LAPORAN PRAKTIKUM

IOT

MQTT Simulator

RICKY JOSUA SILAEN

11323028

DIII-TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI DEL

FAKULTAS VOKASI

24/25

Code Program

```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()
```

Penjelasan:

Kode di atas adalah implementasi kelas `Simulator` yang bertugas untuk memuat, mengelola, dan menjalankan topik-topik yang dikonfigurasikan dalam sebuah file JSON. Berikut penjelasan setiapbagian:

Kelas `Simulator` dimulai dengan metode `_init__`, yang diinisialisasi dengan file pengaturan(`settings_file`). Metode ini mengatur pengaturan klien default dengan membuat objek
`ClientSettings`, di mana atribut seperti `clean`, `retain`, `qos`, dan `time_interval`

ditentukan nilaiawalnya. Setelah itu, ia memuat daftar topik dengan memanggil metode `load_topics`.

```
def __init__(self, settings_file):
    self_default_client_settings = ClientSettings()
    self.topics = self.load_topics(settings_file)
```

Metode `read_client_settings` digunakan untuk membaca pengaturan klien dari dictionary pengaturanyang diberikan. Metode ini memeriksa apakah setiap atribut ada di dictionary (`settings_dict`), dan jika tidak, akan menggunakan nilai dari pengaturan default yang diberikan (`default`).

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

Metode `load_topics` memuat file pengaturan yang diberikan, membaca isinya sebagai JSON, lalu mengatur konfigurasi broker menggunakan objek `BrokerSettings`. Setelah itu, metode ini membaca setiap topik yang terdefinisi dalam file JSON. Berdasarkan tipe topik (`single`, `multiple`, atau `list`),ia membuat topik-topik dengan format URL yang sesuai dan menambahkannya ke dalam daftar
`topics`. Tipe `single` membuat satu topik berdasarkan `PREFIX`, tipe `multiple` membuat beberapatopik dengan rentang ID, dan tipe `list` membuat beberapa topik berdasarkan item dalam daftar
`LIST`.

```
def load_topics(self, settings_file):
    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 v3.11
    )

    topics = []
    for topic in config['TOPICS']:
        topic_data = topic['DATA']
        topic_type = topic['TYPE']
        topic_prefix = topic['PREFIX']

        if topic_type == "single":
            topic_url = f"{topic_prefix}"
            topics.append(Topic(topic_url))
        elif topic_type == "multiple":
            for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
                topic_url = f"{topic_prefix}/{id}"
                topics.append(Topic(topic_url))
        elif topic_type == "list":
            for item in topic['LIST']:
                topic_url = f"{topic_prefix}/{item}"
                topics.append(Topic(topic_url))

    return topics
```

Metode `run` bertugas menjalankan semua topik dalam daftar. Untuk setiap topik, ia mencetak pesan"Starting" dengan URL topik, lalu memanggil metode `start` untuk memulai proses topik. Setelah semua topik dimulai, metode `join` dipanggil untuk memastikan prosesnya selesai.

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

    for topic in self.topics:
        topic.join()
```

Metode `stop` bertanggung jawab menghentikan semua topik. Untuk setiap topik, ia mencetak pesan"Stopping" dengan URL topik, lalu memanggil metode `stop` untuk mengakhiri prosesnya.

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

Kode ini mengatur bagaimana simulator memuat pengaturan dari file JSON, mengelola topik-topikberdasarkan konfigurasi yang diberikan, lalu menjalankan atau menghentikannya sesuai kebutuhan.