LAPORAN PRAKTIKUM
INTERNET OF THINGS

MQTT

Van Christ John Hutajulu
11323026
DIII 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 mengimplementasikan kelas Simulator yang bertanggung jawab untuk memuat, mengelola, dan menjalankan topik-topik yang dikonfigurasi melalui file JSON. Berikut adalah penjelasan dari setiap bagian:

Kelas Simulator dimulai dengan metode __init__, yang menerima file pengaturan (settings_file) sebagai argumen. Di dalam metode ini, pengaturan default klien diatur dengan membuat objek ClientSettings yang berisi atribut seperti clean, retain, qos, dan time_interval dengan nilai default. Kemudian, metode load_topics dipanggil untuk memuat daftar topik.

Metode read_client_settings berfungsi untuk membaca pengaturan klien dari dictionary yang diberikan. Metode ini memeriksa keberadaan setiap atribut dalam dictionary (settings_dict), dan apabila ada yang tidak ditemukan, nilai dari pengaturan default digunakan.

Metode load_topics membaca file pengaturan yang diberikan dalam format JSON, kemudian mengonfigurasi pengaturan broker menggunakan objek BrokerSettings. Setelah itu, metode ini membaca setiap topik yang ada dalam file JSON dan berdasarkan tipe topik (single, multiple, atau list), topik-topik tersebut dibuat dengan format URL yang sesuai dan ditambahkan ke dalam daftar topics. Tipe single membuat satu topik dengan PREFIX, tipe multiple membuat beberapa topik dalam rentang ID, dan tipe list membuat beberapa topik berdasarkan daftar LIST.

Metode run berfungsi untuk menjalankan seluruh topik yang ada dalam daftar. Untuk setiap topik, metode ini mencetak pesan "Starting" dengan URL topik yang sedang dijalankan, kemudian memanggil metode start untuk memulai proses topik tersebut. Setelah semua topik dimulai, metode join dipanggil untuk memastikan bahwa semua proses selesai.

Metode stop bertanggung jawab untuk menghentikan seluruh topik. Untuk setiap topik, metode ini mencetak pesan "Stopping" dengan URL topik, kemudian memanggil metode stop untuk mengakhiri proses topik tersebut.

Secara keseluruhan, kode ini mengatur cara simulator memuat pengaturan dari file JSON, mengelola topik-topik berdasarkan konfigurasi yang diberikan, dan menjalankan atau menghentikan topik-topik tersebut sesuai kebutuhan.