CICLO: [DAM]
MÓDULO DE [ENTORNOS DE DESARROLLO]

[Tarea Nº 05]

Alumno:
[Juan Carlos Filter Martín]
[redacted]

Contenido

1. Documentos que se adjuntan a este informe.

A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Proyecto

2. Prueba de la siguiente función llamada calificaciones

Código de la tarea función calificaciones

```
public float calificaciones(float examenTeorico, float examenPracticas, boolean
practicaConvalidada){

     if(examenTeorico > 7F){
          System.out.println("Error rango Examen Teorico");
          return -1;
     }else if(examenPracticas > 3F || examenPracticas < 0F){
          System.out.println("Error rango Examen Practico");
          return -1;
     }else {
          if(examenTeorico <= 3.5F){
               if(practicaConvalidada){
                    return examenTeorico+1.5F;
               }else{
                    return examenTeorico+examenPracticas;
               }
          }else{
               return examenTeorico;
          }
     }
```

3. (RA03_f) Se han efectuado pruebas unitarias de clases y funciones.

Pruebas de caja negra

ExamenTeorico: Campo float entre 0 y 7, ambos inclusive.
ExamenPracticas: Campo float entre 0 y 3, ambos inclusive.
PracticaConvalidada: Campo booleano que puede ser true o false.

Condición de entrada	Tipo	Clase Equivalencia Válida	Clase Equivalencia No Válida
ExámenTeorico	Rango	1: 0 <= examenTeorico <= 7	2: examenTeorico < 0 3: examenTeorico > 7
ExámenPracticas	Rango	4: 0 <= examenPracticas <= 3	5: examenPracticas < 0 6: examenPracticas > 3
PrácticasConvalidada	Booleano	7: practicasConvalidada = True 8: practicasConvalidada = False	

4. (RA03_b) Se han definido casos de prueba.

Generar los casos de prueba

Generar los **casos de prueba** para las clases creadas mediante particiones de equivalencia, indicando en cada caso las clases que cubre.

- La función devolverá para entradas correctas un float entre 0 y 10, ambos inclusive.

- La función devolverá para entradas incorrectas un float -1 y mostrará un mensaje de error por consola.

Controlados en el caso de prueba nº 1 y 2

Si la nota del "examenTeorico" es un 3.5 o más, habrá que ver si la "practicaConvalidada" está convalidada (true) o no (false):

- **(1)** Si está convalidada (true) se le suma 1.5 al examenTeorico.
- **(2)** Si no está convalidada (false) se le suma la nota examenPracticas al examenTeorico.

(3) Si la nota del examenTeorico es menor de un 3.5, su nota final será solo la del examenTeorico.

			Entrada		Salida
Nº	Clase de Equivalencia	Exámen Teórico	Exámen Prácticas	Prácticas Convalidada	
(1) **1**	1,4,7	3.5	2	True	5
(2) **2**	1,4,8	3.5	2	False	5.5
(3) **3**	1,4,7	2.5	0.5	True	2.5
4	2,4,8	-2	1	False	-1 ExamenTeórico
5	3,4,7	8	1	True	-1 ExamenTeórico
6	1,5,8	1	-2	False	-1 ExamenPráctico
7	1,6,7	1	4	True	-1 ExamenPráctico

✗ **No podría devolver >10 ya que el exámenTeórico solo puede tener valor máximo 7 y el practico sería 3 dando como resultado un 10 de nota máxima**

✗ **Si da de entradas incorrecta va a mostrar -1 entrando en el if controlado con la condición y devolviéndolo con return -1**

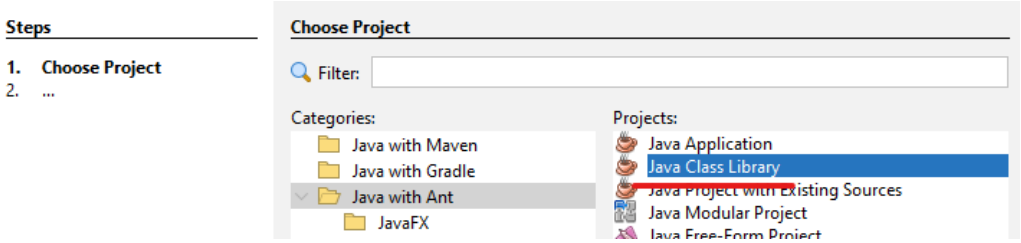
5. (RA03_g) Se han implementado pruebas automáticas.

Añadir una clase JUnit

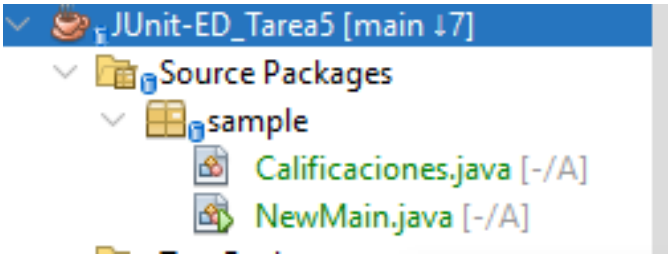
Añadir una clase JUnit con la función calificaciones a un proyecto Netbeans para ejecutar los casos de prueba del apartado anterior.

Se va a mostrar y explicar paso a paso como crear una clase JUnit:

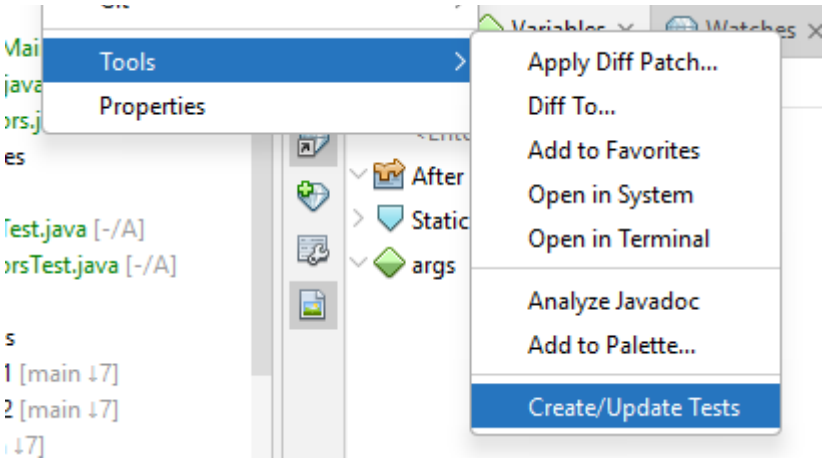
a) Primero crearemos un proyecto Java Class Libray



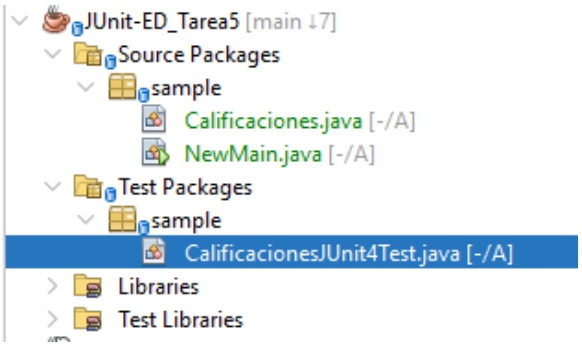
b) Ahora creamos la clase donde va a estar la función calificaciones



c) Posteriormente pulsando con botón derecho en la clase Calificaciones donde se encuentra la función vamos a crear un Test **"Tools > Create/UpdateTests"**



d) Ya tendríamos la clase JUnit donde se podrán crear el test o los distintos test que necesitemos usar fuera el caso de que hubiera más de un método.



Crear un test con función calificaciones para realizar las pruebas

e) Por ultimo tendríamos que crear el test con las diferentes pruebas que se realizarán recogidas de las pruebas de caja negra realizada anteriormente.

```
@Test
public void testCalificaciones() {
    System.out.println("* JUnit4Test: testCalificaciones()");
    assertEquals(5, Calificaciones.calificaciones(3.5f, 2, true), 0.1);
    assertEquals(5.5, Calificaciones.calificaciones(3.5f, 2, false), 0.1);
    assertEquals(2.5, Calificaciones.calificaciones(2.5f, 0.5f, true), 0.1);
    assertEquals(-1, Calificaciones.calificaciones(-2, 1, false), 0.1);
    assertEquals(-1, Calificaciones.calificaciones(8, 1, true), 0.1);
    assertEquals(-1, Calificaciones.calificaciones(1, -2, false), 0.1);
    assertEquals(-1, Calificaciones.calificaciones(1, 4, true), 0.1);
```

✗ Se va a explicar lo significa con el siguiente ejemplo:

```
@Test
public void testCalificaciones() {
    System.out.println("* JUnit4Test: testCalificaciones()");
    assertEquals(5, Calificaciones.calificaciones(3.5f, 2, true), 0.1);
    assertEquals(5.5, Calificaciones.calificaciones(3.5f, 2, false), 0.1)
```

■ **@Test** : Es necesario para que este realice las pruebas
Si este fuera indicado con @ignore se daría el caso de que el test sería ignorado.

■ **AsserEquals**: es un método de pruebas para verificar que el valor sea igual al esperado.

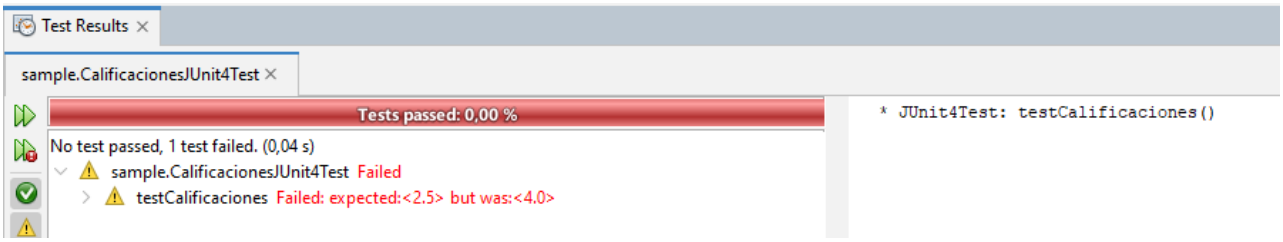
■ **5.5f**: es el valor esperado

■ **Calificaciones.calificaciones**: Es la clase y el método donde se va a realizar la prueba seguido de los parámetros que se piden en ese método.

■ 0.1: Se utiliza para cuando se espera un resultado de punto flotante y calcular con precisión el resultado. (el calculo solamente será realizado hasta 1 decimal)

1º Intento de Test de prueba JUnit:

Como se puede observar el test de prueba da error, significa que la función está mal realizada

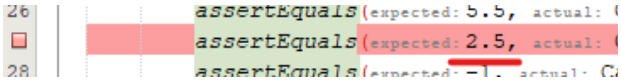


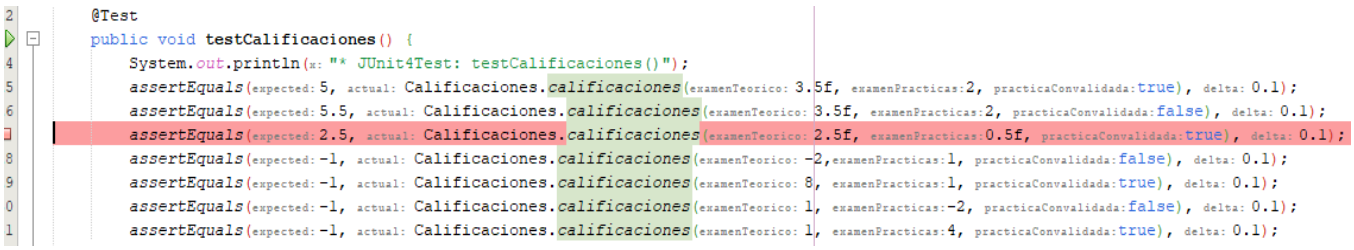
Este error lo que está diciendo que ha fallado la prueba que se esperaba 2.5 dando como resultado 4.0 (Error en la tercera prueba)

6. (RA03_e) Se han utilizado las herramientas de depuración para examinar y modificar el comportamiento de un programa en tiempo de ejecución.

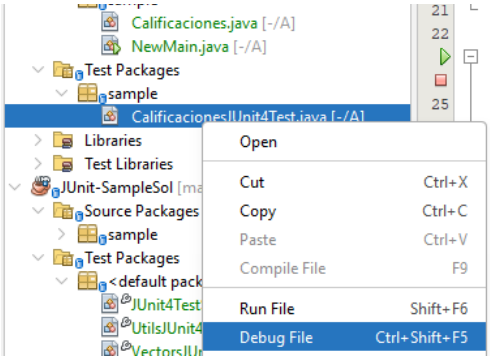
Herramienta de depuración

Vamos a **usar el depurador** usando un **breakpoint** en la tercera prueba para encontrar los errores posibles:

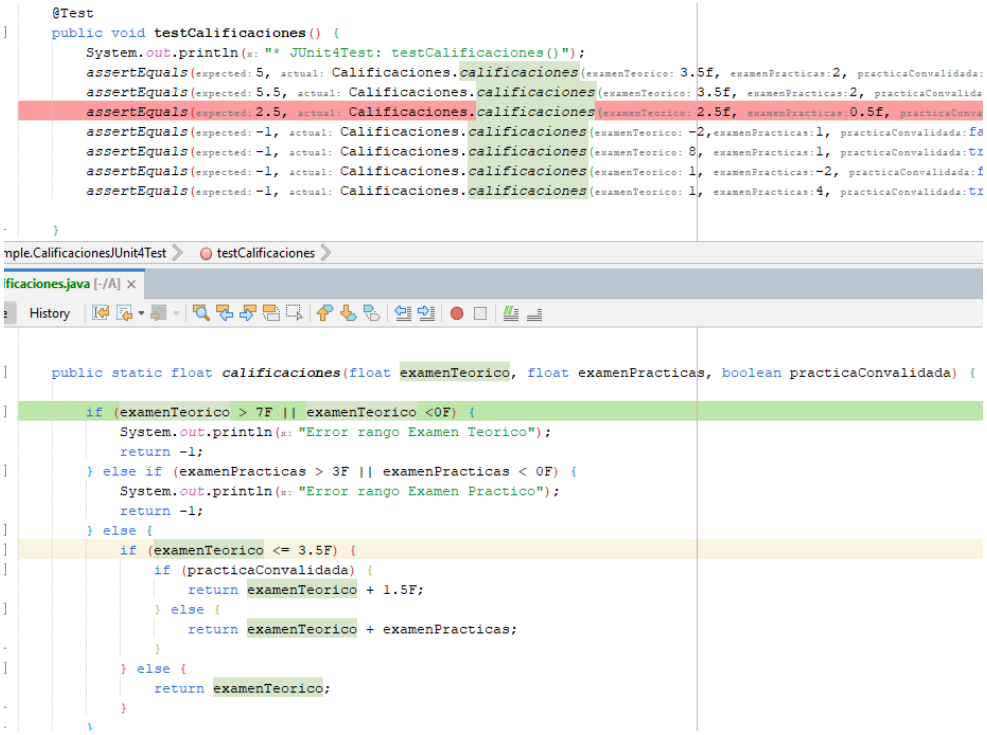




Ejecutamos la clase JUnit con el depurador

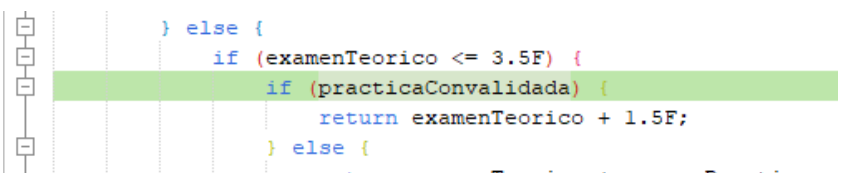


Al ejecutar el depurador y entramos al método calificaciones mediante Step into podemos hacer un seguimiento paso a paso de su ejecución



Error nº1

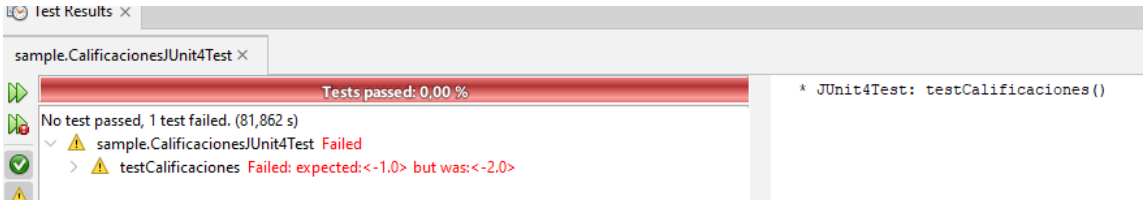
Como se puede observar **entra en el if (examenTeorico <= 3.5F)** cuando debería ir directamente al else



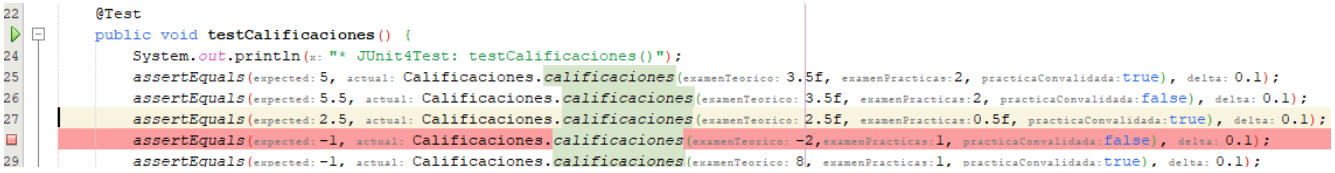
ERROR1: Tendríamos que cambiar el if :
if (examenTeorico >= 3.5F)

```
if (examenTeorico > 7F) {
    System.out.println("Error rango Examen Teorico");
    return -1;
} else if (examenPracticas > 3F || examenPracticas < 0F) {
    System.out.println("Error rango Examen Practico");
    return -1;
} else {
    if (examenTeorico >= 3.5F) {
        if (practicaConvalidada) {
            return examenTeorico + 1.5F;
        } else {
            return examenTeorico + examenPracticas;
```

Volvemos a pasar el depurador con el mismo breakpoint y comprobamos que ahora no da error, saltando a la siguiente prueba pero en esta prueba vuelve indicar otro error diferente.

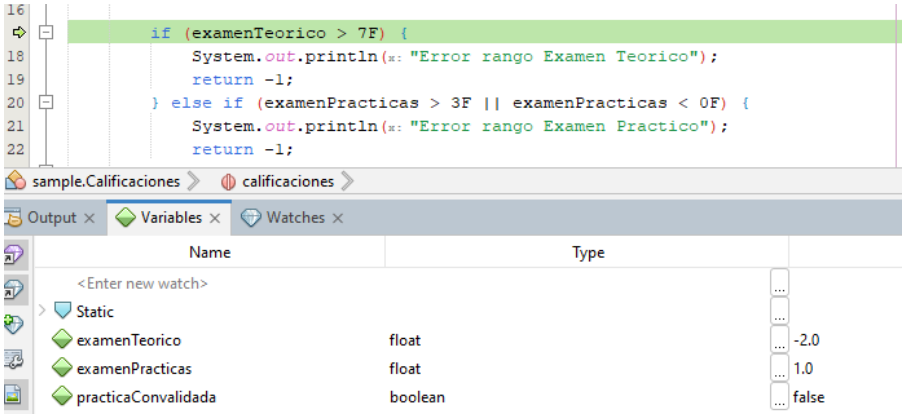


Entonces vamos a realizar indicar con un punto de ruptura para comprobar donde ocurre el fallo.



Error nº2

Ejecutamos el debug entramos al método calificaciones y podemos comprobar que estamos dando un dato “-2.0” en examenTeorico entonces este nos debería devolver -1 entrando en el primer if



ERROR2Tendríamos que cambiar el if :
if(examenTeorico > 7F || examenTeorico <0F)

```
public static float calificaciones(float examenTeorico, float exa

    if (examenTeorico > 7F || examenTeorico <0F) {
        System.out.println(x: "Error rango Examen Teorico");
        return -1;
    } else if (examenPracticas > 3F || examenPracticas < 0F) {
```

2º Intento de Test de prueba JUnit:

Una vez terminada la depuración y solucionado el error, volvemos a ejecutar las pruebas con JUnit para ver que ahora dan los resultados esperados.

→ Se puede comprobar que no hay errores de Tests