Bài 7: SO SÁNH VỚI ARRAYLIST- VECTOR

Vector và ArrayList kiểm tra tốc độ thực hiện cùng nhiệm vụ trong khoảng thời gian xác định. Cả hai tập hợp xử lý cùng một nhiệm vụ như nhau và kết quả thực thi cho thấy Vector thực hiện tốt trong thao tác đa luồng hơn ArrayList.

Nhiệm Vụ:

- Vector và ArrayList thực hiện thao tác thêm các phần tử vào tập hợp là các số nguyên ngẫu nhiên trong khoảng (0 đến 10)
- Số lần thực hiện 30.000 lần
- Thời gian thực hiện: 160000ms.
- Kết quả: Vector Arraylist thêm được: 30.000 phần tử còn Arraylist được: 29994 phần tử

Hướng dẫn cài đặt. Sinh viên thực hiện theo hướng dẫn và mở rộng tìm hiểu kiến thức tạo luồng chạy song song.

Tổng quan cài đặt:

- Xây dựng 2 luồng: Thread vectorWorker; và Thread arrayListWorker;
- Thực thi phương thức run() và gọi 2 luồng khởi động.
- Bật bộ đếm thời gian
- In kích thước 2 tập hợp sau khoảng thời gian đã đặt sẵn.

```
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Random;
6  import java.util.Timer;
7  import java.util.TimerTask;
   import java.util.Vector;
9
10 public class VectorSynchronizeSample {
       private static final Vector<Integer> VECTOR = new Vector<>();
12     private static final List<Integer> ARRAYLIST = new ArrayList<>();
13     public static void main(String[] args) throws InterruptedException {
14         Thread vectorWorker;
15         vectorWorker = new Thread(new Runnable() {...16 lines });
31         Thread arrayListWorker;
32         arrayListWorker= new Thread(new Runnable() {
           public void run() {...13 lines }
46         });
47         vectorWorker.start();
48         arrayListWorker.start();
49         new Timer().schedule(new TimerTask() {
50         @Override
           public void run() {
52             System.out.println("Vector: Size" + VECTOR.size());
53             System.out.println("\nArrayList: Size" + ARRAYLIST.size());
54             System.exit(0);
55         }
56     }, 16000);
57     }
```

Chi tiết cài đặt:

VectorSynchronizeSample.java

Tạo lớp thử nghiệm
public class VectorSynchronizeSample {

```
private static final Vector<Integer> VECTOR = new Vector<>();
private static final List<Integer> ARRAYLIST = new
ArrayList<>();
```

```
public static void main(String[] args) throws
InterruptedException {
```

```
//khai báo luồng vector
Thread vectorWorker;
//khởi tạo luồng
vectorWorker = new Thread(new Runnable() {
@Override
public void run() {
     //đặt tên luồng và thực thi 30.000 lần lặp
     Thread.currentThread().setName("Vector_worker");
     for (int index = -1; ++index < 30000;) {
          Thread thr = new Thread("Vector_worker_index_" +
          index) {
               @Override
               public void run() {
               //Thêm các phần tử ngẫu nhiên vào tập hợp
               VECTOR.add(new Random().nextInt(10));
               System.out.println(this.getName());
             }
             };
             thr.start();
           }
           }
        });
```

```
//khai báo luồng ArrayList
Thread arrayListWorker;
//khởi tạo luồng
arrayListWorker= new Thread(new Runnable() {
public void run() {
     //đặt tên luồng và thực thi 30.000 lần lặp
     Thread.currentThread().setName("Arraylist_worker");
     for (int index = -1; ++index < 30000;) {
          Thread thr = new Thread("Arraylist_worker_index_" +
          index) {
               @Override
               public void run() {
                    //Thêm các phần tử ngẫu nhiên vào tập hợp
                    ARRAYLIST.add(new Random().nextInt(10));
                    System.out.println(this.getName());
```

```
                    }
                };
                thr.start();
            }
        }
        });
vectorWorker.start();
arrayListWorker.start();
new Timer().schedule(new TimerTask() {
@Override
     public void run() {
          System.out.println("Vector: Size" + VECTOR.size());
          System.out.println("\nArrayList: Size" +
          ARRAYLIST.size());
          System.exit(0);
        }
    }, 16000);
    }
}
```