PROJECT NO.1

학번 : 2020254016
이름 : 박민우

1. 과제 내용

Durable Rule 패키지를 사용하여 현업의 문제 해결을 위한 규칙을 15개 이상 만들어서 실행될 수 있는 시스템을 구성하시오.

2. 시스템 구성

1) Durable_Rule 패키지 설치

```
!pip install durable_rules
```

```
Collecting durable_rules
  Downloading https://files.pythonhosted.org/packages/91/f2/3194b4eaf7260bae0a4f046dfdda5db2ed9d58b3d7a051e375b3c14d8f26/durable_rules-2.0.28.tar.gz (57kB)
     |████████████████████████████████| 61kB 2.8MB/s
Building wheels for collected packages: durable-rules
  Building wheel for durable-rules (setup.py) ... done
  Created wheel for durable-rules: filename=durable_rules-2.0.28-cp37-cp37m-linux_x86_64.whl size=176355 sha256=b56380949f995c49764ee8c01e6b70b77f0c53e6657f6
  Stored in directory: /root/.cache/pip/wheels/97/f5/6b/dabbc5d4c2571374b0e0c49d8c80c449220d3061c20b057ba4
Successfully built durable-rules
Installing collected packages: durable-rules
Successfully installed durable-rules-2.0.28
```

2) 시스템 구성

```
from durable.lang import *

with ruleset('company'):
    @when_all((m.predicate == '컨트롤러') & (m.object == '수정'))
    def color(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '색변환값', 'object': '수정필요' })

    @when_all((m.predicate == '리플렉터') & (m.object == '재설계'))
    def light(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '빛', 'object': '퍼짐' })

    @when_all(c.first << (m.predicate == '컨트롤러') & (m.object == '수정'),
              (m.predicate == '리플렉터') & (m.object == '재설계') & (m.subject == c.first.subject))
    def led(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '구형 LED 모듈에 맞게', 'object': '설계됨' })

    @when_all((m.predicate == '디자인') & (m.object == '재설계'))
    def design(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '외형', 'object': '투박함' })

    @when_all((m.predicate == 'Line레이저 길이') & (m.object == '조절'))
    def distance(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'Line레이저 길이', 'object': '거리에 따라 길어짐' })

    @when_all((m.predicate == 'Line레이저 길이') & (m.object == '거리에 따라 길어짐'))
    def angle(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'Line레이저 길이', 'object': '쏘는 각도에 따라 길어짐' })

    @when_all((m.predicate == '안테나위치') & (m.object == '수정'))
    def signal(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '전파강도', 'object': '개선필요' })

    @when_all((m.predicate == '전파강도') & (m.object == '개선필요'))
    def sw(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'S/W', 'object': '수정' })

    @when_all((m.predicate == '리시버') & (m.object == '사이즈수정'))
    def receiver(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '리시버', 'object': '폭 큼' })

    @when_all(+m.subject)
    def output(c):
        print('Fact: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.object))

assert_fact('company', { 'subject': '무영등', 'predicate': '컨트롤러', 'object': '수정' })
assert_fact('company', { 'subject': '무영등', 'predicate': '리플렉터', 'object': '재설계' })
assert_fact('company', { 'subject': '무영등', 'predicate': '디자인', 'object': '재설계' })
assert_fact('company', { 'subject': '레이저프리젠터', 'predicate': 'Line레이저 길이', 'object': '조절' })
assert_fact('company', { 'subject': '레이저프리젠터', 'predicate': '안테나위치', 'object': '수정' })
assert_fact('company', { 'subject': '레이저프리젠터', 'predicate': '리시버', 'object': '사이즈수정' })
```

3. 결과

```
Fact: 무영등 색변환값 수정필요
Fact: 무영등 컨트롤러 수정
Fact: 무영등 구형 LED 모듈에 맞게 설계됨
Fact: 무영등 빛 퍼짐
Fact: 무영등 리플렉터 재설계
Fact: 무영등 외형 투박함
Fact: 무영등 디자인 재설계
Fact: 레이저프리젠터 Line레이저 길이 쏘는 각도에 따라 길어짐
Fact: 레이저프리젠터 Line레이저 길이 거리에 따라 길어짐
Fact: 레이저프리젠터 Line레이저 길이 조절
Fact: 레이저프리젠터 S/W 수정
Fact: 레이저프리젠터 전파강도 개선필요
Fact: 레이저프리젠터 안테나위치 수정
Fact: 레이저프리젠터 리시버 폭 큼
Fact: 레이저프리젠터 리시버 사이즈수정
{'$s': 1, 'id': 'sid-0', 'sid': '0'}
```