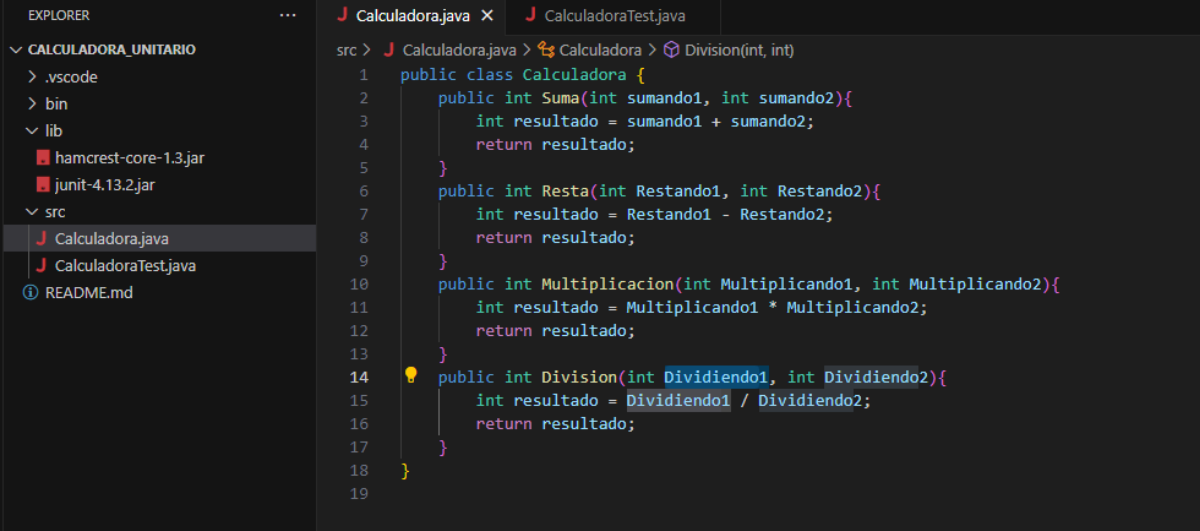


## Ejercicio Calculadora Simple:

Creamos un Proyecto de java en mi caso es "CALCULADORA\_UNITARIO".

Creamos un archivo llamado Calculadora donde hacemos la clase calculadora con sus diferentes métodos que son Sumar, Restar, Multiplicar y Dividir.

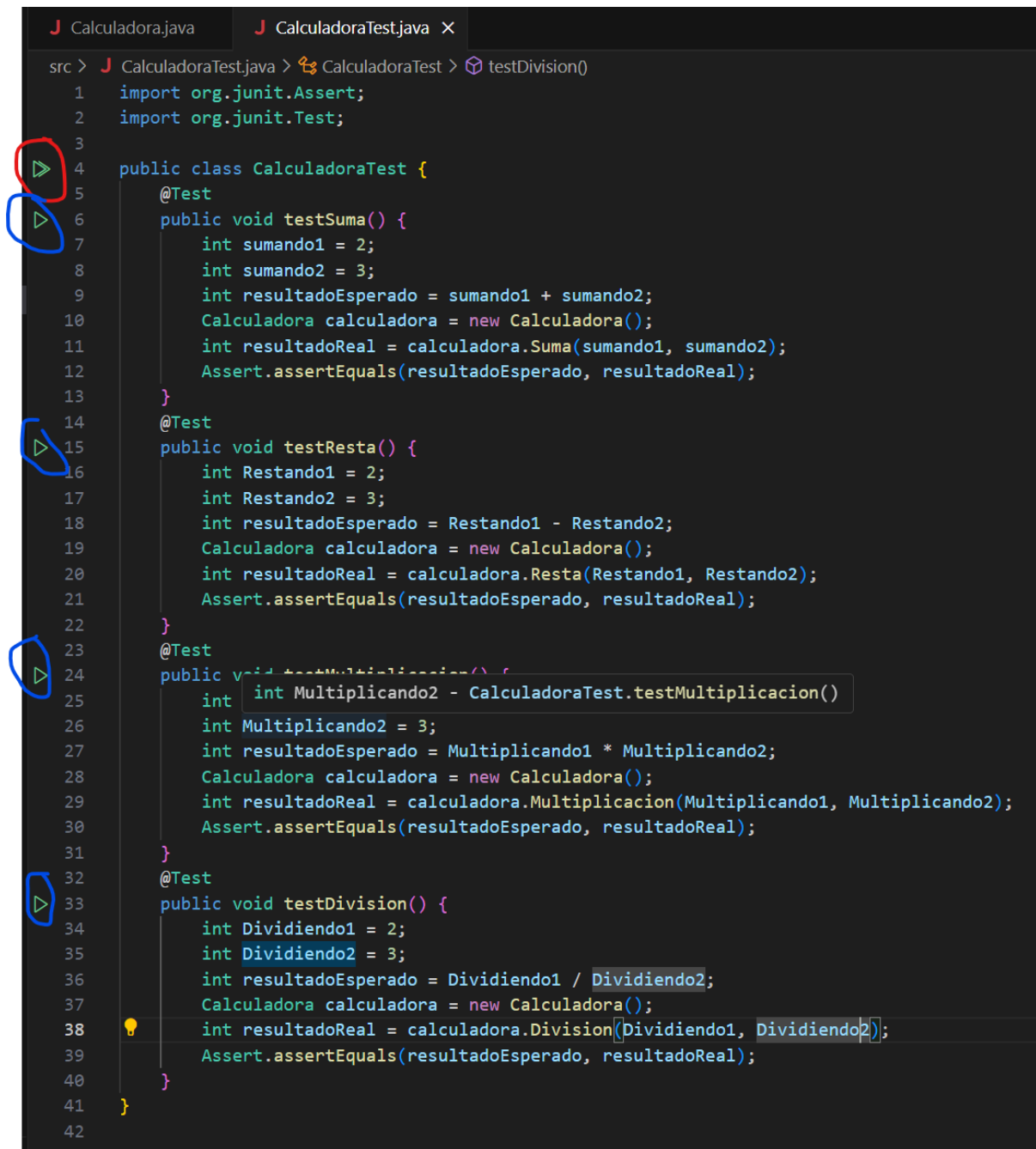


```
src > J Calculadora.java > Calculadora > Division(int, int)
1  public class Calculadora {
2      public int Suma(int sumando1, int sumando2){
3          int resultado = sumando1 + sumando2;
4          return resultado;
5      }
6      public int Resta(int Restando1, int Restando2){
7          int resultado = Restando1 - Restando2;
8          return resultado;
9      }
10     public int Multiplicacion(int Multiplicando1, int Multiplicando2){
11         int resultado = Multiplicando1 * Multiplicando2;
12         return resultado;
13     }
14     public int Division(int Dividiendo1, int Dividiendo2){
15         int resultado = Dividiendo1 / Dividiendo2;
16         return resultado;
17     }
18 }
19
```

Con lo hecho en la Tarea anterior creamos un archivo de **JUnit** para hacer los test en mi caso **CalculadoraTest.java** y creamos metodos para hacer los tests en este caso

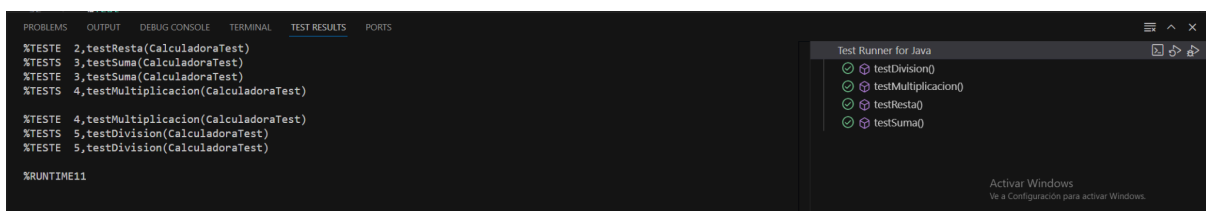
**TestSuma, TestResta, TestMultiplicación, TestDivision.**

Usamos los **Assert** que son métodos que trae la librería **JUnit** que permite comprobar si el resultado es correcto. En estos casos, comparamos el resultado que esperamos con el que realmente devuelve las funciones.



```
src > J CalculadoraTest.java > CalculadoraTest > testDivision()
1  import org.junit.Assert;
2  import org.junit.Test;
3
4  public class CalculadoraTest {
5      @Test
6      public void testSuma() {
7          int sumando1 = 2;
8          int sumando2 = 3;
9          int resultadoEsperado = sumando1 + sumando2;
10         Calculadora calculadora = new Calculadora();
11         int resultadoReal = calculadora.Suma(sumando1, sumando2);
12         Assert.assertEquals(resultadoEsperado, resultadoReal);
13     }
14     @Test
15     public void testResta() {
16         int Restando1 = 2;
17         int Restando2 = 3;
18         int resultadoEsperado = Restando1 - Restando2;
19         Calculadora calculadora = new Calculadora();
20         int resultadoReal = calculadora.Resta(Restando1, Restando2);
21         Assert.assertEquals(resultadoEsperado, resultadoReal);
22     }
23     @Test
24     public void testMultiplicacion() {
25         int Multiplicando1 = 2;
26         int Multiplicando2 = 3;
27         int resultadoEsperado = Multiplicando1 * Multiplicando2;
28         Calculadora calculadora = new Calculadora();
29         int resultadoReal = calculadora.Multiplicacion(Multiplicando1, Multiplicando2);
30         Assert.assertEquals(resultadoEsperado, resultadoReal);
31     }
32     @Test
33     public void testDivision() {
34         int Dividiendo1 = 2;
35         int Dividiendo2 = 3;
36         int resultadoEsperado = Dividiendo1 / Dividiendo2;
37         Calculadora calculadora = new Calculadora();
38         int resultadoReal = calculadora.Division(Dividiendo1, Dividiendo2);
39         Assert.assertEquals(resultadoEsperado, resultadoReal);
40     }
41 }
42
```

Si le das a las flechitas verdes se ejecutará el Test. Como se puede apreciar la Flechita en el círculo rojo ejecutaría todos los test y los azules solo uno específico.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
%TESTE 2,testResta(CalculadoraTest)
%TESTS 3,testSuma(CalculadoraTest)
%TESTE 3,testSuma(CalculadoraTest)
%TESTS 4,testMultiplicacion(CalculadoraTest)
%TESTE 4,testMultiplicacion(CalculadoraTest)
%TESTS 5,testDivision(CalculadoraTest)
%TESTE 5,testDivision(CalculadoraTest)
%RUNTIME11

Test Runner for Java
testDivision0
testMultiplicacion0
testResta0
testSuma0

Activar Windows
Ve a Configuración para activar Windows.
```

Al darle en este caso al darle a la flechita verde que estaba rodeada de rojo ejecutará todos los test y si no ha habido problemas debería salir esto.

## Ejercicio 2: Testing y Debugging

Ahora importamos el archivo ComparacionesEnteros.java. Como en el apartado anterior le creamos un archivo ComparacionesEnterosTest.java.

### (COMPARACIONES ENTEROS CORREGIDOS)

```
J ComparacionesEnteros.java X
src > J ComparacionesEnteros.java > ComparacionesEnteros
1 public class ComparacionesEnteros {
2
3     // 1. Verifica si los dos números son iguales
4     public static boolean sonIguales(int a, int b) {
5         boolean resultado = (a == b);
6         return resultado;
7     }
8
9     // 2. Verifica si el primer número es mayor que el segundo
10    public static boolean esMayor(int a, int b) {
11        boolean resultado = (a > b);
12        return resultado;
13    }
14
15    // 3. Verifica si el primer número es menor que el segundo
16    public static boolean esMenor(int a, int b) {
17        boolean resultado = (a < b);
18        return resultado;
19    }
20
21    // 4. Verifica si el primer número es divisible por el segundo (sin resto)
22    public static boolean esDivisible(int a, int b) {
23        // Primero verifica si el divisor es cero para evitar error
24        boolean divisorEsCero = (b == 0);
25        boolean resultado;
26
27        if (divisorEsCero) {
28            resultado = false;
29        } else {
30            int residuo = a % b;
31            resultado = (residuo == 0);
32        }
33
34        return resultado;
35    }
36
37    // 5. Verifica si ambos números son pares
38    public static boolean sonAmbosPares(int a, int b) {
39        boolean primerNumeroPar = (a % 2 == 0);
40        boolean segundoNumeroPar = (b % 2 == 0);
41        boolean resultado = primerNumeroPar & segundoNumeroPar;
42
43        return resultado;
44    }
45
46    // 6. Verifica si al menos uno de los dos números es positivo
47    public static boolean alMenosUnoPositivo(int a, int b) {
48        boolean primerNumeroPositivo = (a > 0);
49        boolean segundoNumeroPositivo = (b > 0);
50        boolean resultado = primerNumeroPositivo || segundoNumeroPositivo;
51
52        return resultado;
53    }
54
55    // 7. Verifica si la suma de los dos números es par
56    public static boolean sumaEsPar(int a, int b) {
57        int suma = a + b;
58        boolean sumaPar = (suma % 2 == 0);
59        return sumaPar;
60    }
61
62 }
```

**(TEST DE TODOS LOS CASOS DE COMPARACIONES ENTEROS):**

- Verifica si al menos uno de los dos números es positivo

```
5 public class ComparacionesEnterosTest {
6     @Test
7     public void testAlMenosUnoPositivoCaso1() {
8         //Arrange or given
9         int num1=1;
10        int num2=-3;
11        boolean expectedResult =true;
12        // Act or when
13        boolean actualResult = ComparacionesEnteros.alMenosUnoPositivo(num1, num2);
14        //Assert or then
15        Assert.assertEquals(expectedResult, actualResult);
16    }
17    @Test
18    public void testAlMenosUnoPositivoCaso2() {
19        int num1=-1;
20        int num2=-3;
21        boolean expectedResult =false;
22        // Act or when
23        boolean actualResult = ComparacionesEnteros.alMenosUnoPositivo(num1, num2);
24        //Assert or then
25        Assert.assertEquals(expectedResult, actualResult);
26    }
27    @Test
28    public void testAlMenosUnoPositivoCaso3() {
29        int num1=1;
30        int num2=3;
31        boolean expectedResult =true;
32        // Act or when
33        boolean actualResult = ComparacionesEnteros.alMenosUnoPositivo(num1, num2);
34        //Assert or then
35        Assert.assertEquals(expectedResult, actualResult);
36    }
37 }
```

- Verifica si el primer número es mayor que el segundo

```
71
72     @Test
73     public void testEsMayorcaso1() {
74         //Arrange or given
75         int num1=4;
76         int num2=3;
77         boolean expectedResult =true;
78         // Act or when
79         boolean actualResult = ComparacionesEnteros.esMayor(num1, num2);
80         //Assert or then
81         Assert.assertEquals(expectedResult, actualResult);
82     }
83     @Test
84     public void testEsMayorcaso2() {
85         //Arrange or given
86         int num1=3;
87         int num2=3;
88         boolean expectedResult =false;
89         // Act or when
90         boolean actualResult = ComparacionesEnteros.esMayor(num1, num2);
91         //Assert or then
92         Assert.assertEquals(expectedResult, actualResult);
93     }
94     @Test
95     public void testEsMayorcaso3() {
96         //Arrange or given
97         int num1=3;
98         int num2=4;
99         boolean expectedResult =false;
100        // Act or when
101        boolean actualResult = ComparacionesEnteros.esMayor(num1, num2);
102        //Assert or then
103        Assert.assertEquals(expectedResult, actualResult);
104    }
105
```

- Verifica si el primer número es menor que el segundo

```
105
106     @Test
107     public void testEsMenorcaso1() {
108         //Arrange or given
109         int num1=2;
110         int num2=3;
111         boolean expectedResult =true;
112         // Act or when
113         boolean actualResult = ComparacionesEnteros.esMenor(num1, num2);
114         //Assert or then
115         Assert.assertEquals(expectedResult, actualResult);
116     }
117     @Test
118     public void testEsMenorcaso2() {
119         int num1=3;
120         int num2=3;
121         boolean expectedResult =false;
122         // Act or when
123         boolean actualResult = ComparacionesEnteros.esMenor(num1, num2);
124         //Assert or then
125         Assert.assertEquals(expectedResult, actualResult);
126     }
127     @Test
128     public void testEsMenorcaso3() {
129         int num1=4;
130         int num2=3;
131         boolean expectedResult =false;
132         // Act or when
133         boolean actualResult = ComparacionesEnteros.esMenor(num1, num2);
134         //Assert or then
135         Assert.assertEquals(expectedResult, actualResult);
136     }
137 }
```

- Verifica si ambos números son pares

```
138     @Test
139     public void testSonAmbosParescaso1() {
140         //Arrange or given
141         int num1=2;
142         int num2=2;
143         boolean expectedResult =true;
144         // Act or when
145         boolean actualResult = ComparacionesEnteros.sonAmbosPares(num1, num2);
146         //Assert or then
147         Assert.assertEquals(expectedResult, actualResult);
148     }
149     @Test
150     public void testSonAmbosParescaso2() {
151         int num1=1;
152         int num2=2;
153         boolean expectedResult =false;
154         // Act or when
155         boolean actualResult = ComparacionesEnteros.sonAmbosPares(num1, num2);
156         //Assert or then
157         Assert.assertEquals(expectedResult, actualResult);
158     }
159 }
```

- Verifica si los dos números son iguales

```
160     @Test
161     public void testSonIgualescaso1() {
162         //Arrange or given
163         int num1=3;
164         int num2=3;
165         boolean expectedResult =true;
166         // Act or when
167         boolean actualResult = ComparacionesEnteros.sonIguales(num1, num2);
168         //Assert or then
169         Assert.assertEquals(expectedResult, actualResult);
170     }
171     @Test
172     public void testSonIgualescaso2() {
173         //Arrange or given
174         int num1=2;
175         int num2=3;
176         boolean expectedResult =false;
177         // Act or when
178         boolean actualResult = ComparacionesEnteros.sonIguales(num1, num2);
179         //Assert or then
180         Assert.assertEquals(expectedResult, actualResult);
181     }
182 }
```



- Verifica si la suma de los dos números es par

```
182
183     @Test
184     public void testSumaEsParcaso1() {
185         //Arrange or given
186         int num1=2;
187         int num2=3;
188         boolean expectedResult =false;
189         // Act or when
190         boolean actualResult = ComparacionesEnteros.sonIguales(num1, num2);
191         //Assert or then
192         Assert.assertEquals(expectedResult, actualResult);
193     }
194     @Test
195     public void testSumaEsParcaso2() {
196         //Arrange or given
197         int num1=2;
198         int num2=2;
199         boolean expectedResult =true;
200         // Act or when
201         boolean actualResult = ComparacionesEnteros.sonIguales(num1, num2);
202         //Assert or then
203         Assert.assertEquals(expectedResult, actualResult);
204     }
205 }
```

- Verifica si el primer número es divisible por el segundo (sin resto)  
Primero verifica si el divisor es cero para evitar error

```
38      @Test
39      public void testEsDivisiblecaso1() {
40          //Arrange or given
41          int num1=4;
42          int num2=2;
43          boolean expectedResult =true;
44          // Act or when
45          boolean actualResult = ComparacionesEnteros.esDivisible(num1, num2);
46          //Assert or then
47          Assert.assertEquals(expectedResult, actualResult);
48      }
49      @Test
50      public void testEsDivisiblecaso2() {
51          //Arrange or given
52          int num1=4;
53          int num2=3;
54          boolean expectedResult =false;
55          // Act or when
56          boolean actualResult = ComparacionesEnteros.esDivisible(num1, num2);
57          //Assert or then
58          Assert.assertEquals(expectedResult, actualResult);
59      }
60      @Test
61      public void testEsDivisiblecaso3() {
62          //Arrange or given
63          int num1=4;
64          int num2=0;
65          boolean expectedResult =false;
66          // Act or when
67          boolean actualResult = ComparacionesEnteros.esDivisible(num1, num2);
68          //Assert or then
69          Assert.assertEquals(expectedResult, actualResult);
70      }
71  }
```

## (PRUEBA DE FUNCIONAMIENTO)

```
✓  ComparacionesEnterosTest
  ✓  testAlMenosUnoPositivoCaso1()
  ✓  testAlMenosUnoPositivoCaso2()
  ✓  testAlMenosUnoPositivoCaso3()
  ✓  testEsDivisiblecaso1()
  ✓  testEsDivisiblecaso2()
  ✓  testEsDivisiblecaso3()
  ✓  testEsMayorcaso1()
  ✓  testEsMayorcaso2()
  ✓  testEsMayorcaso3()
  ✓  testEsMenorcaso1()
  ✓  testEsMenorcaso2()
  ✓  testEsMenorcaso3()
  ✓  testSonAmbosParescaso1()
  ✓  testSonAmbosParescaso2()
  ✓  testSonIgualescaso1()
  ✓  testSonIgualescaso2()
  ✓  testSumaEsParcaso1()
  ✓  testSumaEsParcaso2()
> ✓  OperacionesMixtasTest
```

**(ARCHIVO OPERACIONES MIXTAS CORREGIDO):**

```
J OperacionesMixtas.java X
src > J OperacionesMixtas.java > OperacionesMixtas > formatearNumero(float, int)
1  public class OperacionesMixtas {
2      /*
3      */
9      public static String concatenarNumeroTexto(int numero, String texto) {
10         String numeroComoTexto = Integer.toString(numero);
11         String resultado = numeroComoTexto + " " + texto;
12         return resultado;
13     }
14
15
16     /*
17     * 2. Devuelve el String "verdadero" o "falso" dependiendo del valor booleano que recibe
18     */
19     public static String booleanComoTexto(boolean valor) {
20         String resultado;
21         if (valor) {
22             resultado = "verdadero";
23         } else {
24             resultado = "falso";
25         }
26         return resultado;
27     }
28
29     /*
30     * 3. Concatena dos cadenas de texto dejando un espacio en medio
31     * Ejemplo: concatenarTextos("Hola", "Mundo");
32     * devuelve: "Hola Mundo"
33     */
34     public static String concatenarTextos(String texto1, String texto2) {
35         String resultado = texto1 + " " + texto2;
36         return resultado;
37     }
38
39     /*
40     * 4. Devuelve un mensaje sobre si el número dado es positivo, negativo o cero
41     */
42     public static String descripcionNumero(int numero) {
43         String resultado;
44         if (numero < 0) {
45             resultado = "El número es negativo";
46         } else if (numero > 0) {
47             resultado = "El número es positivo";
48         } else {
49             resultado = "El número es cero";
50         }
51         return resultado;
52     }
53 }
```

```
J OperacionesMixtas.java X
src > J OperacionesMixtas.java > OperacionesMixtas > formatearNumero(float, int)
1 public class OperacionesMixtas {
2
3
4
54
55     /*
56      * 5. Devuelve el texto del primer parámetro
57      * en mayúsculas o minúsculas dependiendo del
58      * valor booleano del segundo parámetro
59      */
60     public static String cambiarTextoAMayusculas(String texto, boolean aMayusculas) {
61         String resultado;
62         if (aMayusculas) {
63             resultado = texto.toUpperCase();
64         } else {
65             resultado = texto.toLowerCase();
66         }
67         return resultado;
68     }
69
70     /*
71      * 6. Devuelve una descripción con dos valores flotantes concatenados
72      * con dos decimales y el símbolo de Euro detrás
73      * Ejemplo: (descripcionConFloat(3.14f, 2.71f));
74      * devuelve: Los valores son: 3.14€ y 2.71€
75      */
76     public static String descripcionConFloat(float valor1, float valor2) {
77         String valor1ComoTexto = Float.toString(valor1) + "€";
78         String valor2ComoTexto = Float.toString(valor2) + "€";
79         String resultado = "Los valores son: " + valor1ComoTexto + " y " + valor2ComoTexto;
80         return resultado;
81     }
82
83     /*
84      * 7. Recibe un float y lo devuelve como String con el número de decimales
85      * que se le pase como segundo parámetro.
86      * Ejemplo: formatearNumero(17.89945f, 2);
87      * devuelve: "17.90" (se redondea automáticamente)
88      */
89     public static String formatearNumero(float numero, int decimales) {
90         String numeroFormateado = String.format("%"+decimales+".f", numero);
91         String resultado = numeroFormateado;
92         return resultado;
93     }
94 }
```

## (OPERACIONES MIXTAS TESTS):

- Devuelve el String "verdadero" o "falso" dependiendo del valor booleano que recibe

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
1  import org.junit.Assert;
2  import org.junit.Test;
3
4  public class OperacionesMixtasTest {
5      @Test
6      public void testBooleanComoTextocaso1() {
7          //Arrange or given
8          boolean valor=true;
9          String expectedResult = "verdadero";
10         // Act or when
11         String actualResult = OperacionesMixtas.booleanComoTexto(valor);
12         //Assert or then
13         Assert.assertEquals(expectedResult, actualResult);
14     }
15     @Test
16     public void testBooleanComoTextocaso2() {
17         //Arrange or given
18         boolean valor=false;
19         String expectedResult = "falso";
20         // Act or when
21         String actualResult = OperacionesMixtas.booleanComoTexto(valor);
22         //Assert or then
23         Assert.assertEquals(expectedResult, actualResult);
24     }
25 }
```

- Devuelve el texto del primer parámetro en mayúsculas o minúsculas dependiendo del valor booleano del segundo parámetro.

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
4  public class OperacionesMixtasTest {
5      }
6
25
26     @Test
27     public void testCambiarTextoAMayusculascaso1() {
28         String texto= "TextTaco";
29         boolean aMayusculas= true;
30         //Arrange or given
31         String expectedResult = "TEXTTACO";
32         // Act or when
33         String actualResult = OperacionesMixtas.cambiarTextoAMayusculas(texto, aMayusculas);
34         //Assert or then
35         Assert.assertEquals(expectedResult, actualResult);
36     }
37     @Test
38     public void testCambiarTextoAMayusculascaso2() {
39         String texto= "TextTaco";
40         boolean aMayusculas= false;
41         //Arrange or given
42         String expectedResult = "texttaco";
43         // Act or when
44         String actualResult = OperacionesMixtas.cambiarTextoAMayusculas(texto, aMayusculas);
45         //Assert or then
46         Assert.assertEquals(expectedResult, actualResult);
47     }
48 }
```

- Concatena un número y una cadena de texto dejando un espacio en medio.  
Ejemplo: concatenarNumeroTexto(5, "manzanas"); devuelve: "5 manzanas".

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
4   public class OperacionesMixtasTest {
47  }
48
49  @Test
50  public void testConcatenarNumeroTexto() {
51      int numero=5;
52      String texto= "manzanas";
53      //Arrange or given
54      String expectedResult = "5 manzanas";
55      // Act or when
56      String actualResult = OperacionesMixtas.concatenarNumeroTexto(numero, texto);
57      //Assert or then
58      Assert.assertEquals(expectedResult, actualResult);
59  }
60
```

- Concatena dos cadenas de texto dejando un espacio en medio.  
Ejemplo: concatenarTextos("Hola", "Mundo"); devuelve: "Hola Mundo"

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
4   public class OperacionesMixtasTest {
60
61  @Test
62  public void testConcatenarTextos() {
63      String texto1="Hola";
64      String texto2="Mundo";
65      //Arrange or given
66      String expectedResult = "Hola Mundo";
67      // Act or when
68      String actualResult = OperacionesMixtas.concatenarTextos(texto1, texto2);
69      //Assert or then
70      Assert.assertEquals(expectedResult, actualResult);
71  }

```

- Devuelve una descripción con dos valores flotantes concatenados con dos decimales y el símbolo de Euro detrás. Ejemplo: (descripcionConFloat(3.14f, 2.71f); devuelve: Los valores son: 3.14€ y 2.71€

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
4   public class OperacionesMixtasTest {
73  @Test
74  public void testDescripcionConFloat() {
75      float valor1=3.14f;
76      float valor2=2.71f;
77      //Arrange or given
78      String expectedResult = "Los valores son: 3.14€ y 2.71€";
79      // Act or when
80      String actualResult = OperacionesMixtas.descripcionConFloat(valor1, valor2);
81      //Assert or then
82      Assert.assertEquals(expectedResult, actualResult);
83  }
84
```

- Devuelve un mensaje sobre si el número dado es positivo, negativo o cero

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
4   public class OperacionesMixtasTest {
85  @Test
86  public void testDescripcionNumero() {
87      int valor1=3;
88      //Arrange or given
89      String expectedResult = "El número es positivo";
90      // Act or when
91      String actualResult = OperacionesMixtas.descripcionNumero(valor1);
92      //Assert or then
93      Assert.assertEquals(expectedResult, actualResult);
94  }
95
```



- Recibe un float y lo devuelve como String con en número de decimales que se le pase como segundo parámetro. Ejemplo: `formatearNumero(17.89945f, 2);` devuelve: "17.90" (se redondea automáticamente)

```
J OperacionesMixtasTest.java M X
src > J OperacionesMixtasTest.java > ...
 4  public class OperacionesMixtasTest {
95
96      @Test
97      public void testFormatearNumero() {
98          float descripcion = 17.89945f;
99          int numero=2;
100          //Arrange or given
101          String expectedResult = "17,90";
102          // Act or when
103          String actualResult = OperacionesMixtas.formatearNumero(descripcion, numero);
104          //Assert or then
105          Assert.assertEquals(expectedResult, actualResult);
106      }
107  }
108
```

## (PRUEBA DE FUNCIONAMIENTO)

```
✓ OperacionesMixtasTest 35ms
  ✓ testBooleanComoTextocaso1()...
  ✓ testBooleanComoTextocaso2()...
  ✓ testCambiarTextoAMayusculasc...
  ✓ testCambiarTextoAMayusculasc...
  ✓ testConcatenarNumeroTexto()...
  ✓ testConcatenarTextos() 1.0ms
  ✓ testDescripcionConFloat() 12ms
  ✓ testDescripcionNumero() 0.0ms
  ✓ testFormatearNumero() 20ms
```