ED03_Tarea

Gianfranco Álvarez Valencia DAM 30/01/2022

Índice

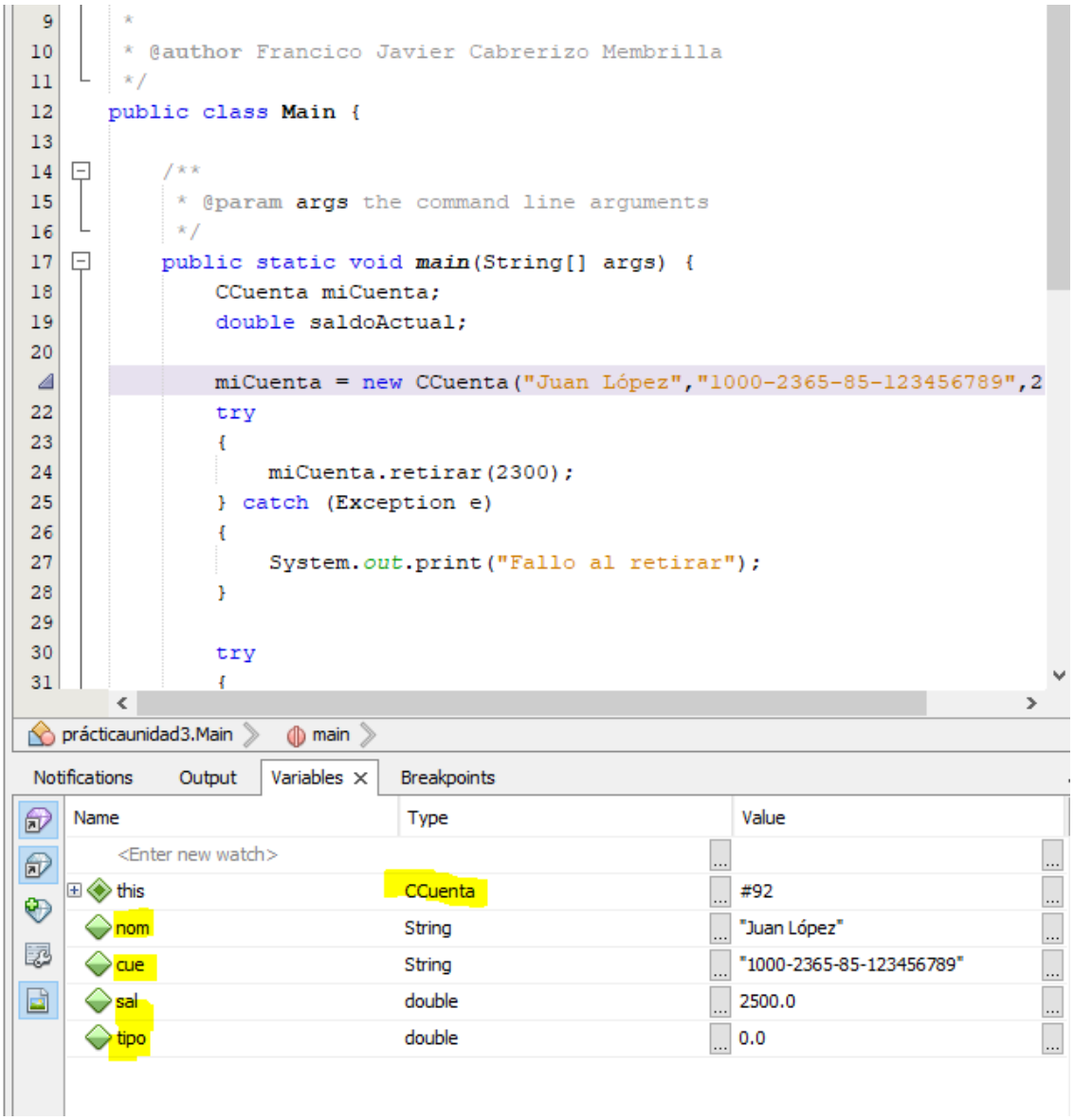
1. Punto 1 3
2. Punto 2 6
3. Punto 3 10
4. Punto 4 14

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:



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

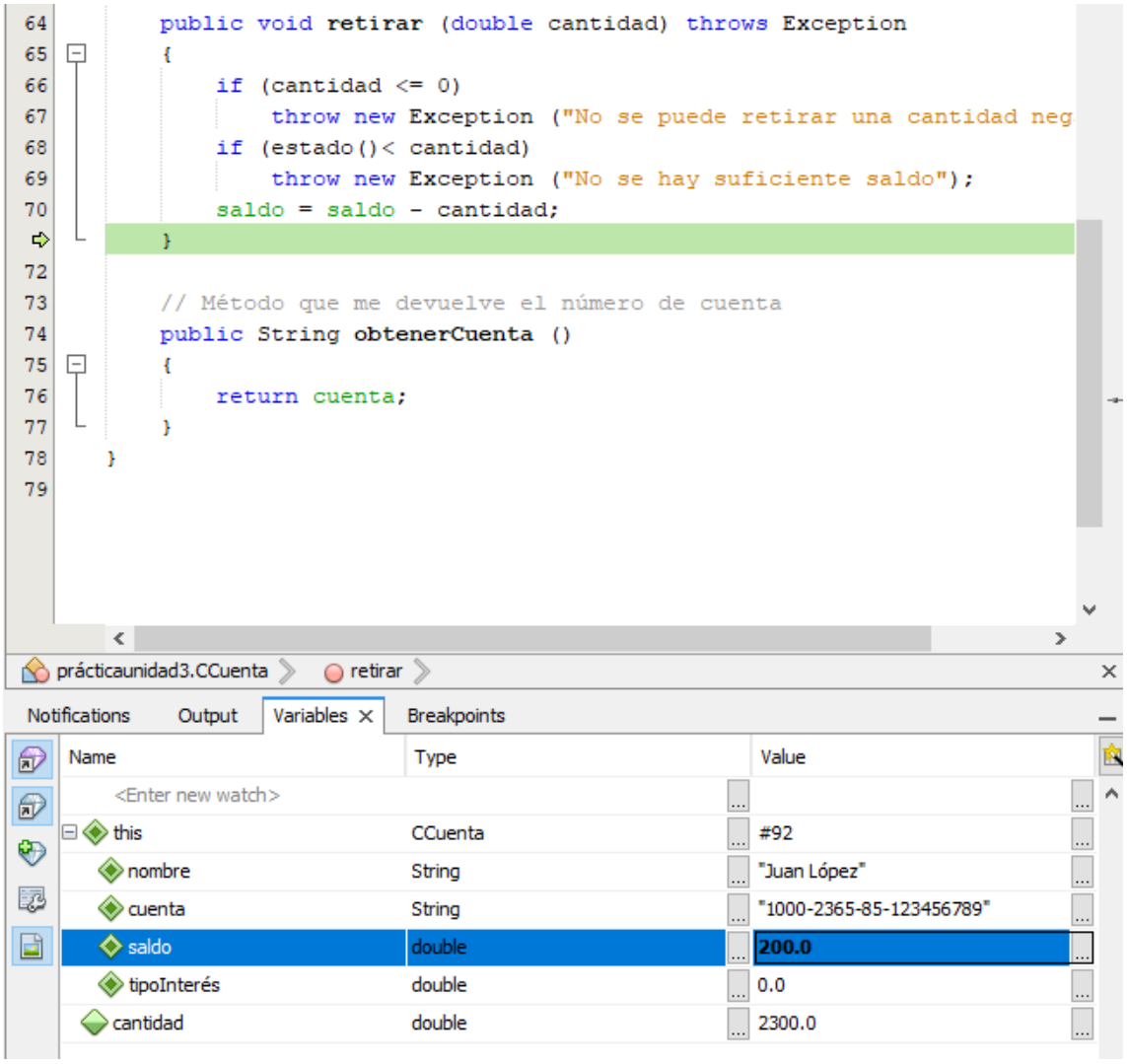
```
     */
    public void retirar (double cantidad) throws Exception
    {
        if (cantidad <= 0)
            throw new Exception ("No se puede retirar una cantidad neg
        if (estado()< cantidad)
            throw new Exception ("No se hay suficiente saldo");
        saldo = saldo - cantidad;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

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

ácticaunidad3.CCuenta › retirar

cations | Output | Variables | Breakpoints

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:



```
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;
        }
72
73      // Método que me devuelve el número de cuenta
74      public String obtenerCuenta ()
75      {
76          return cuenta;
77      }
78  }
79
```

prácticaunidad3.CCuenta › retirar

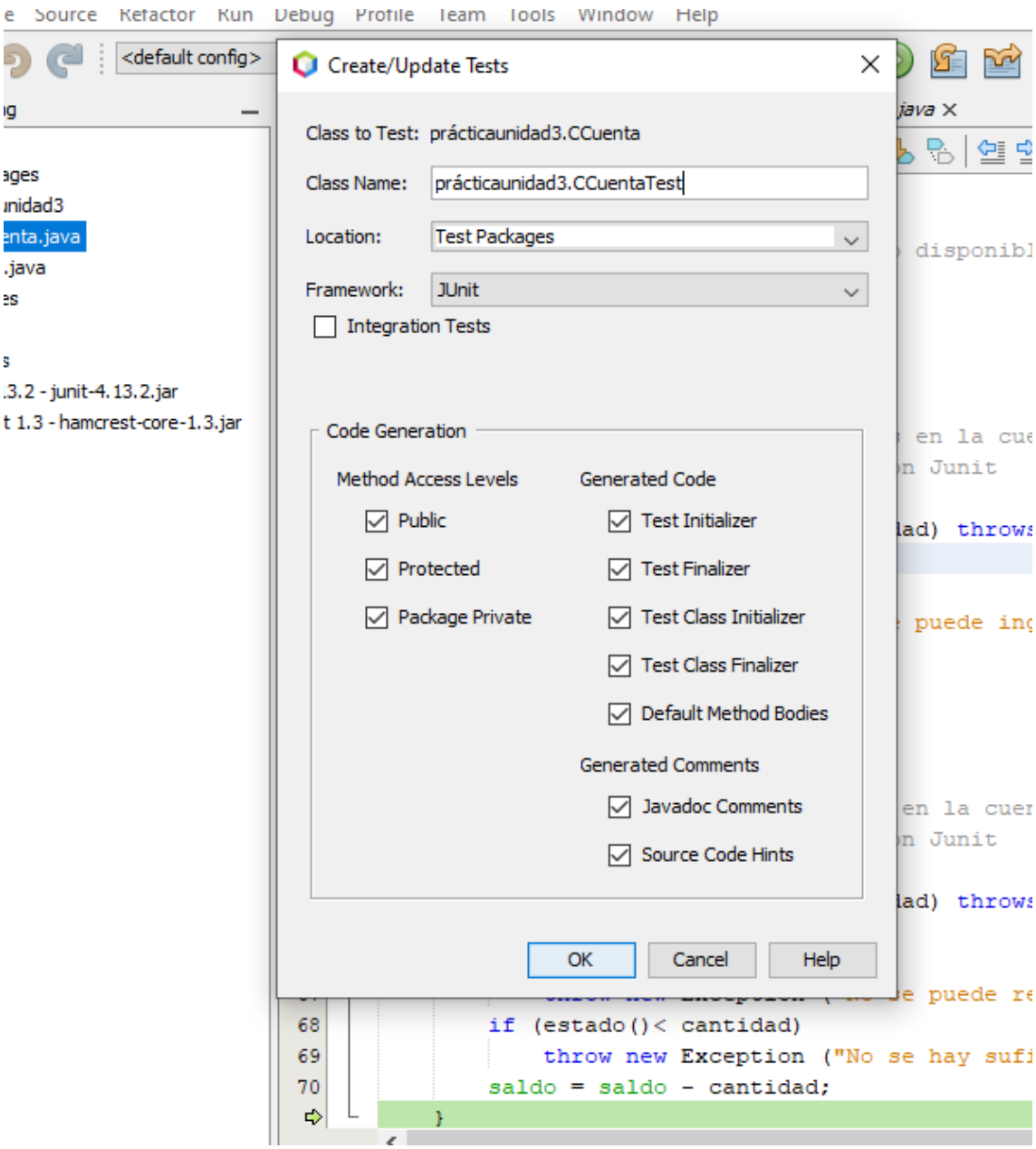
Notifications | Output | Variables | Breakpoints

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**.

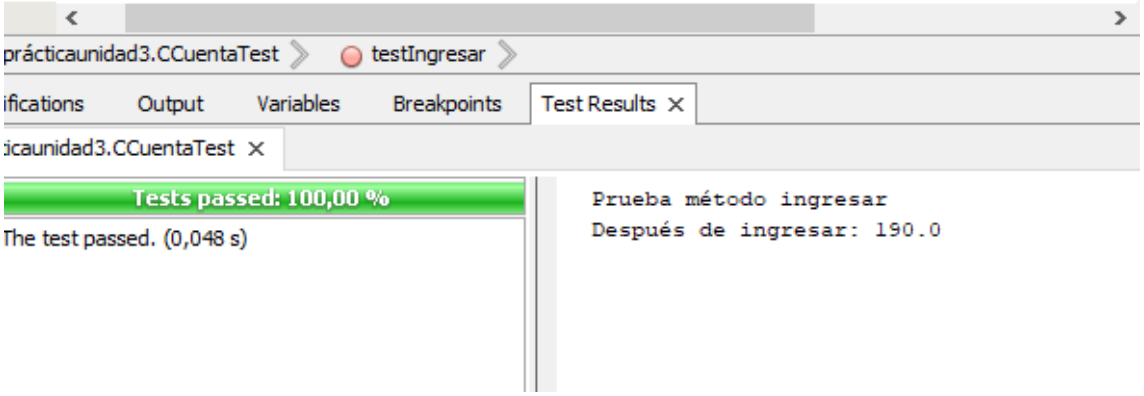
Para las pruebas hay que crear una clase Test de JUNIT con los valores por defecto:



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

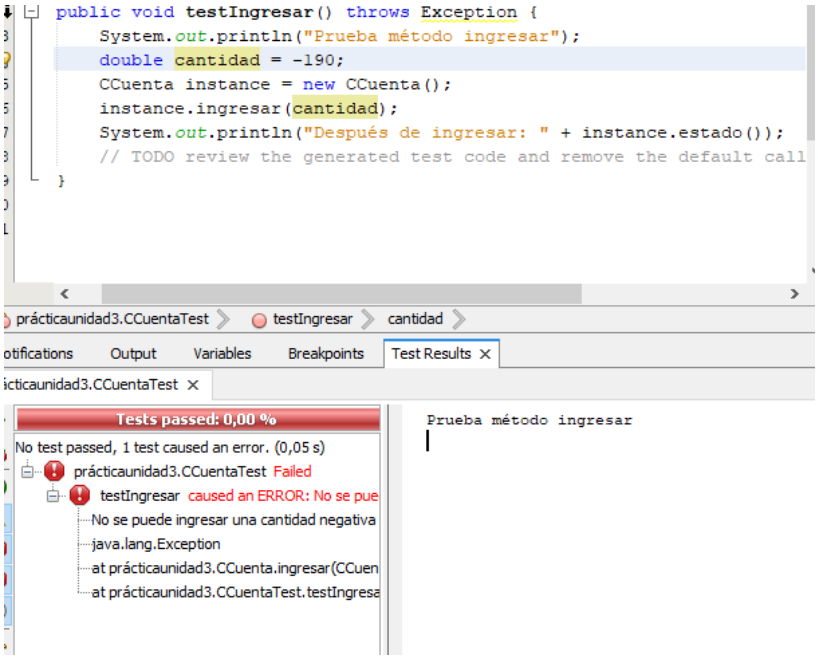
Prueba del método de test con valor 190:

```java
/**
 * 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
}
```



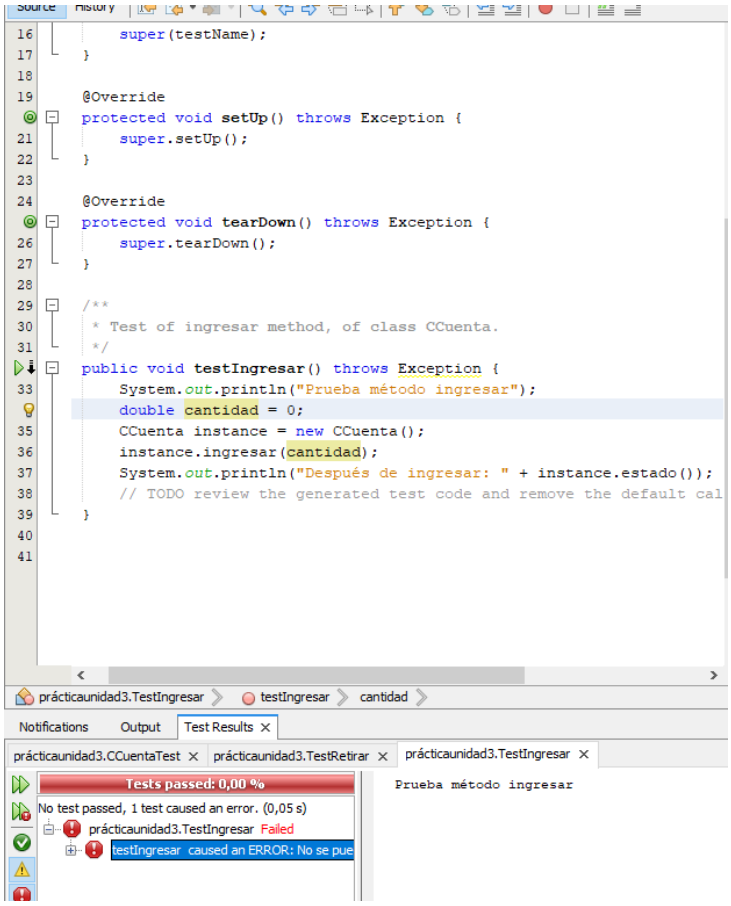
La ejecución del test es correcta.

En caso de poner valor negativo -190:



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

Prueba al ingresare la cantidad 0:



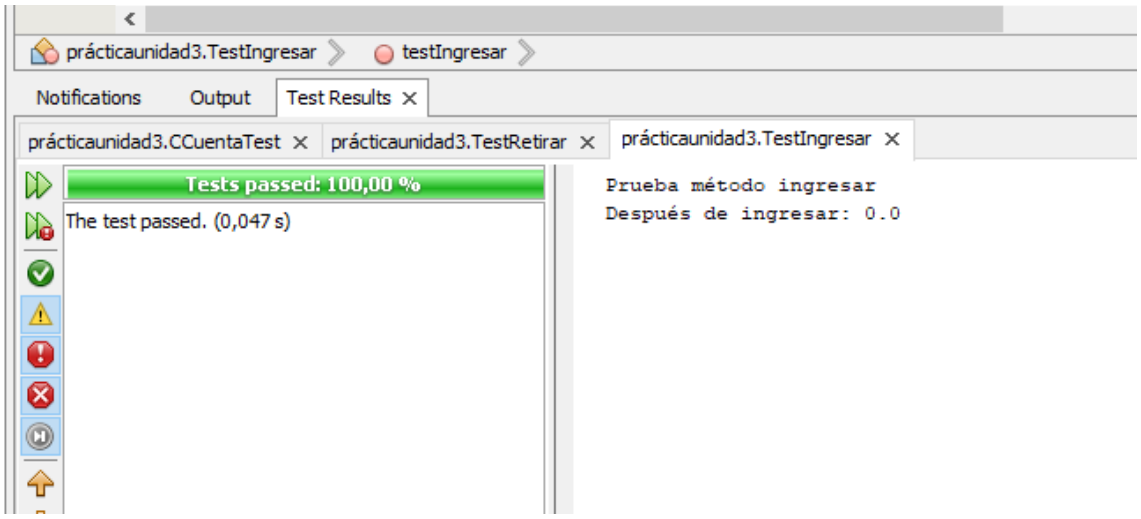
Da un error controlado en el método que no debería dar ya que 0 no es un valor negativo:

```
     */
    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;
    }
```

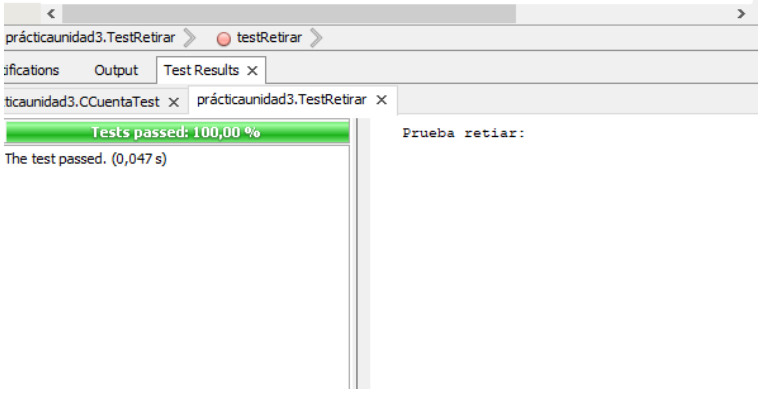
Ahora la ejecución es correcta:



3. Punto 3

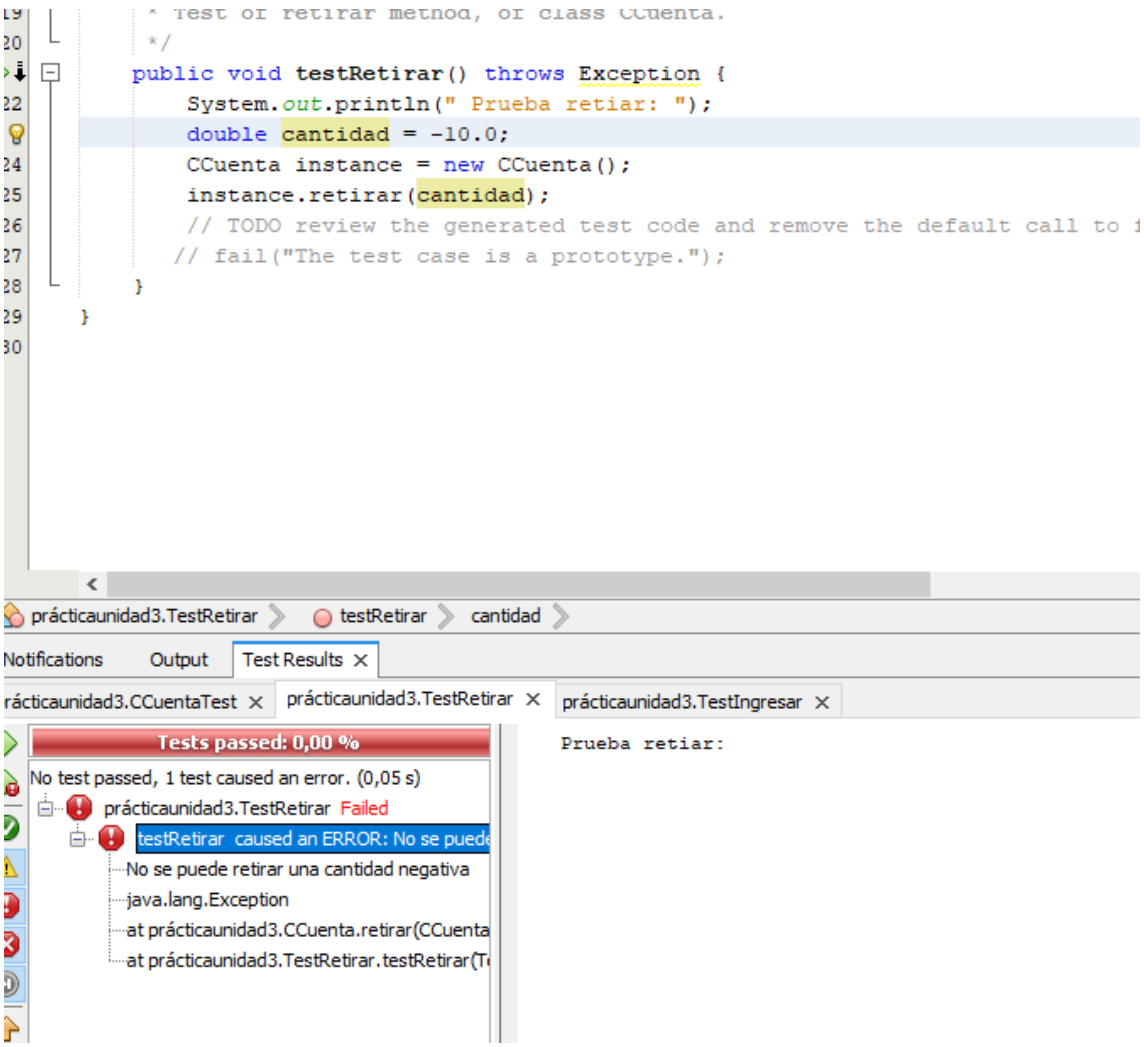
Prueba a retirar una cantidad, 0:

```
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 retiar: ");
        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.

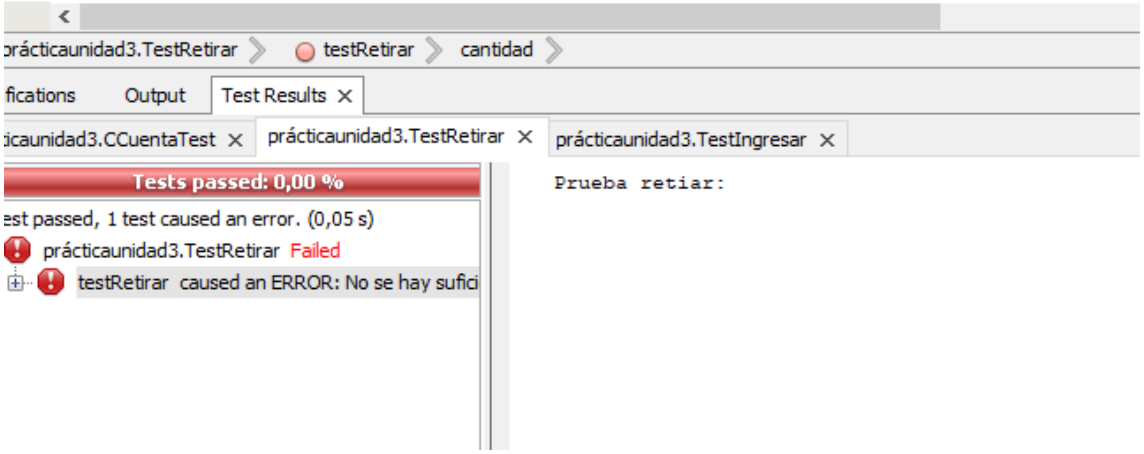
Prueba una cantidad negativa:



Salta el error controlado en el método, funciona correctamente.

Prueba a retirar una cantidad positiva:

```
    public void testRetirar() throws Exception {
        System.out.println(" Prueba retiar: ");
        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.");
    }
}
```

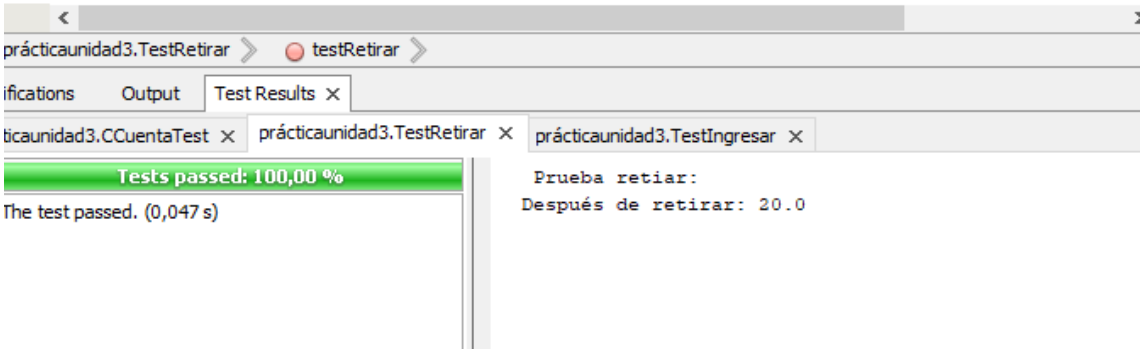


Da error controlado en el método ya que no hay saldo:

```
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 retiar: ");
        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 fa
       // fail("The test case is a prototype.");
    }
}
```

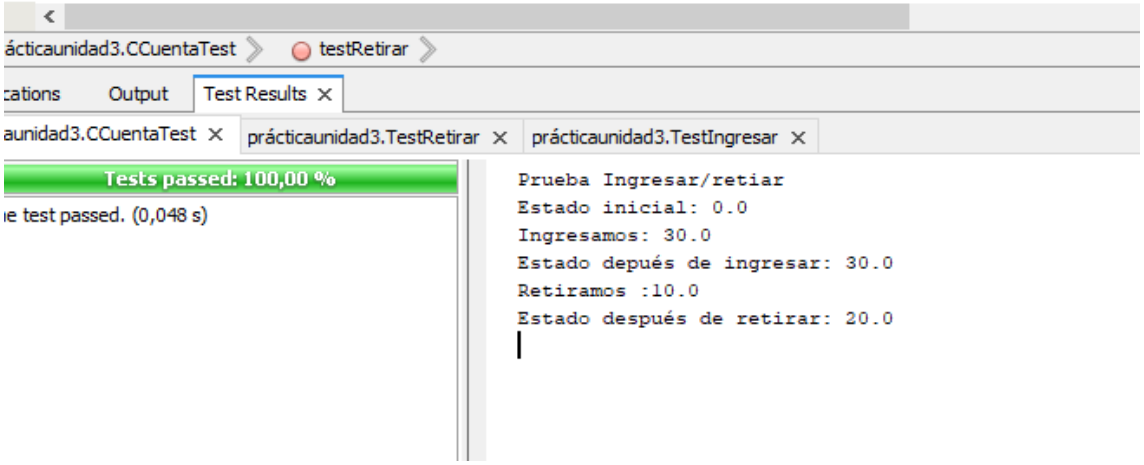


Podemos ver que el estado se ha actualizado.

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 depués 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.