

REALIZACIÓN DE PRUEBAS **CON JUNIT** **DESARROLLO DE** **INTERFACES - 7**

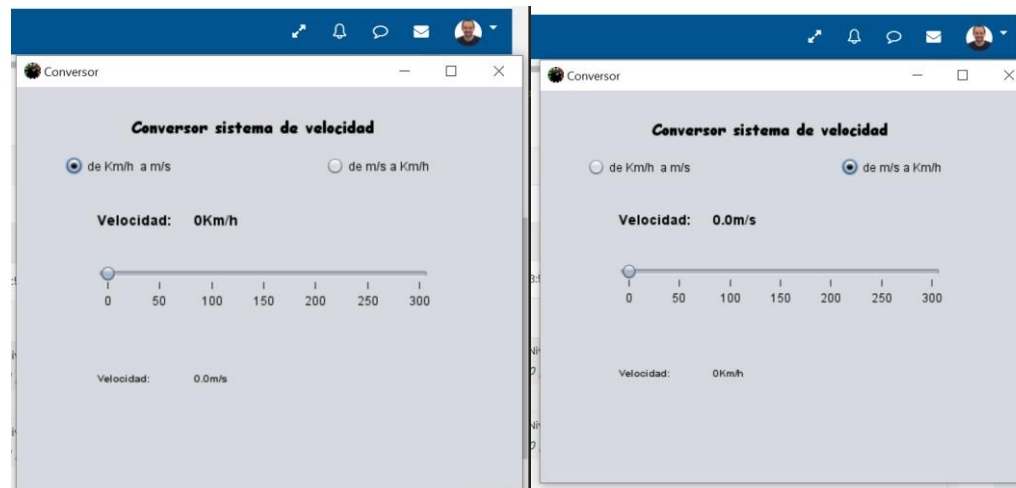
Alexandre Zerva Moreno

Desarrollo de Aplicaciones Multiplataforma

- Crear un proyecto nuevo que incorporará dos funciones para convertir unidades de velocidad, una para convertir Kilómetros por hora (Km/h) a metros por segundo (m/s) y otra para la conversión inversa (m/s a Km/h).

Se crea aplicación con dos JRadioButtons agrupados, y un JSlider el cual iremos moviendo y nos mostrará la velocidad en los JTextViews.

Dicha velocidad se mostrará (en el texto superior) según la opción escogida en los botones y en la parte de abajo se mostrará el valor convertido.



- Una vez implementadas ambas funciones, realizar las pruebas unitarias para ambas funciones.

The screenshot displays a Java IDE environment with three main components:

- Test Results Panel (Top Left):** Shows the results of three unit tests:
 - Test1 exitosa (0.038 s)
 - Test2 exitosa (0.003 s)
 - Test3 exitosa (0.0 s)
- Application Window (Center):** Titled "Convertor sistema de velocidad", it features two radio buttons for "de Km/h a m/s" and "de m/s a Km/h". The "de m/s a Km/h" option is selected. Below the buttons, it displays "Velocidad: 180.0m/s" and a slider ranging from 0 to 300. At the bottom, it shows "Velocidad: 50Km/h".
- Code Editor (Right):** Contains the source code for the `ConvertMsToKm` class. The code includes imports for `converter.Converter`, `org.junit.After`, `org.junit.Assert.assertEquals`, `org.junit.Before`, and `org.junit.Test`. It defines a `ConvertMsToKm` class with a `Converter` instance and three test methods: `Test1`, `Test2`, and `Test3`. Each test method uses `assertEquals` to verify the conversion of 10, 50, and 100 m/s to Km/h.

At the bottom of the IDE, a browser window shows a user profile page for "Alexandre Zerva Moreno" on a platform called "educacionadista...".

```
public double getKm() {  
    return km;  
}
```

```
public void setKm(double km) {  
    this.km = km;  
}
```

```
public double getMs() {  
    return ms;  
}
```

```
public void setMs(double ms) {  
    this.ms = ms;  
}
```

```
public static double convertKmToMs(double kmh) {  
    double ms = kmh * 0.27778;  
    return ms;  
}
```

```
public static double convertMsToKm(double ms) {  
    double kmh = ms * 3.6;  
    return kmh;  
}
```

```
}
```

- Tipo de integración que se ha realizado (ascendente o descendente)

El tipo de integración es ascendente porque las pruebas no se ejecutan a partir de algo construido y no se ha automatizado.

PD: No queda nada claro estoy perdido, no lo siguiente en este aspecto, ya me he leído el temario 25 veces y cuando más leo más me lio.

The screenshot displays an IDE with two main panels. The left panel shows the 'Test Results' window for 'ConvertMsToKm'. It indicates that all 4 tests passed successfully (exitosas) with a total duration of 0.536 seconds. The tests listed are: ConvertMsToKm (0.536 s), Test1 (0.004 s), Test2 (0.001 s), Test3 (0.0 s), and testIntegracion (0.306 s). Below this, the 'Test Output' window shows the execution details for each test, including the conversion of 10 m/s to 36.0 Km/h, 50 m/s to 180.0 Km/h, and 100 m/s to 360.0 Km/h, as well as the execution of the integration test.

The right panel shows the 'Source' window with the Java code for 'ConvertMsToKm.java'. The code includes three unit tests (Test1, Test2, Test3) and an integration test (testIntegracion). Each unit test calls the 'convertMsToKm' method with a specific input value (10, 50, and 100 m/s) and asserts that the result is equal to the expected value (36, 180, and 360 Km/h) within a delta of 0.01. The integration test creates a 'ViewConversor' object and calls its 'pruebaIntegracion' method, asserting that it returns 0.

```
29 }
30
31 @Test
32 public void Test1() {
33     System.out.println("Test1");
34     double actual = con.convertMsToKm(10);
35     double expcted = 36;
36     double delta = 0.01;
37     assertEquals(expcted, actual, delta);
38     System.out.println("Velocidad en m/s: 10 Velocidad en Km/h: " + expcted);
39 }
40
41
42 @Test
43 public void Test2() {
44     System.out.println("Test2");
45     double actual = con.convertMsToKm(50);
46     double expcted = 180;
47     double delta = 0.01;
48     assertEquals(expcted, actual, delta);
49     System.out.println("Velocidad en m/s: 50 Velocidad en Km/h: " + expcted);
50 }
51
52
53 @Test
54 public void Test3() {
55     System.out.println("Test3");
56     double actual = con.convertMsToKm(100);
57     double expcted = 360;
58     double delta = 0.01;
59     assertEquals(expcted, actual, delta);
60     System.out.println("Velocidad en m/s: 100 Velocidad en Km/h: " + expcted);
61 }
62
63
64
65 @Test
66 public void testIntegracion(){
67     ViewConversor vc = new ViewConversor();
68     assertEquals(0,vc.pruebaIntegracion(), 0);
69 }
70
71 }
```

- *Realizando las pruebas de regresión.*

The screenshot displays an IDE with two main panels. The left panel shows the 'Results of tests' (Resultados de las pruebas) for the 'ConvertMsToKm' class. A green progress bar indicates 100.00% success. Below this, a list of test results is shown: 'All 4 tests successful (0.448 s)', 'ConvertMsToKm successful', 'Test1 successful (0.004 s)', 'Test2 successful (0.0 s)', 'Test3 successful (0.0 s)', and 'testIntegracion successful (0.401 s)'. The right panel shows the source code of 'ConversionToKm.java'. The code includes a 'setUp' method, an 'after' method, and three test methods: 'Test1', 'Test2', and 'Test3'. Each test method calls 'convertKmToMs' with a specific velocity value (10, 50, and 100 respectively) and asserts the result against an expected value. The output of the tests is visible in the bottom-left pane, showing the conversion of 10 m/s to 36.0 Km/h, 50 m/s to 180.0 Km/h, and 100 m/s to 360.0 Km/h, along with an integration test result.

```
Conversion de ms a Km/h
Test1
Velocidad en m/s: 10 Velocidad en Km/h: 36.0
Test realizado.

Conversion de ms a Km/h
Test2
Velocidad en m/s: 50 Velocidad en Km/h: 180.0
Test realizado.

Conversion de ms a Km/h
Test3
Velocidad en m/s: 100 Velocidad en Km/h: 360.0
Test realizado.

Conversion de ms a Km/h
Ejecutando prueba integraci3n.
Test realizado.
```

```
21
22 @Before
23 public void setUp() {
24     con = new Converter(0, 0);
25     System.out.println("Conversion de Km/h a ms");
26 }
27
28 @After
29 public void after() {
30     System.out.println("Test realizado correctamente.\n");
31 }
32
33
34 @Test
35 public void Test1() {
36
37     System.out.println("Test1");
38     double actual = con.convertKmToMs(10);
39     double expected = 2.7778;
40     double delta = 0.01; //Al ser un double generamos margen de error
41     assertEquals(expected, actual, delta);
42     System.out.println("Velocidad en Km/h: 10 velocidad convertida a ms = 2.7778");
43 }
44
45 @Test
46 public void Test2() {
47     System.out.println("Test2");
48     double actual = con.convertKmToMs(50);
49     double expected = 13.889;
50     double delta = 0.01;
51     assertEquals(expected, actual, delta);
52     System.out.println("Velocidad en Km/h: 50 velocidad convertida a ms = 13.889");
53 }
54
55 @Test
56 public void Test3() {
57     System.out.println("Test3");
58     double actual = con.convertKmToMs(100);
59     double expected = 27.778;
60     double delta = 0.01;
61     assertEquals(expected, actual, delta);
62     System.out.println("Velocidad en Km/h: 100 velocidad convertida a ms = 27.778");
63 }
64
65 @Test
66 public void testIntegracion() {
67     ViewConversor vc = new ViewConversor();
68     assertEquals(0, vc.pruebaIntegracion(), 0);
69 }
70
71 }
```