```java
public class Board {
  private String subject;
  private String content;
  private String writer;

  public Board(String subject, String content, String writer) {
    this.subject = subject;
    this.content = content;
    this.writer = writer;
  }

  public String getSubject() { return subject; }
  public void setSubject(String subject) { this.subject = subject; }
  public String getContent() { return content; }
  public void setContent(String content) { this.content = content; }
  public String getWriter() { return writer; }
  public void setWriter(String writer) { this.writer = writer; }
}
```

```java
public class VectorExample {
  public static void main(String[] args) {
    //Vector 컬렉션 생성
    List<Board> list = new Vector<>();
    //작업 스레드 객체 생성
    Thread threadA = new Thread() {
      @Override
      public void run() {
        //객체 1000개 추가
        for(int i=1; i<=1000; i++) {
          list.add(new Board("제목"+i, "내용"+i, "글쓴이"+i));
        }
      }
    };

    //작업 스레드 객체 생성
    Thread threadB = new Thread() {
      @Override
      public void run() {
        //객체 1000개 추가
        for(int i=1001; i<=2000; i++) {
          list.add(new Board("제목"+i, "내용"+i, "글쓴이"+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 객체 수 얻기
    int size = list.size();
    System.out.println("총 객체 수: " + size);
    System.out.println();
  }
}
```

실행 결과

```
총 객체 수: 2000
```

ArrayList<>()로 변경시 실행 결과는 2000개가 나오지 않거나, PC에 따라 에러가 발생할 수 있다. 그 이유는 ArrayList는 두 스레드가 동시에 add() 메소드를 호출할 수 있기 때문에 경합이 발생해 결국은 하나만 저장되기 때문이다. 반면 Vector의 add()는 동기화 메소드이므로 한 번에 하나의 스레드만 실행할 수 있어 경합이 발생하지 않는다.

```java
List<Board> list = new ArrayList<>();
```