Estudiantes: Jesús David Barrios 201921887 & Sergio Peñuela 201922873

Universidad de los Andes

Febrero 2022

Caso 1: Infraestructura Computacional

El programa realizado está compuesto por 3 clases Java: Principal, Buzón y Proceso. Principal es la clase que contiene el método main del programa. Se encarga de la lectura del archivo de configuración, la creación y ejecución de las instancias de Buzón y Proceso con sus respectivos atributos, y la lectura y validación del número de mensajes que deben ser enviados en el ciclo proceso-buzón. Además, desde el main, se avisa cuando la ejecución de todos los procesos se acaba, y por lo tanto el programa. Para realizar esto, la clase principal crea una CyclicBarrier, que se incluye como parámetro en la creación de los procesos. Así, se asegura el encuentro de los procesos, y no se avisa del fin del programa hasta que verdaderamente todos los procesos terminaron.

En cuanto a la clase Buzón, se crean 4 instancias siguiendo la descripción del caso. La clase tiene 4 atributos: un identificador, el tamaño que tiene el buzón, un arreglo para almacenar los elementos, y un contador del número de elementos almacenados. Cada buzón funciona como un monitor de los procesos, para asegurar la exclusión mutua de los envíos y recepciones desde y hacia cada buzón. Para esto, los métodos de la clase Buzón tienen la etiqueta synchronized o un bloque sincronizado dentro del método. Adicional al constructor, los getters, y setters, la clase cuenta con 4 métodos: anadirMensajeActivo, anadirMensajePasivo, retirarMensajeActivo y retirarMensajePasivo. Los métodos pasivos son sincronizados por completo (dado que la espera activa saca a los Threads del monitor). Por su parte, en los métodos activos, se utilizan bloques sincronizados, dado que, si se espera activamente y todo el método está sincronizado, ningún otro thread puede entrar al monitor para cambiar la condición que el primero está esperando. En este orden de ideas, solo se sincronizan los cambios a los elementos del buzón, más no la espera.

La espera activa se implementó con un while simple, en el que el thread se quedaba iterando hasta que la condición dejara de cumplirse. Por su parte, para la espera pasiva, dentro del while se incluyó un wait, con lo cual el thread entra a estado dormido hasta ser notificado. Para la notificación, al final de la ejecución de los métodos de recepción y envío, el thread realizaba un notify para avisar al thread que está en espera. La notificación se hacía en los métodos activos y pasivos, dado que no se sabía el tipo de recepción o envío del thread que comparte el buzón.

Por su parte, la clase Proceso, la cual extiende de Thread, tiene 9 atributos: los 4 dados por el archivo de configuración, los dos buzones con los cuales se comunica, la barrera explicada anteriormente, el contador de mensajes (utilizado solo por el proceso 1), y un booleano que establece si el thread es el proceso 1 o no. Adicional al constructor, los getters, y setters, la clase tiene el método run y el método transformarMensajes. En el método transformarMensajes, cada proceso añade 3 caracteres como se establece en el enunciado. Por su parte, en el método run, se maneja la lógica de envío, transformación, y recepción de mensajes mediante los buzones. El proceso 1, teniendo el número de mensajes a enviar, comienza enviando todos los mensajes, y luego envía el mensaje “FIN”. Cuando

los otros Threads reciben el mensaje que contiene la palabra “FIN”, estos transmiten el mensaje y terminan su ejecución (avisan y se detienen en la barrera).

El archivo de configuración es un archivo txt que se encuentra dentro del fichero src dentro de la raíz del proyecto. El nombre del archivo es: casoPrueba.txt. Este sigue la configuración dada en el enunciado:

<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 4> <tiempo espera de transformación> <tipo de envío> <tipo de recepción>

Si se quiere probar el programa con una configuración diferente a la que está en el archivo, se deben cambiar los valores, respetando el formato establecido. Cuando se corre el archivo, se pregunta por el número de mensajes que se deben enviar, y se debe ingresar este número por consola. El programa, en su funcionamiento normal, imprimiría en consola un resultado como a continuación:

```
Lectura realizada correctamente.

Ingrese el número de mensajes que quiere que envíe el proceso 1:
2
Transformando mensaje - Thread: 1 - Tiempo de espera: 20ms - Mensaje: Mensaje 1:  1AS -
Transformando mensaje - Thread: 1 - Tiempo de espera: 20ms - Mensaje: Mensaje 2:  1AS -
Transformando mensaje - Thread: 2 - Tiempo de espera: 40ms - Mensaje: Mensaje 1:  1AS - 2SS -
Transformando mensaje - Thread: 3 - Tiempo de espera: 60ms - Mensaje: Mensaje 1:  1AS - 2SS - 3SA -
Transformando mensaje - Thread: 2 - Tiempo de espera: 40ms - Mensaje: Mensaje 2:  1AS - 2SS -
Transformando mensaje - Thread: 2 - Tiempo de espera: 40ms - Mensaje: FIN 2SS -
Transformando mensaje - Thread: 3 - Tiempo de espera: 60ms - Mensaje: Mensaje 2:  1AS - 2SS - 3SA -
Transformando mensaje - Thread: 4 - Tiempo de espera: 80ms - Mensaje: Mensaje 1:  1AS - 2SS - 3SA - 4AA -
Thread 2 terminó su ejecución.
Transformando mensaje - Thread: 3 - Tiempo de espera: 60ms - Mensaje: FIN 2SS - 3SA -
Transformando mensaje - Thread: 4 - Tiempo de espera: 80ms - Mensaje: Mensaje 2:  1AS - 2SS - 3SA - 4AA -
Thread 3 terminó su ejecución.
Transformando mensaje - Thread: 4 - Tiempo de espera: 80ms - Mensaje: FIN 2SS - 3SA - 4AA -
Thread 4 terminó su ejecución.
Thread 1 terminó su ejecución.
Todos los procesos han terminado correctamente.
```