UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

DEPARTAMENTO DE INGENIERÍA DE SOFTWARE

LIC. INGENIERÍA DE SOFTWARE

APLICACIONES PARA AMBIENTES DISTRIBUIDOS

LABORATORIO #2

SEBASTIÁN SÁNCHEZ 8-944-2002

FACILITADOR:
Prof. Regis Rivera

Grupo 1SF251

ABRIL, 2025

Introducción

En este laboratorio estaremos hablando de sincronización en Java, debido a que los programas multiproceso a menudo pueden llegar a una situación en la que varios subprocesos intentan acceder a los mismos recursos y finalmente producen resultados erróneos e imprevistos.

La sincronización de Java se utiliza para asegurarse de que, mediante algún método de sincronización, solo un subproceso pueda acceder al recurso en un momento dado.

Problema 1 -

```
class SyncDemo {
    public static void main (String args[]) {
        Sender send = new Sender();
        ThreadedSend S1 = new ThreadedSend(" Hola ", send);
        ThreadedSend S2 = new ThreadedSend (" Adios ", send);

        S1.start();
        S2.start();

        try {
            S1.join();
            S2.join();
        }
        catch (Exception e) {
            System.out.println("Interrumpido");
        }
    }
}

class ThreadedSend extends Thread {
    private String msg;
    Sender sender;

    ThreadedSend(String m, Sender obj)
    {
        msg = m;
        sender =obj;
    }

    public void run()
    {
        synchronized (sender)
        {
            sender.send(msg);
        }
    }
}

class Sender {
    public void send(String msg)
    {
        System.out.println("Enviando\t" + msg);
        try {
```

```
            Thread.sleep(1000);
        }
        catch (Exception e) {
            System.out.println("Hilo Interrumpido");
        }
        System.out.println("\n" + msg + "Enviado");
    }
}
```

Problema 2

```
import java.io.*;

public class GFG {

    public static void main(String args[]){

        final Test obj = new Test() ;

        Thread a = new Thread() {
            public void run() { obj.test_function(15); }
        };

        Thread b = new Thread() {
            public void run() { obj.test_function(30); }
        };

        a.start();
        b.start();
    }
}

class Test {
    synchronized void test_function(int n)
    {

        for (int i = 1; i <= 3; i++) {
            System.out.println(n + i);
            try {
                Thread.sleep(500);
            }
            catch (Exception e) {
                System.out.println(e);
            }
        }
    }
}
```

Problema 3

```
import java.io.*;
import java.util.*;

public class SychroTest {

    public static void main(String[] args)
    {
        PrintTest p = new PrintTest();

        Thread1 t1 = new Thread1(p);
        Thread2 t2 = new Thread2(p);

        t1.start();
        t2.start();
    }
}

class PrintTest extends Thread {
    public void printThread(int n){
        for (int i = 0; i < 10; i++) {
            System.out.println("Thread " + n + " está trabajando...");
            try {
                Thread.sleep(600);
            } catch (Exception ex) {
                System.out.println(ex.toString());
            }
        }
        System.out.println("----------------------");
        try {
            Thread.sleep(1000);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
}

class Thread1 extends Thread {

    PrintTest test;

    Thread1(PrintTest p){ test = p; }

    public void run() {
```

```
        test.printThread(1);
    }
}

class Thread2 extends Thread {
    PrintTest test;

    Thread2(PrintTest p) {test = p;}
    public void run() {test.printThread(2); }
}
```

Problema 4

```
class SynchroTest {

    public static void main(String[] args)
    {

        PrintTest p = new PrintTest();

        Thread1 t1 = new Thread1(p);
        Thread2 t2 = new Thread2(p);

        t1.start();
        t2.start();
    }
}

class PrintTest extends Thread {

    synchronized public void printThread(int n)
    {
        for (int i = 0; i < 10; i++) {
            System.out.println("Thread " + n + " está trabajando...");

            try {
                Thread.sleep(600);
            } catch (Exception ex) {
                System.out.println(ex.toString());
            }
        }
        System.out.println("------------");
        try {
            Thread.sleep(1000);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
}

class Thread1 extends Thread {

    PrintTest test;

    Thread1(PrintTest p){ test = p; }
```

```
    public void run() {
        test.printThread(1);
    }
}

class Thread2 extends Thread {
    PrintTest test;

    Thread2(PrintTest p) {test = p;}
    public void run() {test.printThread(2); }
}
```