Relatório Técnico: Sistema de Publicadores e Assinantes com Broker Usando Classes

1. Objetivo do Código

Este código visa implementar um sistema de comunicação do tipo publish/subscribe com uma estrutura de programação orientada a objetos. Os Publicadores (publishers) enviam mensagens para tópicos específicos, e assinantes (subscribers) recebem essas mensagens com base nos tópicos aos quais estão inscritos. Um intermediário (broker) é responsável por gerenciar a comunicação entre publicadores e assinantes. O código foi estruturado em classes que são instanciadas assíncronamente com a biblioteca threading.

2. Estrutura Geral do Código

O sistema foi desenvolvido usando três classes principais: Publisher (publicador), Subscriber (assinante) e Broker (intermediário). Essas classes representam os principais componentes do sistema de mensagens.

3. Classes e Funcionalidades

3.1. Classe Publisher (Publicador)

Função: Envia mensagens para um tópico específico.

Componentes:

- **publisher**: nome do publicador.
- **queue**: fila de mensagens compartilhada entre os publicadores e o broker.
- **lifetime**: ciclo de vida do publicador (número de mensagens que ele enviará antes de encerrar sua execução).

Métodos:

- **__init__**(self, publisher, queue): Inicializa o publicador e inicia sua execução em uma thread separada.
- **create**(self, publisher, queue): Configura o publicador, define o tempo de vida e chama o método publish para enviar mensagens a cada 2 segundos enquanto o tempo de vida for positivo.
- **publish**(self): Seleciona um tópico aleatório e envia uma mensagem para a fila compartilhada.
- **lifetime**(self): Define o tempo de vida aleatoriamente entre 10 e 50 ciclos.
- **select_topic**(self): Escolhe um tópico aleatório.

3.2. Classe Subscriber (Assinante)

Função: Recebe mensagens de tópicos aos quais está inscrito.

Componentes:
- **name**: nome do assinante.
- **list_of_subscribers**: dicionário contendo os tópicos e seus assinantes.
- **channel**: dicionário que mapeia cada assinante à sua fila individual.
- **listening**: indica se o assinante está ativo.
- **topics**: lista de tópicos aos quais o assinante está inscrito.

Métodos:
- **__init__**(self, name, list_of_subscribers, channel, topics): Inicializa o assinante e inicia sua execução em uma thread separada.
- **generate**(self, name, list_of_subscribers, channel, selected_topics): Cria a fila individual do assinante e chama os métodos de inscrição e recebimento de mensagens.
- **subscription**(self, selected_topics): Inscreve o assinante nos tópicos desejados ou escolhe aleatoriamente um tópico se nenhum for especificado.
- **subscribe**(self, topic): Adiciona o assinante ao tópico especificado.
- **recieve**(self): Recebe mensagens do canal associado ao assinante e possui um timeout que encerra a execução do Assinante.

3.3. Classe Broker

Função: Atua como intermediário entre publicadores e assinantes, recebendo mensagens e distribuindo-as para os assinantes corretos.

Componentes:
- **messages_queue**: fila compartilhada que contém todas as mensagens enviadas pelos publicadores.
- **list_of_subscribers**: dicionário que mapeia tópicos aos assinantes inscritos.
- **subscribers_channels**: dicionário que mapeia cada assinante à sua fila individual.

Métodos:
- **__init__**(self, messages_queue, list_of_subscribers, subscribers_channels): Inicializa o broker e inicia sua execução em uma thread separada.
- **create**(self, messages_queue, list_of_subscribers, subscribers_channels): Monitora a fila de mensagens e distribui as mensagens para os assinantes de acordo com os tópicos.
- **read_delegate**(self): Lê uma mensagem da fila compartilhada e delega seu envio.
- **post**(self, message, topic): Envia a mensagem para todos os assinantes do tópico correspondente.

4. Fluxo de Execução

1. Publicadores: Os publicadores criam mensagens associadas a tópicos aleatórios e as enviam para a fila compartilhada. Cada publicador tem um tempo de vida limitado, que define quantas mensagens ele irá postar antes de encerrar sua execução.

2. Assinantes: Os assinantes se inscrevem em tópicos específicos ou selecionam tópicos aleatoriamente. Eles aguardam por mensagens em suas filas individuais e recebem notificações sempre que uma mensagem do tópico ao qual estão inscritos é postada. Se nehuma mensagem for recebida no período de 60 segundos sua execução é encerrada.

3. Broker: O broker fica monitorando a fila de mensagens e, sempre que uma mensagem é recebida, encaminha-a para todos os assinantes inscritos no tópico daquela mensagem.

5. Vantagens e Limitações

Vantagens:

- Uso de Classes: A modularidade foi aumentada com a implementação de classes, tornando o código mais legível e fácil de manter.
- Concorrência: O uso de threads permite que o sistema seja executado de forma concorrente, simulando um sistema distribuído.
- Fila Segura: A utilização de Queue garante que a comunicação entre as threads seja feita de maneira segura e sem condições de corrida.

Limitações:

- Atribuição Aleatória de Tópicos: Tanto publicadores quanto assinantes escolhem tópicos aleatoriamente, o que pode gerar cenários onde publicadores enviam mensagens para tópicos sem assinantes.
- Encerramento: O broker encerra sua execução ao detectar que restam apenas ele e a thread principal, o que poderia ser substituído por um controle mais explícito de término.

5. Melhorias Possíveis

1. Melhor controle das assinaturas: Atualmente, os assinantes podem se inscrever em múltiplos tópicos, mas poderia ser substituído por um controle que permita a atualização das assinaturas dinamicamente.

2. Encerramento mais Robusto: Um mecanismo de controle do encerramento do broker com uma condição de parada ou um sinal de parada seria um pouco mais trabalhoso de implementar, mas seria mais eficiente.

7. Conclusão

O código apresentado é capaz de simular com sucesso o funcionamento do sistema publisher/subscriber adaptando algumas condições para que o programa não execute de forma indeterminada.