Enunciado.

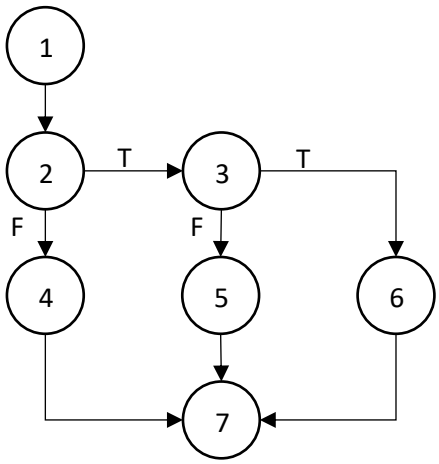
En esta tarea se considera una clase **Java** CCuenta que dispone de los métodos main, ingresar y retirar. Este es el código de los métodos main, ingresar y retirar que deberás tener en cuenta para resolver la tarea:

Deberás realizar un documento donde dar respuesta a los siguientes apartados:

1. Realiza un análisis de caja blanca completo del método ingresar.
   - **Código corregido**

```
/* Metodo para ingresar cantidades en la cuenta. Modifica el saldo.
 * Este metodo va a ser probado con Junit
 */
public int ingresar(double cantidad)
{
        int iCodErr;

   if (cantidad < 0)
   {
      if (cantidad == -3)
      {
        System.out.println("Error detectable en pruebas de caja blanca");
       iCodErr = 2;
      } else
      {
        System.out.println("No se puede ingresar una cantidad negativa");
       iCodErr = 1;
      }
   }
   else
   {
      // Depuracion. Punto de parada. Solo en el 3 ingreso
      dSaldo = dSaldo + cantidad;
      iCodErr = 0;
   }
       // Depuracion. Punto de parada cuando la cantidad es menor de 0
   return iCodErr;
}
```

- **Creación del grafo**

Grafo	Nodo	Líneas de código
	1	public int ingresar(double cantidad) { int iCodErr;
	2	if (cantidad < 0)
	3	if (cantidad == -3)

	4	{
		dSaldo = dSaldo + cantidad;
		iCodErr = 0;
		}

	5	{
		System.out.println("No se puede ingresar una cantidad negativa");
		iCodErr = 1;
		}

	6	{
		System.out.println("Error detectable en pruebas de caja blanca");
		iCodErr = 2;
		}

	7	return iCodErr; }

- **Complejidad ciclomática**

Método de cálculo	Complejidad	Comentarios
Nº de regiones	3	Hay que considerar la región exterior.
Nº de aristas – nº de nodos +2	8-7+2=3	
Nº de condiciones +1	2+1=3	Nodos 2 y 3

- **Caminos de prueba**
  - Camino 1: 1 – 2 – 4 – 7
  - Camino 2: 1 – 2 – 3 – 5 – 7
  - Camino 3: 1 – 2 – 3 – 6 – 7

- **Casos de uso, resultados esperados y análisis**

Caminos / Casos de uso	**Datos**	**Salidas**
	cantidad	
1	4	iCodErr=0
2	-1	iCodErr=1 ("No se puede ingresar una cantidad negativa"
3	-3	iCodErr=2 ("Error detectable en pruebas de caja blanca"

2. Realiza un **análisis de caja negra**, incluyendo valores límite y conjetura de errores del método retirar. Debes considerar que este método recibe como parámetro la cantidad a retirar, que no podrá ser menor a 0. Además, en ningún caso esta cantidad podrá ser mayor al saldo actual. Al tratarse de pruebas funcionales no es necesario conocer los detalles del código, pero te lo pasamos para que lo tengas.

➢ **Determinar las clases de equivalencia**

Tipo de dato	**Clase válida**	**Clase no válida**
Caracteres admitidos	Números	Letras
Rango de valores aceptados	Valor >= 0	Valor < 0
Conjunto de valores aceptados	Valor <= Saldo actual	Valor > Saldo actual

➢ **Análisis de valores límite**

	Clase de equivalencia		**Valores límite**	
	Clase válida	**Clase no válida**	**Clase válida**	**Clase no válida**
Valor >= 0	10,53	-5,13	0,1	-0,1
Valor <= Saldo actual. P.E. 995,27	900,00	1000,50	995,26 995,27	995,28

➢ **Conjetura de errores**

Como conjetura de errores vamos a comprobar el valor 0, por tratarse de un número que comúnmente puede causar errores en cuentas matemáticas.

➢ **Casos de uso, resultados esperados y análisis**

Como tenemos que tratar posibles errores teniendo en cuenta el saldo actual, vamos a tomar como saldo 995,27.

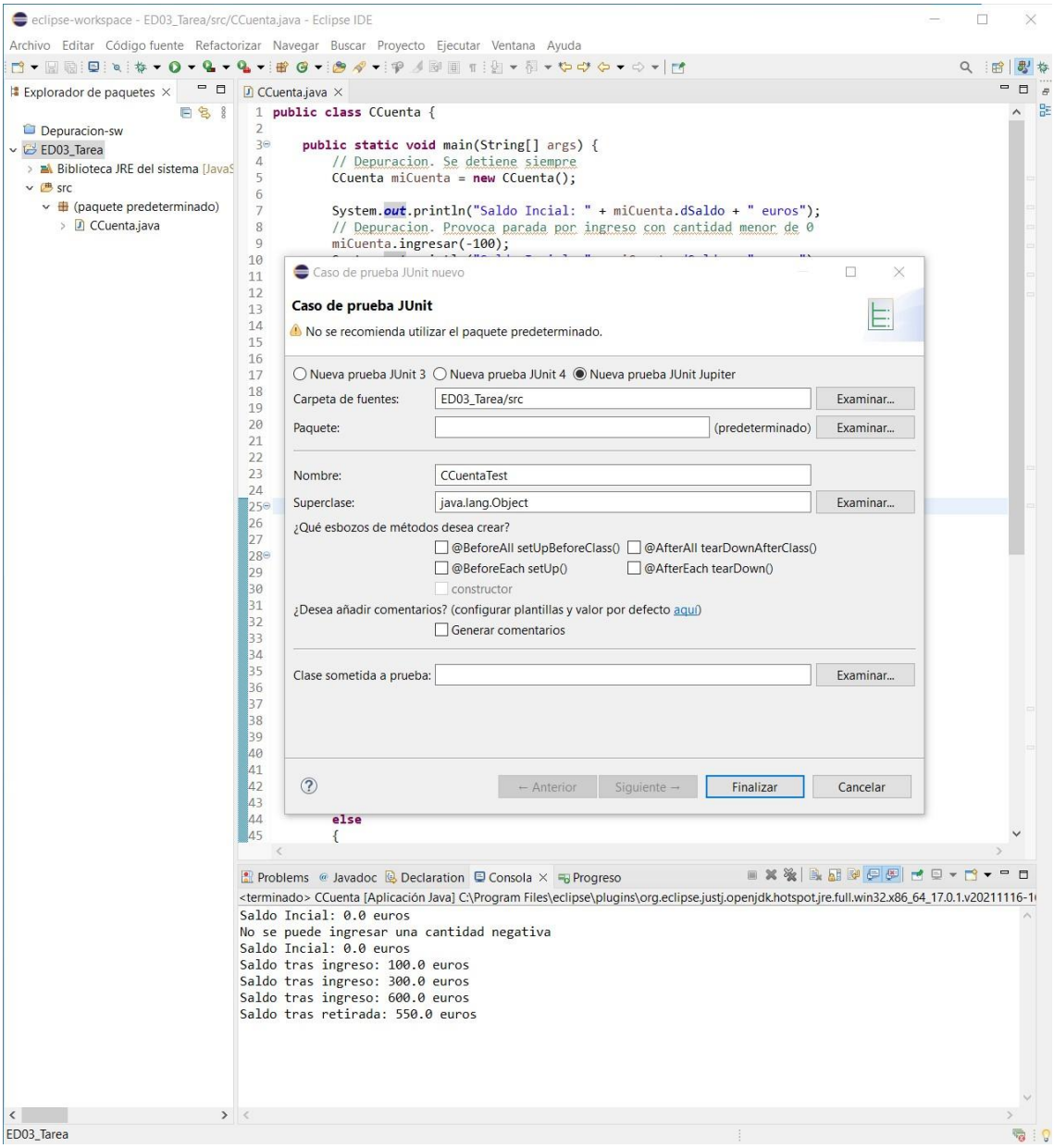
Condición de entrada	Clases de equivalencia	Clases válidas	COD	Clases no válidas	COD
Cantidad a retirar	Rango de valores aceptados	Un número mayor que 0	C1B1	Una letra	C1E1
				Un número menor que 0	C1E2
		Valor límite 0	C1B2	Valor límite - 0,1	C1E3
	Conjunto de valores aceptados respecto de saldo	Un número positivo menor que saldo	C1B3	Un número mayor que saldo	C1E4
		Valor límite 995,27	C1B4	Valor límite 995,28	C1E5

Casos de prueba	Clases de equivalencia	Condiciones de entrada	Resultado esperado
		Cantidad a retirar	
CP1	C1B1	5	Saldo actual en cuenta (990,27)
CP2	C1B2	0	Saldo actual en cuenta (995,27)
CP3	C1B3	900	Saldo actual en cuenta (95,27)
CP4	C1B4	995,27	Saldo actual en cuenta (0)
CP6	C1E1	E	Error de carácter introducido.
CP7	C1E2	-5	No se puede retirar una cantidad negativa
CP8	C1E3	-0,1	No se puede retirar una cantidad negativa
CP9	C1E4	1000	Saldo insuficiente
CP10	C1E5	995,28	Saldo insuficiente

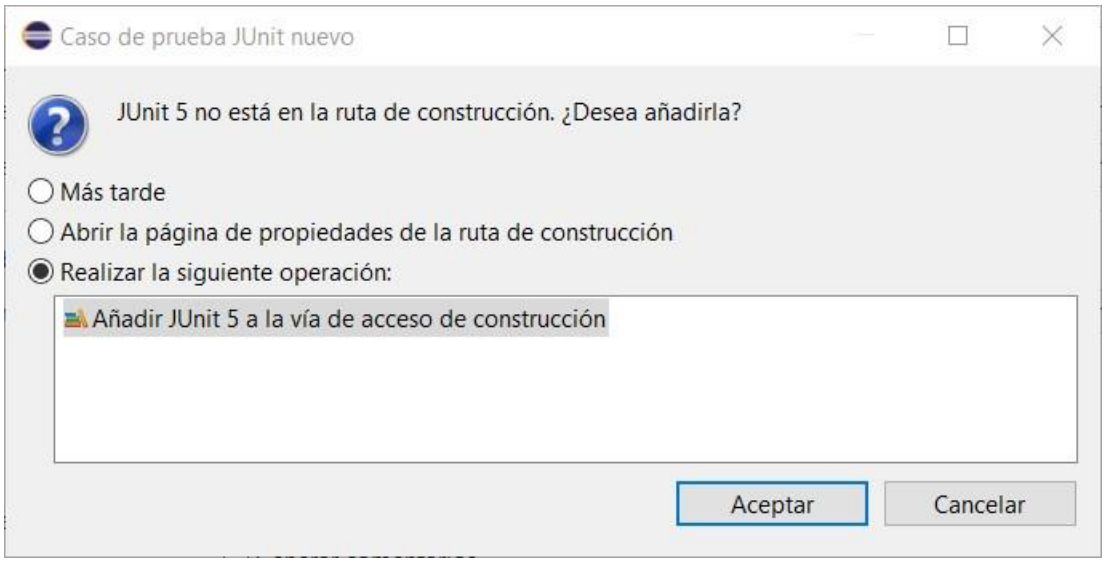
3. Crea la clase CCuentaTest del tipo Caso de prueba JUnit 5 en Eclipse que nos permita pasar las pruebas unitarias de caja blanca del método ingresar. Los casos de prueba ya los habrás obtenido en el primer apartado del ejercicio 1 y tendrás que aplicarlo en el código de la prueba. Copia el código fuente correspondiente que te proporcionamos.

```
25  /* Metodo para ingresar cantidades en la cuenta. Modifica el saldo.
26   * Este metodo va a ser probado con Junit
27   */
28  public int ingresar(double cantidad)
29  {
30      int iCodErr;
31
32      if (cantidad < 0)
33      {
34          if (cantidad == -3)
35          {
36              System.out.println("Error detectable en pruebas de caja blanca");
37          iCodErr = 2;
38          } else
39          {
40              System.out.println("No se puede ingresar una cantidad negativa");
41          iCodErr = 1;
42          }
43      }
44      else
45      {
46          // Depuracion. Punto de parada. Solo en el 3 ingreso
47          dSaldo = dSaldo + cantidad;
48          iCodErr = 0;
49      }
50      // Depuracion. Punto de parada cuando la cantidad  es menor de 0
51      return iCodErr;
52  }
```

Metodo *ingresar* sobre el cual vamos a hacer las pruebas unitarias de caja blanca.



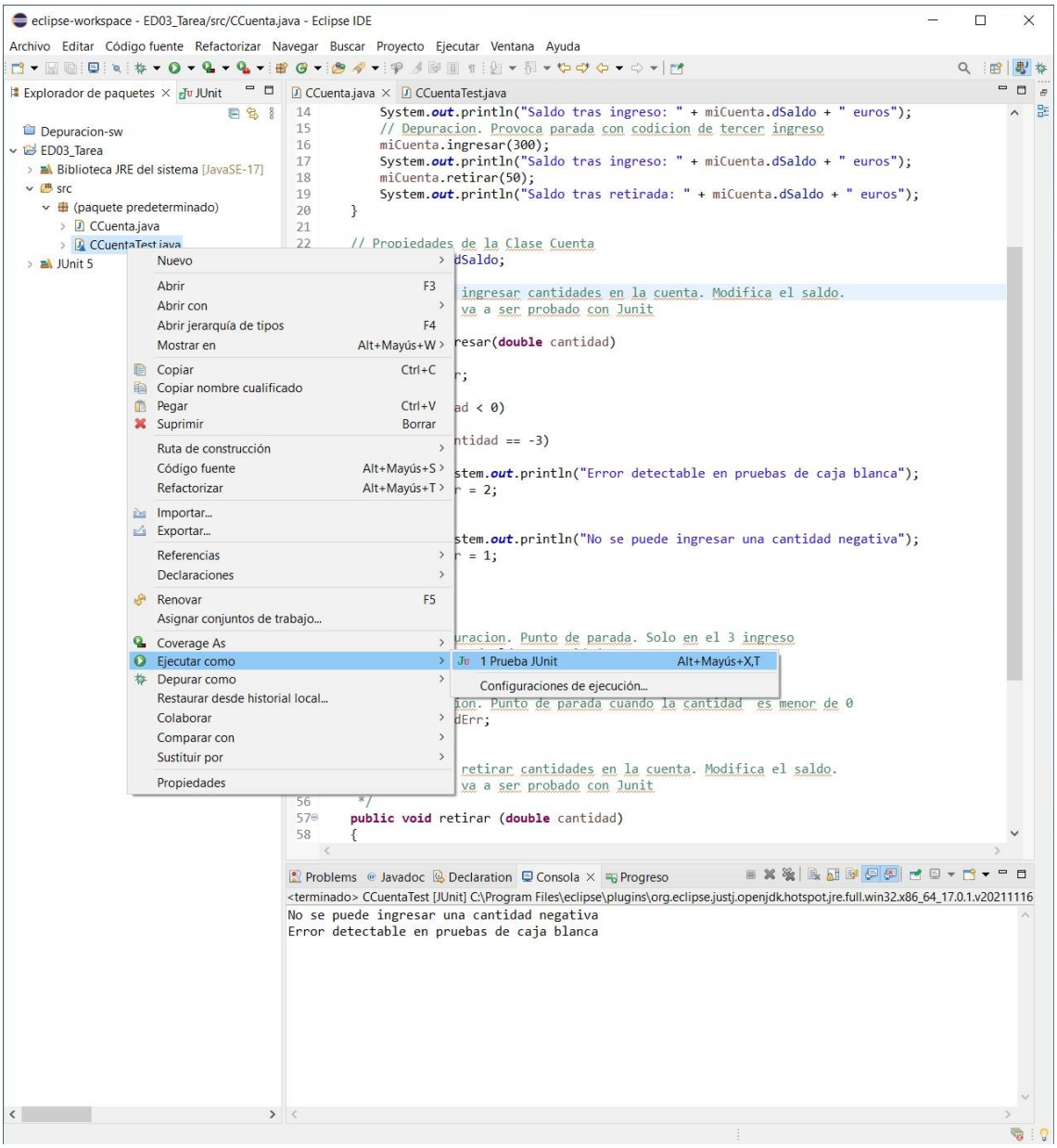
Creamos la clase de prueba del tipo *Caso de prueba JUnit 5* a través de eclipse y lo llamamos *CCuentaTest.*



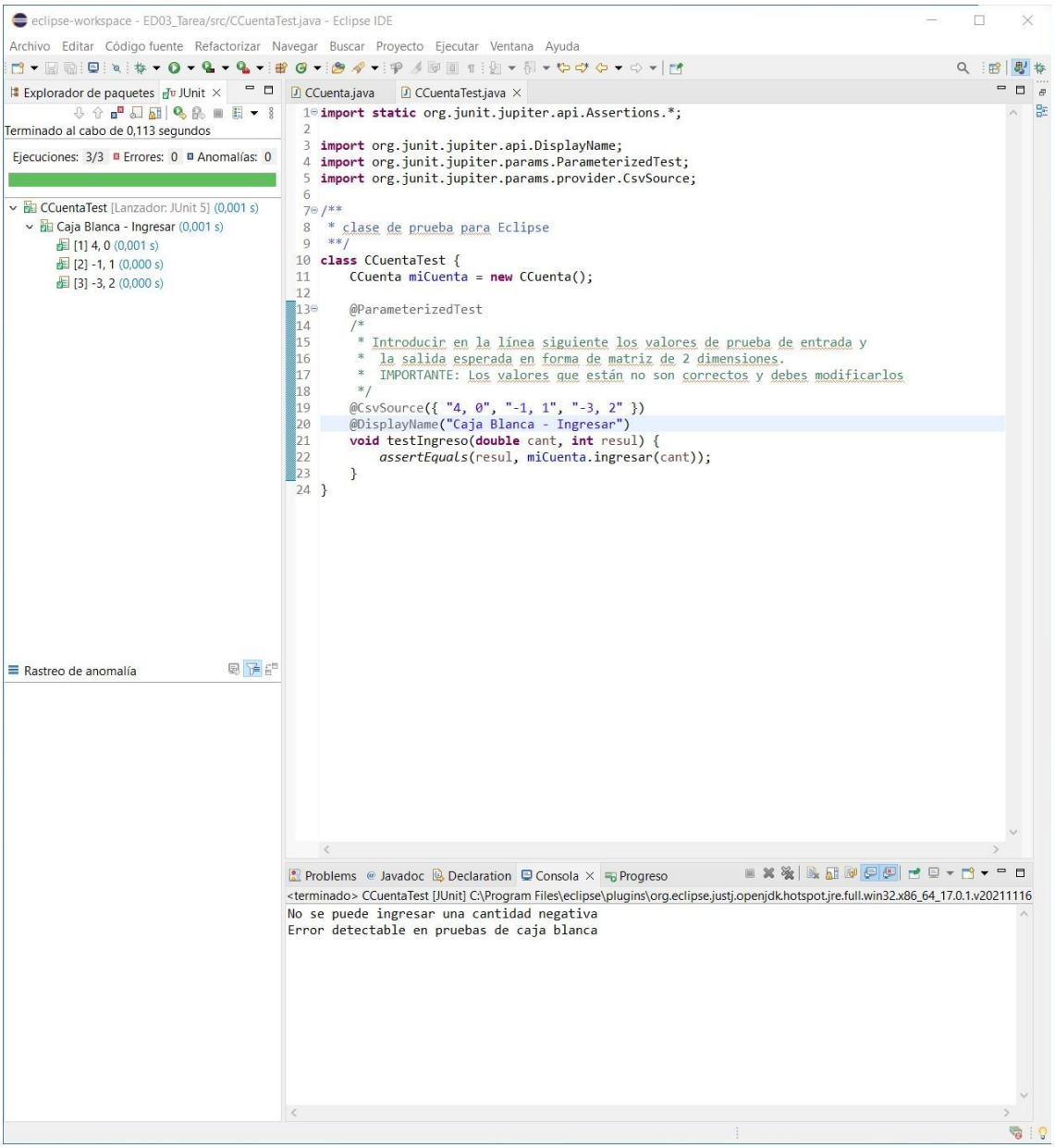
Seleccionamos Junit 5.

```java
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;

/**
 * clase de prueba para Eclipse
 **/
class CCuentaTest {
    CCuenta miCuenta = new CCuenta();

    @ParameterizedTest
    /*
     * Introducir en la línea siguiente los valores de prueba de entrada y
     *  la salida esperada en forma de matriz de 2 dimensiones.
     *  IMPORTANTE: Los valores que están no son correctos y debes modificarlos
     */
    @CsvSource({ "4, 0", "-1, 1", "-3, 2" })
    @DisplayName("Caja Blanca - Ingresar")
    void testIngreso(double cant, int resul) {
        assertEquals(resul, miCuenta.ingresar(cant));
    }
}
```

Copiamos el código fuente que se nos proporciona y modificamos los datos de los valores de prueba, de acuerdo a los calculados en el ejercicio 1.



Ejecutamos la clase de prueba pulsando con el botón derecho sobre su nombre / ejecutar como / Prueba JUnit.

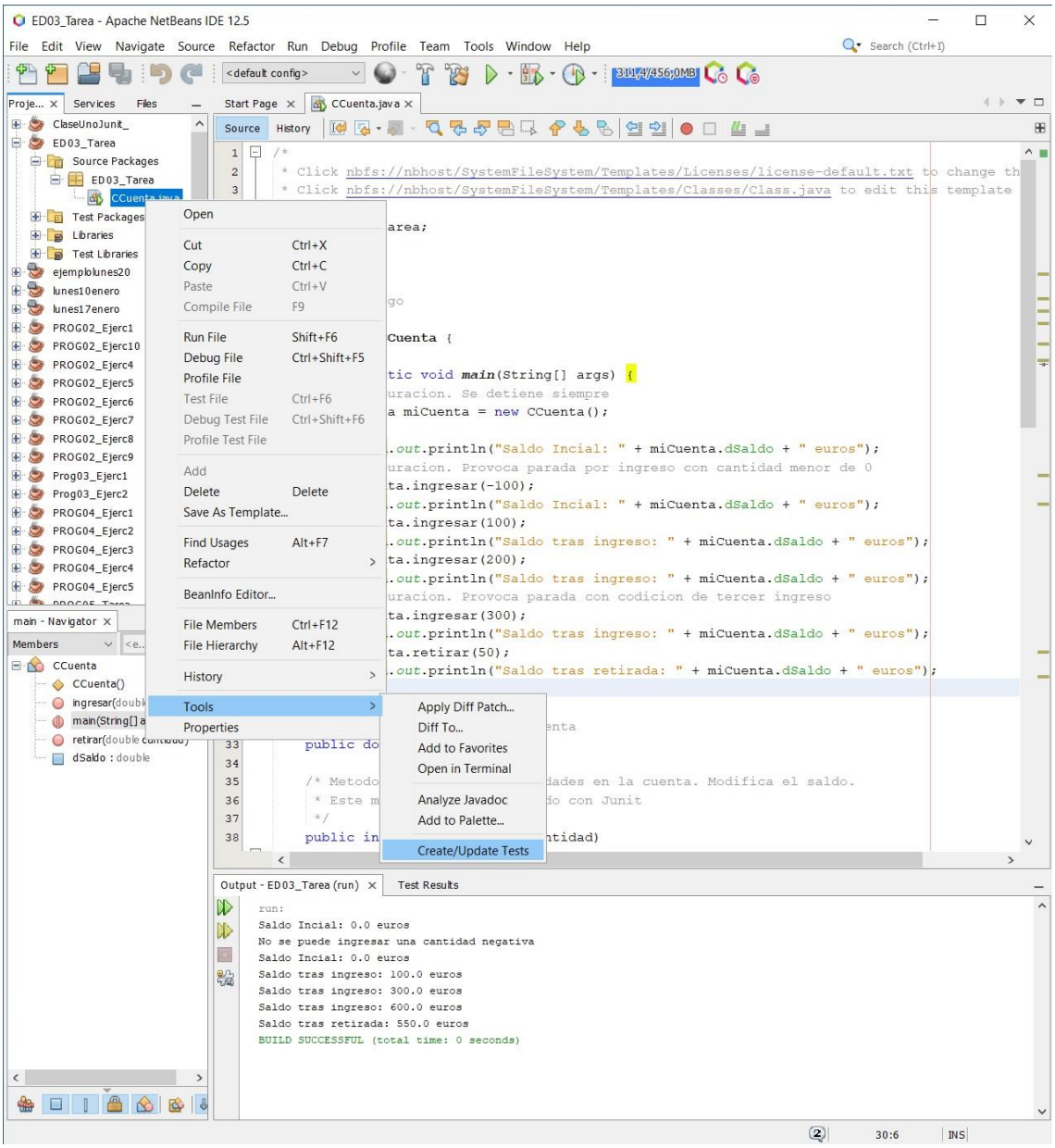


La prueba ha sido pasada sin errores.

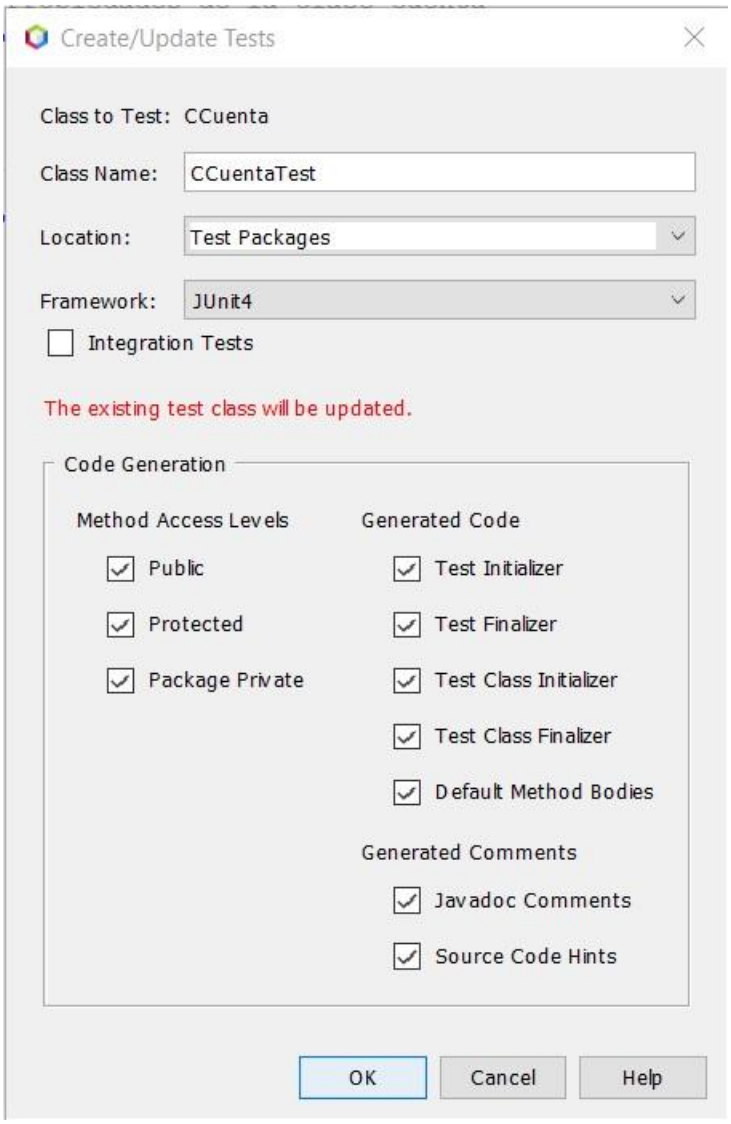
4. Crea la clase CCuentaTest del tipo **Caso de prueba JUnit** 4 en Netbeans que nos permita pasar las pruebas unitarias de caja blanca del método ingresar. Los casos de prueba ya los habrás obtenido en el primer apartado del ejercicio 1 1 y tendrás que aplicarlo en el código de la prueba. Copia el código fuente de esta clase que te proporcionamos. Será necesario quitar las librerías de Junit 5.6 y poner las de Junit 4.13.2 y Hamcrest 1.3

```
25      /* Metodo para ingresar cantidades en la cuenta. Modifica el saldo.
26       * Este metodo va a ser probado con Junit
27       */
28      public int ingresar(double cantidad)
29      {
30          int iCodErr;
31
32          if (cantidad < 0)
33          {
34              if (cantidad == -3)
35              {
36                  System.out.println("Error detectable en pruebas de caja blanca");
37                  iCodErr = 2;
38              } else
39              {
40                  System.out.println("No se puede ingresar una cantidad negativa");
41                  iCodErr = 1;
42              }
43          }
44          else
45          {
46              // Depuracion. Punto de parada. Solo en el 3 ingreso
47              dSaldo = dSaldo + cantidad;
48              iCodErr = 0;
49          }
50          // Depuracion. Punto de parada cuando la cantidad  es menor de 0
51          return iCodErr;
52      }
```

Metodo *ingresar* sobre el cual vamos a hacer las pruebas unitarias de caja blanca.



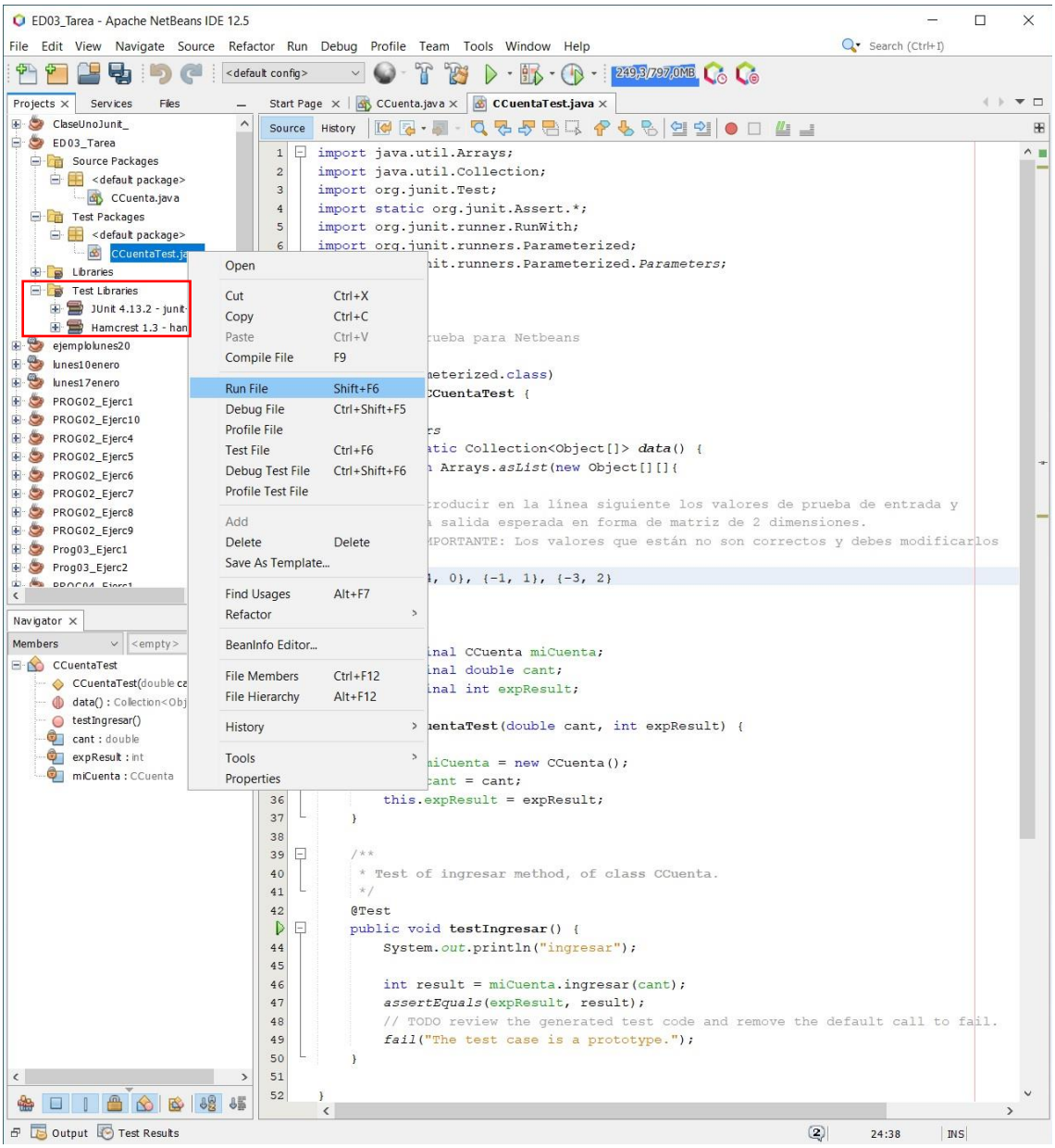
Creamos la clase de prueba del tipo *Caso de prueba JUnit 4* a través de NetBeans.



Lo llamamos *CCuentaTest* y lo creamos sobre JUnit4.

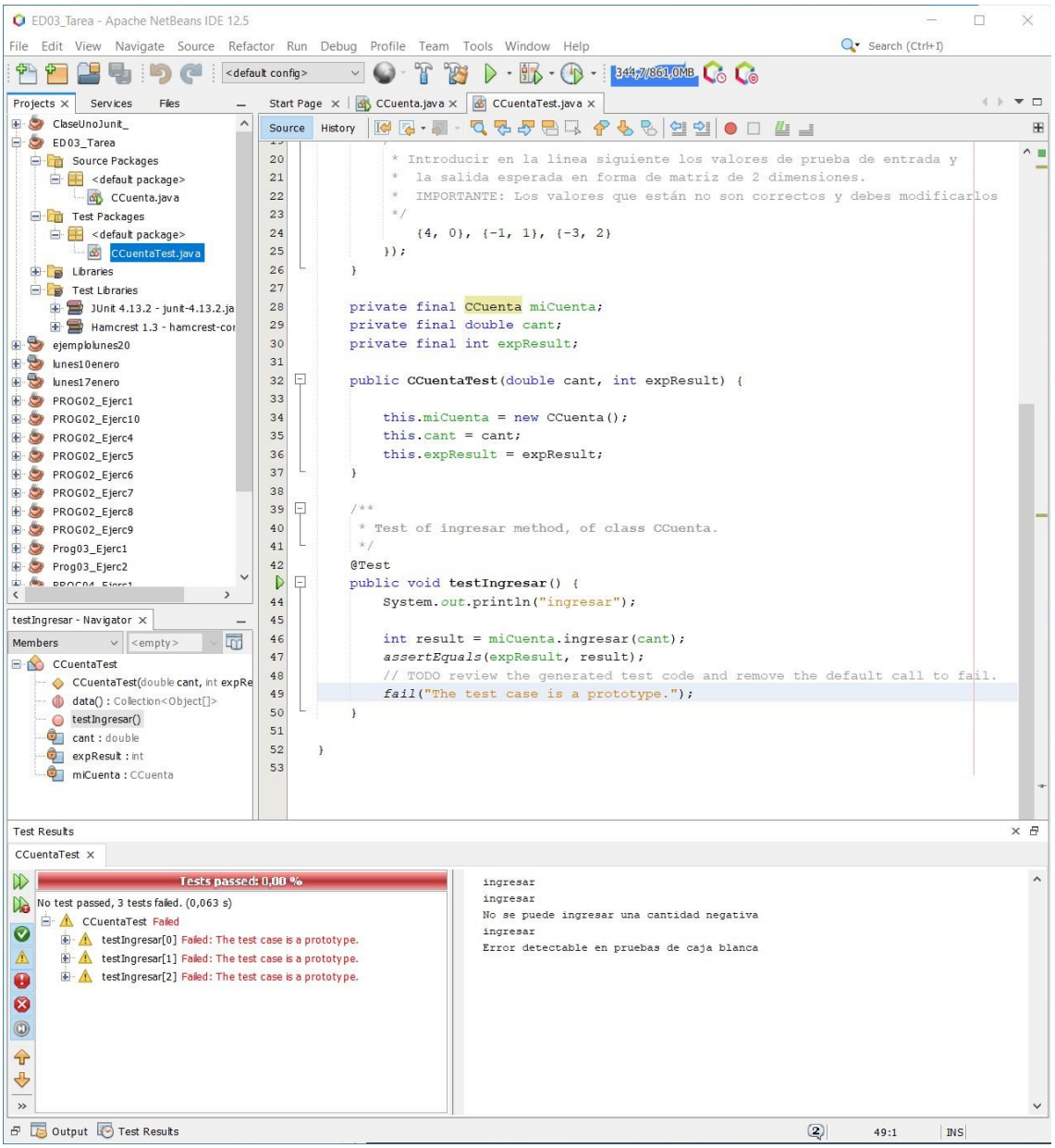
```java
import java.util.Arrays;
import java.util.Collection;
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

/**
 *
 * clase de prueba para Netbeans
 */
@RunWith(Parameterized.class)
public class CCuentaTest {

    @Parameters
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][]{
        /*
         * Introducir en la línea siguiente los valores de prueba de entrada y
         *  la salida esperada en forma de matriz de 2 dimensiones.
         *  IMPORTANTE: Los valores que están no son correctos y debes modificarlos
         */
            {4, 0}, {-1, 1}, {-3, 2}
        });
    }

    private final CCuenta miCuenta;
    private final double cant;
    private final int expResult;

    public CCuentaTest(double cant, int expResult) {

        this.miCuenta = new CCuenta();
        this.cant = cant;
        this.expResult = expResult;
    }

    /**
     * Test of ingresar method, of class CCuenta.
     */
    @Test
    public void testIngresar() {
        System.out.println("ingresar");

        int result = miCuenta.ingresar(cant);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

}
```

Copiamos el código fuente que se nos proporciona y modificamos los datos de los valores de prueba, de acuerdo a los calculados en el ejercicio 1.



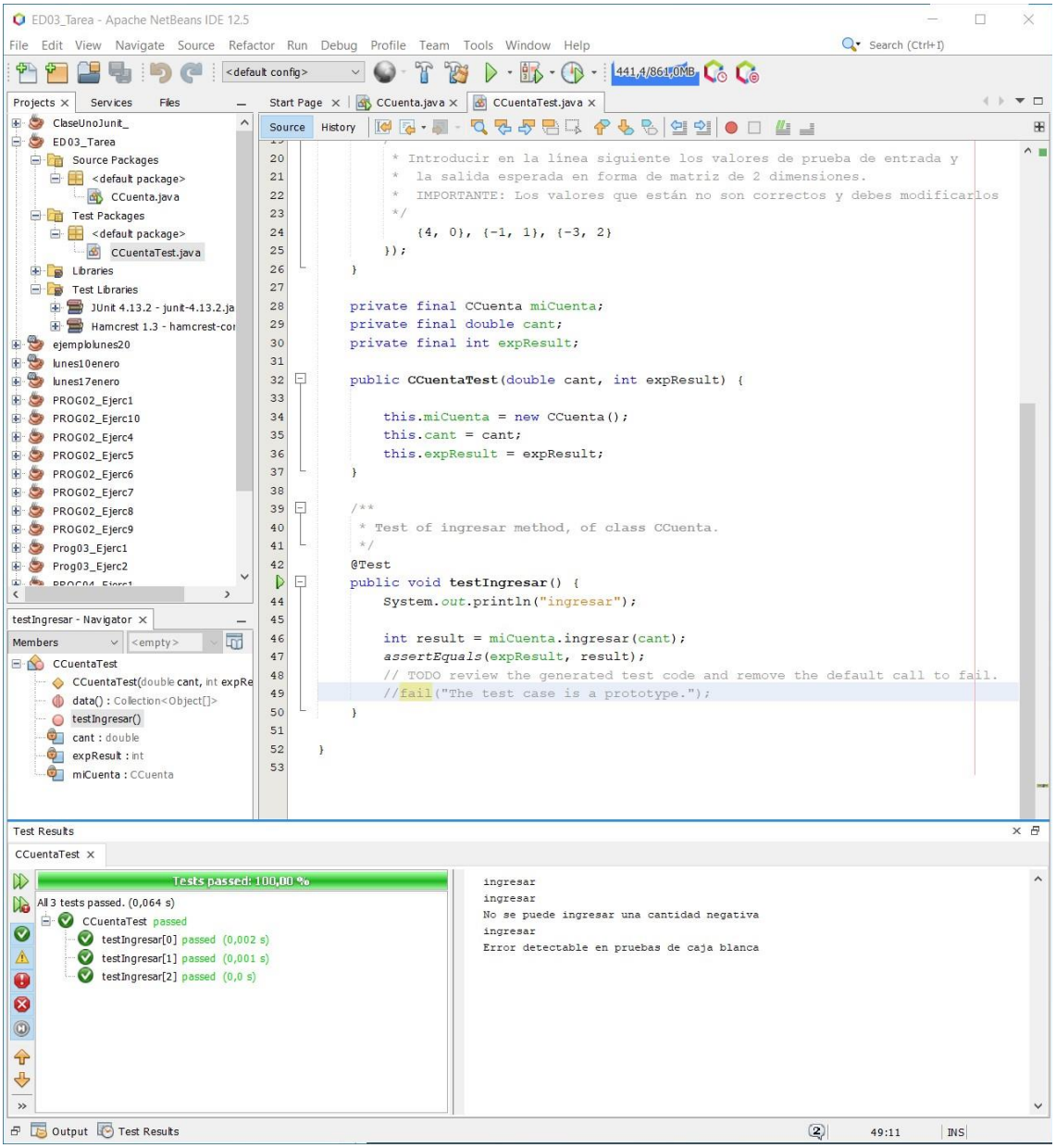
Como podemos observar, al crear la clase prueba sobre JUnit 4, ya se nos han cargado sus librerías y no las de JUnit 5.

Ejecutamos la clase de prueba pulsando con el botón derecho sobre su nombre / Run File.



Nos muestra un error y nos indica que el test ha sido pasado un 0,0 %.

En el código del test se ha quedado una instrucción de llamada predeterminada para el error *fail("…");* que se debe eliminar para que los datos se comprueben con *assertEqual* sin saltar un mensaje de error cada vez que se utiliza ese método.



Ahora sí, la prueba ha sido pasada sin errores.

5. Genera los siguientes **puntos de ruptura** para validar el comportamiento del método ingresar en modo depuración.

   A. Punto de parada sin condición al crear el objeto miCuenta en la función main.

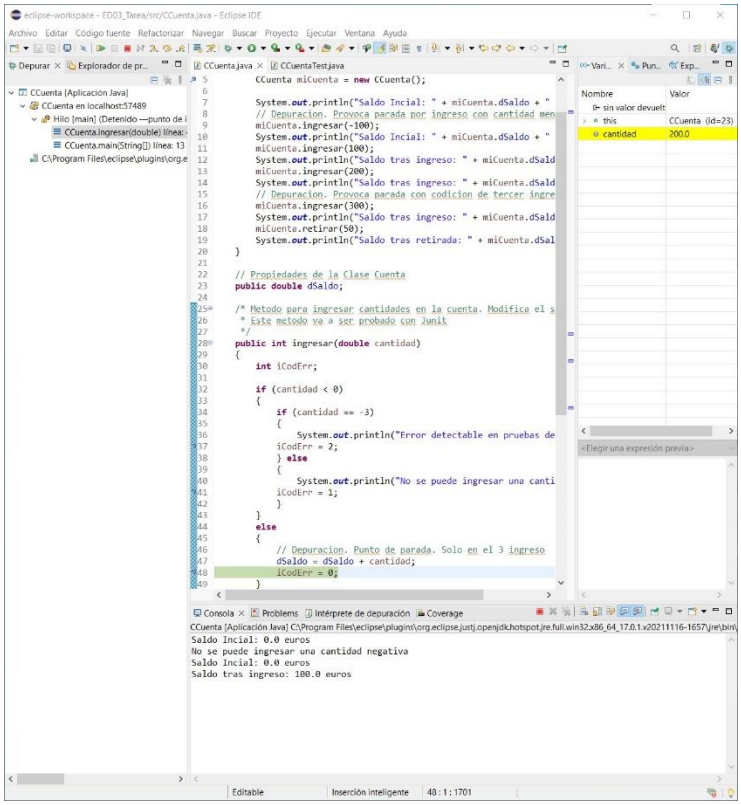
   B. Punto de parada en cada instrucción del método ingresar que devuelva un código de error (Nota importante: Se debe corregir el código para que el flujo de programa pase por estos 3 puntos de parada)

   ➢ **Puntos de ruptura en Eclipse**

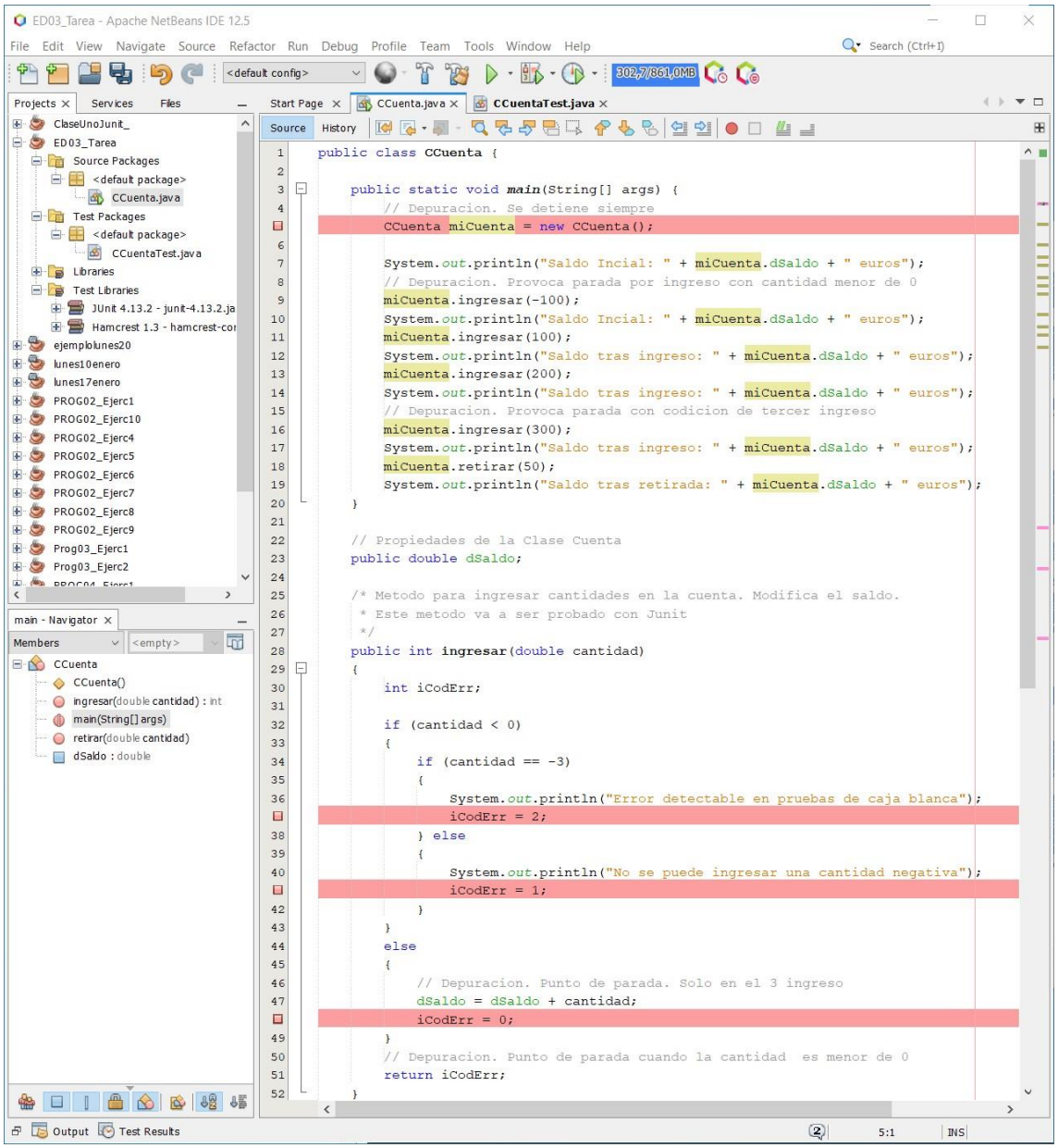
```
<?xml version="1.0" encoding="UTF-8"?>
<breakpoints>
<breakpoint enabled="true" persistant="true" registered="true">
<resource path="/ED03_Tarea/src/CCuenta.java" type="1"/>
```

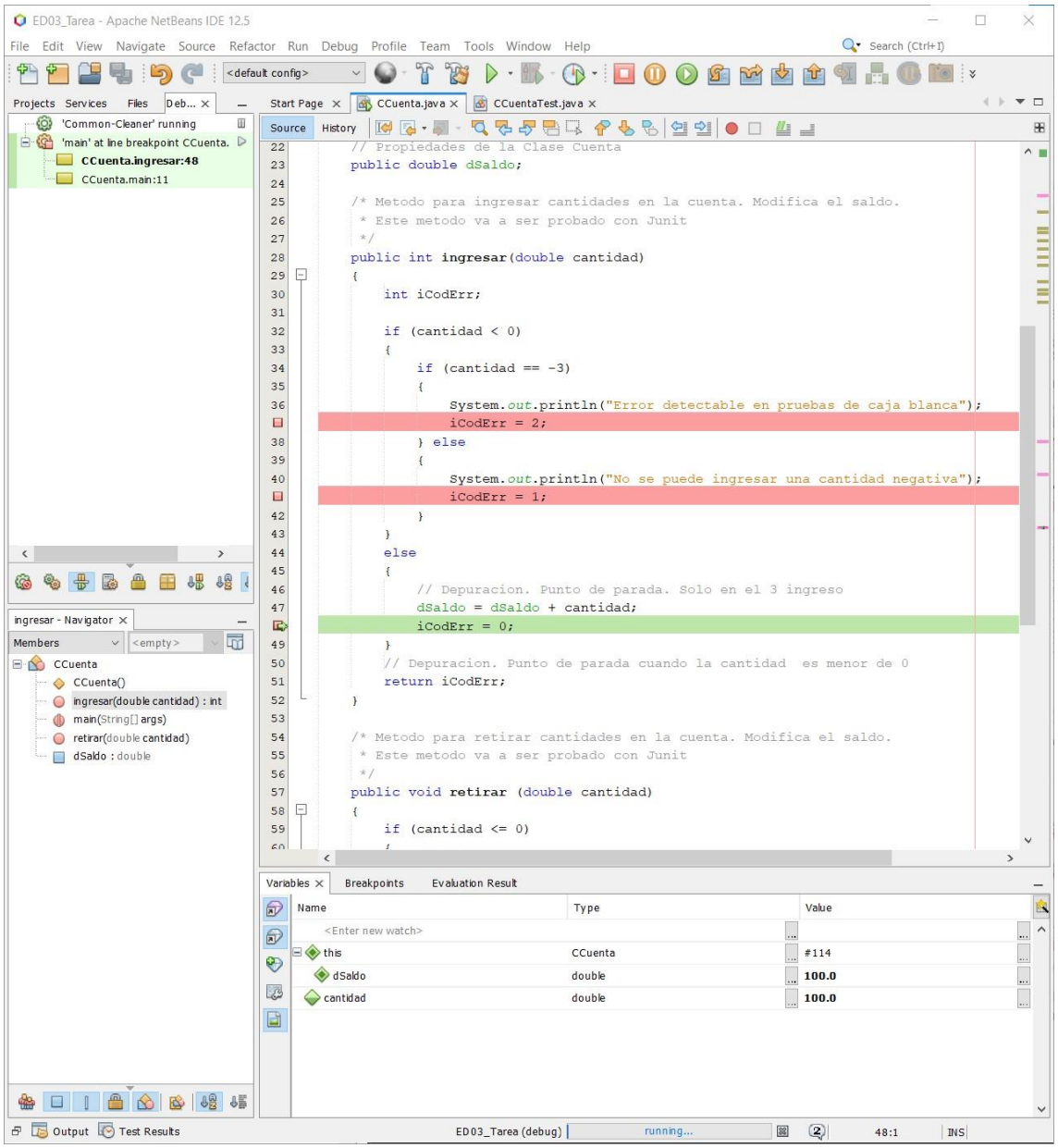
<marker charStart="113" lineNumber="5" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="113"/>
<attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="=ED03_Tarea/src<{CCuenta.java[CCuenta"/>
<attrib name="charEnd" value="151"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
<attrib name="message" value="Punto de interrupción de línea:CCuenta [línea: 5] - main(String[])"/>
<attrib name="org.eclipse.jdt.debug.core.installCount" value="0"/>
<attrib name="org.eclipse.debug.core.id" value="org.eclipse.jdt.debug"/>
<attrib name="org.eclipse.jdt.debug.core.typeName" value="CCuenta"/>
<attrib name="workingset_name" value=""/>
<attrib name="workingset_id" value="org.eclipse.debug.ui.breakpointWorkingSet"/>
</marker>
</breakpoint>
<breakpoint enabled="true" persistant="true" registered="true">
<resource path="/ED03_Tarea/src/CCuenta.java" type="1"/>
<marker charStart="1369" lineNumber="37" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="1369"/>
<attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="=ED03_Tarea/src<{CCuenta.java[CCuenta"/>
<attrib name="charEnd" value="1390"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
<attrib name="message" value="Punto de interrupción de línea:CCuenta [línea: 37] - ingresar(double)"/>
<attrib name="org.eclipse.jdt.debug.core.installCount" value="0"/>
<attrib name="org.eclipse.debug.core.id" value="org.eclipse.jdt.debug"/>
<attrib name="org.eclipse.jdt.debug.core.typeName" value="CCuenta"/>
<attrib name="workingset_name" value=""/>
<attrib name="workingset_id" value="org.eclipse.debug.ui.breakpointWorkingSet"/>
</marker>
</breakpoint>
<breakpoint enabled="true" persistant="true" registered="true">
<resource path="/ED03_Tarea/src/CCuenta.java" type="1"/>
<marker charStart="1511" lineNumber="41" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="1511"/>
<attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="=ED03_Tarea/src<{CCuenta.java[CCuenta"/>
<attrib name="charEnd" value="1532"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
<attrib name="message" value="Punto de interrupción de línea:CCuenta [línea: 41] - ingresar(double)"/>

<attrib name="org.eclipse.jdt.debug.core.installCount" value="0"/>
<attrib name="org.eclipse.debug.core.id" value="org.eclipse.jdt.debug"/>
<attrib name="org.eclipse.jdt.debug.core.typeName" value="CCuenta"/>
<attrib name="workingset_name" value=""/>
<attrib name="workingset_id" value="org.eclipse.debug.ui.breakpointWorkingSet"/>
</marker>
</breakpoint>
<breakpoint enabled="true" persistant="true" registered="true">
<resource path="/ED03_Tarea/src/CCuenta.java" type="1"/>
<marker charStart="1701" lineNumber="48" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="1701"/>
<attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="=ED03_Tarea/src<{CCuenta.java[CCuenta"/>
<attrib name="charEnd" value="1725"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
<attrib name="message" value="Punto de interrupción de línea:CCuenta [línea: 48] - ingresar(double)"/>
<attrib name="org.eclipse.jdt.debug.core.installCount" value="0"/>
<attrib name="org.eclipse.debug.core.id" value="org.eclipse.jdt.debug"/>
<attrib name="org.eclipse.jdt.debug.core.typeName" value="CCuenta"/>
<attrib name="workingset_name" value=""/>
<attrib name="workingset_id" value="org.eclipse.debug.ui.breakpointWorkingSet"/>
</marker>
</breakpoint>
</breakpoints>



- **Puntos de ruptura en NetBeans**




ED03_Tarea - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects Services Files Deb...
'Common-Cleaner' running
'main' at line breakpoint CCuenta.
CCuenta.ingresar:48
CCuenta.main:11
Start Page
CCuenta.java
CCuentaTest.java
Source History
22 // Propiedades de la Clase Cuenta
23 public double dSaldo;
24
25 /* Metodo para ingresar cantidades en la cuenta. Modifica el saldo.
26 * Este metodo va a ser probado con Junit
27 */
28 public int ingresar(double cantidad)
29 {
30 int iCodErr;
31
32 if (cantidad < 0)
33 {
34 if (cantidad == -3)
35 {
36 System.out.println("Error detectable en pruebas de caja blanca");
38 iCodErr = 2;
38 } else
39 {
40 System.out.println("No se puede ingresar una cantidad negativa");
42 iCodErr = 1;
43 }
44 }
45 else
46 {
47 // Depuracion. Punto de parada. Solo en el 3 ingreso
47 dSaldo = dSaldo + cantidad;
49 iCodErr = 0;
50 }
50 // Depuracion. Punto de parada cuando la cantidad es menor de 0
51 return iCodErr;
52 }
53
54 /* Metodo para retirar cantidades en la cuenta. Modifica el saldo.
55 * Este metodo va a ser probado con Junit
56 */
57 public void retirar (double cantidad)
58 {
59 if (cantidad <= 0)
ingresar - Navigator
Members
<empty>
CCuenta
CCuenta()
ingresar(double cantidad) : int
main(String[] args)
retirar(double cantidad)
dSaldo : double
Variables
Breakpoints
Evaluation Result
Name
Type
Value
<Enter new watch>
this
CCuenta
#114
dSaldo
double
100.0
cantidad
double
100.0
Output
Test Results
ED03_Tarea (debug)
running...
48:1
INS