TAREA1HILOS

CODIGO 1:

```
class LifeOff implements Runnable {
   protected int countDown = 10;
   private static int taskCount = 0;
   private final int id = taskCount;

   public LifeOff() {}

   public LifeOff(int countDown) {
       this.countDown = countDown;
   }

   @Override
   public void run() {
       while (countDown-- > 0) {
           System.out.println("#" + id + "(" + countDown + "), ");
           countDown--;
       }
       System.out.println("Lanzamiento {" + id + "}!");
   }


}


public class Main  {

   public static void main(String[] args) {

       LifeOff lifeOff = new LifeOff();
       lifeOff.run();

       LifeOff lifeOff2 = new LifeOff();
       lifeOff2.run();

       LifeOff lifeOff3 = new LifeOff();
       lifeOff3.run();

       System.out.println("Comienza la cuenta atras");


   }

}
```

Daniel Remiro Laguna

TAREA1HILOS

CODIGO 2

```
class LifeOff2 extends Thread {
   protected int countDown = 10;
   private static int taskCount = 0;
   private final int id = taskCount++;

   public LifeOff2() {}

   public LifeOff2(int countDown) {
       this.countDown = countDown;
   }

   @Override
   public void run() {
       while (countDown-- > 0) {
           System.out.print("#" + id + "(" + countDown + "), ");
           Thread.yield();
       }
       System.out.println("Lanzamiento {" + id + "}!");
   }
}

public class Main2 {

   public static void main(String[] args) {

       Thread t1 = new Thread(new LifeOff2());
       Thread t2 = new Thread(new LifeOff2());
       Thread t3 = new Thread(new LifeOff2());

       t1.start();
       t2.start();
       t3.start();

       System.out.println("Comienza la cuenta atras (dicho desde el
hilo main)");
   }
}
```

Daniel Remiro Laguna

TAREA1HILOS

1.

¿Qué está pasando con la ejecución del programa anterior?

Tras ejecutarlo, ¿el mensaje "Comienza la cuenta atrás!" está puesto en el sitio correcto?

intenta crear más instancias del la clase LiftOff y haz que se ejecuten todas (dentro del main)

Si observas la salida de aplicación, ¿está haciendo algo diferente a una aplicación monohilo? ¿Qué puedes extraer de la salida del programa?

En el primer programa, donde la clase LifeOff implementa Runnable pero en main se invoca directamente el método .run(), se observa un comportamiento totalmente secuencial y monohilo.

Al llamar a lifeOff.run(), no se está iniciando un nuevo hilo de ejecución. Simplemente se está ejecutando el método run() como cualquier otro método ordinario, dentro del contexto del hilo principal (el hilo main).

El hilo main ejecuta lifeOff.run() y debe esperar a que este termine por completo. Solo entonces, procede a crear lifeOff2 y ejecutar lifeOff2.run(), esperando nuevamente a que termine. Este proceso se repite para todas las instancias.

El mensaje "Comienza la cuenta atras" aparece al final de toda la ejecución. Esto es coherente con un programa monohilo, ya que esa línea de código en main solo puede ejecutarse después de que todas las llamadas previas (las llamadas a .run()) se hayan completado en orden.

La salida del programa es ordenada.. Se ve la cuenta atrás completa de la primera tarea, seguida de la cuenta atrás completa de la segunda, y así sucesivamente. No existe ninguna diferencia funcional con una aplicación monohilo estándar que llama a varios métodos uno detrás de otro.

Daniel Remiro Laguna

TAREA1HILOS

2.

Copia el ejemplo original de la "Cuenta Atrás" y haz que la clase LiftOff ahora herede de Thread. Ahora, en vez de llamar directamente al método run, haz que los threads llamen al método start().

El mensaje "Comienza la cuenta atrás!" ¿aparece ahora en el sitio correcto? ¿Porqué sale antes si en el código está después?

Crea nuevas instancias de LiftOff y has que se lancen en el main ¿En qué ha cambiado ahora la ejecución de las aplicación respecto a una aplicación monohilo? ¿Qué puedes extraer de la salida del programa?

Al modificar la clase LifeOff para que herede de Thread y al invocar el método .start() en main, el comportamiento de la aplicación eambia.

El método .start() le indica a la Java Virtual Machine (JVM) que debe crear un nuevo hilo de ejecución. La JVM es quien, internamente, se encarga de llamar al método run() de ese objeto en el contexto de ese nuevo hilo.

El mensaje "Comienza la cuenta atrase" aparece ahora casi inmediatamente al ejecutar el programa, a menudo antes de que las cuentas atrás hayan comenzado o avanzado significativamente.

Cuando el hilo main llama a t1.start(), no espera a que el hilo t1 termine su trabajo. Simplemente ejecuta al iinicio y continúa inmediatamente con su siguiente instrucción (t2.start(), t3.start(), etc.). Tras dar la última orden de inicio, el hilo main ejecuta su última instrucción (System.out.println(...)) y finaliza su parte. Los nuevos hilos t1, t2, etc.," comienzan a ejecutar su método run() de forma paralela.Ahora existen múltiples hilos ejecutándose simultáneamente.

Las impresiones de las diferentes cuentas atrás aparecen mezclada.

Daniel Remiro Laguna

TAREA1HILOS

3.

Copia el código de ThreadMethodsExample y divídelo en dos clases. Por un lado, una que contenga a la clase que extiende de Thread y otra que simplemente tenga el método main y el código para crear y lanzar los hilos..

Una vez dividido el código cambia U3S2_ThreadMethodsExample para que implemente la interfaz Runnable. Haz los cambios oportunos en la otra clase para que todo vuelva a funcionar como antes.

```
public class U3S2_ThreadMethodsExample implements Runnable {

   @Override
   public void run() {

        String threadName = Thread.currentThread().getName();
        System.out.println("["+threadName+"] " + "Inside the
thread");
        System.out.println("["+threadName+"] " + "Priority: "
                + Thread.currentThread().getPriority());
        Thread.yield();
        System.out.println("["+threadName+"] " + "Id: "
                + Thread.currentThread().getId());
        System.out.println("["+threadName+"] " + "ThreadGroup: "
                +
Thread.currentThread().getThreadGroup().getName());
        System.out.println("["+threadName+"] " + "ThreadGroup
count: "
                +
Thread.currentThread().getThreadGroup().activeCount());
   }
}
```

TAREA1HILOS

```java
public class LanzadorHilos {

    public static void main(String[] args) {
        // main thread
        Thread.currentThread().setName("Main");

        System.out.println(Thread.currentThread().getName());
        System.out.println(Thread.currentThread().toString());

        ThreadGroup even = new ThreadGroup("Hilos impares");
        ThreadGroup odd = new ThreadGroup("Hilos pares");

        Thread localThread = null;

        for (int i = 0; i < 10; i++) {

            Runnable tarea = new U3S2_ThreadMethodsExample();

            ThreadGroup grupo = (i % 2 == 0) ? even : odd;
            String nombre = "Hilo" + i;

            localThread = new Thread(grupo, tarea, nombre);

            localThread.setPriority(i + 1);
            localThread.start();
        }

        try {
            localThread.join();
        } catch (InterruptedException ex) {
            System.out.println("error");
        }

        System.out.println("Main thread ending");
    }
}
```

Daniel Remiro Laguna