Bloque 2. Paradigmas de la Programación

Tarea 4: Definir el concepto de liveness en concurrencia. Poner algún ejemplo.

1.Definición

Garantiza que algo bueno sucederá en algún momento a lo largo del tiempo.

Liveness es un término empleado para denotar la capacidad de un programa concurrente de ejecutarse de manera oportuna. Es una propiedad de un programa concurrente que nos indica que en algún momento a lo largo de la vida del programa se ejecutará una parte determinada del programa que llegará a un estado deseable.

Para cumplir esta propiedad, el programa debe asegurar la ausencia de deadlocks, livelocks e inanición.

Ejemplo en la vida real: en una carrera, se va a garantizar que mínimo una persona llegue a la meta y gane. Algo bueno, (como es ganar la medalla), finalmente sucede.

Normalmente este término va acompañado de otra propiedad, safety.

2.Ejemplo

Teniendo en cuenta lo que hemos explicado previamente, podemos intuir que un código con exclusión mutua que evitará los problemas de inanición, deadlocks y demás será suficiente para considerarlo liveness.

En este código concurrente de java podemos apreciar el uso de synchronized, método con el que lograremos dicha exclusión mutua y que permitirá hacer uso de variables compartidas a un único hilo cada vez.

```
// A Java program to demonstrate working of
// synchronized.
import java.io.*;
import java.util.*;

// A Class used to send a message
class Sender
{
    public void send(String msg)
    {
        System.out.println("Sending\t"  + msg );
        try
        {
            Thread.sleep(1000);
        }
        catch (Exception e)
        {
            System.out.println("Thread  interrupted.");
        }
        System.out.println("\n" + msg + "Sent");
    }
}

// Class for send a message using Threads
class ThreadedSend extends Thread
{
    private String msg;
    Sender  sender;

    // Recieves a message object and a string
    // message to be sent
    ThreadedSend(String m,  Sender obj)
    {
```

```
        msg = m;
        sender = obj;
    }

    public void run()
    {
        // Only one thread can send a message
        // at a time.
        synchronized(sender)
        {
            // synchronizing the snd object
            sender.send(msg);
        }
    }
}

// Driver class
class SyncDemo
{
    public static void main(String args[])
    {
        Sender snd = new Sender();
        ThreadedSend S1 =
            new ThreadedSend( " Hi " , snd );
        ThreadedSend S2 =
            new ThreadedSend( " Bye " , snd );

        // Start two threads of ThreadedSend type
        S1.start();
        S2.start();

        // wait for threads to end
        try
        {
            S1.join();
            S2.join();
        }
        catch(Exception e)
        {
            System.out.println("Interrupted");
        }
    }
}
```

Cuyo resultado de ejecución no variará, siendo este siempre el mismo:

```
Sending      Hi

 Hi Sent
Sending      Bye

 Bye Sent
```

2.Bibliografía

1.4. Safety and Liveness | Coursera. (2020). Retrieved 29 October 2020, from https://www.coursera.org/lecture/cloud-computing/1-4-safety-and-liveness-sFeOE

Liveness (The Java™ Tutorials > Essential Classes > Concurrency). (2020). Retrieved 29 October 2020, from https://docs.oracle.com/javase/tutorial/essential/concurrency/liveness.html

Synchronized in Java - GeeksforGeeks. (2020). Retrieved 29 October 2020, from https://www.geeksforgeeks.org/synchronized-in-java/