



# Instituto Politecnico Nacional

## Escuela Superior de Cómputo



Bocanegra Heziquio Yestlanezi

Practica 4

Hilos

2. Pégle al topo

21 de mayo de 2021

2CM13

Programación Orientada a Objetos

I need someone

## Contenido

Introduccion.....	3
Hilos (Threads) en Java .....	3
Definicion Thread .....	3
Uso de Subclase .....	4
RunnableThread.java .....	4
Desarrollo .....	6
Conclusion.....	8
Bocanegra Heziquio Yestlanezi .....	8
Código 1 Compilacion del código desde (CMD).....	6
Código 2 Ejecución del código desde (CMD).....	6
Código 3 Programa ejecutándose en Java .....	7





# Introducción

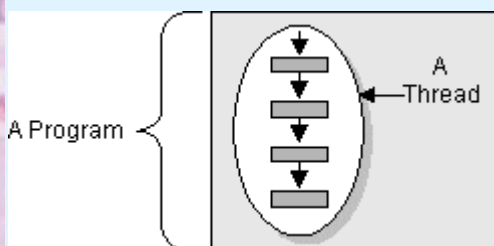
## Hilos (Threads) en Java

En la primera parte de este curso vimos procesos en sistemas operativos que surgen de Unix. En esos ejemplos cada proceso se caracterizaba por tener un único control de flujo, es decir la ejecución del programa seguía una secuencia única. Cuando creamos un nuevo proceso con fork, creamos una nueva secuencia que se ejecuta concurrentemente con el proceso padre, pero no comparten las zonas de datos y la comunicación entre ellos es muy limitada. Por ello aparecen las pipes y los otros mecanismos de comunicación entre procesos. Los hilos son otra forma de crear la posibilidad de concurrencia de actividades; sin embargo, la gran diferencia es que los hilos comparten el código y el acceso a algunos datos en forma similar a como un objeto tiene acceso a otros objetos. En Java un hilo es un objeto con capacidad de correr en forma concurrente el método run(). En cierta manera es como tener dos "program counters" para un mismo código. Una diferencia con los procesos es que carece de sentido y no es posible en este enfoque hacer mutar un proceso con algo similar a exec().

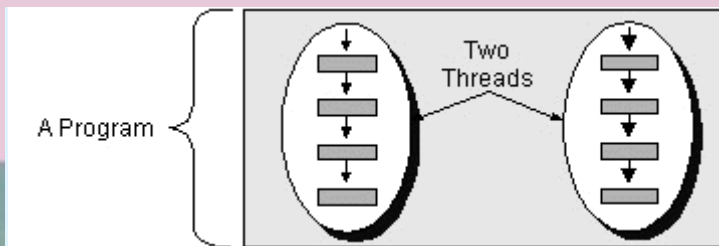
### Definición Thread

Una thread es un único flujo de control dentro de un programa. Algunas veces es llamado contexto de ejecución porque cada thread debe tener sus propios recursos, como el program counter y el stack de ejecución, como el contexto de ejecución. Sin embargo, toda thread en un programa aún comparte muchos recursos, tales como espacio de memoria y archivos abiertos. Threads también son llamadas procesos livianos (lightweight process).

NOTA: Es mucho más simple crear y destruir una thread que un proceso.



**Thread Paralelas y Concurrentes**  
Cuando dos threads corren en paralelo, ambas están siendo corridas al mismo tiempo en diferentes CPUs. Dos thread concurrentes están en progreso, o intentando de obtener tiempo de ejecución de la CPU al mismo tiempo, pero no necesariamente están corriendo en forma simultánea en dos CPUs diferentes.



## Uso de Subclase

Cuando se crea una subclase de Thread, la subclase debería definir su propio método run() para sobre montar el método run() de la clase Thread. La tarea concurrente es desarrollada en este método run().

**Ejecución del método run()**  
Una instancia de la subclase es creada con new, luego llamamos al método start() de la thread para hacer que la máquina virtual Java ejecute el método run(). Ojo para iniciar la concurrencia invocamos a start(), así invocamos a run() en forma indirecta. Si invocamos a run() directamente, se comportará como el llamado a cualquier método llamado dentro de un mismo hilo (sin crear uno independiente).

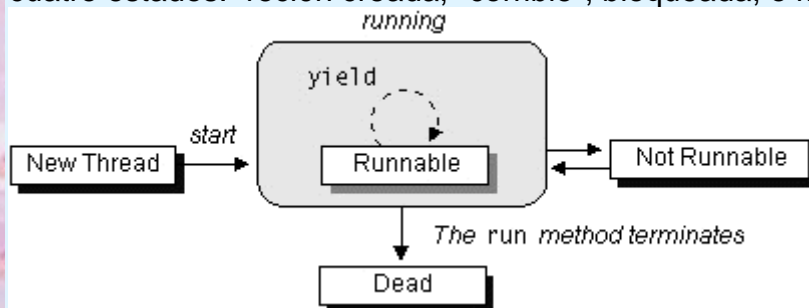
**Implementación de la Interfaz Runnable**  
La interfaz Runnable requiere que sólo un método sea implementado, el método run(). Primero creamos una instancia de esta clase con new, luego creamos una instancia de Thread con otra sentencia new y usamos el objeto recién creado en el constructor. Finalmente, llamamos el método start() de la instancia de Thread para iniciar la tarea definida en el método run().

### RunnableThread.java

Una instancia de una clase que defina el método run() - ya sea como subclase de Thread o implementando la interfaz Runnable - debe ser pasada como argumento en la creación de una instancia de Thread. Cuando el método start() de esta instancia es llamado, Java run time sabe qué método run() ejecutar.

### Ciclo de Vida de una Thread

Cada hilo, después de su creación y antes de su destrucción, estará en uno de cuatro estados: recién creada, "corriente", bloqueada, o muerta.



**Recién creada (New thread):** entra aquí inmediatamente después de su creación. Es decir luego del llamado a new. En este estado los datos locales son ubicados e iniciados. Luego de la invocación a start(), el hilo pasa al estado "corriente".

Corrible (Runnable): Aquí el contexto de ejecución existe y el hilo puede ocupar la CPU en cualquier momento. Este estado puede subdividirse en dos: Corriendo y encolado. La transición entre estos dos estados es manejado por el itinerador de la máquina virtual.

Nota: Un hilo que invoca al método yield() voluntariamente se mueve a sí misma al estado encolado desde el estado corriendo.

Bloqueada (not Runnable): Se ingresa cuando: se invoca suspend(), el hilo invoca el método wait() de algún objeto, el hilo invoca sleep(), el hilo espera por alguna operación de I/O, o el hilo invoca join() de otro hilo para esperar por su término. El hilo vuelve al estado Corrible cuando el evento por que espera ocurre.

Muerta (Dead): Se llega a este estado cuando el hilo termina su ejecución (concluye el método run) o es detenida por otro hilo llamando al su método stop(). Esta última acción no es recomendada.

El siguiente código puede ser usado para evitar el llamado a stop() y así evitar estados de inconsistencia. Invocando a safeStop() se consigue terminar el hilo la próxima vez que la variable done es chequeada.

```
boolean done = false;
public void run() {
while(!done) {
....
}
}
public safeStop() {
done = true;
}
```

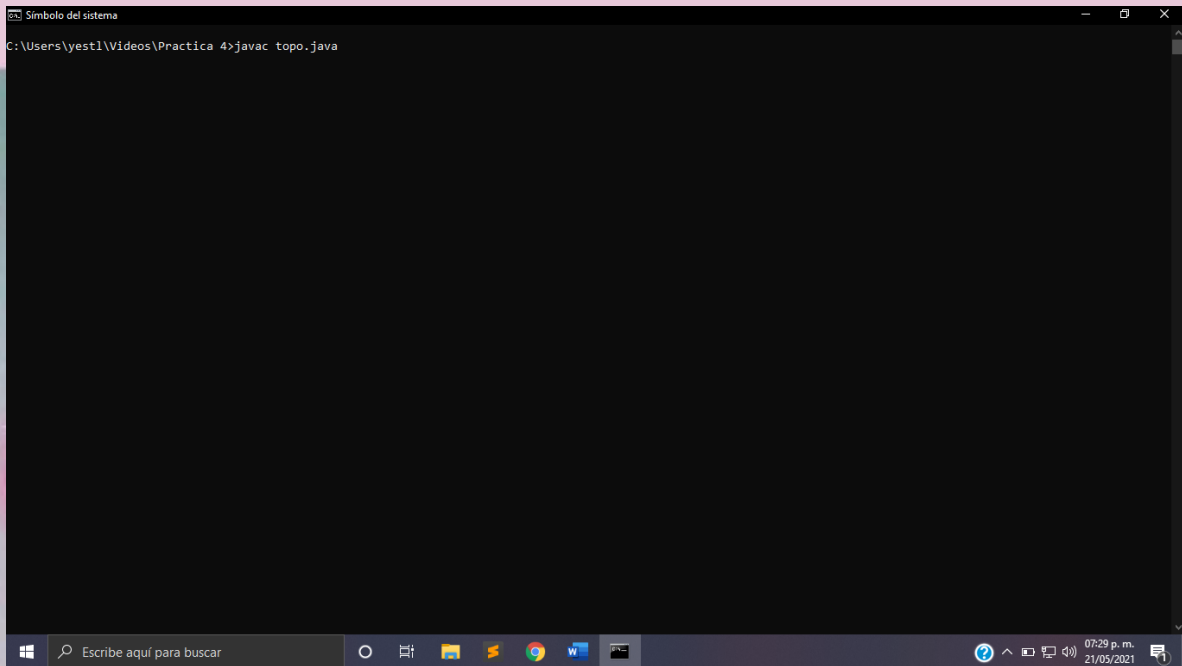
El método isAlive() invocado sobre un hilo, permite saber si aún puede correr.

#### Sincronización de hilos

Todos los hilos de un programa comparten el espacio de memoria, haciendo posible que dos hilos accedan la misma variable o corran el mismo método de un objeto al "mismo tiempo". 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.

Por ejemplo, si en un sistema un método permite hacer un depósito y luego dos hilos lo invocan, es posible que al final sólo un depósito quede registrado al ejecutar las instrucciones en forma traslapadas.

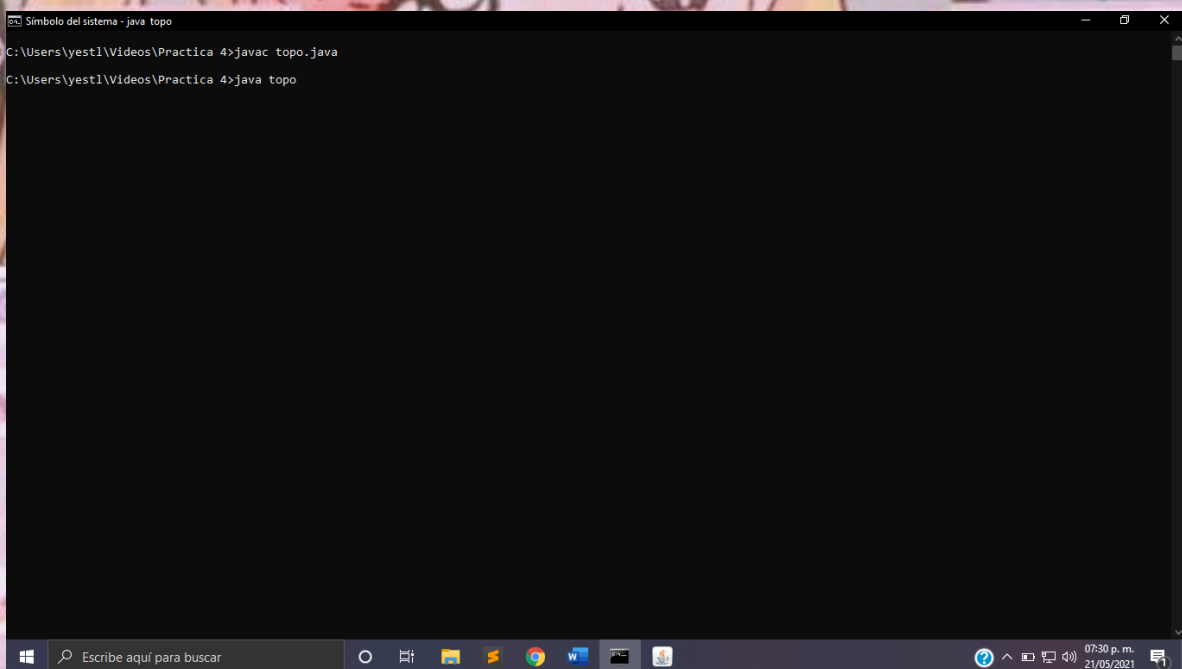
# Desarrollo



A screenshot of a Windows Command Prompt window titled "Símbolo del sistema". The command prompt shows the directory path `C:\Users\yestl\Videos\Practica 4` and the command `javac topo.java` has been entered. The Windows taskbar is visible at the bottom, showing the search bar and several application icons. The system clock in the bottom right corner indicates the time is 07:29 p. m. on 21/05/2021.

```
Símbolo del sistema
C:\Users\yestl\Videos\Practica 4>javac topo.java
```

*Código 1 Compilación del código desde (CMD)*

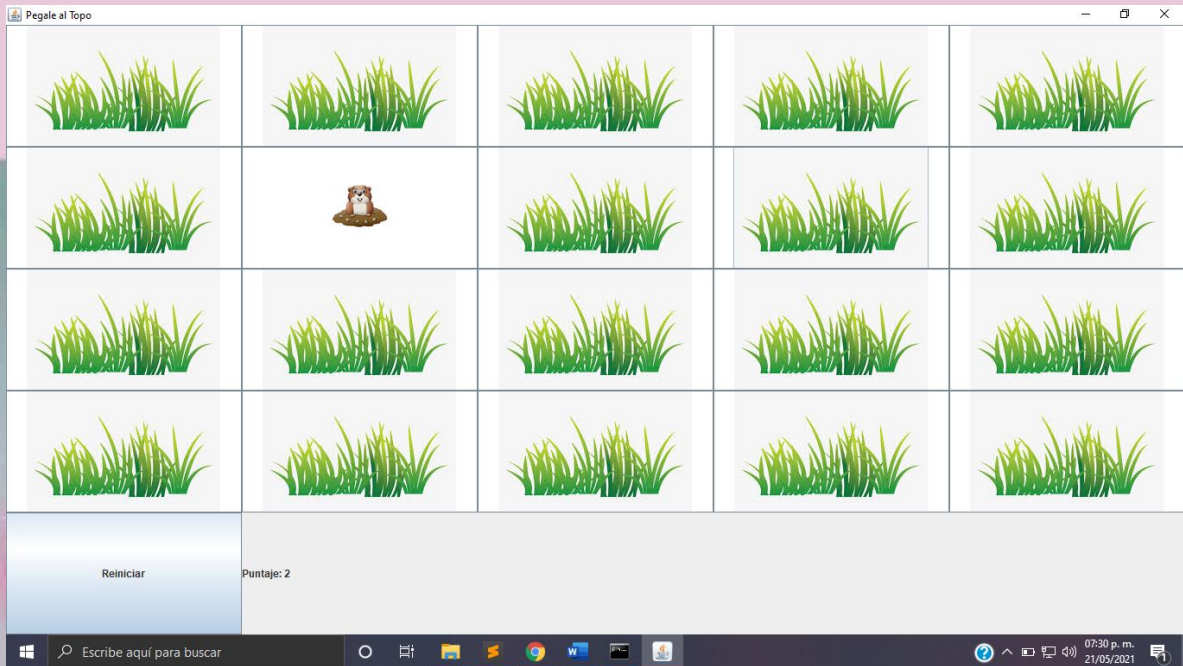


A screenshot of a Windows Command Prompt window titled "Símbolo del sistema - java topo". The command prompt shows the directory path `C:\Users\yestl\Videos\Practica 4` and the command `javac topo.java` has been entered. Below this, the command `java topo` has been entered. The Windows taskbar is visible at the bottom, showing the search bar and several application icons. The system clock in the bottom right corner indicates the time is 07:30 p. m. on 21/05/2021.

```
Símbolo del sistema - java topo
C:\Users\yestl\Videos\Practica 4>javac topo.java
C:\Users\yestl\Videos\Practica 4>java topo
```

*Código 2 Ejecución del código desde (CMD)*





*Código 3 Programa ejecutándose en Java*



## Conclusion

Bocanegra Heziquio Yestlanezi

Al hablar de multi-hilo pudiera parecer que necesitamos más de un procesador para realizar dichas tareas pero no es así, el procesador mismo junto con la máquina virtual de Java gestionan el flujo de trabajo y dan la impresión de que se puede ejecutar más de algún proceso al mismo tiempo (aunque en términos estrictos eso no es posible), de cualquier manera no ahondaré en el funcionamiento del procesador, basta con entender que en Java, 2 o más procesos pueden ejecutarse al mismo tiempo dentro de una misma aplicación y para ello son necesarios los Threads o hilos.

La programación multihilo sin duda tiene mucho campo de aplicación, desde los sistemas operativos hasta en la tecnología que usamos cotidianamente como los celulares, cajeros etc.

En la elaboración de este trabajo obtuvimos conceptos que nos darán la fluidez para desarrollar aplicaciones javas utilizando multihilo, con la finalidad de poder realizar más de una tarea a la vez y administrando correctamente los recursos del ordenador.

Respecto a la práctica, cambiar de Java3D a Java y buscar más librerías fue complejo, sobre todo la parte donde debe de ser aleatorio para que las imágenes del topo vayan saliendo y el juego sea interactivo.

