Set-Associative Cache Design

Design:

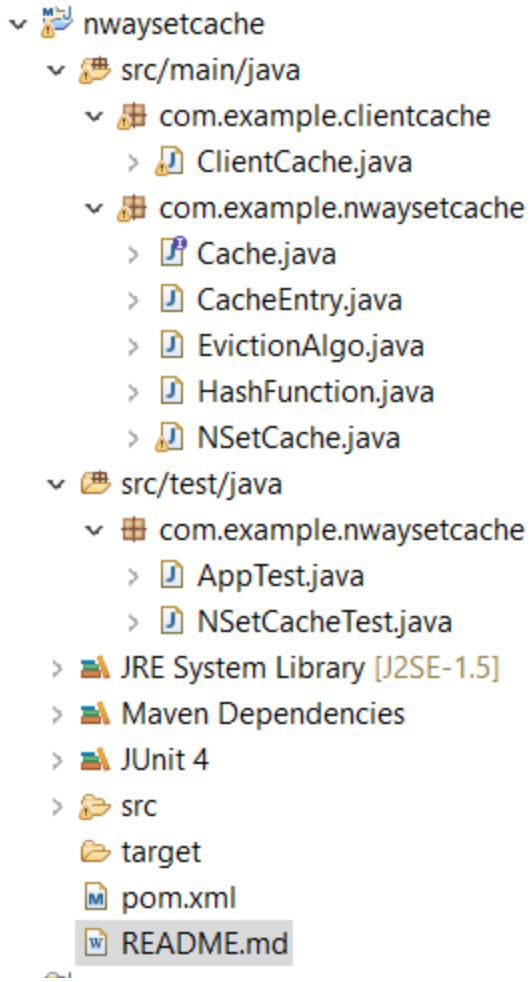
An array of CacheEntry<V> is taken. CacheEntry is a POJO that represent an entry in the Cache. It contains tag (or key), data (or Value) and timestamp. Whenever we are inserting an entry into the cache, the key is hashed, and we determine startIndex and endIndex which represents a set. Basically, that key is always associated with that set. If the set is fully occupied one of the entry is evicted and the currently inserted key takes the evicted position.

startIndex and endIndex are calculated as follows:

```
startIndex = (intKey % setNumber) * entryNumber;
endIndex = startIndex + entryNumber - 1;
```

Whenever we want to get a value based on key, we now search between startIndex and endIndex to retrieve the value. If it is not present then we get it from the database.

Structure of project:



Cache.java: It's an interface with following method declarations.

V get(K key) – used to retrieve data related to given key.

Void put(K key, V value) – used to put data into the cache associated with given key.

void clear() – used to reset the cache

V remove(K key) – used to remove data from the cache related to given key.

CacheEntry.java – It's a simple POJO that represents a single entry in the cache. It has instance variables, tag(key), data(Value) and timestamp.

NSetCache.java – It implements the cache.java interface.

EvictionAgo.java – It contains the LRU and MRU replacement algos.

HashFunction.java – It contains hashing algorithm for key mapping to cache address location.

ClientCache.java – Client can implement an alternative replacement algorithm here.