

## Laboratorio de Software

### Práctica nº 9

#### Temas

- Threads. Ciclo de vida.
- Sincronización de threads.
- Ejecutores

1. Implemente una aplicación Swing que muestre una imagen en forma intermitente cada 1 segundo, en cualquier lugar de la pantalla. Evalúe distintos mecanismos para hacerlo.

2. Analice el siguiente código y responda:

```
public class TestSynchronized extends Thread {
    String[] frase = {"España", "en", "los", "diarios", "de", "mi",
"vejez"};

    public TestSynchronized(String id){
        super(id);
    }
    public void run() {
        synchronized(System.out){
            for (int i = 0; i < frase.length; i++){
                System.out.print(this.getName()+" "+ frase[i]+"\\n");
            }
        }
    }
    public static void main(String[] args) {
        TestSynchronized t1 = new TestSynchronized("Thread 1");
        TestSynchronized t2 = new TestSynchronized("Thread 2");
        TestSynchronized t3 = new TestSynchronized("Thread 3");
        t1.start();t2.start();t3.start();
    }
}
```

- a.- ¿Cuál es el efecto del **synchronized(System.out)**?
- b.- ¿Qué tipo de **lock** hace el código dado?

3. Implemente una aplicación Swing que simule una carrera, donde cada participante está representado por un objeto **Runnable**. Para ello, cree una clase llamada **Corredor** que dibuje una línea horizontal desde la largada hasta la llegada.

a.- Use un ejecutor con un pool de tamaño 5 para ejecutar. Luego cambie el tamaño del pool a 3 y observe la ejecución de los threads.

1.	_____
2.	_____
3.	_____
4.	_____
5.	_____

b.- Supongamos que se quiere saber si un corredor abandona la carrera, retornando algún valor predefinido o en el peor de los casos, disparando una excepción. ¿Podría hacerlo con el método run() de la interface **Runnable**?. Analice la interface **Callable**, usando la documentación de la API y observe sus ventajas.