HILOS EN JAVA

THREADS

CARACTERÍSTICAS

- Es un proceso en ejecución dentro de un programa.
- Contiene recursos propios.
- Comparte el espacio de memoria.
- Dentro de un mismo programa puede haber varios hilos en ejecución.
- Los hilos pueden comunicarse entre sí.
- Implementan prioridad y sincronización.

CREACIÓN DE THREADS

- Existen dos formas: crear una nueva clase como subclase de la clase Thread o declarar una clase e implementar la interfaz Runnable.
  - Uso de la SubClase:
    Crear la subclase de Thread y sobreescribir el método run() con el código que se ejecutará por el hilo.
  - Uso de la interface Runnable:
    La clase debe implementar la interface Runnable, desarrollar el método run() con el código que se ejecutará por el hilo.

EJECUCIÓN DE UN THREAD

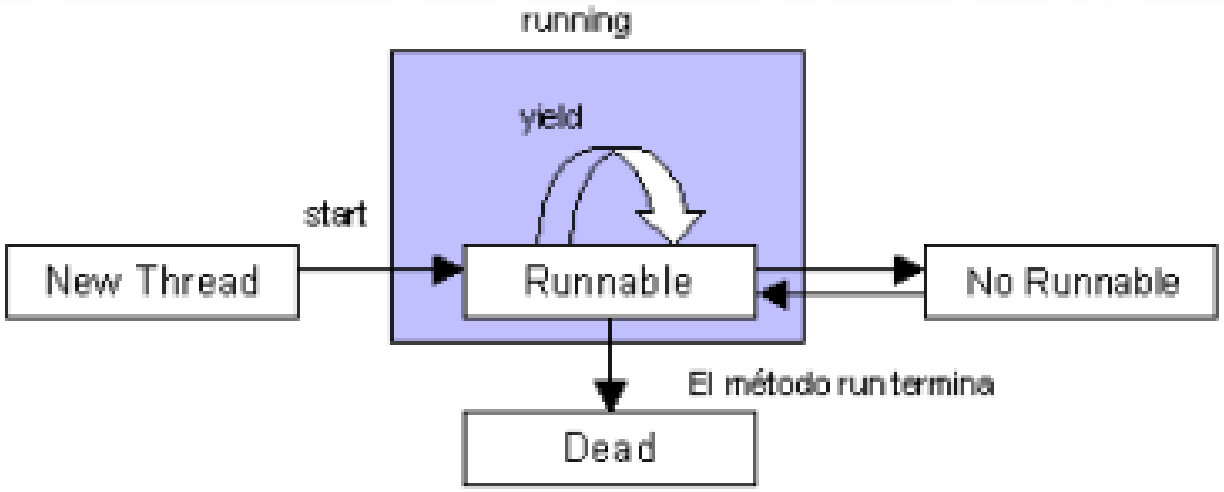
- Primero se debe instanciar el Thread.
- Ejecutar el método start().
  - Esto es para que la JVM lo pueda ejecutar como un thread realizando la llamada al método run().

```java
public class MiHilo extends Thread {
        public MiHilo(String nombre) {
            super(nombre);
        }

        public void run() {
            for(int i=0; i<10; i++) {
                System.out.println("Hilo " + this.getName()
                + " paso: " + i);
            }
            System.out.println("Hilo " + this.getName()
            + " finalizado.");
        }

        public static void main(String args[]) {

            MiHilo h1 = new MiHilo("@Lucas");
            MiHilo h2 = new MiHilo("@Fede");

            h1.start();
            h2.start();
        }
}
```

```java
public class MiSegundoHilo implements Runnable {
    private String nombre;

    MiSegundoHilo(String nombre) {
        this.nombre=nombre;
    }

    public String getNombre() {
        return nombre;
    }

    public void run() {
        for(int i=0; i<10; i++) {
            System.out.println("SegundoHilo "
            + this.getNombre() + " paso: " + i);
        }
        System.out.println("SegundoHilo "
        + this.getNombre() + " finalizado.");
    }

    public static void main(String args[]) {
        Runnable h1 = new MiSegundoHilo("@Lucas");
        Runnable h2 = new MiSegundoHilo("@Fede");

        new Thread(h1).start();
        new Thread(h2).start();
    }
}
```

ESTADOS DE UN THREAD

1. Creación del Thread
2. En ejecución
   El contexto de ejecución existe y el hilo puede ocupar el CPU en cualquier momento.
3. En no ejecución
4. Finalizado



SINCRONIZACIÓN

- Problema
  - Todos los hilos de un programa comparten el espacio de memoria, haciendo posible que dos hilos accedan a la misma variable o mismo espacio de memoria. Se crea así la necesidad de disponer de un mecanismo para bloquear el acceso de un hilo a un dato crítico si el dato está siendo usado por otro hilo.
- Solución
  - Recurrir a métodos sincronizados, esto evita que dos invocaciones de métodos sincronizados del mismo objeto se mezclen. Cuando un hilo ejecuta un método sincronizado de un objeto, todos los hilos que invoquen métodos sincronizados del objeto se bloquearán hasta que el primer hilo termine con el objeto.
  - Con esto nos aseguramos que al terminar un método sincronizado, se garantizará que todos los hilos verán los cambios realizados sobre el objeto.

```java
public class Contador {
    private int x;

    public synchronized void incX() {
        x++;
    }

    public String toString() {
        return "x es " + x;
    }
}
```

```
Thread @Lucas x es 1
Thread @Fede x es 2
Thread @Lucas x es 3
Thread @Fede x es 4
Thread @Lucas x es 5
Thread @Fede x es 6
Thread @Lucas x es 7
Thread @Lucas x es 9
Thread @Lucas x es 10
Thread @Lucas x es 11
Thread @Lucas x es 12
Thread @Lucas x es 13
Thread @Lucas x es 14
Thread @Lucas finalizado.
Thread @Fede x es 8
Thread @Fede x es 15
Thread @Fede x es 16
Thread @Fede x es 17
Thread @Fede x es 18
Thread @Fede x es 19
Thread @Fede x es 20
Thread @Fede finalizado.
```

```java
public class Hilo extends Thread {
    private Contador c;

    public Hilo(String nombre, Contador c) {
        super(nombre);
        this.c = c;
    }

    public void run() {
        for(int i=0; i<10; i++) {
            c.incX();
            System.out.println("Thread "
            + this.getName() + " " + c);
        }
        System.out.println("Thread "
        + this.getName() + " finalizado.");
    }

    public static void main(String args[]) {
        Contador c1 = new Contador();

        Hilo h1 = new Hilo("@Lucas", c1);
        Hilo h2 = new Hilo("@Fede", c1);

        h1.start();
        h2.start();
    }
}
```