**Informe Caso 1**

**Integrantes:**

* Juan Pablo Lora Hernández – 202012524
* Carlos Andrés García Gómez – 202013993
* Johan Sebastián Cáceres Charari –202014171

**Diagrama de clases**

Diagrama

Descripción generada automáticamente

**Entradas del programa**

* Se recibe por consola como entero el número de mensajes a generar y transformar.
* Se recibe por consola como entero el numero asociado a la capacidad de los buzones que van a interactuar con los procesos intermedios.
* Se recibe por consola como entero el numero asociado a la capacidad de los buzones que van a interactuar con los procesos inicial y final.

**Salidas claras**

* Se imprime por consola el arreglo con los mensajes que fueron enviados y transformados, esta lista incluye los ‘fin’ que se generaron cuando el proceso inicial deposito el último mensaje generado.
* El programa, para un mayor entendimiento de los procesos ejecutados también imprime:
  + Cuando un proceso almacena un mensaje.
  + Cuando un proceso extrae(lee) un mensaje.
  + Cuando un proceso transforma un mensaje.

**Arquitectura de comunicación.**

**Explicación de clases:**

Clase Buzon: Está clase representa los buffers que necesitamos para ejercer la comunicación de los diferentes procesos y que reciban los mensajes que se van a transformar

Clase ProcesoInicial: Está clase representa el primer proceso que recibe los mensajes que deseamos transformar y los va a pasar hacia el primer buffer, en este proceso se garantiza la espera semiactiva por medio del método yield. Esta clase hace uso de un método llamado almacenar Semiactivamente, esto con el fin de garantizar que no se ejecutara wait()9omo en el método almacenar.

Clase ProcesoIntermedio: Está clase representa los diferentes procesos de transformación de cada nivel donde se reciben los mensajes y se añade la cadena correspondiente de transformación. Estos procesos realizan espera pasiva, por lo que, si no pueden ejecutarse por falta de recursos, proceden a dormirse y a esperar a ser notificados.

Clase ProcesoFinal: Está clase representa el proceso final donde recibiremos todos los mensajes transformados y el correspondiente indicador de “FIN” donde se indica que las transformaciones han terminado. Aquí se integra también la respuesta final. Este proceso realiza una espera semiactiva por medio de Yield, al momento de extraer un mensaje llama al método .extraerPasivamente con el fin de garantizar que no se ejecute el wait() que está en el método extraer.

**Funcionamiento de la sincronización**

Para la parte de la sincronización principalmente la utilizamos en los métodos críticos de la clase Buzón como almacenar y extraer, estableciendo además de su función general, una espera semiactiva donde para cumplir con las respectivas restricciones.

**FUNCIONAMIENTO GLOBAL DEL SISTEMA**

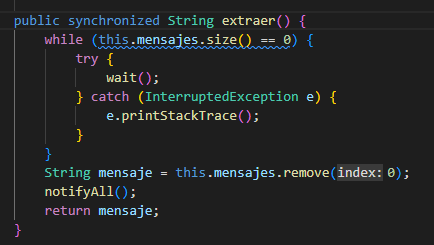
Una vez ingresados los parámetros por consola al programa, se procede a crear los buzones (buffers), se crean los threads (procesos) y se inician. Tenemos 3 tipos de procesos que se inician, “ProcesoInicial”,” ProcesoIntermedio” y “ProcesoFinal” y buffers de 2 tamaños.

El thread “ProcesoInicial” se encarga de crear el número de mensajes que se le ha pasado por parámetro con el formato “Mx”, donde “x” representa un numero para diferenciar cada mensaje. Posteriormente va a almacenarlos en el primer buffer grande; una vez el proceso inicial ha enviado todos los mensajes que ha creado al buffer, va a crear un nuevo mensaje “FIN” y lo enviara también al buffer.

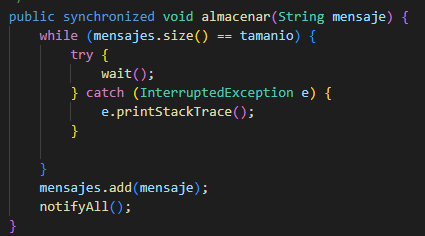
**Comunicación Pasiva Y validación de parejas de objetos que interactúan**

Los threads de proceso intermedio se encargan de transformar los mensajes, pasarlos al buffer consiguiente y notifica. Cuando no pueden ejecutarse, estos procesos realizan una espera pasiva por lo que utilizaran el monitor solo cuando se les sea notificado:

Este es el método extraer, ubicado en el buffer, este método hace uso de synchronized para asegurarse que haya mensajes por extraer, sino hay mensajes procede a esperar a ser notificado, garantizando la espera pasiva.



Por otra parte, tenemos el método almacenar, ubicado en la clase buzón(buffer), este método garantiza la espera pasiva porque verifica si el buzón puede está lleno, si lo está, ejecuta wait() y espera a ser notificado de que el buzón ya no está lleno.

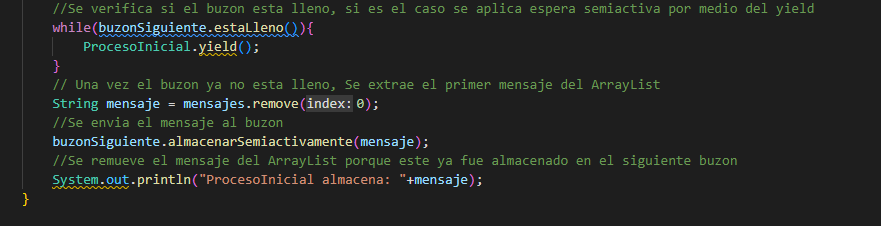


Estos métodos al hacer uso de synchronized en su declaración permiten que java haga uso de un monitor, que nos garantiza la concurrencia entre los procesos de almacenar y extraer mensajes de un buzón.

• Comunicación activa. Y semiactiva

En los procesos inicial y final se debe garantizar una espera semiactiva que está definida como una espera activa pero que después de ejecutarse realiza en java un llamado al método yield que permite ceder el procesador por un instante para después seguir ejecutándose.

En el caso del proceso inicial, nos cercioramos de que el buzón no esté lleno para poder ejecutar el método almacenarSemiactivamente, si el buzón está lleno el Proceso ejecuta el método yield, que garantiza que el procesador es liberado e inmediatamente es vuelto a solicitar para garantizar la parte activa del proceso.



Después de que todos los procesos intermedios terminan su ejecución, en el proceso final existe un arreglo que nos permite ir guardando los mensajes transformados, una vez pasan los tres mensajes de fin, este arreglo con la lista de mensajes transformados es impresa en consola, en este último proceso también evidenciamos espera semiactiva, se comprueba que el buzonanterior no este vacío, si llega a estarlo se ejecuta el método yield para garantizar que el procesador se libera y se vuelve a requerir.

