SmartHunter marcador

Programación concurrente y Distribuída

Curso 2011-12

Miguel Telleria, Laura Barros, J.M. Drake

`telleriam AT unican.es`

Computadores y Tiempo Real

http://www.ctr.unican.es

Objetivo

- En el programa **SmartHunter** crear una partición **marcador** para llevar la cuenta de:
  - Jets inofensivos
  - Misiles lanzados (por batería)
  - Jets que han impactado.
  - Jets ofensivos sin misil
  - Jets derribados (por baterias)

```
inofensivos: 0   no asignados: 0    impactos:  0
con misil/explotados: E -> 0/0     N -> 0/0     O -> 0/0    S -> 1/0
```

- Lo complicºado:
  - Saber donde poner las llamada ← **Esto os lo daremos hecho siempre**
  - Gestionar la concurrencia
- Lo fácil
  - Distribuirlo

Interfaz

```
public interface  Marcador
{
    // Se llama para esperar a que se inicialicen los datos
    public void init();

    // Lo llama AirRaid cuando genera un nuevo jet inofensivo
    public void nuevo_jet_inofensivo();

    // Lo llama AirRaid cuando genera un nuevo jet ofensivo
    public void nuevo_jet_ofensivo();

    // Lo llama AirRaid cuando detecta el impaco
    public void nuevo_impacto();

    // Lo llama missil cuando se le asigna un jet
    public void nuevo_misil_lanzado(String cardinalidad);

    // Lo llama SpeedFalcon cuando se le hace explotar
    public void jet_explotado(String cardinalidad);
}
```

Datos que mantiene

```
public class Marcador
{
    private Object lock_num_jets_inofensivos;
    private int num_jets_inofensivos;

    private Object lock_num_jets_no_asignados;
    private int num_jets_no_asignados;

    private Object lock_num_impactos;
    private int num_impactos;

    private  Hashtable<String, Object> locks_num_misiles_asignados;
    private Hashtable<String, Integer> nums_misiles_asignados;

    private Hashtable<String, Object> locks_num_misiles_explotados;
    private Hashtable<String, Integer> nums_misiles_explotados;
    private Object lock_print;

    // Estos se inicializan los primeros
    private boolean inicializados = false;
    private Object lock_init = new Object();

    ...
```

Inicializacion

```
public Marcador()
{
    lock_num_jets_inofensivos = new Object();
    num_jets_inofensivos = 0;
    ...

    locks_num_misiles_asignados = new Hashtable<String,Object>();
    nums_misiles_asignados = new Hashtable<String,Integer>();
    locks_num_misiles_explotados = new Hashtable<String, Object>();
    nums_misiles_explotados = new Hashtable<String, Integer>();

    for (String cardinalidad : GeoData.BATTERY_NAME)
    {
        locks_num_misiles_asignados.put(cardinalidad, new Object());
        nums_misiles_asignados.put(cardinalidad, 0);
        locks_num_misiles_explotados.put(cardinalidad, new Object());
        nums_misiles_explotados.put(cardinalidad, 0);
    }
    ...
    // Cuando ya esta todo inicializado

    synchronized(lock_init)
    {
        inicializados = true;
        lock_init.notify();
    }

}
```

Sincronismo de espera en inicialización

```
public void init()
{
    synchronized(lock_init)
    {
        while (!inicializados)
        {
            try {
                lock_init.wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Distribución

- Partiendo de la versión monoprocesador
  - Distribuir el marcador a una partición distinta con RMI
  - Distribuir el marcador a una partición distinta con ICE

- Desactivar el lanzamiento de misiles para testear la detección de los impactos.

- Nota sobre la implementación de Ice y funciones con '_':
  - Ice no soporta por defecto identificadores con subrayados en slice.
  - A partir de ice 3.4, sí se pueden usar dando la opción `–underscore` al slice2java.

```
slice2java --tie --underscore SmartHunterMarc.ice
```