LAPORAN PRAKTIKUM
IOT

Aan Kristian Sitinjak
11323033
D-III TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
2024/2025

```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()
```

Aan Sitinjak 033

Kelas TopicSimulator adalah program yang dibuat untuk mempermudah pengelolaan dan simulasi berbagai topik yang didefinisikan dalam sebuah file JSON. Inti dari program ini adalah memuat daftar topik dari file konfigurasi, mengatur formatnya, lalu menjalankan simulasi atau menghentikannya sesuai kebutuhan.

Ketika kelas ini pertama kali dijalankan, ia membaca file konfigurasi untuk mengatur pengaturan broker (seperti alamat host dan port) serta daftar topik. Topik-topik ini dapat berupa:

Topik tunggal (single) – membuat satu topik sederhana.
Topik berjangka (multiple) – menghasilkan beberapa topik berdasarkan rentang ID tertentu.
Topik berbasis daftar (list) – membuat beberapa topik berdasarkan elemen dalam sebuah daftar.
Selain itu, simulator juga memiliki pengaturan klien bawaan seperti clean, retain, qos, dan interval yang dapat diubah sesuai kebutuhan. Jika pengaturan tambahan tidak diberikan, simulator akan menggunakan nilai bawaan.

Program ini memiliki dua fungsi utama:

Menjalankan simulasi – Memulai semua topik dalam daftar dan memberikan informasi tentang topik mana yang sedang berjalan.
Menghentikan simulasi – Mengakhiri semua topik yang sedang aktif.
Misalnya, jika sebuah file JSON berisi pengaturan untuk beberapa topik dengan berbagai tipe, simulator akan memprosesnya dan menghasilkan daftar topik yang siap untuk disimulasikan. Selama simulasi berjalan, Anda akan melihat informasi seperti:
Salin kode
Simulasi dimulai untuk topik: topic/single
Simulasi dimulai untuk topik: topic/multiple/1
Simulasi dimulai untuk topik: topic/list/item1
Program ini sangat berguna jika Anda membutuhkan alat sederhana untuk mengelola banyak topik sekaligus, terutama untuk pengujian atau simulasi sistem berbasis topik seperti broker MQTT. Ia dirancang agar fleksibel, mudah dipakai, dan dapat diandalkan untuk berbagai skenario simulasi.

Aan Sitinjak 033