# Prácticas Concurrencia y Distribución (20/21)

Arno Formella, Anália García Lourenço

semana 15 – 21 febrero

## Práctica 2: Comportamiento básico de los hilos

**Objetivos:** Sincronización simple de los hilos al final de ejecución, medición de tiempo de ejecución.

**Material adicional:** Son de especial interés los siguientes enlaces

- https://docs.oracle.com/javase/10/docs/api/java/lang/Thread.html
- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

1. Determinación de la terminación del hilo.

   Este ejercicio explora cómo determinar cuando termina un hilo o un grupo de hilos. Para ello, debes seguir estos pasos:

   *a*) **Configuración del problema:** Duplicar el código de la semana pasada. En particular, escribe una clase `MyThread` que extiende `Thread` (o implementa `Runnable`) e imprima el nombre del hilo en ejecución. (Ten en cuenta que tu versión podría tener una estructura ligeramente diferente y/o con diferentes nombres de clase, pero debes lograr esencialmente el mismo resultado).

   *b*) **Detalles:** en el método `main` de la clase principal, crea una lista (o una matriz) de hilos e inícialos. Inmediatamente después de iniciar los hilos, desde el hilo principal, imprime a la pantalla un mensaje, p.ej. *el programa ha terminado*. Para crear una lista de hilos debes tener una variable `number_of_threads` que recoge el numero de hilos especificado por línea de comando (mira ejercicio de la semana anterior):

   ```
   int number_of_threads;

   List<Thread> threadList =
     new ArrayList<Thread>(number_of_threads);
   ```

```
for(int i=1; i<=number_of_threads; ++i) {
  //...
}
```

¿Cuál es el resultado de tu código? ¿En qué orden se imprimen los hilos?

*c*) **Modificar la clase principal:** ahora queremos imprimir un mensaje desde el hilo principal cuando todos los otros hilos han terminado. Con el método `isAlive()` en un bucle de control en la rutina principal, se puede determinar el estado de cada hilo (es decir, si todavía está vivo). La estructura del bucle/`isAlive()` para capturar el estado de cada subproceso debería ser similar la siguiente:

```
while(t.isAlive()) {
  try {
  }
  catch() {
  }
}
```

donde `t` es uno de los hilos.

*d*) **Modificación con `join`:** Dado que la técnica anterior de bucle-`isAlive()` se utiliza con tanta frecuencia, Java tiene un método especial llamado `join()` que hace esencialmente lo mismo. Reemplaza el bucle/`isAlive()` de arriba por el `join()` ¿Funciona exactamente igual? ¿Afecta la ejecución de los subprocesos en ejecución?

2. Medición del tiempo de ejecución.

   Queremos mapear el ciclo de vida de los hilos en un programa concurrente registrando los tiempos cuando cambian sus estados. Sigue los siguientes pasos para investigar el comportamiento de los hilos:

   *a*) **Configuración del problema:** escribe una clase principal (que contiene `main`) y una clase que extiende `Thread` como en problemas anteriores. En la clase principal, comienza la ejecución de una lista de hilos que van a realizar una operación matemática (que se explica en el próximo paso).

   *b*) **Implementación de la clase `MyThread`:** el hilo debe ejecutar una operación de cálculo intensivo. Para esto, una prueba históricamente interesante es la *prueba de remojo PDP-11*, (vea https://en.wikipedia.org/wiki/PDP-11) que se basa en la aplicación continua de funciones trascendentes. Para esto, podemos aplicar continuamente la tangente y su inverso, seguidos por la raíz cúbica. Una implementación puede ser la siguiente:

```
for(int i=0; i<1000000; ++i) {
  double d=tan(atan(
    tan(atan(
      tan(atan(
        tan(atan(
          tan(atan(123456789.123456789))
        ))
      ))
    ))
  ));
  cbrt(d);
}
```

*c*) **Diagrama de ejecución de hilos:** ahora queremos entender lo que hace cada hilo durante su ciclo de vida. Para ello, debes insertar diagnósticos (utilizando `System.Print.` o `Logger`) en tu código para mapear el ciclo de vida de cada hilo. Debes distinguir entre el momento en que se ejecuta y el momento en que termina. ¿Puedes distinguir entre la creación del hilo, la ejecución y la terminación final? ¿Qué problemas encuentras cuando intentas determinar las diferentes fases? El siguiente segmento de código podría ser útil (recuerda que el mensaje debe también incluir el tiempo):

```
LOGGER.debug("Soy {}", Thread.currentThread().getName());
```

*d*) **Análisis. Tiempo de ejecución respecto a número de hilos:** Estudia el comportamiento de tu código a medida que aumentas la cantidad de subprocesos implementados; es decir, comienza con un hilo y aumenta hasta un gran número de hilos, imprimiendo el tiempo total de ejecución. Sería útil ejecutar el código Java en un script bash (o en Windows, utilizando algo similar) que guarde los resultados de la medición en un archivo de texto simple.

*e*) **Haz una gráfica del tiempo de ejecución:** con el archivo del paso anterior, utiliza un programa degeneración de gráficas como por ejemplo *gnuplot*, *matplotlib* o *seaborn* para hacer gráficas del tiempo de ejecución en función del número de subprocesos (es decir, el eje $x$ es el número de subprocesos, mientras que el eje $y$ es el tiempo de ejecución). A partir de estas gráficas, ¿qué conclusiones puedes sacar?