PRO 5.3 - Junit con pruebas parametrizadas

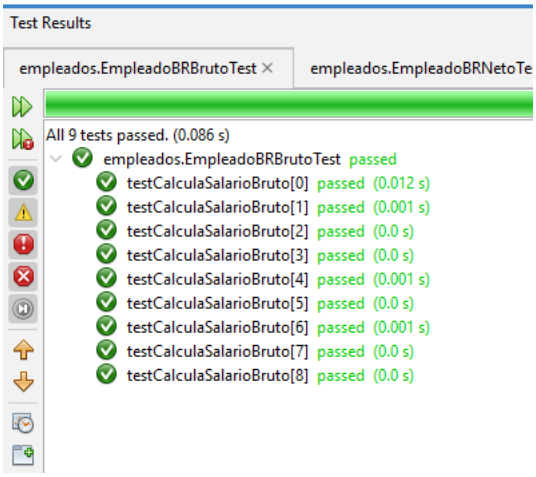
1. Clase EmpleadoBRBrutoTest

El código de la clase EmpleadoBRBrutoTest para hacer las pruebas parametrizadas es el siguiente:

```
package empleados;

import static empleados.Tipoempleado.vendedor;
import java.util.Arrays;
import java.util.Collection;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class EmpleadoBRBrutoTest {

    private Tipoempleado tipo;
    private float ventas;
    private float horasExtra;
    private Object salarioBruto;

public EmpleadoBRBrutoTest(Tipoempleado tipo, float ventas, float horasExtra, Object salarioBruto) {
    this.tipo = tipo;
    this.ventas = ventas;
    this.horasExtra = horasExtra;
    this.salarioBruto = salarioBruto;
}

@BeforeClass
public static void setUpClass() {
}

@AfterClass
public static void tearDownClass() {
}

@Before
public void setUp() {
}

@After
public void tearDown() {
}

@Parameters
public static Collection<Object[]> salario(){
    return Arrays.asList(new Object[][]{{Tipoempleado.vendedor, 2000f, 8f, 1360f}, {Tipoempleado.vendedor, 1500f, 3f, 1260f},
        {Tipoempleado.vendedor, 1499.99f, 0f, 1100f}, {Tipoempleado.encargado, 1250f, 8f, 1760f}, {Tipoempleado.encargado, 1000f, 0f, 1600f},
        {Tipoempleado.encargado, 999.99f, 3f, 1560f}, {Tipoempleado.encargado, 500f, 0f, 1500f}, {Tipoempleado.encargado, 0f, 8f, 1660f},
        {null, 1500f, 8f, BRException.class}});
}
```

```java
    @Test
    public void testCalculaSalarioBruto() throws Exception {

        EmpleadoBR empleado = new EmpleadoBR(tipo: this.tipo, ventasMes: this.ventas, horasExtra: this.horasExtra);

        try {
            float salario = empleado.calculaSalarioBruto(tipo: this.tipo, ventasMes: this.ventas, horasExtra: this.horasExtra);
            assertEquals((Float)this.salarioBruto, actual: salario, delta: 0.01);
        } catch(BRException e){
            if(this.salarioBruto == BRException.class){
                return;
            } else {
                fail(message:"Excepción inesperada");
            }
        }

    }

}
```

Resultado de las pruebas:



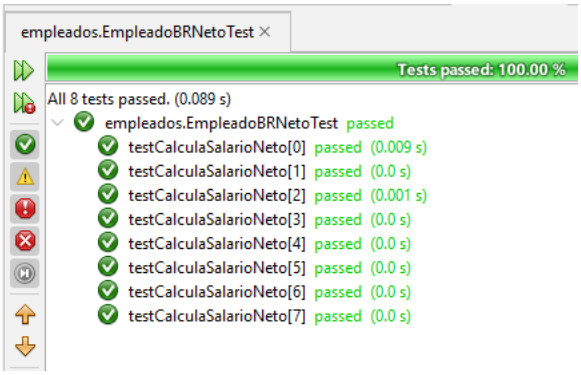
Se ha tenido que añadir a la clase EmpleadoBR dos constructores para hacer las pruebas de salario bruto y de salario neto (mostrada abajo), y las variables a probar:

```java
private Tipoempleado tipo;
private float ventasMes;
private float horasExtra;

public EmpleadoBR(Tipoempleado tipo, float ventasMes, float horasExtra){
    this.tipo = tipo;
    this.ventasMes = ventasMes;
    this.horasExtra = horasExtra;
}

public EmpleadoBR(float ventasMes){
    this.ventasMes = ventasMes;
}
```

2. Clase EmpleadoBRNeto Test

```
package empleados;

import java.util.Arrays;
import java.util.Collection;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class EmpleadoBRNetoTest {

    private float ventasMes;
    private float salarioNeto;

    public EmpleadoBRNetoTest(float ventasMes, float salarioNeto) {
        this.ventasMes = ventasMes;
        this.salarioNeto = salarioNeto;
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    @Parameters
    public static Collection<Object[]> salario(){
        return Arrays.asList(new Object[][]{{2000f, 1640f},{1500f, 1230f},{1499.99f, 1259.9916f},{1250f, 1050f},{1000f, 840f},
            {999.99f, 999.99f},{500f, 500f},{0f, 0f}});
    }


    /**
     * Test of calculaSalarioNeto method, of class EmpleadoBR.
     */
    @Test
    public void testCalculaSalarioNeto() throws Exception {
        EmpleadoBR empleado = new EmpleadoBR(ventasMes: this.ventasMes);
        float resultado = empleado.calculaSalarioNeto(salarioBruto: this.ventasMes);
        assertEquals(expected: this.salarioNeto, actual: resultado, delta: 0.01);
    }

}
```

Resultado de las pruebas:



3. Suite de pruebas conteniendo las clases anteriores

```
@RunWith(Suite.class)
@Suite.SuiteClasses({empleados.EmpleadoBRBrutoTest.class, empleados.EmpleadoBRNetoTest.class})
```