

Laboratorio de Software

Práctica nº 7

Temas

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

1. Implemente en Kotlin una aplicación que muestre la hora actual en consola y la actualice cada 1 segundo. Evalúe distintos mecanismos para hacerlo.

2. Analice el siguiente código Kotlin y responda:

```
class TestSynchronized(id: String) : Thread(id) {
    var frase: Array<String> = arrayOf("UNLP", "PÚBLICA", "AHORA", "Y", "SIEMPRE")
    override fun run() {
        synchronized(System.out) {
            for (palabra in frase)
                println("${this.name} : ${palabra} ")
        }
    }
}

fun main(args: Array<String>) {
    val t1 = TestSynchronized("Thread 1")
    val t2 = TestSynchronized("Thread 2")
    val t3 = 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 que simule una carrera de 100 metros, donde cada participante está representado por un objeto **thread**. Para ello, cree un programa que muestre por consola la cantidad de metros recorrida por cada corredor.

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.

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. Analice la interface **Callable**, usando la documentación de la API y observe sus ventajas.

