**LRU Cache wiki**

1. Programming language used:

I have used Java for implementing LRU cache as it provides functions for synchronizing during multi-threading and it also provides Hashmap class used in LRU cache implementation.

1. Logic used and Space time complexity:

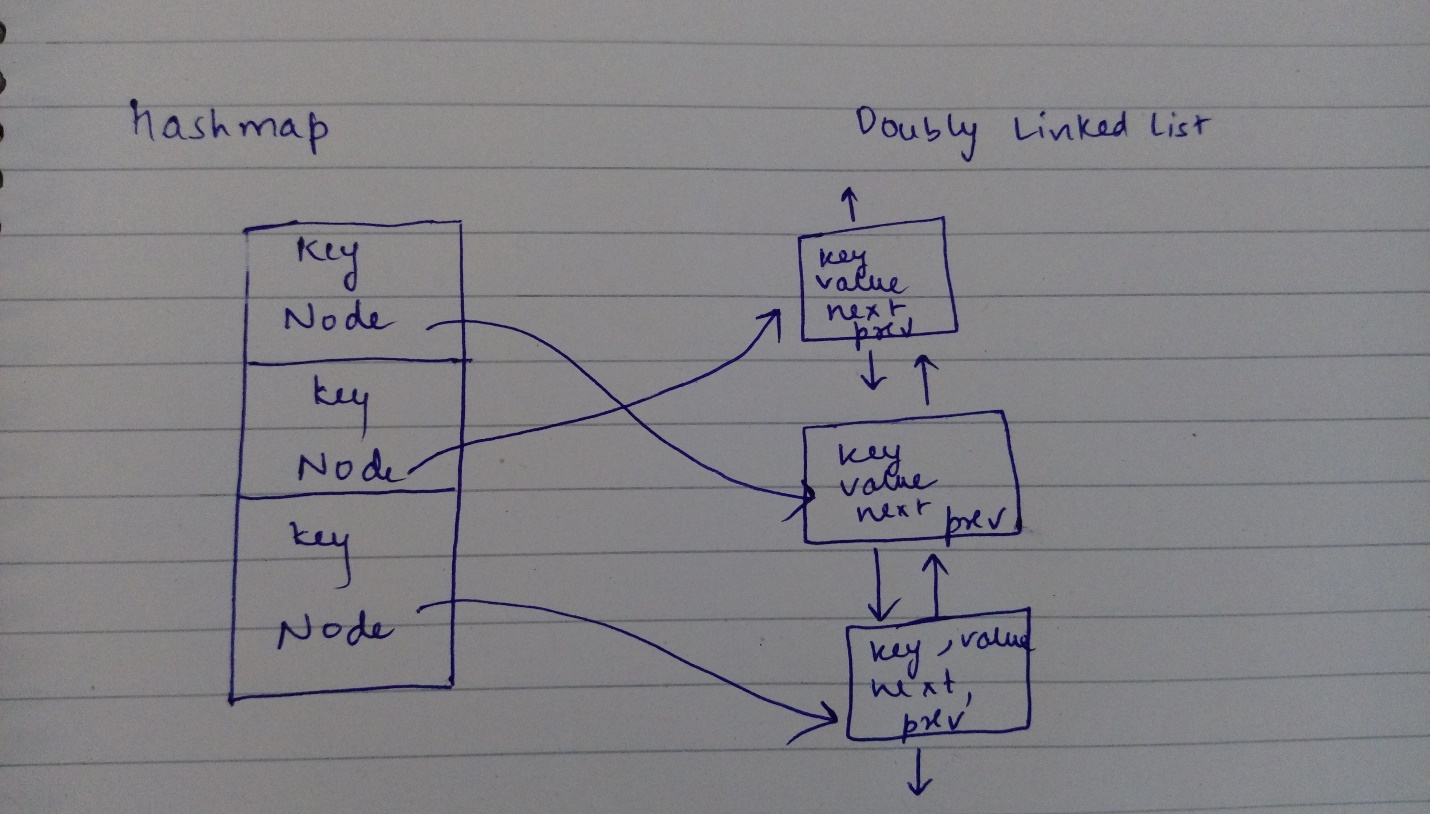
LRU Cache replaces the least recently used page when the disk is full. The main operations on the cache are data access and adding a new data. For fast data access (O(1) complexity) , we need a Hashmap. But we also need to store an order in which data is accessed so that we can remove the least recently used page.

So, I have used 2 data structures for fast data access and data addition. A doubly linked list like structure has been used to maintain an order of pages accessed.The node stores key and value of the key. When a key is accessed, the corresponding node is removed and added to the head of the linked list. In this way, the least recently used data is at the tail of the linked list. Since removing a node from a doubly linked list is O(1), hence a doubly linked list has been chosen. A Hashmap stores the key and and pointer to the corresponding linked list node and hence data access is O(1) complexity.

So,

**Put ( key, value) :O(1)**

**Get (key) : O(1)**



1. Implementation:

Doubly Linked list has been implemented as a java class. Hashmap provided by java.utils has been used. For making thread safe, synchronized block has been used while accessing get () and put() , locked on a shared static lock. LRUCache.java and Node.java implement the above logic.

1. Test Bench:

Testcase1() is a simple single threaded function which creates a LRU Cache of size 5 and adds 6 data to it. The LRU data is replaced.

Testcase2() implemets two thread accessing a common shared LRU Cache. Only one thread is adding is key and value (6,6) while the other thread tries to access this. This is done without any error.   
Also, since synchronized block is used, if multiple threads have their own LRU caches, there is no data contention.