**AE-3. Pruebas de software**  **con JUnit**

**Grupo 10:**

Lidia Díaz Mendoza

Sergio Martínez Rivera

Carlos Rábago Torcates

**URL del repositorio:** <https://github.com/Cambixx/WS_Actividad_ED.git>

**ÍNDICE**

1. Resta (realizado por Sergio Martínez Rivera )

2. Suma (realizado por Carlos Rábago Torcates)

3. Producto (realizado por Lidia Díaz Mendoza )

4. Cociente (realizado por Sergio Martínez Rivera )

**1. RESTA**

* He realizado las pruebas de los métodos de la clase resta, para ello lo primero que he tenido que hacer es crear una nueva carpeta que he llamado test donde van a ir los JUnit test case que utilizaremos para realizar el ejercicio.
* He creado el test case y lo he llamado testResta.
* Le he creado a la clase resta un constructor vacío y le he añadido 4 propiedades de tipo real y 2 enteras.
* Para poder trabajar desde el test case con un objeto resta, y utilizar sus getter y setter he tenido que importarme el paquete donde se encuentra dicha clase. Como hasta ahora habíamos trabajado con el paquete por defecto he tenido que crear un nuevo paquete al que he nombrado como principal y he movido alli todas las clases. Una vez realizado lo anterior ya he podido hacer el import principal.resta; y ponerme a trabajar en las pruebas.
* Me he declarado un objeto resta que he llamado r1 para trabajar con todas las pruebas.
* He creado un método llamado limpiar() que es @BeforeEach y se va a encargar de crear vacío el objeto r1 siempre antes de realizar cada @test.
* El primer método con @test será el de la resta de dos números reales. Para hacer la prueba he utilizado los set y así he introducido los dos valores. He creado una variable ResultadoEsperado y le he puesto el valor que debe darme la operación y la variable ResultadoObtenido va a tener el valor resultante de la resta de la llamada a los get de los atributos que previamente había dado valor con los set.
* Compruebo que el método hace bien su función comparando el resultadoEsperado con el resultadoObtenido con AssertEquals.
* Con los dos siguientes métodos hago prácticamente lo mismo, pero trabajando con enteros en vez de con reales.
* Para el método de acumulado he llamado con el objeto al propio método y le he introducido varios valores.
* He creado de nuevo la variable ResultadoEsperado a la que he dado el valor resultante de sumar los valores que he ido metiendo al método. Y por otro lado la variable ResultadoObtenido tiene el valor de Acumulado, es decir de getAcumulado(). Con AssertEquals de nuevo comparo el ResultadoObtenido con el ResultadoEsperado.

\*Al trabajar Carlos y yo de manera simultanea nos hemos dado cuenta antes de subir mi parte a github que el también había creado la carpeta para trabajar los test y habia movido las clases a un nuevo paquete que no era el de por defecto y por tanto el proyecto me salía con conflictos. Nos hemos puesto de acuerdo y he borrado y trabajado a partir de sus modificaciones. Una vez solucionado lo he subido al repositorio local.

**2. SUMA**

**3. PRODUCTO**

**4. COCIENTE**

* Para poder trabajar con el cociente he realizado lo mismo que con la clase Resta al principio. Es decir, he creado el constructor y los atributos de la clase Cociente.
* He creado un test case de Cociente y en él me he importado el paquete con import principal.Cociente;
* También me he creado un objeto tipo Cociente para poder hacer las pruebas, y he creado un método limpiar con la misma función de crear un objeto nuevo vacío de Cociente antes de cada @test con el @BeforeEach.
* Los @test de los métodos de dividir enteros y reales no difieren demasiado de los que he realizado con los métodos de Resta. Son en esencia lo mismo, pero en vez de darle a la variable ResultadoObtenido el valor resultante de una resta lo será de una división.
* Al método de número inverso, además de comprobar que el Resultadoesperado sea igual que el obtenido elevando el numero introducido por –1, le hago otra prueba que consiste en preguntar a través de una sentencia if si el resultado de multiplicar el número introducido por el ResultadoObtenido da igual a la unidad.
* Con el método de la raiz cuadrada, introduzco un valor con el set y compruebo si la raiz cuadrada de ese número indroducido es igual al ResultadoEsperado. Por otro lado, compruebo si realiza bien la operación multiplicando el resultado obtenido por sí mismo y preguntando si es igual al número introducido previamente.