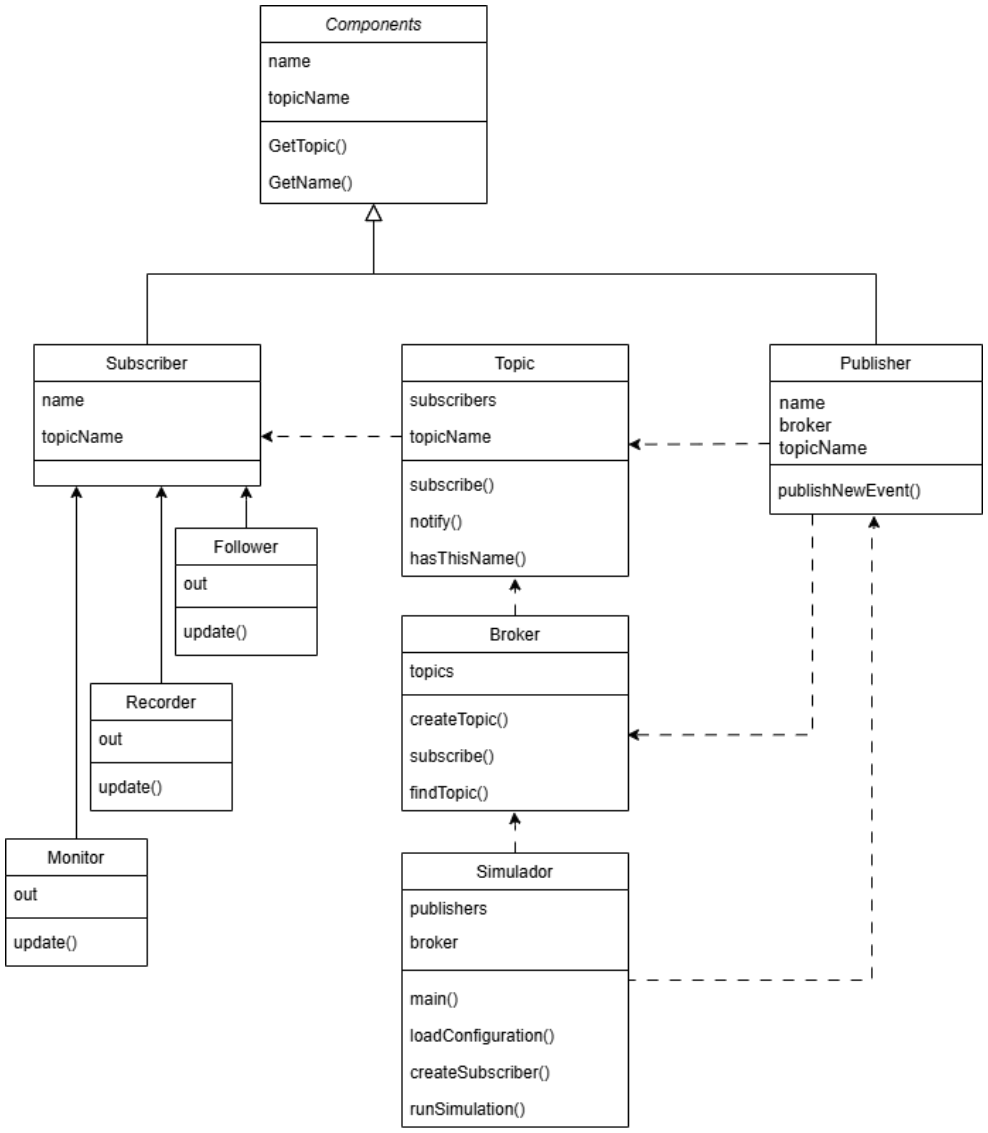
Diagrama UML



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 mediante el método abstracto `update()`. Proporciona la base para implementar diferentes tipos de procesamiento de mensajes.

Follower:
Subclase de Subscriber que simplemente retransmite los mensajes recibidos a un archivo de salida con el formato `<nombre> <tópico> <mensaje>`.

Recorder:
Subclase de Subscriber que extrae dos valores enteros de cada mensaje y los registra en un archivo de salida con el formato `<nombre>,<tópico>,<valor1>,<valor2>`.

Monitor:
Subclase de Subscriber diseñada para vigilar posiciones. Calcula la distancia euclidiana desde el origen usando coordenadas (x,y) y genera alertas cuando esta distancia supera 500 unidades.

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.

Simulador:
Esta es la clase principal que coordina todo el funcionamiento del sistema. Esta clase está encargada de leer las configuraciones de publicadores y suscriptores desde un archivo txt. Se crean los publicadores asociandolos a temas específicos, se crean los suscriptores según su tipo (Follower, Recorder, Monitor) y se registran los suscriptores en los temas correspondientes a través del broker

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

Inicialización del Sistema:
- El Simulador crea un Broker central
- Lee la configuración del archivo y crea Publisher y Subscriber según especificaciones
- Cada Publisher se registra con un tópico a través del Broker
- Cada Subscriber se suscribe a un tópico específico mediante el Broker

Publicación de Mensajes:

- El usuario ingresa mensajes con formato <publicador> <mensaje>
- El Simulador identifica el Publisher por su nombre
- El Publisher invoca su método publishNewEvent() con el mensaje
- El Publisher envía el mensaje al Topic asociado
- El Topic ejecuta notify() para distribuir el mensaje

Recepción y Procesamiento:

- El Topic invoca el método update() de cada Subscriber registrado
- Cada tipo de Subscriber procesa el mensaje según su implementación específica:
    - Follower: Retransmite el mensaje completo
    - Recorder: Extrae y registra valores numéricos
    - Monitor: Analiza coordenadas y genera alertas si superan el umbral

Dificultades

1)Ciclo de dependencias entre Publisher-Broker-Topic

2)Modelado del problema, específicamente en traducir conceptos abstractos a clases concretas.

3)Sintaxis en Java

Soluciones implementadas

A lo largo del desarrollo de la tarea, el equipo logró comprender progresivamente el funcionamiento del sistema Publisher-Subscriber mientras reforzaba su dominio de Java. Inicialmente, la abstracción del patrón generó dudas sobre las interacciones entre componentes, pero mediante la implementación incremental y el análisis de ejemplos prácticos, se clarificó el rol clave del Broker como intermediario y la necesidad de desacoplar los módulos. Paralelamente, el trabajo sirvió para consolidar conceptos fundamentales de Java como herencia y manejo de excepciones.