GTU Department of Computer Engineering

CSE 222/505 - Spring 2021 Homework 5

Report

Muhammet Fikret ATAR

1801042693

Problem solution approach;

Part1:

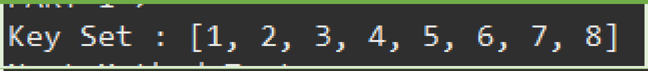
In this part ,I wrote a custom iterator class Map Iterator to iterate through the keys in a HashMap data structure in Java. I first gathered my hashmap keys(I used the keySet() method to access the key sets in the class I created.) in an array so that functions can be used. Thanks to this array, I would be able to iterate through the keys. I also created 2 different types of constructors. One of them is the zero parameter constructor, this constructor provides the iterator to start from any random key. I implement my other methods over the random key that occurs. In the other constructer, the initial key is given and my methods start the action on the given key.Apart from these, the hasnext method is in the next method. I did not need to run a test for the hasnext in the driver class. Hasnext works properly in the next method.

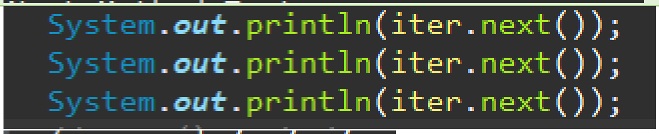
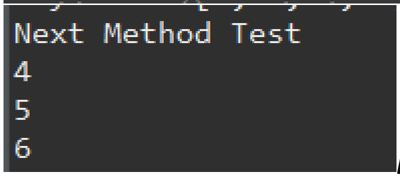
Test Cases:

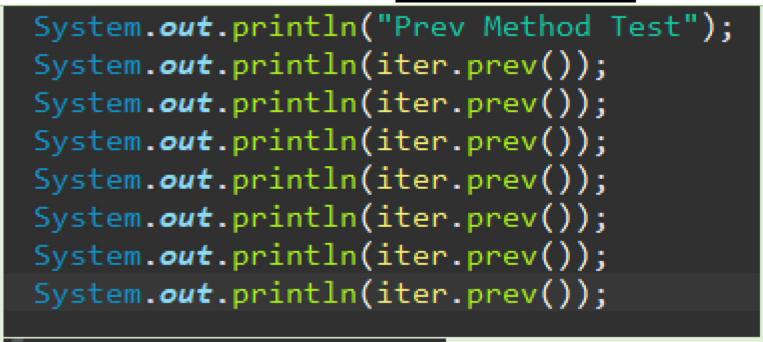
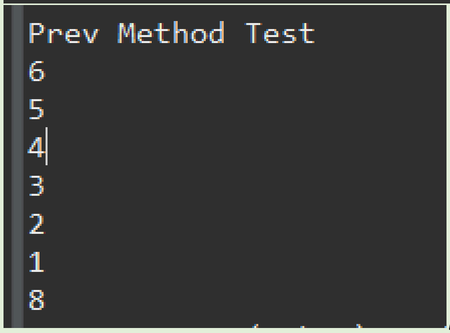
Part1:

Test ID	Test Cases	Test Steps	Expected Result	Actual Results	Pass /Fail
T1	Fill hashmap	Call Put Method	Fill key and values	As expected	***PASS***
T2	next method	Call the method	Returns the next key	As expected	***PASS***
T3	prev method	Call the method	Points to the previous key	As expected	***PASS***
T4	hasNext method	Call the method	Returns True if there are still not-iterated keys in next method	As expected	***PASS***
T5	MapIterator (K key)	Call the method with proper parameter	Iterator should start from the given key	As expected	***PASS***
T6	MapIterator ()	Call the method	Iterator should start from the any key	As expected	***PASS***

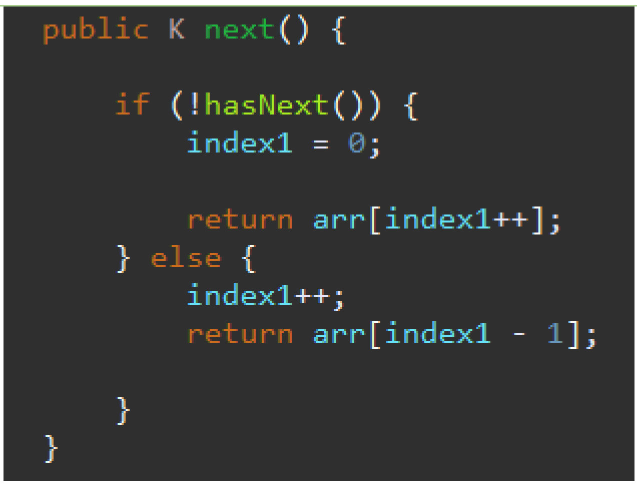
Running And Result For Part1:

Test	*Result*
T1	 `Key Set : [1, 2, 3, 4, 5, 6, 7, 8]`

T2	 `System.out.println(iter.next());`
	`System.out.println(iter.next());`
	`System.out.println(iter.next());`
	 `Next Method Test`
	`4`
	`5`
	`6`
	Begin key random

T3	 `System.out.println("Prev Method Test");`
	`System.out.println(iter.prev());`
	`System.out.println(iter.prev());`
	`System.out.println(iter.prev());`
	`System.out.println(iter.prev());`
	`System.out.println(iter.prev());`
	`System.out.println(iter.prev());`
	`System.out.println(iter.prev());`
	 `Prev Method Test`
	`6`
	`5`
	`4`
	`3`
	`2`
	`1`
	`8`
	Begin key random

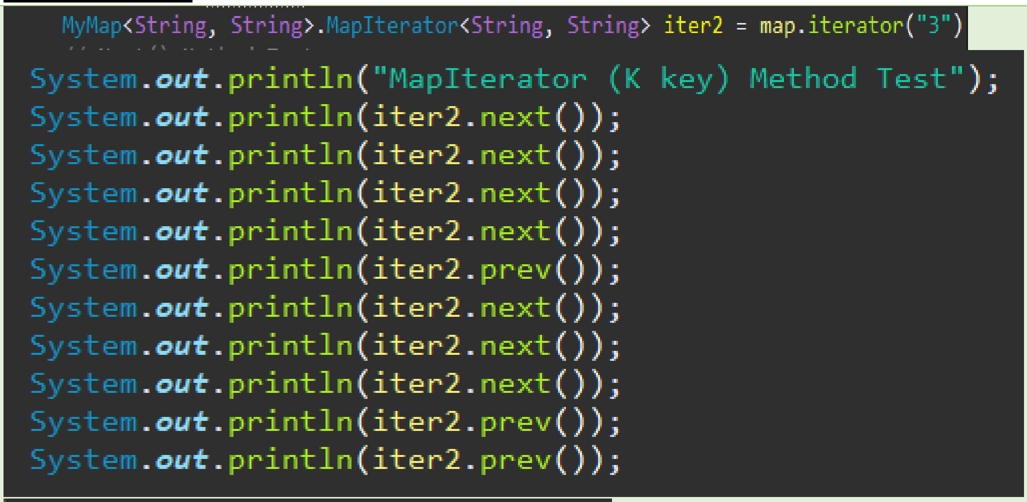
__T4__



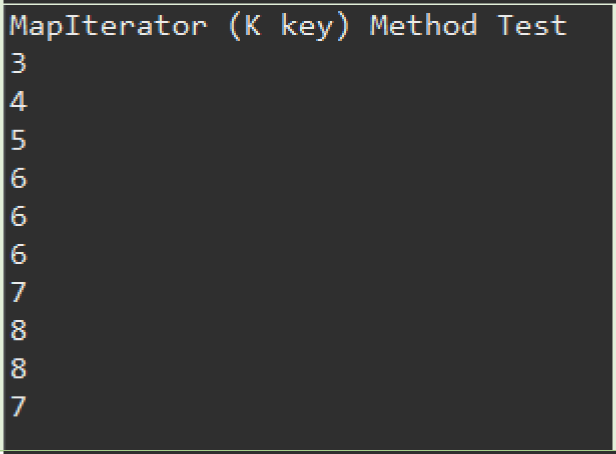
```
public K next() {

    if (!hasNext()) {
        index1 = 0;

        return arr[index1++];
    } else {
        index1++;
        return arr[index1 - 1];

    }
}
```

__HasNext method is running in next() method__

__T5__

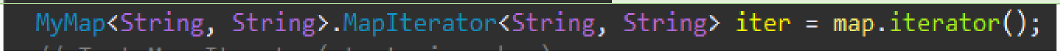


```
MyMap<String, String>.MapIterator<String, String> iter2 = map.iterator("3")
System.out.println("MapIterator (K key) Method Test");
System.out.println(iter2.next());
System.out.println(iter2.next());
System.out.println(iter2.next());
System.out.println(iter2.next());
System.out.println(iter2.prev());
System.out.println(iter2.next());
System.out.println(iter2.next());
System.out.println(iter2.next());
System.out.println(iter2.prev());
System.out.println(iter2.prev());
```

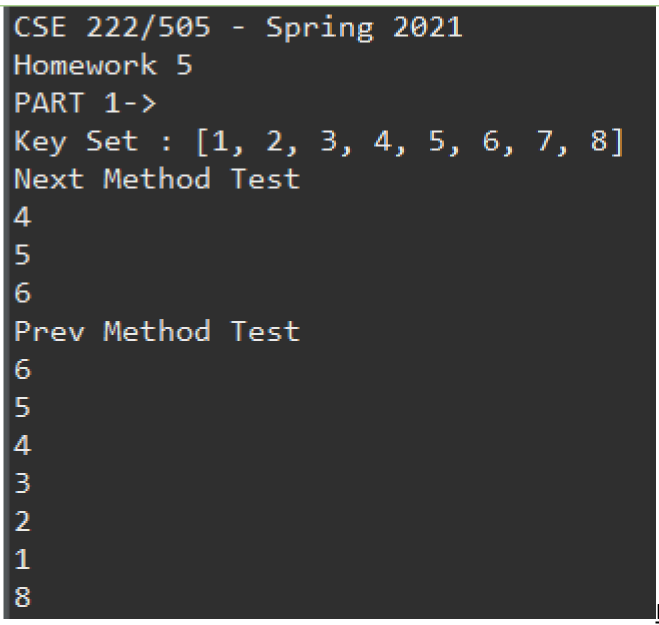


```
MapIterator (K key) Method Test
3
4
5
6
6
6
7
8
8
7
```

__T6__



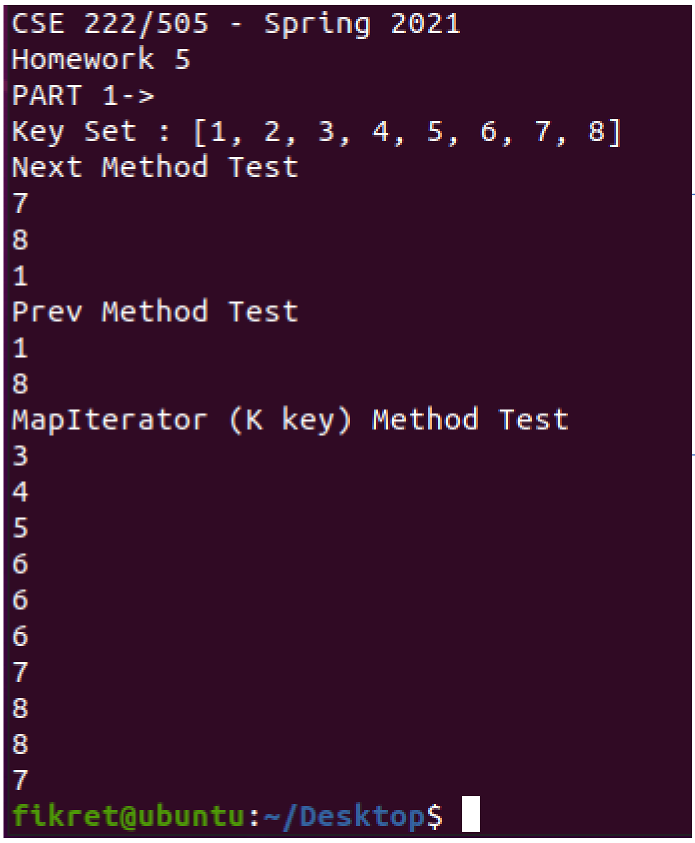
```
MyMap<String, String>.MapIterator<String, String> iter = map.iterator();
```



```
CSE 222/505 - Spring 2021
Homework 5
PART 1->
Key Set : [1, 2, 3, 4, 5, 6, 7, 8]
Next Method Test
4
5
6
Prev Method Test
6
5
4
3
2
1
8
```

Key created randomly by MapIterator ()

Driver Output



```
CSE 222/505 - Spring 2021
Homework 5
PART 1->
Key Set : [1, 2, 3, 4, 5, 6, 7, 8]
Next Method Test
7
8
1
Prev Method Test
1
8
MapIterator (K key) Method Test
3
4
5
6
6
6
7
8
8
7
fikret@ubuntu:~/Desktop$
```

Initial number 7 create randomly

Initial number 3 given by user

Problem solution approach;

Part2:

Problem is implementing KWHashMap interface using the chaining technique for hashing by using linked lists, using the chaining technique for hashing by using TreeSet and using the Coalesced hashing technique.

My approach to this problem was as follows;

I created HashtableChainLinkedList and HashtableChainTreeSet classes for this part. For all these three classes, I hold entry class as a inner class.For these hashing methods ,i implemented KWHashMap interface in the book to organize hash table. In addition to the book I added the rehash and remove methods to classes. Remove method checks elements of the table one by one, and once it finds a match, removes it from the index and returns that removed value.Rehash method created a hash table that is bigger than the old hash table and, it puts every element of the old has table to the new hash table.

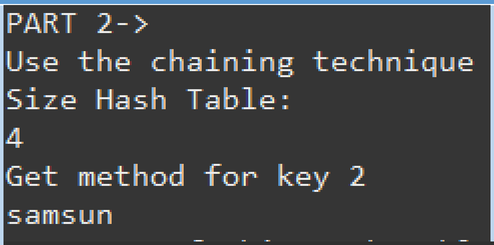
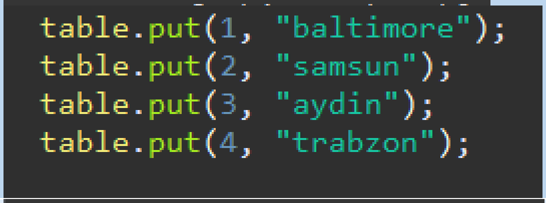
TestCases:

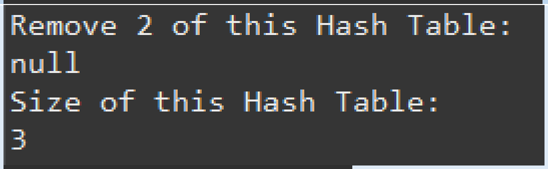
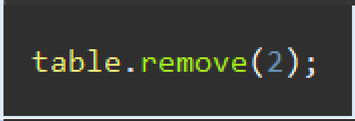
Test ID	Test Cases	Test Steps	Expected Result	Actual Results	Pass /Fail
T1	Fill hashmap	Call Put Method	Fill key and values	As expected	***PASS***
T2	Remove() method	Call the method	Remove keys	As expected	***PASS***
T3	Get method	Call the method	Get value	As expected	***PASS***
T4	Size method	Call the method	Return size	As expected	***PASS***
T5	Rehash method	Call the method with proper parameter	Created a hash table	As expected	***PASS***

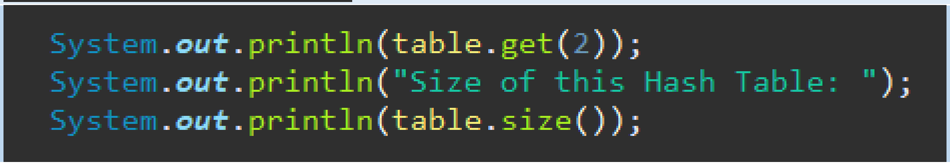
Running And Result;

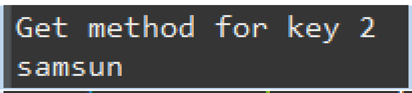
Hashing by using linked lists and by using TreeSet*:*

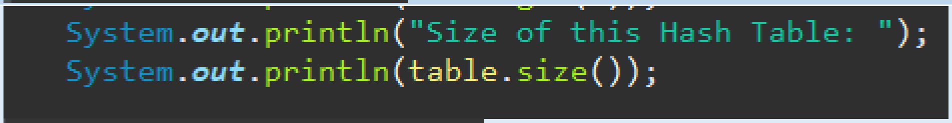
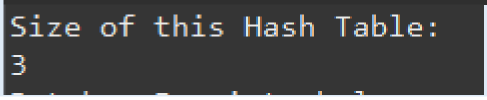
Test	***Result***

T1	 `PART 2->`
	`Use the chaining technique`
	`Size Hash Table:`
	`4`
	`Get method for key 2`
	`samsun`
	 `table.put(1, "baltimore");`
	`table.put(2, "samsun");`
	`table.put(3, "aydin");`
	`table.put(4, "trabzon");`

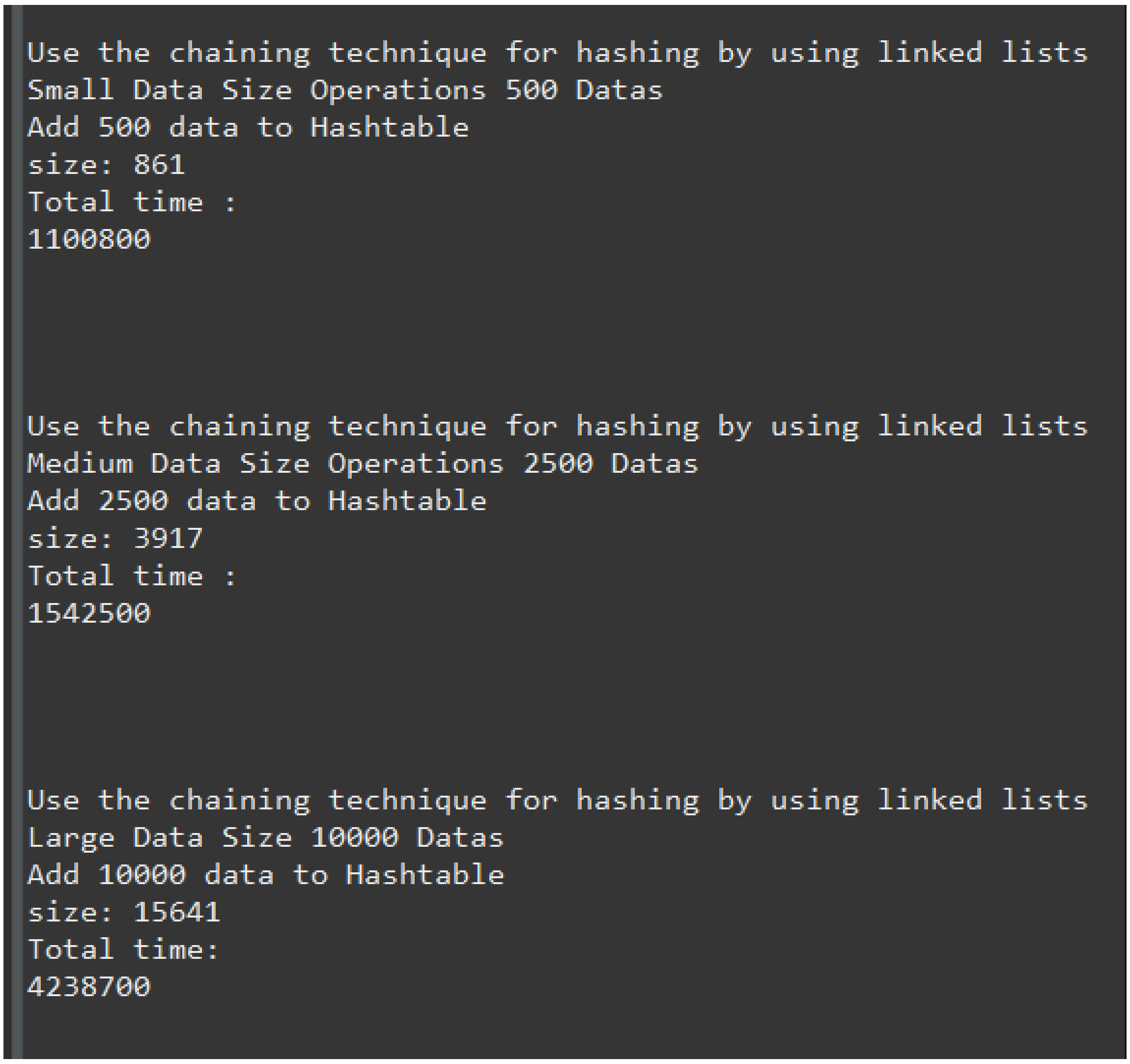
T2	 `Remove 2 of this Hash Table:`
	`null`
	`Size of this Hash Table:`
	`3`
	 `table.remove(2);`

T3	 `System.out.println(table.get(2));`
	`System.out.println("Size of this Hash Table: ");`
	`System.out.println(table.size());`

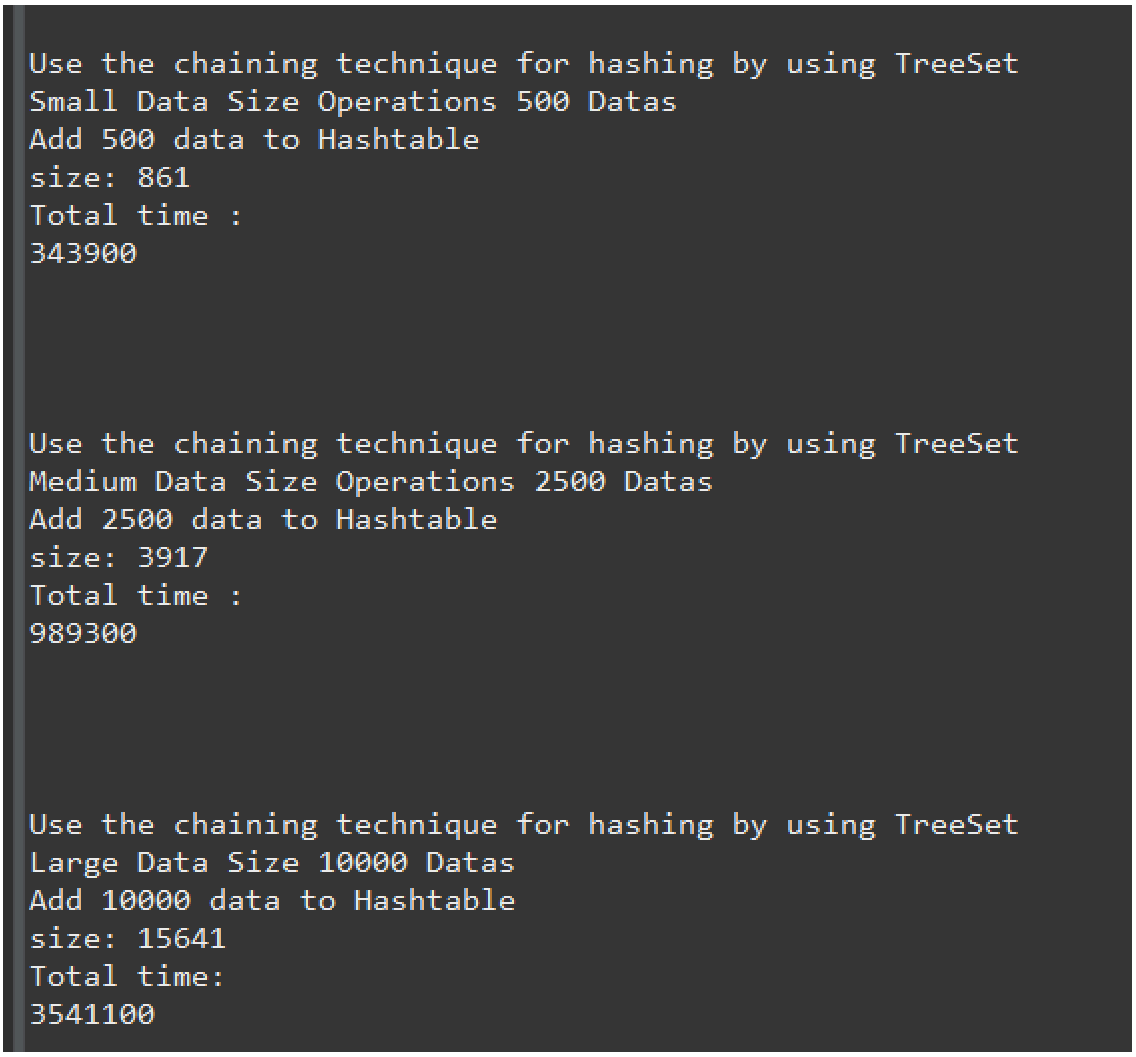
	 `Get method for key 2` `samsun`

T4	 `System.out.println("Size of this Hash Table: ");`
	`System.out.println(table.size());`
	 `Size of this Hash Table:`
	`3`

Test all the three hash table implementations empirically. Use small, medium, and large-sized data and hash tables in suitable sizes for testing;



```
Use the chaining technique for hashing by using linked lists
Small Data Size Operations 500 Datas
Add 500 data to Hashtable
size: 861
Total time :
1100800



Use the chaining technique for hashing by using linked lists
Medium Data Size Operations 2500 Datas
Add 2500 data to Hashtable
size: 3917
Total time :
1542500



Use the chaining technique for hashing by using linked lists
Large Data Size 10000 Datas
Add 10000 data to Hashtable
size: 15641
Total time:
4238700
```



```
Use the chaining technique for hashing by using TreeSet
Small Data Size Operations 500 Datas
Add 500 data to Hashtable
size: 861
Total time :
343900




Use the chaining technique for hashing by using TreeSet
Medium Data Size Operations 2500 Datas
Add 2500 data to Hashtable
size: 3917
Total time :
989300




Use the chaining technique for hashing by using TreeSet
Large Data Size 10000 Datas
Add 10000 data to Hashtable
size: 15641
Total time:
3541100
```

Note: ***I could not do the third part of the second part, so it is not in the report.***