**Опишіть, у чому полягає методика модульного тестування?**

Методика модульного тестування полягає у перевірці, що кожен блок програмного коду працює належним чином. Модульне тестування виконується розробниками під час розробки проекту. Модульні тести ізолюють блок коду та перевіряють правильність його виконання. Блоком коду може бути функція, метод, процедура, модуль чи об'єкт.

**Опишіть, у чому полягає методика тестування програмного продукту відповідно до ISO/IEC/IEEE 29119 або IEEE\_829**

Вона полягає у дотриманні вимог, якими повинні були керуватися розробники піід час написання функціоналу згідно затвердженої SRS, дослідженні відповіді системи для усіх можливих комбінацій вхідних даних, дослідженні часу виконання тощо.

**Опишіть основні види ассертів для використання у модульних тестах.**

*1. Перевірка вхідних аргументів на початку виконання функції:*

*Якщо буде знайдене недопустиме значення будь-якого аргументу, то поблизу з*

*викликом функції можна віднайти неправильну роботу програми.*

*2. Перевірка даних, якими оперує функція, всередині коду функції*

*Якщо недопустимі дані будуть всередині функції, то баги будуть у безпосередній*

*близості із місцем перевірки.*

**Додайте файл з кодом програми для вашої інформаційної системи відповідно до методики TDD. Опишіть 3 ітерації (тест-код, тест-код, тест-код) розробки коду відповідно до TDD.**

**html**

<div id="mocha"></div>

<form class="test-form">

<input name="first-name" type="text" data-validation="alphabetical" />

<input name="age" type="text" data-validation="numeric" />

</form>

**script.js**

(function () {

'use strict';

const validationRules = new Map([

['alphabetical', /^[a-z]+$/i],

['numeric', /^[0-9]+$/]

]);

// Implementation

function createValidationQueries(inputs) {

return Array.from(inputs).map(input => ({

name: input.name,

type: input.dataset.validation,

value: input.value

}));

}

function validateItem(validation, validationRules) {

if (!validationRules.has(validation.type)) {

return false;

}

return validationRules.get(validation.type).test(validation.value);

}

function validateForm(form) {

const result = {

get isValid() {

return this.errors.length === 0;

},

errors: []

};

for (let validation of createValidationQueries(form.querySelectorAll('input'))) {

let isValid = validateItem(validation, validationRules);

if (!isValid) {

result.errors.push(

new Error(`${validation.value} is not a valid ${validation.name} value`)

);

}

}

return result;

}

// Test Setup

mocha.setup('bdd');

const { expect } = chai;

describe('the form validator', function () {

let form;

beforeEach(function () {

form = document.querySelector('.test-form').cloneNode(true);

});

describe('the validateForm function', function () {

it('should validate a form with all of the possible validation types', function () {

const name = form.querySelector('input[name="first-name"]');

const age = form.querySelector('input[name="age"]');

name.value = 'Bob';

age.value = '42';

const result = validateForm(form);

expect(result.isValid).to.be.true;

expect(result.errors.length).to.equal(0);

});

it('should return an error when a name is invalid', function () {

const name = form.querySelector('input[name="first-name"]');

const age = form.querySelector('input[name="age"]');

name.value = '!!!';

age.value = '42';

const result = validateForm(form);

expect(result.isValid).to.be.false;

expect(result.errors[0]).to.be.instanceof(Error);

expect(result.errors[0].message).to.equal('!!! is not a valid first-name value');

});

it('should return an error when an age is invalid', function () {

const name = form.querySelector('input[name="first-name"]');

const age = form.querySelector('input[name="age"]');

name.value = 'Greg';

age.value = 'a';

const result = validateForm(form);

expect(result.isValid).to.be.false;

expect(result.errors[0]).to.be.instanceof(Error);

expect(result.errors[0].message).to.equal('a is not a valid age value');

});

it('should return multiple errors if more than one field is invalid', function () {

const name = form.querySelector('input[name="first-name"]');

const age = form.querySelector('input[name="age"]');

name.value = '!!!';

age.value = 'a';

const result = validateForm(form);

expect(result.isValid).to.be.false;

expect(result.errors[0]).to.be.instanceof(Error);

expect(result.errors[0].message).to.equal('!!! is not a valid first-name value');

expect(result.errors[1]).to.be.instanceof(Error);

expect(result.errors[1].message).to.equal('a is not a valid age value');

});

});

describe('the createValidationQueries function', function () {

it(

'should map input elements with a data-validation attribute to an array of validation objects',

function () {

const name = form.querySelector('input[name="first-name"]');

const age = form.querySelector('input[name="age"]');

name.value = 'Bob';

age.value = '42';

const validations = createValidationQueries([name, age]);

expect(validations.length).to.equal(2);

expect(validations[0].name).to.equal('first-name');

expect(validations[0].type).to.equal('alphabetical');

expect(validations[0].value).to.equal('Bob');

expect(validations[1].name).to.equal('age');

expect(validations[1].type).to.equal('numeric');

expect(validations[1].value).to.equal('42');

}

);

});

describe('the validateItem function', function () {

const validationRules = new Map([

['alphabetical', /^[a-z]+$/i]

]);

it(

'should return true when the passed item is deemed valid against the supplied validation rules',

function () {

const validation = {

type: 'alphabetical',

value: 'Bob'

};

const isValid = validateItem(validation, validationRules);

expect(isValid).to.be.true;

}

);

it(

'should return false when the passed item is deemed invalid',

function () {

const validation = {

type: 'alphabetical',

value: '42'

};

const isValid = validateItem(validation, validationRules);

expect(isValid).to.be.false;

}

);

it(

'should return false when the specified validation type is not found',

function () {

const validation = {

type: 'foo',

value: '42'

};

const isValid = validateItem(validation, validationRules);

expect(isValid).to.be.false;

}

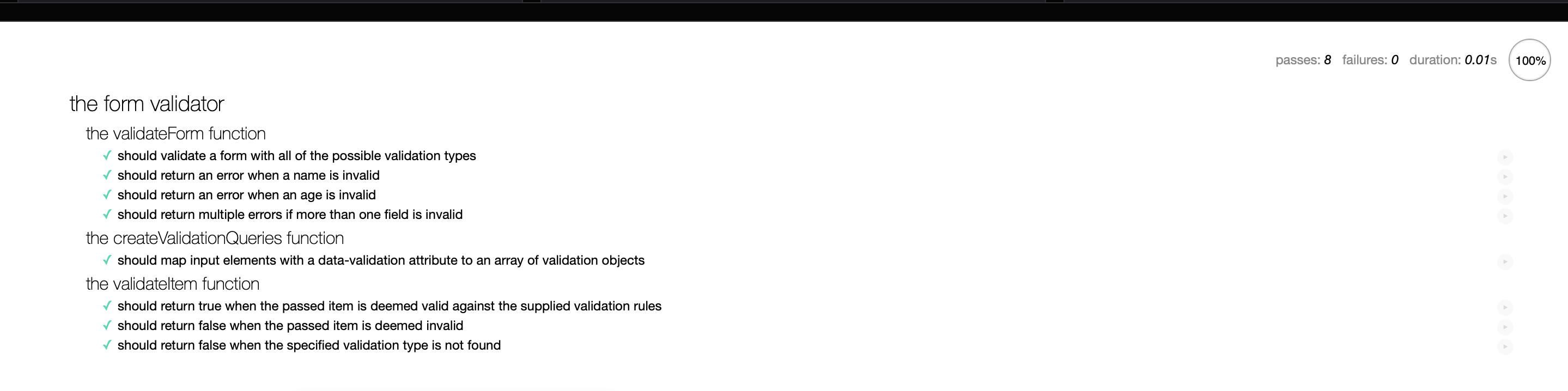
);

});

});

mocha.run();

}());



**Які основні види рефакторингу існують?**

1. Зміна сигнатури методу (change method signature)

2. Інкапсуляція поля (encapsulate field)

3. Виділення методу (extract method)

4. Переміщення методу (move method)

5. Заміна умовного оператору поліморфізмом (replace conditional with polymorphism)

**Додайте файл з описом коду до та після рефакторингу. Застосуйте 5-6 видів рефакторингу.**

**Чому методологія TDD є більш ефективною та економить час при тестуванні?**

Тому що debugging займає значно менше часу і надає повний звіт щодо помилки. І що головне, якщо зміниться розробник, то він також буде розуміти, що не так завдяки цим тестам. TDD надає можливість постійного і швидкого отримання відомостей щодо стану коду в реальному часі.

**Опишіть цикл розробки програмного забезпечення за допомогою методології TDD?**

Спочатку пишеться тест, що покриває бажану зміну, потім пишеться програмний код, який реалізує бажану поведінку системи і дозволяє пройти написаний тест.

1. Опис текстового сценарію

2. запуск і впевненість у роботонеспроможності

3. опис функціоналу, для якого були описані тести

4. перевірка роботоспроможності тестів після імплементації

5. удосконалення коду

6. повторення

**Яким чином забезпечується покриття коду та покриття тестами в методології TDD?**

Першочергово пишеться тест на функціонал або зміни функціоналу, які будуть реалізовуватись. Таким чином TDD дозволяє на етапі розробки одразу показати коректність або некоректність роботи розроблюваного функціоналу.

**Чи є розробка нового функціоналу програми рефакторингом?**

Ні, так як рефакторинг представляє зміну коду без зміни функціоналу.

**Чим відрізняється рефакторинг від реінженирингу?**

Рефакторинг - зміна коду без зміни функціоналу

Реінжениринг - повна зміна у проекті через появу нових підходів, або зміна архітектури

**Висновки**

У ході виконання лабораторної роботи я ознайомилася із TDD підходом та рефакторингом.