Cache engine

- Definitions
- Simple cache engine
- Max size eviction
- Timer
- Life time
- Idle time
- Example

Definition

Cache

A cache is a collection of temporary data that either duplicates data located elsewhere or is the result of a computation. Data that is already in the cache can be repeatedly accessed with minimal costs in terms of time and resources.

Cache hit

When a data element is requested from cache and the element exists for the given key, it is referred to as a cache hit (or simply, "a hit").

Cache miss

When a data element is requested from cache and the element does not exist for the given key, it is referred to as a cache miss (or simply, "a miss").

Caches are in the middle between source of date and consumer.

You can use caches with databases, files, web services, long computations.

Simple cache engine

Map<Key, Value> is the most simple cache.

It supports operations:

- put
- get
- size

Additional operations, not supported by Map:

- max_size, evict old value if size == max_size
- set time of life, evict by timeout
- set time of idle, evict if not accessed
- get hits count
- get misses count

Max size eviction

We need to specify the maximum size of the cache.

We need to evict old element to get space for new one.

Example

```
if (elements.size() == maxElements) {
  K firstKey = elements.keySet().iterator().next();
  elements.remove(firstKey);
}
```

This is eviction with type FIFO (if elements is LinkedHashMap)

java.util.Timer

Allows to schedule tasks for future execution in a background thread.

Example

```
TimerTask task = new TimerTask() {
  @Override
  public void run() {
    //do something
  }
};
timer.schedule(task, 5000);
```

Task will be executed 5 seconds after scheduling.

Life time

Life time is the amount of time to live for an element from its creation date.

After this time the element must be evicted from the cache.

Example

```
new TimerTask() {
  @Override
  public void run() {
    MyElement element = elements.get(key);
    if (element == null || element.getCreationTime() + lifeTime < System.currentTimeMillis())) {
      elements.remove(key);
    }
  }
};
```

Idle time

Idle time the amount of time to live for an element from its last accessed or modified date.

After this time the element must be evicted from the cache.

Example

```
new TimerTask() {
  @Override
  public void run() {
    MyElement element = elements.get(key);
    if (element == null || element.getAccessTime() + idleTime < System.currentTimeMillis())) {
      elements.remove(key);
    }
  }
};
```