**UNIVERSIDAD DE LOS ANDES**

**Departamento de ingeniería**

**Ingeniería de Sistemas – Infraestructura computacional**

**Profesor: Harold Castro**

**CASO 1**

**Jonathan Obed Cabanzo Certuche - 201911749**

**William Felipe Méndez Ardila - 202012662**

**2022**

**Funcionamiento global del sistema:**

El programa descrito a continuación tiene como función principal comunicar mensajes entre procesos, llamados Mensajeros, a través de buffers (Buzones) con una capacidad máxima, también definida, de almacenamiento. La definición de las características de los Mensajeros y los Buzones se realiza a través de un archivo **config.properties.**  
El programa funciona de manera que el primer proceso envía una cantidad ***n*** de mensajes a el buzón de entrega correspondiente (Los buzones de entrega y de recogida de cada Mensajero se definen en el constructor de su clase), el Buzón guarda el mensaje recibido y, seguido a ello, el segundo Mensajero retira el mensaje guardado en el buzón, lo modifica, y lo envía a otro buzón para que otro proceso realice el mismo ejercicio. Este proceso lo realiza cada mensajero con cada mensaje hasta que, finalmente, el cuarto Mensajero enviará el resultado final del mensaje a el primer Mensajero y este ultimo imprimirá el resultado, y le dará fin a el programa.

Los mensajeros pueden realizar envíos de mensajes de manera pasiva o activa y, asimismo, pueden sacar mensajes de los buzones de manera pasiva o activa (Estas características se definen en el archivo **config.properties**).  
  
**Configuraciones:**

* caso.buzon“N” : capacidad máxima de mensajes de cada buzón.
* caso.msg(i:)\_tEspera : define el tiempo en milisegundos de espera (demora) para cada Mensajero.
* caso.msg(i:)\_tEnvio : define el tipo de envió del mensajero. **True** para activo, **false** para pasivo.
* caso.msg(i:)\_tEnvio : define el tipo de recepción del mensajero. **True** para activo, **false** para pasivo.

A continuación, un ejemplo del archivo:  
  
Captura de pantalla de computadora

Descripción generada automáticamente

**Diseño y funcionamiento del programa:**

En primer aspecto se define una clase principal llamada **Main** la cual tiene como atributos diferentes listas para todas las características de los Mensajeros y los Buzones, tales como: los mensajeros, los buzones, los nombres de los buzones, tipos de recepción y envío, etc.

Para ello se define un método llamado **cargarDatos()**, el cual extrae los datos del archivo **config** y los añade en cada una de las listas de la siguiente manera:

Captura de pantalla de computadora

Descripción generada automáticamente

Seguido a ello, se define el método **Poblar(),** el cual usa ciclos e inicializa cada Mensajero y Buzón extrayendo los datos de las listas anteriormente definidas. También, se crea el método **iniciar()** el cual le dará inicio con el método **start()** a cada Mensajero como un Thread diferente. Finalmente, se crea un método llamado **enviarMensaje(),** el cual le enviará los primeros mensajes al primer buzón.

El metodo **main()** define el funcionamiento del programa de la siguiente manera:

En primer lugar, se usa el método **cargar()** y **poblar()** anteriormente descritos con el fin de inicializar los Mensajeros y Buzones. Posteriormente, se realiza una petición al usuario para poder definir la cantidad de mensajes que se van a enviar, verificando que no el input no exceda la capacidad máxima de los buzones.

Captura de pantalla de computadora

Descripción generada automáticamente

Además, se utiliza el método **iniciar()** para comenzar la ejecución del **run()** de cada mensajero, y se envían todos los mensajes que se deben pasar al primer Buzón.

Captura de pantalla de computadora

Descripción generada automáticamente

**Comunicación entre Mensajeros y Buzones**

En la clase Mensajero y Buzón ocurre lo siguiente una vez el método principal llama al **start()** de cada mensajero.

El Thread de cada mensajero se estará ejecutando mientras la variable **corriendo** se mantenga en **true.** Esta variable únicamente será **false** cuando el mensajero reconozca el mensaje final, que será “FIN” por definición.

Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Primero, el mensajero invoca el método para recoger el último mensaje de su buzón de recogida. Llama el método pasivo o activo según la definición del mensajero. Esto se ve de la siguiente manera en la clase Buzón:

Cuando se invoca de manera pasiva, el Mensajero debe quedar en espera si no hay mensajes en el buzón, a través del método **wait(),** hasta que el otro mensajero que está manipulando el mismo Buzón realice un **notifyAll()** y pueda entrar a verificar de nuevo si hay mensajes para sacar.

**Texto

Descripción generada automáticamente**

Cuando se invoca de manera activa, el Mensajero pregunta constantemente a través del **while** hasta que finalmente pueda acceder a sacar el mensaje del buzón y realice el **notifyAll()** a los Thread en espera.

Texto

Descripción generada automáticamente

**Estos métodos deben tener una sección critica synchronized para garantizar que un solo mensajero pueda manipular los datos del objeto Buzón (en este caso el número de mensajes del buzón y la lista de mensajes). Por ejemplo, para cada par de mensajeros debe haber un orden. Si un mensajero está sacando un mensaje del buzón, el mensajero que los añade no podrá añadir mensajes al Buzón hasta que el primer mensajero los saque, y viceversa.**

Cuando el mensajero saque el último mensaje del Buzón lo guarda en una variable denominada **msgActual** y termina el método.

Posteriormente, en el **run()** del mensajero verifica con un **if** que este mismo aun este **corriendo**, es decir, si el mensaje actual (que será el mensaje que acaba de sacar previamente del Buzón) contiene un “FIN, esto quiere decir que ya no debe modificar el mensaje, sino enviarlo al siguiente Buzón y acabar el **run().**

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

El método escribir() modificará el mensaje actual (**msgActual**) y le añadirá su id, seguido de una letra (“A”: activo, “P”: pasivo) que representa el valor de sus propiedades de recogida y recepción.   
  
Una vez termine el método **escribir()** se enviará el mensaje al buzón de entrega a través de la función **entregar()**. Esta función realizará un llamado a los métodos de añadir en el Buzón, para que este ultimo guarde los valores del mensaje actual. Su invocación será pasiva o activa según se defina.

Captura de pantalla de computadora

Descripción generada automáticamente

Cuando se invoca de manera activa, el método preguntará constantemente si el Buzón no está lleno, hasta que eventualmente la condición le permita acceder a añadir el mensaje a la lista de mensajes (se le suma uno al condicional porque se asume que el mensaje ya se añadió) y realicé el **notifyAll()** a los Thread en cola.

Captura de pantalla de computadora

Descripción generada automáticamente

Cuando se invoca de manera pasiva, el Mensajero debe quedar en espera si el número de mensajes es igual a la capacidad máxima del Buzón, a través del método **wait(),** hasta que el otro mensajero del mismo Buzón realice un **notifyAll()** y pueda entrar a verificar si el Buzón ya tiene espacio para añadir el mensaje.

Captura de pantalla de computadora

Descripción generada automáticamente

Finalmente, de vuelta a método **main()** de la clase principal **Main**, se establece una barrera como condicional de un **while** para verificar constantemente si el mensajero 1 ya termino de enviar todos sus mensajes, cuando termine de enviarlos se establece la barrera (true) e imprime el resultado final.

Captura de pantalla de computadora

Descripción generada automáticamente

**Calendario

Descripción generada automáticamente con confianza mediaValidación de resultados**

**Imagen de la pantalla de una computadora

Descripción generada automáticamente con confianza baja**

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**