LIBRARY USAGE

Setup and Build

CacheBuilder<String, String> builder = new CacheBuilder<>();

1. Use default settings —> *builder.setDefault*();
2. Set N-Way —> *builder.setNWay*(i);
3. Set size —> *builder.setSize*(s);
4. Set Eviction Algorithm
   - LRU —> *builder.setLRU*();
   - MRU —> *builder.setMRU*();
   - Custom Algorithm —> *builder.setCustomAlgo*(Eviction<Key, Value> customAlgorithm);
5. Build and get Cache instance

   —> Cache<Key, Value> cache = *builder.build*();

Operations

Cache<Key, Value> cache = builder.*build*();

1. Put data —> *cache.put*(key, value);
2. Get data —> *cache.get*(key);
3. Set new value to a data —> *cache.put*(key, newValue);
4. Remove data —> *cache.remove*(key);
5. Clear cache —> *cache.clear*();
6. Check usage amount —> *cache.size*();

Custom Eviction Algorithm

1. Make sure your custom class ***extends Eviction<Key, Value>***

2. Make sure your custom class has a Constructor with no parameter, as well as a Constructor take ***Bag<Key, Value>*** as parameter and use ***super*** constructor
3. Make sure your custom class ***override evict(), name()*** methods.
4. useful API:
   - query Node's key by access order —> .getInOrder(int idx)
   - query Node's key by reversed access order —> .getRevOrder(int idx)
   - query Node's key by frequency order —> .freqOrder(int idx)
   - query Node's key by reversed frequency order —> .freqRevOrder(int idx)
   - query Node's key by specific frequency —> .oneFreq(int f)