GIT

DEPARTMENT OF COMPUTER ENGINEERING

CSE 222/505 – Spring 2021

REPORT FOR HOMEWORK 5

SÜLEYMAN GÖLBOL

1801042656

1. PROBLEM SOLUTION APPROACH

My solution approach for part1 was to make MapIterator class a inner class.
Reason:
In the assignment, it was wanted that the constructors both (one of them is One parameter constructor and gets a key value to begin iterate, and other was zero parameter constructor to begin something any key.) to not take other parameter like Map itself.
But it was a problem because without using parameter I couldn't access the private data. If I made map public, it would have violate the rules of Object Oriented Programming so I decided to create MyHashMap class (that extends from real HashMap) and to create methods that call MapIterator's constructors and make MapIterator as inner class to easily accessing outer class' data.

My solution approach for part2 was using Coalesced hashing technique in putting element with quadratic probing.
Normally quadratic probing is using index that to be wanted to put, and if this hash index if full; using quadratic probing as hashIndex = index + counter*counter.
But when I tried to use this in put method, firstly it didn't work. Because this technique acting differently in my method. So I created a method called chechNextFull and I checked the next index that coalesced hashing shows. If there is also element in next index, I called this method again in a while loop. Until checkNextFull is false, I set the next index with element that I will add and I made sure that nexh hash index is not same with our index. Or it wouldn't make any sense.
I made the next index as integer, not string or else. So if next index is null, it shows as -1. -1 means that it doesn't show any element in next.

2. TEST CASES

5.1) CREATING A MAP AND PUTTING ELEMENTS

```
public static void iterating_part1(){
    MyHashMap<Integer, String> myMap = new MyHashMap<Integer, String>();

    myMap.put(1801042656, "Suleyman Golbol");
    myMap.put(1801042601, "Baran Solmaz");
    myMap.put(1801042634, "Muhammed Oguz");
    myMap.put(1801042647, "Kadir Saglam");

    System.out.println("Map<Integer,String> is:");
    System.out.println( myMap.toString() );
```

5.2) CREATING ITERATOR WITH MapIterator CLASS.

```
//Creating iterator with MapIterator that begins with 1801042656 key.
System.out.println("USING ONE PARAMETER CONSTRUCTOR:");
MyHashMap<Integer, String>.MapIterator iter = myMap.MapIterator(1801042656);
```

5.3) USING PREV() METHOD TO ITERATE TO PREVIOUSLY AND GETTING VALUES OF SOME KEYS.

```
System.out.println("USING PREV() METHOD:");
System.out.println( iter.prev() );
System.out.println( iter.prev() );
System.out.println( myMap.get(1801042634) );
System.out.println( myMap.get(1801042656) );
```

5.4) USING HASNEXT() AND NEXT() METHOD

```
System.out.println("USING HASNEXT() AND NEXT() METHOD:");
while( iter.hasNext() == true){
    System.out.println( iter.next() );
}
```

5.5) USING ZERO PARAMETER CONSTRUCTOR

```
System.out.println("USING ZERO PARAMETER CONSTRUCTOR:");
MyHashMap<Integer, String>.MapIterator iter2 = myMap.MapIterator();
System.out.println( iter2.next() );
```

5.6) TRY CATCH TO CATCH THE THROWED EXCEPTION WITH ZERO CONSTRUCTOR.

```
System.out.println("USING CONCTRUCTOR IN EMPTY MAP(TRY CATCH TO CATCH THE THROWED EXCEPTION):");
try{
    MyHashMap<Integer, String> emptyMap = new MyHashMap<Integer, String>();
    MyHashMap<Integer, String>.MapIterator iter3 = emptyMap.MapIterator();
}catch(IndexOutOfBoundsException i){
    System.out.println("IndexOutOfBoundsException caught. Please first add elements.\n");
}
```

5.7) TESTING LINKED LIST AND FIRSTLY ADDING ELEMENTS AND PERFORMANCE.

```
System.out.println("---ADDING ELEMENTS---");
long nano_start = System.nanoTime();

for(int i=0; i<10; i++)
    table_ll.put(i, "value" + i );

long nano_end = System.nanoTime();
System.out.println( table_ll.toString() );
```

5.8) GETTING ELEMENTS(EXISTING AND NON EXISTING) AND GETTING TIME PERFORMANCE.

```
System.out.println("---GETTING ELEMENTS(EXISTING KEY)---");
nano_start = System.nanoTime();
System.out.println( table_ll.get(8) );
System.out.println( table_ll.get(5) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );

System.out.println("---GETTING ELEMENTS(NON-EXISTING KEY)---");
nano_start = System.nanoTime();
System.out.println( table_ll.get(15) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

5.9) TRYING REMOVE ELEMENTS (EXISTING AND NON EXISTING) AND GETTING TIME PERFORMANCE.

```
System.out.println("---REMOVING ELEMENT(EXISTING DATA)---");
nano_start = System.nanoTime();
System.out.println( table_ll.remove(5) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );

System.out.println("---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---");
nano_start = System.nanoTime();
System.out.println( table_ll.remove(15) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

Note: ALL OF THESE OPERATIONS MADE FOR CHAINLINKEDLIST FOR:
SMALL DATA(10 ELEMENTS)
MEDIUM DATA(500 ELEMENTS)
LARGE DATA(10000 ELEMENTS)

5.10) TESTING TREESET AND FIRSTLY ADDING ELEMENTS AND TESTING PERFORMANCE.

```
System.out.println("\n-------TREESET TEST--------");
HashTableChainwithTreeSet<Integer, String>    table_ts = new HashTableChainwithTreeSet<Integer,String>();
```

```
System.out.println("---ADDING ELEMENTS---");
long nano_start = System.nanoTime();

for(int i=0; i<10; i++)
    table_ts.put(i, "value" + i );

long nano_end = System.nanoTime();
System.out.println( table_ts.toString() );

System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

5.11) GETTING ELEMENTS(EXISTING AND NON EXISTING) AND GETTING TIME PERFORMANCE.

```
System.out.println("---GETTING ELEMENTS(EXISTING KEY)---");
nano_start = System.nanoTime();
System.out.println( table_ts.get(3) );
System.out.println( table_ts.get(5) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );

System.out.println("---GETTING ELEMENTS(NON-EXISTING KEY)---");
nano_start = System.nanoTime();
System.out.println( table_ts.get(15) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

5.12) TRYING REMOVE ELEMENTS (EXISTING AND NON EXISTING) AND GETTING TIME PERFORMANCE.

```
System.out.println("---REMOVING ELEMENT(EXISTING DATA)---");
nano_start = System.nanoTime();
System.out.println( table_ts.remove(5) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );

System.out.println("---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---");
nano_start = System.nanoTime();
System.out.println( table_ts.remove(15) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

Note: ALL OF THESE OPERATIONS MADE FOR CHAIN TREE SET FOR:
SMALL DATA(10 ELEMENTS)
MEDIUM DATA(500 ELEMENTS)
LARGE DATA(10000 ELEMENTS)

5.13) TESTING COALESCED HASING CLASS AND FIRSTLY ADDING ELEMENTS AND TESTING PERFORMANCE.

```
System.out.println("\n-------HASHTABLE COALESCED HASHING CLASS TEST--------");
HashTableCoalescedHashing<Integer, String>    table_ch = new HashTableCoalescedHashing<Integer,String>();
```

```
System.out.println("---ADDING ELEMENTS---");
long nano_start = System.nanoTime();
table_ch.put(3, "valueOf3");
table_ch.put(12, "valueOf12");
table_ch.put(13, "valueOf13");
table_ch.put(25, "valueOf25");
table_ch.put(23, "valueOf23");
table_ch.put(51, "valueOf51");
table_ch.put(42, "valueOf42");
long nano_end = System.nanoTime();

System.out.println("---PRINTINGG ELEMENTS---\n(Note: If next=-1 it means null)");
System.out.println( table_ch.toString() );

System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

5.14) GETTING ELEMENTS(EXISTING AND NON EXISTING) AND GETTING TIME PERFORMANCE.

```
System.out.println("---GETTING ELEMENTS(EXISTING KEY)---");
nano_start = System.nanoTime();
System.out.println( table_ch.get(12) );
System.out.println( table_ch.get(13) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );

System.out.println("---GETTING ELEMENTS(NON-EXISTING KEY)---");
nano_start = System.nanoTime();
System.out.println( table_ch.get(49) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

5.15) TRYING REMOVE ELEMENTS (EXISTING AND NON EXISTING) AND GETTING TIME PERFORMANCE.

```
System.out.println("---REMOVING ELEMENT(EXISTING DATA: Key13)---");
nano_start = System.nanoTime();
System.out.println( table_ch.remove(13) );
System.out.println( table_ch.toString() );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );

System.out.println("---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---");
nano_start = System.nanoTime();
System.out.println( table_ch.remove(49) );
nano_end = System.nanoTime();
System.out.println("Performance time-> " + (nano_end-nano_start) + " nanosconds." );
```

Note: ALL OF THESE OPERATIONS MADE FOR COALESCED HASHING:
SMALL DATA(10 ELEMENTS)
MEDIUM DATA(500 ELEMENTS)
LARGE DATA(10000 ELEMENTS)

6. RUNNING AND RESULTS

Creating map and adding elements to it and printing.

```
sglbl@SglblPC:/mnt/c/Araclar/GTU/2.Sınıf/2.Dönem/Cse222/hw5$ make
javac -d classfiles *.java
java -cp classfiles Main
--------------PART1--------------
Map<Integer,String> is:
{1801042601=Baran Solmaz, 1801042634=Muhammed Oguz, 1801042656=Suleyman Golbol, 1801042647=Kadir Saglam}
```

Using one parameter constructor and using prev() hasNext() and next() methods.

```
USING ONE PARAMETER CONSTRUCTOR:
USING PREV() METHOD:
1801042634
1801042601
Muhammed Oguz
Suleyman Golbol
USING HASNEXT() AND NEXT() METHOD:
1801042634
1801042656
1801042647
```

Using zero parameter constructor in same map and printing next item.

```
USING ZERO PARAMETER CONSTRUCTOR:
1801042601
```

Using zero parameter constructor in empty map and catching error.

```
 USING CONCTRUCTOR IN EMPTY MAP(TRY CATCH TO CATCH THE THROWED EXCEPTION):
 IndexOutOfBoundsException caught. Please first add elements.
```

For hash table with linked list adding elements to it.

```
---------------PART2--------------
-------LINKEDLIST TEST--------
-----FOR SMALL DATA-----
---ADDING ELEMENTS---
[
0=>value0,
1=>value1,
2=>value2,
3=>value3,
4=>value4,
5=>value5,
6=>value6,
7=>value7,
8=>value8,
9=>value9,
 ]
```

Getting performance of it.

```
Performance time-> 45760000 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for small data.

```
---GETTING ELEMENTS(EXISTING KEY)---
value8
value5
Performance time-> 427000 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 224800 nanosconds.
---REMOVING ELEMENT(EXISTING DATA)---
value5
Performance time-> 618900 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 225800 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 206900 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for medium data.

```
-----FOR MEDIUM DATA-----
---ADDING ELEMENTS---
Performance time-> 8537800 nanosconds.
---GETTING ELEMENTS(EXISTING KEY)---
value243
value280
Performance time-> 393400 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 236800 nanosconds.
---REMOVING ELEMENT(EXISTING DATA)---
value280
Performance time-> 184400 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 160500 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 271000 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for large data.

```
-----FOR LARGE DATA-----
---ADDING ELEMENTS---
Performance time-> 30780300 nanosconds.
---GETTING ELEMENTS(EXISTING KEY)---
value8702
value1402
Performance time-> 352600 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 179700 nanosconds.
---REMOVING ELEMENT(EXISTING DATA)---
value1402
Performance time-> 163600 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 56100 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 190400 nanosconds.
```

For HashTableChainWithTreeSet class:
Adding elements and getting performance for small data.

```
-------TREESET TEST--------
-----FOR SMALL DATA-----
---ADDING ELEMENTS---
[
0=>value0,
1=>value1,
2=>value2,
3=>value3,
4=>value4,
5=>value5,
6=>value6,
7=>value7,
8=>value8,
9=>value9,
 ]
Performance time-> 13392700 nanosconds.
```

Try to get elements(exist and non-exist) and removing element for small data.

```
---GETTING ELEMENTS(EXISTING KEY)---
value3
value5
Performance time-> 772500 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 198100 nanosconds.
---REMOVING ELEMENT(EXISTING DATA)---
value5
Performance time-> 231800 nanosconds.
```

Try to remove non-existing element and try to find removed element for small data.

```
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 114100 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 228800 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for medium data.

```
-----FOR MEDIUM DATA-----
---ADDING ELEMENTS---
Performance time-> 1685700 nanosconds.
---GETTING ELEMENTS(EXISTING KEY)---
value211
value280
Performance time-> 388000 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 190400 nanosconds.
---REMOVING ELEMENT(EXISTING DATA)---
value280
Performance time-> 240700 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 197500 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 171200 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for large data.

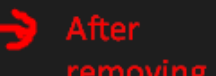
```
-----FOR LARGE DATA-----
---ADDING ELEMENTS---
Performance time-> 16408400 nanosconds.
---GETTING ELEMENTS(EXISTING KEY)---
value5456
value1402
Performance time-> 738300 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 144700 nanosconds.
---REMOVING ELEMENT(EXISTING DATA)---
value1402
Performance time-> 142700 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 138700 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 206500 nanosconds.
```

FOR HASHTABLE COALESCED HASHING CLASS TEST:
Adding elements and getting performance of it.

```
-------HASHTABLE COALESCED HASHING CLASS TEST--------
-----FOR SMALL DATA-----
---ADDING ELEMENTS---
---PRINTINGG ELEMENTS---
(Note: If next=-1 it means null)
[
HashIndex is 0 key is: null next:-1
HashIndex is 1 key is: 51 next:-1
HashIndex is 2 key is: 12 next:6
HashIndex is 3 key is: 3 next:4
HashIndex is 4 key is: 13 next:7
HashIndex is 5 key is: 25 next:-1
HashIndex is 6 key is: 42 next:-1
HashIndex is 7 key is: 23 next:-1
HashIndex is 8 key is: null next:-1
HashIndex is 9 key is: null next:-1
 ]
Performance time-> 114900 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element.

```
---GETTING ELEMENTS(EXISTING KEY)---
valueOf12
valueOf13
Performance time-> 1397600 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 119300 nanosconds.
---REMOVING ELEMENT(EXISTING DATA: Key13)---
valueOf13
[
HashIndex is 0 key is: null next:-1
HashIndex is 1 key is: 51 next:-1
HashIndex is 2 key is: 12 next:6
HashIndex is 3 key is: 3 next:4
HashIndex is 4 key is: 23 next:-1
HashIndex is 5 key is: 25 next:-1
HashIndex is 6 key is: 42 next:-1
HashIndex is 7 key is: null next:-1
HashIndex is 8 key is: null next:-1
HashIndex is 9 key is: null next:-1
 ]
Performance time-> 727500 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 218100 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 77900 nanosconds.
```



Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for medium data.

```
-----FOR MEDIUM DATA-----
---ADDING ELEMENTS---
Performance time-> 2680000 nanosconds.
---GETTING ELEMENTS(EXISTING KEY)---
value211
value280
Performance time-> 205500 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 61000 nanosconds.
---REMOVING ELEMENT(EXISTING DATA: Key280)---
value280
Performance time-> 67900 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 214300 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 253600 nanosconds.
```

Try to get elements(exist and non-exist) and try to remove elements(exist and non-exist) and try to find removed element for large data.

```
-----FOR LARGE DATA-----
---ADDING ELEMENTS---
Performance time-> 10097200 nanosconds.
---GETTING ELEMENTS(EXISTING KEY)---
value8764
value1402
Performance time-> 574000 nanosconds.
---GETTING ELEMENTS(NON-EXISTING KEY)---
null
Performance time-> 216900 nanosconds.
---REMOVING ELEMENT(EXISTING DATA: Key1402)---
value1402
Performance time-> 232500 nanosconds.
---TRYING REMOVE ELEMENT(NON-EXISTING DATA)---
null
Performance time-> 153300 nanosconds.
---SEARCHING FOR REMOVED ELEMENT(SO NON-EXISTING DATA)---
null
Performance time-> 83600 nanosconds.
sglbl@SglblPC:/mnt/c/Araclar/GTU/2.Sınıf/2.Dönem/Cse222/hw5$
```