**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**

**TRABAJOS PRÁCTICOS de SISTEMAS OPERATIVOS**

**TRABAJO PRÁCTICO 06**

**Threads, programación concurrente**

**Grupo**

Debianers

**Integrantes**

Yosshua Cisneros - 179889

Rodrigo Plauchu - 182671

Mauricio Gutiérrez - 183014

**Fecha (s) de elaboración del trabajo práctico**

**16 de Mayo de 2022**

Trabajo Práctico 06

Threads, programación concurrente.

**OBJETIVO**

Usted deberá conservar toda la funcionalidad pedida anteriormente, que usted ya la llevó a cabo en la solución del Trabajo Práctico 05, por lo que tendrá que transportar esta funcionalidad a esta programación multithread.

**INTRODUCCIÓN**

En esta práctica se buscará replicar el trabajo práctico 05 utilizando los Threads de Java para obtener mejores tiempos de ejecución, así como practicar y poner a prueba lo visto en clase. También, buscaremos cómo se visualiza en Process Explorer y lo compararemos con los resultados del trabajo práctico 05.

**DESARROLLO**

Retome la aplicación concurrente (Trabajo Práctico 05, sin pipes), de la generación de estados de cuenta, resolviéndola con *threads* en lugar de *procesos pesados*.

Ahora usted tendrá en *EdoCtaClientesC.java* la clase con el programa principal mientras que *MovimientosClienteC.java* será la clase Thread derivada. Así durante la ejecución tendremos el Thread “padre” que inicia en *main(…)* en *EdoCtaClientesC*, así como los Threads “hijos” *MovimientosClienteC* que elaborarán los estados de cuenta.

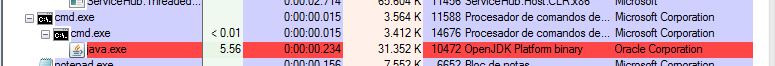
Además, recolecte tres tiempos de ejecución tanto de la aplicación concurrente con procesos pesados (Trabajo Práctico 05) como otros tres de la actual Práctica 08w.

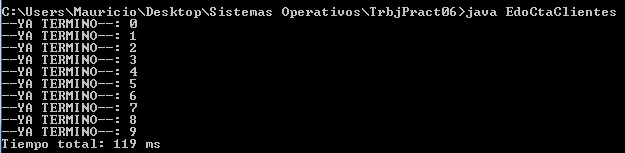
Tome muestras tanto del padre como de algún hijo.

Trabajo Práctico 05: **149 ms, 321 ms, 385 ms**.

Trabajo Práctico 06: **76 ms , 145 ms , 140 ms.**

Con *Process Explorer* capture un “pantallazo” de la ejecución de esta aplicación.





Ahora, en la práctica 06, ¿Cuántos y cuáles son los programas ordinarios que conforman el programa concurrente? **Debido a que un solo proceso genera los threads, se crean 10 threads concurrentes.**

En la misma práctica 06 ¿Cuántos procesos ligeros (threads) fueron ejecutados? **10 procesos ligeros o threads** y ¿Cuántos procesos tradicionales o pesados? **1 sólo proceso pesado Main**

¿De los resultados recolectados como justifica que este programa concurrente con threads es más barato en recursos y más rápido? **Es más rápido ya que cada proceso se ejecuta en un thread diferente.** **Por otro lado, es más barato en recursos porque todos los threads hacen referencia a la misma parte de código, lo único que cambia son los valores de los registros que cada uno de ellos usa.**

**CONCLUSIÓN**

Cada vez tenemos mayor capacidad computacional, pero difícilmente la utilizamos. Por medio de los threads podemos atender procesos de forma concurrente y demandar más recursos al CPU. Con ello podemos aprovechar los recursos de nuestras computadoras, pues en la actualidad los procesadores son multicore y solo así podemos atender realmente al mismo tiempo los diferentes threads.

A su vez, es importante practicar con este tipo de manejo de procesos para la creación de aplicaciones que sean capaces de atender múltiples solicitudes de forma concurrente.