**JUnit**

**Definición  
 Configuración local  
JUnit: Test framework  
 1. Fixtures**

**2. Test Suites  
 3. Test Runners  
 4. JUnit Classes**

JUnit es un marco de pruebas unitarias para el lenguaje de programación Java. JUnit ha sido importante en el desarrollo del desarrollo dirigido por pruebas, y es uno de una familia de marcos de pruebas unitarias conocidos colectivamente como xUnit, que se originó con JUnit.

JUnit promueve la idea de "primero probar y luego codificar", que hace hincapié en la configuración de los datos de prueba para un trozo de código que puede ser probado primero y luego implementado. Este enfoque es como "probar un poco, codificar un poco, probar un poco, codificar un poco". Aumenta la productividad del programador y la estabilidad del código del programa, lo que a su vez reduce el estrés del programador y el tiempo dedicado a la depuración.

Un caso de prueba de unidad es una parte del código que garantiza que otra parte del código (método) funciona como se espera. Para lograr los resultados deseados rápidamente, se requiere un marco de pruebas. JUnit es un marco de pruebas unitarias perfecto para el lenguaje de programación Java.

Un **Unit Case Test** (caso de prueba unitaria) escrito formalmente se caracteriza por una entrada conocida y una salida esperada, que se elabora antes de ejecutar la prueba. La entrada conocida debe probar una precondición y la salida esperada debe probar una postcondición.

Debe haber al menos dos casos de pruebas unitarias para cada requisito: una prueba positiva y otra negativa. Si un requisito tiene sub-requisitos, cada sub-requisito debe tener al menos dos casos de prueba como positivo y negativo.

**Configuración del entorno local**

JUnit es un framework para Java, por lo que el primer requisito es tener el JDK instalado en su máquina, más información disponible en: [JUnit - Environment Setup - Tutorialspoint](https://www.tutorialspoint.com/junit/junit_environment_setup.htm)

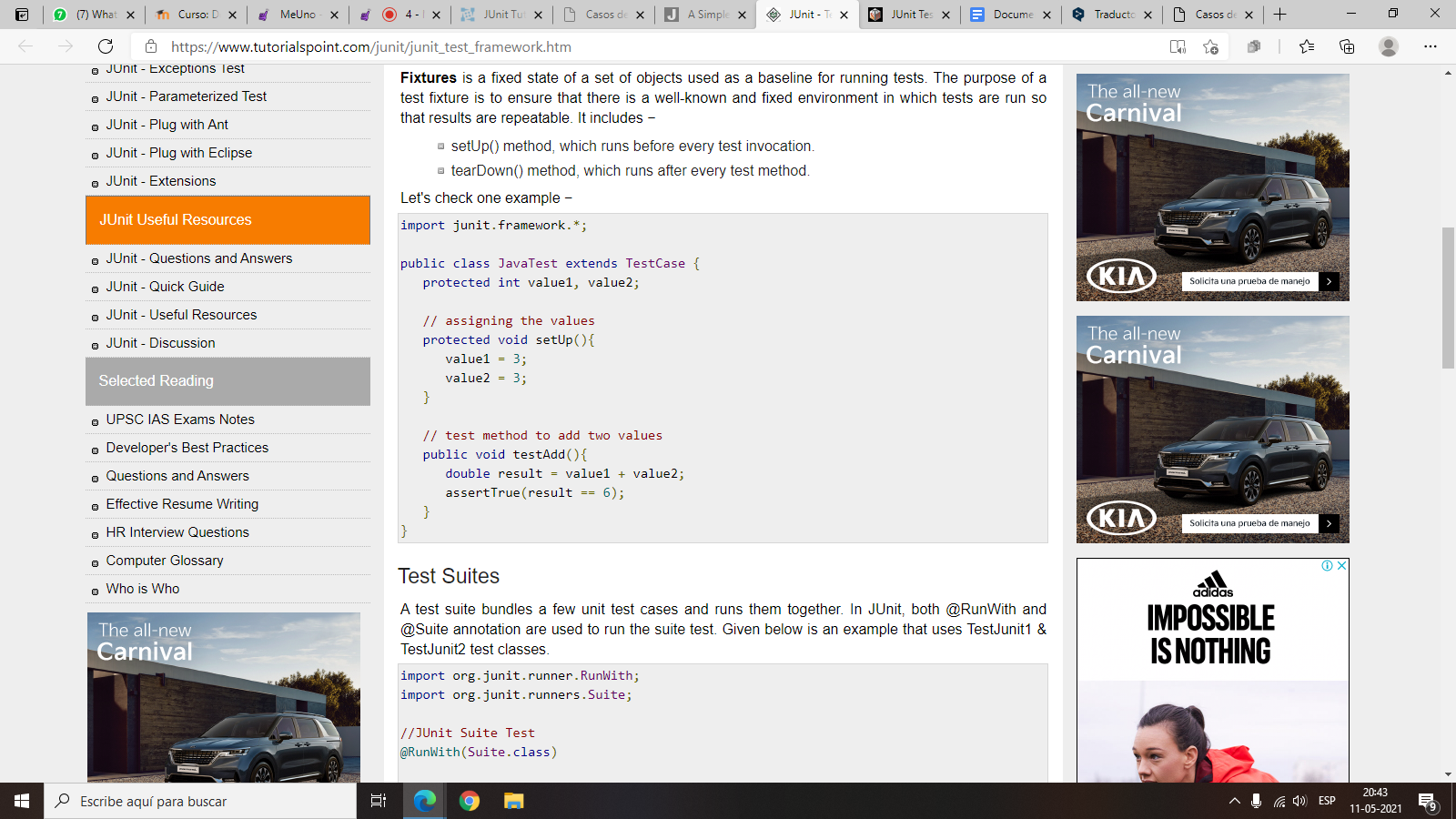
**JUnit: Test Framework**

Unit es un *“Regression Testing Framework”* (marco de pruebas de regresión, en español) utilizado por los desarrolladores para implementar pruebas unitarias en Java, y acelerar la velocidad de programación y aumentar la calidad del código. JUnit Framework puede integrarse fácilmente con cualquiera de los siguientes elementos: Eclipse, Ant y Maven.

El marco de pruebas de JUnit proporciona las siguientes características importantes:

1. **Fixtures:** es un estado fijo de un conjunto de objetos que se utiliza como línea de base para ejecutar las pruebas. El propósito de un fixture de pruebas es asegurar que haya un entorno conocido y fijo en el que se ejecuten las pruebas para que los resultados sean repetibles. Incluyen:
   1. setUp() method: que se ejecuta antes de cada invocación de la prueba.
   2. tearDown() method: que se ejecuta después de cada método de prueba.

Ejemplo:



1. **Test Suites:** Un conjunto de pruebas agrupa algunos casos de pruebas unitarias y los ejecuta juntos. En JUnit, las anotaciones @RunWith y @Suite se utilizan para ejecutar el conjunto de pruebas. A continuación se muestra un ejemplo que utiliza las clases de prueba TestJunit1 y TestJunit2:

Ejemplos:

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

//JUnit Suite Test

@RunWith(Suite.class)

@Suite.SuiteClasses({

TestJunit1.class ,TestJunit2.class

})

public class JunitTestSuite {

}

import org.junit.Test;

import org.junit.Ignore;

import static org.junit.Assert.assertEquals;

public class TestJunit1 {

String message = "Robert";

MessageUtil messageUtil = new MessageUtil(message);

@Test

public void testPrintMessage() {

System.out.println("Inside testPrintMessage()");

assertEquals(message, messageUtil.printMessage());

}

}

--------------------------------------------------------------------------------------------------------------------------

import org.junit.Test;

import org.junit.Ignore;

import static org.junit.Assert.assertEquals;

public class TestJunit2 {

String message = "Robert";

MessageUtil messageUtil = new MessageUtil(message);

@Test

public void testSalutationMessage() {

System.out.println("Inside testSalutationMessage()");

message = "Hi!" + "Robert";

assertEquals(message,messageUtil.salutationMessage());

}

}

1. **Test Runners:** se utiliza para ejecutar los casos de prueba. Aquí hay un ejemplo que asume que la clase de prueba TestJunit ya existe.

Ejemplo:

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(TestJunit.class);

for (Failure failure : result.getFailures()) {

System.out.println(failure.toString());

}

System.out.println(result.wasSuccessful());

}

}

1. **JUnit Classes:** Las clases JUnit son clases importantes, utilizadas para escribir y probar JUnits. Algunas de las clases importantes son
   1. Assert: Contiene un conjunto de métodos de afirmación
   2. TestCase: Contiene un caso de prueba que define el fixture para ejecutar múltiples pruebas.
   3. TestResult: Contiene métodos para recoger los resultados de la ejecución de un caso de prueba.