```
package cn.sxt.mycollection;
/**
 * 自定义一个HashMap
 * 增加泛型
 * @author 江
 *
 *
 */
public class SxtHashMap04<K,V> {
    Node3[] table;   //位桶数组
    int size;            //存放的键位长度

    public SxtHashMap04() {
            table=new Node3[16];     //长度一般定义为2的整数幂
    }

    @Override
        public String toString() {
      //(10:aa,20:bb)
                    StringBuilder sb=new StringBuilder("(");

                    for(int i=0;i<table.length;i++) {
                            Node3 temp=table[i];
                            while(temp!=null) {
                                    sb.append(temp.key+":"+temp.value+",");
                                    temp=temp.next;
                            }
                    }
                    sb.setCharAt(sb.length()-1,')');
                    return sb.toString();
        }

    public int myHash(int hashcode,int length) {
            System.out.print("hash in myhash:"+(hashcode&(length-1))+"\t");
```

```
        System.out.println("hash in myhash: "+(hashcode%(length-1)));
        return hashcode&(length-1);
}

public void put(K key,V value) {

        //定义新的节点对象
        Node3 newNode=new Node3();
        newNode.hash=myHash(key.hashCode(),table.length);
        newNode.key=key;
        newNode.value=value;
        newNode.next=null;

        Node3 temp=table[newNode.hash];
        Node3 iterLast=null;   //正在遍历的最后一个元素
        boolean keyRepeat=false;

        if(temp==null) {
                //此处数组元素为空，则直接将节点放进去
                table[newNode.hash]=newNode;
                size++;
        }else {
                //若此处数组不为空，则遍历对应链表
                while(temp!=null) {
                        //判断key是否重复，重复则覆盖
                        if(temp.key.equals(key)){
                              keyRepeat=true;
                            System.out.println("重复了");
                            temp.value=value;   //只需覆盖value即可
                            break;
                        }else {
                                //key不重复时，则遍历下一个
                                iterLast=temp;
                                temp=temp.next;
                        }
```

```
                    }
                    if(!keyRepeat) {
                            //没有发生key重复的情况，则添加到链表的最后
                    iterLast.next=newNode;
                    size++;
                    }
            }
}

public V get(K key) {
        int hash=myHash(key.hashCode(),table.length);
        V value=null;

        Node3 temp=table[hash];
        //遍历bucket数组
        if(temp!=null) {
                //遍历链表
                while(temp!=null) {
                        if(temp.key.equals(key)) {
                                value=(V)temp.value;
                                break;
                        }else {
                                temp=temp.next;
                        }
                }
        }
        return value;
}


public static void main(String[] args) {
      SxtHashMap04<Integer,String> m=new SxtHashMap04<>();
      m.put(10,"aa");
      m.put(20, "bb");
      m.put(30, "cc");
```

```
            m.put(20, "dd");

            m.put(53, "gg");
            m.put(69, "hh");
            m.put(85, "ii");
            System.out.println(m);

            System.out.println(m.get(69));
}
}
package cn.sxt.mycollection;
/**
  * 增加泛型
  * @author 江
  *
  * @param <K>
  * @param <V>
  */
public class Node3<K,V> {
            int hash;
            K key;
            V value;
            Node3 next;
}
```