INSTITUTO TECNOLÓGICO DE COSTA RICA
CURSO: ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE
PROFESOR: ANDRÉS VÍQUEZ VÍQUEZ
LABORATORIO: TS-MOCKITO

En este taller vamos a desarrollar un ejemplo con la biblioteca **ts-mockito** para comprender mejor como se implementan los dobles de prueba.

1. Cree un nuevo proyecto llamado **`lab-qa2`**:

```
ng new lab-qa2
```

2. Instale el siguiente componente:

```
npm install ts-mockito --save-dev
```

3. Cree la estructura de las clases **`Client`**, **`Count`** y **`Sucursal`** y copie los contenidos de cada una que se encuentran en la carpeta del laboratorio:

```
ng g class Client --type=model
ng g interface Count
ng g class Sucursal --type=model
```

   Luego proceda a documentar en las clases de prueba **`client.model.spec.ts`** y **`sucursal.model.spec.ts`** las líneas de las pruebas que crean objetos de sus clases respectivas, para así descartar los errores asociados con la creación de objetos y su definición de los constructores de sus clases.

   Al revisar las clases se dará cuenta que la clase **`Count`** es una interfaz, por lo que para realizar las pruebas de integración debe utilizar un doble de prueba.

4. En la clase de prueba **`Sucursal`** importe la biblioteca **`ts-mockito`**:

```
import { mock, when, instance } from 'ts-mockito';
```

5. Realice la importación de las clases **`Client, Sucursal`** y **`Count`** en la clase de prueba **`Sucursal`** según se muestra a continuación:

```
import { Client } from './client.model';
import { Count } from './count';
import { Sucursal } from './sucursal.model';
```

6. Declare las siguientes variables globales en la clase de prueba **`Sucursal`**:

```
describe('Sucursal', () => {
  let cliente: Client;
  let sucursal: Sucursal;
  let cuenta: Count;
  var withdrawlAmount2000 = 200000;
  var numeroCuenta = 12345;
  var balance = 100000;
});
```

7. Agregue los siguientes métodos antes de la llave de cierre del código anterior:

```
beforeEach(() => {
    sucursal = new Sucursal("Alajuela", "Alajuela");
    cliente = new Client("Juan", "Pérez", "25-01-76", "2401-3117", "Alajuela",
"jperez@gmail.com");
    sucursal.setClientes(cliente);
    cuenta = mock<Count>();
});
```

8. En primer lugar, vamos a verificar el saldo de una cuenta:

```
it('1. Saldo de cuenta', function () {
    when(cuenta.getCantidadDinero()).thenReturn(balance);
    let mockito = instance(cuenta);
    expect(mockito.getCantidadDinero()).toBe(balance);
});
```

A través de la función **`when`** establecemos el comportamiento deseado a llamar al método **`getCantidadDinero`**, para posteriormente crear una instancia y comprobar el comportamiento esperado.

9. Seguidamente, vamos a probar si al agregarle una cuenta a un cliente, realmente el cliente posee la cuenta. El proceso es similar al anterior, pero ahora necesitamos una instancia de cuenta:

```
it('2. Agregar nueva cuenta a cliente', function () {
    var cuenta = mock<Count>();
    let mockito = instance(cuenta);
    cliente.setCuentas(mockito);
    expect(cliente.getCuentas().length).toBe(1);
});
```

10. En este caso de prueba, se validará el proceso de realizar un retira válido. Debido a que el método **`retirar`** que se encuentra en la clase **`Client`** necesita métodos de la clase **`Count`** que aún no han sido implementados como **`getCantidadDinero`** y **`retirar`**, se necesitará utilizar un doble de prueba:

```
it('3. Retirar monto válido', function () {
    var balanceAmount3000 = 300000;
    when(cuenta.getCantidadDinero()).thenReturn(balanceAmount3000);
    when(cuenta.getNumCuenta()).thenReturn(numeroCuenta);
    when(cuenta.retirar(withdrawlAmount2000)).thenReturn(balance);
    let mockito = instance(cuenta);
    cliente.setCuentas(mockito);
    var saldo = cliente.retirar(withdrawlAmount2000, numeroCuenta);
    expect(saldo).toBe(balance);
});
```

11. Ahora, se probará retirar un monto de una cuenta, pero ésta no tiene fondos suficientes por lo que debe generar una excepción.

```
it('4. Retirar más de lo permitido', function () {
    when(cuenta.getCantidadDinero()).thenReturn(balance);
    when(cuenta.getNumCuenta()).thenReturn(numeroCuenta);
    let mockito = instance(cuenta);
    cliente.setCuentas(mockito);
    expect(function() {
      cliente.retirar(withdrawlAmount2000, numeroCuenta);
    }).toThrowError(Error, "Fondos insuficientes");
});
```

12. Con un panorama más claro de los dobles de prueba, proceda a desarrollar lo siguiente:

    a. Elabore un caso de prueba para verificar que cuando un cliente apertura una cuenta en una sucursal, el monto de la cuenta sea de 5000 colones.
    b. Suponiendo que la cuenta está en cero, implemente un caso de prueba en donde se realizan dos depósitos válidos y verifique que el saldo de la cuenta coincide con la suma de los dos depósitos.
    c. Verifique mediante un caso de prueba que al liquidar una cuenta de un cliente de una sucursal, la cantidad de cuentas del cliente disminuye en 1.

Nota: puede agregar nuevos métodos en la clase **`Client`** para desarrollar los puntos anteriores.

Referencia de apoyo: https://medium.com/passionate-people/testing-your-typescript-code-with-ts-mockito-ac439deae33e