LAPORAN PRAKTIKUM

IOT

Roy Sumurung Sigalingging
11323029
D3 Teknologi Informasi

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI

```python
import json
from topic import Topic
from data_classes import BrokerSettings, ClientSettings

class Simulator:
    def __init__(self, settings_file):
        self.default_client_settings = ClientSettings(
            clean=True,
            retain=False,
            qos=2,
            time_interval=10
        )
        self.topics = self.load_topics(settings_file)

    def read_client_settings(self, settings_dict: dict, default: ClientSettings):
        return ClientSettings(
            clean=settings_dict.get('CLEAN_SESSION', default.clean),
            retain=settings_dict.get('RETAIN', default.retain),
            qos=settings_dict.get('QOS', default.qos),
            time_interval= settings_dict.get('TIME_INTERVAL', default.time_interval)
        )

    def load_topics(self, settings_file):
        topics = []
        with open(settings_file) as json_file:
            config = json.load(json_file)
            broker_settings = BrokerSettings(
                url=config.get('BROKER_URL', 'localhost'),
                port=config.get('BROKER_PORT', 1883),
                protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
            )
            broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
            # read each configured topic
            for topic in config['TOPICS']:
                topic_data = topic['DATA']
                topic_payload_root = topic.get('PAYLOAD_ROOT', {})
                topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
                if topic['TYPE'] == 'single':
                    # create single topic with format: /{PREFIX}
                    topic_url = topic['PREFIX']
                    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
                elif topic['TYPE'] == 'multiple':
                    # create multiple topics with format: /{PREFIX}/{id}
                    for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
                        topic_url = topic['PREFIX'] + '/' + str(id)
                        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
                elif topic['TYPE'] == 'list':
                    # create multiple topics with format: /{PREFIX}/{item}
                    for item in topic['LIST']:
                        topic_url = topic['PREFIX'] + '/' + str(item)
                        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
        return topics

    def run(self):
        for topic in self.topics:
            print(f'Starting: {topic.topic_url} ...')
            topic.start()
        for topic in self.topics:
```

```python
    def run(self):
        for topic in self.topics:
            print(f'Starting: {topic.topic_url} ...')
            topic.start()
        for topic in self.topics:
            # workaround for Python 3.12
            topic.join()

    def stop(self):
        for topic in self.topics:
            print(f'Stopping: {topic.topic_url} ...')
            topic.stop()
```

Kode di atas adalah implementasi kelas Simulator yang dirancang untuk menangani simulasi komunikasi berbasis MQTT dengan memuat, mengelola, dan menjalankan topik-topik yang didefinisikan dalam sebuah file JSON. Berikut penjelasan rinci setiap bagian:

Kelas Simulator dimulai dengan metode __init__, yang bertugas menginisialisasi pengaturan klien MQTT menggunakan objek ClientSettings. Objek ini memiliki atribut seperti clean, retain, qos, dan time_interval yang menentukan pengaturan default. Selain itu, metode ini memuat daftar topik untuk simulasi dengan memanggil fungsi _load_topics, yang membaca konfigurasi dari file JSON.

Metode _read_client_settings digunakan untuk membaca dan mengatur pengaturan klien MQTT dari sebuah dictionary (settings_dict). Jika atribut tertentu tidak ditemukan dalam dictionary, metode ini akan menggunakan nilai default yang didefinisikan sebelumnya melalui parameter default.

Metode _load_topics membaca file JSON yang diberikan dan mengonversinya menjadi konfigurasi yang dapat diproses. Konfigurasi ini mencakup pengaturan broker MQTT (melalui objek BrokerSettings) dan daftar topik. Berdasarkan jenis topik yang didefinisikan (single, range, atau list), metode ini akan membuat URL topik dengan format yang sesuai:

- **Tipe single**: Membuat satu topik berdasarkan PREFIX.
- **Tipe range**: Membuat beberapa topik dengan rentang ID, seperti /prefix/1, /prefix/2, dll.
- **Tipe list**: Membuat beberapa topik berdasarkan elemen yang ada di dalam daftar LIST.

Setiap topik yang berhasil dibuat ditambahkan ke daftar self.topics untuk dikelola lebih lanjut.

Metode run bertugas untuk menjalankan simulasi pada semua topik yang ada dalam daftar self.topics. Metode ini mencetak pesan "Starting" dengan URL setiap topik, lalu memanggil metode start pada masing-masing topik untuk memulai proses simulasi. Setelah semua topik dimulai, metode join digunakan untuk memastikan bahwa semua proses selesai dijalankan.

Metode stop bertanggung jawab untuk menghentikan simulasi pada semua topik. Untuk setiap topik, metode ini mencetak pesan "Stopping" dengan URL topik, lalu memanggil metode stop pada masing-masing topik untuk menghentikan prosesnya.

Secara keseluruhan, kode ini memberikan kerangka kerja yang fleksibel untuk mengelola simulasi komunikasi berbasis MQTT. Dengan memuat konfigurasi dari file JSON, sistem ini memungkinkan pengguna untuk dengan mudah mengatur berbagai macam topik dan menjalankannya secara paralel dalam sebuah simulasi.