Applying the Java ScheduledExecutor Service to TimedMemoizerEx

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

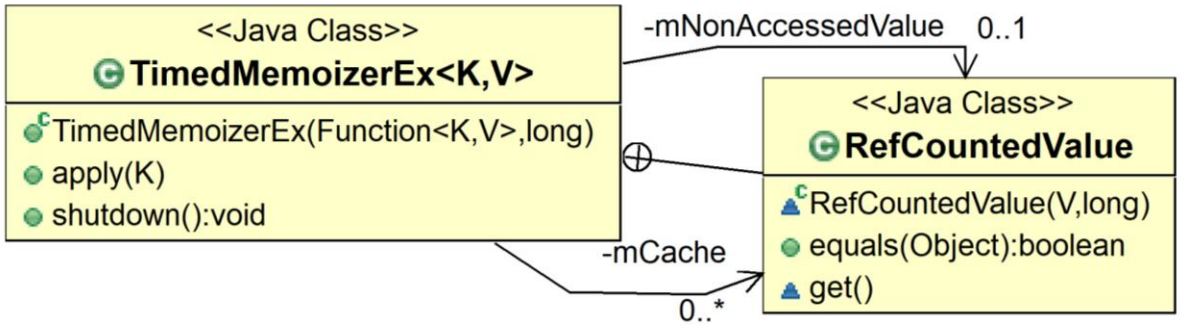
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

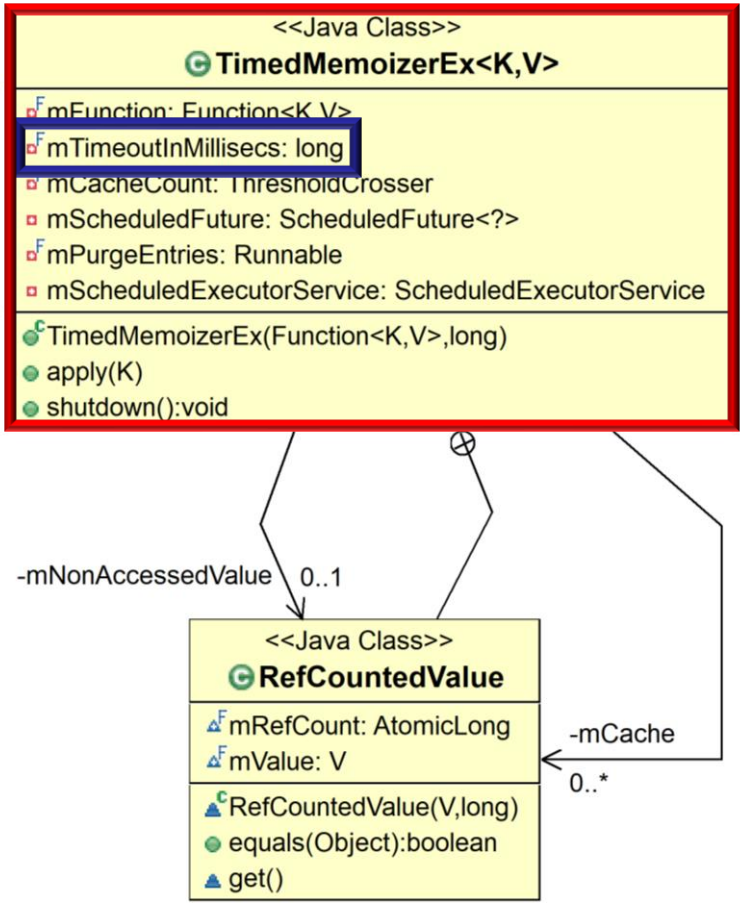
- Learn how to create a TimedMemoizerEx that applies the ScheduledExecutor Service to remove stale entries



Overview of TimedMemoizerEx

Overview of TimedMemoizerEx

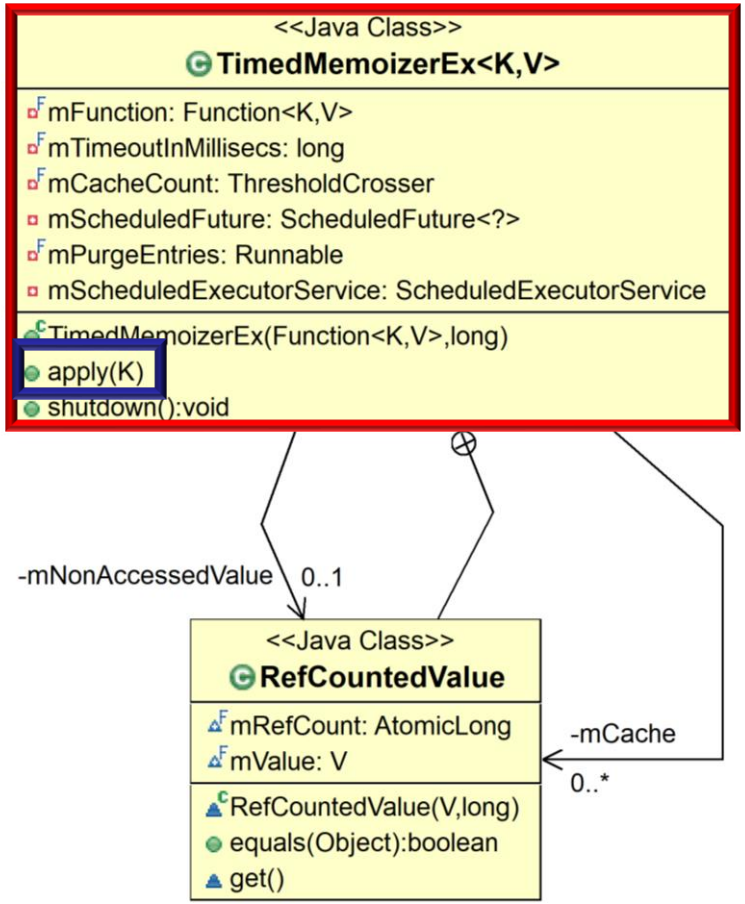
- TimedMemoizerEx maps a key to the value produced by a function, but *efficiently* limits the time a key/value pair remains cached



See PrimeExecutorCompletionService/app/src/main/java/vandy/mooc/prime/utils/TimedMemoizerEx.java

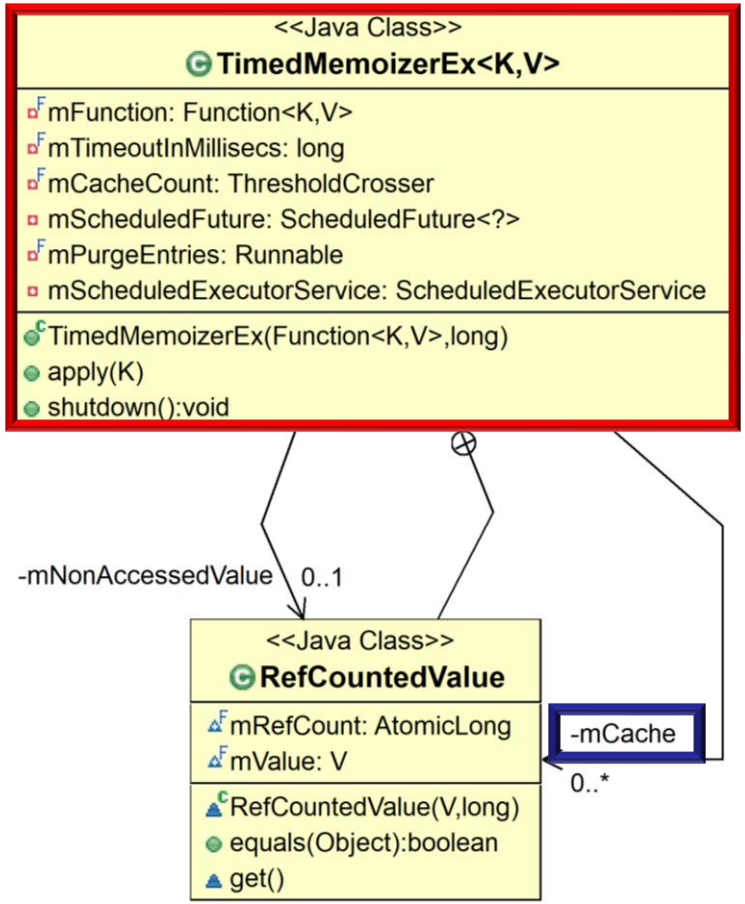
Overview of TimedMemoizerEx

- TimedMemoizerEx maps a key to the value produced by a function, but *efficiently* limits the time a key/value pair remains cached
  - If a value has been computed for a key it is returned rather than calling the function to compute it again



Overview of TimedMemoizerEx

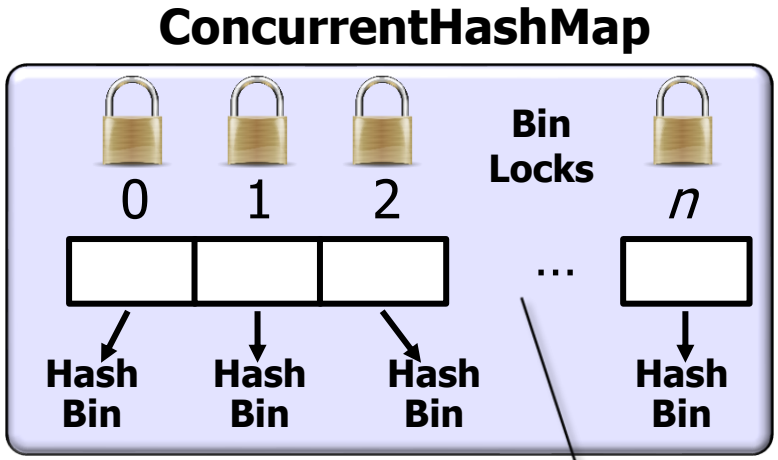
- TimedMemoizerEx uses ConcurrentHashMap to minimize synchronization overhead



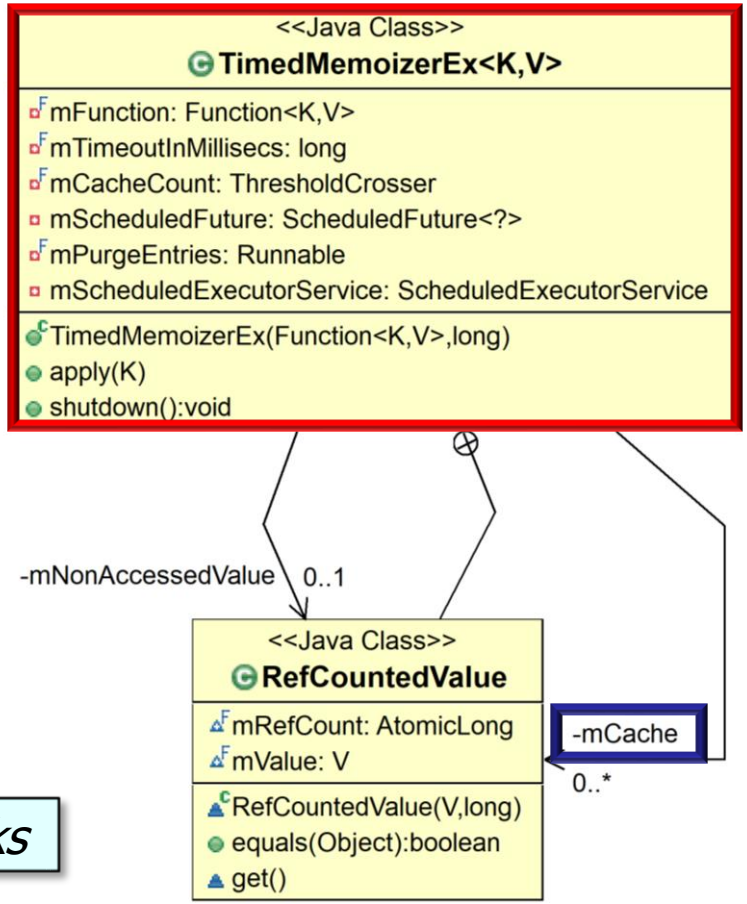
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html

Overview of TimedMemoizerEx

- TimedMemoizerEx uses ConcurrentHashMap to minimize synchronization overhead
  - A different lock guards each hash bin



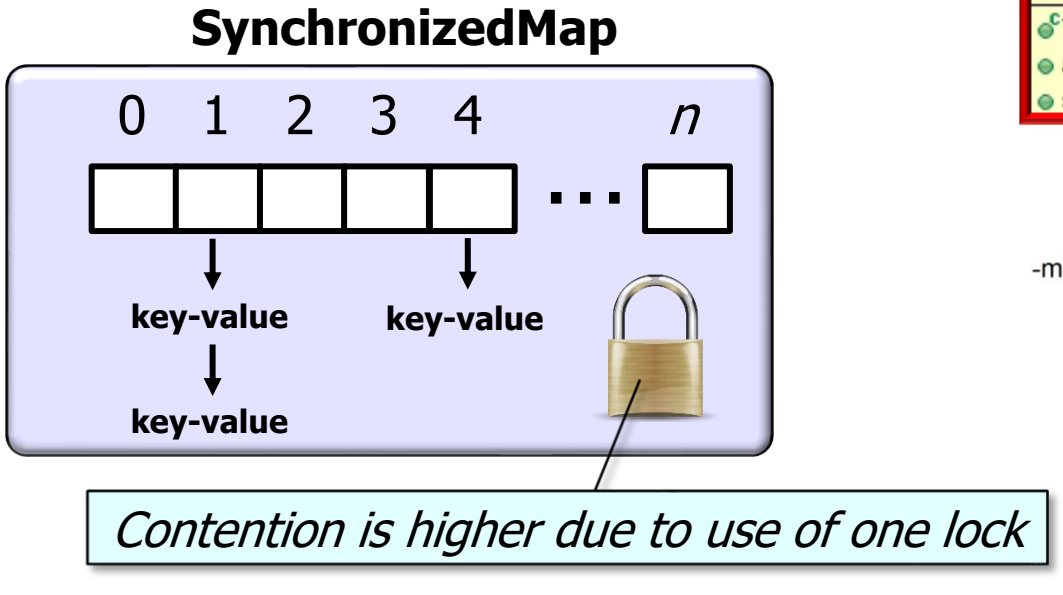
Contention is low due to use of multiple locks

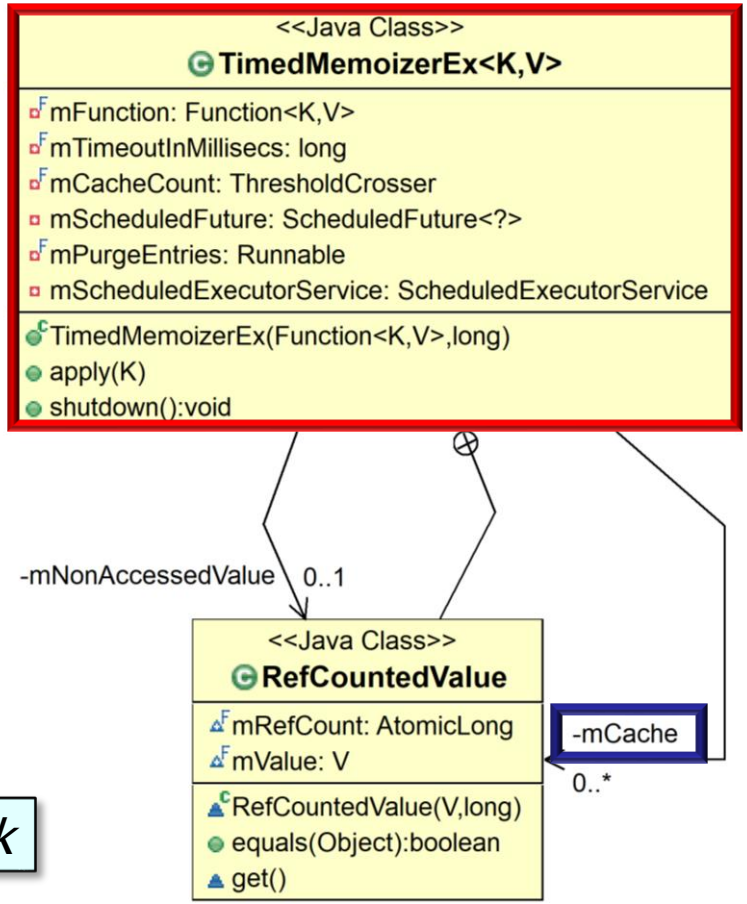


See www.ibm.com/developerworks/java/library/j-jtp08223

Overview of TimedMemoizerEx

- TimedMemoizerEx uses ConcurrentHashMap to minimize synchronization overhead
  - A different lock guards each hash bin
  - A SynchronizedMap just uses one lock

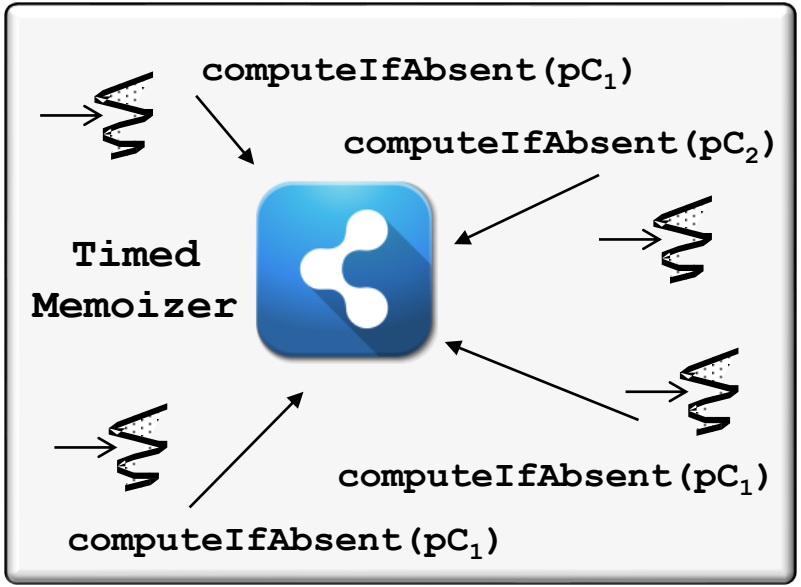




See codepumpkin.com/hashtable-vs-synchronizedmap-vs-concurrenthashmap

Overview of TimedMemoizerEx

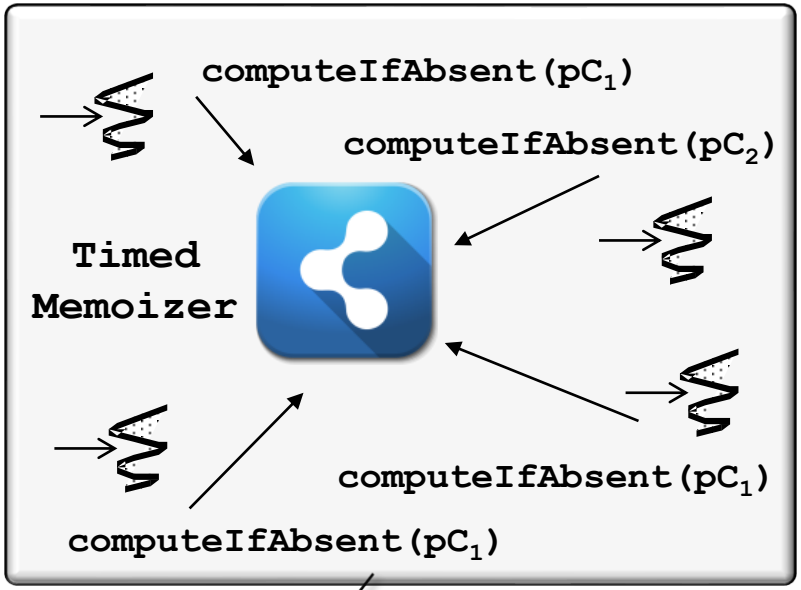
- TimedMemoizerEx uses ConcurrentHashMap to minimize synchronization overhead
  - A different lock guards each hash bin
  - computeIfAbsent() ensures only one call to function runs when a key & value are first added to the cache



See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html#computeIfAbsent

Overview of TimedMemoizerEx

- TimedMemoizerEx uses ConcurrentHashMap to minimize synchronization overhead
  - A different lock guards each hash bin
  - computeIfAbsent() ensures only one call to function runs when a key & value are first added to the cache

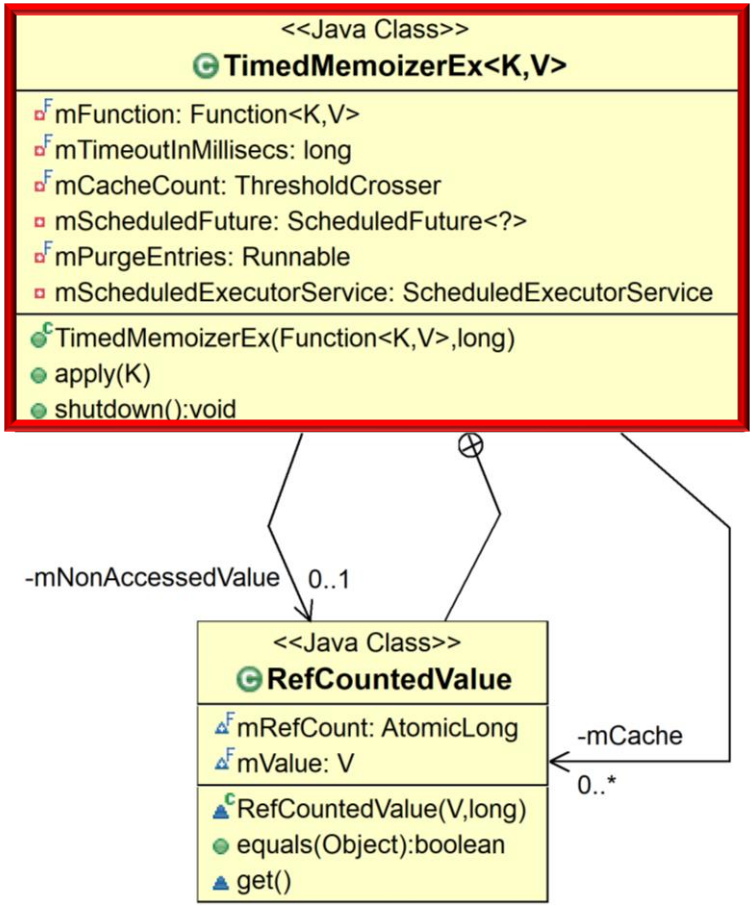


Only one computation per key is performed even if multiple threads call computeIfAbsent() simultaneously using same key

Eliminates FutureTask (ashkrit.blogspot.com/2014/12/what-is-new-in-java8-concurrenthashmap.html)

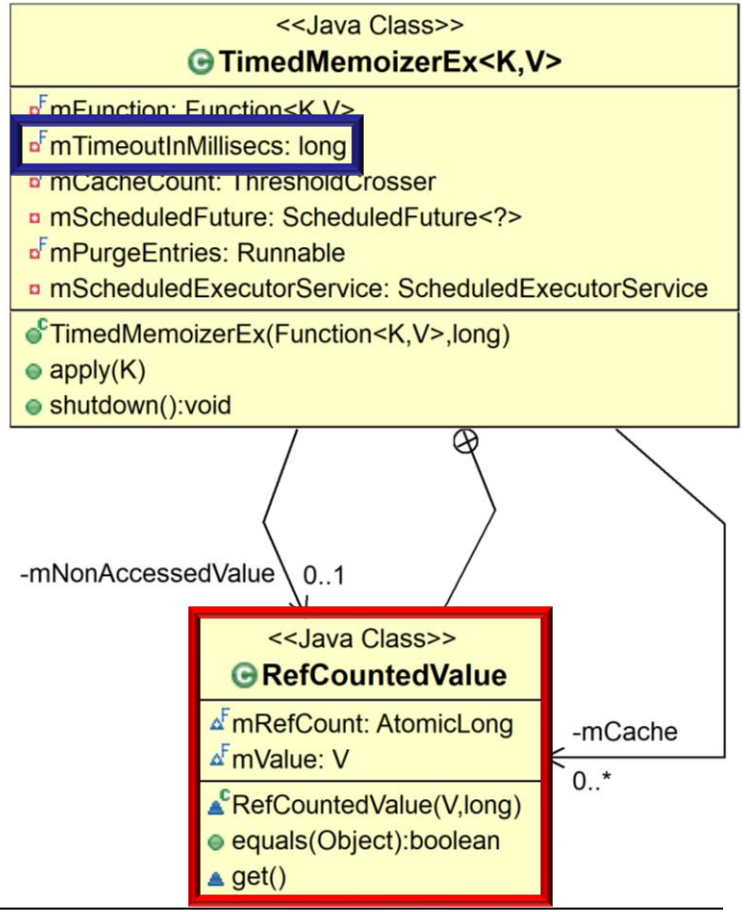
Overview of TimedMemoizerEx

- If a key isn't accessed within a given period TimedMemoizerEx purges it from the map



Overview of TimedMemoizerEx

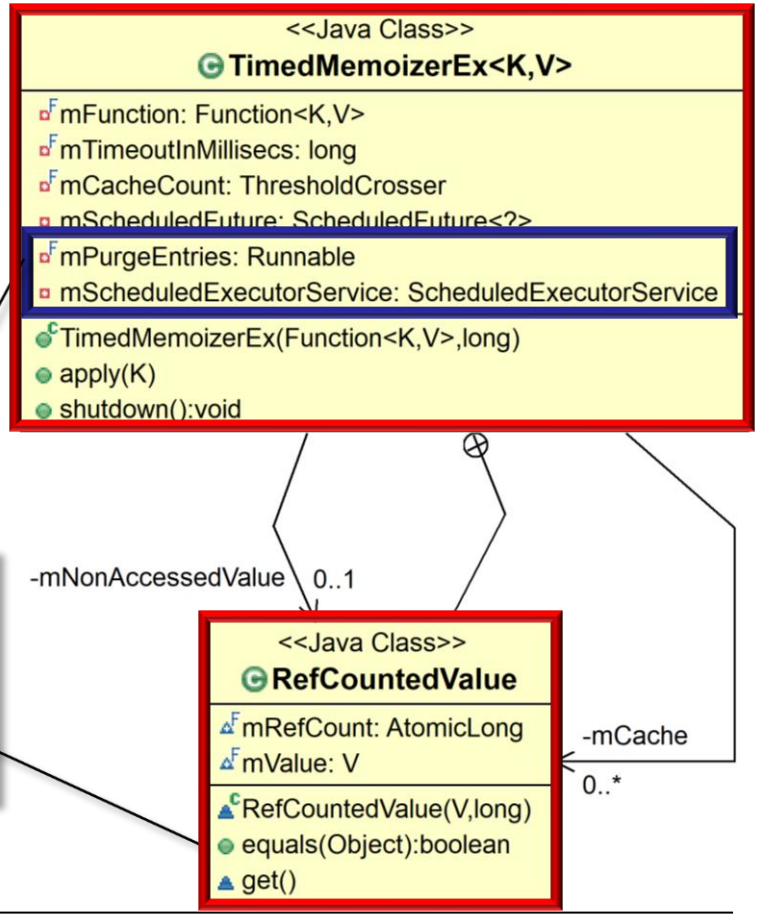
- If a key isn't accessed within a given period TimedMemoizerEx purges it from the map
  - RefCountedValue tracks # of times a key is referenced within a given # of millisecs



Overview of TimedMemoizerEx

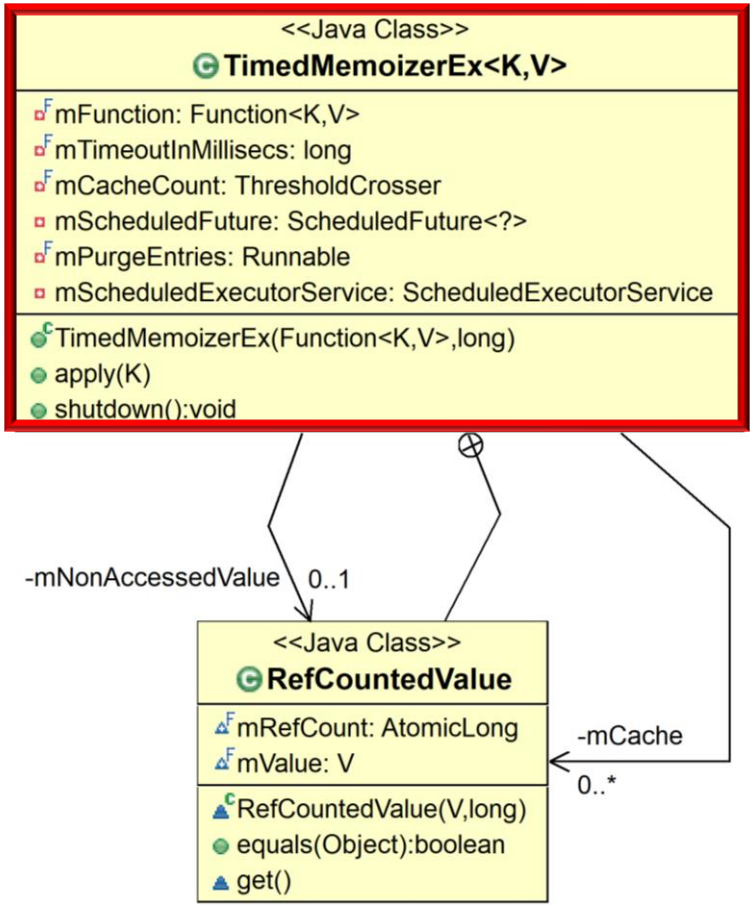
- If a key isn't accessed within a given period TimedMemoizerEx purges it from the map
  - RefCountedValue tracks # of times a key is referenced within a given # of millisecs
  - Timeout logic is performed by scheduling a single "mPurgeEntries" runnable

ScheduledExecutorService runs mPurgeEntries periodically to purge entries from cache that haven't been accessed recently



Overview of TimedMemoizerEx

- If a key isn't accessed within a given period TimedMemoizerEx purges it from the map
  - RefCountedValue tracks # of times a key is referenced within a given # of millisecs
  - Timeout logic is performed by scheduling a single "mPurgeEntries" runnable
  - This implementation consumes much less memory than the previous one



End of Applying the Java ScheduledExecutor Service to TimedMemoizerEx