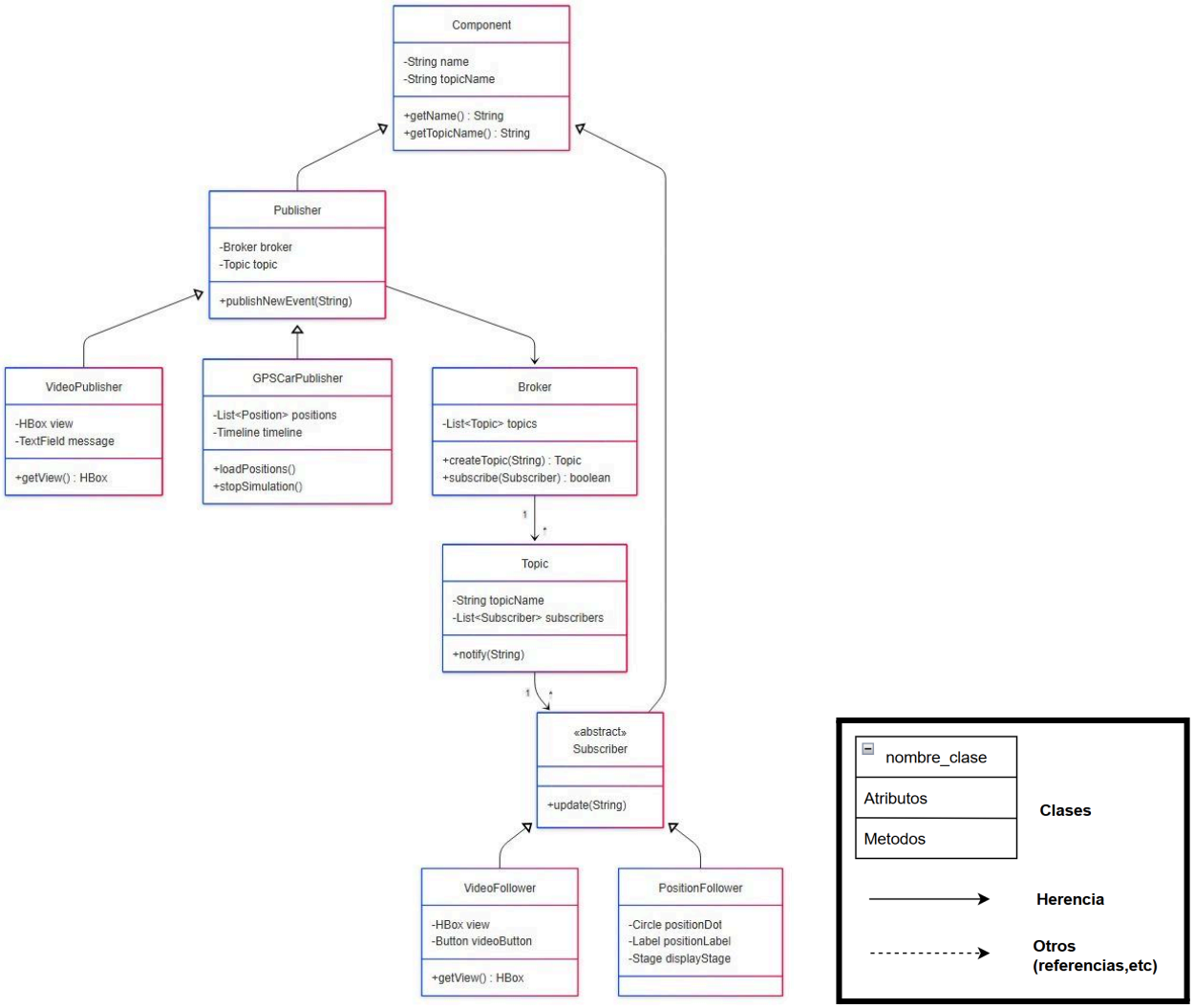
Diagrama UML



Explicación de la solución

Component:
Clase base que proporciona los atributos comunes de nombre y tópico, junto con métodos para su obtención. Sirve como clase padre para Publisher y Subscriber, centralizando el código compartido.

Publisher:
Extiende Component y se encarga de publicar mensajes en un tópico específico. Durante su inicialización, verifica si el tópico existe en el broker y lo crea si es necesario. Mantiene referencias al broker y al tópico.

Subscriber (abstracta):
Clase abstracta que extiende Component y define la interfaz común para todos los suscriptores (VideoFollower y VideoPublisher) mediante el método abstracto `update()`. Proporciona la base para implementar diferentes tipos de procesamiento de mensajes.

Topic:
Representa un canal de comunicación específico que mantiene una lista de suscriptores interesados. Se encarga de notificar a todos los suscriptores cuando recibe un nuevo mensaje.

Broker:
Actúa como intermediario central, gestionando tópicos y coordinando la comunicación entre publicadores y suscriptores. Proporciona métodos para crear tópicos, buscarlos por nombre y registrar suscriptores.

VideoFollower:
Representa un suscriptor especializado en videos dentro del patrón Publisher-Subscriber. Se encarga de recibir URLs de videos, actualizar su interfaz gráfica con el último enlace recibido y proporcionar un botón para su reproducción. Implementa el método update() para procesar los mensajes recibidos del broker.

VideoPublisher:
Publicador especializado en contenido de video que extiende la funcionalidad base de Publisher. Proporciona un campo de texto para que el usuario ingrese URLs de videos y los publique automáticamente al presionar Enter. Mantiene una referencia visual en el panel izquierdo del simulador.

PubSubsPatternSimulator:
Clase principal que inicia y gestiona la aplicación gráfica del simulador. Configura la interfaz de usuario con menús para crear publicadores y suscriptores, paneles de visualización, y diálogos para capturar entradas de usuario. Coordina la interacción entre los componentes del patrón Publisher-Subscriber.

Interacción entre Clases:

El funcionamiento del sistema se basa en las siguientes interacciones clave:

Inicialización del Sistema:

- PubSubsPatternSimulator crea un Broker central y configura la interfaz gráfica.
- Los usuarios pueden crear publicadores (Publisher) y suscriptores (Subscriber) a través de la interfaz.
- Cada publicador (como GPSCarPublisher o VideoPublisher) se registra con un tópico específico mediante el Broker.
- Los suscriptores (como PositionFollower o VideoFollower) se suscriben a un tópico existente a través del Broker.

Publicación de Mensajes:

- Para GPS:
    - GPSCarPublisher carga posiciones desde un archivo y simula el movimiento de un vehículo interpolando coordenadas.
    - Publica mensajes en formato "x,y" periódicamente en el tópico asignado.
- Para Video:
    - VideoPublisher permite al usuario ingresar una URL de video y publicarla en el tópico asociado.
- El Broker gestiona la distribución de mensajes desde el tópico a los suscriptores registrados.

Recepción y Procesamiento:

- Para GPS:
    - PositionFollower recibe las coordenadas y las visualiza en un mapa gráfico, ajustando la posición de un punto en tiempo real.
- Para Video:
    - VideoFollower recibe la URL del video y habilita un botón para reproducir el contenido en una ventana emergente.
- El Topic notifica a todos los suscriptores mediante el método update(), que cada suscriptor implementa según sus necesidades.

Dificultades

1) **Gestión de Tiempo Real en GPS:** La interpolación de posiciones en GPSCarPublisher requirió un manejo preciso del tiempo para simular el movimiento del vehículo de manera fluida.

2) **Sincronización en la Interfaz Gráfica:** Actualizar la UI desde hilos secundarios (como en PositionFollower) implicó el uso de Platform.runLater() para evitar problemas de concurrencia.

3) **Validación de Datos:** La carga de archivos en GPSCarPublisher y la reproducción de video en VideoFollower requirieron manejo de excepciones para casos como archivos corruptos o URLs inválidas.

Soluciones implementadas

A lo largo del desarrollo del sistema, el equipo enfrentó desafíos como la sincronización de datos en tiempo real, la interpolación de coordenadas GPS y la gestión de recursos multimedia. Para resolverlos, se implementó una arquitectura basada en el patrón Publisher-Subscriber, donde el Broker actúa como intermediario centralizado, garantizando un desacoplamiento eficiente entre publicadores (GPSCarPublisher, VideoPublisher) y suscriptores (PositionFollower, VideoFollower). Se optimizó la interpolación de posiciones en GPSCarPublisher para un movimiento suave y se utilizó Platform.runLater() en los suscriptores para actualizar la interfaz gráfica sin conflictos de hilos. Además, se incorporó manejo de excepciones para validar archivos GPS y URLs de video, junto con feedback claro al usuario mediante alertas. Estas soluciones no solo resolvieron los problemas técnicos, sino que también reforzaron el dominio de JavaFX, concurrencia y patrones de diseño, resultando en un sistema modular, escalable y fácil de mantener.