Fizz Buzz

Table of Contents

1. 仕様

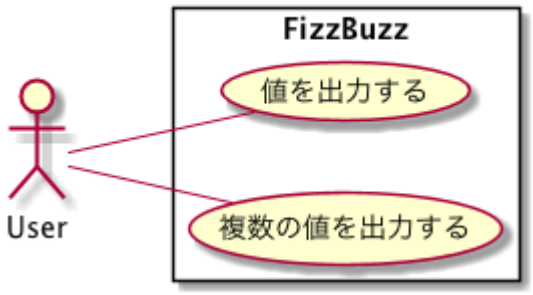
- ３で割り切れる場合は「Fizz」を出力する。
- ５で割り切れる場合は「Buzz」を出力する。
- 両者で割り切れる場合は「FizzBuzz」を出力する。
- 上記以外の場合は与えられた数字を出力する。
- 指定された回数だけ繰り返し実行する。

2. 設計

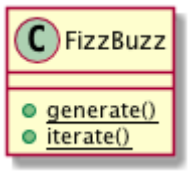
2.1. TODOリスト

- [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

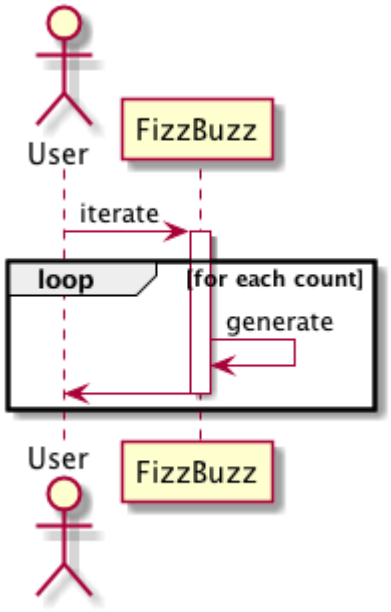
2.2. ユースケース図



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

```
/* eslint-disable import/no-extraneous-dependencies */
const chai = require('chai');
const FizzBuzz = require('../../src/fizz_buzz');

const { expect } = chai;

describe('Tests FizzBuzz', () => {
  it('3ならばFizzを返す', () => {
    expect(FizzBuzz.generate(3)).to.equal('Fizz');
  });

  it('6ならばFizzを返す', () => {
    expect(FizzBuzz.generate(6)).to.equal('Fizz');
  });

  it('5ならばBuzzを返す', () => {
    expect(FizzBuzz.generate(5)).to.equal('Buzz');
  });

  it('10ならばBuzzを返す', () => {
    expect(FizzBuzz.generate(10)).to.equal('Buzz');
  });

  it('50ならばBuzzを返す', () => {
    expect(FizzBuzz.generate(50)).to.equal('Buzz');
  });

  it('15ならばFizzBuzzを返す', () => {
    expect(FizzBuzz.generate(15)).to.equal('FizzBuzz');
  });

  it('30ならばFizzBuzzを返す', () => {
    expect(FizzBuzz.generate(30)).to.equal('FizzBuzz');
  });

  it('1ならば1を返す', () => {
    expect(FizzBuzz.generate(1)).to.equal(1);
  });

  it('101ならば101を返す', () => {
    expect(FizzBuzz.generate(101)).to.equal(101);
  });

  it("5ならば[1, 2, 'Fizz', 4, 'Buzz']を返す", () => {
    expect(FizzBuzz.iterate(5)).to.eql([1, 2, 'Fizz', 4, 'Buzz']);
  });
});
```

```
/* eslint-disable import/no-extraneous-dependencies */

const chai = require('chai');
const app = require('../../app.js');

const { expect } = chai;

describe('Tests FizzBuzzFunction', () => {
  let event;

  beforeEach(() => {
    event = {
      body: '{}',
    };
  });

  it('3ならばFizzを返す', async () => {
    event = {
      queryStringParameters: { number: '3' },
    };
    const result = await app.generate(event);

    expect(result).to.be.an('object');
    expect(result.statusCode).to.equal(200);
    expect(result.body).to.be.an('string');
    expect(result.body).to.equal('Fizz');
  });

  it("5ならば[1, 2, 'Fizz', 4, 'Buzz']を返す", async () => {
    event = {
      body: '{"count": "5"}',
    };
    const result = await app.iterate(event);

    expect(result).to.be.an('object');
    expect(result.statusCode).to.equal(200);
    expect(result.body).to.be.an('array');
    expect(result.body).to.eql([1, 2, 'Fizz', 4, 'Buzz']);
  });
});
```

3.2. プロダクトコード

```
module.exports = class FizzBuzz {
  static generate(number) {
    let value = number;
    if (number % 3 === 0 && number % 5 === 0) {
      value = 'FizzBuzz';
    } else if (number % 3 === 0) {
      value = 'Fizz';
    } else if (number % 5 === 0) {
      value = 'Buzz';
    }
    return value;
  }

  static iterate(count) {
    const array = [];
    for (let i = 0; i < count; i += 1) {
      array.push(FizzBuzz.generate(i + 1));
    }
    return array;
  }
};
```

```
/* eslint-disable no-console,max-len,prefer-destructuring */

const FizzBuzz = require('./src/fizz_buzz');

const createResponse = (statusCode, body) => ({
  statusCode,
  body,
  headers: {
    'Content-Type': 'application/json',
    'Access-Control-Allow-Origin': '*',
  },
});

exports.generate = async (event) => {
  let number;
  let data;
  if (event.queryStringParameters !== null && event.queryStringParameters !==
undefined) {
    if (event.queryStringParameters.number !== null &&
event.queryStringParameters.number !== undefined) {
      number = event.queryStringParameters.number;
    }
  }

  try {
    data = FizzBuzz.generate(number);
  } catch (err) {
```

```
    console.log(`Application error occurred: ${err}`);
    return createResponse(500, err);
  }

  console.log(`Application execute with params: ${number}`);
  return createResponse(200, data);
};

exports.iterate = async (event) => {
  const params = {
    Item: JSON.parse(event.body),
  };
  let data;

  try {
    data = FizzBuzz.iterate(params.Item.count);
  } catch (err) {
    console.log(`Application error occurred: ${err}`);
    return createResponse(500, err);
  }

  console.log(`Application execute with params: ${JSON.stringify(params.Item)}`);
  return createResponse(200, data);
};
```

4. 参照

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