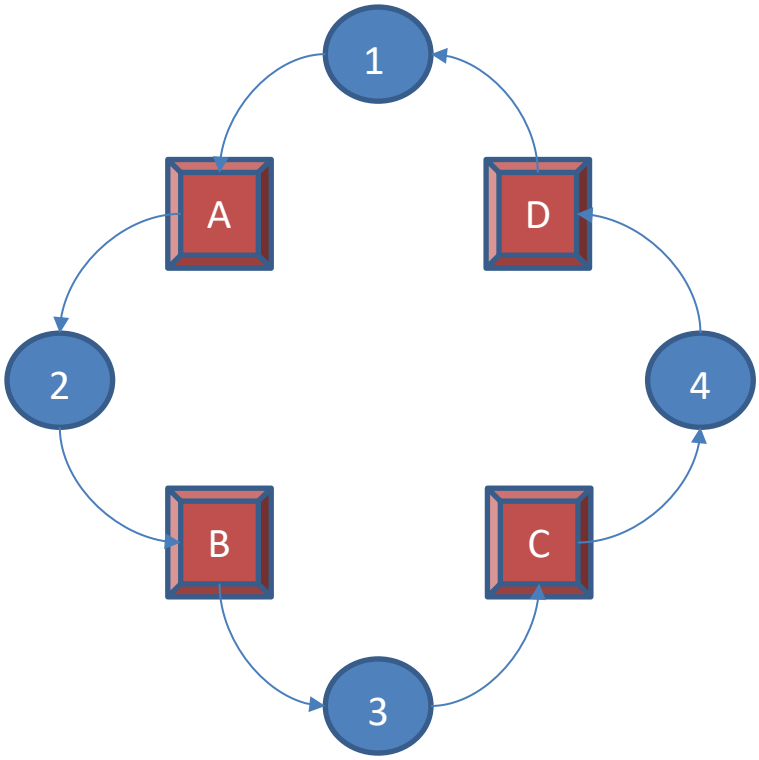
Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
202210

Caso 1
Manejo de la Concurrencia

Vamos a implementar una arquitectura de comunicación circular como la que describe la figura:



Tenemos 4 procesos (threads 1-4) que se comunican a través de 4 buzones (A-D). Tanto los procesos como los buzones son configurables.

- La configuración de los buzones especifica el número de mensajes que puede almacenar (tamaño)
- La configuración de los procesos incluye su velocidad (demoras) y el estilo de comunicación que utiliza tanto para enviar mensajes como para recibirlos (pasivo/activo)
- Un buzón ofrece servicios de almacenamiento y retiro tanto pasivos como activos
- El proceso 1 es el único que sabe cuántos mensajes se van a enviar. Al final envía un mensaje "FIN" que cuando un proceso lo ve pasar, termina su ejecución.

Objetivo

Diseñar un mecanismo de comunicación para implementar la arquitectura descrita. Para este caso, los procesos serán *threads* en la misma máquina (en realidad debería ser un sistema distribuido; este es solo un prototipo). La

comunicación siempre pasa a través de los buzones. Por ejemplo, para comunicar el proceso 1 con el proceso 2, el proceso 1 debe almacenar el mensaje en el buzón A y el proceso 2 debe retirar el mensaje de ese mismo buzón.

El sistema recibe (por teclado) el número de mensajes a transmitir y una vez implementada la arquitectura, los mensajes empiezan a circular desde el proceso 1 hasta regresar al origen. Cuando un proceso recibe un mensaje, le agrega un texto para indicar que el mensaje pasó por ahí. El texto debe incluir el identificador del proceso y una marca para identificar el tipo de envío y recepción que realiza el proceso. Por ejemplo "1AS" sería un texto que podría indicar que el proceso 1 recibió el mensaje de forma PASIVA y lo transmitió de forma ACTIVA.

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

Funcionamiento

El thread que representa al proceso 1 envía los mensajes iniciales y espera la recepción de los mensajes transformados a lo largo del ciclo, imprimiéndolos para verificar el correcto funcionamiento. Una vez ha enviado todos los mensajes, envía un mensaje de "FIN" para indicar a los procesos en el ciclo que deben terminar. Los otros procesos ejecutan un ciclo recibiendo, trasformando y enviando los mensajes. El tipo de recepción y envío (síncrono/asíncrono) debe corresponder con el especificado en un archivo de configuración. Una vez reciben el mensaje de "FIN" terminan su ejecución. El sistema debe avisar, antes de terminar, que ya todos los procesos (threads) han terminado su ejecución.

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. El número de mensajes total a enviar no debe exceder la suma de los tamaños de todos los buzones.

La transformación de un mensaje debe agregar un tiempo de espera para simular esta operación. Esto es, además de efectivamente modificar el mensaje, un thread debe esperar un tiempo antes de poder enviar el mensaje. Este tiempo viene especificado en el archivo de configuración.

Se debe definir la manera de evidenciar que el programa funciona correctamente. La salida debe ser clara para que se puede validar fácilmente la correcta operación del sistema.

La configuración para la ejecución debe estar almacenada en un archivo cuyo formato se presenta a continuación. El número de procesos y buzones siempre es 4.

<id Buzón A> <tamaño buzón A>

…

<id Buzón D> <tamaño buzón D>

<id Proceso 1> <tiempo espera de transformación> <tipo de envío> <tipo de recepción>

…

<id Proceso 1> <tiempo espera de transformación> <tipo de envío> <tipo de recepción>

Un ejemplo de dicho archivo sería:

```
A 5
B 4
C 2
D 4
1 20 true false
2 40 true true
3 60 false true
4 80 false false
```

En este ejemplo los buzones son de tamaño 5, 4, 2 y 4 respectivamente. El tercer proceso debe esperar 60 milisegundos antes de enviar un mensaje que ha recibido y modificado. Este proceso hace una recepción pasiva (false) y hace un envío activo (true).

Condiciones de entrega

- En un archivo .zip entregar el código fuente del programa, y un documento Word explicando el diseño 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*.zip
- El trabajo se realiza en grupos de máximo **2** personas de la misma sección. 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).
- Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota de todos los miembros.
- El proyecto debe ser entregado por BloqueNeón 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 lunes 21 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 impacta la nota de cada ítem.

Configuraciones: 7%

Arquitectura de comunicación: 18%

Comunicación activa: 30%

Comunicación pasiva: 30%

Informe: 15%