Reporte Caso 1

Infraestructura Computacional

Grupo 12:

Esteban Rico…

Tony Santiago Montes Buitrago – 202014562

Leandro Esteban Yara Ramirez – 202013928

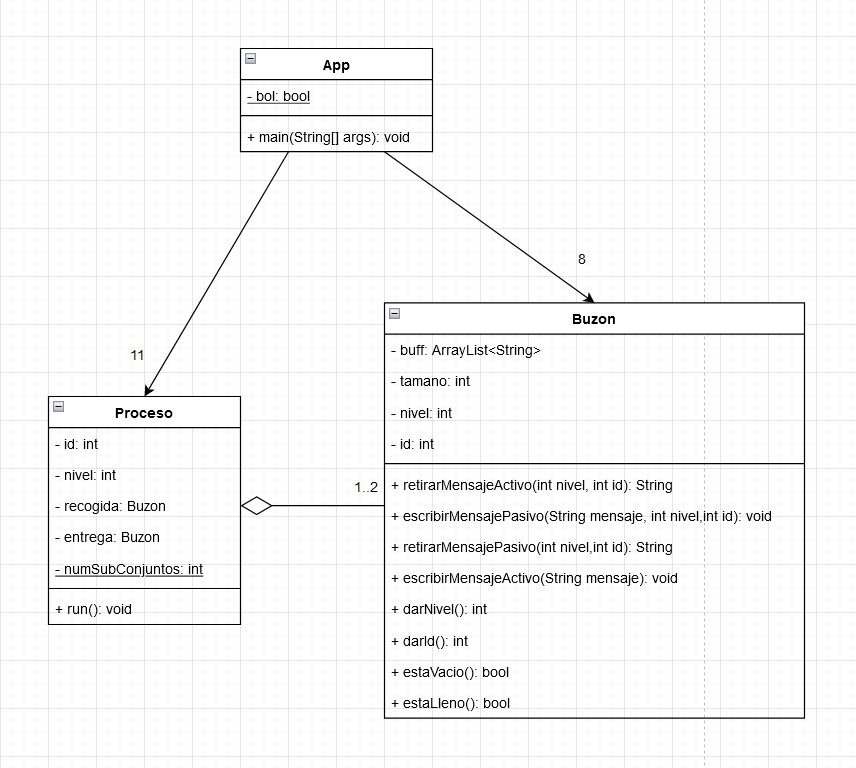
# Entradas del programa

* El número de mensajes a generar (subconjuntos) como un número entero.
* El tamaño de los buzones extremos (inicial y final) como un número entero.
* El tamaño de los buzones intermedios como un número entero.

# Salidas del programa:

* Cada vez que un proceso retira o envía un mensaje de un buzón se imprime una notificación en consola que indica el nivel y el id del thread que realizó la operación junto al mensaje; el formato del mensaje que se imprime es:
* Cuando llegan los mensajes de fin al buzón final se imprime una notificación del tipo contador que muestra cuántos mensajes de “FIN” han llegado a él. Se espera que lleguen 3 mensajes de “FIN” en total; el formato del mensaje de fin que se imprime es:
* Cuando llegan todos los mensajes de “FIN”, se indica que el programa ha terminado y termina de ejecutarse; es decir, se imprime el mensaje:

# Diagrama UML:



# Funcionamiento del programa:

### Inicialización

Al recibir los parámetros de entrada establecidos en el numeral (1.) se crean el buzón inicial y final, luego se inicializan los buzones intermedios en forma de una matriz Buzon[][] esta matriz contiene todos los buzones de los X niveles y Y transformaciones a hacer. Posteriormente, se definen los procesos inicial y final, luego se inicializan los procesos intermedios en forma de una matriz Proceso[][] esta matriz contiene también todos los procesos de los X niveles y Y transformaciones a hacer. Finalmente se corren todos los procesos.

Para la recepción, procesamiento y envío de mensajes se utilizaron 4 métodos en la clase Buzón:

Escribir mensaje pasivo (sincronizado): Recibe como parámetro el mensaje a enviar y el nombre del thread que escribe y revisa si el buzón de salida está lleno. Si es así, queda en estado dormido con el método wait hasta que lo despierten. Cuando sea notificado o si el buzón de salida tiene espacio inicialmente, envía el mensaje al buffer y notifica que se añadió un nuevo elemento.

Escribir mensaje activo: Recibe como parámetro el mensaje a enviar y el nombre del thread que escribe y revisa si el buzón de salida está lleno. Si es así, sale del procesador para darle oportunidad a otro thread, pero solicita inmediatamente el procesador. Hasta que el buzón no esté lleno o si no lo estaba en un principio, entra en un bloque sincronizado sobre el buzón de salida, al cual le envía el mensaje y notifica que hizo el envío.

Retirar mensaje pasivo (sincronizado): Revisa si el buzón de entrada está vacío. Si es así, queda en estado dormido con el método wait hasta que lo despierten. Cuando sea notificado o si el buzón tenía algún elemento, retira el primer elemento del buzón de entrada, lo almacena para su posterior envío y notifica a todos los threads que esperan sobre ese objeto que retiró un mensaje.

Retirar mensaje activo: Revisa si el buzón de entrada está vacío. Si es así, sale del procesador para darle oportunidad a otro thread, pero solicita inmediatamente el procesador. Hasta que el buzón no está vacío o si tenía un objeto al inicio, entra en un bloque sincronizado sobre el buzón para retirar el primer elemento del buzón de entrada almacenarlo para su posterior envío y notificar a todos los threads que esperan sobre ese objeto que retiró un mensaje.

En el método run hay 3 formas en las que un proceso puede correr según su nivel:

Nivel inicial: Esto indica que el proceso es aquel que realiza el envío de mensajes sobre el buzón inicial. Envía mensajes a través del método “Escribir mensaje activo” con el formato “M” + i, donde i es el número del mensaje actual que crece de 1 en 1, hasta que i sea igual al número de mensajes ingresado por el usuario en el literal (1.). Cuando se terminan de generar mensajes normales, se crean y envían al buzón inicial los 3 mensajes de “FIN” para que cada proceso subsecuente sepa cuando termina de ejecutarse.

Nivel 1 a 3: Esto indica que el proceso es intermedio y conecta dos buzones de comunicación. Genera un String para almacenar el mensaje que se va a recibir y, mientras que esa variable de almacenamiento no sea “FIN”, ejecuta el siguiente fragmento:

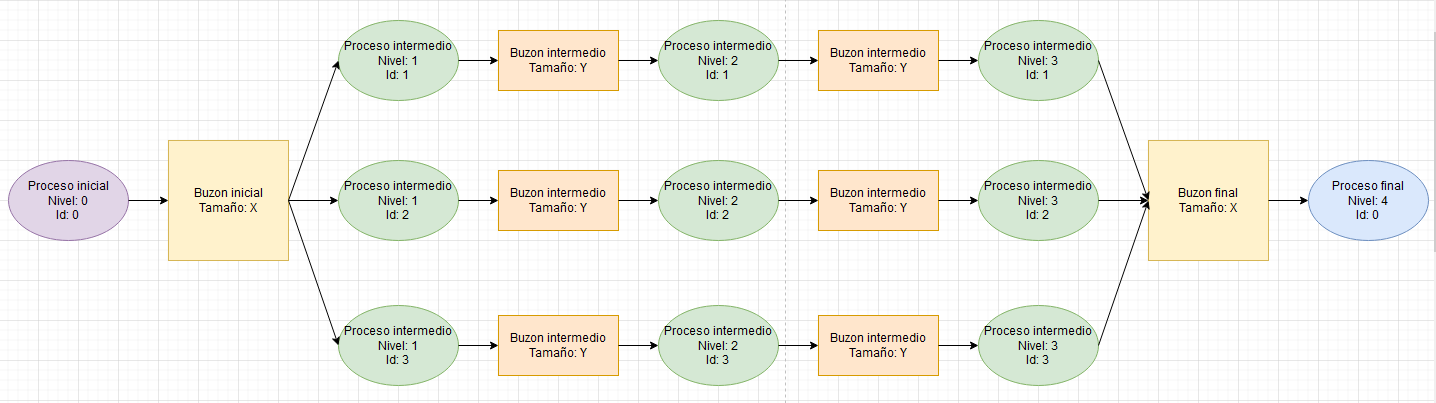
Guarda en la variable de almacenamiento el mensaje recibido por el método “Retirar mensaje pasivo” sobre el buzón de entrada. Si el mensaje recibido no es FIN, realiza la transformación del mensaje para indicar por cuales procesos ha pasado. La modificación consiste en concatenar por la derecha del mensaje original el mensaje “T”NM, donde N es el nivel y M es el id del proceso que está transformando el mensaje. Por último, envía el mensaje modificado al buzón de salida a través del método “Escribir mensaje pasivo”.

Nivel final: Esto indica que el proceso es aquel que retira los mensajes del buzón final y verifica que todos los niveles terminen apropiadamente. Crea un indicador para saber cuantos mensajes de fin han llegado y una variable para guardar el mensaje actual. Mientras la variable de almacenamiento no sea “FIN” o el contador de “FIN” sea menor a 3 se ejecuta el siguiente fragmento:

Se retiran los mensajes del buzón final a través del método “Retirar mensaje activo” y se almacenan en la variable de almacenamiento. Si el mensaje recibido es “FIN”, se aumenta en 1 el contador de “FIN” y se imprime en consola el número actual de fines recibidos.

Cuando finaliza el while (se recibieron los X indicadores de “FIN”) se imprime un mensaje que indica que el programa ha terminado con el procesamiento y envío de mensajes.

# Arquitectura de comunicación:



Morado: Proceso inicial (nivel 0).

Amarillo: Buzones extremos (X es el tamaño de buzones extremos indicado por el usuario en el numeral (1.)).

Verde: Procesos intermedios (niveles 1 a 3).

Naranja: Buzones intermedios (Y es el tamaño de buzones intermedios indicado por el usuario en el numeral (1.)).

Azul: Proceso final (nivel 4).