Fizz Buzz

Table of Contents

1. 仕様

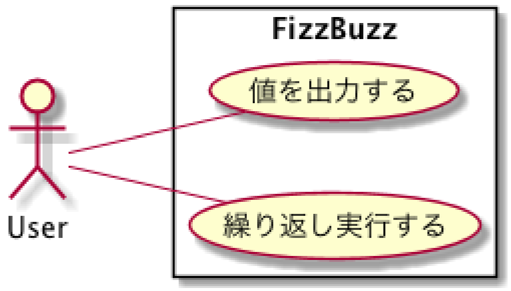
- ３で割り切れる場合は「Fizz」を出力する。
- ５で割り切れる場合は「Buzz」を出力する。
- 両者で割り切れる場合は「FizzBuzz」を出力する。
- 上記以外の場合は与えられた数字を出力する。
- 指定された回数だけ繰り返し実行する。

2. 設計

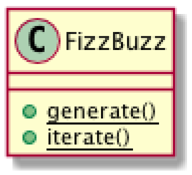
2.1. TODOリスト

- [x] 「Fizz」を出力できるようにする
- [x] 「Buzz」を出力できるようにする
- [x] 「FizzBuzz」を出力できるようにする
- [x] 繰り返し実行できるようにする

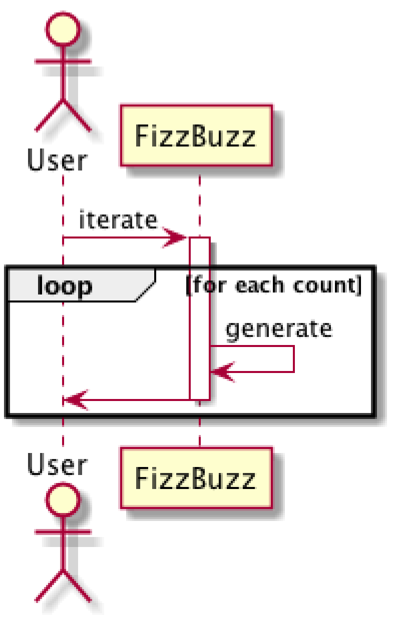
2.2. ユースケース図



2.3. クラス図



2.4. シーケンス図



3. 実装

3.1. テストコード

```
import pytest
from fizz_buzz.fizz_buzz import FizzBuzz


class TestFizzBuzz(object):
    def test_3ならばFizzを返す(self):
        assert FizzBuzz.generate(3) == 'Fizz'

    def test_6ならばFizzを返す(self):
        assert FizzBuzz.generate(6) == 'Fizz'

    def test_5ならばBuzzを返す(self):
        assert FizzBuzz.generate(5) == 'Buzz'

    def test_10ならばBuzzを返す(self):
        assert FizzBuzz.generate(10) == 'Buzz'

    def test_50ならばBuzzを返す(self):
        assert FizzBuzz.generate(50) == 'Buzz'

    def test_15ならばFizzBuzzを返す(self):
        assert FizzBuzz.generate(15) == 'FizzBuzz'

    def test_30ならばFizzBuzzを返す(self):
        assert FizzBuzz.generate(30) == 'FizzBuzz'

    def test_1ならば1を返す(self):
        assert FizzBuzz.generate(1) == 1

    def test_101ならば101を返す(self):
        assert FizzBuzz.generate(101) == 101

    def test_5回繰り返し実行ならば配列を返す(self):
        assert FizzBuzz.iterate(5) == [1, 2, 'Fizz', 4, 'Buzz']

    def test_10回繰り返し実行ならば配列を返す(self):
        assert FizzBuzz.iterate(10) == [1, 2, 'Fizz', 4, 'Buzz', 'Fizz', 7, 8, 'Fizz',
'Buzz']
```

```
import json
import pytest
from fizz_buzz import app


@pytest.fixture()
def apigw_event():
    """ Generates API GW Event"""

    return {
        "body": "{ \"count\": \"5\"}",
        "resource": "/{proxy+}",
        "requestContext": {
            "resourceId": "123456",
            "apiId": "1234567890",
            "resourcePath": "/{proxy+}",
            "httpMethod": "POST",
            "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
            "accountId": "123456789012",
            "identity": {
                "apiKey": "",
                "userArn": "",
                "cognitoAuthenticationType": "",
                "caller": "",
                "userAgent": "Custom User Agent String",
                "user": "",
                "cognitoIdentityPoolId": "",
                "cognitoIdentityId": "",
                "cognitoAuthenticationProvider": "",
                "sourceIp": "127.0.0.1",
                "accountId": ""
            },
            "stage": "prod"
        },
        "queryStringParameters": {
            "number": "3"
        },
        "headers": {
            "Via":
            "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
            "Accept-Language":
            "en-US,en;q=0.8",
            "CloudFront-Is-Desktop-Viewer":
            "true",
            "CloudFront-Is-SmartTV-Viewer":
            "false",
            "CloudFront-Is-Mobile-Viewer":
            "false",
            "X-Forwarded-For":
            "127.0.0.1, 127.0.0.2",
            "CloudFront-Viewer-Country":
            "US",
            "Accept":

 "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
            "Upgrade-Insecure-Requests":
            "1",
            "X-Forwarded-Port":
            "443",
            "Host":
            "1234567890.execute-api.us-east-1.amazonaws.com",
            "X-Forwarded-Proto":
            "https",
            "X-Amz-Cf-Id":
            "aaaaaaaaaae3VYQb9jd-nvCd-de396Uhbp027Y2JvkCPNLmGJHqlaA==",
            "CloudFront-Is-Tablet-Viewer":
            "false",
            "Cache-Control":
            "max-age=0",
            "User-Agent":
            "Custom User Agent String",
            "CloudFront-Forwarded-Proto":
            "https",
            "Accept-Encoding":
            "gzip, deflate, sdch"
        },
        "pathParameters": {
            "proxy": "/examplepath"
        },
        "httpMethod": "POST",
        "stageVariables": {
            "baz": "qux"
        },
        "path": "/examplepath"
    }


def test_3ならばFizzを返す(apigw_event):
    ret = app.generate(apigw_event, "")
    assert ret['statusCode'] == 200

    for key in 'value':
        assert key in ret['body']

    data = json.loads(ret['body'])
    assert data['value'] == 'Fizz'


def test_繰り返しならば配列を返す(apigw_event):
    ret = app.iterate(apigw_event, "")
    assert ret['statusCode'] == 200

    for key in 'values':
        assert key in ret['body']

    data = json.loads(ret['body'])
    assert data['values'] == [1, 2, 'Fizz', 4, 'Buzz']
```

3.2. プロダクトコード

```
class FizzBuzz:
    @staticmethod
    def generate(number):
        value = number

        if value % 3 == 0 and value % 5 == 0:
            value = 'FizzBuzz'
        elif value % 3 == 0:
            value = 'Fizz'
        elif value % 5 == 0:
            value = 'Buzz'

        return value

    @staticmethod
    def iterate(count):
        array = []

        for n in range(count):
            array.append(FizzBuzz.generate(n + 1))

        return array
```

```
import json

from fizz_buzz import FizzBuzz


def generate(event, context):
    """FizzBuzz generate Lambda function
    Arguments:
        event LambdaEvent -- Lambda Event received from Invoke API
        context LambdaContext -- Lambda Context runtime methods and attributes
    Returns:
        dict -- {'statusCode': int, 'body': dict}
    """
    try:
        number = 0

        if 'queryStringParameters' in event:
            number = int(event['queryStringParameters']['number'])

        body = json.dumps({
            'value': FizzBuzz.generate(number)
        })

        print("Application execute with params:" + str(number))
        return __create_response(200, body)
    except Exception as err:
        err_msg = 'Application error occurred:' + str(err.args)
        print(err_msg)
        body = json.dumps({
            'message': err_msg
        })
        return __create_response(500, body)


def iterate(event, context):
    """FizzBuzz iterate Lambda function
    Arguments:
        event LambdaEvent -- Lambda Event received from Invoke API
        context LambdaContext -- Lambda Context runtime methods and attributes
    Returns:
        dict -- {'statusCode': int, 'body': dict}
    """
    try:
        params = json.loads(event['body'])
        count = 0

        if 'count' in params:
            count = int(params['count'])

        body = json.dumps({
            'values': FizzBuzz.iterate(count)
        })

        print("Application execute with params:" + str(params))
        return __create_response(200, body)
    except Exception as err:
        err_msg = 'Application error occurred:' + str(err.args)
        print(err_msg)
        body = json.dumps({
            'message': err_msg
        })
        return __create_response(500, body)


def __create_response(status_code, data):
    return {
        "statusCode": status_code,
        "body": data,
        "headers": {
            'Content-Type': 'application/json',
            'Access-Control-Allow-Origin': '*',
        }
    }
```

4. 参照

- Asciidoctor[http://asciidoctor.org/]
- PlantUML[http://www.plantuml.com]