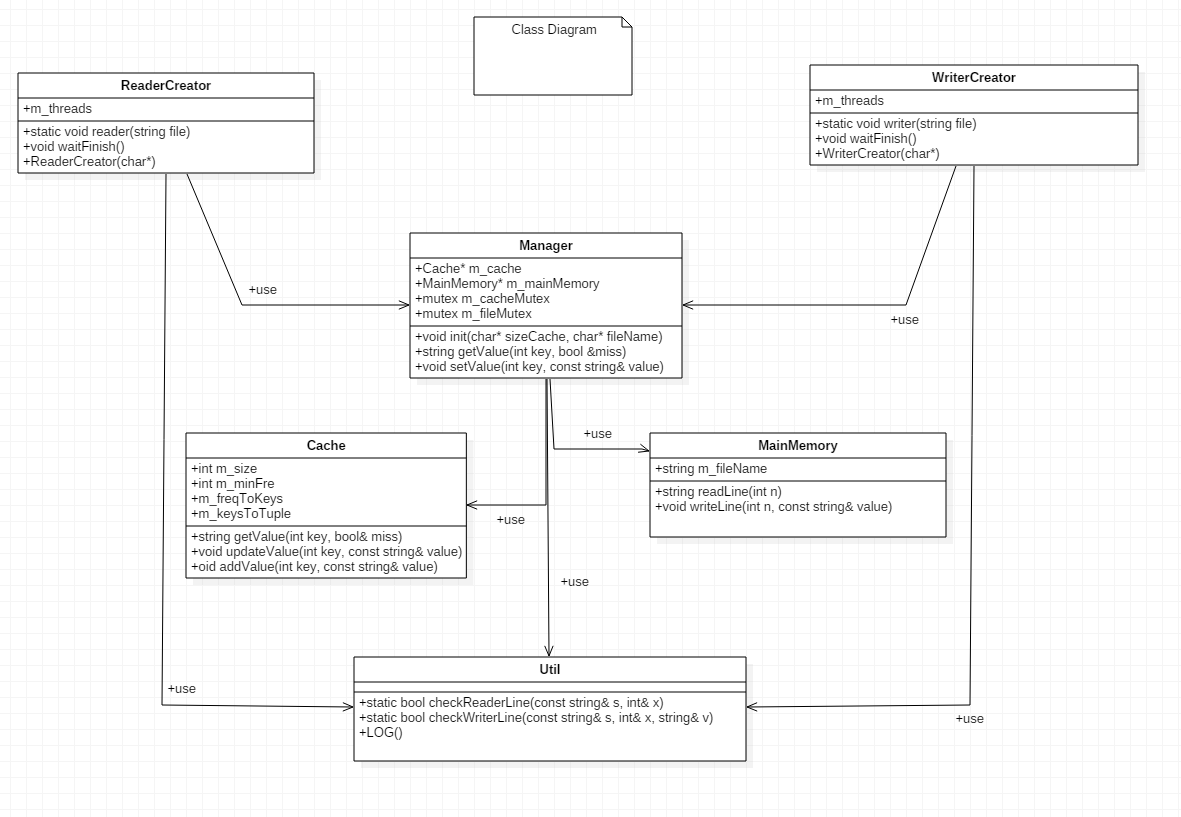
**Introduce:**

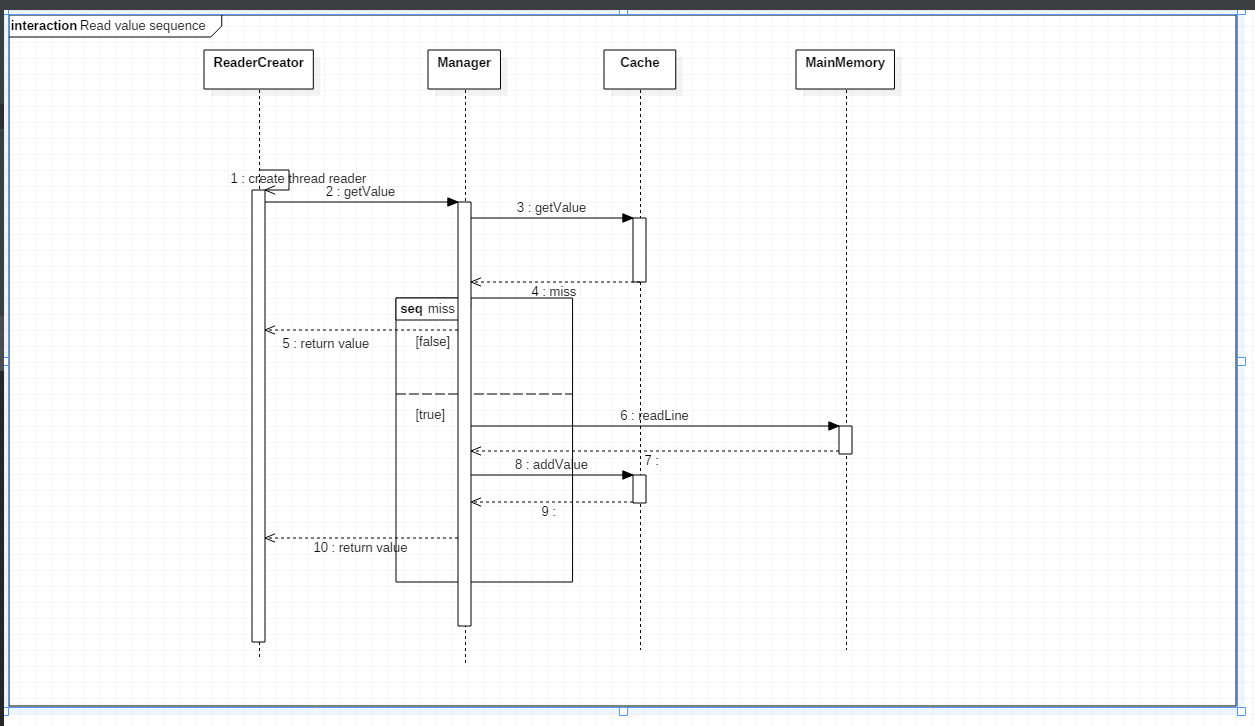
This program demo Least Frequently Used caching. In this program, the least frequently used cache block is removed whenever the cache is overflowed.

1. Class diagram

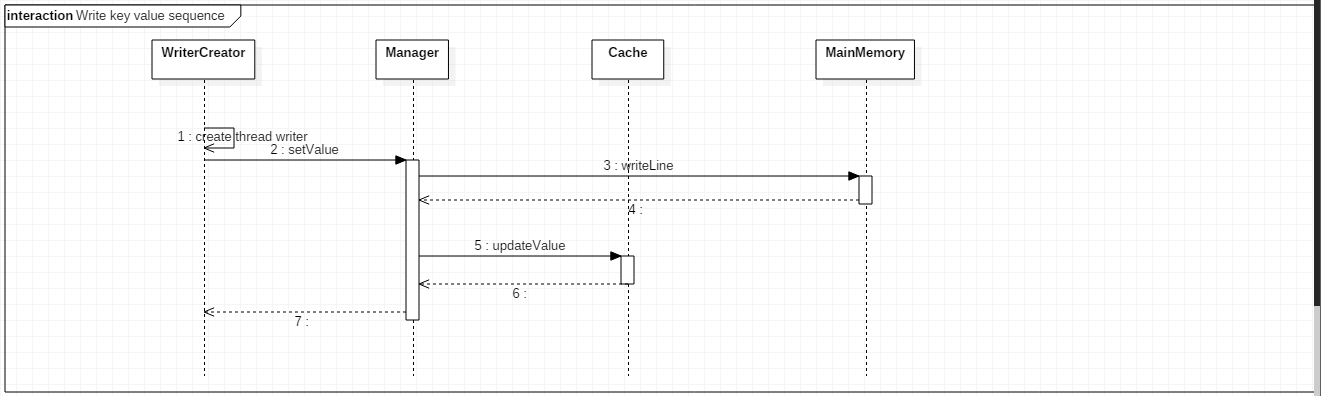


|  |  |
| --- | --- |
| Class | Descriptions |
| ReaderCreator | Create reading threads, each thread will read continuously cache or main memory. |
| WriterCreator | Create writing threads, each thread will write continuously main memory. |
| Manager | Manage multithread requests, check using Cache or Main Memory |
| Cache | Fast access memory, address will be check in Cache first, if hit, don’t need access Main Memory |
| Main Memory | Slow access memory, if cache miss, will be search in Main Memory |
| Util | Use to log activities, check input format and further |

1. Sequence diagram



Reading sequence: When have read requests from reader, Manager will check that key in Cache or not. If hit, return value in Cache. If miss, request to Main Memory.

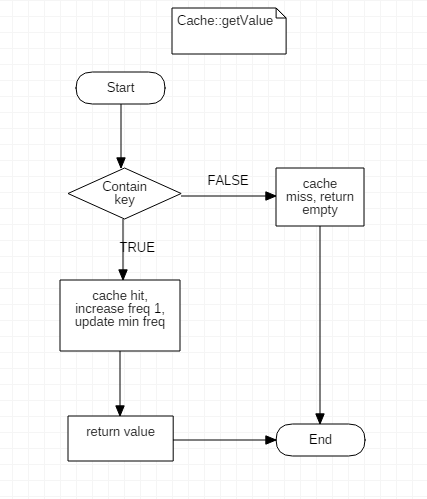


Writing sequence: When have write request from writer, Manager will request Main Memory write line. After that, check in Cache have that key, if exist update new value.

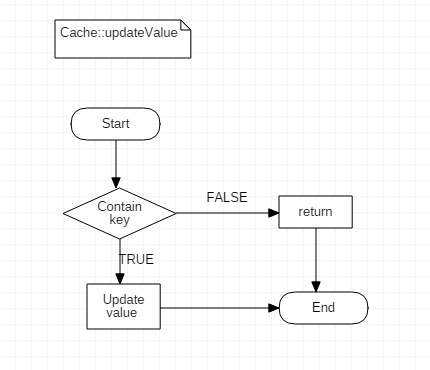
1. Flowchart diagram

In Cache, we store key value of Main Memory in unordered map

* GetValue function (use when reader want to get value from a line): check key is in map or not. If not return cache miss, if have increase frequence 1 and return value



* UpdateValue (use when write new value in Main Memory and recheck in Cache to avoid dirty item): check if key is in Cache, update new value for it.



-AddKey (use when cache miss, access Main Memory and add value got to Cache): If key contain increase frequence 1. Otherwise, check Cache is full, if not insert new key value, if full remove least frequently use and insert new key value

* 