Concurrency

Concurrent Collections

Using Collections with Threads

- multiple threads performing operations on the same collection can be dangerous
- *memory consistency error*
  - two threads have inconsistent views of what should be the same data
  - for example: one thread removes the element, but the other thread "didn't get the memo" and it behaves like the element is still there
- to avoid these kind of errors we have to use thread-safe collections
- these are provided by Java via Concurrent Classes, e.g.

```
Map<String, Integer> myMap = new ConcurrentHashMap<>();
```

Concurrent Collection Classes

Class Name	Java Collection interfaces
ConcurrentHashMap	Map, ConcurrentMap
ConcurrentLinkedQueue	Queue
ConcurrentSkipListMap	Map, SortedMap, NavigableMap, ConcurrentMap, ConcurrentNavigableMap
ConcurrentSkipListSet	Set, SortedSet, NavigableSet
CopyOnWriteArrayList	List
CopyOnWriteArraySet	Set
LinkedBlockingQueue	Queue, BlockingQueue

Synchronized Collections methods

`synchronizedCollection(Collection<T> c)`
`synchronizedList(List<T> list)`
`synchronizedMap(Map<K,V> m)`
`synchronizedNavigableMap(NavigableMap<K,V> m)`
`synchronizedNavigableSet(NavigableSet<T> s)`
`synchronizedSet(Set<T> s)`
`synchronizedSortedMap(SortedMap<K,V> m)`
`synchronizedSortedSet(SortedSet<T> s)`