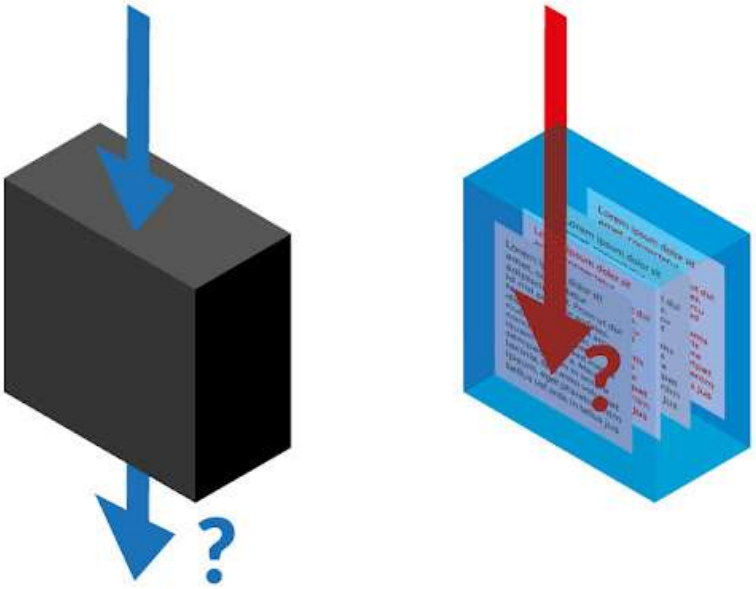
Alfonso Rincón Cuerva

EJERCICIOS UT3



ÍNDICE

ACTIVIDAD 3.1

Vamos a realizar pruebas de caja negra, ya que estas pruebas la estamos realizando en la propia interfaz del programa, a pesar de haberlo creado nosotros. En este caso, hemos realizado pruebas unitarias, las cuales están enfocadas en una parte pequeña del código, como objetos y métodos.

PRUEBA	NUMERO 1	NUMERO 2	SALIDA
Número mayor entre número menor	20	6	3,3333
Número menor entre número mayor	6	20	0,3000
0 entre número positivo	0	3	0,0000
0 entre número negativo	0	-3	-0,0000
Número positivo entre 0	3	0	Infinity
Número negativo entre 0	-3	0	-Infinty
0 entre 0	0	0	NaN
Negativo entre positivo	-5	6	-0,8333
Positivo entre negativo	5	-6	-0,8333
Negativo entre negativo	-5	-6	0,8333
Decimal entre número entero	10,5	2	ERROR
Entero entre número decimal	2	10,5	ERROR
Números de +9 ceros	10000000000	2	ERROR
Decimal negativo entre entero	-10,5	2	ERROR
Decimal entre decimal	1,2	2,2	ERROR
Mismo número	100	100	1,0000
Letras	N	3	ERROR
Otros caracteres	@	2	ERROR

ACTIVIDAD 3.2

Hemos creado una aplicación en Java, a la cual le realizamos pruebas de caja blanca y pruebas de caja negra.

PRUEBAS DE CAJA BLANCA: en estas tenemos el acceso al código, y en este caso, a los métodos **calculaSalarioNeto** y **calculaSalarioBruto**.

- **Pruebas de cubrimiento:** se han realizado pruebas para todos posibles casos. Se realizaron pruebas para el tipo de empleado, al haber probado el funcionamiento tanto del tipo "vendedor" como de "encargado". También se realizaron pruebas para verificar el correcto funcionamiento de la condición de las ventas mensuales, al haber probado casos mayores de 1500, casos entre 1000 y 1500, y menores de 1000.
  En el caso del método calculaSalarioNeto, también se hicieron pruebas de todos los casos.
- **Pruebas de condiciones:** se han comprobado que todas las condiciones funcionan, además de haber realizado diferentes casos de prueba para cubrir todos los posibles casos.
- **Pruebas de bucles:** el código no dispone de bucles.

PRUEBAS DE CAJA NEGRA: estas se encargan de probar la interfaz sin tener en cuenta el código. Tenemos:

- **Pruebas de clases de equivalencia de datos:** se realizaron algunas pruebas para el tipo de empleado en el método calculaSalarioBruto.

calculaSalarioBruto		
Vendedor, 2000, 8	1360	360
vendedor , 1500, 3	1260	260
Encargado, 1499.99, 0	1100	100
encargado , 1250, 8	1760	260
encrgado, 1000, 0	1600	100

- **Prueba de interfaces:** el código no dispone de interfaz, por tanto, no se podrán realizar estas pruebas.

- **Pruebas de valor límite:** en la siguiente tabla tendremos las pruebas que realizamos.

calculaSalarioNeto		
ENTRADA	**SALIDA ESPERADA**	**SALIDA OBTENIDA**
2000	1640	1640
1500	1230	1500
1499.99	1259.9916	1259.9916
1250	1050	1050
1000	840	840
999.99	999.99	999.99
500	500	500
0	0	0
calculaSalarioBruto		
vendedor, 2000, 8	1360	1360
vendedor, 1500, 3	1260	1260
vendedor, 1499.99, 0	1100	1100
encargado, 1250, 8	1760	1760
encargado, 1000, 0	1600	1600
encargado, 999.99, 3	1560	1560
encargado, 500, 0	1500	1500
encargado, 0, 8	1660	1660

ACTIVIDAD 3.3

CÓDIGO DEL BANCO

Clase válida: números de 3 dígitos, o en blanco. El primero mayor que 1.

Caso de prueba válido: 243, 344, “ “, 544, 657, 233, 555

Clase no válida: números de más de 3 dígitos o que empiecen por 1 o menor de 1.

Caso de prueba no válido: 3456, 122, 093, 5945, 23, 1, 0444

CÓDIGO DE SUCURSAL

Clase válida: número de 4 dígitos con el primer número mayor que 0.

Caso de prueba válido: 3113, 2345, 6546, 3655, 8887

Clase no válida: número de más o menos de 4 dígitos con el primer número mayor que 0.

Caso de prueba no válido: 256, 0234, 004, 12356, 44, 5

NÚMERO DE CUENTA

Clase válida: número de 5 dígitos.

Caso de prueba válido: 67898, 12345, 87657, 99883, 23425

Clase no válida: número de menos o más de 5 dígitos.

Caso de prueba no válido: 542346, 4565, 324, 00, 6546446

CLAVE PERSONAL

Clase válida: valor alfanumérico de 5 posiciones

Caso de prueba válido: AX7ul, 34567, Uiol9, aZZik, UIAKZ

Clase no válida: valor alfanumérico de más o de menos de 5 posiciones

Caso de prueba no válido: Ai99, 9982, auid, AJ6hTT, id987jj

ORDEN

Clase válida: en blanco, o con los valores “Talonario” o “Movimientos”

Caso de prueba válido: “ “, “Talonario”, “Movimientos”

Clase no válida: valor mal escrito

Caso de prueba: “Talonaro”, “Hola”, “oIHKmm”, “Movimiento”

ACTIVIDAD 3.4

INTEGRACIÓN INCREMENTAL ASCENDENTE:

Características clave:

- Comienza con la integración y prueba de los módulos de nivel más bajo.
- Los módulos se combinan en grupos más grandes a medida que se avanza en el proceso de integración.
- Se enfoca en la construcción de la funcionalidad básica del sistema antes de agregar componentes más complejos.

Ventajas:

- Permite identificar y corregir errores en los módulos de nivel más bajo antes de integrar componentes más complejos.
- Facilita la detección de problemas y la corrección de errores en las primeras etapas del desarrollo
- Proporciona una base sólida para la integración de componentes más complejos.

Desventajas:

- Puede requerir más tiempo y esfuerzo para completar la integración de todos los componentes.
- Puede ser difícil de implementar si los módulos de nivel más bajo no están disponibles o no están completamente desarrollados.

INTEGRACIÓN INCREMENTAL DESCENDENTE:

Características clave:

- Comienza con la integración y prueba de los módulos de nivel más alto.
- Los módulos se combinan gradualmente en grupos más pequeños a medida que se avanza en el proceso de integración.
- Se enfoca en la construcción de la funcionalidad más compleja del sistema antes de agregar componentes más básicos.

Ventajas:

- Permite probar la funcionalidad más crítica y compleja del sistema en el proceso de desarrollo.
- Facilita la identificación temprana de problemas en los componentes más críticos.
- Proporciona una visión clara de la funcionalidad del sistema a medida que se va construyendo.

Desventajas:

- Puede retrasar la detección de errores en los módulos de nivel más bajo hasta etapas posteriores del desarrollo.
- Puede ser difícil de implementar si los módulos de nivel más alto no están disponibles o no están completamente desarrollados.

ACTIVIDAD 3.5

DESARROLLO DE UNA APLICACIÓN DE SALUD MÓVIL

Pruebas de integración: al necesitar asegurar que la característica de seguimiento de actividad física se sincroniza correctamente con los wearables, lo más indicado es realizar pruebas de integración. Elegimos esto porque estas nos permitirán probar la interacción entre los distintos componentes del sistema, como la aplicación móvil y los dispositivos wearables, y así poder garantizar su correcto funcionamiento.

ACTUALIZACIÓN DE UN SOFTWARE DE CONTABILIDAD

Pruebas de sistema: las pruebas de sistema son la mejor opción, ya que estas nos podrán permitir evaluar todo el sistema, comprobando que las nuevas mejoras no afecten el funcionamiento general del software.

CREACIÓN DE UN JUEGO MÓVIL

Pruebas de aceptación: al ser un nuevo juego para smartphone, lo más recomendable es usar pruebas de aceptación. Con estas pruebas se busca validar que el juego cumpla con las expectativas del usuario, asegurando que las funciones de interacción en línea y los distintos niveles se comporten como se espera.

SISTEMA DE RESERVAS EN LÍNEA PARA HOTELES

Pruebas de integración: en este caso, lo mejor será realizar pruebas de integración, ya que nos ayudarán a que el sistema de reservas se integre de manera adecuada con el sitio web. Esta estrategia permitirá verificar la interacción entre el sistema de reservas y la plataforma web, asegurando que los usuarios puedan realizar reservas de manera fluida y sin problemas.

ACTIVIDAD 3.6

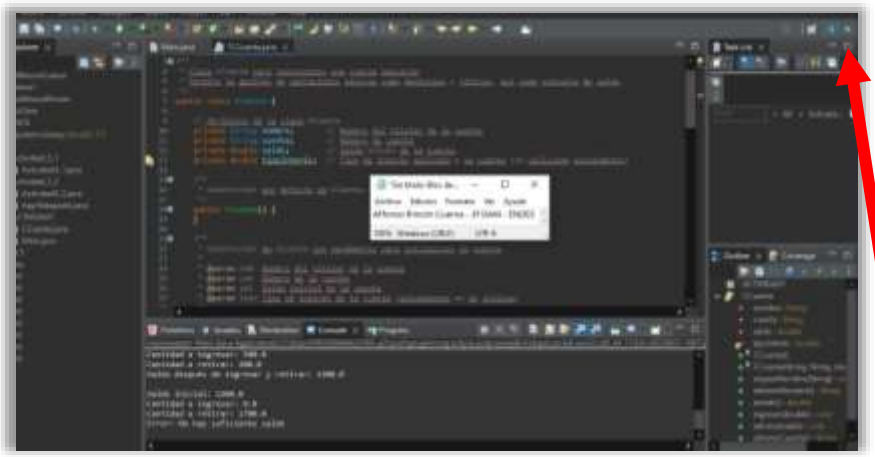
ISO/IEC/IEEE 29119: es un estándar que cubre diferentes aspectos del ciclo de vida del software y proporciona un conjunto de estándares para la gestión de pruebas, procesos de prueba, documentación de prueba y otros aspectos relacionados con las pruebas de software.

I**EEE 829-1998:** es un estándar del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) que se centra en la documentación de pruebas de software. Define los formatos y contenidos para diversos documentos de prueba, como especificaciones de prueba, casos de prueba, procedimientos de prueba o informes de resultados.

PRINCIPALES DIFERENCIAS:

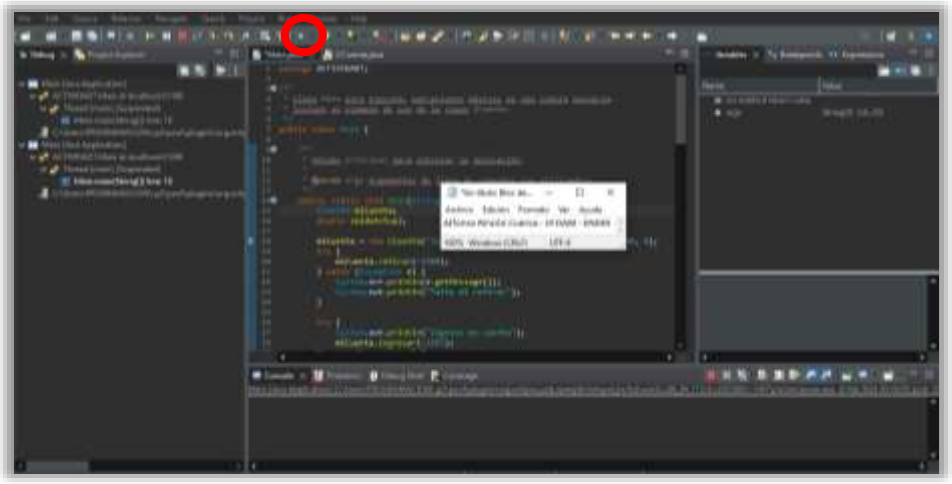
- El ISO abarca aspectos más relacionados con las pruebas de software, mientras que el IEEE se enfoca exclusivamente en la documentación de las pruebas de software.
- La norma ISO/IEC/IEEE 29119 es mucho más amplia que la IEEE 829-1999, ya que puede ser utilizada para diferentes tipos de software, sistemas, hardware o procesos de negocio que requieran pruebas.

ACTIVIDAD 3.7



Vamos a realizar el proceso de depuración en Eclipse. Para ello, lo primero que debemos hacer es abrir nuestro proyecto.

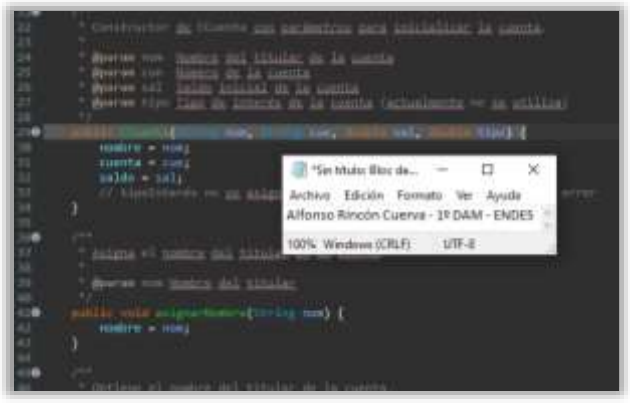
Pulsamos en este botón para cambiar la mesa de trabajo a modo Debug.

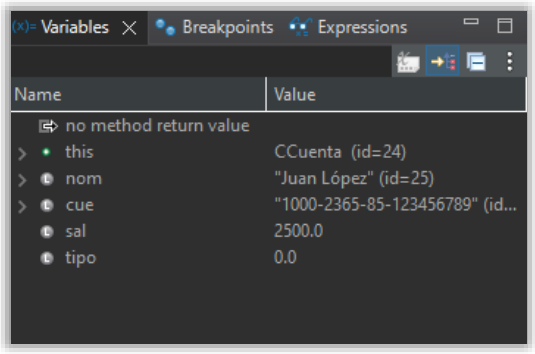


Al hacer esto se nos abrirá este espacio de trabajo, con las pestañas “Variables” y “Debug”.

Pulsamos en el botón que se indica en la foto, para empezar el proceso de depuración.

Tras esto, empieza la depuración. Se ejecutará la primera instrucción del programa.

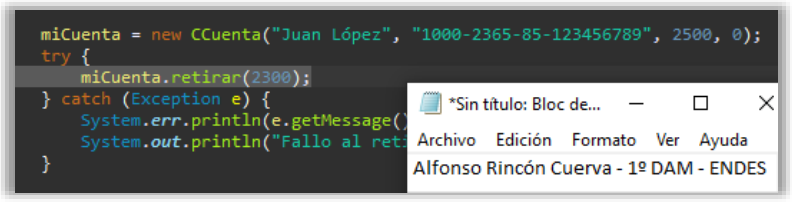


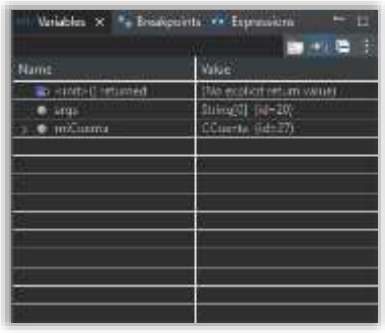


A su vez, en la pestaña de variables, podremos ver distintos datos en función de cada instrucción que vayamos viendo con el depurador.

PRUEBAS A MANO

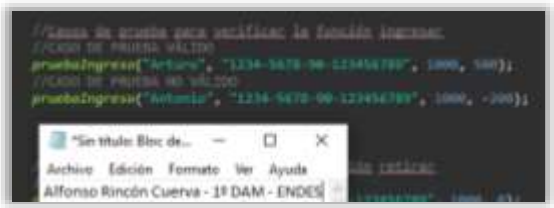
miCuenta.retirar(2300): tras ejecutar esta instrucción en el depurador, obtenemos los siguientes valores en la pestaña de variables.

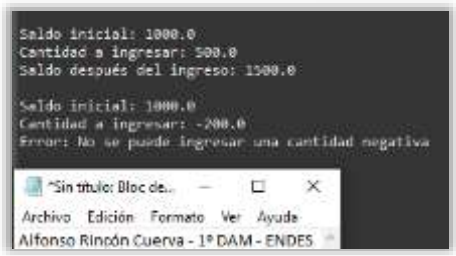




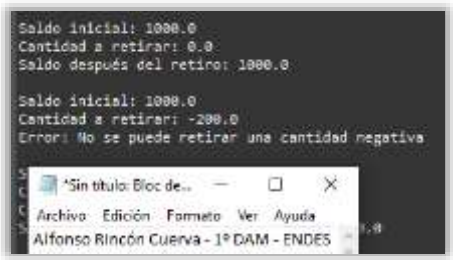
Ahora realizaremos las siguientes pruebas de caja blanca, para ver el correcto funcionamiento de cada método. Se hará 1 prueba con un caso de prueba válido y una con un caso de prueba no válido por cada método.

INGRESAR: para poder realizar las pruebas del método *ingresar*, crearemos un método llamado *pruebaIngreso* en la clase **Main**.

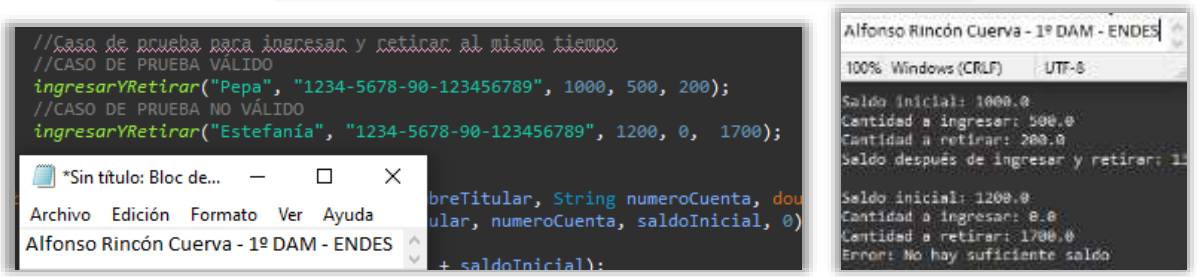




RETIRAR: para poder realizar las pruebas del método *retirar*, crearemos un método llamado *pruebaRetiro* en la clase **Main**.

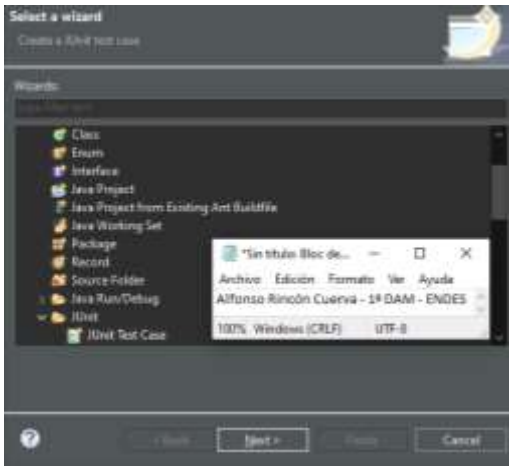


INGRESAR Y RETIRAR: para poder realizar las pruebas de los métodos *retirar* e *ingresar*, crearemos un método llamado *ingresoYRetiro* en la clase **Main**.



PRUEBAS AUTOMATIZADAS

Ahora es el momento de realizar pruebas automatizadas con Junit, la cual es una herramienta integrada en Eclipse especializada en hacer pruebas unitarias automatizadas.

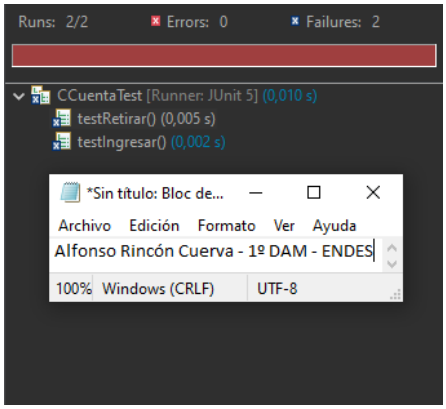


Hacemos Click Dcho en la clase CCuenta, y vamos a *New* > *Other* > *Junit Test Case.*

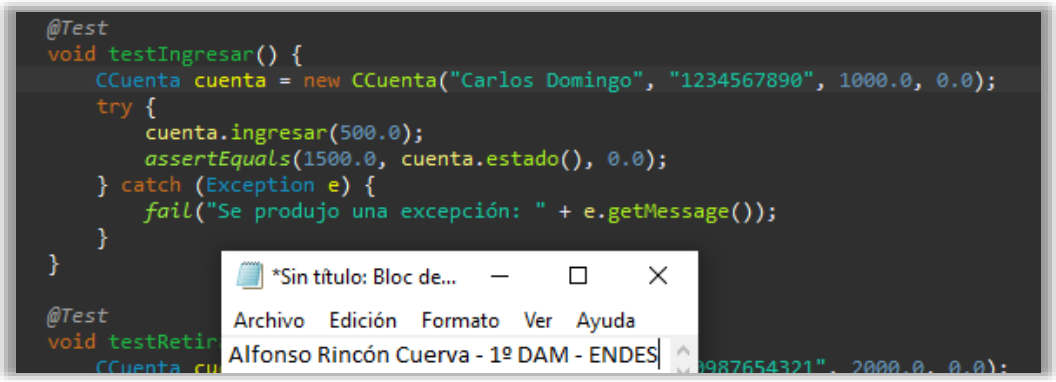
Le daremos a *Next*, y en la siguiente ventana que nos aparece, pulsaremos Next de nuevo. Tendremos que seleccionar los métodos que queremos probar, en este caso ingresar y retirar, y tras seleccionarlos, pulsamos en *Finish*.

Se habrá creado una clase con el nombre de **CCuentaTest**. Hacemos Click Dcho en esta, y *Run As* > *JUnit Test*.

Nos aparecerá la siguiente pesataña.

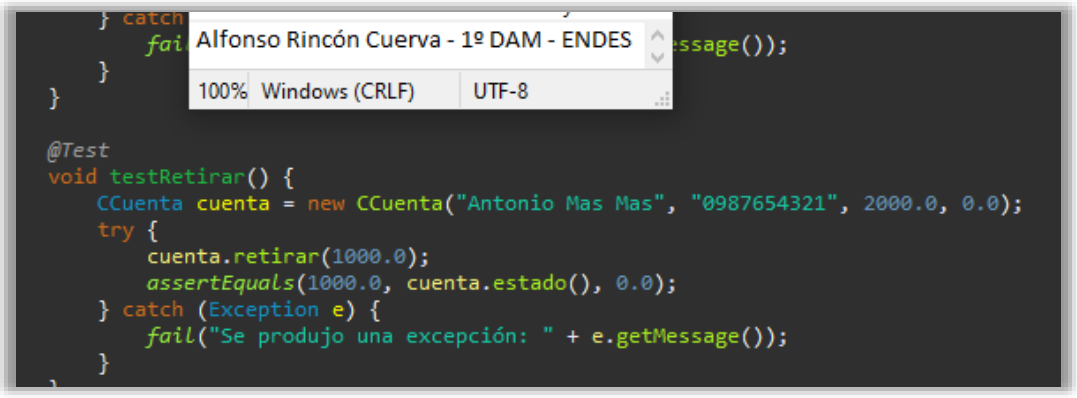


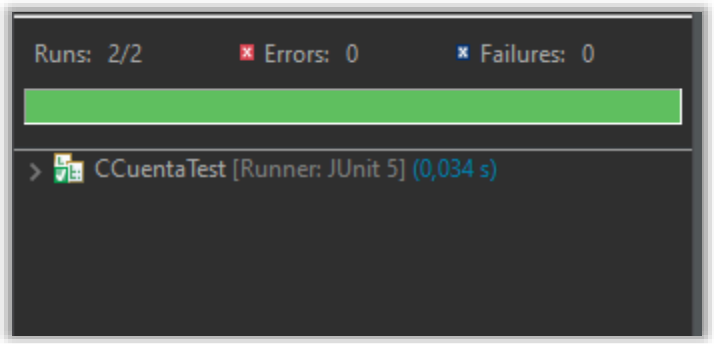
CASO DE PRUEBA INGRESAR: realizamos el primer caso de prueba automatizado en JUnit con el método ingresar.



Tras haber escrito, el código, los ejecutaremos. La pestaña de Junit nos dirá si hay errores o no en las pruebas.

CASO DE PRUEBA RETIRAR: el segundo caso de prueba que haremos será del método retirar.





Como podemos observar, ambos casos de prueba fueron válidos.

BIBLIOGRAFÍA

https://unagaenapuros.wordpress.com/2021/03/31/071-pruebas-de-integracion-ii/

https://www.zaptest.com/es/que-son-las-pruebas-de-integracion-profundizacion-en-los-tipos-el-proceso-y-la-aplicacion