Programación con Java

TAREA **21**

Alumno: Alejandro Rodríguez Jiménez
Profesor: Marc Esteve García
Curso 2024

Índice

Durante el proceso de esta práctica aprenderemos a usar JUnit para realizar pruebas unitarias, pruebas de pequeñas partes del código para contemplar mejor los errores y tratarlos por individual.

Anotaciones comunes:

`@Test`: Marca un método como una prueba.

`@BeforeEach`: Método ejecutado antes de cada prueba.

`@AfterEach`: Método ejecutado después de cada prueba.

`@BeforeAll`: Método ejecutado una vez antes de todas las pruebas.

`@AfterAll`: Método ejecutado una vez después de todas las pruebas.

`@DisplayName`: Establece un nombre de visualización para una prueba.

Estructura de las pruebas:

Antes de crear la prueba deberemos decir que es una prueba, para ello pondremos `@Test` o `@ParameterizedTest`, esta última servirá para pasarle varios valores con los que hará la prueba y así tener una mejor garantía; después de una de estas etiquetas podremos crear el test con "*public void nombreTest() {}*".

Para las verificaciones que harán los test tendremos que configurar los assets, la "norma" que debe seguir:

assertEquals(esperado, actual): Verifica que dos valores sean iguales.

assertNotEquals(noEsperado, actual): Verifica que dos valores no sean iguales.

assertTrue(condición): Verifica que una condición sea verdadera.

assertFalse(condición): Verifica que una condición sea falsa.

assertNull(objeto): Verifica que un objeto sea null.

assertNotNull(objeto): Verifica que un objeto no sea null.

assertSame(esperado, actual): Verifica que dos referencias se refieran al mismo objeto.

assertNotSame(noEsperado, actual): Verifica que dos referencias no se refieran al mismo objeto.

assertArrayEquals(arrayEsperado, arrayActual): Verifica que dos arrays sean iguales.

He tenido algunos problemas a la hora de hacer esta práctica y de entender lo que habría que hacer, por lo que empecé creando calculadoras con pruebas unitarias para sus métodos. Los archivos los voy a dejar y para las capturas sacaré alguna de ahí, pero no explicaré el código ya que no es lo que se pedía.

Ejercicio 01:

Como primer ejercicio, tendremos que generar test pruebas unitarias para los métodos de una calculadora. Por ejemplo: sumar, restar, dividir.

Aquí haremos las verificaciones en las pruebas de la siguiente manera:

> Dentro de la prueba, crearemos un nuevo Modelo y con AssertEqual revisaremos que las dos entradas son iguales. En la parte para el valor esperado pondremos el resultado de la operación y en el valor actual llamaremos al método para hacer la operación correspondiente, añadire un test para cuando la división sea entre 0 para generar una excepción.

Pruebas Unitarias

```java
@Test
void testSumar() {
    Modelo modelo = new Modelo();
    assertEquals(expected:5, modelo.sumar(op1:2, op2:3));
}

@Test
void testRestar() {
    Modelo modelo = new Modelo();
    assertEquals(expected:1, modelo.restar(op1:3, op2:2));
}

@Test
void testMultiplicar() {
    Modelo modelo = new Modelo();
    assertEquals(expected:6, modelo.multiplicar(op1:2, op2:3));
}

@Test
void testDividir() {
    Modelo modelo = new Modelo();
    assertEquals(expected:2, modelo.dividir(op1:6, op2:3));
}

@Test
void testDividirPorCero() {
    Modelo modelo = new Modelo();
    assertThrows(expectedType:ArithmeticException.class, () -> modelo.dividir(op1:1, op2:0));
}
```

Métodos de la aplicación

```java
public double sumar(double op1, double op2) {
    return op1 + op2;
}

public double restar(double op1, double op2) {
    return op1 - op2;
}

public double multiplicar(double op1, double op2) {
    return op1 * op2;
}

public double dividir(double op1, double op2) {
    if (op2 == 0) {
        throw new ArithmeticException(s:"División por cero");
    }
    return op1 / op2;
}
```

Ejercicio 02:

Para el segundo ejercicio, recogeremos una clase de Geometría que ya está creada y le añadiremos un archivo Test con tres pruebas unitarias, mínimo. En estas añadiremos un texto que se mostrará en caso de error y, para el método areaCirculo también, le pasaremos un delta (0.0001) para manejar la precisión del resultado.

Pruebas Unitarias

```java
@BeforeEach
public void setUp() {
    geometria = new Geometria();
}

@Test
public void testAreaCuadrado() {
    int lado = 4; int expected = 16;
    int actual = geometria.areacuadrado(lado);
    assertEquals(expected, actual, message:"El área del cuadrado es incorrecta");
}

@Test
public void testAreaCirculo() {
    int radio = 3; double expected = 3.1416 * radio * radio;
    double actual = geometria.areaCirculo(radio);
    assertEquals(expected, actual, message:"El área del círculo es incorrecta");
}

@Test
public void testAreaTriangulo() {
    int base = 4; int altura = 5; int expected = (base * altura) / 2;
    int actual = geometria.areatriangulo(base, altura);
    assertEquals(expected, actual, message:"El área del triángulo es incorrecta");
}

@Test
public void testAreaRectangulo() {
    int base = 4; int altura = 6; int expected = base * altura;
    int actual = geometria.arearectangulo(base, altura);
    assertEquals(expected, actual, message:"El área del rectángulo es incorrecta");
}

@Test
public void testAreaPentagono() {
    int perimetro = 5; int apotema = 3; int expected = (perimetro * apotema) / 2;
    int actual = geometria.areapentagono(perimetro, apotema);
    assertEquals(expected, actual, message:"El área del pentágono es incorrecta");
}

@Test
public void testAreaRombo() {
    int diagonalMayor = 8; int diagonalMenor = 6;
    int expected = (diagonalMayor * diagonalMenor) / 2;
    int actual = geometria.arearombo(diagonalMayor, diagonalMenor);
    assertEquals(expected, actual, message:"El área del rombo es incorrecta");
}
```

Métodos de la aplicación

```java
public int areacuadrado(int n1) {
    return n1 * n1;
}

public double areaCirculo(int r) {
    final double PI = 3.1416;
    return PI * Math.pow(r, b:2);
}

public int areatriangulo(int a, int b) {
    return (a * b) / 2;
}

public int arearectangulo(int b, int h) {
    return b * h;
}

public int areapentagono(int p, int a) {
    return (p * a) / 2;
}

public int arearombo(int D, int d) {
    return (D * d) / 2;
}

public int arearomboide(int b, int h) {
    return b * h;
}

public int areatrapecio(int B, int b, int h) {
    return ((B + b) / 2) * h;
}
```

Webgrafía

- Para ver API's de java:
  Java API
- El copiloto de confianza:
  ChatGPT
- Tabla de colores:
  Color Picker
- Clase Geometria de José Marin:
  Class Geometria.java