Pruebas Unitarias con JUnit en NetBeans

Para instalar JUnit y Hamcrest hacemos clic derecho sobre libraries del proyecto y luego elegimos el item agregar libreria(Add library).

Abrimos NetBeans y creamos un nuevo proyecto Java.

En el proyecto, siempre en la carpeta de **Test Packages**, creamos una nueva clase llamada "Calculadora" que contenga los métodos que deseamos probar. Por ejemplo, podriamos tener un método para sumar dos números:

public class Calculadora {

public int sumar(int a, int b) {

return a + b;

}

}

Ahora, creamos una nueva clase de prueba llamada "CalculadoraPrueba" para probar la clase "Calculadora". Podemos hacer clic derecho en el proyecto -> Nuevo -> Otros -> Prueba de unidad -> JUnit 4 -> Siguiente. Luego, elige el paquete y el nombre de la clase de prueba.

En la clase de prueba, importamos las clases necesarias:

import org.junit.Test;

import static org.junit.Assert.assertEquals;

Agregamos un método de prueba anotado con @Test que verifique el resultado del método sumar() de la clase "CalculadoraPrueba". Por ejemplo:

public class CaculadoraPrueba {

@Test

public void pruebaSumar() {

Calculadora c = new Calculadora();

int resultado = c.sumar(2, 3);

assertEquals(5, resultado);

}

}

Ejecutamos la prueba haciendo clic derecho en la clase de prueba y seleccionando "Ejecutar archivo".

Si la prueba pasa, veremos un mensaje indicando que se completó con éxito. Si la prueba falla, se mostrará un mensaje de error junto con la ubicación del problema.

Biblioteca Hamcrest

La biblioteca de Hamcrest es una biblioteca de aserciones y coincidencias utilizada en pruebas unitarias y en el desarrollo de software en general. Proporciona una sintaxis más expresiva y legible para realizar verificaciones y afirmaciones en las pruebas.

Hay varios métodos comunes y útiles que se utilizan con frecuencia al trabajar con la biblioteca Hamcrest. A continuación, se presentan algunos de los métodos más utilizados:

1. assertThat: Este método es la piedra angular de Hamcrest y se utiliza para realizar aserciones en general. Toma como argumentos el valor real y uno o más matchers para verificar el valor.

Ejemplo:

assertThat(valor, matcher);

1. is: El método is se utiliza para realizar una coincidencia exacta. Verifica si el valor es igual al valor esperado.

Ejemplo:

assertThat(valor, is(valorEsperado));

1. equalTo: El método equalTo se utiliza para verificar la igualdad entre dos valores.

Ejemplo:

assertThat(valor, equalTo(valorEsperado));

1. not: El método not se utiliza para negar una coincidencia. Verifica si el valor no cumple con el matcher dado.

Ejemplo:

assertThat(valor, not(valorEsperado));

1. greaterThan y lessThan: Estos métodos se utilizan para verificar si un valor es mayor o menor que otro valor dado.

Ejemplo:

assertThat(valor, greaterThan(valorEsperado));

assertThat(valor, lessThan(valorEsperado));

1. hasItem: El método hasItem se utiliza para verificar si una colección contiene un elemento específico.

Ejemplo:

assertThat(coleccion, hasItem(elemento));

1. hasProperty: El método hasProperty se utiliza para verificar si un objeto tiene una propiedad específica con un valor determinado.

Ejemplo:

assertThat(objeto, hasProperty("nombre", equalTo(nombreEsperado)));