Santiago Vela -

Verónica Escobar – 201922197

Caso 1: Manejo de Concurrencia

Infraestructura Computacional

2022-1

Contenido

[1. Introducción 1](#_Toc96256267)

[2. Diagrama de solución 1](#_Toc96256268)

[3. Funcionamiento global del Sistema 1](#_Toc96256269)

[3.1 Transmisión de mensajes 1](#_Toc96256270)

[3.2 Perspectiva de un buffer 2](#_Toc96256271)

[3.3 Sincronización entre threads 2](#_Toc96256272)

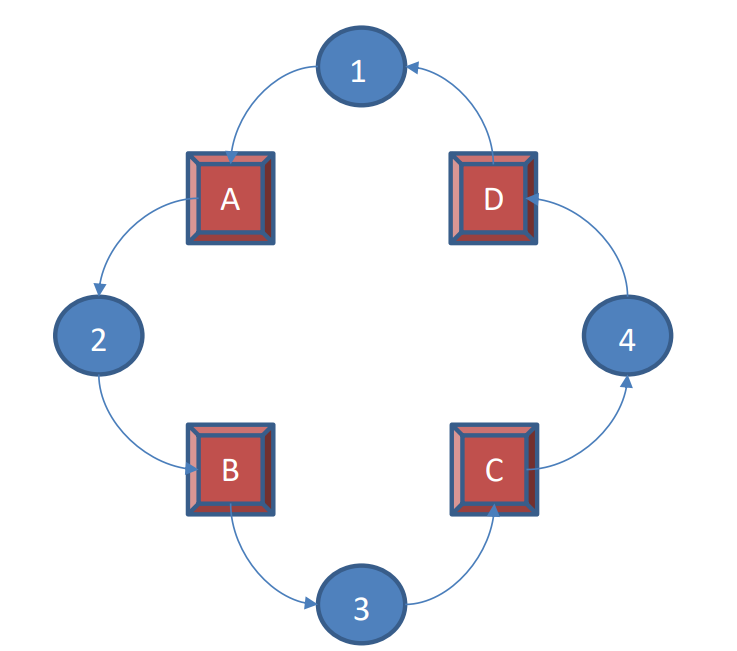
[3.4 Caso especial: Proceso 1 2](#_Toc96256273)

[3.5 Caso especial: Ultimo mensaje (FIN) 2](#_Toc96256274)

[4. Pruebas de funcionamiento 2](#_Toc96256275)

# Introducción

En el presente reporte se documenta el desarrollo e implementación de la simulación de un sistema de mensajes mediante el uso de Threads en un programa en Java.



El problema a solucionar consiste en encontrar una manera que en un sistema con 4 buzones y 4 procesos (threads) que conectan entre si a cada uno de los buzones en un patrón circular (ver figura 1). Cuando se inicia la ejecución el proceso 1 debe mandar el número fijo de mensajes, el cual fue dado por parámetro, al buzón que le sigue a ese thread (en este caso el A). Ya cuando un mensaje está dentro de un buzón este es procesado, y se le añade al final de este un mensaje que indica que ya paso por este buzón. Después, otro thread le pide al buzón sacar el mensaje y decide procesarlo, y al final del mensaje se le añade un mensaje que indica que paso por ese thread y se manda a dormir en thread un tiempo determinado. Eventualmente ese thread/proceso va a pasar el mensaje modificado al siguiente buzón, y el proceso continuara hasta que el mensaje vuelva a llegar al proceso 1. Este proceso se realiza para el número dado de mensajes que se desea dar, y una vez todos los mensajes hayan vuelto a llegar a el proceso 1 se decide mandar un último mensaje llamado fin que es el responsable de matar a todos los thread y terminar la ejecución del sistema de mensajería.

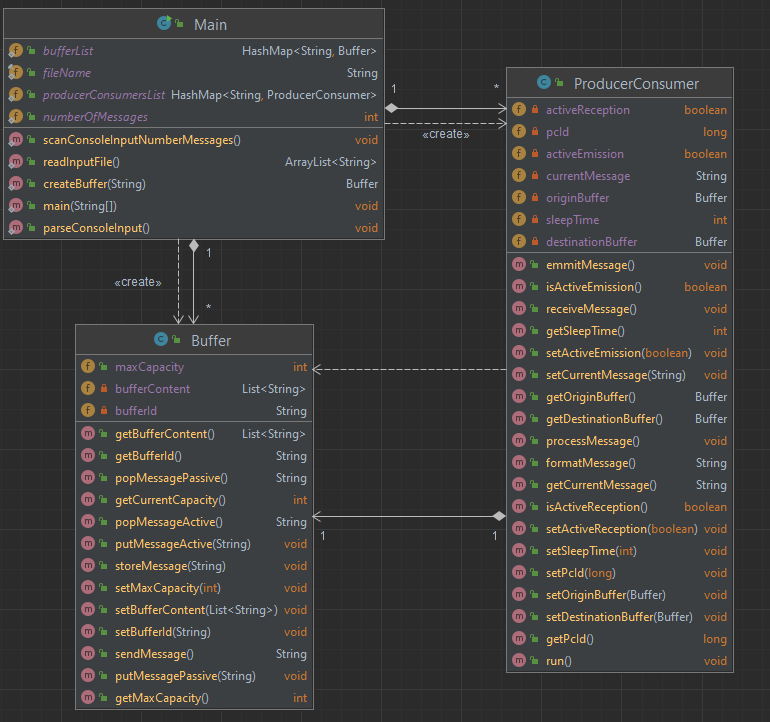
# Diagrama de solución

Para la creación del sistema de mensajería se implementaron 3 clases: Main, ProducerConsumer y Buffer.

En la Main es la responsable de leer la configuración de los buzones (buffers) y los procesos (producerConsumer) e inicializarlos, eventualmente esta misma es responsable de inicializar los threads y autorizar que el proceso 1 sea quien manda el número de mensajes dados por parámetro.

La clase de ProducerConsumer es una mezcla de las clases base de Producer y Consumer del ejemplo dado en clase del problema de Productores y Consumidores. Esta clase tiene la responsabilidad de sacar el mensaje del buzón de origen, procesarlo y eventualmente mandarlo al siguiente buzón.

La última clase del modelo es buffer, es en esta clase que se implementa la mayoría de la sincronización del programa, ya que es aquí que se implementa las funcionalidades de intentar insertar un mensaje (putMessage) y de intentar sacar un mensaje (popMessage) los cuales los threads (ProducerConsumer) que conectan con los buzones ejecutaran para pasar los mensajes. Adicionalmente, buffer tiene métodos específicos par a implementar comunicación activa y pasiva, por eso hay dos versiones de putMessage y popMessage, cada una con el tipo de comunicación que utiliza.



# Funcionamiento global del Sistema

## 3.1 Transmisión de mensajes

* Recibir mensaje
* Procesar mensaje
* Emitir mensaje

## 3.2 Perspectiva de un buffer

## 3.3 Sincronización entre threads

## 3.4 Caso especial: Proceso 1

## 3.5 Caso especial: Ultimo mensaje (FIN)

# Pruebas de funcionamiento