4.4 Lecture Summary

4.4 Concurrent HashMap

Lecture Summary: In this lecture, we studied the *ConcurrentHashMap* data structure, which is available as part of the *java.util.concurrent* standard library in Java. A *ConcurrentHashMap* instance, *chm*, implements the *Map* interface, including the *get(key)* and *put(key, value)* operations. It also implements additional operations specified in the *ConcurrentMap* interface (which in turn extends the *Map* interface); one such operation is *putIfAbsent(key, value)*. The motivation for using *putIfAbsent()* is to ensure that only one instance of *key* is inserted in *chm*, even if multiple threads attempt to insert the same key in parallel. Thus, the semantics of calls to *get()*, *put()*, and *putIfAbsent()* can all be specified by the theory of linearizability studied earlier. However, it is worth noting that there are also some aggregate operations, such as *clear()* and *putAll()*, that cannot safely be performed in parallel with *put()*, *get()* and *putIfAbsent()*.

Motivated by the large number of concurrent data structures available in the *java.util.concurrent* library, this lecture advocates that, when possible, you use libraries such as *ConcurrentHashMap* rather than try to implement your own version.

Optional Reading:

1. Documentation on Java's ConcurrentHashMap class

2. Wikipedia article on Java's ConcurrentMap interface

Mark as completed