Alumno	Pablo Alcalá-Zamora Bermúdez / Luis Miguel Barrios Torres / Francisco Barbero Vazquez
Asignatura	Programación de Servicios y Procesos
Curso	2ª Dam
Año	**2024-2025**
Título de la entrega	**UA3 - Actividad Evaluable**

Explicación de lo que le pedimos a chat gpt

Queremos crear un programa UDP , este proyecto estará separado en varias partes, primero un servidor que recibirá un token de uno de los tres clientes cliente(el token será una clase que se llamará MiembroToken esta actuara como un booleano que solo un cliente podrá poseer y actuar cuando este sea true) . No se debe cerrar el ciclo es decir una vez llegue al tercer cliente se debe enseñar un mensaje que exprese que era el último y el ciclo debe seguir.

Aparte de esto le hemos proporcionado las diferentes funciones como los datagrams pertenecientes a UDP

INFORME SOBRE SIMULACIÓN DE UNA RED TOKEN RING EN JAVA

Introducción

En este informe explicaremos el funcionamiento del programa desarrollado en Java para simular una red **Token Ring** utilizando sockets UDP. La simulación involucra tres Clientes que reciben y pasan un **token** para obtener permiso de ejecución. El programa está diseñado para ejecutarse en un solo terminal usando **hilos**.

1. Concepto de Token Ring

Una red **Token Ring** es un tipo de red en la que un "token" se pasa de un nodo a otro en un orden fijo. Solo el nodo que posee el token en ese momento puede ejecutar su tarea. Cuando termina, lo pasa al siguiente nodo en la red. Este sistema previene colisiones y garantiza que cada nodo tenga su turno.

En nuestra simulación:

- Cada nodo espera recibir el token en un **puerto UDP** específico.
- Cuando recibe el token, realiza una tarea que consiste en esperar por un tiempo determinado.
- Luego, el Cliente pasa el token al siguiente.
- Cuando el **Miembro 3** recibe el token y termina su tarea, el programa entra en un ciclo.

2. Explicación del Código

El programa consta de dos clases principales:

- **MiembroToken**: Define el comportamiento de cada Cliente.
- **TokenRing**: Crea y ejecuta los Clientes en hilos.

Clase MiembroToken

Esta clase implementa `Runnable` para que pueda ejecutarse en un hilo.

Atributos principales:

- **id**: identifica el cliente (1, 2 o 3).
- **puerto**: puerto en el que el cliente espera recibir el token.
- **tengoToken**: indica si el cliente comienza con el token.
- **soyElUltimo**: indica si es el último cliente, en cuyo caso el token regresa al primer cliente.
- **puertoDestino**: determina a qué nodo enviar el token.

Métodos principales:

- **run()**: Ejecuta el Cliente en un bucle infinito
- **recibirToken**(): Espera recibir el token en su puerto designado.
- **enviarToken**`()`: Envía el token al siguiente Cliente en la red.

Clase TokenRing

Esta clase **inicia los tres Clientes** en hilos simultáneos y les asigna los parámetros adecuados.

Proceso de ejecución:

1. **El primer Cliente tiene el token al inicio**.
2. Cuando un Cliente recibe el token:
    - Simula una tarea esperando `X` segundos.
    - Calcula el tiempo transcurrido desde la última vez que tuvo el token.
    - Pasa el token al siguiente nodo.
3. Cuando el **Miembro 3** finaliza su tarea, **devuelve el token al primer Cliente**(Entrando en un ciclo)**.**

3. Ejecución del Programa

Para compilar y ejecutar el programa en un solo terminal:

```
javac TokenRing.java
java TokenRing
```

Ejemplo de salida esperada:

```
Miembro 1 escuchando en el puerto 10000
Miembro 2 escuchando en el puerto 10001
Miembro 3 escuchando en el puerto 10002
Miembro 1 tiene el token y está ejecutando su tarea...
Miembro 1 terminó su tarea en XXXX ms.
Miembro 1 envió el token al puerto 10001
Miembro 2 recibió el token.
Miembro 2 tiene el token y está ejecutando su tarea...
Miembro 2 terminó su tarea en XXXX ms.
Miembro 2 envió el token al puerto 10002
Miembro 3 recibió el token.
Miembro 3 tiene el token y está ejecutando su tarea...
Miembro 3 terminó su tarea en XXXX ms.
Miembro 3 es el último. Finalizando ejecución.
Este proceso entra en ciclo infinito
```