下面两个方法，testPut会存在丢失的情况。而testPutIfAbsent不会

```
public static void testPutIfAbsent() throws InterruptedException{
    ConcurrentHashMap<String, List<String>> map = new ConcurrentHashMap<>();
    Thread threadOne = new Thread (new  Runnable() {
        @Override
        public void run() {
            List<String> list = new ArrayList<>();
            list.add("d1");
            list.add("d2");

             List<String> oldList =  map.putIfAbsent("topicl", list);
             if(null != oldList) {
                 oldList.addAll(list);
             }
        }
    });

    Thread threadTwo = new Thread (new  Runnable() {
        @Override
        public void run() {
            List<String> list = new ArrayList<>();
            list.add("d11");
            list.add("d21");

            List<String> oldList =  map.putIfAbsent("topicl", list);
            if(null != oldList) {
                oldList.addAll(list);
            }
        }
    });

    Thread threadThree = new Thread (new  Runnable() {
        @Override
        public void run() {
            List<String> list = new ArrayList<>();
            list.add("d111");
            list.add("d211");

            List<String> oldList =  map.putIfAbsent("topic2", list);
            if(null != oldList) {
                oldList.addAll(list);
            }
        }
    });

    threadOne.start();
    threadTwo.start();
    threadThree.start();

    Thread.sleep(3000);
    System.out.println(map.toString());
}

```

```
53 private static void testPut() throws InterruptedException {
54     ConcurrentHashMap<String, List<String>> map = new ConcurrentHashMap<>();
55     Thread threadOne = new Thread (new  Runnable() {
56         @Override
57         public void run() {
58             List<String> list = new ArrayList<>();
59             list.add("d1");
60             list.add("d2");
61 
62             map.put("topicl", list);
63         }
64     });
65 
66     Thread threadTwo = new Thread (new  Runnable() {
67         @Override
68         public void run() {
69             List<String> list = new ArrayList<>();
70             list.add("d11");
71             list.add("d21");
72 
73             map.put("topicl", list);
74         }
75     });
76 
77     Thread threadThree = new Thread (new  Runnable() {
78         @Override
79         public void run() {
80             List<String> list = new ArrayList<>();
81             list.add("d111");
82             list.add("d211");
83 
84             map.put("topic2", list);
85         }
86     });
87 
88     threadOne.start();
89     threadTwo.start();
90     threadThree.start();
91 
92     Thread.sleep(3000);
93     System.out.println(map.toString());
94 }
```

总结: put(Kkey,Vvalue)方法判断如果key已经存在， 则使用value覆盖原来的值并 返回原来的值，如果不存在则把 value 放入并返回 null。而是如果 key 已经存在则直接返回原来对应的值并不使用 value 覆盖，如果 key 不存在 则放入 value 并返回 null，另外要注 意， 判断 key 是