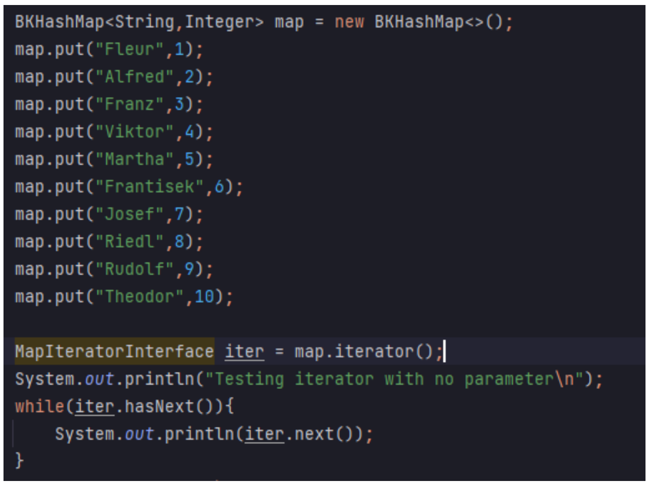
Problem Solutions Approach

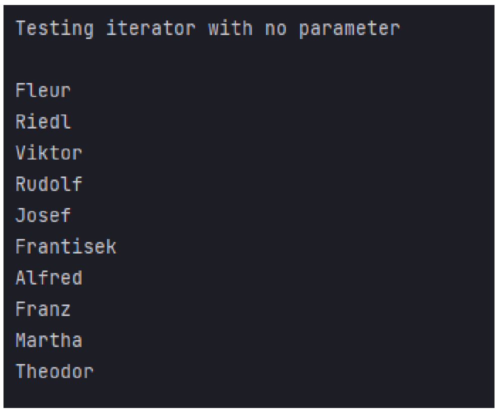
In this project, I had to implement three classes using the open addressing and chaining rules for the set table structure in java requested from me. I implemented a class that keeps the elements in the java array using the open addressing rule. If the table length and modula of the hashcodes are equal, I applied a quadratically increasing rule (as stated in the pdf). In the other two classes where I use the Chaining rule rule, I added them to the structures in the same index when the table length and modules of the incoming keys are equal, using the LinkedList and the TreeSet classes.

Running and Results

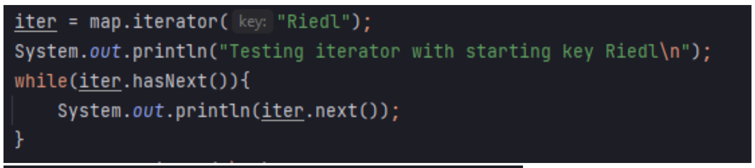
Part1:



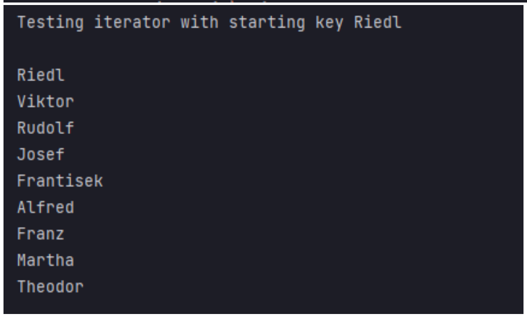
```
BKHashMap<String,Integer> map = new BKHashMap<>();
map.put("Fleur",1);
map.put("Alfred",2);
map.put("Franz",3);
map.put("Viktor",4);
map.put("Martha",5);
map.put("Frantisek",6);
map.put("Josef",7);
map.put("Riedl",8);
map.put("Rudolf",9);
map.put("Theodor",10);

MapIteratorInterface iter = map.iterator();
System.out.println("Testing iterator with no parameter\n");
while(iter.hasNext()){
    System.out.println(iter.next());
}
```



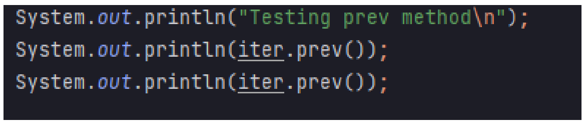
```
Testing iterator with no parameter

Fleur
Riedl
Viktor
Rudolf
Josef
Frantisek
Alfred
Franz
Martha
Theodor
```



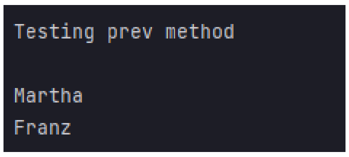
```java
iter = map.iterator( key: "Riedl");
System.out.println("Testing iterator with starting key Riedl\n");
while(iter.hasNext()){
    System.out.println(iter.next());
}
```



```
Testing iterator with starting key Riedl

Riedl
Viktor
Rudolf
Josef
Frantisek
Alfred
Franz
Martha
Theodor
```

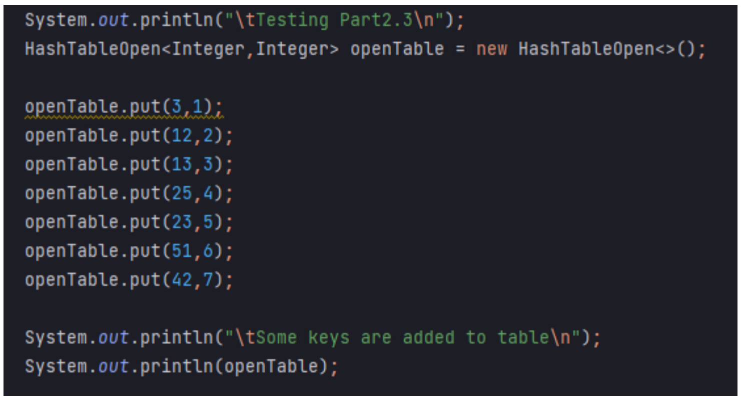


```java
System.out.println("Testing prev method\n");
System.out.println(iter.prev());
System.out.println(iter.prev());
```

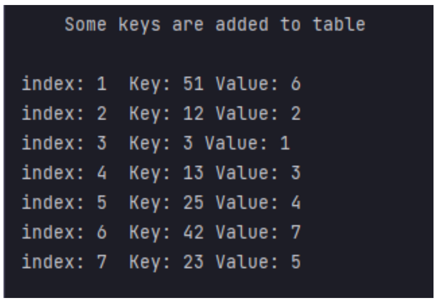


```
Testing prev method

Martha
Franz
```

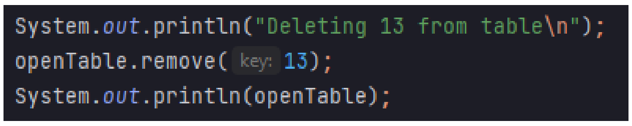
Part 2.3



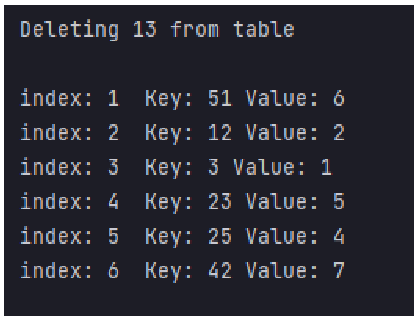
```java
System.out.println("\tTesting Part2.3\n");
HashTableOpen<Integer,Integer> openTable = new HashTableOpen<>();

openTable.put(3,1);
openTable.put(12,2);
openTable.put(13,3);
openTable.put(25,4);
openTable.put(23,5);
openTable.put(51,6);
openTable.put(42,7);

System.out.println("\tSome keys are added to table\n");
System.out.println(openTable);
```



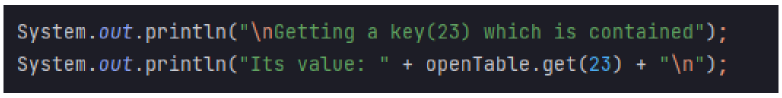
```
	Some keys are added to table

index: 1  Key: 51 Value: 6
index: 2  Key: 12 Value: 2
index: 3  Key: 3 Value: 1
index: 4  Key: 13 Value: 3
index: 5  Key: 25 Value: 4
index: 6  Key: 42 Value: 7
index: 7  Key: 23 Value: 5
```



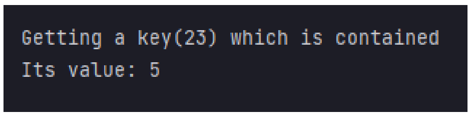
```java
System.out.println("Deleting 13 from table\n");
openTable.remove( key: 13);
System.out.println(openTable);
```



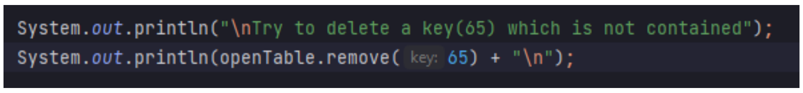
```
Deleting 13 from table

index: 1  Key: 51 Value: 6
index: 2  Key: 12 Value: 2
index: 3  Key: 3 Value: 1
index: 4  Key: 23 Value: 5
index: 5  Key: 25 Value: 4
index: 6  Key: 42 Value: 7
```



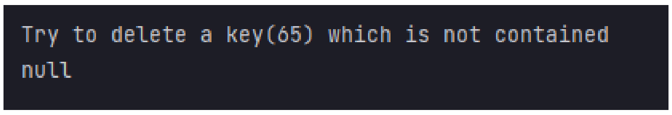
```java
System.out.println("\nGetting a key(23) which is contained");
System.out.println("Its value: " + openTable.get(23) + "\n");
```



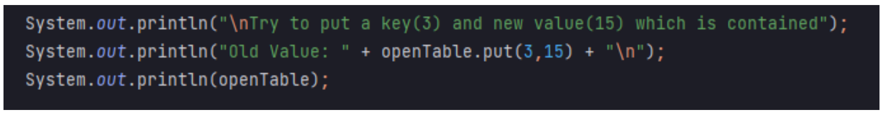
```
Getting a key(23) which is contained
Its value: 5
```



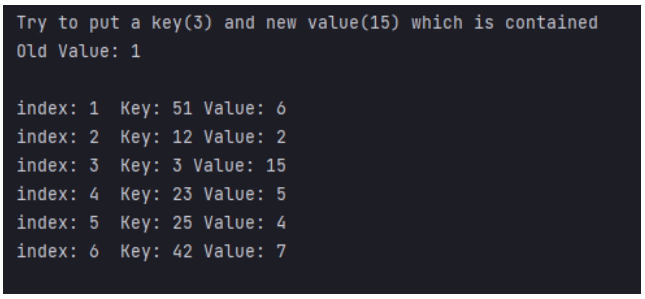
```java
System.out.println("\nTry to delete a key(65) which is not contained");
System.out.println(openTable.remove( key: 65) + "\n");
```



```
Try to delete a key(65) which is not contained
null
```

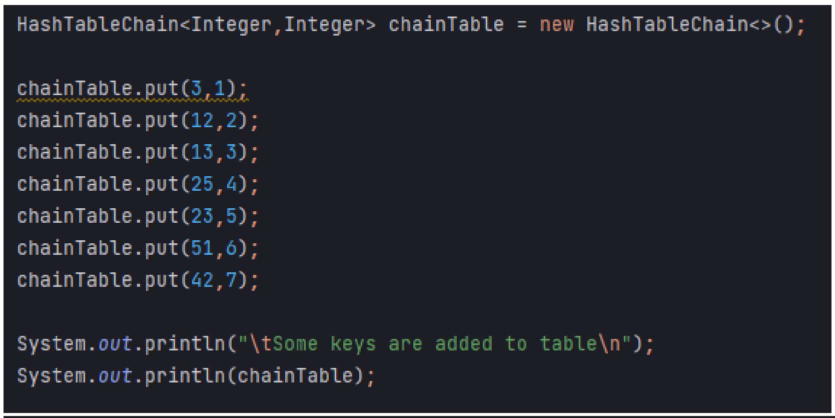


```java
System.out.println("\nTry to put a key(3) and new value(15) which is contained");
System.out.println("Old Value: " + openTable.put(3,15) + "\n");
System.out.println(openTable);
```

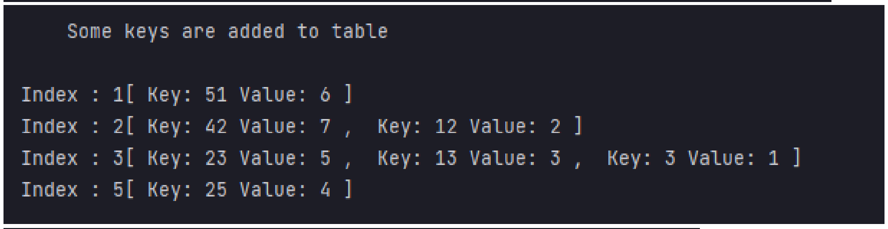


```
Try to put a key(3) and new value(15) which is contained
Old Value: 1

index: 1  Key: 51 Value: 6
index: 2  Key: 12 Value: 2
index: 3  Key: 3 Value: 15
index: 4  Key: 23 Value: 5
index: 5  Key: 25 Value: 4
index: 6  Key: 42 Value: 7
```

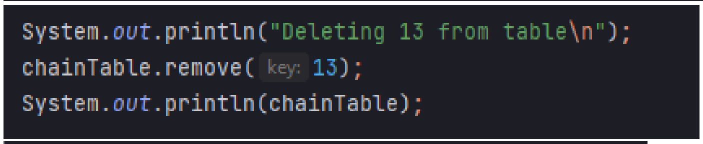
Part 2.1



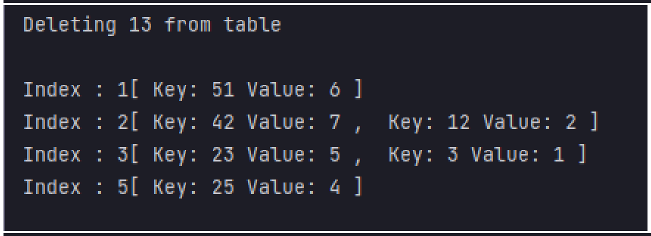
```
HashTableChain<Integer,Integer> chainTable = new HashTableChain<>();

chainTable.put(3,1);
chainTable.put(12,2);
chainTable.put(13,3);
chainTable.put(25,4);
chainTable.put(23,5);
chainTable.put(51,6);
chainTable.put(42,7);

System.out.println("\tSome keys are added to table\n");
System.out.println(chainTable);
```



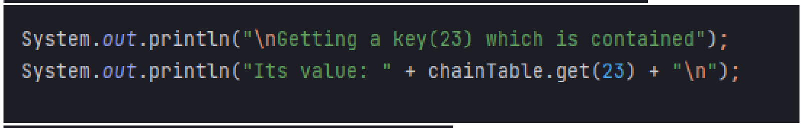
```
    Some keys are added to table

Index : 1[ Key: 51 Value: 6 ]
Index : 2[ Key: 42 Value: 7 ,  Key: 12 Value: 2 ]
Index : 3[ Key: 23 Value: 5 ,  Key: 13 Value: 3 ,  Key: 3 Value: 1 ]
Index : 5[ Key: 25 Value: 4 ]
```



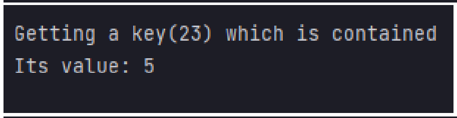
```
System.out.println("Deleting 13 from table\n");
chainTable.remove( key: 13);
System.out.println(chainTable);
```



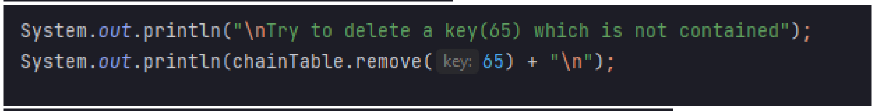
```
Deleting 13 from table

Index : 1[ Key: 51 Value: 6 ]
Index : 2[ Key: 42 Value: 7 ,  Key: 12 Value: 2 ]
Index : 3[ Key: 23 Value: 5 ,  Key: 3 Value: 1 ]
Index : 5[ Key: 25 Value: 4 ]
```



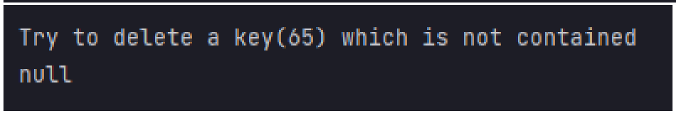
```
System.out.println("\nGetting a key(23) which is contained");
System.out.println("Its value: " + chainTable.get(23) + "\n");
```



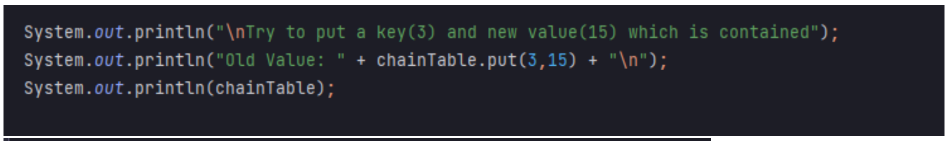
```
Getting a key(23) which is contained
Its value: 5
```

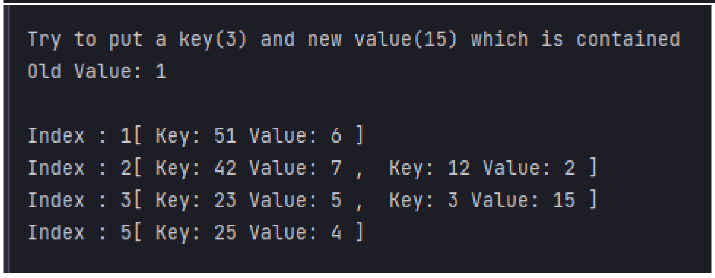


```
System.out.println("\nTry to delete a key(65) which is not contained");
System.out.println(chainTable.remove( key: 65) + "\n");
```

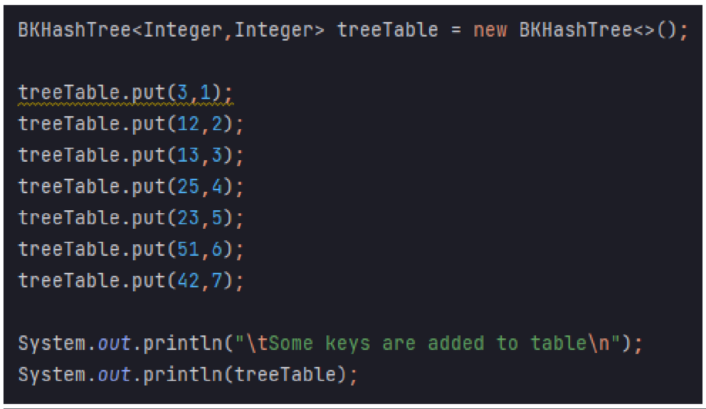


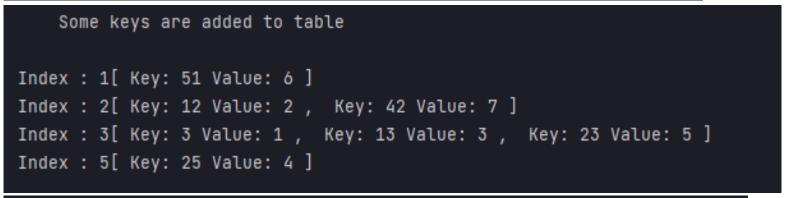
```
Try to delete a key(65) which is not contained
null
```

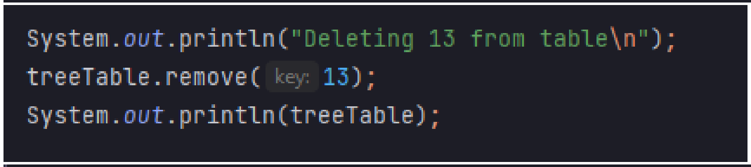

```java
System.out.println("\nTry to put a key(3) and new value(15) which is contained");
System.out.println("Old Value: " + chainTable.put(3,15) + "\n");
System.out.println(chainTable);
```

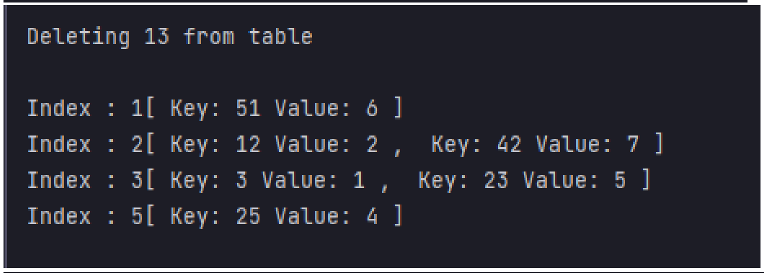

```
Try to put a key(3) and new value(15) which is contained
Old Value: 1

Index : 1[ Key: 51 Value: 6 ]
Index : 2[ Key: 42 Value: 7 ,  Key: 12 Value: 2 ]
Index : 3[ Key: 23 Value: 5 ,  Key: 3 Value: 15 ]
Index : 5[ Key: 25 Value: 4 ]
```

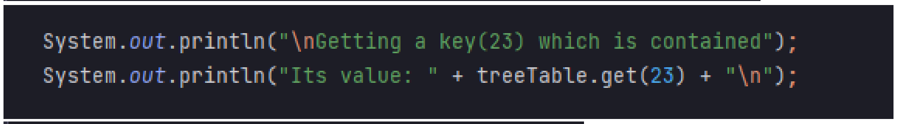
Part 2.2

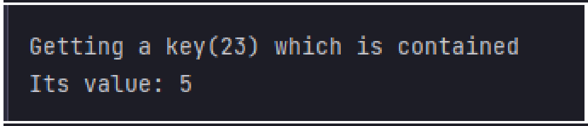

```java
BKHashTree<Integer,Integer> treeTable = new BKHashTree<>();

treeTable.put(3,1);
treeTable.put(12,2);
treeTable.put(13,3);
treeTable.put(25,4);
treeTable.put(23,5);
treeTable.put(51,6);
treeTable.put(42,7);

System.out.println("\tSome keys are added to table\n");
System.out.println(treeTable);
```

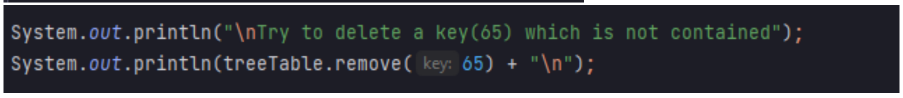

```
    Some keys are added to table

Index : 1[ Key: 51 Value: 6 ]
Index : 2[ Key: 12 Value: 2 ,  Key: 42 Value: 7 ]
Index : 3[ Key: 3 Value: 1 ,  Key: 13 Value: 3 ,  Key: 23 Value: 5 ]
Index : 5[ Key: 25 Value: 4 ]
```

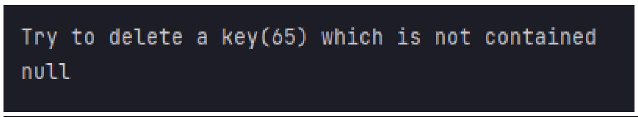

```java
System.out.println("Deleting 13 from table\n");
treeTable.remove( key: 13);
System.out.println(treeTable);
```


```
Deleting 13 from table

Index : 1[ Key: 51 Value: 6 ]
Index : 2[ Key: 12 Value: 2 ,  Key: 42 Value: 7 ]
Index : 3[ Key: 3 Value: 1 ,  Key: 23 Value: 5 ]
Index : 5[ Key: 25 Value: 4 ]
```

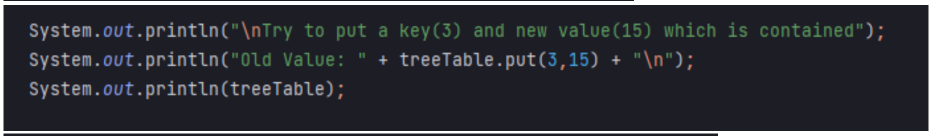

```java
System.out.println("\nGetting a key(23) which is contained");
System.out.println("Its value: " + treeTable.get(23) + "\n");
```


```
Getting a key(23) which is contained
Its value: 5
```

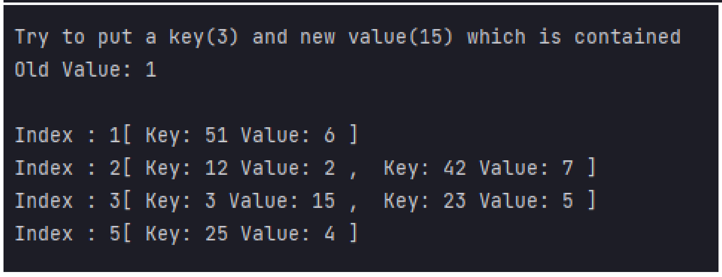

```java
System.out.println("\nTry to delete a key(65) which is not contained");
System.out.println(treeTable.remove( key: 65) + "\n");
```



```
Try to delete a key(65) which is not contained
null
```

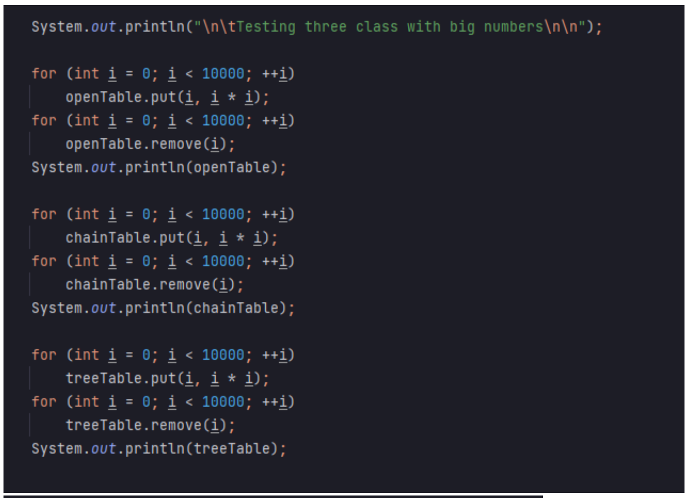


```java
System.out.println("\nTry to put a key(3) and new value(15) which is contained");
System.out.println("Old Value: " + treeTable.put(3,15) + "\n");
System.out.println(treeTable);
```

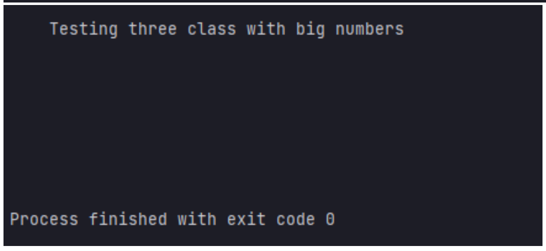


```
Try to put a key(3) and new value(15) which is contained
Old Value: 1

Index : 1[ Key: 51 Value: 6 ]
Index : 2[ Key: 12 Value: 2 ,  Key: 42 Value: 7 ]
Index : 3[ Key: 3 Value: 15 ,  Key: 23 Value: 5 ]
Index : 5[ Key: 25 Value: 4 ]
```

Testing these three classes with big numbers and sizes.



```java
System.out.println("\n\tTesting three class with big numbers\n\n");

for (int i = 0; i < 10000; ++i)
    openTable.put(i, i * i);
for (int i = 0; i < 10000; ++i)
    openTable.remove(i);
System.out.println(openTable);

for (int i = 0; i < 10000; ++i)
    chainTable.put(i, i * i);
for (int i = 0; i < 10000; ++i)
    chainTable.remove(i);
System.out.println(chainTable);

for (int i = 0; i < 10000; ++i)
    treeTable.put(i, i * i);
for (int i = 0; i < 10000; ++i)
    treeTable.remove(i);
System.out.println(treeTable);
```



```
    Testing three class with big numbers






Process finished with exit code 0
```