```java
public class HashtableExample {
  public static void main(String[] args) {
    //Hashtable 컬렉션 생성
    Map<String, Integer> map = new Hashtable<>();

    //작업 스레드 객체 생성
    Thread threadA = new Thread() {
      @Override
      public void run() {
        //엔트리 1000개 추가
        for (int i = 1; i <= 1000; i++) {
          map.put(String.valueOf(i), i);
        }
      }
    };

    //작업 스레드 객체 생성
    Thread threadB = new Thread() {
      @Override
      public void run() {
        //엔트리 1000개 추가
        for (int i = 1001; i <= 2000; i++) {
          map.put(String.valueOf(i), i);
        }
      }
    };

    //작업 스레드 실행
    threadA.start();
    threadB.start();

    //작업 스레드들이 모두 종료될 때까지 메인 스레드를 기다리게 함
    try {
      threadA.join();
      threadB.join();
    } catch (Exception e) {
    }

    //저장된 총 엔트리 수 얻기
    int size = map.size();
    System.out.println("총 엔트리 수: " + size);
    System.out.println();
  }
}
```

실행 결과

```
총 엔트리 수: 2000
```

※ 아래 처럼 변경후 실행하면 2000개가 나오지 않거나, PC에 따라 에러가 발생할 수 있다. HashMap은 두 스레드가 동시에 put() 메소드를 호출할 수 있기 때문에 경합이 발생하고 결국은 하나만 저장되기 때문이다.

```java
Map<String, Integer> map = new HashMap<>();
```