

# ED03\_Tarea

# Índice

1.	Punto 1 .....	3
2.	Punto 2 .....	6
3.	Punto 3 .....	10
4.	Punto 4 .....	14

## 1. Punto 1

2. Realiza una ejecución paso a paso, que verifique el correcto funcionamiento de la aplicación. Indica los valores que marca la inspección de variables tras ejecutar la instrucción

```
miCuenta.retirar(2300)
```

en la función

main

Al iniciar se crea un objeto miCuenta y se inicializan sus atributos, lo podemos ver al relizar la ejecución paso a paso y ver el valor de las variables:

The screenshot shows an IDE with a Java file named `prácticaunidad3.Main`. The code is as follows:

```
9  *
10 * @author Francico Javier Cabrerizo Membrilla
11 */
12 public class Main {
13
14     /**
15     * @param args the command line arguments
16     */
17     public static void main(String[] args) {
18         CCuenta miCuenta;
19         double saldoActual;
20
21         miCuenta = new CCuenta("Juan López", "1000-2365-85-123456789", 2
22     try
23     {
24         miCuenta.retirar(2300);
25     } catch (Exception e)
26     {
27         System.out.print("Fallo al retirar");
28     }
29
30     try
31     {
```

Below the code editor, the `Variables` tab is active, displaying the state of the program:

Name	Type	Value
<Enter new watch>		
this	CCuenta	#92
nom	String	"Juan López"
cue	String	"1000-2365-85-123456789"
sal	double	2500.0
tipo	double	0.0

Al ejecutar retirar saldo podemos ver el valor que se va a retirar, cantidad:

```
    */
    public void retirar (double cantidad) throws Exception
    {
        if (cantidad <= 0)
            throw new Exception ("No se puede retirar una cantidad neg
        if (estado() < cantidad)
            throw new Exception ("No se hay suficiente saldo");
        saldo = saldo - cantidad;
    }

    // Método que me devuelve el número de cuenta
    public String obtenerCuenta ()
    {
        return cuenta;
    }
}
```

ácticaunidad3.CCuenta > retirar >

Variables ×

Name	Type	Value
<Enter new watch>		
this	CCuenta	#92
nombre	String	"Juan López"
cuenta	String	"1000-2365-85-123456789"
saldo	double	2500.0
tipoInterés	double	0.0
cantidad	double	2300.0

El saldo se actualiza después de la ejecución de retirar:

The screenshot shows an IDE with a Java class named `prácticaunidad3.CCuenta`. The `retirar` method is highlighted in green, indicating it is the current execution point. The code defines a `retirar` method that checks for negative amounts and insufficient balance before updating the `saldo` (balance) by subtracting the `cantidad` (amount).

```
64 public void retirar (double cantidad) throws Exception
65 {
66     if (cantidad <= 0)
67         throw new Exception ("No se puede retirar una cantidad neg
68     if (estado() < cantidad)
69         throw new Exception ("No se hay suficiente saldo");
70     saldo = saldo - cantidad;
71 }
72
73 // Método que me devuelve el número de cuenta
74 public String obtenerCuenta ()
75 {
76     return cuenta;
77 }
78 }
79
```

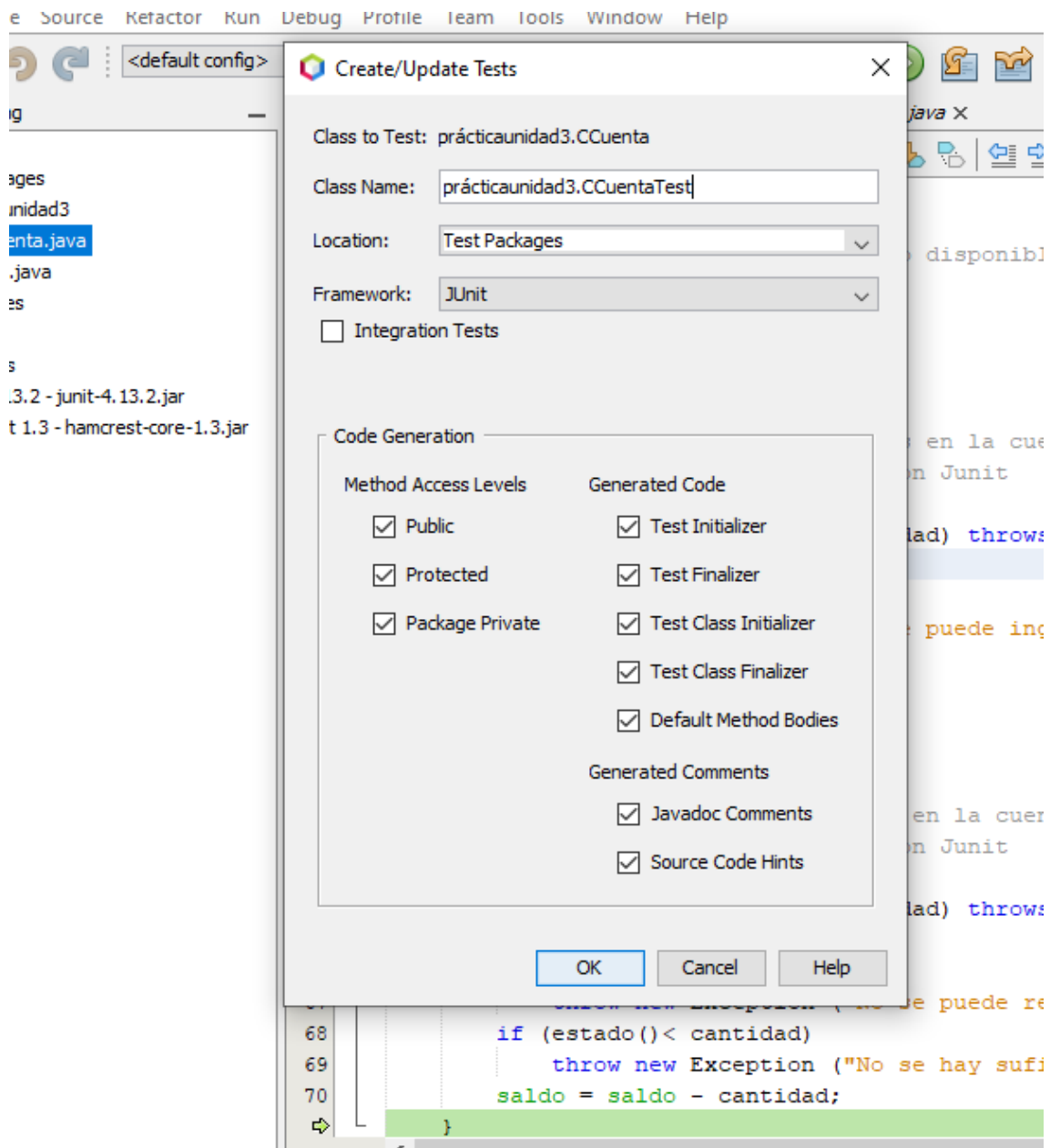
Below the code editor, the `Variables` tab is active, showing the state of the program:

Name	Type	Value
<Enter new watch>		
this	CCuenta	#92
nombre	String	"Juan López"
cuenta	String	"1000-2365-85-123456789"
saldo	double	200.0
tipoInterés	double	0.0
cantidad	double	2300.0

## 2. Punto 2

3. Diseña un caso de prueba que permita verificar el método **ingresar**.

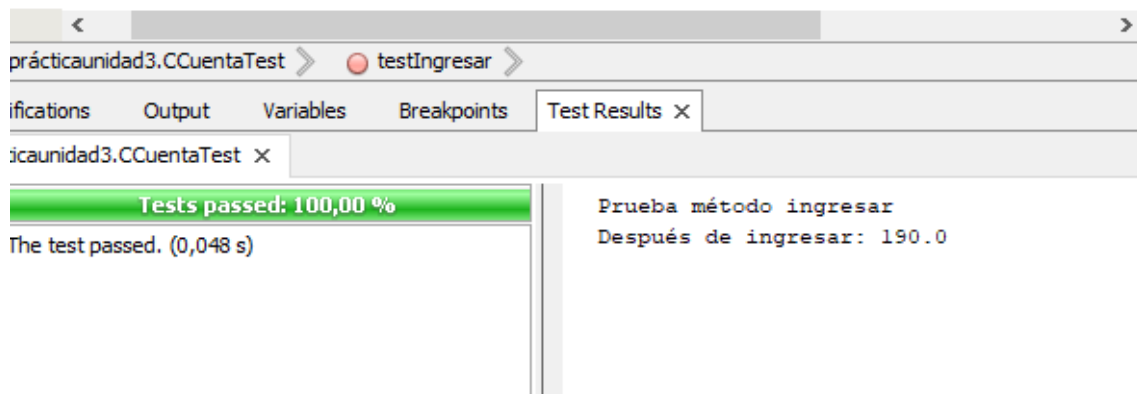
Para las pruebas hay que crear una clase Test de JUNIT con los valores por defecto:



Hay que eliminar los métodos que no se quieren probar y modificar el que queramos.

Prueba del método de test con valor 190:

```
/**
 * Test of ingresar method, of class CCuenta.
 */
public void testIngresar() throws Exception {
    System.out.println("Prueba método ingresar");
    double cantidad = 190;
    CCuenta instance = new CCuenta();
    instance.ingresar(cantidad);
    System.out.println("Después de ingresar: " + instance.estado());
    // TODO review the generated test code and remove the default call
}
```



La ejecución del test es correcta.

En caso de poner valor negativo -190:

```

1 public void testIngresar() throws Exception {
2     System.out.println("Prueba método ingresar");
3     double cantidad = -190;
4     CCuenta instance = new CCuenta();
5     instance.ingresar(cantidad);
6     System.out.println("Después de ingresar: " + instance.estado());
7     // TODO review the generated test code and remove the default call
8 }

```

prácticaunidad3.CCuentaTest > testIngresar > cantidad

Notifications Output Variables Breakpoints Test Results X

prácticaunidad3.CCuentaTest X

Tests passed: 0,00 %

No test passed, 1 test caused an error. (0,05 s)

prácticaunidad3.CCuentaTest Failed

testIngresar caused an ERROR: No se puede ingresar una cantidad negativa

java.lang.Exception

at prácticaunidad3.CCuenta.ingresar(CCuenta.java:10)

at prácticaunidad3.CCuentaTest.testIngresar(CCuentaTest.java:5)

Prueba método ingresar

Da un error controlado en el método por lo tanto la ejecución es correcta.

Prueba al ingresare la cantidad 0:

```

16 super(testName);
17 }
18
19 @Override
20 protected void setUp() throws Exception {
21     super.setUp();
22 }
23
24 @Override
25 protected void tearDown() throws Exception {
26     super.tearDown();
27 }
28
29 /**
30  * Test of ingresar method, of class CCuenta.
31  */
32 public void testIngresar() throws Exception {
33     System.out.println("Prueba método ingresar");
34     double cantidad = 0;
35     CCuenta instance = new CCuenta();
36     instance.ingresar(cantidad);
37     System.out.println("Después de ingresar: " + instance.estado());
38     // TODO review the generated test code and remove the default call
39 }
40
41

```

prácticaunidad3.Testingresar > testIngresar > cantidad

Notifications Output Test Results X

prácticaunidad3.CCuentaTest X prácticaunidad3.TestRetirar X prácticaunidad3.Testingresar X

Tests passed: 0,00 %

No test passed, 1 test caused an error. (0,05 s)

prácticaunidad3.Testingresar Failed

testIngresar caused an ERROR: No se puede ingresar una cantidad negativa

Prueba método ingresar



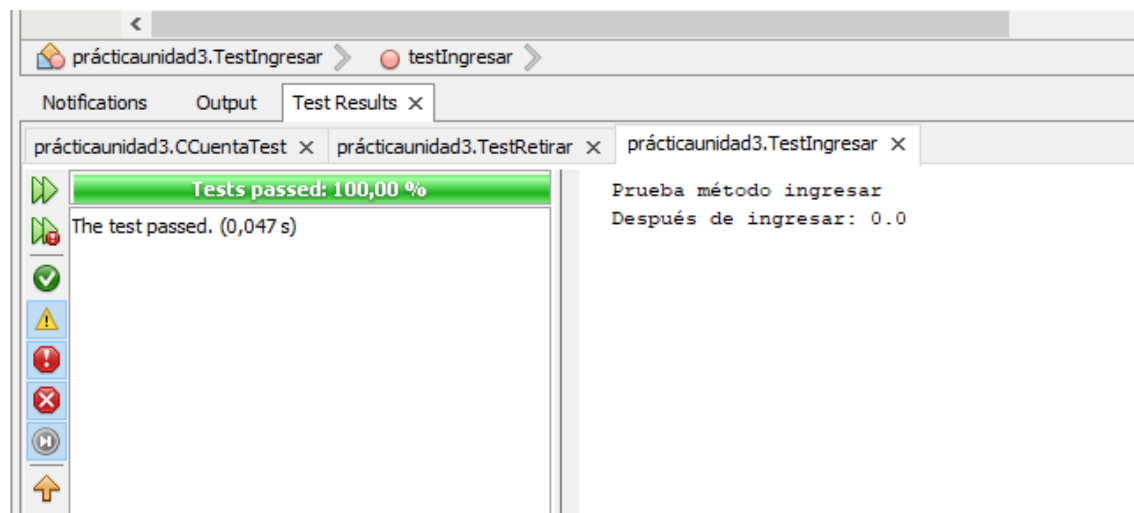
Da un error controlado en el método que no debería dar ya que 0 no es un valor negativo:

```
    */  
    public void ingresar(double cantidad) throws Exception  
    {  
        if (cantidad <= 0)  
            throw new Exception("No se puede ingresar una cantidad negativa");  
        saldo = saldo + cantidad;  
    }
```

Para solucionarlo cambiar a: **cantidad < 0**

```
    public void ingresar(double cantidad) throws Exception  
    {  
        if (cantidad < 0)  
            throw new Exception("No se puede ingresar una cantidad negativa");  
        saldo = saldo + cantidad;  
    }
```

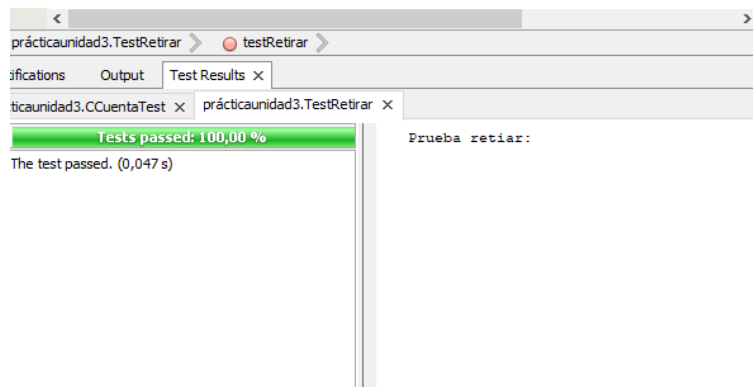
Ahora la ejecución es correcta:



### 3. Punto 3

Prueba a retirar una cantidad, 0:

```
public class TestRetirar extends TestCase {  
    public TestRetirar(String testName) {  
        super(testName);  
    }  
    /**  
     * Test of retirar method, of class CCuenta.  
     */  
    public void testRetirar() throws Exception {  
        System.out.println(" Prueba retirar: ");  
        double cantidad = 0.0;  
        CCuenta instance = new CCuenta();  
        instance.retirar(cantidad);  
        // TODO review the generated test code and remove the default  
        // fail("The test case is a prototype.");  
    }  
}
```



La ejecución es correcta.

Prueba una cantidad negativa:

The screenshot shows an IDE with a Java test method and its execution results. The code is as follows:

```
19  * Test of retirar method, of class CCuenta.  
20  */  
21  public void testRetirar() throws Exception {  
22      System.out.println(" Prueba retirar: ");  
23      double cantidad = -10.0;  
24      CCuenta instance = new CCuenta();  
25      instance.retirar(cantidad);  
26      // TODO review the generated test code and remove the default call to fail  
27      // fail("The test case is a prototype.");  
28  }  
29  }  
30
```

The IDE shows the test results for the method `prácticaunidad3.TestRetirar`. The results indicate that the test failed due to an exception:

Tests passed: 0,00 %  
No test passed, 1 test caused an error. (0,05 s)  
prácticaunidad3.TestRetirar Failed  
testRetirar caused an ERROR: No se puede retirar una cantidad negativa  
java.lang.Exception  
at prácticaunidad3.CCuenta.retirar(CCuenta.java:10)  
at prácticaunidad3.TestRetirar.testRetirar(TestRetirar.java:25)

The output window shows the text: Prueba retirar:

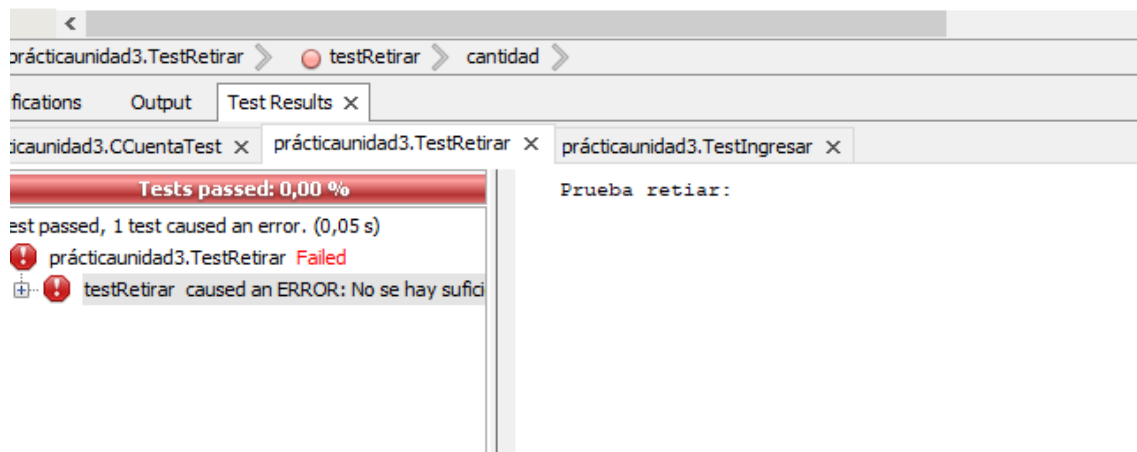
Salta el error controlado en el método, funciona correctamente.

Prueba a retirar una cantidad positiva:

```

public void testRetirar() throws Exception {
    System.out.println(" Prueba retirar: ");
    double cantidad = 10.0;
    CCuenta instance = new CCuenta();
    instance.retirar(cantidad);
    // TODO review the generated test code and remove the default call to
    // fail("The test case is a prototype.");
}
}

```



Da error controlado en el método ya que no hay saldo:

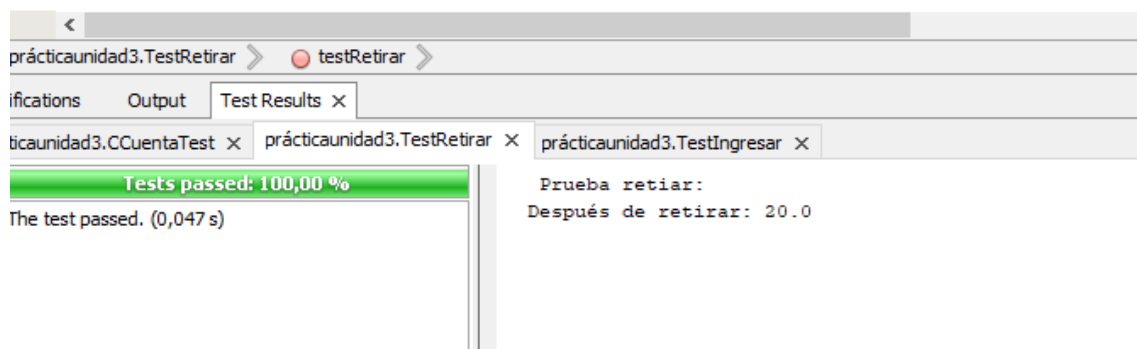
```

public void retirar (double cantidad) throws Exception
{
    if (cantidad < 0)
        throw new Exception ("No se puede retirar una cantidad negativa");
    if (estado() < cantidad)
        throw new Exception ("No se hay suficiente saldo");
    saldo = saldo - cantidad;
}

```

Necesitaremos saldo previo en la cuenta para retirar, para ello ingresaremos primero una cantidad, ingreso, mediante el método ingresar y luego procederemos a retirar:

```
public void testRetirar() throws Exception {  
    System.out.println(" Prueba retirar: ");  
    double cantidad = 10.0;  
    double ingreso = 30.0;  
    CCuenta instance = new CCuenta();  
    instance.ingresar(ingreso);  
    instance.retirar(cantidad);  
    System.out.println("Después de retirar: " + instance.estado());  
    // TODO review the generated test code and remove the default call to fail()  
    // fail("The test case is a prototype.");  
}
```



Podemos ver que el estado se ha actualizado.

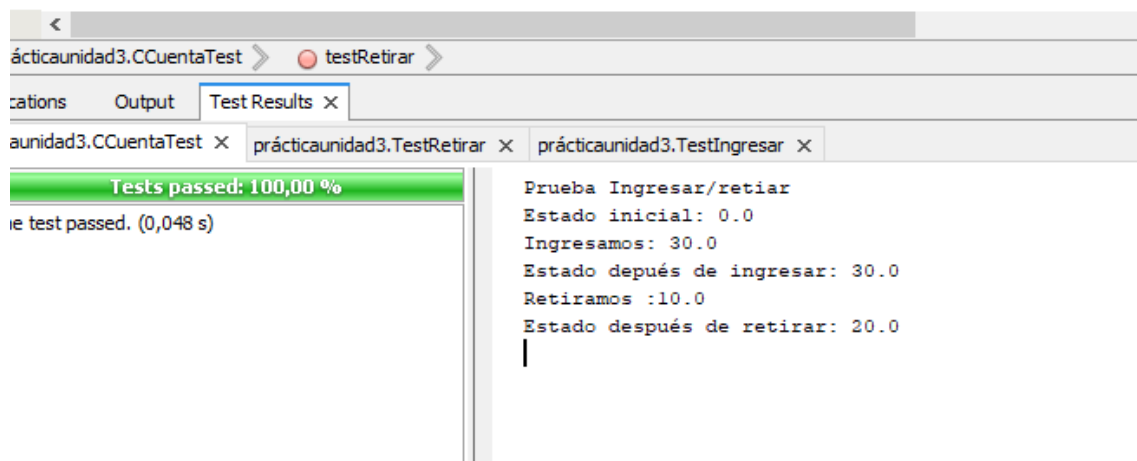
## 4. Punto 4

Es un test parecido al que usamos para retirar ya que para comprobar que retirar se ejecuta correctamente tiene que haber saldo y para esto hay que previamente ingresar mediante el método ingresar previamente o el inicializar el atributo saldo con un valor positivo.

```

    /**
     * Test of retirar ingresar .
     */
    public void testRetirar() throws Exception {
        double retiro = 10.0;
        double ingreso = 30.0;
        CCuenta instance = new CCuenta();
        System.out.println("Prueba Ingresar/retiar ");
        System.out.println("Estado inicial: " + instance.estado());
        System.out.println("Ingresamos: " + ingreso);
        instance.ingresar(ingreso);
        System.out.println("Estado después de ingresar: " + instance.estado());
        System.out.println("Retiramos : " + retiro);
        instance.retirar(retiro);
        System.out.println("Estado después de retirar: " + instance.estado());
    }
}

```



Ingresamos con el método ingresar una cantidad para luego retirar otra y comprobamos que se ejecuta correctamente.

Al final mediante Refactor he renombrado las clases prueba.