# แบบฝึกปฏิบัติการครั้งที่ 6

Write a class SeparateChainingSLLHashMap (use separate chaining approach to resolve collision) based on SinglyLinkedList class. I have supplied **SinglyLinkedList.java** and **List.java** for SinglyLinkedList class and **Map.java** for SeparateChainingSLLHashMap class. The printTbl() method will print the entire hash table.

Use the following main method to verify your code.

```
    public static void main(String[] args) {
        Map hashTbl = new SeparateChainingSLLHashMap(13);
        hashTbl.put(6, "A");
        hashTbl.put(12, "B");
        hashTbl.put(19, "C");
        hashTbl.put(0, "D");
        hashTbl.put(6, "E");
        hashTbl.put(19, "F");
        hashTbl.put(32, "G");
        hashTbl.put(45, "H");
        hashTbl.printTbl();
        System.out.println("key:6, value:"+hashTbl.get(6));
        System.out.println("key 45 exist?: "+hashTbl.containsKey(45));
        System.out.println("key 17 exist?: "+hashTbl.containsKey(17));
        System.out.println("Remove key:45. The table becomes");
        hashTbl.remove(45);
        hashTbl.printTbl();
    }
```

The correct results should be (you can print the table in whatever format you like)

```
table[0]: (0,D)
table[1]: null
table[2]: null
table[3]: null
table[4]: null
table[5]: null
table[6]: (6,E) (19,F) (32,G) (45,H)
table[7]: null
table[8]: null
table[9]: null
table[10]: null
table[11]: null
table[12]: (12,B)
key:6, value:E
key 45 exist?: true
key 17 exist?: false
Remove key:45. The table becomes
table[0]: (0,D)
table[1]: null
table[2]: null
table[3]: null
table[4]: null
table[5]: null
table[6]: (6,E) (19,F) (32,G)
table[7]: null
table[8]: null
table[9]: null
table[10]: null
table[11]: null
table[12]: (12,B)
```

Here is how your hash table would look like in **SeparateChainingSLLHashMap** class.

Suppose h(K) = K % tableSize

```
Map hashTbl = new SeparateChainingSLLHashMap(5);
hashTbl.put(6, “A”);
hashTbl.put(4, “B”);
hashTbl.put(19, “C”);
```