Міністерство освіти і науки україни

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Математичний факультет

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Дисципліна «Автоматизоване тестування»

Лабораторна робота № 2

Виконав студент гр. 8.1214

Ніценко Вадим

Перевірив: Викладач

Лебідь Максим Володимирович

Запоріжжя

2024

**Завдання**

1. Реалізувати підхід TDD при розробці бібліотеки операцій над матрицями.
2. Для модульного тестування використовувати засоби Mocha, Chai.
3. Створити github репозиторій з відображенням процесу TDD.

**Хід роботи**

Пишимо перший тест на те що усі елементи матриці будуть нулями

const Matrix = require('../main.js');

const { assert } = require('chai');

describe('clearMatrix', () => {

it('повинен встановлювати всі елементи в нулі', () => {

const matrix = new Matrix([[5, 5], [5, 5]]);

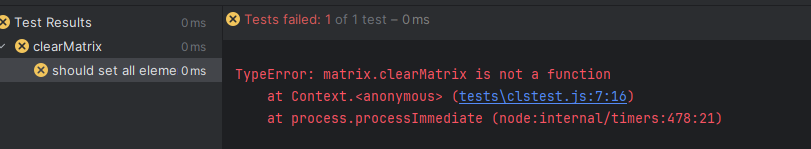
matrix.clearMatrix();

const expected = new Matrix([[0, 0], [0, 0]]);

assert.deepEqual(matrix.data, expected.data);

});

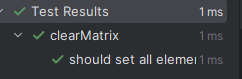
});



Пишимо реалізацію методу

clearMatrix() {

this.data = this.data.map(row => row.map(() => 0));

 return this;

}

Тест 2

Метод повертає true для матриць з однаковими елементами, та false для інших випадків

const Matrix = require('../main.js');

const { assert } = require('chai');

describe('isUniform', () => {

it('should return true for uniform matrices', () => {

const matrix = new Matrix([[3, 3], [3, 3]]);

assert.isTrue(Matrix.isUniform(matrix));

});

it('should return false for non-uniform matrices', () => {

const matrix = new Matrix([[3, 2], [3, 3]]);

assert.isFalse(Matrix.isUniform(matrix));

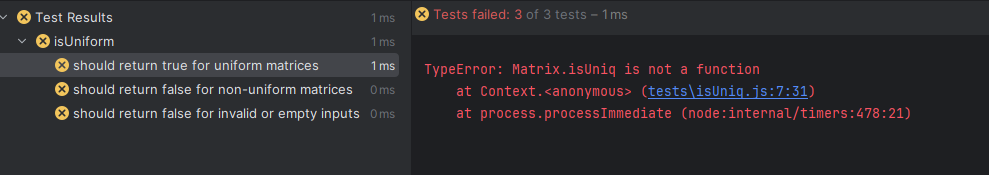
});

it('should return false for invalid or empty inputs', () => {

assert.isFalse(Matrix.isUniform(new Matrix([[]])));

assert.isFalse(Matrix.isUniform(null));

});

});

Реалізуємо метод

static isUniq(matrix) {

if (!matrix || !Array.isArray(matrix.data) || matrix.rows === 0 || matrix.cols === 0) {

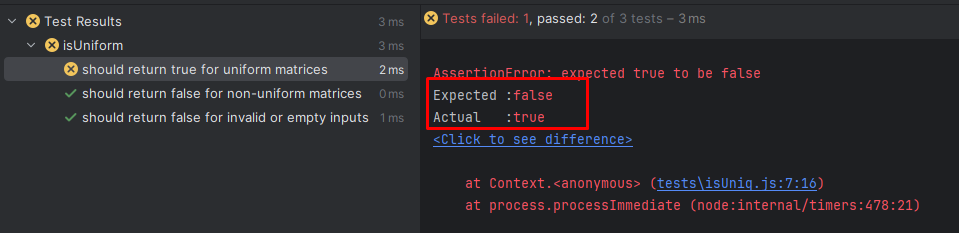
return false;

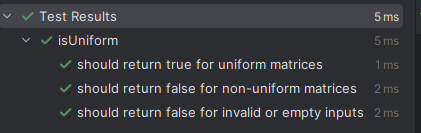
}

const value = matrix.data[0][0];

return matrix.data.every(row => row.every(cell => cell === value));

}



Виправляємо

Тест 3

Метод повинен змінювати рядки та стовпці матриці місцями Наприклад: для матриці [[1, 2], [3, 4]] очікуваний результат після транспонування — [[1, 3], [2, 4]].

const Matrix = require('../main.js');

const { assert } = require('chai');

describe('transpose', () => {

it('повинен транспонувати квадратну матрицю', () => {

const matrix = new Matrix([[1, 2], [3, 4]]);

const transposed = matrix.transpose();

assert.deepEqual(transposed.data, [[1, 3], [2, 4]]);

});

it('повинен транспонувати неквадратну матрицю', () => {

const matrix = new Matrix([[1, 2, 3], [4, 5, 6]]);

const transposed = matrix.transpose();

assert.deepEqual(transposed.data, [[1, 4], [2, 5], [3, 6]]);

});

it('повинен повертати пусту матрицю для порожнього вводу', () => {

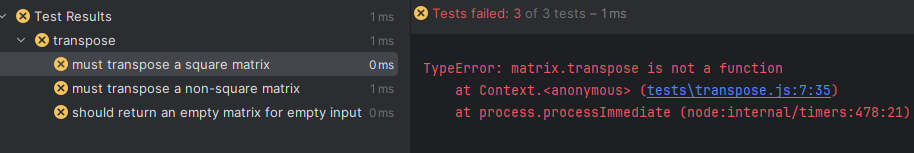
const matrix = new Matrix([]);

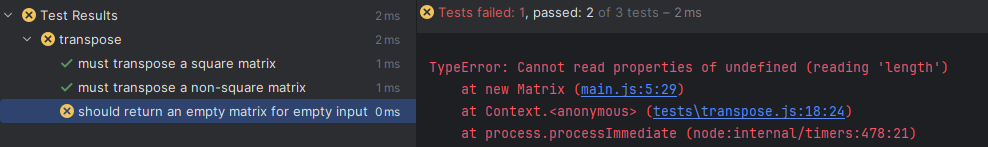
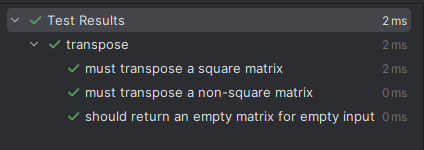
const transposed = matrix.transpose();

assert.deepEqual(transposed.data, []);

});

});





Контрольні запитання

1. Наведіть особливості TDD підходу.

* Цикл "Червоний-Зелений-Рефакторинг": Спочатку пишеться тест, який не проходить (червоний статус), після чого створюється мінімальний код, щоб тест пройшов (зелений статус), і, нарешті, код рефакториться для поліпшення якості та відповідності архітектурним вимогам.
* Фокус на якості коду: TDD дозволяє уникати дублювання коду і створює зручну, протестовану базу, яку простіше розширювати.
* Безперервне тестування: Кожна зміна коду супроводжується тестуванням, що дозволяє швидко знаходити і виправляти помилки

1. Як модульне тестування використовується у методології TDD?

У TDD модульне тестування є основою процесу розробки. Кожен окремий компонент, функція чи метод тестується ізольовано, забезпечуючи незалежність їх роботи. Цей підхід дозволяє гарантувати, що кожна частина програми виконує свої завдання коректно і, відповідно, знижує ймовірність виникнення помилок у великій системі

1. Як враховується специфікація програмного забезпечення та на якому етапі?

Специфікація програмного забезпечення враховується на початковому етапі, перед написанням тестів. Розробник ретельно вивчає вимоги, щоб зрозуміти, як повинна працювати програма, і формулює тестові випадки на основі цих вимог