利用链表和HashMap实现LRU缓存

<https://www.cnblogs.com/lzrabbit/p/3734850.html>

**LRU Cache的链表+HashMap实现**

注：此实现为非线程安全，若在多线程环境下使用需要在相关方法上添加synchronized以实现线程安全操作

package cn.lzrabbit.structure.lru;

import java.util.HashMap;

/\*\*

\* Created by liuzhao on 14-5-12.

\*/

public class LRUCache1<K, V> {

**private final int MAX\_CACHE\_SIZE;**

**private Entry first;**

**private Entry last;**

private HashMap<K, Entry<K, V>> hashMap;

public **LRUCache1**(int cacheSize) {

MAX\_CACHE\_SIZE = cacheSize;

hashMap = new HashMap<K, Entry<K, V>>();

}

public void put(K key, V value) {

**Entry entry = getEntry(key);**

if (entry == null) {

if (hashMap.size() >= MAX\_CACHE\_SIZE) {

**hashMap.remove(last.key);**

**removeLast();**

}

**entry = new Entry();**

**entry.key = key;**

}

**entry.value = value;**

**moveToFirst(entry);**

**hashMap.put(key, entry);**

}

public V get(K key) {

Entry<K, V> entry = **getEntry(key);**

if (entry == null) return null;

**moveToFirst(entry);**

return entry.value;

}

public void remove(K key) {

Entry entry = getEntry(key);

if (entry != null) {

**if (entry.pre != null) entry.pre.next = entry.next;**

**if (entry.next != null) entry.next.pre = entry.pre;**

**if (entry == first) first = entry.next;**

**if (entry == last) last = entry.pre;**

}

**hashMap.remove(key);**

}

private void **moveToFirst**(Entry entry) {

if (entry == first) return;

//先断开当前节点的关系

**if (entry.pre != null) entry.pre.next = entry.next;**

**if (entry.next != null) entry.next.pre = entry.pre;**

if (entry == last) last = last.pre;

if (first == null || last == null) {

first = last = entry;

return;

}

**entry.next = first;//将之前的first设为entry的next**

**first.pre = entry;**

**first = entry;//另当前Entry为first**

entry.pre = null;

}

private void **removeLast**() {

if (last != null) {

last = last.pre;

if (last == null) first = null;

else last.next = null;

}

}

private Entry<K, V> **getEntry**(K key) {

**return hashMap.get(key);**

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

Entry entry = first;

while (entry != null) {

sb.append(String.format("%s:%s ", entry.key, entry.value));

entry = entry.next;

}

return sb.toString();

}

**class Entry<K, V> {**

**public Entry pre;**

**public Entry next;**

**public K key;**

**public V value;**

**}**

}