**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**

**TRABAJOS PRÁCTICOS de SISTEMAS OPERATIVOS**

**TRABAJO PRÁCTICO 05**

**Programa Concurrente y procesos cooperantes**

**Grupo**

Debianers

**Integrantes**

Yosshua Cisneros - 179889

Rodrigo Plauchu - 182671

Mauricio Gutiérrez - 183014

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

**4 de Mayo de 2022**

**Trabajo Práctico 05 wz**

**Programa Concurrente y procesos cooperantes.**

**OBJETIVO:** usted desarrollará un programa concurrente, para generar estados de cuenta, por cliente, incluyendo la relación de los movimientos o transacciones de las diferentes cuentas.

Con esta práctica se cubren temas como programa concurrente, procesos concurrentes cooperantes, y paralelismo abstracto entre el proceso padre y los hijos.

Los cuatro archivos, dos *.txt* y dos *.java*, se encuentran en el fólder ***TrbjPracCA05*** de Comunidad.

**INTRODUCCIÓN**

Los programas concurrentes en SO, como se estudió en clase, son programas que se ejecutan paralelamente, y hay 2 tipos: concurrente cooperante y concurrente independiente. En esta práctica observaremos ambos tipos de programas concurrentes, ya que algunos procesos se ejecutarán sin necesidad de responder o esperar a otros porcesos para terminar. El proceso padre deberá ser un programa concurrente cooperativo ya que necesitará que los procesos hijos terminen y envíen la información para finalizar su ejecución.

**DESARROLLO**

Deberá subir a Comunidad, este reporte en un .zip, junto con los programas Java y las correspondientes respuestas. Antes o después que haya subido a Comunidad, en el laboratorio el profesor revisará el funcionamiento de los programas.

Los archivos del fólder son *movimientos.txt*, *clientes.txt*, *EdoCtaClientes.java* (padre) y *MovimientosCliente.java* (hijo). Los archivos Java se encuentran parcialmente programados, pero usted los puede rehacer.

En el archivo de *clientes.txt*, por cada cliente, en cada renglón se tiene la clave del cliente y el nombre del cliente. En el archivo *movimientos.txt* se encuentran las diferentes transacciones o movimientos bancarios realizados por los clientes. En cada renglón viene la clave del cliente y a continuación la transacción.

Su programa padre *EdoCtaClientes.java* deberá crear y ejecutar, por cada cliente, un subproceso *MovimientosCliente.java* que elaborará un archivo *EdoCta<nombredelcliente>.txt* correspondiente al cliente. Así, si uno de los clientes se llama *Laura* entonces el archivo, con su estado de cuenta, se llamará *EdoCtaLaura.txt*.

Puede que en el archivo *movimientos.txt* se encuentre información de movimientos de otros clientes que no están registrados en *clientes.txt*; dichas transacciones no las debe tomar en cuenta.

Como siempre, el proceso padre debe esperar, haciendo otras cosas (ciclar y volver a preguntar), a que los procesos hijos terminen.

También, agregue en cada estado de cuenta, el tiempo total de ejecución expresado en milisegundos, así como el total de transacciones. También, en el proceso padre, despliegue el tiempo total de ejecución en milisegundos.

Después de que la anterior parte funcione, modifique la aplicación considerando que puede darse que algún *cliente* no aparezca en *movimientos.txt*; en este caso no debe generar el archivo del estado de cuenta. El hijo correspondiente deberá “avisarle” al padre que dicho estado de cuenta no se generó y el padre al final, y sólo cuando todos los procesos hayan terminado, deberá desplegar el reporte de aquellos estados de cuenta que no se produjeron, por nombre del cliente.

Además conteste:

¿Cuáles son los programas secuenciales ordinarios que conforman este programa concurrente?

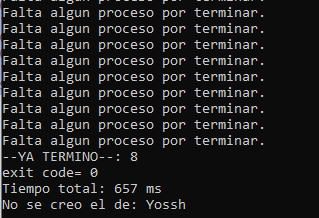
**Los programas secuenciales ordinarios son los de MovimientosCliente.java que crea el padre EdoCtaClientes.java.**

¿Cuántos procesos concurrentes cooperantes interactúan en memoria en este programa concurrente? Expréselo en términos de procesos padres e hijos.

**1 proceso padre y 10 procesos hijos si todos los clientes cuentan con movimientos.**

Agregar imágenes de *ProcessExplorer*, que muestre el árbol jerárquico mostrando varios hijos.





Se obtuvieron las siguientes capturas de pantalla donde se muestran los procesos hijos creados y el reporte generado por el proceso padre al recibir nombres de clientes que no cuentan con ningún movimiento.

**OBSERVACIONES:**

No use PIPEs.

No deberá hacer procesamiento secuencial.

El programa padre *EdoCtaClientes.java* está parcialmente programado; muy poco.

Se recomienda que maneje un arreglo de procesos, tanto para lanzar la ejecución de los procesos hijos así como para esperar su terminación.

El programa hijo *MovimientosCliente.java* es un esqueleto con algún código.

Recuerde que las formas de “comunicarse” son: al lanzar la ejecución del proceso hijo y cuando el hijo termina a través de *exit(…)*;

En el fólder ***TrbjPracCA05/archiv*** vienen una serie de ejercicios en JAVA para manipulación de archivos y fechas. Dentro del fólder, el archivo *inf.txt* le indica que hace cada programa.

**CONCLUSIÓN**

En este trabajo práctico se visualizaron los dos tipos diferentes de procesos concurrentes: independientes y cooperantes. Por un lado, el proceso padre es un proceso cooperante ya que necesita que los procesos hijos envíen información de que el proceso se realizó exitosamente o no. Por otra parte, los procesos hijos son independientes ya que se ejecutan por separado y sólo envían información al padre cuando terminan. La importancia del tipo de procesos concurrentes es importante debido al tiempo que pueden tomar al ejecutarse y a la información que se comparten entre procesos. Debido al procesamiento paralelo de los diferentes procesos concurrentes creados en esta práctica, la diferencia de tiempo no es tan notable, pero para procesos que requieran mayores recursos el tiempo de ejecución podría ser representativo.