**GIT Department of Computer Engineering CSE 222/505 - Spring 2021 Homework 5 # Report**

**Yeşim Yalçın**

**200104004094**

**1-) System Requirements**

**-**The system is written using Java jdk 15.0.1

-You can use the makefile then use java DriverCode command to run the program.

**2-) Problem Solution Aproach**

-Firstly I have implemented the KWHashMap interface.

-For the ChainHashMap class that uses chaining technique with LinkedList, I have rewritten the code in our book and added some things.

-For the ChainHashMap2 class that uses chaining technique with TreeSet, I have used a similar way I used in ChainHashMap while implementing. I have made the Entry class inside comparable and used TreeSet iterator to reach&modify the elements as a difference.

-For the CoalescedHashMap class that uses open addressing with quadratic probing, I have added new data fields. I have added numDeletes data field to store the entries which beomes empty. During removal, it is possible to swap entries if they are connected however in the end still one entry needs to become null. If the entries become null, the order is broken therefore I put a dummy entry DELETED inside them and store their amount inside numDeletes. While rehashing I do not reinsert DELETED items again.

I have made the LOAD\_TRESHOLD modifyable in the constructor. I have done that only to be able to demonstrate the example that is given in homework pds. Otherwise I have designed this class in a way that it will always have a prime number capacity and it’s treshold will never be greater than 0.48. I have done it like that to prevent infinite loops.

-I have added getCapacity() and printMap() methods to be able to test the codes in a more detailed way. I have not used printArray method many times in the driver code not to make the output so long. I used it only for the small sized tables.

-For the MapIterator class, I have chose to create a subclass of HashMap and created an IterableHashMap class. MapIterator class uses the iterators of set to do it’s operations.

**3-) Test Cases and Running Results**

-I have tested each class with 4 different sized tables.

-Small sized: 7 elements

-Small-Medium sized: 75 elements

-Medium sized: 1000 elements

-Large sized: 10000 elements

During my tests I firstly used put methods for the elements, used remove method with 3, 6, 300, 3000 elements, used get method for all elements after both put and remove methods. The only difference is I have also used printMap method for the small sized tables. Note that put, get, remove methods are used for both existing and non existing elements.

Here is the performance results of all classes. The results are the average of 5 different runtimes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Map Types** | **Small-Medium Table**  **(7-75 Elements)** | **Medium Table**  **(1000 Elements)** | **Large Table**  **(10000 Elements)** |
| **ChainHashMap1** | 2,16326 ms | 7,71826 ms | 78,42042 ms |
| **ChainHashMap2** | 2,7544 ms | 10,0854 ms | 79,91082 ms |
| **CoalescedHashMap** | 2,10294 ms | 7,09758 ms | 74,0597 ms |

**1-) Driver Code Tests:**

A-) ChainHashMap Driver Code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **1** | Creating an empty map and adding 3 elements | Successfuly adding | The same | Pass |
| **2** | Removing 3 elements | Successfuly removing | The same | Pass |
| **3** | Adding more elements (total 75) | Successfuly adding | The same | Pass |
| **4** | Removing 6 elements | Successfuly removing | The same | Pass |
| **5** | Creating an empty map and adding 1000 elements | Successfuly adding | The same | Pass |
| **6** | Removing 300 elements | Successfuly removing | The same | Pass |
| **7** | Creating an empty map and adding 10000 elements | Successfuly adding | The same | Pass |
| **8** | Removing  3000 elements | Successfuly removing | The same | Pass |

B-) ChainHashMap2 Driver Code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **9** | Creating an empty map and adding 3 elements | Successfuly adding | The same | Pass |
| **10** | Removing 3 elements | Successfuly removing | The same | Pass |
| **11** | Adding more elements (total 75) | Successfuly adding | The same | Pass |
| **12** | Removing 6 elements | Successfuly removing | The same | Pass |
| **13** | Creating an empty map and adding 1000 elements | Successfuly adding | The same | Pass |
| **14** | Removing 300 elements | Successfuly removing | The same | Pass |
| **15** | Creating an empty map and adding 10000 elements | Successfuly adding | The same | Pass |
| **16** | Removing  3000 elements | Successfuly removing | The same | Pass |

C-) CoalescedHashMap Driver Code

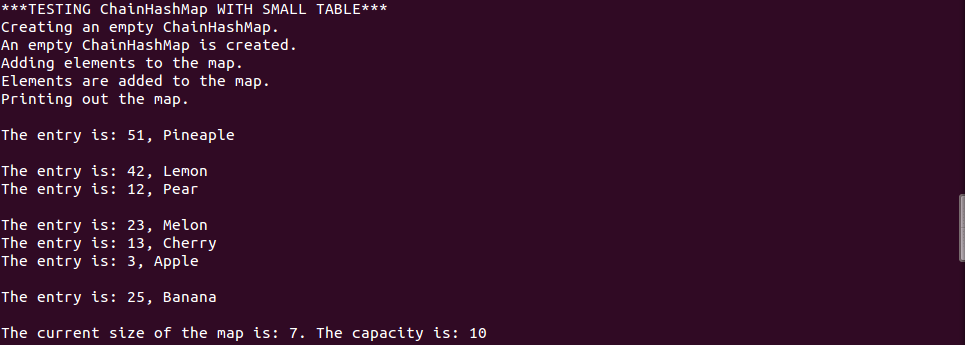
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **17** | Creating an empty map and adding 3 elements | Successfuly adding | The same | Pass |
| **18** | Removing 3 elements | Successfuly removing | The same | Pass |
| **19** | Adding more elements (total 75) | Successfuly adding | The same | Pass |
| **20** | Removing 6 elements | Successfuly removing | The same | Pass |
| **21** | Creating an empty map and adding 1000 elements | Successfuly adding | The same | Pass |
| **22** | Removing 300 elements | Successfuly removing | The same | Pass |
| **23** | Creating an empty map and adding 10000 elements | Successfuly adding | The same | Pass |
| **24** | Removing  3000 elements | Successfuly removing | The same | Pass |

D-) MapIterator Driver Code

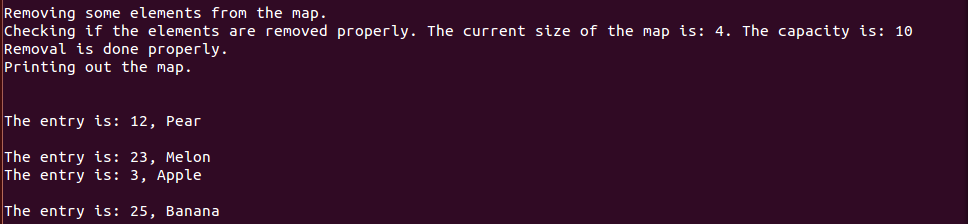
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **25** | Creating an IterableHashMap with 16 elements, iterating over it with next() from the start | Printing out all the elements in order | The same | Pass |
| **26** | Iterating over the map with next() from the middle | Printing out all the elements in order starting from the given key | The same | Pass |
| **27** | Iterating over the map with previous() from the start in | Printing out all the elements in reverse order | The same | Pass |
| **28** | Iterating over the map with previous() from the middle | Printing out all the elements in reverse order starting from the given key | The same | Pass |

**2-) The Results**

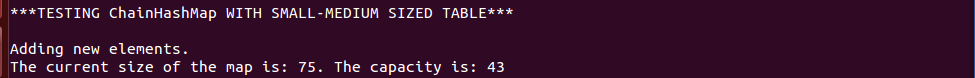
1-) Creating an empty ChainHashMap and adding 7 elements inside it.

****

2-) Removing 3 elements and checking if they are removed properly by using get method.



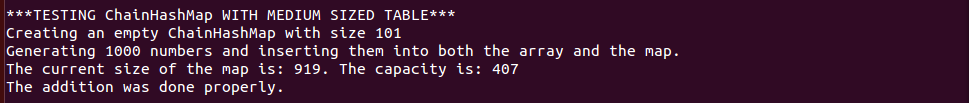
3-) Adding more elements inside the map (total 75).



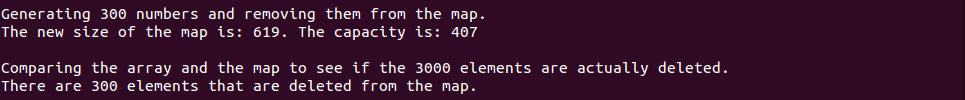
4-) Removing 6 elements from the map and checking if they are removed properly by using get method.



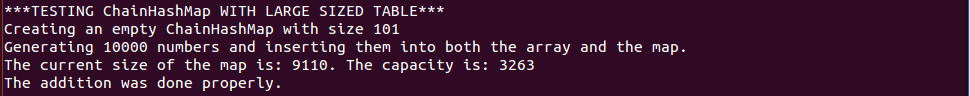
5-) Creating an empty ChainHashMap with size 101 and adding 1000 elements inside it. Using get method for all elements to check if properly added.



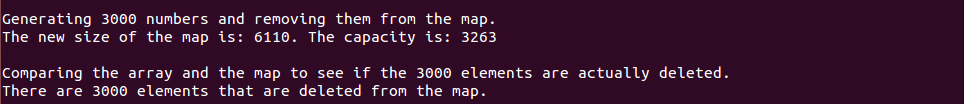
6-) Removing 300 elements and using get method for all removed elements to see if exactly 300 elements are removed.



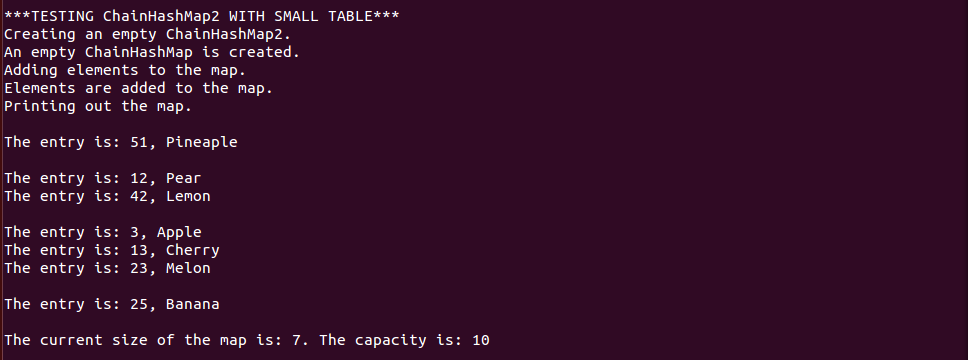
7-) Creating an empty ChainHashMap with size 101 and adding 10000 elements inside it. Using get method for all elements to check if properly added.



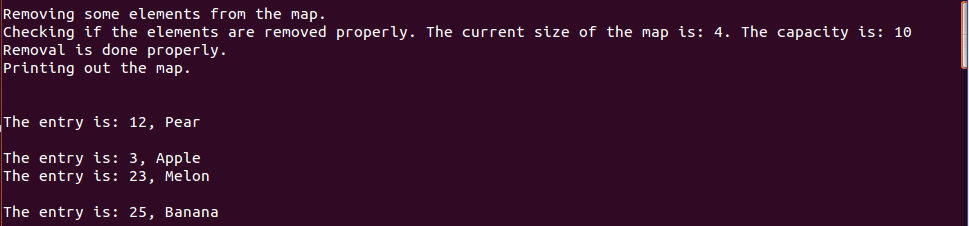
8-) Removing 3000 elements and using get method for all removed elements to see if exactly 3000 elements are removed.



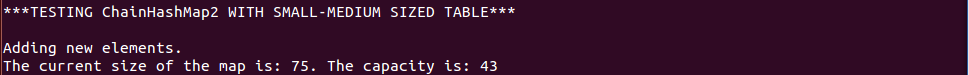
9-) Creating an empty ChainHashMap2 and adding 7 elements inside it.



10-) Removing 3 elements and checking if they are removed properly by using get method.



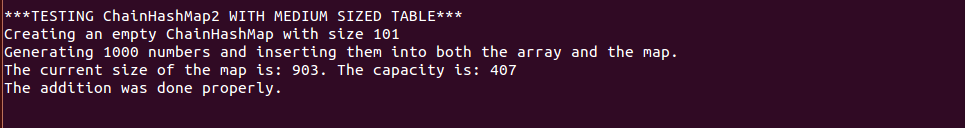
11-) Adding more elements inside the map (total 75).



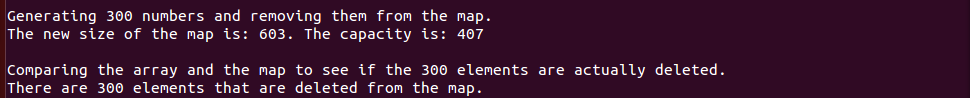
12-) Removing 6 elements from the map and checking if they are removed properly by using get method.



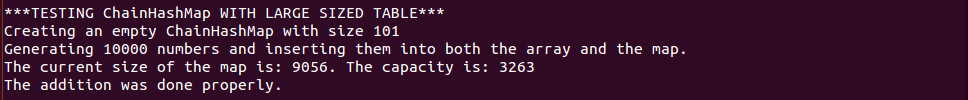
13-) Creating an empty ChainHashMap2 with size 101 and adding 1000 elements inside it. Using get method for all elements to check if properly added.



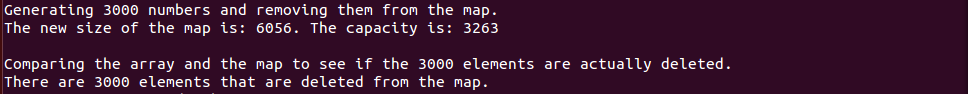
14-) Removing 300 elements and using get method for all removed elements to see if exactly 300 elements are removed.



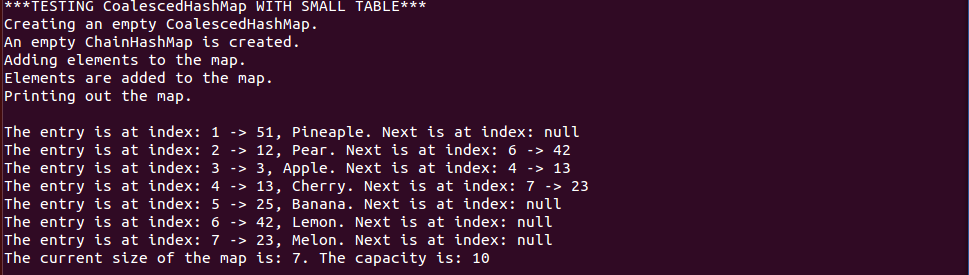
15-) Creating an empty ChainHashMap2 with size 101 and adding 10000 elements inside it. Using get method for all elements to check if properly added.



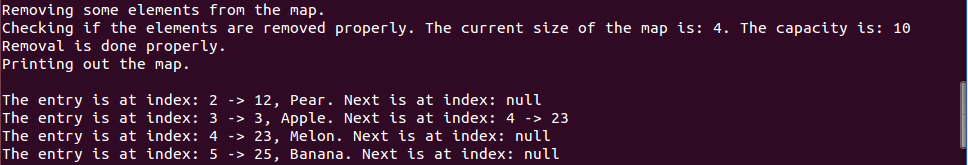
16-) Removing 3000 elements and using get method for all removed elements to see if exactly 3000 elements are removed.



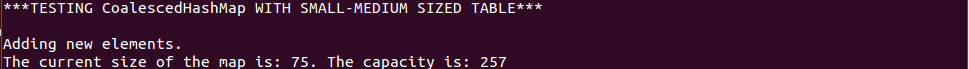
17-) Creating an empty CoalescedHashMap and adding 7 elements inside it



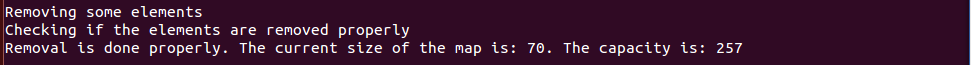
18-) Removing 3 elements and checking if they are removed properly by using get method.



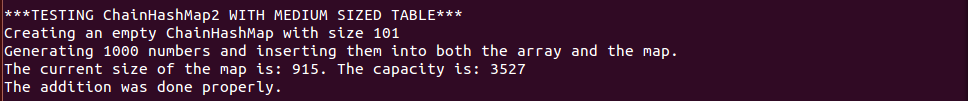
19-) Adding more elements inside the map (total 75).



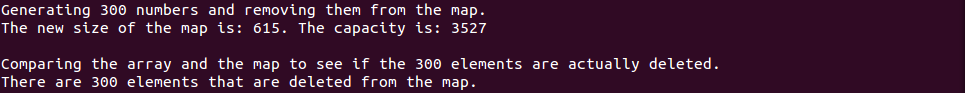
20-) Removing 6 elements from the map and checking if they are removed properly by using get method.



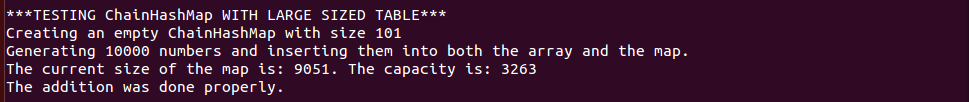
21-) Creating an empty CoalescedHashMap with size 101 and adding 1000 elements inside it. Using get method for all elements to check if properly added.



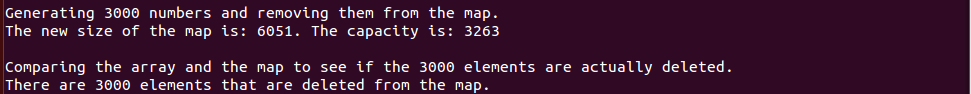
22-) Removing 300 elements and using get method for all removed elements to see if exactly 300 elements are removed.



23-) Creating an empty CoalescedHashMap with size 101 and adding 10000 elements inside it. Using get method for all elements to check if properly added.



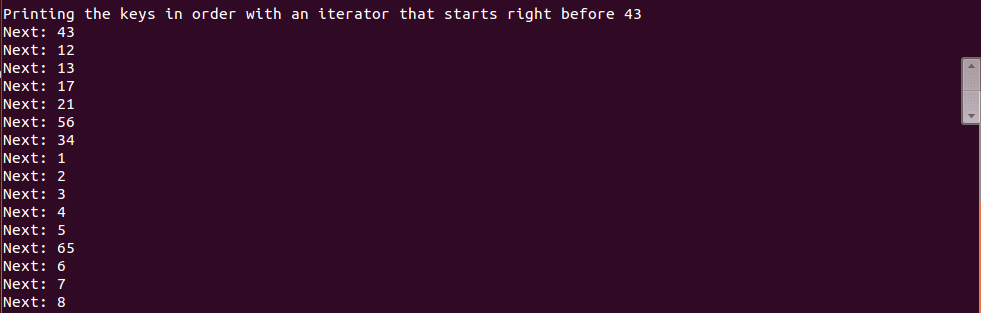
24-) Removing 3000 elements and using get method for all removed elements to see if exactly 3000 elements are removed.



25-) Creating an IterableHashMap with 16 elements. Creating an iterator with no parameter constructor and printing all the elements with next() method.



26-) Creating an iterator with 1 parameter constructor and printing all the elements with next() method.



27-) Creating an iterator with no parameter constructor and printing all the elements with previous() method.



28-) Creating an iterator with 1 parameter constructor and printing all elements with previous() method.

