Laboratorio de Software
Práctica nº 8

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

1. Implemente 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 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 que simule una carrera de 100 metros, donde cada participante está representado por un objeto ***Runnable***. Para ello, cree una clase llamada **Corredor** que muestre por consola la cantidad de metros recorrida.

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. ¿Podría hacerlo con el método run() de la interface **Runnable**?. Analice la interface **Callable**, usando la documentación de la API y observe sus ventajas.