参考：<http://www.source-code.biz/snippets/java/6.htm>

**An LRU cache class based on java.util.LinkedHashMap**

An LRU (least recently used) cache is used to buffer a limited number of the MRU (most recently used) objects of a class in memory.

|  |  |
| --- | --- |
| API documentation: | [LRUCache.html](http://www.source-code.biz/snippets/java/LRUCache.html) |
| Download full package: | [LRUCache.zip](http://www.source-code.biz/snippets/java/LRUCache.zip) |

import java.util.LinkedHashMap;  
import java.util.Collection;  
import java.util.Map;  
import java.util.ArrayList;  
  
/\*\*  
\* An LRU cache, based on <code>LinkedHashMap</code>.  
\*  
\* <p>  
\* This cache has a fixed maximum number of elements (<code>cacheSize</code>).  
\* If the cache is full and another entry is added, the LRU (least recently used) entry is dropped.  
\*  
\* <p>  
\* This class is thread-safe. All methods of this class are synchronized.  
\*  
\* <p>  
\* Author: Christian d'Heureuse, Inventec Informatik AG, Zurich, Switzerland<br>  
\* Multi-licensed: EPL / LGPL / GPL / AL / BSD.  
\*/  
public class LRUCache<K,V> {  
  
private static final float hashTableLoadFactor = 0.75f;  
  
private LinkedHashMap<K,V> map;  
private int cacheSize;  
  
/\*\*  
\* Creates a new LRU cache.  
\* @param cacheSize the maximum number of entries that will be kept in this cache.  
\*/  
public LRUCache (int cacheSize) {  
 this.cacheSize = cacheSize;  
 int hashTableCapacity = (int)Math.ceil(cacheSize / hashTableLoadFactor) + 1;  
 map = new LinkedHashMap<K,V>(hashTableCapacity, hashTableLoadFactor, true) {  
 // (an anonymous inner class)  
 private static final long serialVersionUID = 1;  
 @Override protected boolean removeEldestEntry (Map.Entry<K,V> eldest) {  
 return size() > LRUCache.this.cacheSize; }}; }  
  
/\*\*  
\* Retrieves an entry from the cache.<br>  
\* The retrieved entry becomes the MRU (most recently used) entry.  
\* @param key the key whose associated value is to be returned.  
\* @return the value associated to this key, or null if no value with this key exists in the cache.  
\*/  
public synchronized V get (K key) {  
 return map.get(key); }  
  
/\*\*  
\* Adds an entry to this cache.  
\* The new entry becomes the MRU (most recently used) entry.  
\* If an entry with the specified key already exists in the cache, it is replaced by the new entry.  
\* If the cache is full, the LRU (least recently used) entry is removed from the cache.  
\* @param key the key with which the specified value is to be associated.  
\* @param value a value to be associated with the specified key.  
\*/  
public synchronized void put (K key, V value) {  
 map.put (key, value); }  
  
/\*\*  
\* Clears the cache.  
\*/  
public synchronized void clear() {  
 map.clear(); }  
  
/\*\*  
\* Returns the number of used entries in the cache.  
\* @return the number of entries currently in the cache.  
\*/  
public synchronized int usedEntries() {  
 return map.size(); }  
  
/\*\*  
\* Returns a <code>Collection</code> that contains a copy of all cache entries.  
\* @return a <code>Collection</code> with a copy of the cache content.  
\*/  
public synchronized Collection<Map.Entry<K,V>> getAll() {  
 return new ArrayList<Map.Entry<K,V>>(map.entrySet()); }  
  
} // end class LRUCache

// Test routine for the LRUCache class.  
public static void main (String[] args) {  
 LRUCache<String,String> c = new LRUCache<String, String>(3);  
 c.put ("1", "one"); // 1  
 c.put ("2", "two"); // 2 1  
 c.put ("3", "three"); // 3 2 1  
 c.put ("4", "four"); // 4 3 2  
 if (c.get("2") == null) throw new Error(); // 2 4 3  
 c.put ("5", "five"); // 5 2 4  
 c.put ("4", "second four"); // 4 5 2  
 // Verify cache content.  
 if (c.usedEntries() != 3) throw new Error();  
 if (!c.get("4").equals("second four")) throw new Error();  
 if (!c.get("5").equals("five")) throw new Error();  
 if (!c.get("2").equals("two")) throw new Error();  
 // List cache content.  
 for (Map.Entry<String, String> e : c.getAll())  
 System.out.println (e.getKey() + " : " + e.getValue()); }

Author: [Christian d'Heureuse](mailto:chdh@source-code.biz) ([www.source-code.biz](http://www.source-code.biz/), [www.inventec.ch/chdh](http://www.inventec.ch/chdh))  
[Index](http://www.source-code.biz/snippets/java/)