CSE222 DATA STRUCTURES AND ALGORITHMS

HOMEWORK 5
REPORT

HATİCE SEVRA GENÇ 1801042611

1.Problem Solution Approach

- PART1

I created the SevraHashMap class in Part1. Inside this class there is an inner class called MapIterator. Required functions are implemented there. I used the mapIterator function in SevraHashMap to create an iterator.

I implemented all the functions in Part1 as specified and tested them in the driver function.

- PART2

For the first item of Part2, I used the class called ChainLinked. This class creates a linked list in collision situations. The functions of this class work fine.

I used ChainTree for item 2 of Part. A very similar structure to the class in the first item. The functions of this class work fine.

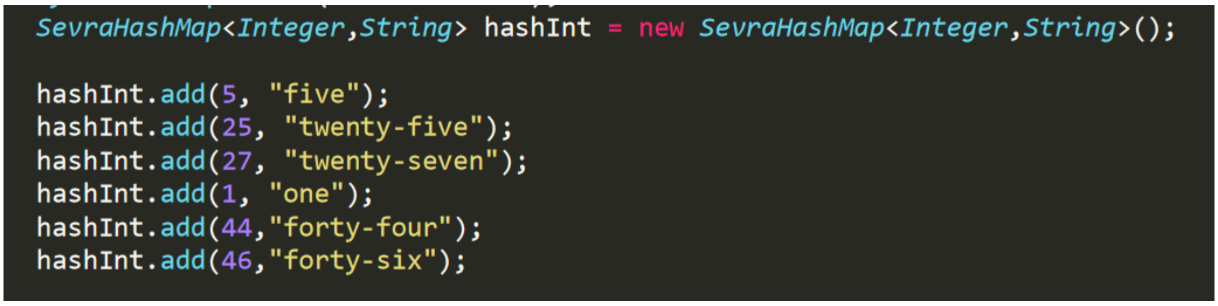
In the 3rd item, I created a class called Coalesced. This class creates a HashTable using the HashTableOpen class. In case of collision, it performs a quadratic probe transaction with the quadraticProbe() function. the next variable keeps the next index of the collision number. It works properly.

I had some problems with the performance comparison process. I did the add / remove / search operations with a small amount of data. However, when the number of data increased, I got errors.

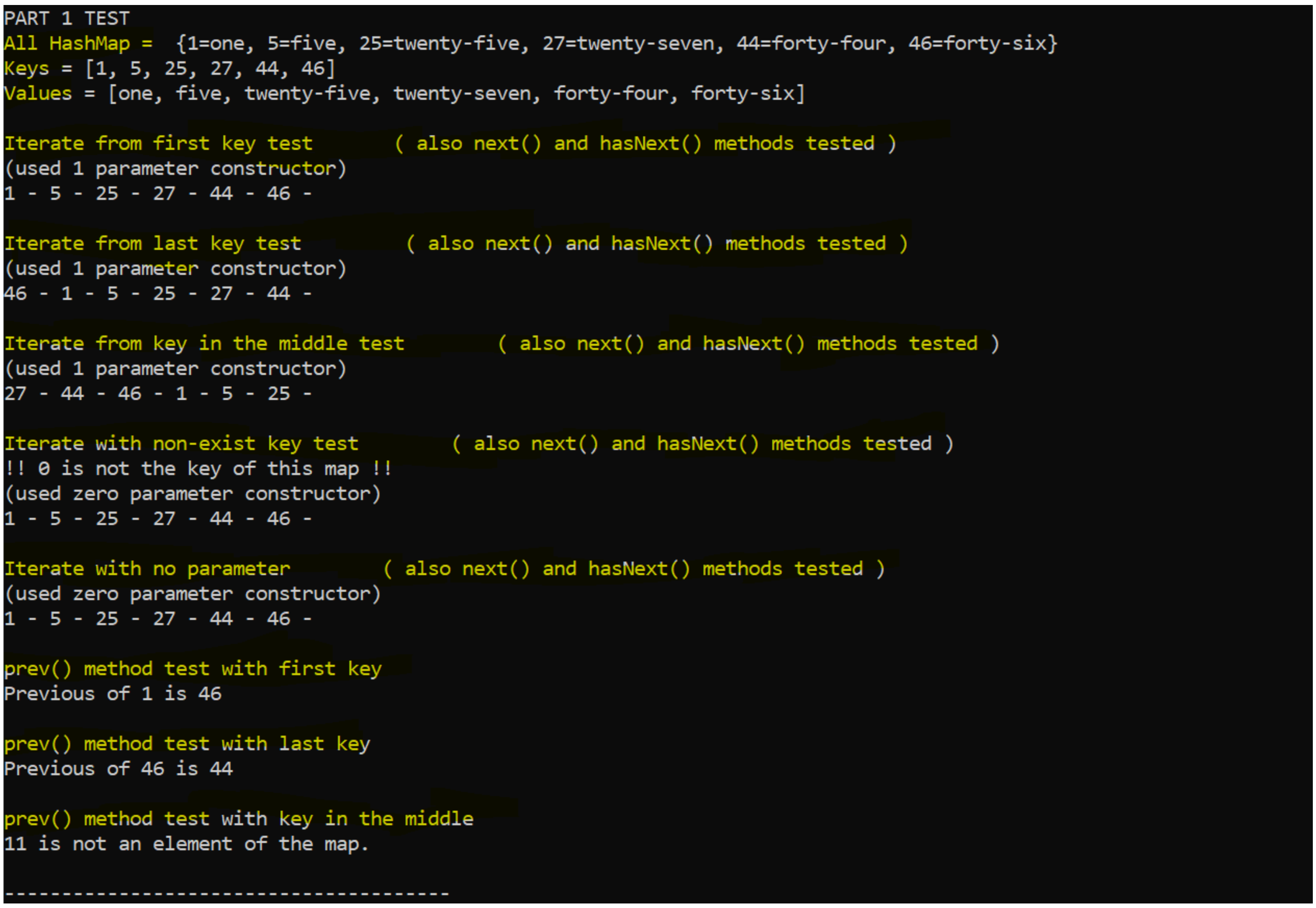
I couldn't fix this problem because I didn't have enough time :(

2.Running Command and Results

a)Part1 test HashMap

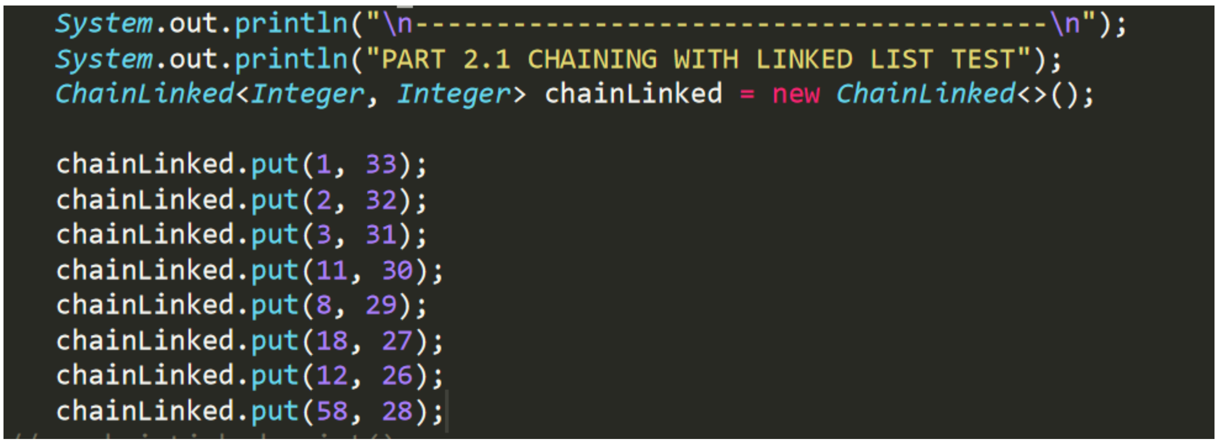


```java
SevraHashMap<Integer,String> hashInt = new SevraHashMap<Integer,String>();

hashInt.add(5, "five");
hashInt.add(25, "twenty-five");
hashInt.add(27, "twenty-seven");
hashInt.add(1, "one");
hashInt.add(44,"forty-four");
hashInt.add(46,"forty-six");
```

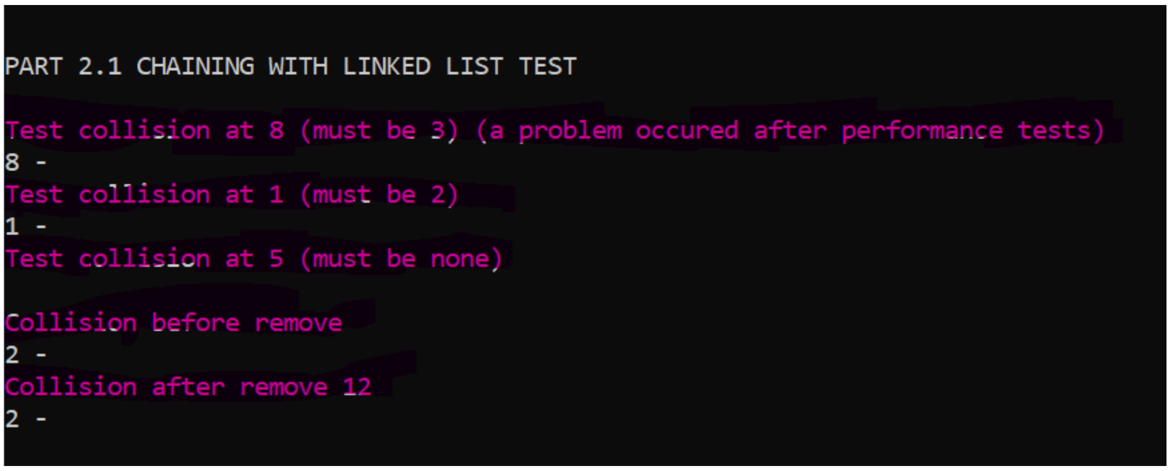


```
PART 1 TEST
All HashMap =  {1=one, 5=five, 25=twenty-five, 27=twenty-seven, 44=forty-four, 46=forty-six}
Keys = [1, 5, 25, 27, 44, 46]
Values = [one, five, twenty-five, twenty-seven, forty-four, forty-six]

Iterate from first key test       ( also next() and hasNext() methods tested )
(used 1 parameter constructor)
1 - 5 - 25 - 27 - 44 - 46 -

Iterate from last key test         ( also next() and hasNext() methods tested )
(used 1 parameter constructor)
46 - 1 - 5 - 25 - 27 - 44 -

Iterate from key in the middle test        ( also next() and hasNext() methods tested )
(used 1 parameter constructor)
27 - 44 - 46 - 1 - 5 - 25 -

Iterate with non-exist key test        ( also next() and hasNext() methods tested )
!! 0 is not the key of this map !!
(used zero parameter constructor)
1 - 5 - 25 - 27 - 44 - 46 -

Iterate with no parameter          ( also next() and hasNext() methods tested )
(used zero parameter constructor)
1 - 5 - 25 - 27 - 44 - 46 -

prev() method test with first key
Previous of 1 is 46

prev() method test with last key
Previous of 46 is 44

prev() method test with key in the middle
11 is not an element of the map.

---------------------------------------
```

a)Part2 test Chaining with LinkedList

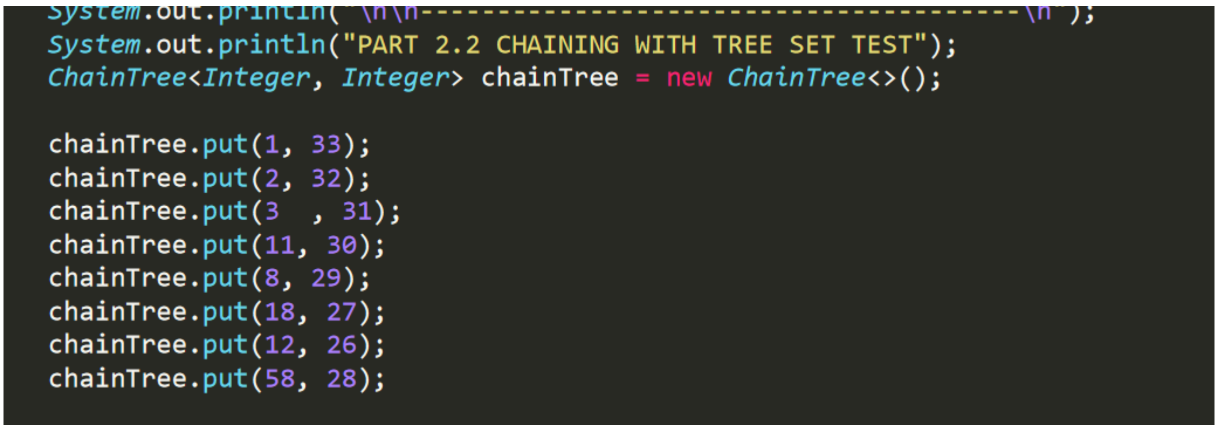


```
System.out.println("\n---------------------------------------\n");
System.out.println("PART 2.1 CHAINING WITH LINKED LIST TEST");
ChainLinked<Integer, Integer> chainLinked = new ChainLinked<>();

chainLinked.put(1, 33);
chainLinked.put(2, 32);
chainLinked.put(3, 31);
chainLinked.put(11, 30);
chainLinked.put(8, 29);
chainLinked.put(18, 27);
chainLinked.put(12, 26);
chainLinked.put(58, 28);
```

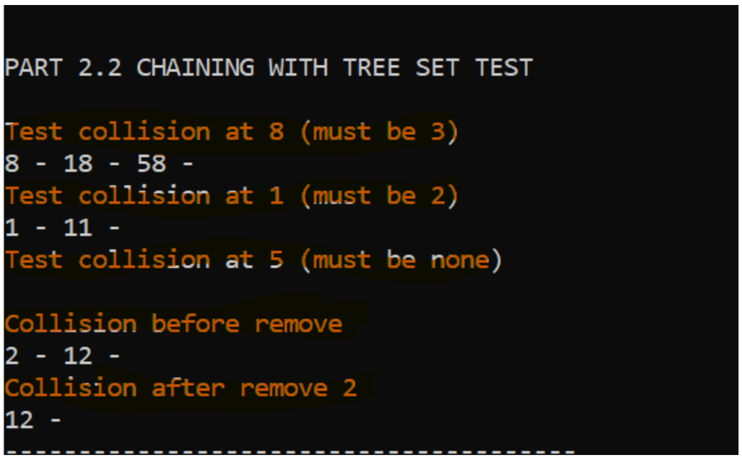


```
PART 2.1 CHAINING WITH LINKED LIST TEST

Test collision at 8 (must be 3) (a problem occured after performance tests)
8 -
Test collision at 1 (must be 2)
1 -
Test collision at 5 (must be none)

Collision before remove
2 -
Collision after remove 12
2 -
```

c)Part2 test Chaining with TreeSet

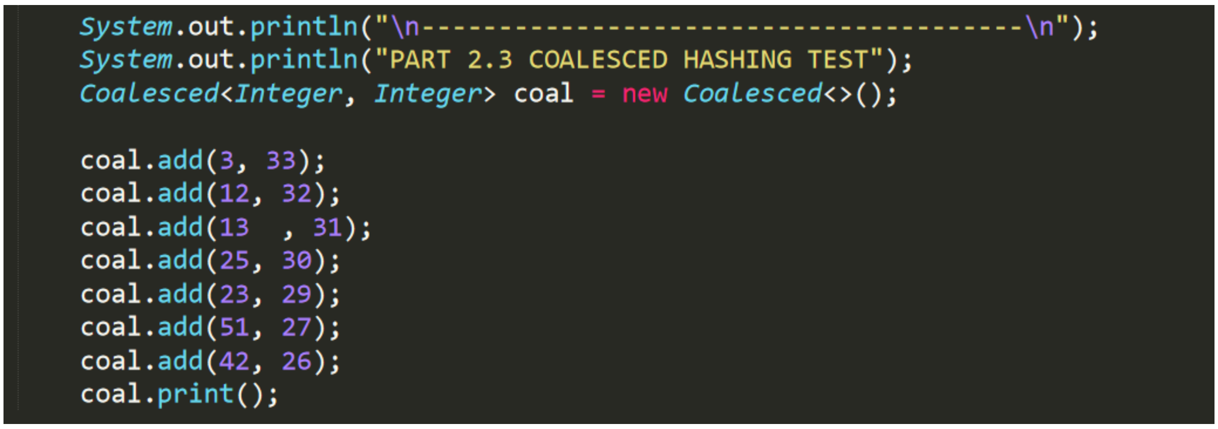


```
System.out.println("\n\n---------------------------------------\n");
System.out.println("PART 2.2 CHAINING WITH TREE SET TEST");
ChainTree<Integer, Integer> chainTree = new ChainTree<>();

chainTree.put(1, 33);
chainTree.put(2, 32);
chainTree.put(3  , 31);
chainTree.put(11, 30);
chainTree.put(8, 29);
chainTree.put(18, 27);
chainTree.put(12, 26);
chainTree.put(58, 28);
```

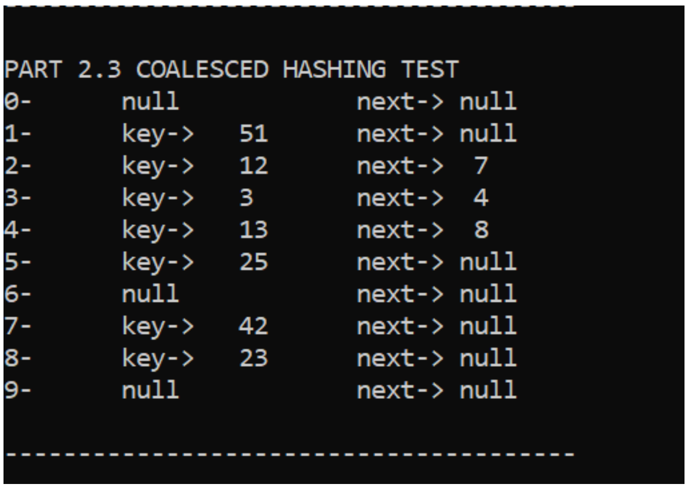


```
PART 2.2 CHAINING WITH TREE SET TEST

Test collision at 8 (must be 3)
8 - 18 - 58 -
Test collision at 1 (must be 2)
1 - 11 -
Test collision at 5 (must be none)

Collision before remove
2 - 12 -
Collision after remove 2
12 -
---------------------------------------
```

a)Part2 test Coalesced Method

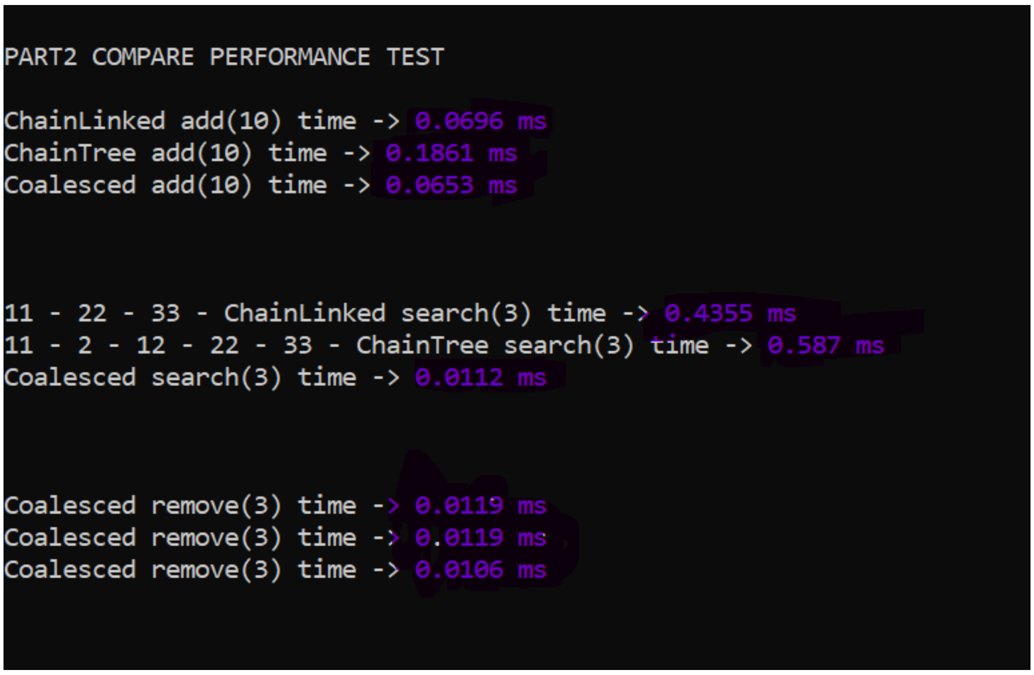


```
    System.out.println("\n----------------------------------------\n");
    System.out.println("PART 2.3 COALESCED HASHING TEST");
    Coalesced<Integer, Integer> coal = new Coalesced<>();

    coal.add(3, 33);
    coal.add(12, 32);
    coal.add(13  , 31);
    coal.add(25, 30);
    coal.add(23, 29);
    coal.add(51, 27);
    coal.add(42, 26);
    coal.print();
```



```
---------------------------------------

PART 2.3 COALESCED HASHING TEST
0-      null            next-> null
1-      key->   51      next-> null
2-      key->   12      next->  7
3-      key->   3       next->  4
4-      key->   13      next->  8
5-      key->   25      next-> null
6-      null            next-> null
7-      key->   42      next-> null
8-      key->   23      next-> null
9-      null            next-> null

---------------------------------------
```

a)Part2 test Performance Results



```
PART2 COMPARE PERFORMANCE TEST

ChainLinked add(10) time -> 0.0696 ms
ChainTree add(10) time -> 0.1861 ms
Coalesced add(10) time -> 0.0653 ms



11 - 22 - 33 - ChainLinked search(3) time -> 0.4355 ms
11 - 2 - 12 - 22 - 33 - ChainTree search(3) time -> 0.587 ms
Coalesced search(3) time -> 0.0112 ms



Coalesced remove(3) time -> 0.0119 ms
Coalesced remove(3) time -> 0.0119 ms
Coalesced remove(3) time -> 0.0106 ms
```