

Práctica 5 - Entornos de Desarrollo

Instrucciones:

1. Crea un nuevo proyecto en github, pon nombre Practica5.
2. Crea la interfaz ICalculadora con las operaciones sumar, restar, multiplicar y dividir.
3. Crea la clase calculadora que utilice la interfaz anterior e implementa los métodos.
4. Realiza el primer commit a github.
5. Luego, crea los test unitarios en JUnit (igual no es necesario descargarlo porque puede venir incluido en vuestro IDE) de estos cuatro métodos.
6. Realiza el segundo commit.
7. Realiza diferentes pruebas para validar que los test funcionan y expónlos resultados obtenidos, % de tests pasados etc.

Nota: Si deseas ver los criterios y algunos recursos de utilidad para esta práctica lo que puede en este pdf del enunciado.

Procedimiento

1. **Capturas del código:**
 - Calculadora.java (Clase):

```

package edu.poniperro;

public class Calculadora implements
    ICalculadora{
    /**
     *
     * Clase principal Calculadora implementa
     * ndo la interfaz
     *
     */

    /**
     * obtenemos los dos numeros y realizamo
     * s la suma
     * @param numero1
     * @param numero2
     * @return int
     */
    @Override
    public int sumar(int numero1, int
    numero2) {
        return numero1 + numero2;
    }

    /**
     * obtenemos los dos numeros y realizamo
     * s la resta
     *
     * @param numero1
     * @param numero2
     * @return double
     */
    @Override
    public int restar(int numero1, int
    numero2) {
        return numero1 - numero2;
    }

    /**
     * obtenemos los dos numeros y realizamos
     * la multiplicacion
     * @param numero1
     * @param numero2
     * @return int
     */

    @Override
    public int multiplicar(int numero1, int
    numero2) {
        return numero1 * numero2;
    }

    /**
     * obtenemos los dos numeros y realizamos
     * la division
     *
     * @param numero1
     * @param numero2
     * @return int
     */

    @Override
    public int dividir(int numero1, int
    numero2) {
        return numero1 / numero2;
    }
}

```

- ICalculadora.java (Interfaz):

```
package edu.poniperro;

public interface ICalculadora {

    int sumar(int numero1, int numero2);

    int restar(int numero1, int numero2);

    int multiplicar(int numero1, int numero2
);

    int dividir(int numero1, int numero2);
}
```

2. Tests:

- ICalculadoraTest.java:

```

package edu.poniperro;

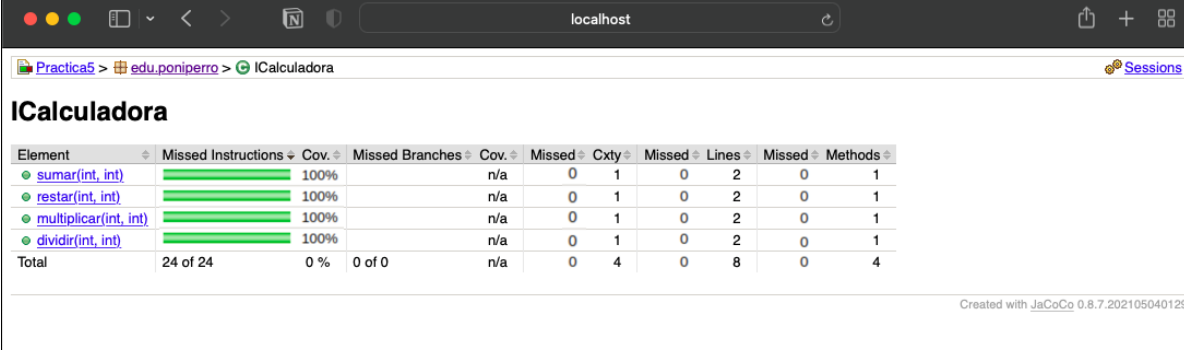
import org.junit.Before;
import org.junit.Test;

import static
org.junit.Assert.assertArrayEqu
als
;
import static
org.junit.Assert.assertEquals;

public class ICalculadoraTest {
    ICalculadora calc = null;
    @Before
    public void iniciar(){
        calc = new Calculadora
();
    }
    @Test
    public void testSumar(){
        assertEquals("1+1", 2,
calc.sumar(1,1),.001);
    }
    @Test
    public void testRestar(){
        assertEquals("4-2", 2,
calc.restar(4,2),.001);
    }
    @Test
    public void testMultiplicar
(){
        assertEquals("4x4",16,
calc.multiplicar(4, 4), .001);
    }
    @Test
    public void testDividir(){
        assertEquals("4/2",2,
calc.dividir(4, 2),.001);
    }
}

```

3. Balance de tests pasados:

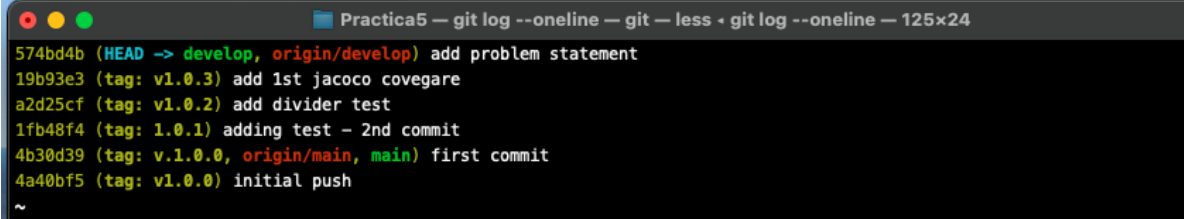


The screenshot shows a web browser displaying the JaCoCo report for a project named 'ICalculadora'. The report is titled 'ICalculadora' and shows a table with columns: Element, Missed Instructions, Cov., Missed Branches, Cov., Missed Ctxy, Missed Lines, Missed Methods. The table lists four elements: sumar(int, int), restar(int, int), multiplicar(int, int), and dividir(int, int). Each element has a green bar indicating 100% coverage. The total row shows 24 of 24 instructions covered, 0% coverage for missed instructions, 0 of 0 missed branches, n/a for missed ctxy, 0 of 0 missed lines, and 0 of 0 missed methods.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Ctxy	Missed Lines	Missed Methods
sumar(int, int)	0	100%	n/a	n/a	0	2	0
restar(int, int)	0	100%	n/a	n/a	0	2	0
multiplicar(int, int)	0	100%	n/a	n/a	0	2	0
dividir(int, int)	0	100%	n/a	n/a	0	2	0
Total	24 of 24	0 %	0 of 0	n/a	0	8	4

Created with JaCoCo 0.8.7.202105040129

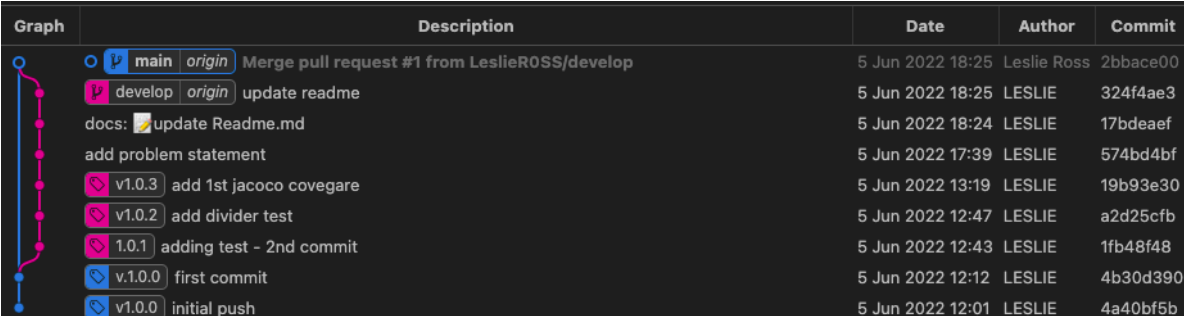
4. Git log y versionado semántico:



The screenshot shows the output of the command 'git log --oneline' in a terminal window. The output lists the commit history from the HEAD (develop) branch, showing the initial push, first commit, and subsequent commits for adding tests and updating the README.

```
574bd4b (HEAD -> develop, origin/develop) add problem statement
19b93e3 (tag: v1.0.3) add 1st jacoco coverage
a2d25cf (tag: v1.0.2) add divider test
1fb48f4 (tag: 1.0.1) adding test - 2nd commit
4b30d39 (tag: v.1.0.0, origin/main, main) first commit
4a40bf5 (tag: v1.0.0) initial push
```

5. Git graph, working tree:



The screenshot shows the Git graph and commit history. The graph on the left shows the main branch (blue) and the develop branch (pink). The commit history table on the right lists the commits with their descriptions, dates, authors, and commit hashes.

Graph	Description	Date	Author	Commit
main origin	Merge pull request #1 from LeslieROSS/develop	5 Jun 2022 18:25	Leslie Ross	2bbace00
develop origin	update readme	5 Jun 2022 18:25	LESLIE	324f4ae3
docs: update Readme.md		5 Jun 2022 18:24	LESLIE	17bdeaef
add problem statement		5 Jun 2022 17:39	LESLIE	574bd4bf
v1.0.3	add 1st jacoco coverage	5 Jun 2022 13:19	LESLIE	19b93e30
v1.0.2	add divider test	5 Jun 2022 12:47	LESLIE	a2d25cfb
1.0.1	adding test - 2nd commit	5 Jun 2022 12:43	LESLIE	1fb48f48
v.1.0.0	first commit	5 Jun 2022 12:12	LESLIE	4b30d390
v1.0.0	initial push	5 Jun 2022 12:01	LESLIE	4a40bf5b

Trabajo realizado por



Leslie Ross Aranibar Pozo - DAW dual

