

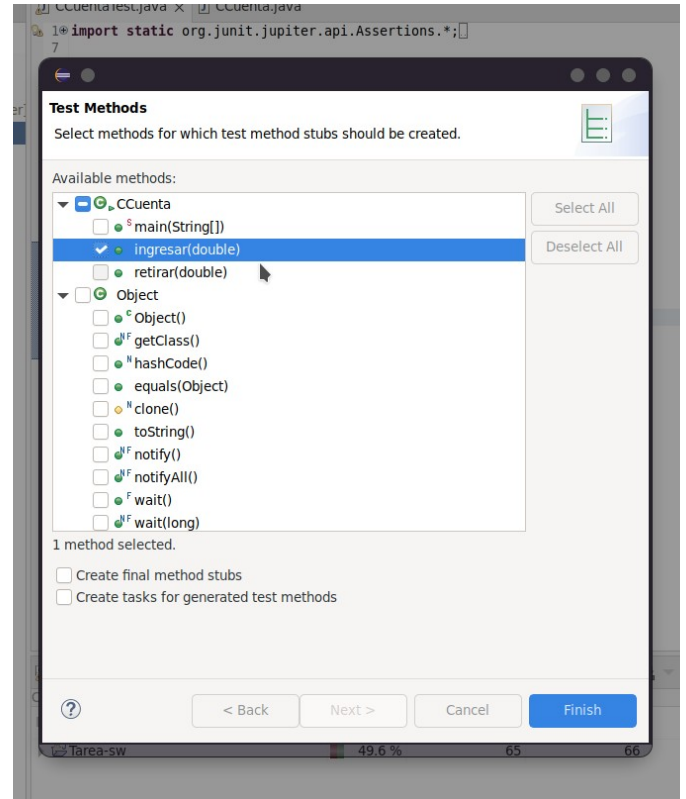
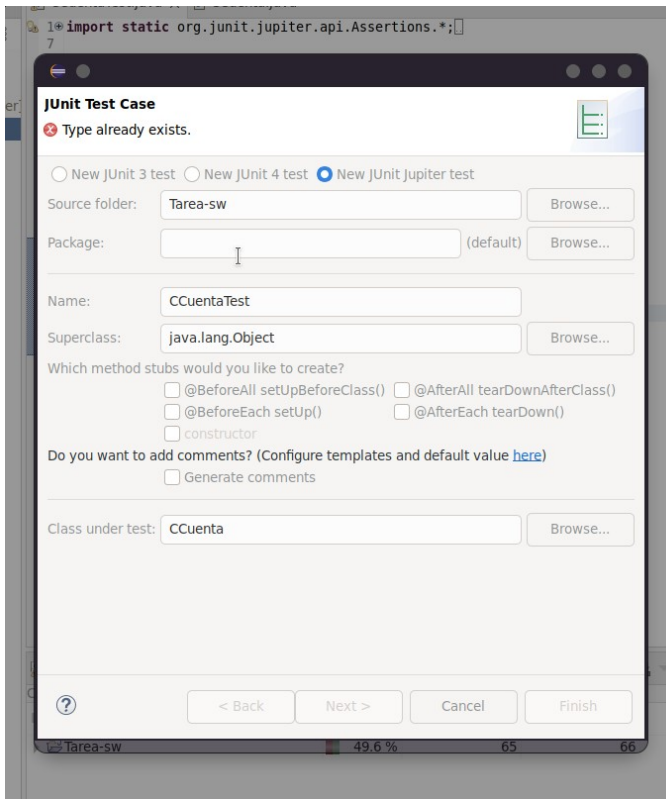
ED03

Alumno: Kevin Zamora Amela

Entornos de Desarrollo / Tarea 3: Pruebas Unitarias con Junit

1. Análisis de caja blanca (Método ingresar):

1.1. Creación de la clase CCuentaTest:



1.2. Código de partida:

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

import org.junit.jupiter.api.Test;

class CCuentaTest {

    @Test
    void testIngresar() {
        fail("Not yet implemented");
    }

}
```

ED03

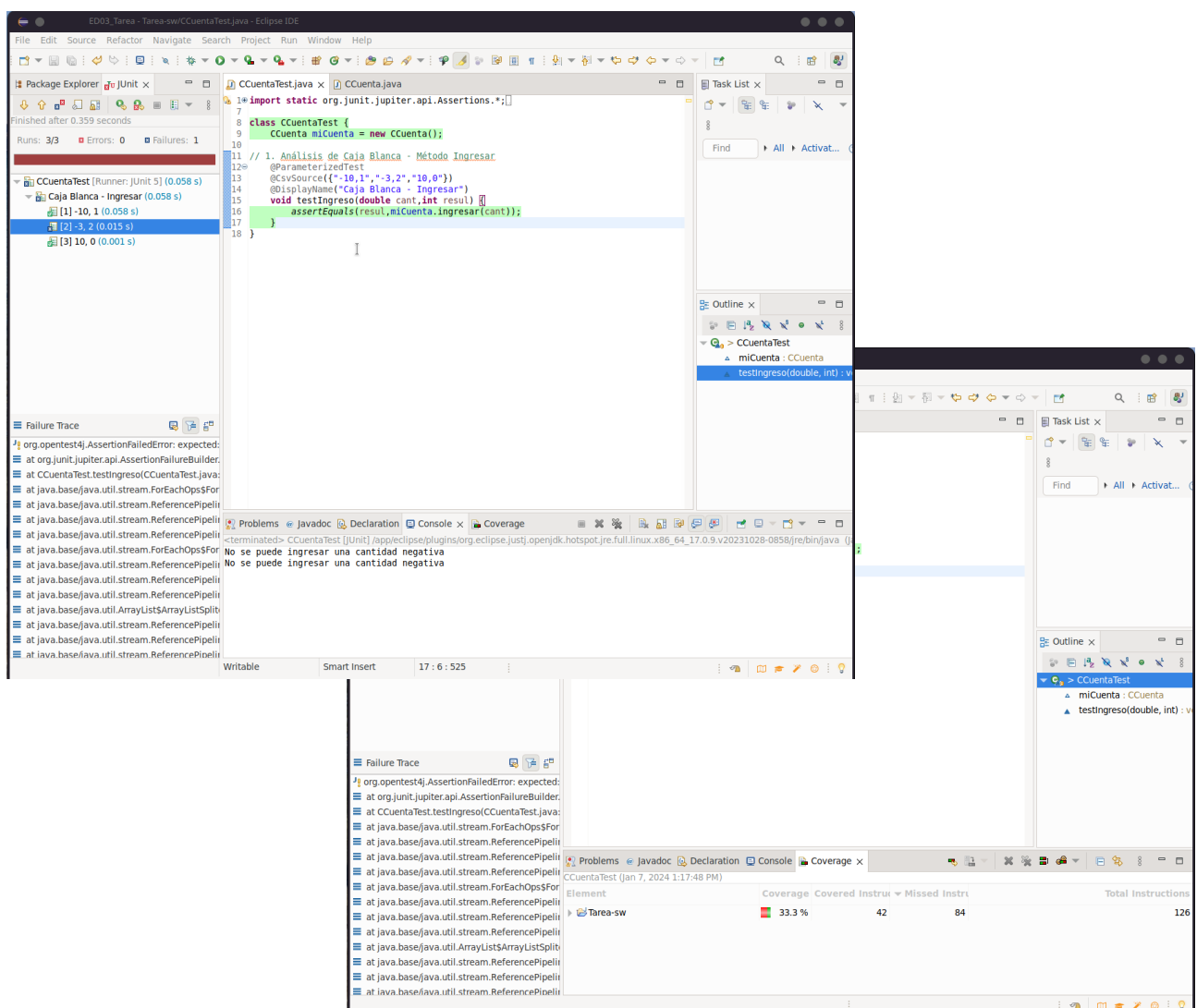
1.3. Código y Ejecución:

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

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

class CCuentaTest {
    CCuenta miCuenta = new CCuenta();

    // 1. Análisis de Caja Blanca - Método Ingresar
    @ParameterizedTest
    @CsvSource({"-10,1","-3,2","10,0"})
    @DisplayName("Caja Blanca - Ingresar")
    void testIngreso(double cant,int resul) {
        assertEquals(resul,miCuenta.ingresar(cant));
    }
}
```



ED03

2. Análisis de caja negra (Método retirar):

2.1. Código orientativo (Emulación del código de caja blanca sobre la función requerida):

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

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

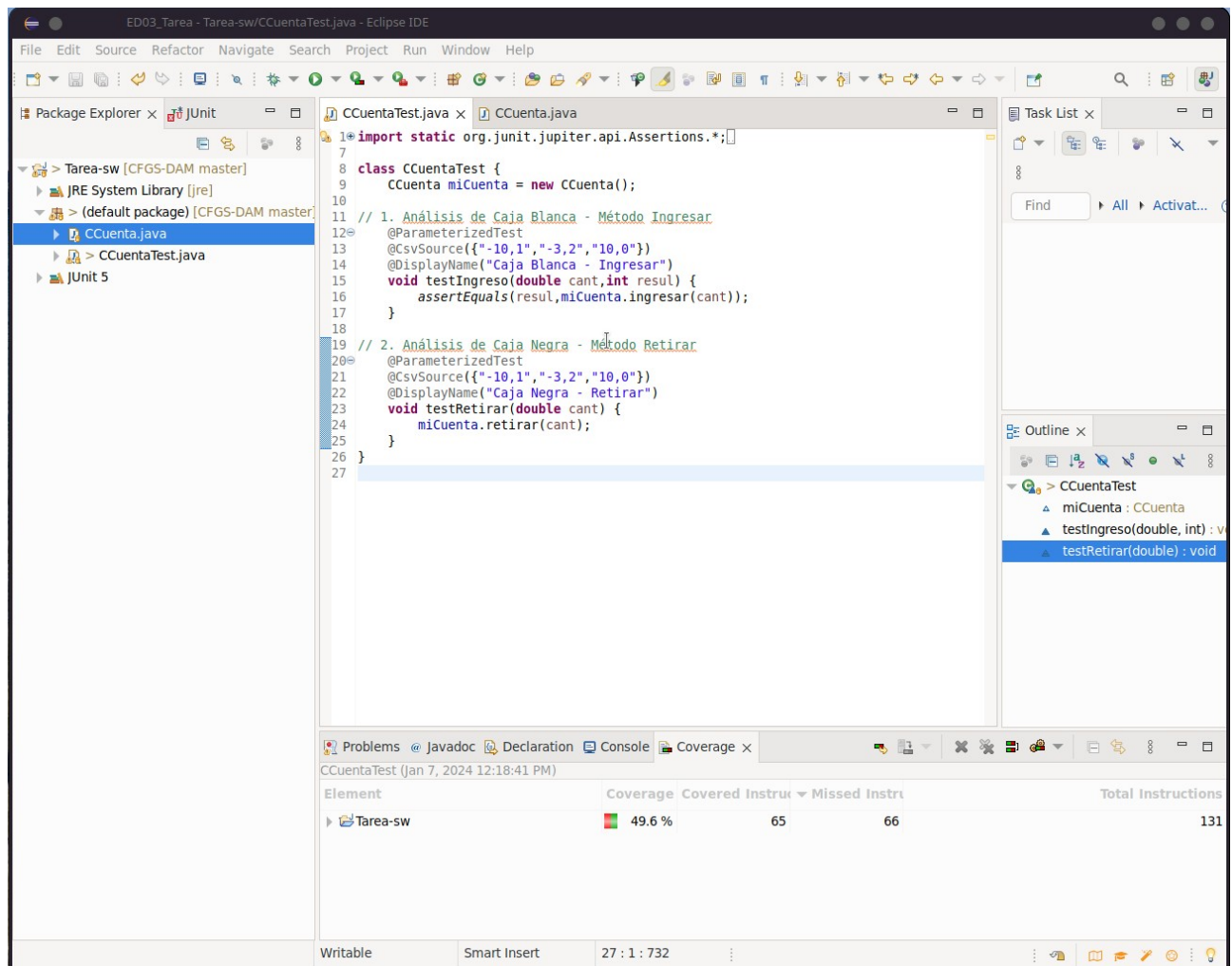
class CCuentaTest {
    CCuenta miCuenta = new CCuenta();

    // 1. Análisis de Caja Blanca - Método Ingresar (Código del primer
    ejercicio)
    @ParameterizedTest
    @CsvSource({"-10,1","-3,2","10,0"})
    @DisplayName("Caja Blanca - Ingresar")
    void testIngreso(double cant,int resul) {
        assertEquals(resul,miCuenta.ingresar(cant));
    }

    // 2. Análisis de Caja Blanca - Método Retirar
    @ParameterizedTest
    @CsvSource({"-10,1","-3,2","10,0"})
    @DisplayName("Caja Negra - Retirar")
    void testRetirar(double cant) {
        miCuenta.retirar(cant);
    }
}
```

ED03

2.2. Ejecución del código orientativo (para comprender así mejor su funcionamiento):



2.3. Análisis de caja negra:

2.3.1. Pasos de la prueba:

1. Abrimos/Ejecutamos la aplicación.
2. Introducimos la cantidad deseada.
3. Seleccionamos si queremos 'ingresar' o 'retirar' dinero.
4. El sistema ejecuta la/el función/método seleccionado y haciendo uso de la cantidad introducida, trata de realizar la función de dicho método y estando las pruebas unitarias en ejecución, pasa a verificar si la operación en cuestión se realiza o no con éxito, devolviéndonos en consecuencia un mensaje informativo, por pantalla/consola.

2.3.2. Resultado esperado:

En este caso, obviamos e ignoramos la posibilidad de que el usuario/a no tenga una cuenta. En su defecto: (Método de Ingreso) Si el importe introducido resulta correcto según los parámetros, se procederá a ingresar nuestro dinero, mostrándonos a su vez un mensaje de respuesta, cuando dicha operación ya haya sido realizada correctamente o no. Por otro lado: (Método de retirada) Si se quiere retirar dinero de nuestra cuenta, el sistema comprobará si tenemos suficiente saldo en esta como para extraer la cantidad introducida y acto seguido, nos comunicará una respuesta adecuada, en consecuencia.

ED03

2.3.3. Estado del caso de prueba: PASA (si se OBTIENE/INGRESA LA CANTIDAD DESEADA (si se superan todas las comprobaciones necesarias para realizar las tareas demandadas, actuando por parte de la entidad) / RECIBE EL CORRESPONDIENTE MENSAJE INFORMATIVO (en su defecto)).

2.3.4 Nombre del caso de prueba: PE: “ Verifique el ingreso/retirada de ‘efectivo’ en cuenta”.

2.3.5. Pasos de la prueba:

1. Abrimos/Ejecutamos la aplicación.
2. Introducimos la cantidad deseada.
3. Seleccionamos si queremos ‘ingresar’ o ‘retirar’ dinero.
4. El sistema ejecuta la/el función/método seleccionado y haciendo uso de la cantidad introducida, trata de realizar la función de dicho método y estando las pruebas unitarias en ejecución, pasa a verificar si la operación en cuestión se realiza o no con éxito, devolviéndonos en consecuencia un mensaje informativo, por pantalla/console.

2.3.6. Resultado esperado: El intento de retirada de la cantidad introducida debería fallar y aparecer en consecuencia el mensaje de error preestablecido en nuestro programa a considerar, al no disponer de suficiente saldo como para sacar ninguna cantidad de efectivo, debido a que nuestros parámetros mínimo, máximo y total presentan unos valores bastante pequeños.

2.3.7. Estado del caso de prueba: PASA (si se muestra el mensaje de error debido a la falta de saldo).

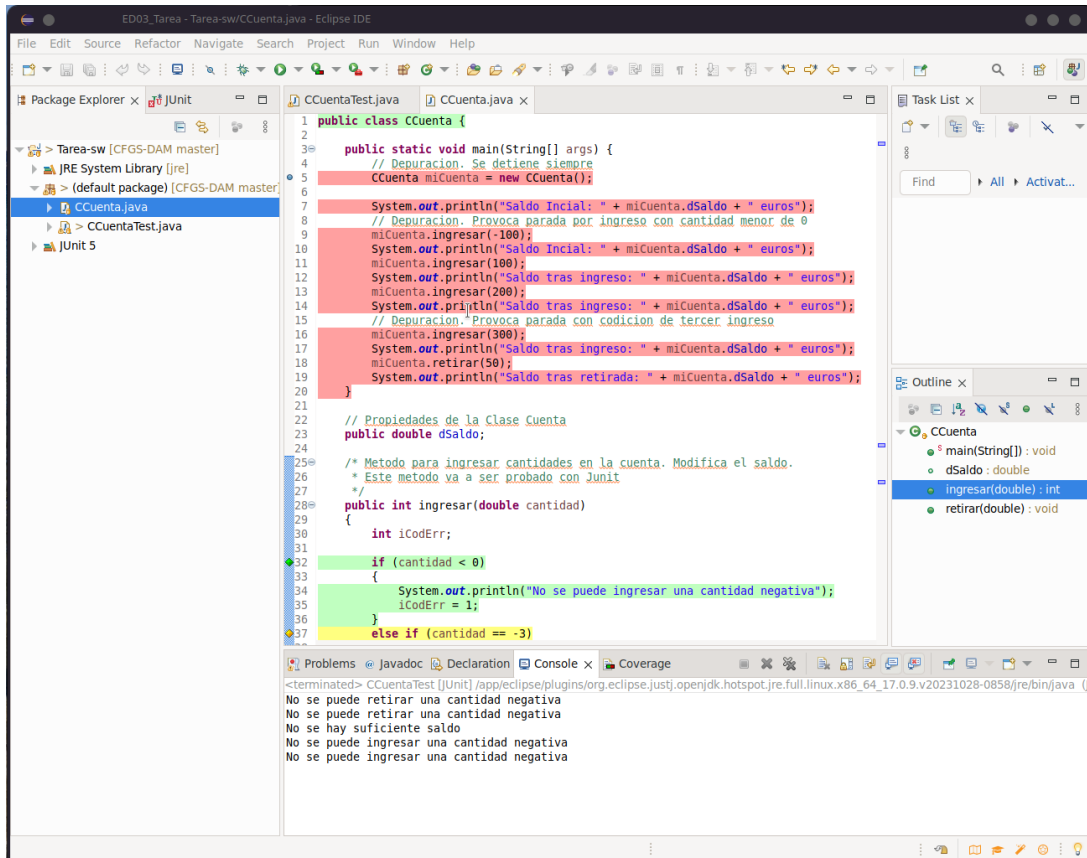
3. Crear la clase CcuentaTest:

Este paso ya ha sido realizado e ilustrado previamente durante la realización de la presente tarea, debiéndose principalmente al hecho de que para la realización y ejecución de los dos ejercicios previos, este procedimiento ya ha resultado necesario.

ED03

4. Generar puntos de ruptura:

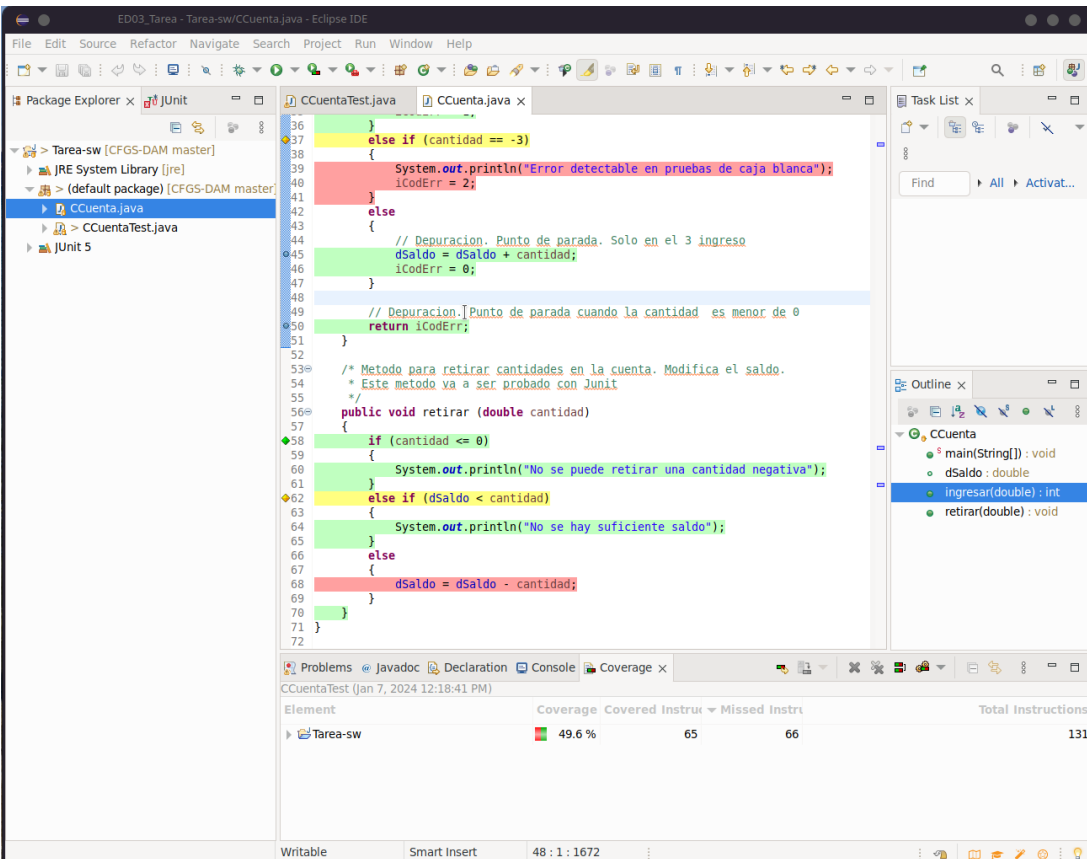
4.1. Capturas de pantalla:



```
1 public class CCuenta {
2
3     public static void main(String[] args) {
4         // Depuracion. Se detiene siempre
5         CCuenta miCuenta = new CCuenta();
6
7         System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
8         // Depuracion. Provoca parada por ingreso con cantidad menor de 0
9         miCuenta.ingresar(-100);
10        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
11        miCuenta.ingresar(100);
12        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
13        miCuenta.ingresar(200);
14        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
15        // Depuracion. Provoca parada con codicion de tercer ingreso
16        miCuenta.ingresar(300);
17        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
18        miCuenta.retirar(50);
19        System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
20    }
21
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            System.out.println("No se puede ingresar una cantidad negativa");
35            iCodErr = 1;
36        }
37        else if (cantidad == -3)
38        {
39            System.out.println("Error detectable en pruebas de caja blanca");
40            iCodErr = 2;
41        }
42        else
43        {
44            // Depuracion. Punto de parada. Solo en el 3 ingreso
45            dSaldo = dSaldo + cantidad;
46            iCodErr = 0;
47        }
48        // Depuracion. Punto de parada cuando la cantidad es menor de 0
49        return iCodErr;
50    }
51
52    /* Metodo para retirar cantidades en la cuenta. Modifica el saldo.
53     * Este metodo va a ser probado con JUnit
54     */
55    public void retirar (double cantidad)
56    {
57        if (cantidad <= 0)
58        {
59            System.out.println("No se puede retirar una cantidad negativa");
60        }
61        else if (dSaldo < cantidad)
62        {
63            System.out.println("No se hay suficiente saldo");
64        }
65        else
66        {
67            dSaldo = dSaldo - cantidad;
68        }
69    }
70 }
71
72
```

Console Output:

```
<terminated> CCuentaTest [JUnit] /app/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.linux.x86_64_17.0.9.v20231028-0858/jre/bin/java (J
No se puede retirar una cantidad negativa
No se puede retirar una cantidad negativa
No se hay suficiente saldo
No se puede ingresar una cantidad negativa
No se puede ingresar una cantidad negativa
No se puede ingresar una cantidad negativa
```



```
36 }
37 else if (cantidad == -3)
38 {
39     System.out.println("Error detectable en pruebas de caja blanca");
40     iCodErr = 2;
41 }
42 else
43 {
44     // Depuracion. Punto de parada. Solo en el 3 ingreso
45     dSaldo = dSaldo + cantidad;
46     iCodErr = 0;
47 }
48 // Depuracion. Punto de parada cuando la cantidad es menor de 0
49 return iCodErr;
50 }
51
52 /* Metodo para retirar cantidades en la cuenta. Modifica el saldo.
53 * Este metodo va a ser probado con JUnit
54 */
55 public void retirar (double cantidad)
56 {
57     if (cantidad <= 0)
58     {
59         System.out.println("No se puede retirar una cantidad negativa");
60     }
61     else if (dSaldo < cantidad)
62     {
63         System.out.println("No se hay suficiente saldo");
64     }
65     else
66     {
67         dSaldo = dSaldo - cantidad;
68     }
69 }
70 }
71
72
```

Console Output:

```
CCuentaTest (Jan 7, 2024 12:18:41 PM)
Element Coverage Covered Instru Missed Instr Total Instructions
Tarea-sw 49.6 % 65 66 131
```

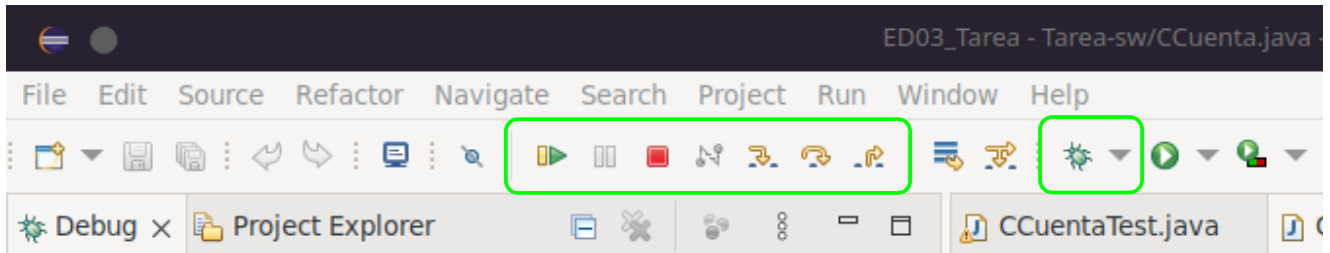
4.2. Exportación de puntos de ruptura:

```
<?xml version="1.0" encoding="UTF-8"?>
<breakpoints>
  <breakpoint enabled="true" persistent="true" registered="true">
    <resource path="/Tarea-sw/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="/Tarea-sw/&lt;
{CCuenta.java[CCuenta]"/>
      <attrib name="charEnd" value="151"/>
      <attrib name="org.eclipse.debug.core.enabled" value="true"/>
      <attrib name="message" value="Line breakpoint:CCuenta [line: 5] - main(String[])/>
      <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" persistent="true" registered="true">
    <resource path="/Tarea-sw/CCuenta.java" type="1"/>
    <marker charStart="1600" lineNumber="45"
type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
      <attrib name="charStart" value="1600"/>
      <attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
      <attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="/Tarea-sw/&lt;
{CCuenta.java[CCuenta]"/>
      <attrib name="charEnd" value="1636"/>
      <attrib name="org.eclipse.debug.core.enabled" value="true"/>
      <attrib name="message" value="Line breakpoint:CCuenta [line: 45] - ingresar(double)/>
      <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" persistent="true" registered="true">
    <resource path="/Tarea-sw/CCuenta.java" type="1"/>
    <marker charStart="1761" lineNumber="50"
type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
      <attrib name="charStart" value="1761"/>
      <attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
      <attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="/Tarea-sw/&lt;
{CCuenta.java[CCuenta]"/>
      <attrib name="charEnd" value="1784"/>
      <attrib name="org.eclipse.debug.core.enabled" value="true"/>
      <attrib name="message" value="Line breakpoint:CCuenta [line: 50] - ingresar(double)/>
      <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>
```

ED03

4.3. Depuración:

Principales opciones durante la depuración (debug):



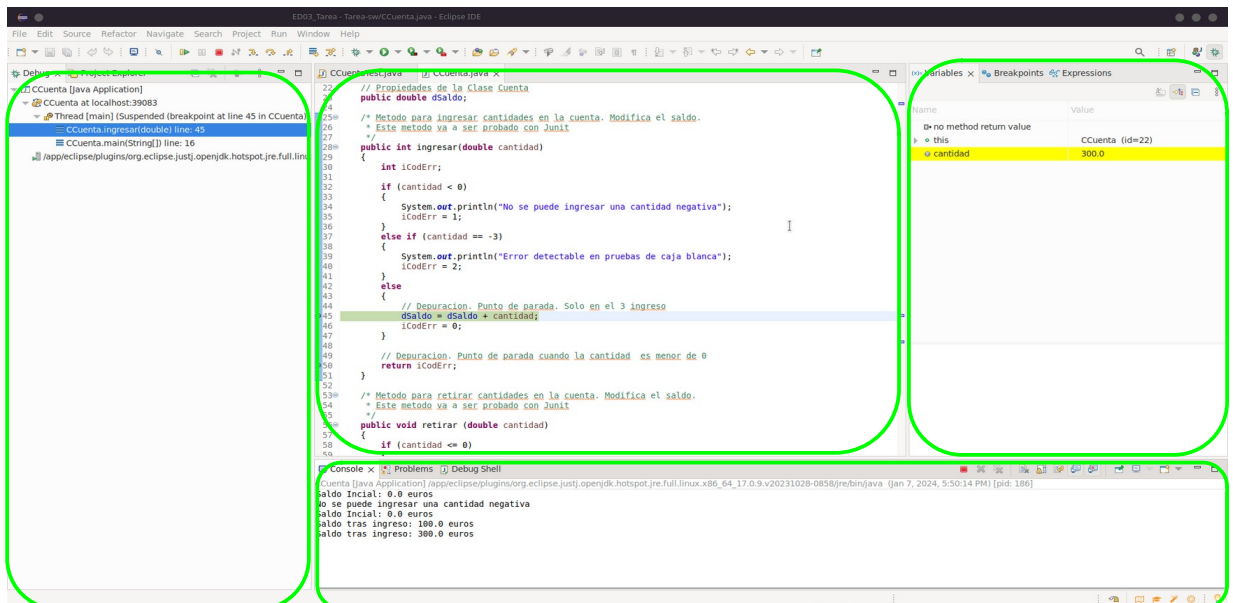
Botón de la derecha: Iniciar el proceso de depuración.

Opciones de la izquierda (Aparecen tras iniciar la ejecución):

Continuar, Pausa, Detención, Desconectar, Entrar, Saltar

Ejecución del proceso de depuración:

Durante el proceso de depuración, la ventana de nuestro IDE se encuentra dividida en una serie de secciones funcionales, las cuales son las siguientes:



- Como hemos visto en la captura anterior, en la 'segunda barra de opciones' nos aparecen todas las funciones disponibles durante esta forma de ejecutar y poner a prueba nuestra aplicación. En el interior de un cuadrado verde se encuentran las principales opciones básicas, para poder controlar la ejecución y movernos a través del programa en cuestión.
- En la sección ubicada a nuestra izquierda, encontramos el proceso o procesos en ejecución, así como también su estado: si se encuentra aún funcionando o si ya ha terminado y sobre qué puerto del sistema está 'corriendo', entre otros posibles datos de interés.

ED03

- En la sección central, se nos muestra el programa en cuestión, sobre el cual se nos irá mostrando el recorrido que va trazando dicho programa, durante su ejecución.
- En la sección derecha, en la pestaña seleccionada por defecto, nos van apareciendo los valores de cada variable y cómo estos van cambiando a lo largo de la ejecución. En la pestaña de su derecha, encontramos los diferentes puntos de ruptura/interrupción que hemos definido, pudiendo activarlos y desactivarlos a nuestro antojo.
- En la sección inferior se nos muestran la consola, los errores y la consola de depuración, mediante las cuales podemos ir controlando cómo evoluciona nuestro programa, así como también, cuál resulta su respuesta en cada momento.

PD: Puede que me haya dejado alguna sección más que pueda resultar de interés, aunque las comentadas representan a las más usadas e 'importantes'.

Anexo: Más capturas durante la ejecución:

