Universidad de los Andes
Ingeniería de Sistemas y Computación

ISIS 2203 Infraestructura Computacional

Taller manejo de threads

El propósito es construir un programa que sume todos los elementos de una matriz usando *threads*. Habrá un *thread* por cada fila de la matriz; cada *thread* se encarga de sumar los elementos de su fila. El `main`, después de generar los *threads*, espera a que terminen, y entonces recoge los resultados parciales, los acumula, e imprime el total en la consola.

A continuación encuentra partes del programa:

- Variables estáticas de la clase:

```
private static int [][] M;        //La matriz que se desea sumar
private static int total = 0;     //El total de la suma
```

- Atributos propios de cada thread:

```
private int id;
private int suma;
public boolean terminó;
```

- Método para inicializar la matriz:

Inicializa la matriz con los números en secuencia desde cero; cambiarlo para efectuar pruebas.

```
private static void crearMatriz ( int n ) {

  M = new int [n][n];
  int m = 0;

  for ( int i = 0; i < n; i++ ) {
    for ( int j = 0; j < n; j++ ) {
      M[i][j] = m;
      m++;
    }
  }
}
```

1) Escriba el método constructor de la clase. ¿Qué parámetros debe tener? ¿Qué atributos debe inicializar?

2) Escriba el `main` de la clase (suponemos que la clase se llama `T`). ¿Cómo puede hacer el `main` para saber cuándo terminó un *thread*?

```
public static void main(String[] args) {

  int nThreads = 10;  //Número de threads; un valor cualquiera.
  T [] t;   //Vector para los threads

  crearMatriz( nThreads );    //Inicializar la matriz

  //A continuación, escribir código para crear los threads

  //A continuación, escribir código para esperar que los threads terminen
  //y recoger los resultados parciales
```

```
  System.out.println( total );
}
```

3) Escriba el método `run` de la clase.

4) Con respecto al programa anterior, describa y explique dónde se necesita sincronización entre los *threads* y en qué consiste dicha sincronización.

5) Modifique el programa anterior de la siguiente manera: los *threads* calculan la suma parcial, como antes, pero después ellos mismos se encargan de acumular el resultado en la variable `total`. En cuanto al `main`, él espera a que todos los *threads* terminen, y entonces imprime el resultado total en consola.

6) Con respecto al programa anterior, describa y explique dónde se necesita sincronización entre los *threads* y en qué consiste dicha sincronización. También describa y explique dónde puede haber conflictos en el acceso a las variables.

7) Modifique el programa para que calcule el máximo de la matriz. Pruébelo varias veces; ¿sale siempre la misma respuesta? ¿es la respuesta correcta? En sus pruebas incluya cambiar el orden de inicialización de la matriz (en particular, pruébelo inicializando la matriz al contrario: de 99 hacia abajo).