Project 1 Document

Author: Yan Wang & Zishan Qin

Project Requirement

1. A values store with three interfaces: Put, Get and Remove.
2. The value byte array can be arbitrarily large, up to 1 GB.
3. Issue of multiple threadings.

Assumption

The total size of value store will not exceed the limit of memory

Design

1. We built a class called ValueStore. In this class, we used HashMap to store tuples, which is (key, byte [ ] data) in this project. Methods (put, get, remove) of HashMap are borrowed to generate the “put”, “get”, “remove” interfaces of ValueStore.
2. ValueStore is serialized, so that every time we close the program, the object (containing data) will be saved to the hard drive, and every time we run the program, the object (data) will be reloaded.
3. Generating 1GB data on the fly is painful. In this project, we generated a 1GB file, and read in the file as the 1GB data.
4. All three methods “put”, “get”, “remove” are synchronized. Every method will wait until it is notified by other threadings. After the method is implemented, it will notify other threadings.
5. For each method, we built a “client” class, which implements runnable, to mimic different clients calling different methods.

Test

Test 1GB

One 1GB data is generated in the program, and to be used for testing the methods. Please note that in order to avoid “out of heap memory” error, we need to put in “Debug Configuration-> arguments-> VM arguments” such setting “-Xmx2G”, so as to set the heap memory as 2G.

```
The length is data0 is 1
The length is data1 is 1
The length is data2 is 1073741824
```

Test multiple threadings

Multiple threadings are generated to mimic multiple client. Except for the first threading, we don’t control the order of other threadings. Therefore, we generated a bunch of threadings for

“get”, so as to display the “put” and “remove” go well. Result of this test shows below. (It takes quite a while to operate on 1GB, so we used small data for later test.)

```
Reading key 0 with first element of value as null
Reading key 0 with first element of value as null
Writing key 0 with first element of value as -32
Reading key 0 with first element of value as -32
Reading key 0 with first element of value as -32
Writing key 0 with first element of value as -31
Reading key 0 with first element of value as -31
Reading key 0 with first element of value as -31
Writing key 0 with first element of value as 0
Reading key 0 with first element of value as 0
Reading key 0 with first element of value as 0
Removing key 0 with first element of value as 0
Reading key 0 with first element of value as null
Writing key 0 with first element of value as -31
Reading key 0 with first element of value as -3
```

Test Serialization

If something is stored, the we can get it after the program is closed and run again. Above, value in key 0 is -31. After reruning the program, it is still -31.

```
Reading key 0 with first element of value as -31
Reading key 0 with first element of value as -31
...
```