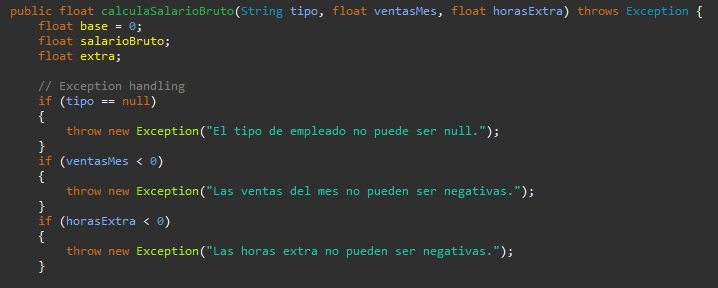
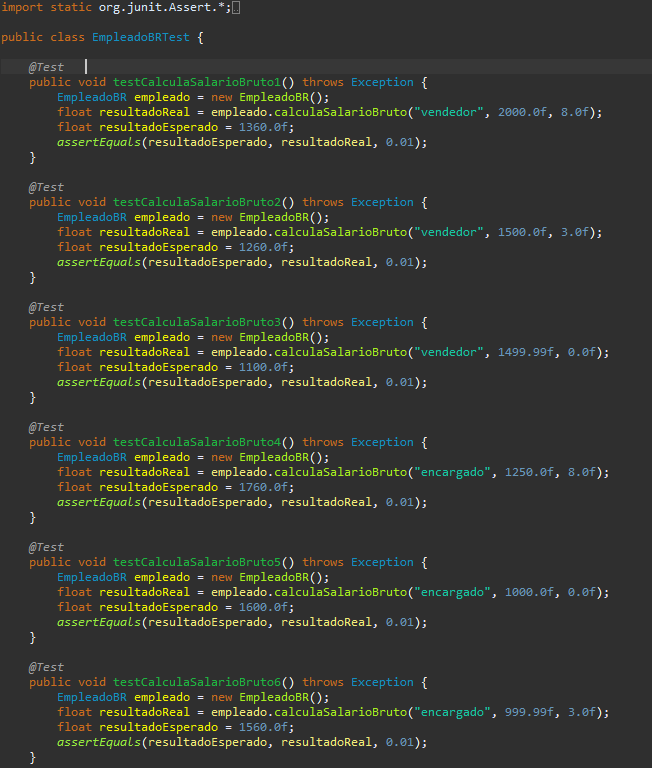
Práctica 2 de JUnit

**Ejercicio 1: Realiza con Eclipse el resto de pruebas y comenta los resultados.**

Para realizar las pruebas de calcularSalarioBruto es necesario añadir el control de excepciones a dicha función en la clase EmpleadoBR.



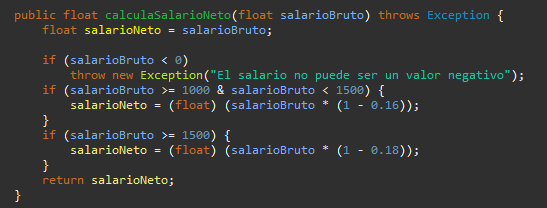
Además, se deberá añadir “throws Exception” al final de la declaración de la función. Al añadirlo en EmpleadoBR, también deberá añadirse a todas las funciones “testCalculaSalarioBrutoX” esa misma línea de “throws Exception”, para evitar errores.



Todo esto es para poder usar “@Test(expected=Exception.class)” en los tests de calculaSalarioBruto, que se necesitará usar en los últimos tres casos de prueba de testCalcularSalarioBrutoX.

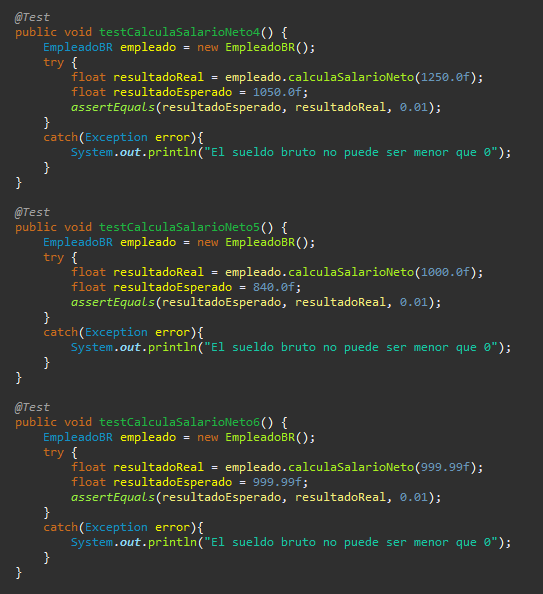


En el caso de textCalcularSalarioNeto9 (es decir, el último), lanza una excepción, y para poder detectarla, hay que añadir “throws Exception” al declarar la función y además el decorativo “@Test(exception=Exception.class)”.



Tras realizar estos ajustes los casos de prueba se ejecutarán sin problemas.

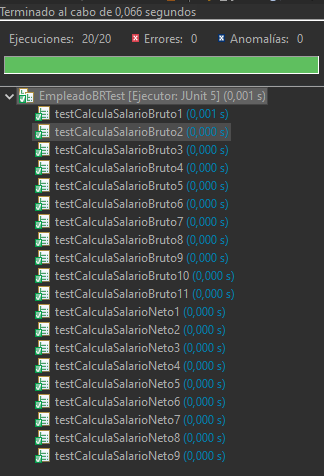






Test 9, que espera una excepción.

Resultados:



**Ejercicio 2: Realiza los test mediante pruebas parametrizadas.**

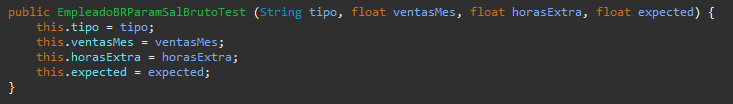
Se deberán realizar dos clases de tests parametrizados, una para CalcularSalarioBruto y otra para CalcularSalarioNeto.

En el caso del cálculo del salario bruto, se deben usar parámetros que tengan las siguientes características: una cadena para el tipo, un float para las ventas del mes, y otro float para las horas extra. Además, se debe añadir otro dato, un float, para el resultado esperado. Por desgracia, desconozco como probar las excepciones con pruebas parametrizadas, así que se omitirán en las dos clases de tests parametrizados.

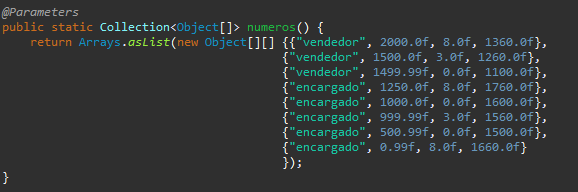
EmpleadoBR: Tests parametrizados del cálculo del salario bruto:

Empezamos con: *@RunWith*(Parameterized.class)

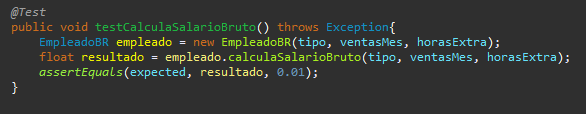
La clase necesita un constructor:



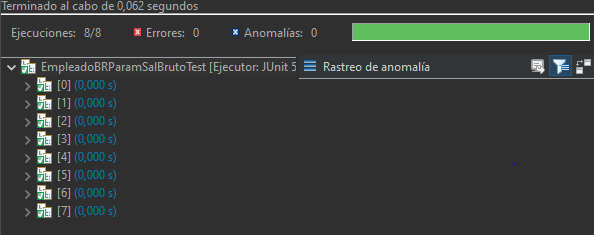
Luego, debemos inicializar el conjunto de parámetros:



Y, por último, implementamos una función genérica para comprobar los resultados y los parámetros:



Resultados:

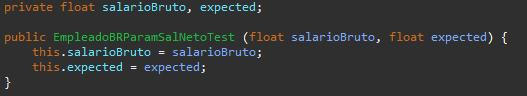


En el caso del cálculo del salario neto, se deben usar parámetros que tengan las siguientes características: un float para el salario bruto, y un float para el resultado esperado. Por desgracia, desconozco como probar las excepciones con pruebas parametrizadas, así que se omitirán en las dos clases de tests parametrizados.

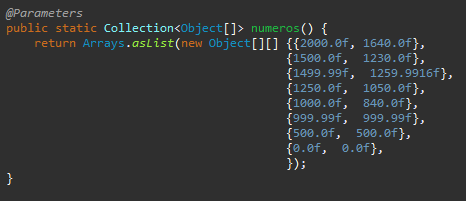
EmpleadoBR: Tests parametrizados del cálculo del salario bruto:

Empezamos con: *@RunWith*(Parameterized.class)

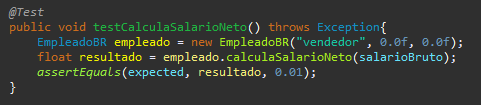
La clase necesita un constructor, como el anterior test:



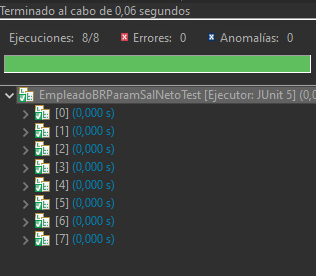
Inicialización del conjunto de parámetros:



Creamos una función test genérica para probar los parámetros:

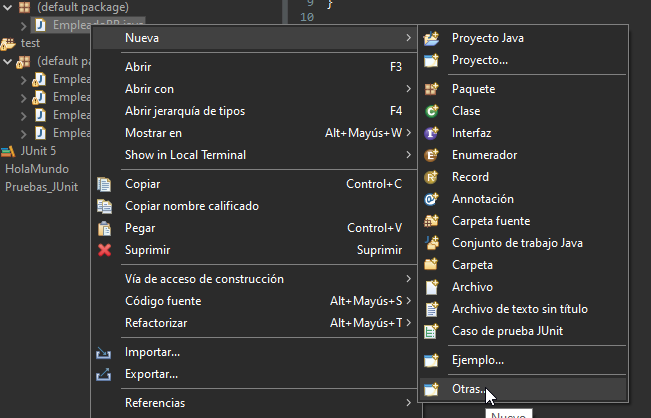


Resultados:



**Ejercicio 3: Realiza con Eclipse una suite de pruebas.**

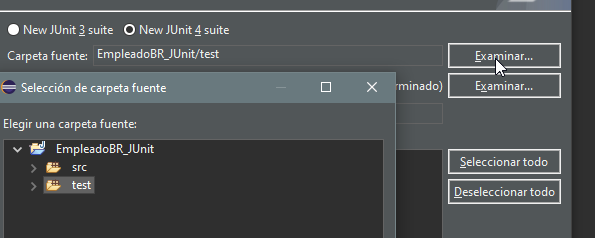
Primero tenemos que crear la suite:



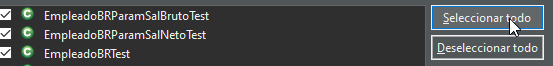
Clic derecho > Nueva > Otras.



Buscamos suite de pruebas JUnit y seleccionamos siguiente.

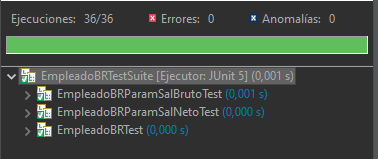


Seleccionamos examinar y escogemos la carpeta en la que tenemos todos los tests.



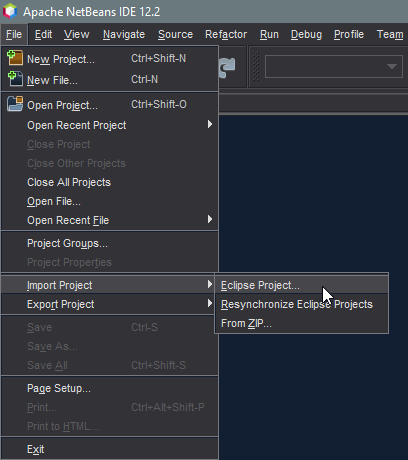
Seleccionamos las clases para incluirlas en la suite.

Ejecutemos la suite para comprobar que todo funciona:

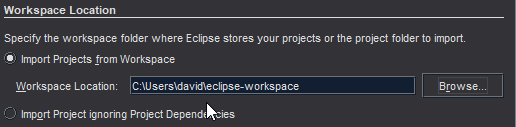


**Ejercicio 4: Realiza lo mismo para el IDE Netbeans.**

Desde Netbeans se puede importar un proyecto de Eclipse, así que importamos el proyecto que hemos realizado en Eclipse a Netbeans.

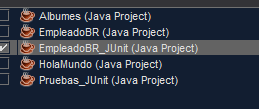


Tendremos que dar la carpeta del espacio de trabajo de Eclipse.

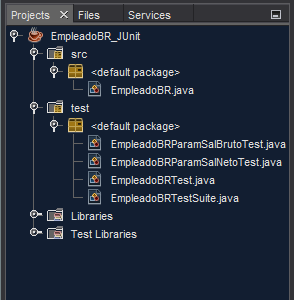


Seleccionamos siguiente, y seleccionamos los proyectos a importar.

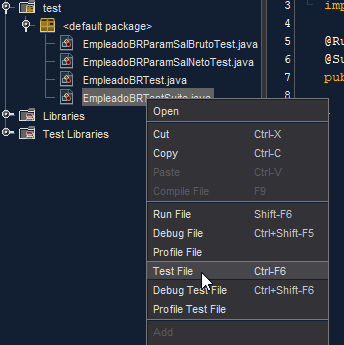
ç



Una vez importado ya tenemos nuestras clases.



En Netbeans, para ejecutar los clasos de prueba, debemos ir a los tests, seleccionarlos con clic derecho, y seleccionar *Test File*.



Obtendremos los mismos resultados:

