Santiago Rodríguez – 202013371

Juan Andrés Santiago – 201821950

DOCUMENTO DE SOPORTE

1. **Propuesta de diseño (UML):**

La clase **Principal** es la clase principal del programa, que tiene como atributos los parámetros generales de ejecución y el método main (). Se relaciona con un **Buffer (**buzón) y un **Proceso.**

La clase **Buffer** (buzón)recibe, por medio de su constructor, un tamaño inicial asignado y el total de mensajes pendientes por entregar. Además, tiene un método para determinar si el buzón tiene mensajes. Asimismo, tiene métodos de insertar mensaje el cual se divide si es por espera pasiva o espera activa y de igual forma ocurre con los métodos de sacar el mensaje.

La clase **Proceso**, que hereda de *Thread,* recibe por su constructor un identificador, la referencia a un **Buffer de envío** y a un **Buffer de recepción**, la velocidad (tiempo de espera de transformación), el tipo de espera de envio(pasivo-activo), el tiepo de espera de recepción(pasivo-activo) y, por último, el número de mensajes a enviar. Además, sobrescribe el método run ().

1. **Funcionamiento:**

* **Funcionalidad general del programa:**

En el código main del programa, después de inicializar las variables globales del sistema mediante un archivo. Posterior, crea los cuatro buzones con los parámetros respectivos. Asimismo, crea los cuatro procesos con los parámetros respectivos, diferenciando el proceso 1 que es el único que tiene el conocimiento de cuantos mensajes se van a enviar. Finalmente, se le da start a todos los thread y comienza la ejecución.

Texto

Descripción generada automáticamenteEn el siguiente código de run de la clase **Proceso** se revisa que el buffer de envío no esté lleno, debido a que, si esto ocurre, el productor no puede enviar mensaje. En el caso de que el buffer de envío no esté lleno, el proceso procede a enviar los n mensajes al buffer. Por otro lado, se revisa que el buffer de recepción tenga mensajes, debido a que en el caso de que no existan mensajes, no hay nada por recuperar del buffer. En el caso de que existan mensajes, si el tipo de recepción es de espera activa o pasiva. Posterior a esta revisión, se realiza la extracción y transformación del mensaje. Finalmente, se chequea que no se haya puesto el mensaje “FIN” por el proceso 1, ya que eso acabaría el ciclo.

* **Sincronización entre objetos que interactúan:**

En la clase **Buffe**r, en el método de insertMessagePasive() se verifica que el buffer no esté lleno, en el caso de que no, le añade el mensaje y notifica. Después cuando se llene el buffer, se le avisa al proceso y realiza el wait (). Esto ocurre únicamente cuando el tipo de envío del proceso es de **espera pasiva**. En el otro caso (insertMessageActive()), realiza la misma verificación, sin embargo, en la situación que el buffer este lleno, realiza una **espera activa**, preguntando constantemente hasta que se libere espacio.

Texto

Descripción generada automáticamente

Asimismo, en la clase **Buffe**r, en el método de retrieveMessagePasive() se verifica que el buffer no esté vacío y que no haya mensajes pendientes, en el caso que, si ocurra, se realiza el wait(). Posteriormente si hay mensajes pendientes, se remueve el mensaje y se notifica. Al final, retorna null notificando al proceso que no hay más mensajes. Esto ocurre únicamente cuando el tipo de envío del proceso es de **espera pasiva**. En el otro caso (retrieveMessageActive()), realiza la misma verificación, sin embargo, en la situación que el buffer este vacío y haya mensajes pendientes, realiza una **espera activa**, preguntando constantemente hasta que quede vacío.

**Texto

Descripción generada automáticamente**

* **RESULTADO:**