Programación de Servicios y Procesos

TEMA 2

Ventajas de los hilos frente a los procesos

- Se tarda menos tiempo en crear un hilo de una tarea existente que en crear un nuevo proceso
- Se tarda menos tiempo en terminar un hilo que en terminar un proceso
- Se tarda menos tiempo en cambiar entre dos hilos de una misma tarea que en cambiar entre dos procesos (porque los recursos no cambian, por ejemplo)
- Es más sencillo la comunicación (paso de mensajes por ejemplo) entre hilos de una misma tarea que entre diferentes procesos
- Cuando se cambia de un proceso a otro, tiene que intervenir el núcleo del sistema operativo para que haya protección. Cuando se cambia de un hilo a otro, puesto que la asignación de recursos es la misma, no hace falta que intervenga el sistema operativo

Cómo crear hilos

Heredar Thread	Implementar Runnable	Utilizando una clase anónima
La clase que estamos creando no podrá heredar de otra	Forma de utilizarla diferente	Rápida y cómoda de usar pero no recomendable

Heredar Thread

- Al heredar esta clase, tenemos que implementar el método **run()**
- Este método contendrá el código que queremos ejecutar cuando lancemos el hilos con **start()**.

```
public class Tarea extends Thread {
  @Override
  public void run() {
      System.out.println("Soy un hilo");
    }
  }
}
```

```
public class Programa {
  public static void main(String args[]) {
    Tarea tarea = new Tarea();
    tarea.start();
    System.out.println("Yo soy el hilo principal y sigo
haciendo mi trabajo");
    System.out.println("Fin del hilo principal");
  }
}
```

Implementar Runnable

- Si necesitamos heredar de otra clase
- Implementamos la clase runnable y desarrollamos el método **run()**
- Para ejecutar el hilo, tendremos que crear un objeto de tipo Thread y debemos pasarle nuestra clase en el constructor

```
public class Tarea extends OtraClase
implements Runnable {
  @Override
  public void run() {
      System.out.println("Soy un hilo");
  }
}
```

```
public class Programa {
  public static void main(String args[]) {
    Tarea tarea = new Tarea();
    Thread hilo = new Thread(tarea);
    hilo.start();
    System.out.println("Yo soy el hilo principal y sigo
haciendo mi trabajo");
    System.out.println("Fin del hilo principal");
  }
}
```

Hilo como clase anónima

- Solo recomendable para hilos poco complejos

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

    Thread hilo = new Thread(new Runnable() {
      @Override
      public void run() {
        for (int i = 0; i < 10; i++) {
        System.out.println("Soy un hilo y esto es lo que hago");
      }
    });

    hilo.start();
    System.out.println("Yo soy el hilo principal y sigo haciendo mi trabajo");
    System.out.println("Fin del hilo principal");
  }
}
```

Consideraciones

- Siempre se debe sobreescribir el método run() e implementar allí lo que tiene que hacer el hilo
- **Los problemas vienen cuando existen varios hilos. Hay que tener en cuenta que pueden compartir datos y código y encontrarse en diferentes estados de ejecución**
    - **Condiciones de carrera**
- La ejecución de nuestra aplicación será thread-safe si se puede garantizar una correcta manipulación de los datos que comparten los hilos de la aplicación sin resultados inesperados
- En el caso de aplicaciones multihilo, también nos puede interesar sincronizar y comunicar unos hilos con otros

Sincronización de hilos

- **join()**
  - Se espera la terminación del hilo que invoca a este método antes de continuar
- **Thread.sleep(int)**
  - El hilo que ejecuta esta llamada permanece dormido durante el tiempo especificado como parámetro (en ms)
- **isAlive()**
  - Comprueba si el hilo permanece activo todavía (no ha terminado su ejecución)
- **yield()**
  - Sugiere al planificador que sea otro hilo el que se ejecute (no se asegura)

Sincronización de recursos

- Cuando un método acceda a una **variable** que esté **compartida** deberemos proteger dicha sección crítica, usando **synchronized**. Se puede poner todo el método synchronized o marcar un trozo de código más pequeño.

```
public synchronized void incrementar(){
      cuenta=cuenta+1;
}
```

```
public  void incrementar(){
        System.out.println("Otras cosas");
        synchronized(this){
                cuenta=cuenta+1;
        }
        System.out.println("Mas cosas...");
}
```