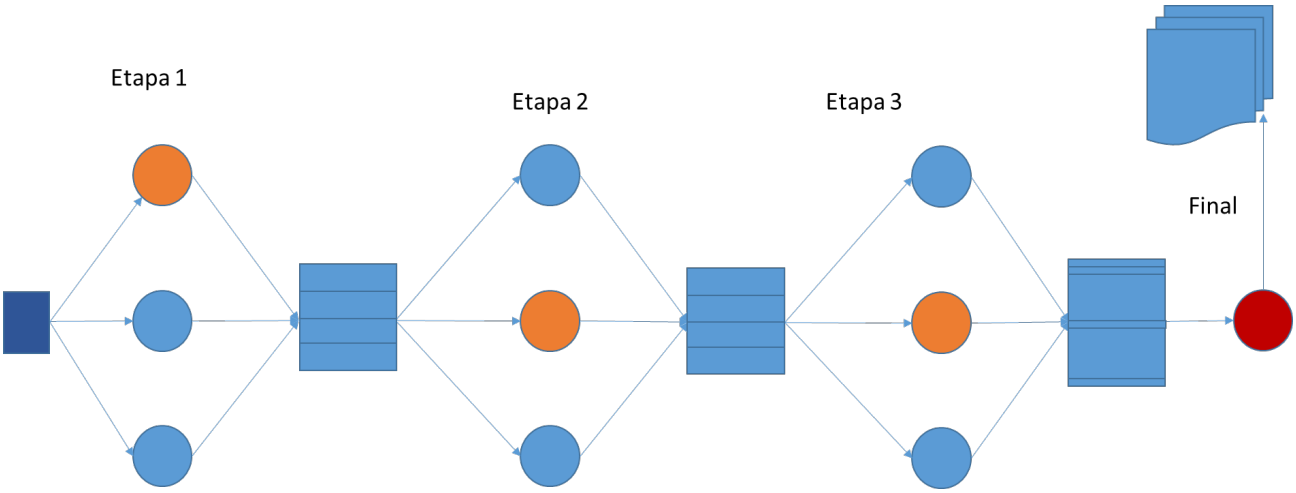
Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
202310

**Caso 1**
**Manejo de la Concurrencia**

En este caso vamos a simular el comportamiento de una planta de producción de dos tipos de productos. El proceso de producción tiene 3 etapas. En la primera etapa se crea el producto y se le asigna un número consecutivo para su identificación (con la ayuda de un objeto que asigna este consecutivo). En la segunda y tercera etapa se transforman para lograr el producto final y se termina la producción. Si bien en el paso de etapa en etapa no es necesario respetar el orden, el proceso final (rojo) debe imprimir en estricto orden los productos generados.




Para la comunicación entre las etapas existe un buffer para que los procesos de la etapa i-1 coloquen ahí los productos, y los procesos de la etapa i los retomen para continuar el proceso. Los primeros buffers tienen un tamaño limitado (especificado por consola), pero el buffer final de donde son recogidos los productos para su entrega (impresión) tiene un tamaño ilimitado. Note que como el buffer final es infinito siempre es posible colocar productos allí y por lo tanto no hay diferencia de comportamiento entre un tipo de proceso y el otro.

Todas las etapas tienen el mismo número de procesos (ingresado por consola): uno que realiza su comunicación con los buffers con espera semiactiva (proceso naranja en la gráfica) y el resto que realizan su comunicación con espera pasiva (procesos azules).

El proyecto debe ser realizado en java, usando *threads*. Para la sincronización solo pueden usar directamente las funcionalidades de Java vistas en clase: `synchronized`, `wait`, `notify`, `notifyAll`, `sleep`, `yield`, `join` y las `CyclicBarrier`.

**Funcionamiento**

Los procesos (naranjas, azules y rojo) se representarán con threads. Los buzones y el objeto identificador serán creados en el programa principal.

Los procesos en la etapa 1 crean todos el mismo número de productos (cantidad especificada por consola). Ese número es conocido por los procesos de las etapas 2, 3 y final. Cada proceso sabe entonces cuántos productos debe procesar.

El acceso a los buzones debe ser controlado para evitar incoherencias en el sistema. Un buzón que está lleno no puede aceptar más mensajes. Un buzón que está vacío no puede entregar mensajes.

- Los procesos naranjas envían de y reciben manera semiactiva (si el buzón donde van a depositar está lleno, o de donde van a retirar está vacío, ceden el procesador, pero vuelven a solicitarlo inmediatamente).
- Los procesos azules envían y reciben de manera pasiva (si no puede completar la acción de envío y/o recepción, esperan a que les avisen que ya pueden realizar la acción).
- El proceso final recibe de manera activa (si el buzón de donde va a retirar no tiene lo que necesita, vuelva a intentarlo inmediatamente).
- Los mensajes dejados en el buzón por un tipo de proceso (naranja/azul) solo deben ser recibidos por procesos del mismo tipo: un proceso azul no debe recibir productos de un proceso naranja y viceversa.

La transformación de un mensaje debe agregar un tiempo de espera para simular esta operación. Esto es, además de efectivamente modificar el mensaje (agregando alguna marca, un thread debe esperar un tiempo aleatorio de entre 50 y 500 milisegundos antes de intentar enviar el mensaje.

El programa debe generar mensajes suficientemente claros para evidenciar el buen funcionamiento de este. En particular debe ser claro que todos los mensajes llegaron a destino, los nodos que intervinieron en la transformación de cada mensaje, las demoras que se aplicaron en cada mensaje para simular su transformación y que no hay mensajes por recibir en el nodo final. Recuerde que su programa será analizado por un tercero que no lo conoce y debe entender fácilmente los resultados.

**Condiciones de entrega**

- En un archivo .zip entregar el código fuente del programa, y un documento Word explicando el diseño (diagrama de clase) y funcionamiento del programa, así como la validación realizada. En particular, para cada pareja de objetos que interactúan, explique cómo se realiza la sincronización, así como el funcionamiento global del sistema. El nombre del archivo debe ser: caso1_*login1*_*login2*_*login3*.zip
- El trabajo se realiza en los grupos definidos en el curso para este caso. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- Habrá una coevaluación del trabajo en grupo. Cada miembro del grupo evaluará a sus dos compañeros y las notas recibidas por un estudiante ponderarán la nota final de caso para ese estudiante.
- En el parcial se incluirá una pregunta sobre el desarrollo de alguna de las funcionalidades del caso. La nota obtenida en esa pregunta puede afectar la nota de todos los miembros del grupo.
- El proyecto debe ser entregado por Bloque Neón por uno solo de los integrantes del grupo. **Al comienzo del documento Word, deben estar los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente, sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- **Se debe entregar por BNe a más tardar el miércoles 22 de febrero a las 23:55 (p.m.)**

**RUBRICA:**

Tengan en cuenta que el código debe correr en los casos de prueba definidos para la sustentación. El funcionamiento y calidad del programa entregado pondera la nota de cada ítem: p.e. no compila; compila, pero no corre; corre, pero no termina, etc.)

Entradas y salidas claras del programa: 10%

Arquitectura de operación: 15%

Comunicación activa (y orden de los resultados): 20%

Comunicación semi-activa: 20%

Comunicación pasiva: 20%

Informe: 15%