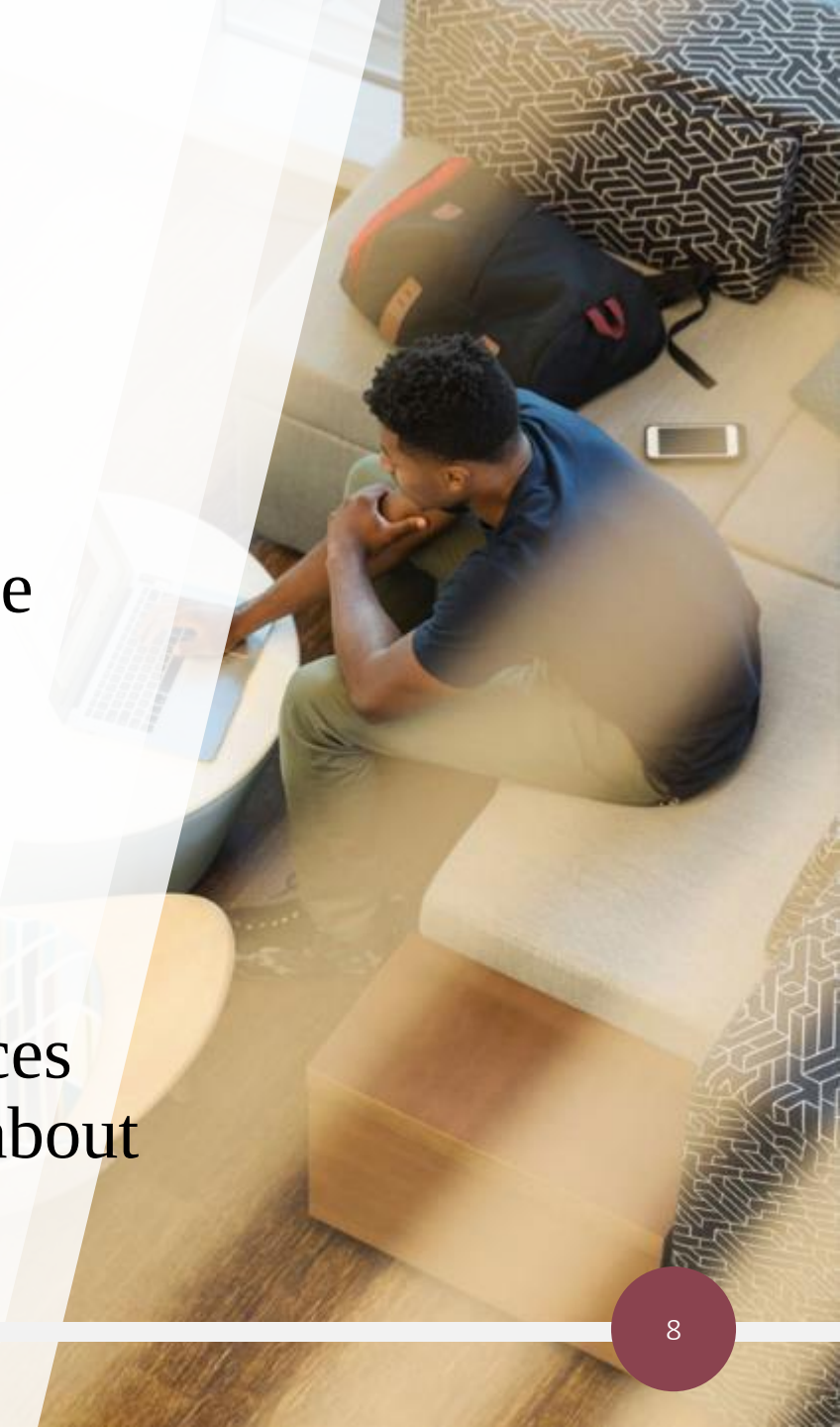
- SkipListCollections.java

CopyOnWrite Collections

- *CopyOnWriteArrayList* and *CopyOnWriteArraySet*

- Suitable for situations where you read a lot but write very little.

- When an object is added to or deleted from the collection, a copy is made of the entire collection.
  - not talking about the object state that the references held in the collection refer to but we are talking about the references themselves.

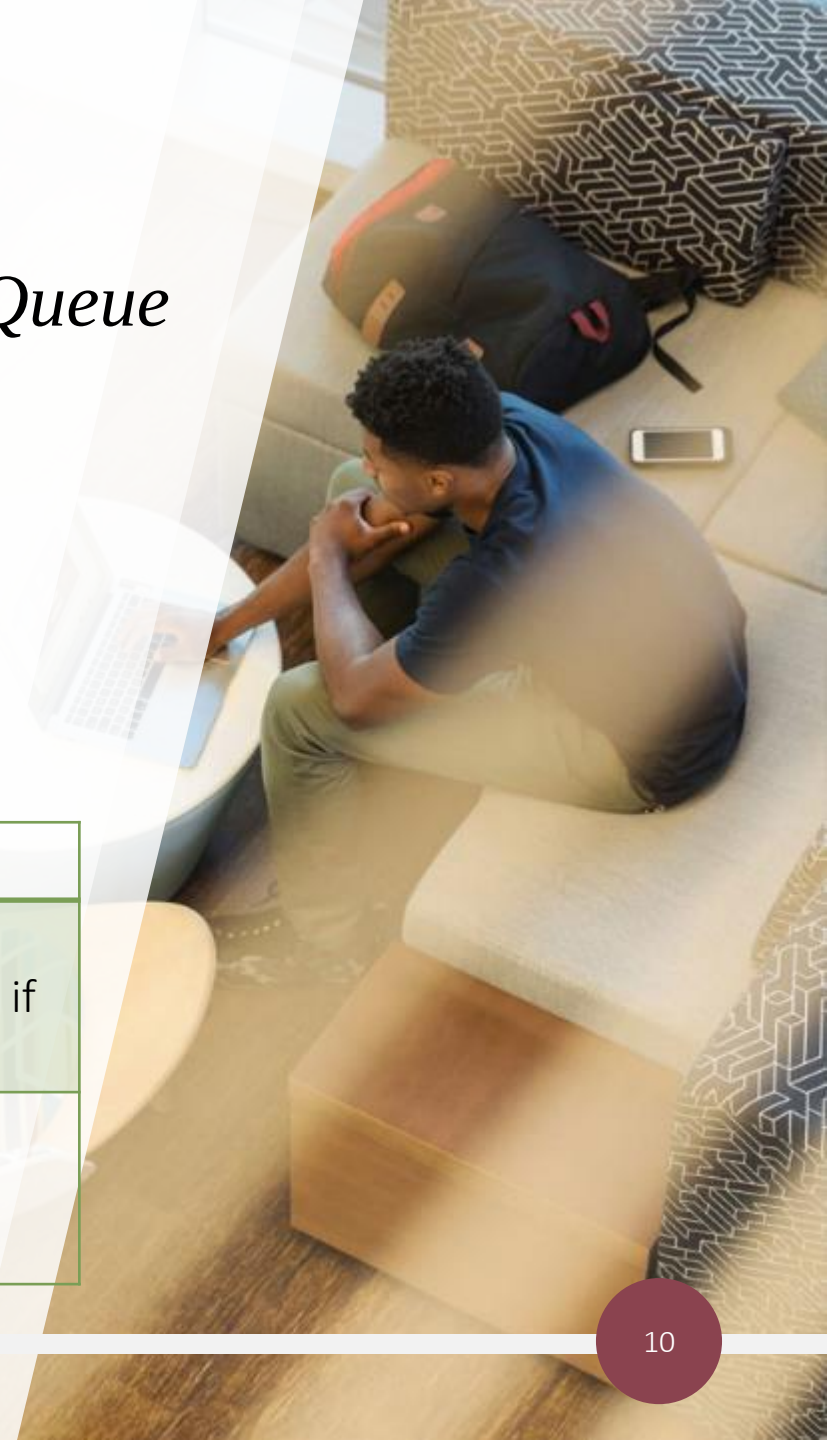


- CopyOnWriteCollections.java

Blocking queues

- While *ConcurrentLinkedQueue* and *LinkedBlockingQueue* are both thread-safe *Queue* implementations, *LinkedBlockingQueue* adds some extra methods for blocking i.e. waiting for a certain time period.

Method name	Description
E poll(long timeout, TimeUnit unit)	Retrieves and removes the head of this queue, waiting up to the specified time if necessary (queue may be empty)
offer(E e, long timeout, TimeUnit unit)	Inserts the element into the queue, waiting up to the specified time if necessary (queue may be full)

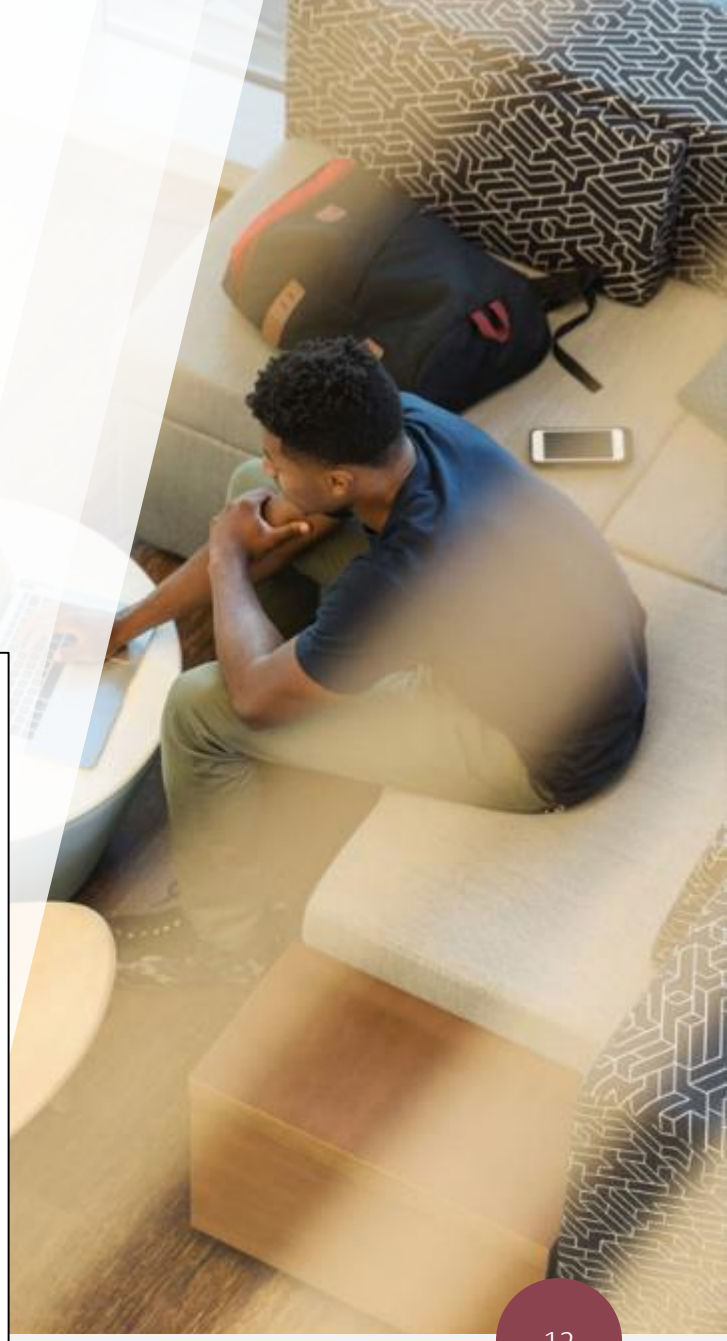


- BlockingQueueExample.java

synchronized Collections

- We can get a synchronized version of a non-concurrent collection using the *Collections* utility class.

```
java.util.Collections.synchronizedCollection(Collection)
java.util.Collections.synchronizedList(List)
java.util.Collections.synchronizedMap(Map)
java.util.Collections.synchronizedNavigableMap(NavigableMap)
java.util.Collections.synchronizedNavigableSet(NavigableSet)
java.util.Collections.synchronizedSet(Set)
java.util.Collections.synchronizedSortedMap(SortedMap)
java.util.Collections.synchronizedSortedSet(SortedSet)
```

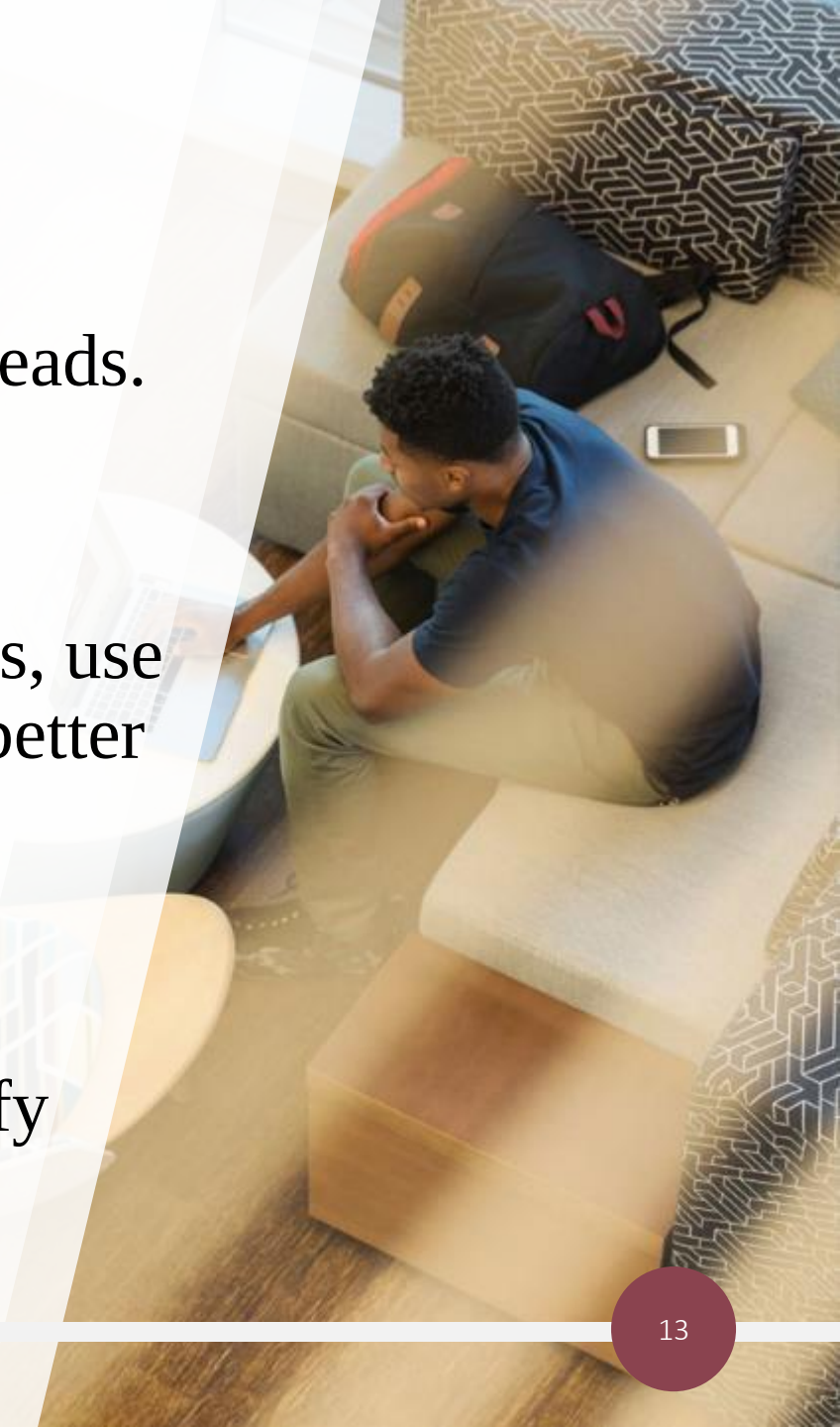


synchronized Collections

- Useful if you are given an *existing* non-concurrent collection and you want to share it among several threads.

- However, if you know when you are *creating* your collection that you require concurrency across threads, use the concurrent collections outlined in the overview (better performance).

- Note: synchronized collections also throw *ConcurrentModificationException* if you try to modify them inside a loop (unlike concurrent collections).



- SynchronizedCollection.java