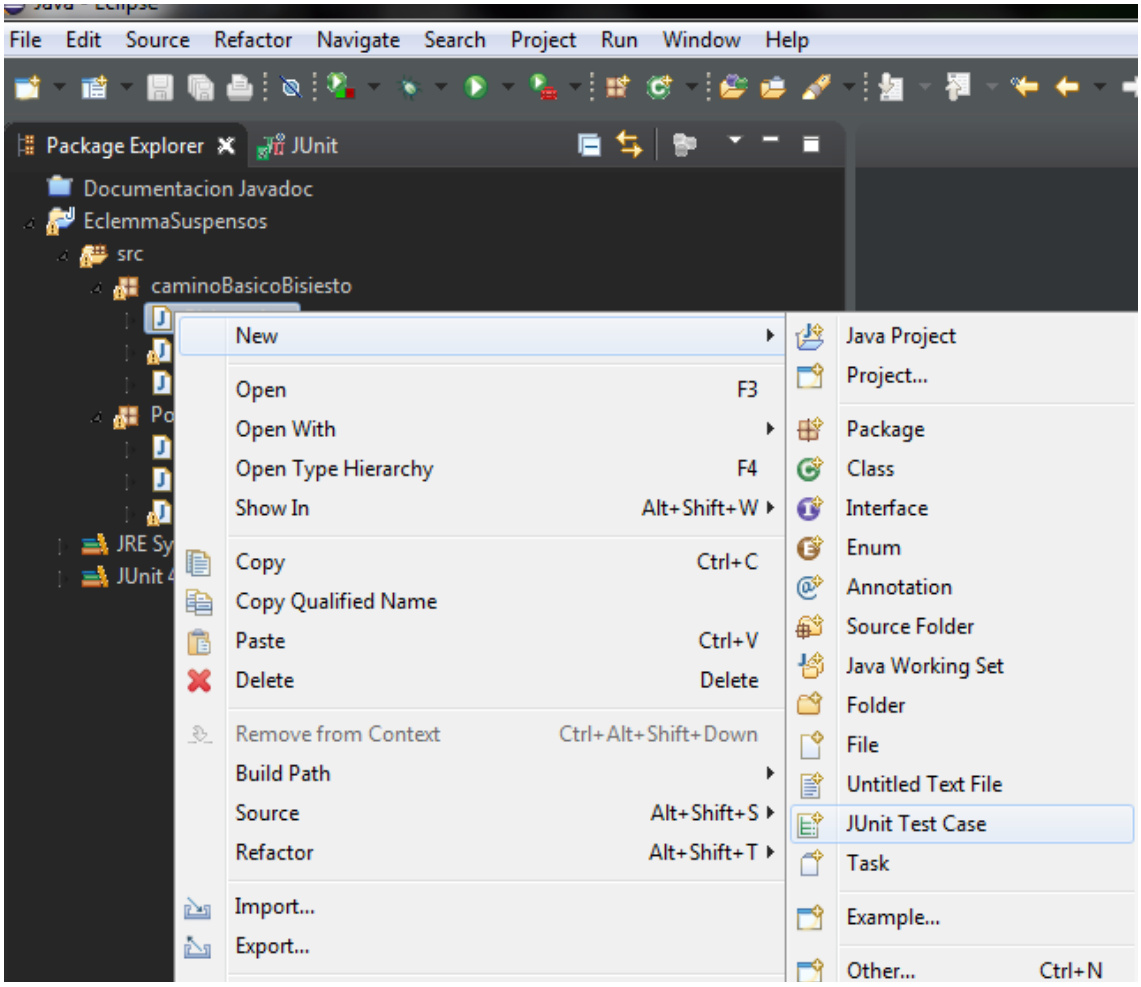
Eclemma

Autor: Antonio Luque Bravo

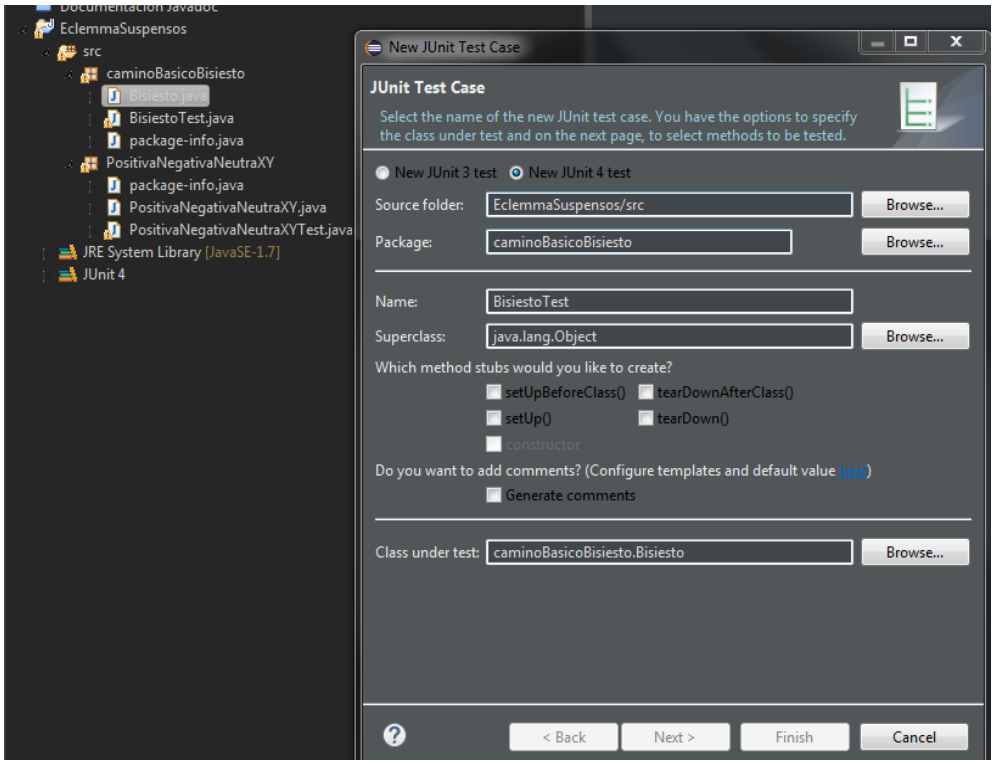
1 ECLEMMA, COMO GENERAR LAS PRUEBAS Y COMO REALIZARLAS.

1.1 Código de Bisiesto.

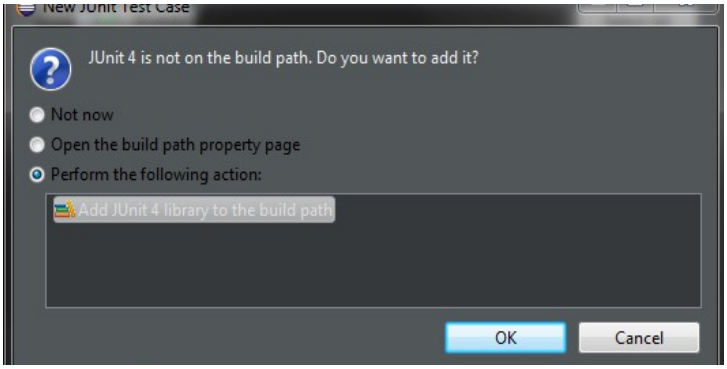
Para poder generar las pruebas, debemos irnos a nuestro código en Eclipse. Una vez allí damos clic derecho en la clase que queremos hacer las pruebas y creamos un JUnit Test Case.



Acto seguido configuramos el JUnit Test Case tal y como se muestra en la siguiente imagen:



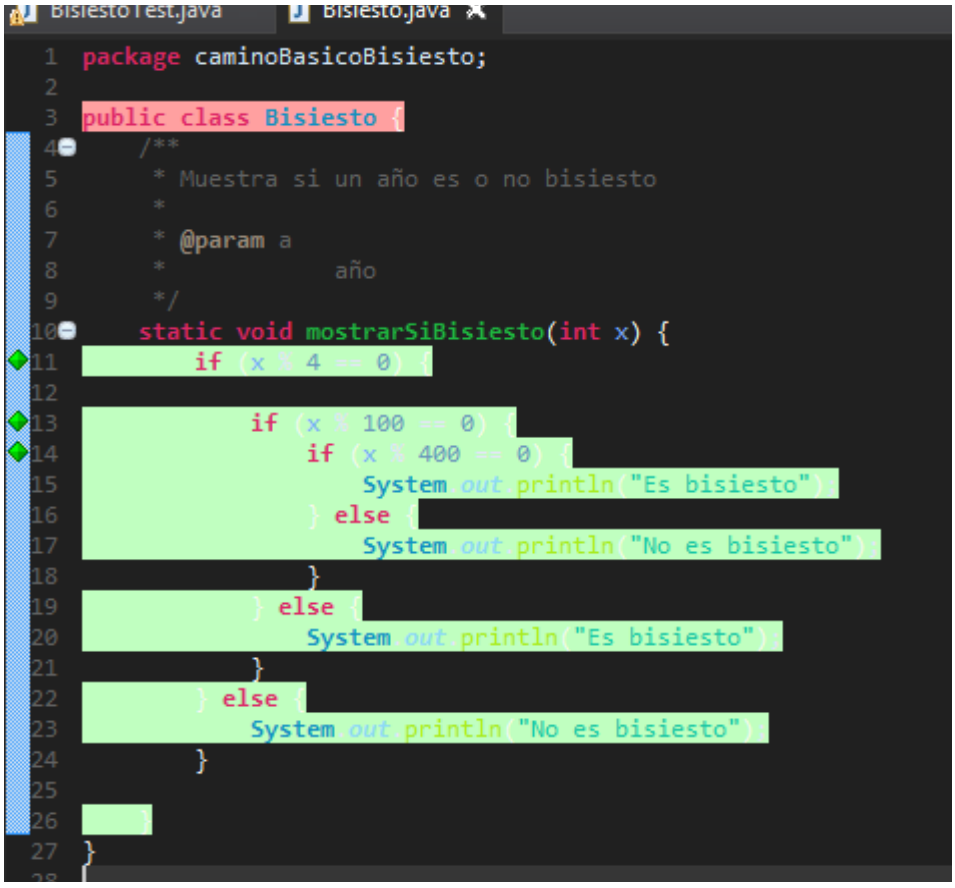
Cuando terminemos de configurarlo le damos a Finalizar, y en esa clase empezaremos a hacer nuestras correspondientes pruebas, seguidamente nos pedirá que añadamos la librería de JUnit 4, le decimos que sí y ya se creara la clase donde hacemos las pruebas de nuestro código (Bisiesto.java).



Debemos de hacer las pruebas dependiendo de la complejidad ciclomática de la clase:

```java
package caminoBasicoBisiesto;

import static org.junit.Assert.*;

public class BisiestoTest {

    @Test
    public void testMostrarSiBisiesto() {
        Bisiesto.mostrarSiBisiesto(100);
        Bisiesto.mostrarSiBisiesto(2016);
        Bisiesto.mostrarSiBisiesto(2);
        Bisiesto.mostrarSiBisiesto(0);
    }
}
```

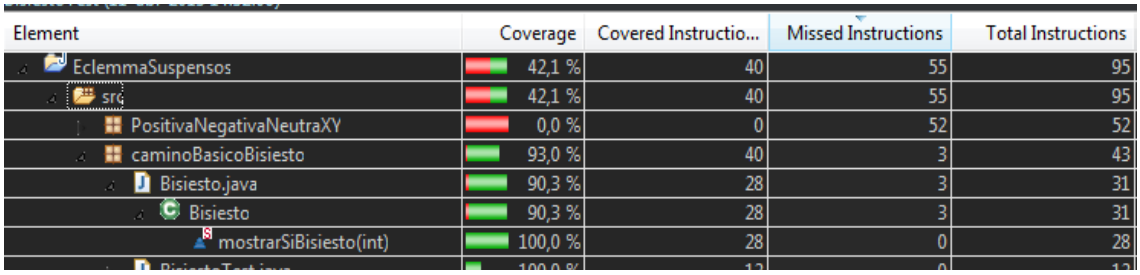
Cuando nos salga todo el método que estamos probando en color verde, hemos hecho bien nuestras pruebas, hemos cubierto todas las posibilidades de nuestra clase.



```java
package caminoBasicoBisiesto;

public class Bisiesto {
    /**
     * Muestra si un año es o no bisiesto
     *
     * @param a
     *            año
     */
    static void mostrarSiBisiesto(int x) {
        if (x % 4 == 0) {

            if (x % 100 == 0) {
                if (x % 400 == 0) {
                    System.out.println("Es bisiesto");
                } else {
                    System.out.println("No es bisiesto");
                }
            } else {
                System.out.println("Es bisiesto");
            }
        } else {
            System.out.println("No es bisiesto");
        }

    }
}
```

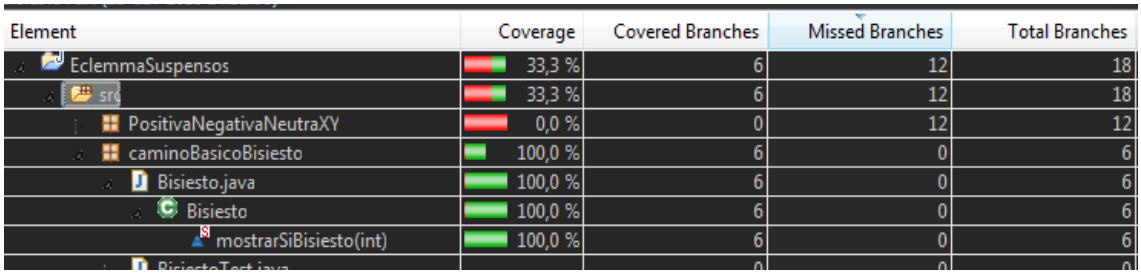
Tenemos que cubrir todas las ramas hasta que en el Coverage nos salga 100% en el método que estamos probando, he aquí los distintos tipos de coberturas:

Vista de las instrucciones cubiertas:



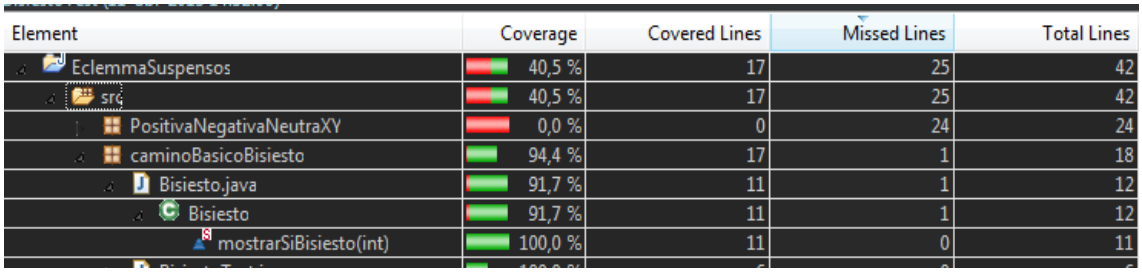
Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
EclemmaSuspensos	42,1 %	40	55	95
src	42,1 %	40	55	95
PositivaNegativaNeutraXY	0,0 %	0	52	52
caminoBasicoBisiesto	93,0 %	40	3	43
Bisiesto.java	90,3 %	28	3	31
Bisiesto	90,3 %	28	3	31
mostrarSiBisiesto(int)	100,0 %	28	0	28

Vista de las ramas cubiertas:



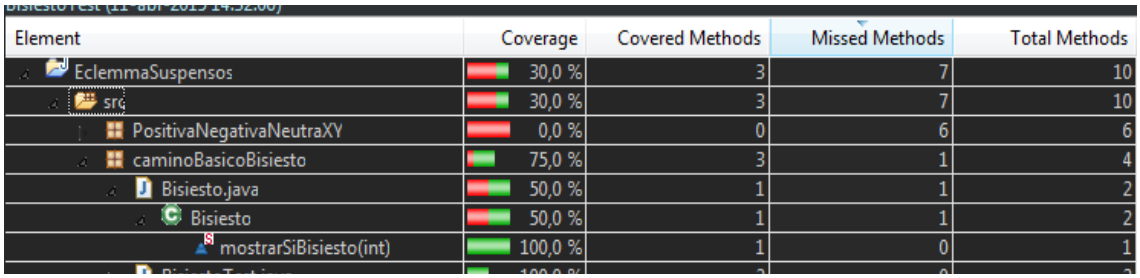
Element	Coverage	Covered Branches	Missed Branches	Total Branches
EclemmaSuspensos	33,3 %	6	12	18
src	33,3 %	6	12	18
PositivaNegativaNeutraXY	0,0 %	0	12	12
caminoBasicoBisiesto	100,0 %	6	0	6
Bisiesto.java	100,0 %	6	0	6
Bisiesto	100,0 %	6	0	6
mostrarSiBisiesto(int)	100,0 %	6	0	6

Vista de las líneas de código cubiertas:



Element	Coverage	Covered Lines	Missed Lines	Total Lines
EclemmaSuspensos	40,5 %	17	25	42
src	40,5 %	17	25	42
PositivaNegativaNeutraXY	0,0 %	0	24	24
caminoBasicoBisiesto	94,4 %	17	1	18
Bisiesto.java	91,7 %	11	1	12
Bisiesto	91,7 %	11	1	12
mostrarSiBisiesto(int)	100,0 %	11	0	11

Vista de los métodos cubiertos:

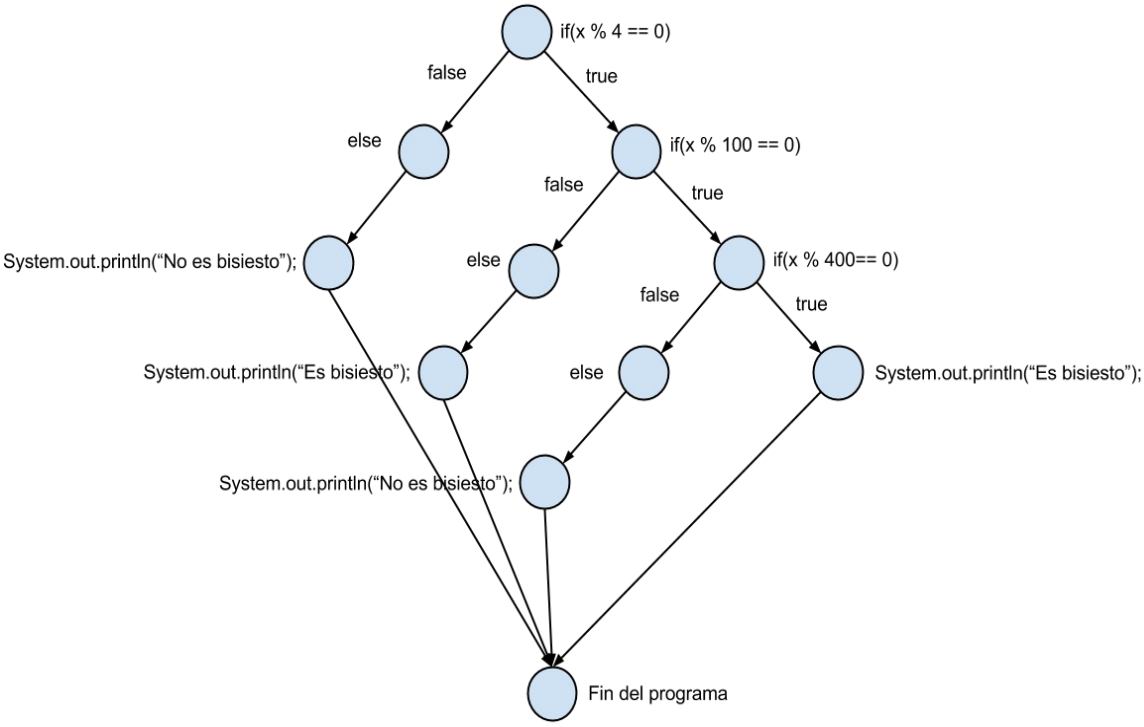


Element	Coverage	Covered Methods	Missed Methods	Total Methods
EclemmaSuspensos	30,0 %	3	7	10
src	30,0 %	3	7	10
PositivaNegativaNeutraXY	0,0 %	0	6	6
caminoBasicoBisiesto	75,0 %	3	1	4
Bisiesto.java	50,0 %	1	1	2
Bisiesto	50,0 %	1	1	2
mostrarSiBisiesto(int)	100,0 %	1	0	1

Vista de la complejidad cubierta:

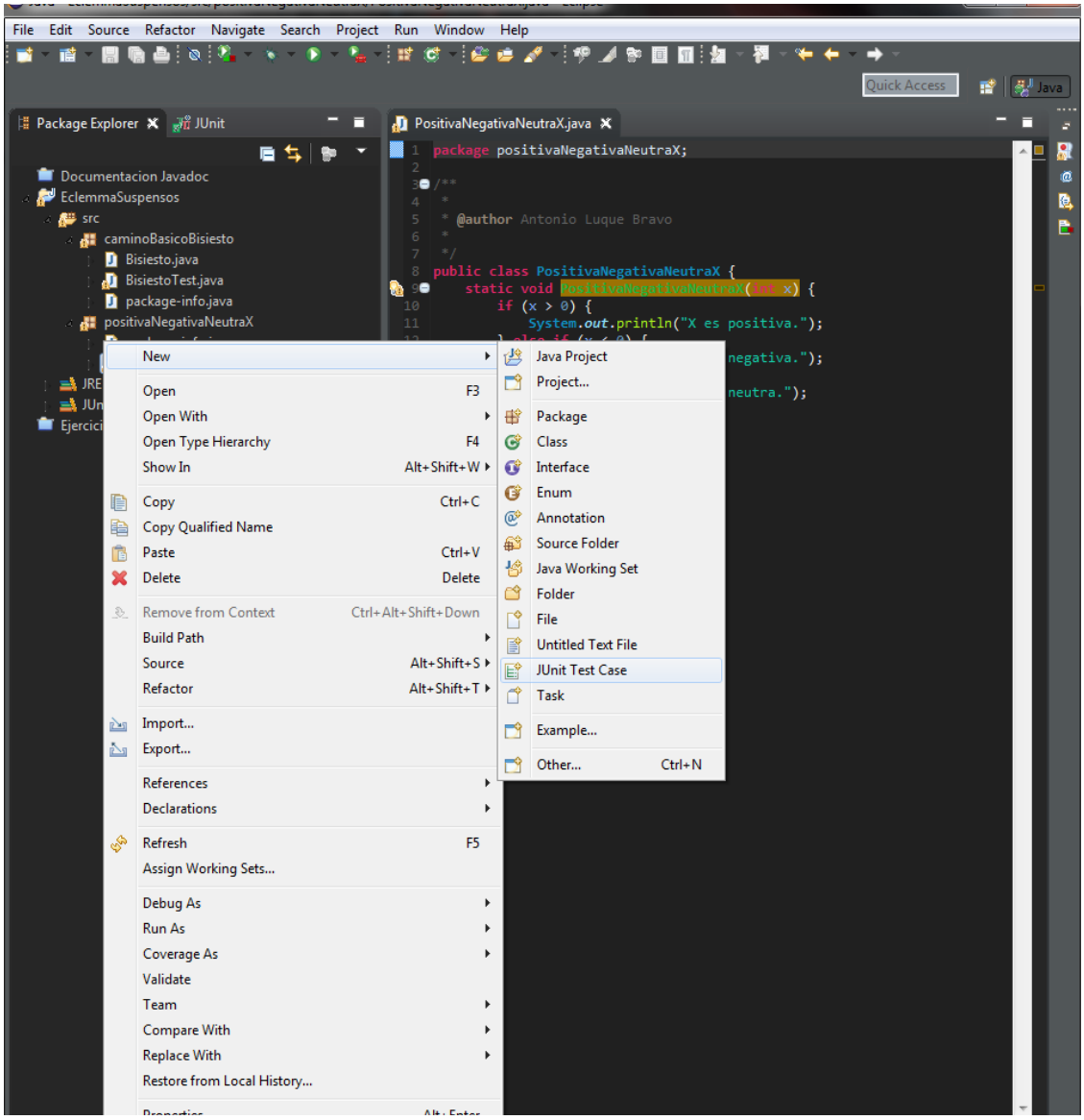
Element	Coverage	Covered Complexity	Missed Complexity	Total Complexity
EclemmaSuspensos	31,6 %	6	13	19
src	31,6 %	6	13	19
PositivaNegativaNeutraXY	0,0 %	0	12	12
caminoBasicoBisiesto	85,7 %	6	1	7
Bisiesto.java	80,0 %	4	1	5
Bisiesto	80,0 %	4	1	5
mostrarSiBisiesto(int)	100,0 %	4	0	4

Y su grafo es el siguiente:

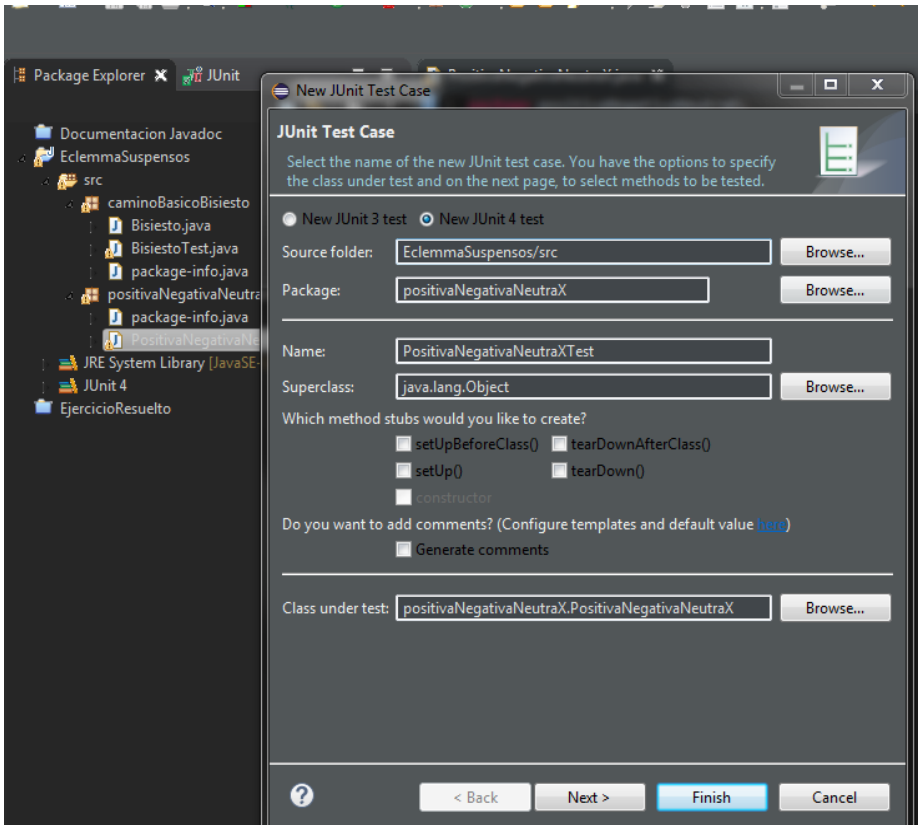


1.2 Código de PositivaNegativaNeutraX

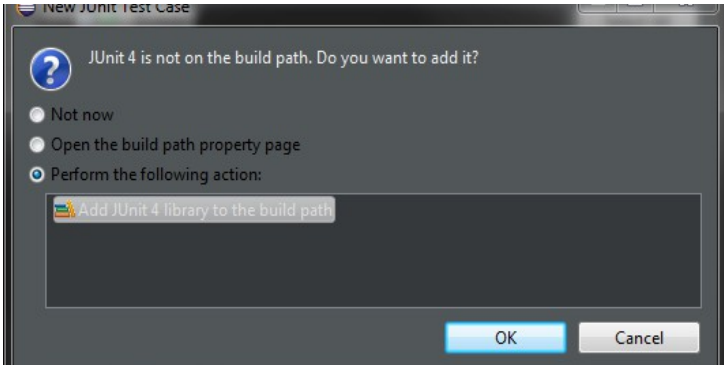
Para poder generar las pruebas, debemos irnos a nuestro código en Eclipse. Una vez allí damos clic derecho en la clase que queremos hacer las pruebas y creamos un JUnit Test Case.



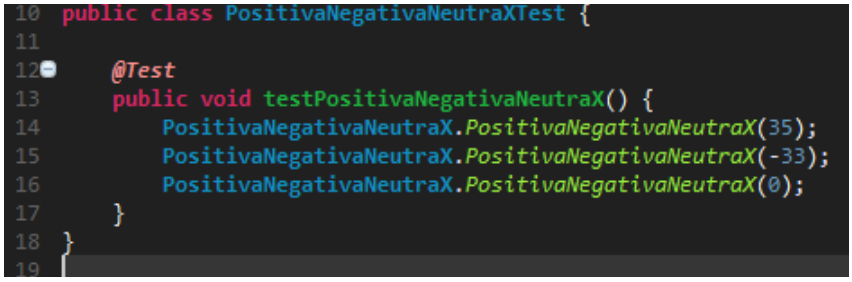
Acto seguido configuramos el JUnit Test Case tal y como se muestra en la siguiente imagen:



Cuando terminemos de configurarlo le damos a Finalizar, y en esa clase empezaremos a hacer nuestras correspondientes pruebas, seguidamente nos pedirá que añadamos la librería de JUnit 4, le decimos que sí y ya se creara la clase donde hacemos las pruebas de nuestro código.

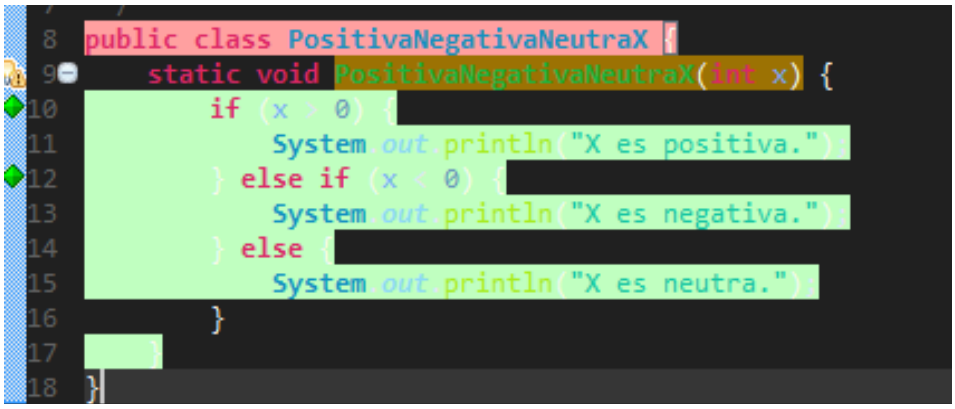


Debemos de hacer las pruebas dependiendo de la complejidad ciclomática de la clase (en este caso es 3):



```
10 public class PositivaNegativaNeutraXTest {
11
12     @Test
13     public void testPositivaNegativaNeutraX() {
14         PositivaNegativaNeutraX.PositivaNegativaNeutraX(35);
15         PositivaNegativaNeutraX.PositivaNegativaNeutraX(-33);
16         PositivaNegativaNeutraX.PositivaNegativaNeutraX(0);
17     }
18 }
```

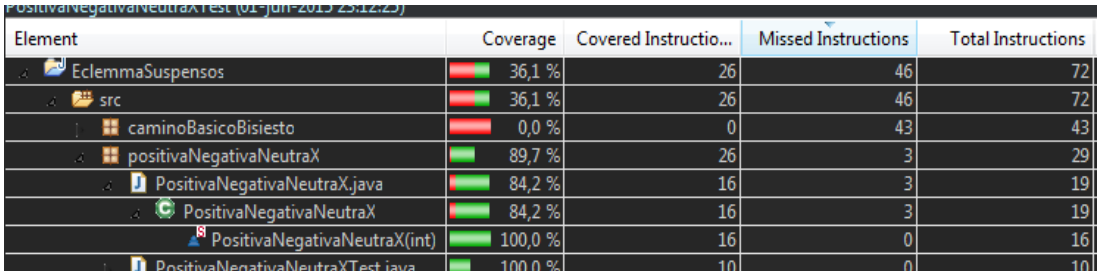
Cuando nos salga todo el método que estamos probando en color verde, hemos hecho bien nuestras pruebas, hemos cubierto todas las posibilidades de nuestra clase.



```
8  public class PositivaNegativaNeutraX {
9      static void PositivaNegativaNeutraX(int x) {
10         if (x > 0) {
11             System.out.println("X es positiva.");
12         } else if (x < 0) {
13             System.out.println("X es negativa.");
14         } else {
15             System.out.println("X es neutra.");
16         }
17     }
18 }
```

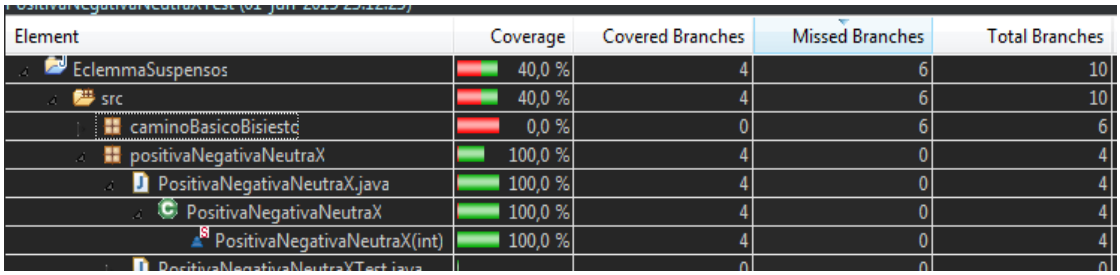
Tenemos que cubrir todas las ramas hasta que en el Coverage nos salga 100% en el método que estamos probando, he aquí los distintos tipos de coberturas:

Vista de las instrucciones cubiertas:



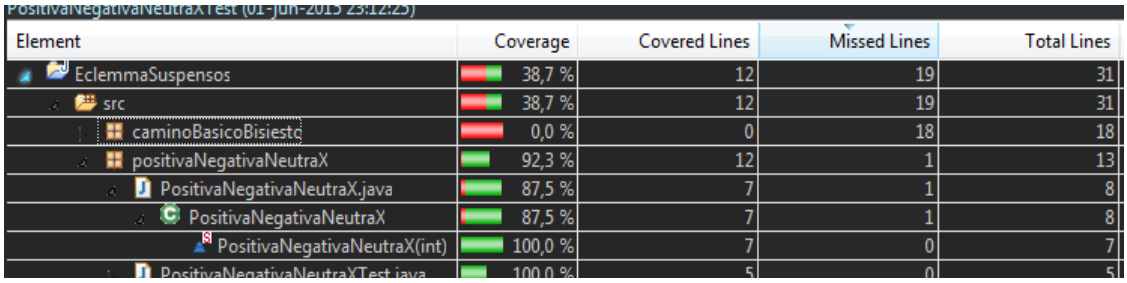
Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
EclemmaSuspensos	36,1 %	26	46	72
src	36,1 %	26	46	72
caminoBasicoBisiesto	0,0 %	0	43	43
positivaNegativaNeutraX	89,7 %	26	3	29
PositivaNegativaNeutraX.java	84,2 %	16	3	19
PositivaNegativaNeutraX	84,2 %	16	3	19
PositivaNegativaNeutraX(int)	100,0 %	16	0	16

Vista de las ramas cubiertas:



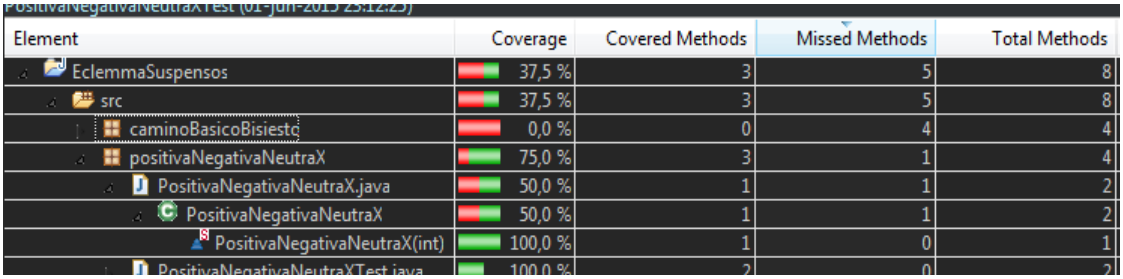
Element	Coverage	Covered Branches	Missed Branches	Total Branches
EclemmaSuspensos	40,0 %	4	6	10
src	40,0 %	4	6	10
caminoBasicoBisiesto	0,0 %	0	6	6
positivaNegativaNeutraX	100,0 %	4	0	4
PositivaNegativaNeutraX.java	100,0 %	4	0	4
PositivaNegativaNeutraX	100,0 %	4	0	4
PositivaNegativaNeutraX(int)	100,0 %	4	0	4

Vista de las líneas de código cubiertas:



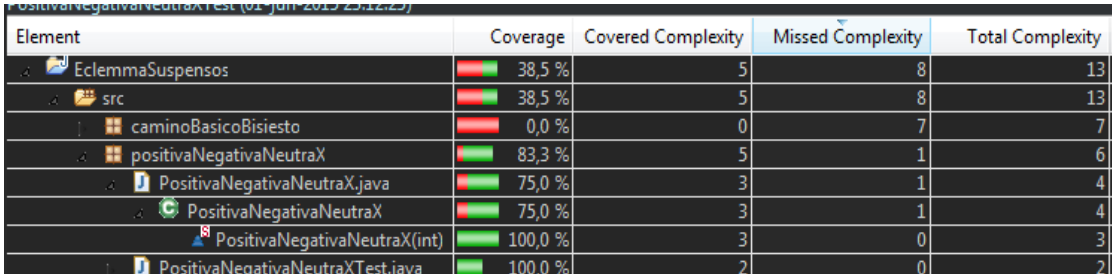
Element	Coverage	Covered Lines	Missed Lines	Total Lines
EclemmaSuspensos	38,7 %	12	19	31
src	38,7 %	12	19	31
caminoBasicoBisiesto	0,0 %	0	18	18
positivaNegativaNeutraX	92,3 %	12	1	13
PositivaNegativaNeutraX.java	87,5 %	7	1	8
PositivaNegativaNeutraX	87,5 %	7	1	8
PositivaNegativaNeutraX(int)	100,0 %	7	0	7

Vista de los métodos cubiertos:



Element	Coverage	Covered Methods	Missed Methods	Total Methods
EclemmaSuspensos	37,5 %	3	5	8
src	37,5 %	3	5	8
caminoBasicoBisiesto	0,0 %	0	4	4
positivaNegativaNeutraX	75,0 %	3	1	4
PositivaNegativaNeutraX.java	50,0 %	1	1	2
PositivaNegativaNeutraX	50,0 %	1	1	2
PositivaNegativaNeutraX(int)	100,0 %	1	0	1

Vista de la complejidad cubierta:



Element	Coverage	Covered Complexity	Missed Complexity	Total Complexity
EclemmaSuspensos	38,5 %	5	8	13
src	38,5 %	5	8	13
caminoBasicoBisiesto	0,0 %	0	7	7
positivaNegativaNeutraX	83,3 %	5	1	6
PositivaNegativaNeutraX.java	75,0 %	3	1	4
PositivaNegativaNeutraX	75,0 %	3	1	4
PositivaNegativaNeutraX(int)	100,0 %	3	0	3

Y aquí el grafo perteneciente al código: