Objectifs de la feuille

- Écriture de test unitaires (TDD & BDD)
- Mise en place d'indicateurs de couverture de tests

Exercice 1 *Test Driven Development*

1.1 Exemple de TU en Javascript:

- création du dossier du tp : **`mkdir but3-tp-tdd`**
- Se mettre dans le dossier : **`cd but3-tdd`**
- Initialisation du projet : **`npm init -y`**
- Installation de jest : **`npm install --save-dev jest`**
- Modification de **`package.json`** dans la section **`scripts`** :

```
"scripts": {
  "test": "jest"
}
```

- Création dossier **`src`** et **`__tests__`** :
  - **`mkdir src`**
  - **`mkdir __tests__`**

- Dans **`src`** créer le fichier **`sum.js`** suivant :

```
function sum(a, b) {
    return a + b;
}

module.exports = sum;
```

- Dans **`__tests__`** créer le fichier **`sum.test.js`** suivant :

```
const sum = require('../src/sum');

test('adds 1 + 2 to equal 3', () => {
    expect(sum(1, 2)).toBe(3);
});
```

- Lancer le test avec **`npm test`**

```
> but3-tdd@1.0.0 test
> jest

 PASS  __tests__/sum.test.js
  ✓ adds 1 + 2 to equal 3 (1 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.191 s, estimated 1 s
```

1.2 C'est à vous ! Écrire un programme qui retourne les entiers de 1 à 100 en prenant en compte les spécifications suivantes :

- Pour les multiples de 3, remplacez le nombre par "Fizz"
- Pour les multiples de 5, remplacez le nombre par "Buzz"
- Pour les multiples de 15, remplacez le nombre par "FizzBuzz"

Les tests unitaires suivants sont fournis :

```
const fizzBuzz = require('../src/fizzbuzz');

test('Pour n allant --> 8', () => {
    expect(fizzBuzz(8)).toEqual([1, 2, "Fizz", 4, "Buzz", "Fizz", 7, 8]);
});

test('Pour n allant --> 15', () => {
    expect(fizzBuzz(15)).toEqual([1, 2, "Fizz", 4, "Buzz", "Fizz", 7, 8, "Fizz", "Buzz", 11, "Fizz", 13, 14, "FizzBuzz"]);
});
```

1.3 Un dernier pour la route ! Écrire un programme qui permet d'écrire N étape de la suite de CONWAY. Les tests unitaires suivants sont fournis

```
const conway = require('../src/conway');

test('Pour n allant --> 2', () => {
    expect(conway(2)).toEqual(["1", "11"]);
});

test('Pour n allant --> 4', () => {
    expect(conway(4)).toEqual(["1", "11", "21", "1211"]);
});
```

1.4 Vos résultats de tests doivent ressembler à ceci :

```
> but3-tdd@1.0.0 test
> jest

 PASS  __tests__/conway.test.js
 PASS  __tests__/fizzbuzz.test.js
 PASS  __tests__/sum.test.js

Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.399 s, estimated 1 s
```

- Qu'en est-il de la couverture de nos tests ? **`npx jest –coverage`**
- Si vous voulez obtenir une version html :

  `npx jest --coverage –coverageReporters=html`
- Un répertoire coverage contenant un fichier index.html vous attend.

1.5 Rajoutez un test pour obtenir 100 % de couverture sur la suite de CONWAY.

Exercice 2 *Behavior Driven Development*

Cucumber est un outil open source utilisé pour faire du Behavior Driven Development (BDD). Il permet de décrire les comportements attendus d'une application sous forme de scénarios écrits en langage naturel, accessibles à tous les membres de l'équipe (développeurs, testeurs, chefs de projet, utilisateurs finaux, etc.). Les scénarios **Cucumber** sont écrits dans un format appelé **Gherkin**, qui utilise une syntaxe simple et lisible pour décrire les étapes d'un scénario. Chaque scénario est composé de trois parties :

- **Given** : décrit l'état initial du système avant que l'utilisateur n'effectue une action.
- **When** : décrit l'action effectuée par l'utilisateur.
- **Then** : décrit le résultat attendu après l'action de l'utilisateur.

2.1 Ajout de **Cucumber** sur le projet

- Ajout de **Cucumber** : **`npm install --save-dev @cucumber/cucumber`**
- Ajout d'un nouveau script dans **`package.json :`**

```
"scripts": {
    "test": "jest",
    "behavior": "npx cucumber-js"
  }
```

- Création du fichier à la racine **`cucumber.js`** avec le contenu suivant

```
module.exports = {
    default: {
    }
};
```

- Création dossier **`features`** à la racine
- Création dossier support : **`features/steps`**
- Dans **`features`** créer le fichier **`sum.feature`** suivant :

```
Feature: Sum two numbers

Scenario: Add two numbers
    Given I have a sum function
    When I add 1 and 2
    Then the result should be 3
```

- Dans **`steps`** créer le fichier **`sum-steps.js`** suivant :

```
const { Given, When, Then } = require('@cucumber/cucumber');
const assert = require('assert');
const sum = require('../../src/sum');

let result;

Given('I have a sum function', function () {
    // This step is more for context, no action needed
});

When('I add {int} and {int}', function (a, b) {
    result = sum(a, b);
});

Then('the result should be {int}', function (expectedResult) {
    assert.strictEqual(result, expectedResult);
});
```

- Lancer le test de comportement : **`npm run behavior`** ⇒

```
> but3-tdd@1.0.0 behavior
> npx cucumber-js

...

1 scenario (1 passed)
3 steps (3 passed)
```

2.2 C'est à vous ! Écrivez les tests de comportement pour le programme Fizzbuzz et CONWAY.

Exercice 3 *Test End To End*

- Créez un nouveau dossier pour votre projet et naviguez-y **`first-steps-playwright`**
- Exécutez la commande d'installation : **`npm init -y`** puis **`npm init playwright@latest`**
- Répondez aux différentes questions lors de l'installation

```
✓ Do you want to use TypeScript or JavaScript? · JavaScript
✓ Where to put your end-to-end tests? · e2e
✓ Add a GitHub Actions workflow? (y/N) · false
✓ Install Playwright browsers (can be done manually via 'npx playwright install')? (Y/n) · true
✓ Install Playwright operating system dependencies (requires sudo / root - can be done manually via 'sudo npx playwright install-deps')? (y/N) · false
```

- Supprimer le fichier **`example.spec.js`** dans le répertoire **`e2e`**
- Création d'un fichier **`test.e2e.spec.js`** dans le répertoire **`e2e`** avec le contenu suivant :

```
const { test, expect } = require('@playwright/test');

test('Test de base sur un site existant', async ({ page }) => {
    await page.goto('https://example.com');
    await expect(page).toHaveTitle('Example Domain');
});
```

- Revenez au terminal et lancez le test avec la commande : **`npx playwright test`**.
- *(Optionnel) Ajout d'un raccourcis dans le package.json dans scripts pour lancer les tests plus rapidement*

Ici, nous vérifions que le site https://example.com possède bien un titre intitulé «Example Domain». D'autres tests à titre d'exemple se trouve dans le fichier **`tests-examples/demo-todo-app.spec.js`**