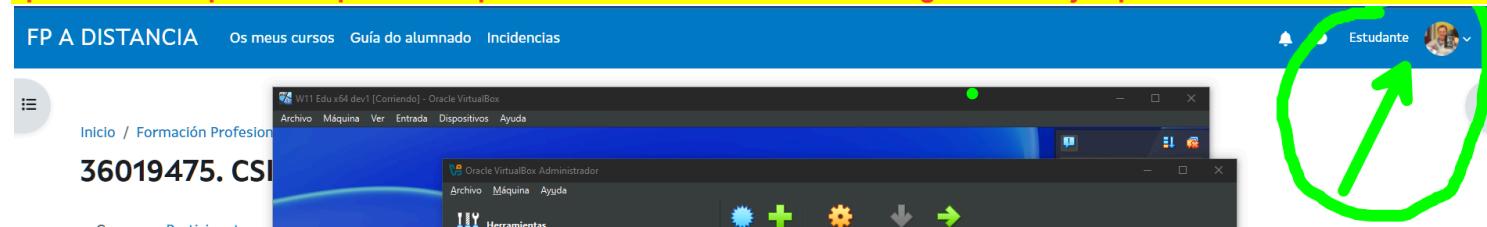


**AVISO:** Las tareas son trabajo individual y no deben ser compartidas con otros compañer@s. Si se detecta o se sospecha que una tarea es copia total o parcial de otra, serán anuladas las 2 tareas, independientemente de quién realizara la versión original. La nota de ambas tareas será de 0 puntos

En los apartados en los que es necesario entregar las capturas de pantalla, éstas deben tener como fondo de pantalla la plataforma de fpdistancia con tu usuario mostrando claramente la foto de tu perfil. Aquellos apartados/subapartados que no cumplan esta condición no serán corregidos. Por ejemplo:



## Actividad:

En esta tarea se considera una clase **Java CCuenta** (ver fichero .zip adjunto) 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:

**Los pasos 3 y 4 siguientes de la tarea se realizarán 2 veces:**

- 1) usando el entorno de desarrollo Eclipse
- 2) usando el entorno de desarrollo NetBeans.

y deberás realizar un documento donde dar respuesta a los siguientes apartados:

1. Realiza un **análisis de caja blanca** completo del método ingresar.
  2. Realiza un **análisis de caja negra**, incluyendo valores límite y conjectura 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.
  3. Crea la clase CCuentaTest del tipo **Caso de prueba JUnit** 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. Copia el código fuente de esta clase en el documento.
  4. Genera los siguientes **puntos de ruptura** para validar el comportamiento del método ingresar en modo depuración.
    - Punto de parada sin condición al crear el objeto miCuenta en la función main.
    - Punto de parada en la instrucción return del método ingresar sólo si la cantidad a ingresar es menor de 0.
    - Línea 20 del código del método ingresar que se presenta más adelante.
    - Punto de parada en la instrucción donde se actualiza el saldo, sólo deberá parar la tercera vez que sea actualizado. Línea 16 del código del método ingresar que se presenta más adelante.
- Exportar los puntos de ruptura creados: selecciona los tres puntos de ruptura generados y guardas el fichero. El fichero tendrá la extensión bkpt, la cambias por txt. Ahora abres el fichero y copias el contenido íntegramente al documento.

RESPUESTAS:

## 1. Análisis de caja blanca del método ingresar

Flujo del código:

- Si cantidad == -3:
  - Error en el código: La condición está después de cantidad < 0, por lo que nunca se ejecuta.
  - Resultado esperado (incorrecto): Imprime mensaje específico y retorna código 2.
  - Resultado real: Retorna código 1 (se detecta como cantidad < 0).
- Si cantidad < 0 (y cantidad ≠ -3):
  - Imprime "No se puede ingresar una cantidad negativa" y retorna código 1.
- Si cantidad >= 0:
  - Actualiza el saldo (dSaldo += cantidad) y retorna código 0.

Cobertura de caminos:

- Camino 1: cantidad = -5 → Código 1 (correcto).
- Camino 2: cantidad = -3 → Falla (retorna 1 en lugar de 2).
- Camino 3: cantidad = 100 → Código 0 (correcto).

Es necesario corregir el orden de las condiciones:

```
if (cantidad == -3) { ... }
else if (cantidad < 0) { ... }
```

Modificar pruebas JUnit para reflejar el error actual (-3,1 en lugar de -3,2).

## 2. Análisis de caja negra del método retirar.

- Valores límite de entrada (cantidad):
  - Límite inferior: 0 (inválido, retorna error).
  - Límite superior: saldo\_actual (inválido si cantidad > saldo\_actual).
  - Valores válidos:  $0.01 \leq \text{cantidad} \leq \text{saldo\_actual}$ .
- Casos de prueba críticos:

Cantidad	Saldo Inicial	Resultado Esperado
-50	200	Error: "No se puede retirar negativa"
0	200	Error: "No se puede retirar negativa"
50	100	Saldo final: 50
100	100	Saldo final: 0
150	100	Error: "No hay suficiente saldo"

- Conjetura de errores:

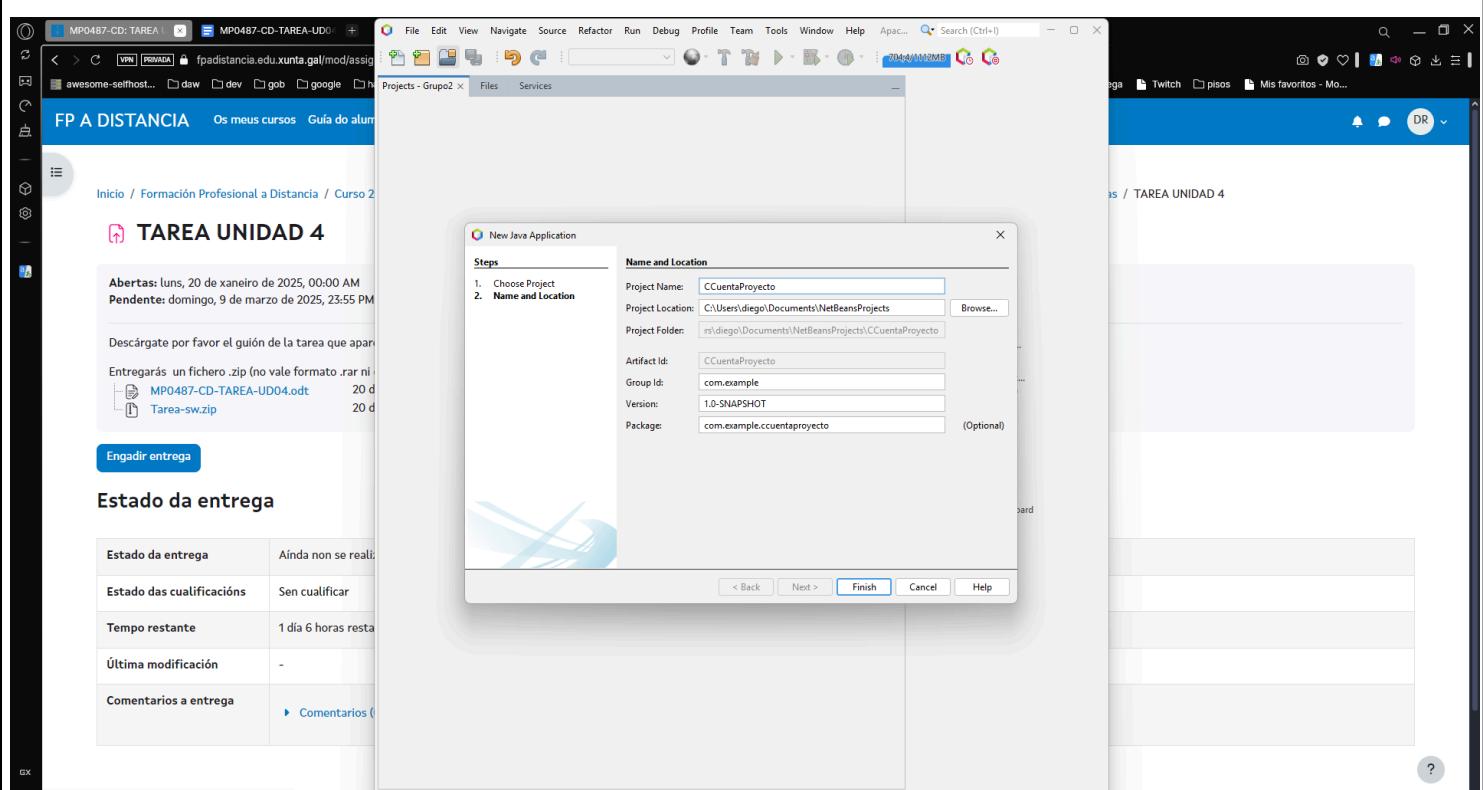
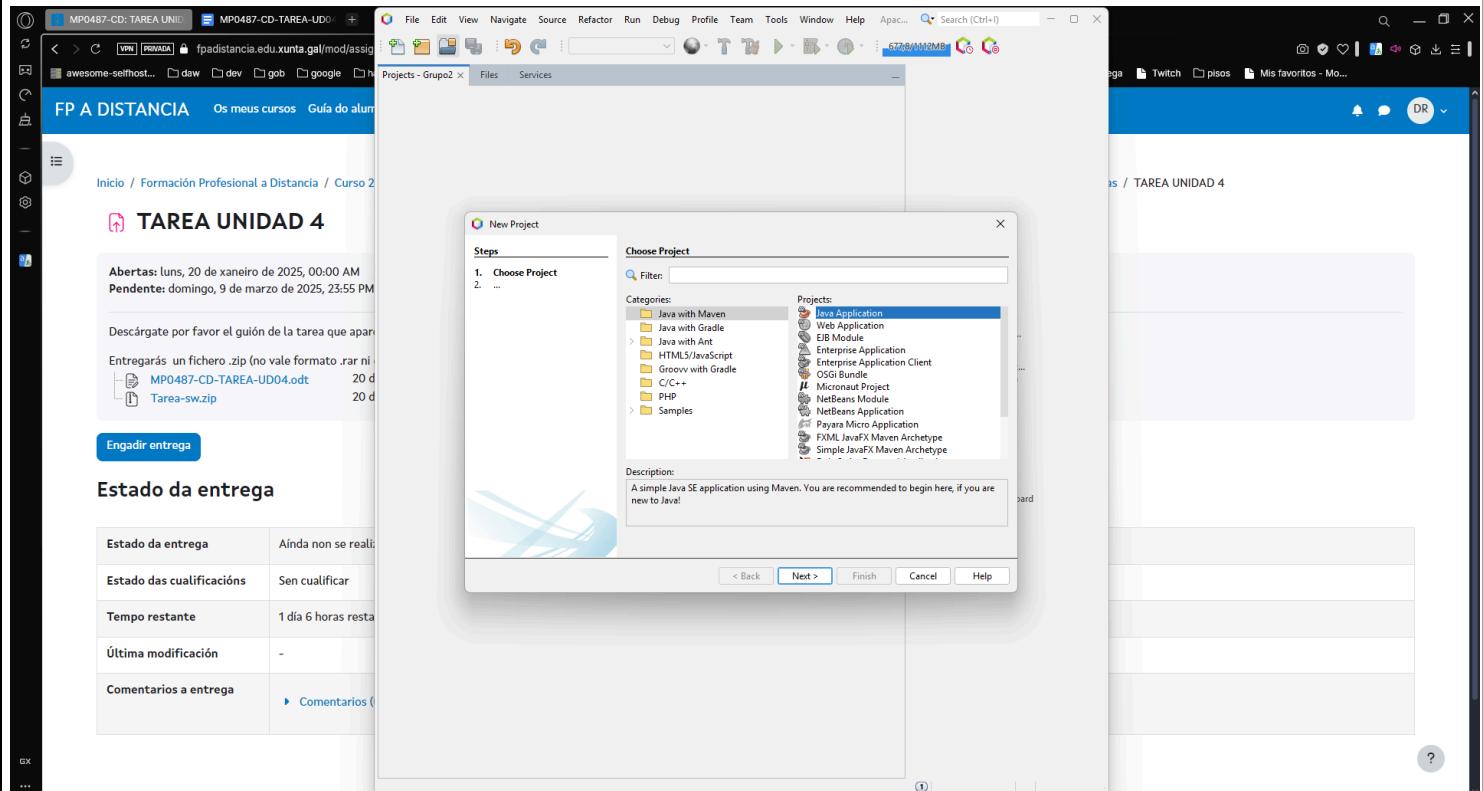
- Caso 1: cantidad = 0.01 con saldo = 0.01 → Éxito (saldo = 0).
- Caso 2: cantidad = saldo\_actual + 0.01 → Error de saldo insuficiente.
- Caso 3: cantidad = "abc" → Error en tiempo de ejecución (no manejado).

## PASOS 3 y 4: SOLUCIÓN EN NETBEANS

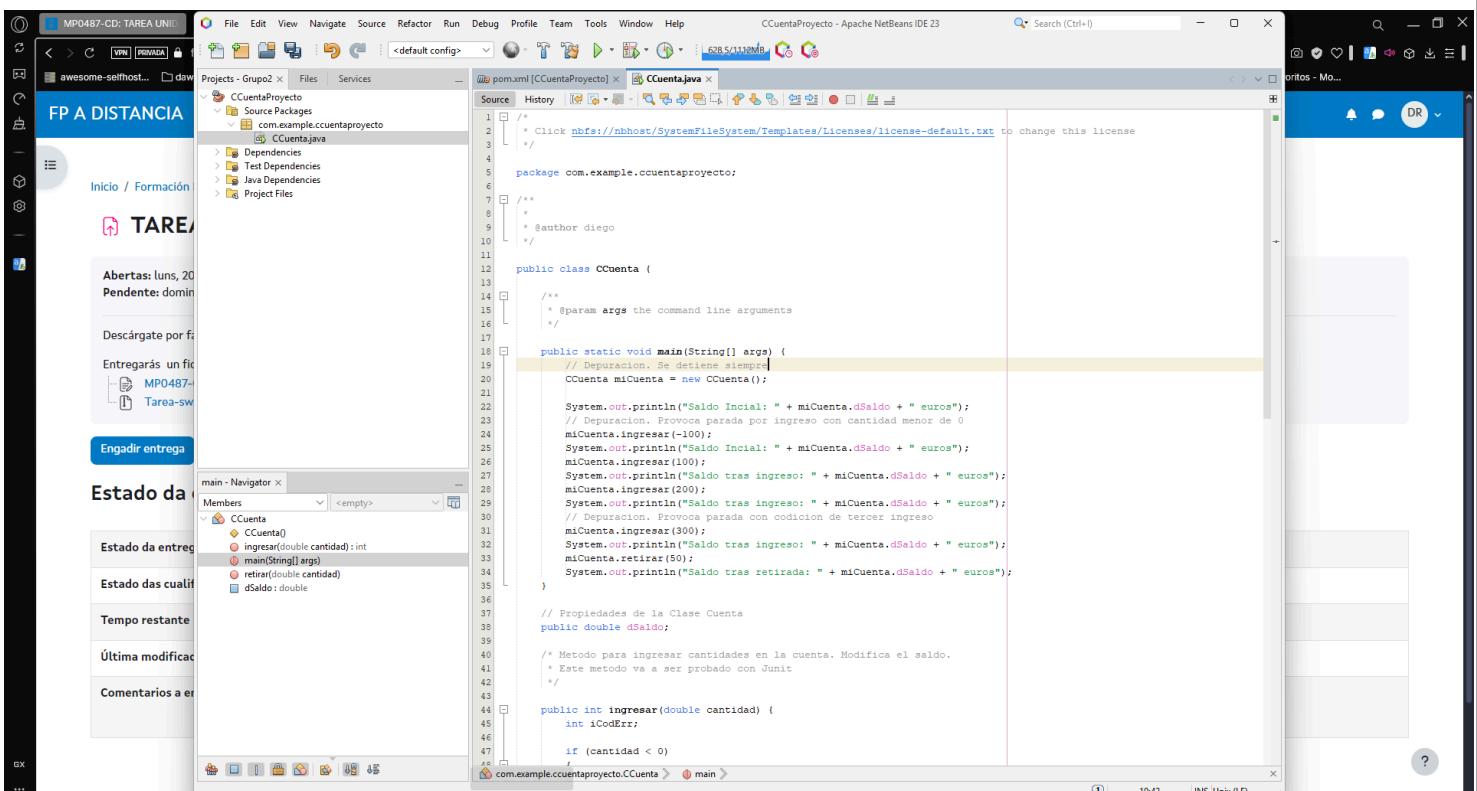
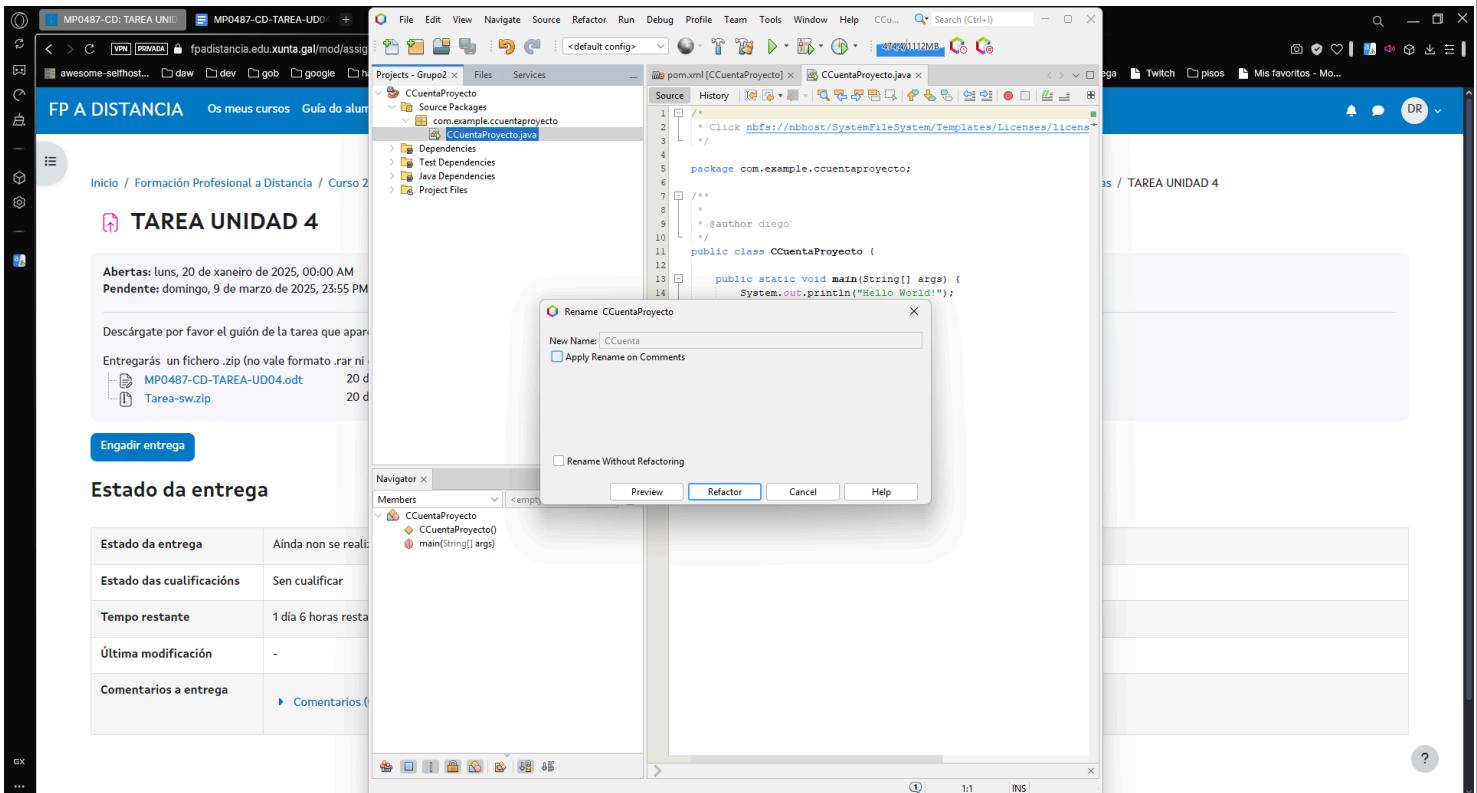
### 3. Crea la clase CCuentaTest del tipo Caso de prueba JUnit

Capturas de pantalla ilustrando el proceso detallado

Lo primero es crear un proyecto Maven en Netbeans.



Renombrar la clase principal CCuentaProyecto.java a CCuentaProyecto y copiar el código aportado en el enunciado.



```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */
package com.example.ccuentaproyecto;

/**
 *
 * @author diego
 */

public class CCuenta {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        // Depuracion. Se detiene siempre
        CCuenta miCuenta = new CCuenta();

        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
        // Depuracion. Provoca parada por ingreso con cantidad menor de 0
        miCuenta.ingresar(-100);
        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
        miCuenta.ingresar(100);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        miCuenta.ingresar(200);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        // Depuracion. Provoca parada con codicion de tercer ingreso
        miCuenta.ingresar(300);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        miCuenta.retirar(50);
        System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
    }

    // Propiedades de la Clase Cuenta
    public double dSaldo;

    /* 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;

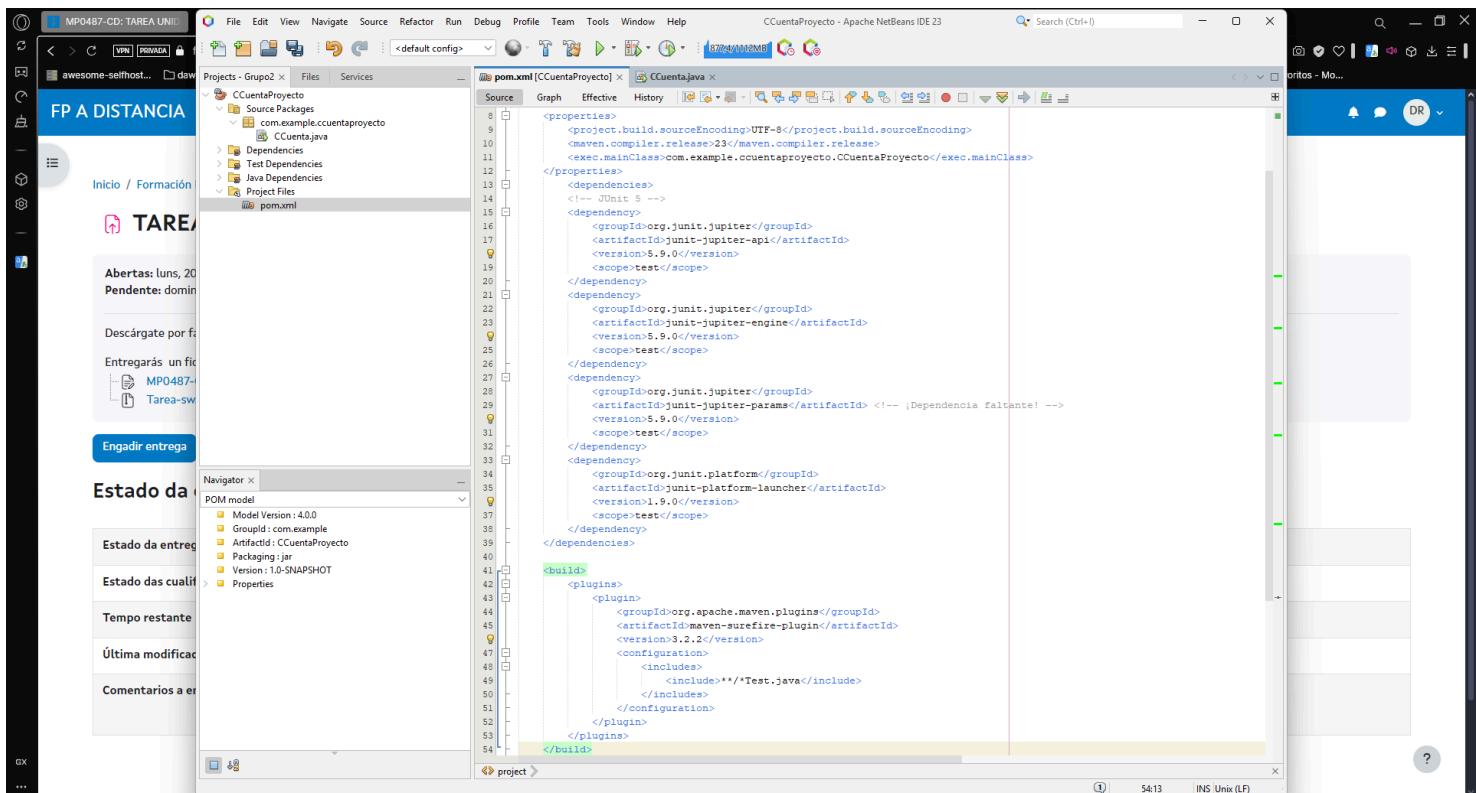
        if (cantidad < 0)
        {
            System.out.println("No se puede ingresar una cantidad negativa");
            iCodErr = 1;
        }
        else if (cantidad == -3)
        {
            System.out.println("Error detectable en pruebas de caja blanca");
            iCodErr = 2;
        }
        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;  
}  
  
/* Metodo para retirar cantidades en la cuenta. Modifica el saldo.  
 * Este metodo va a ser probado con Junit  
 */  
public void retirar (double cantidad) {  
    if (cantidad <= 0) {  
        System.out.println("No se puede retirar una cantidad negativa");  
    }  
    else if (dSaldo < cantidad) {  
        System.out.println("No se hay suficiente saldo");  
    }  
    else {  
        dSaldo = dSaldo - cantidad;  
    }  
}  
}
```

Modificar el archivo pom.xml añadiendo lo siguiente para utilizar la librería JUnit.



```

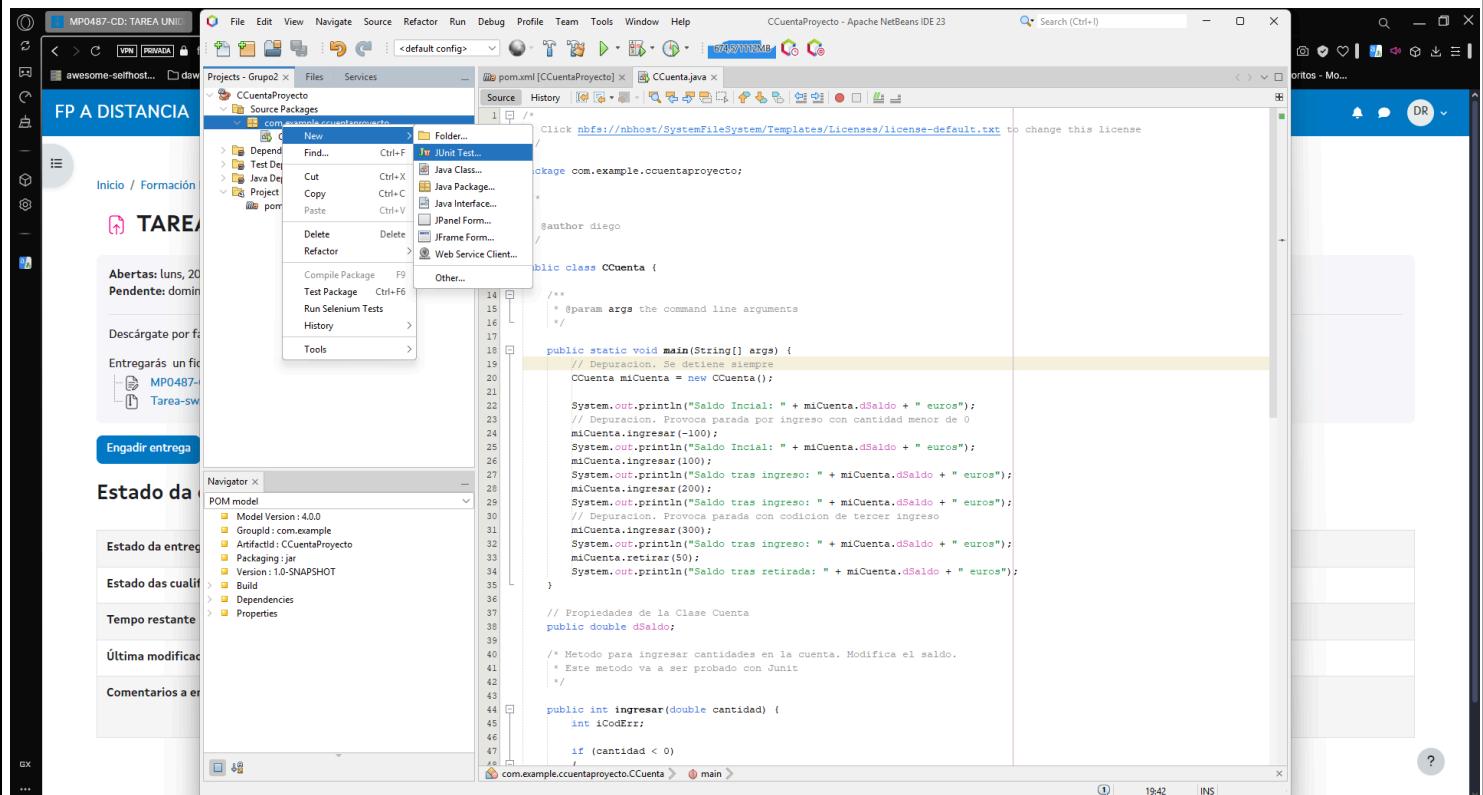
<project>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.release>23</maven.compiler.release>
        <exec.mainClass>com.example.cuentaproyecto.CuentaProyecto</exec.mainClass>
    </properties>
    <dependencies>
        <!-- JUnit 5 -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.9.0</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.9.0</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-params</artifactId> <!-- ;Dependencia faltante! -->
            <version>5.9.0</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-launcher</artifactId>
            <version>1.9.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.2.2</version>
                <configuration>
                    <includes>
                        <include>**/*Test.java</include>
                    </includes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>

```

```
<dependencies>
    <!-- JUnit 5 -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.9.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.9.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-params</artifactId> <!-- ¡Dependencia faltante! -->
        <version>5.9.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.platform</groupId>
        <artifactId>junit-platform-launcher</artifactId>
        <version>1.9.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>

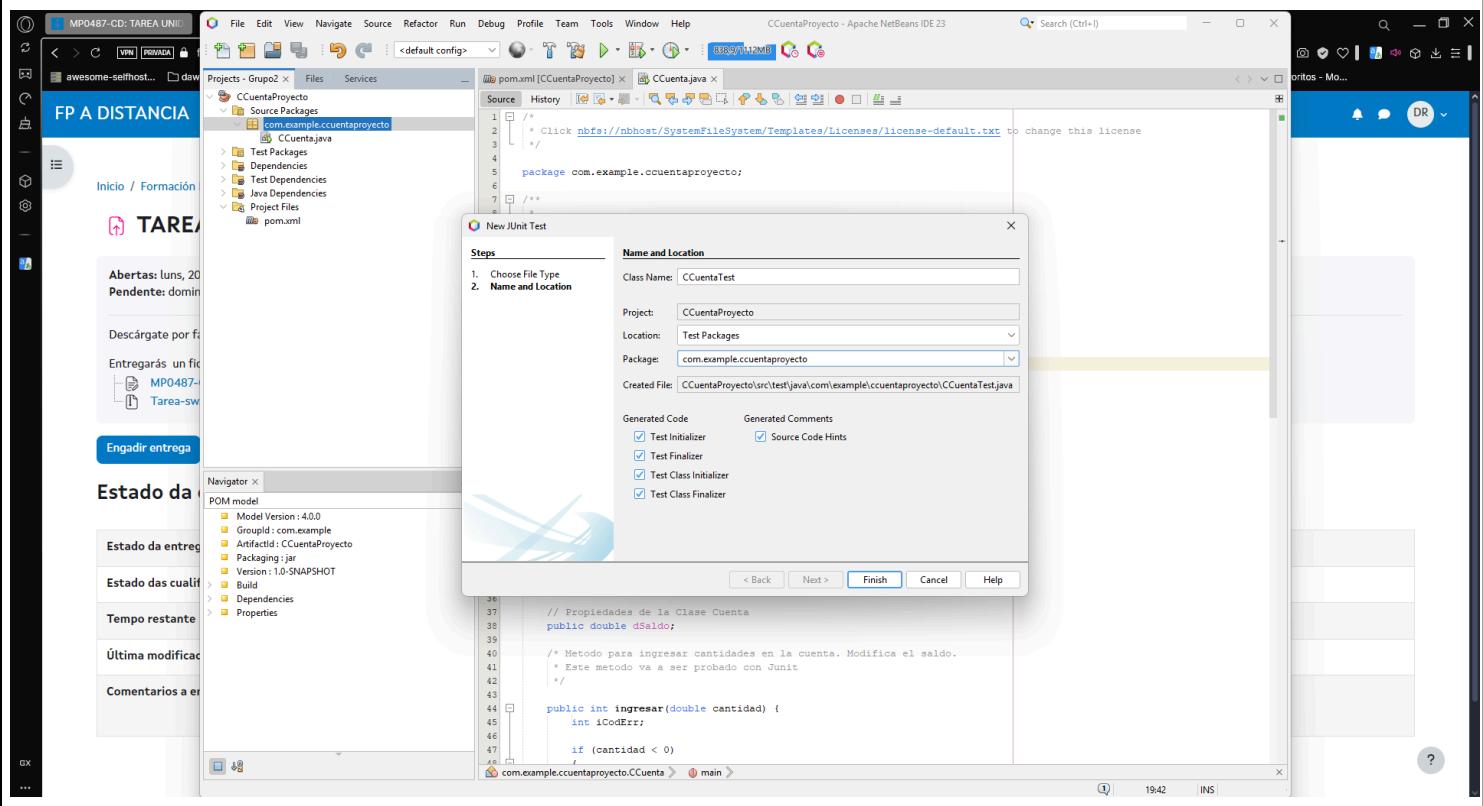
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.2.2</version>
            <configuration>
                <includes>
                    <include>**/*Test.java</include>
                </includes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Crear la clase CCuentaTest con el contenido aportado en el enunciado para ejecutar los test.



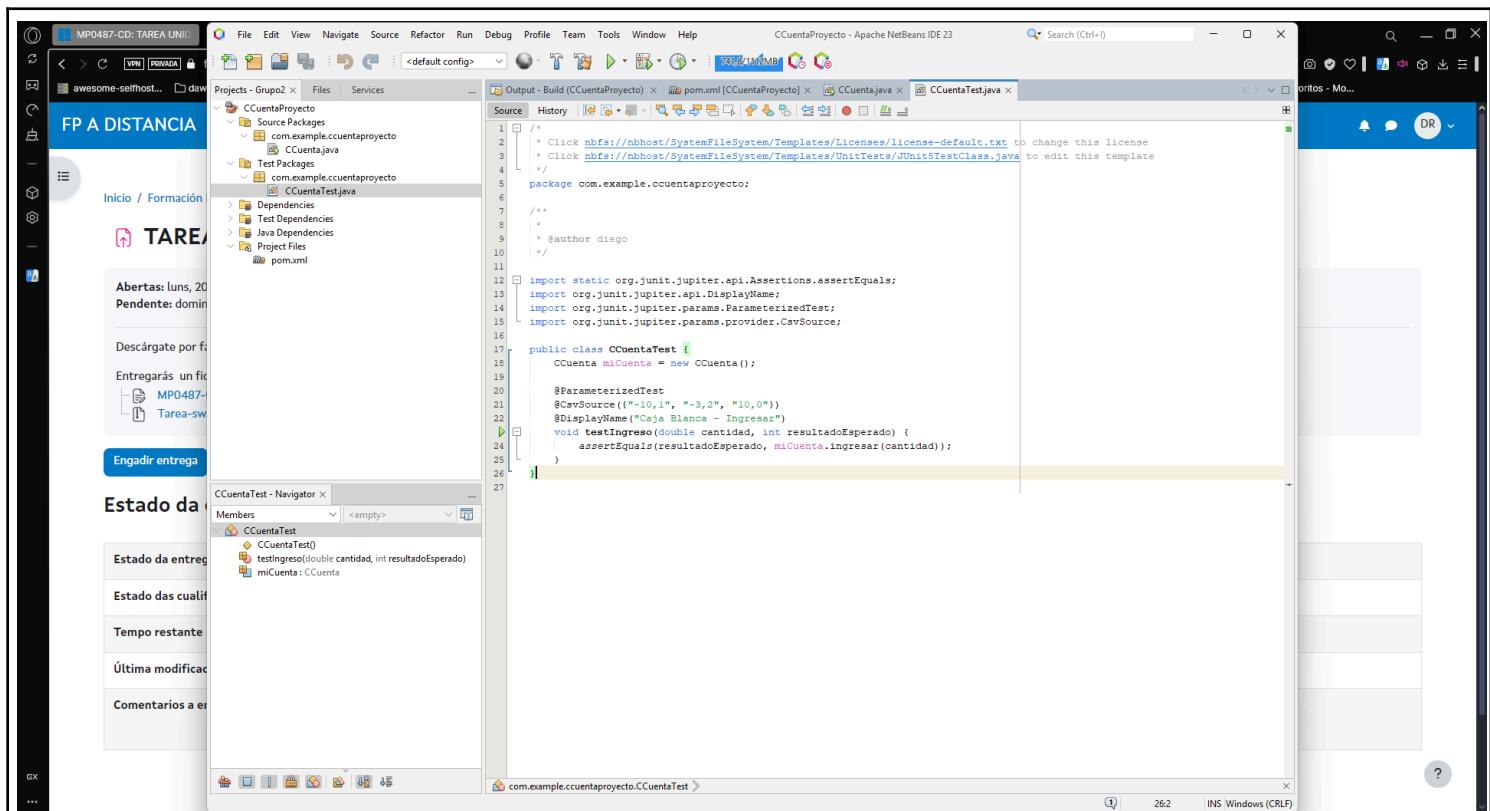
The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** The project is named "CCuentaProyecto". The "Source Packages" node contains "com.example.ccuentaproyecto" which has "CCuenta.java".
- Code Editor:** The code for "CCuenta.java" is visible, defining a class "CCuenta" with methods like "ingresar" and "retirar".
- Context Menu:** A context menu is open over the code editor, with the "New" option expanded. Under "New", "JUnit Test..." is selected.
- Navigator:** The Navigator panel shows the POM model, including Model Version: 4.0.0, GroupId: com.example, ArtifactId: CCuentaProyecto, Packaging: jar, and Version: 1.0-SNAPSHOT.
- Status Bar:** The status bar at the bottom right shows "19:42 INS".

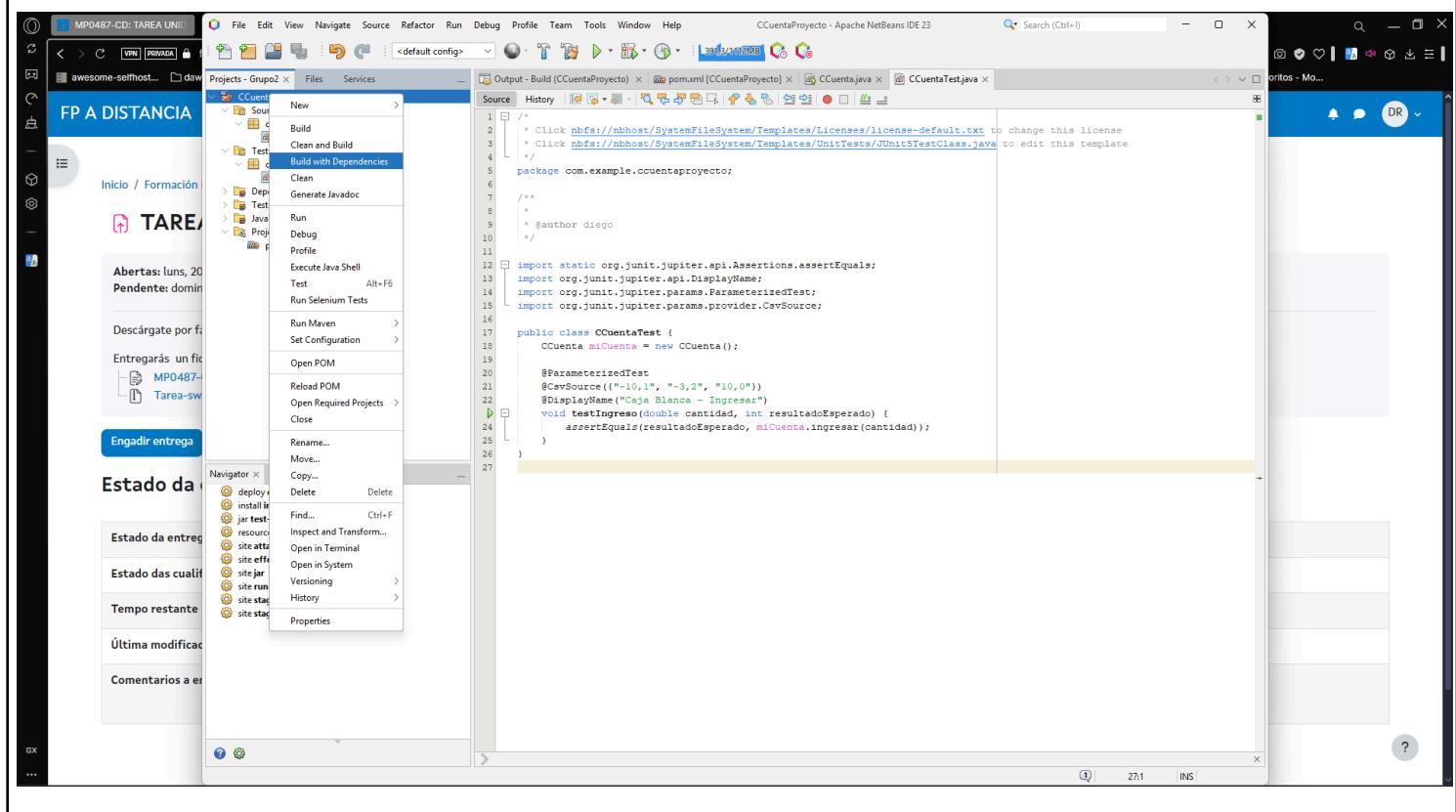


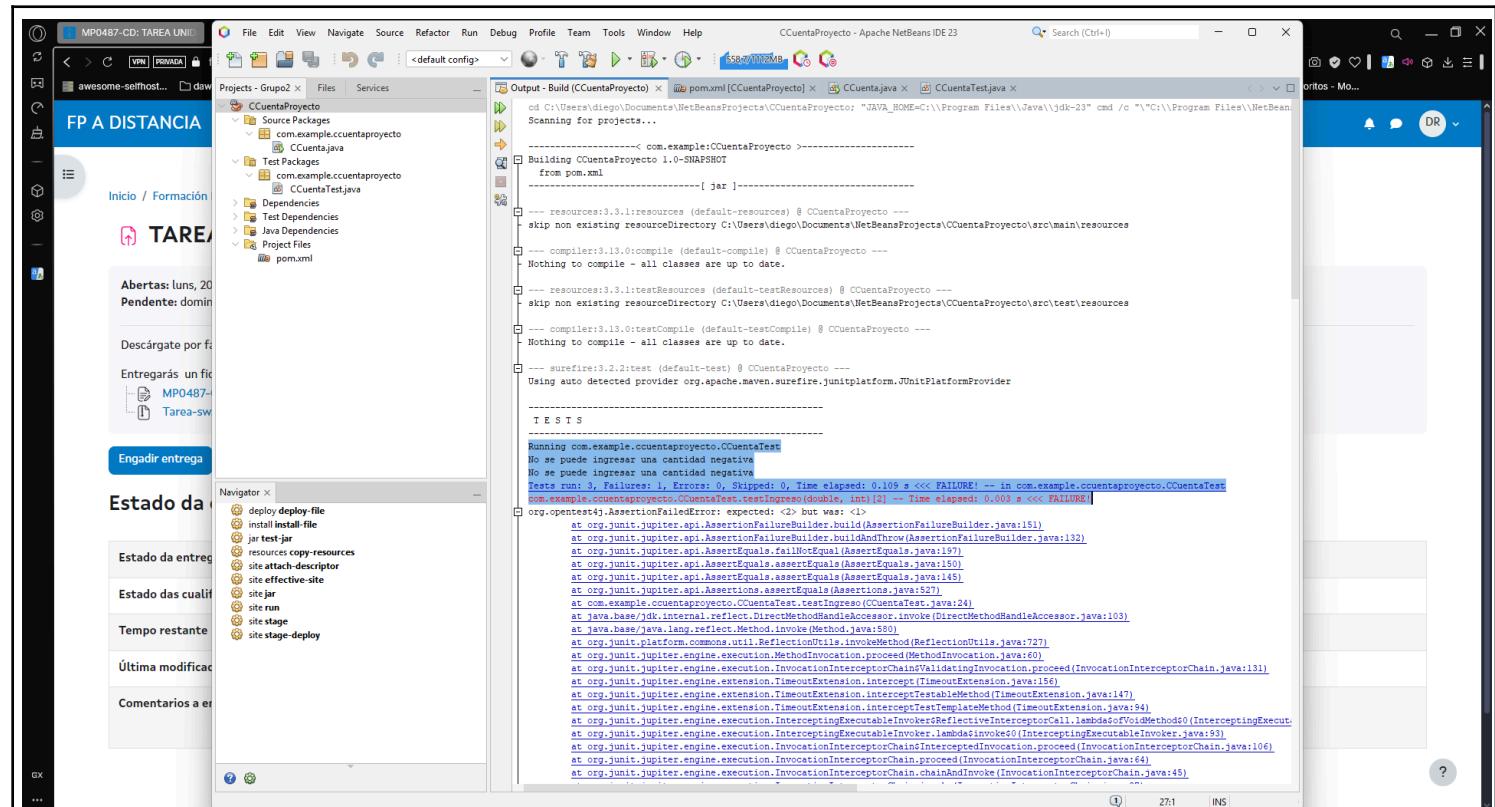
The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** The project is named "CCuentaProyecto". The "Source Packages" node contains "com.example.ccuentaproyecto" which has "CCuenta.java".
- Code Editor:** The code for "CCuenta.java" is visible.
- New JUnit Test Dialog:** A dialog box titled "New JUnit Test" is open, showing the "Name and Location" step. The "Class Name" field is set to "CCuentaTest", "Project" is "CCuentaProyecto", "Location" is "Test Packages", and "Package" is "com.example.ccuentaproyecto".
- Generated Code:** The generated code for "CCuentaTest.java" is shown in the code editor, including imports for "java.util.List" and "org.junit.Test".
- Generated Comments:** Checkboxes for "Test Initializer", "Test Finalizer", "Test Class Initializer", and "Test Class Finalizer" are checked.
- Navigator:** The Navigator panel shows the POM model, including Model Version: 4.0.0, GroupId: com.example, ArtifactId: CCuentaProyecto, Packaging: jar, and Version: 1.0-SNAPSHOT.
- Status Bar:** The status bar at the bottom right shows "19:42 INS".



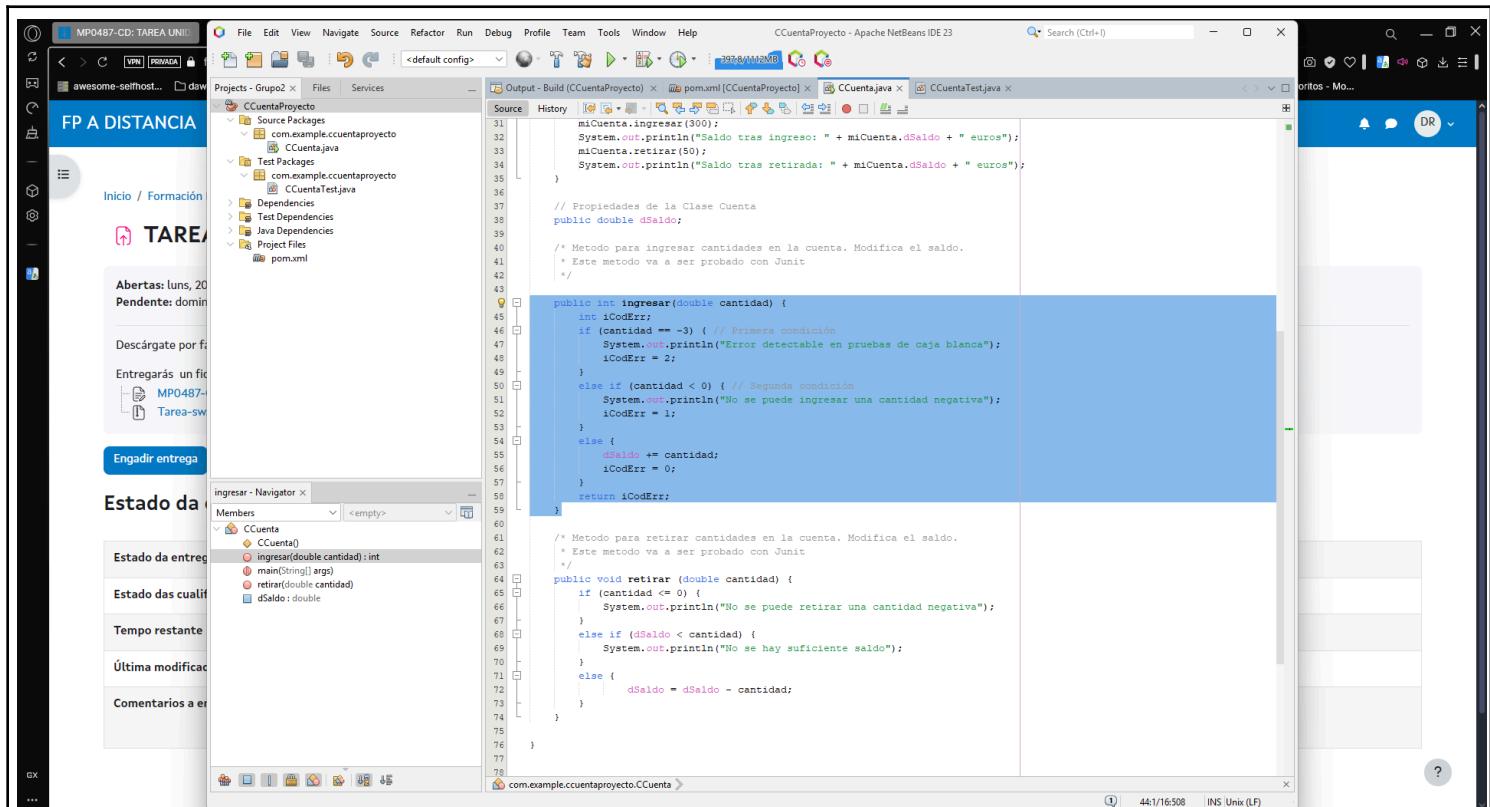
Compilar el proyecto dará error porque los tests (pruebas unitarias) recién configurados fallan debido a un error de lógica (la segunda condición del método es inalcanzable).





Para solucionarlo es necesario modificar el código de la clase CCuenta.java (método ingresar). El código corregido es el siguiente:

```
public int ingresar(double cantidad) {
    int iCodErr;
    if (cantidad == -3) { // Primera condición
        System.out.println("Error detectable en pruebas de caja blanca");
        iCodErr = 2;
    }
    else if (cantidad < 0) { // Segunda condición
        System.out.println("No se puede ingresar una cantidad negativa");
        iCodErr = 1;
    }
    else {
        dSaldo += cantidad;
        iCodErr = 0;
    }
    return iCodErr;
}
```



```

    miCuenta.ingresar(300);
    System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
    miCuenta.retirar(50);
    System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");

    // Propiedades de la Clase Cuenta
    public double dSaldo;

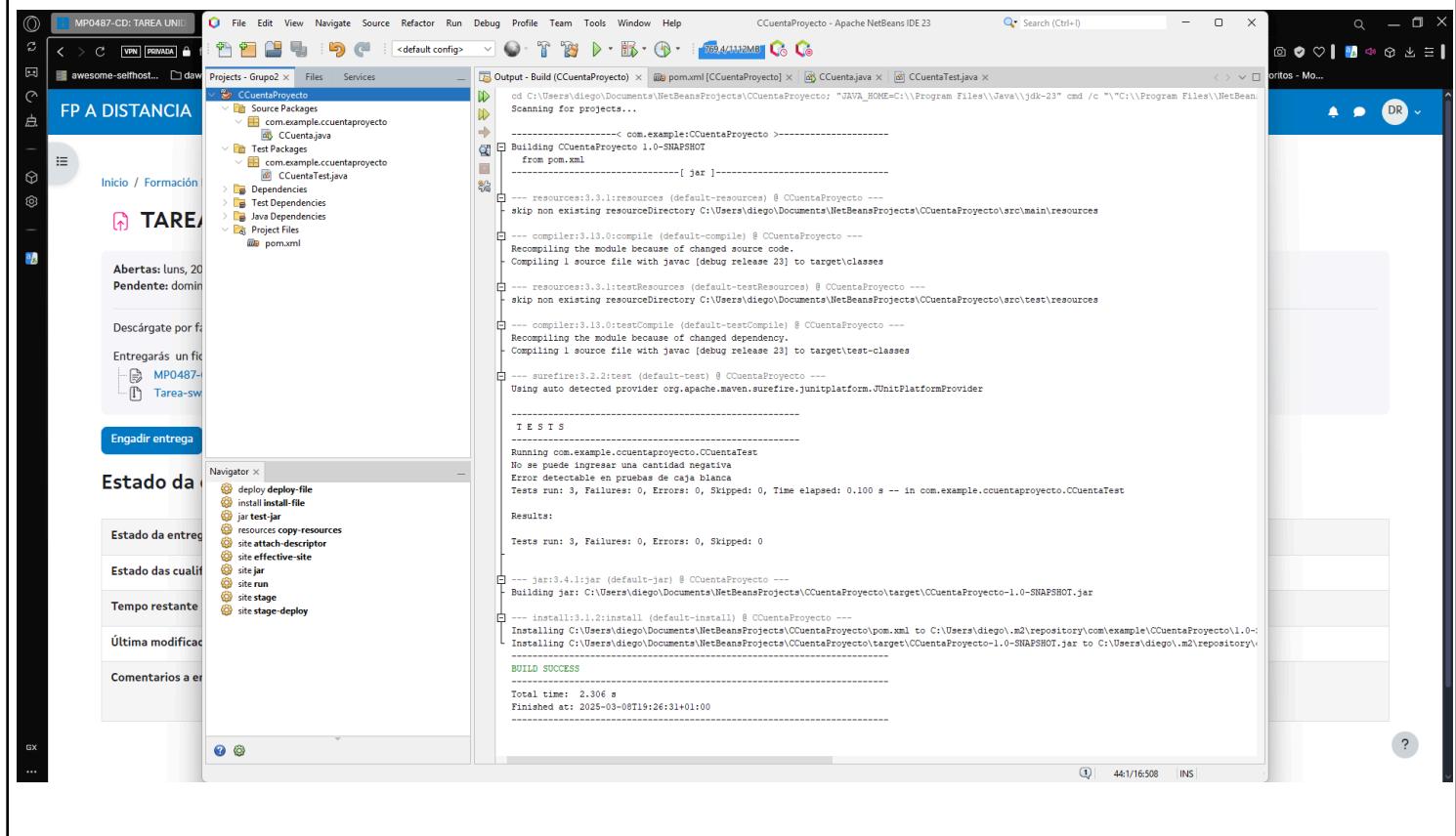
    /* Método para ingresar cantidades en la cuenta. Modifica el saldo.
     * Este método va a ser probado con JUnit
     */

    public int ingresar(double cantidad) {
        int iCodErr;
        if (cantidad == -3) { // Primera condición
            System.out.println("Error detectable en pruebas de caja blanca");
            iCodErr = 2;
        }
        else if (cantidad < 0) { // Segunda condición
            System.out.println("No se puede ingresar una cantidad negativa");
            iCodErr = 1;
        }
        else {
            dSaldo += cantidad;
            iCodErr = 0;
        }
        return iCodErr;
    }

    /* Método para retirar cantidades en la cuenta. Modifica el saldo.
     * Este método va a ser probado con JUnit
     */
    public void retirar (double cantidad) {
        if (cantidad < 0) {
            System.out.println("No se puede retirar una cantidad negativa");
        }
        else if (dSaldo < cantidad) {
            System.out.println("No se hay suficiente saldo");
        }
        else {
            dSaldo = dSaldo - cantidad;
        }
    }
}

```

Ahora el proyecto compila correctamente superando los tests.



```

    cd C:\Users\diego\Documents\NetBeansProjects\CCuentaProyecto; "JAVA_HOME=C:\\Program Files\\Java\\jdk-23" cmd /c "\C:\\Program Files\\NetBeans\\NetBeans 2023.1\\bin\\nb2023.1.exe" build

    Scanning for projects...
    Building CCuentaProyecto 1.0-SNAPSHOT
    from pom.xml
    [INFO] ---[jar]---

    --- resources:3.3:resources (default-resources) @ CCuentaProyecto ---
    skip non existing resourceDirectory C:\Users\diego\Documents\NetBeansProjects\CCuentaProyecto\src\main\resources

    --- compiler:3.13.0:compile (default-compile) @ CCuentaProyecto ---
    Recompiling the module because of changed source code.
    Compiling 1 source file with javac (debug release 23) to target\classes

    --- resources:3.3:testResources (default-testResources) @ CCuentaProyecto ---
    skip non existing resourceDirectory C:\Users\diego\Documents\NetBeansProjects\CCuentaProyecto\src\test\resources

    --- compiler:3.13.0:testCompile (default-testCompile) @ CCuentaProyecto ---
    Recompiling the module because of changed dependency.
    Compiling 1 source file with javac (debug release 23) to target\test-classes

    --- surefire:3.2.2:test (default-test) @ CCuentaProyecto ---
    Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider

    -----
    T E S T S
    -----
    Running com.example.ccuentaproyecto.CCuentaTest
    No se puede ingresar una cantidad negativa
    Error detectable en pruebas de caja blanca
    Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.100 s -- in com.example.ccuentaproyecto.CCuentaTest
    Results:
    Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

    --- jar:3.4.1:jar (default-jar) @ CCuentaProyecto ---
    Building jar: C:\Users\diego\Documents\NetBeansProjects\CCuentaProyecto\target\CCuentaProyecto-1.0-SNAPSHOT.jar

    --- install:3.1.2:install (default-install) @ CCuentaProyecto ---
    Installing C:\Users\diego\Documents\NetBeansProjects\CCuentaProyecto\pom.xml to C:\Users\diego\.m2\repository\com\example\CCuentaProyecto\1.0-SNAPSHOT\CCuentaProyecto-1.0-SNAPSHOT.jar
    Installing C:\Users\diego\Documents\NetBeansProjects\CCuentaProyecto\target\CCuentaProyecto-1.0-SNAPSHOT.jar to C:\Users\diego\.m2\repository\com\example\CCuentaProyecto\1.0-SNAPSHOT\CCuentaProyecto-1.0-SNAPSHOT.jar

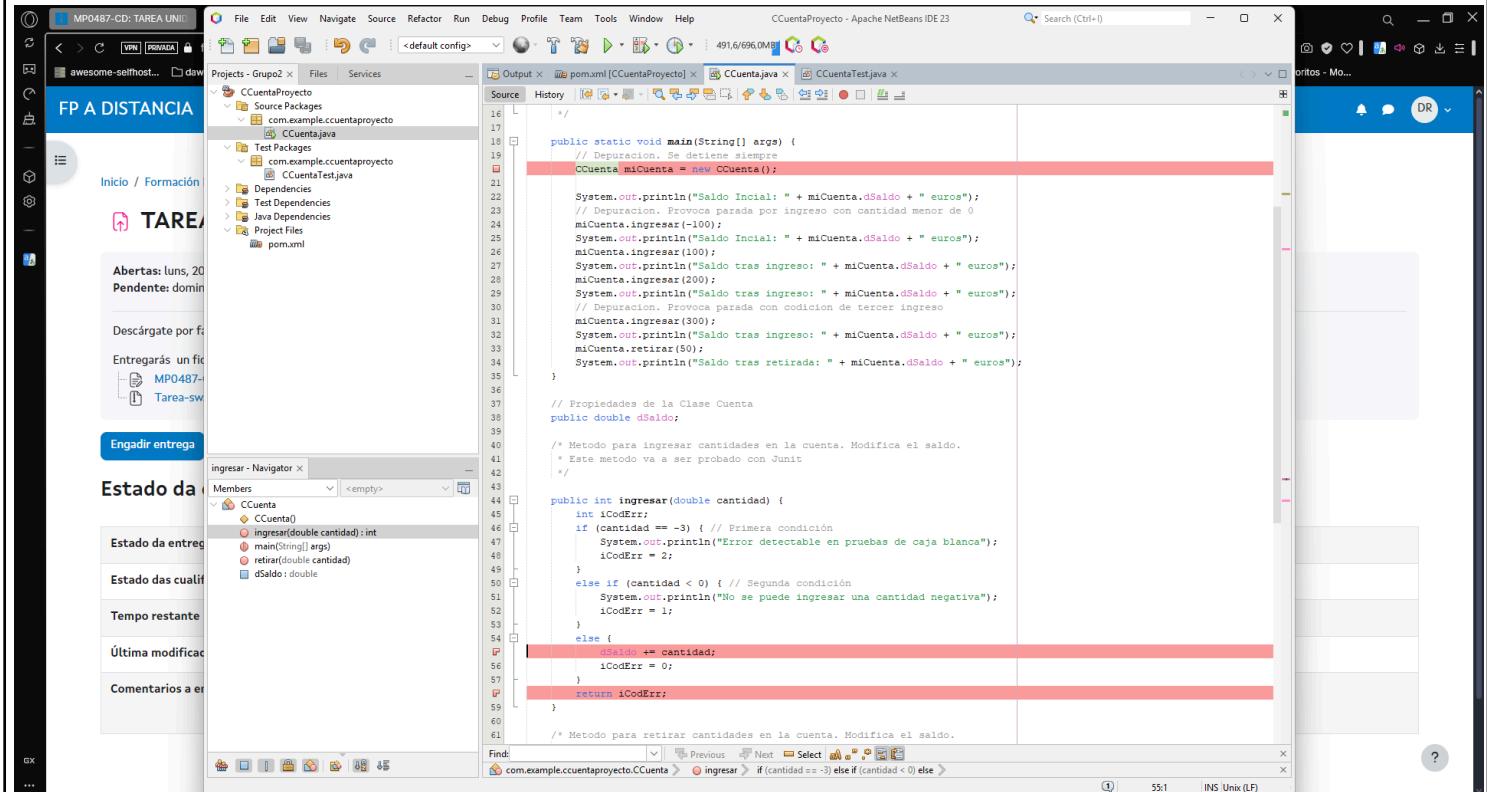
    BUILD SUCCESS
    Total time: 2.306 s
    Finished at: 2025-03-08T19:26:31+01:00
    -----

```

#### 4. Genera los siguientes puntos de ruptura para validar el comportamiento del método ingresar...

Capturas de pantalla ilustrando el proceso detallado

Añadir puntos de ruptura en las siguientes líneas:

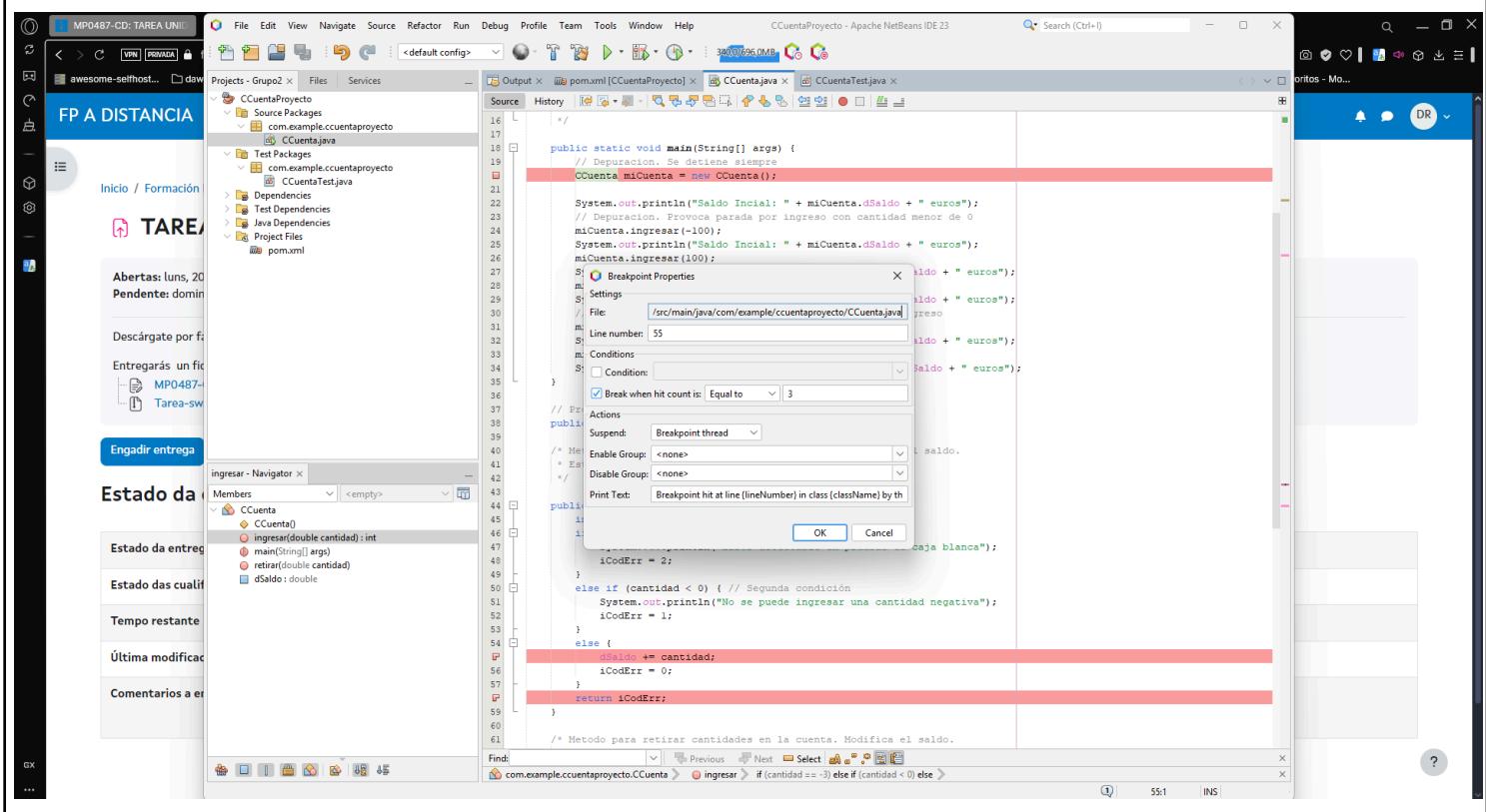


```

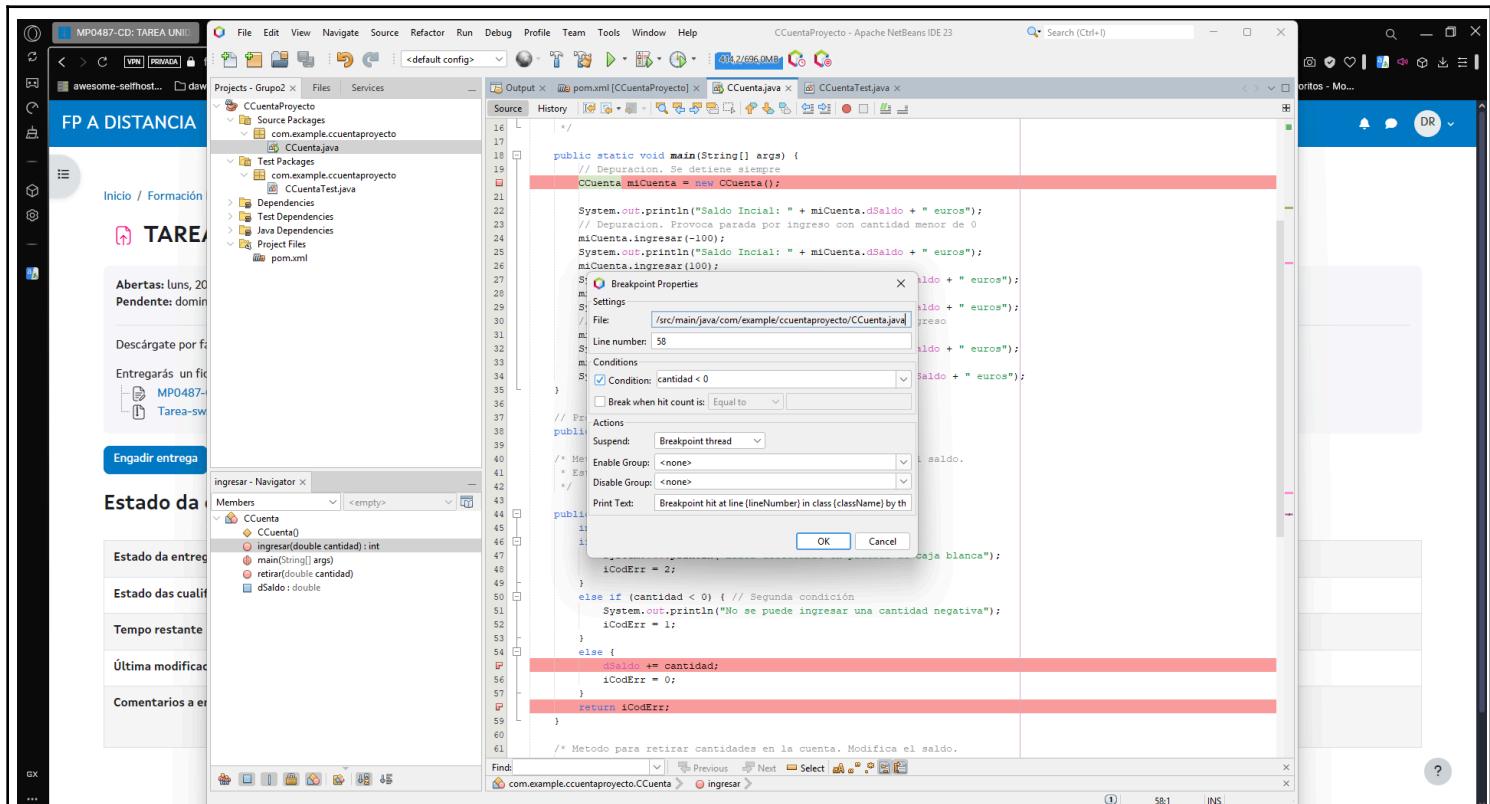
16
17
18   public static void main(String[] args) {
19     // Depuración. Se detiene siempre
20     CCuenta miCuenta = new CCuenta();
21
22     System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
23     // Depuración. Provee parada por ingreso con cantidad menor de 0
24     miCuenta.ingresar(-100);
25     System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
26     miCuenta.ingresar(100);
27     System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
28     miCuenta.ingresar(200);
29     System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
30     miCuenta.ingresar(300);
31     System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
32     miCuenta.retirar(50);
33     System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
34
35   }
36
37   /* Propiedades de la Clase Cuenta
38   public double dSaldo;
39
40   * Método para ingresar cantidades en la cuenta. Modifica el saldo.
41   * Este método va a ser probado con Junit
42   */
43
44   public int ingresar(double cantidad) {
45     int iCodErr;
46     if (cantidad == -3) { // Primera condición
47       System.out.println("Error detectable en pruebas de caja blanca");
48       iCodErr = 2;
49     }
50     else if (cantidad < 0) { // Segunda condición
51       System.out.println("No se puede ingresar una cantidad negativa");
52       iCodErr = 1;
53     }
54     else {
55       dSaldo += cantidad;
56       iCodErr = 0;
57     }
58     return iCodErr;
59   }
60
61   /* Método para retirar cantidades en la cuenta. Modifica el saldo.

```

Modificar las propiedades de los dos últimos puntos de ruptura:



The screenshot shows the NetBeans IDE interface with the CCuenta.java file open. A breakpoint has been set at line 55, which is part of the 'ingresar' method. The 'Breakpoint Properties' dialog is displayed, showing the condition 'Break when hit count is: Equal to 3'. The 'File' field is set to 'src/main/java/com/example/ccuentaproyecto/CCuenta.java'. The 'Line number' field is set to 55. The 'Conditions' section is expanded, showing an empty condition field.



Breakpoint Properties

File: /src/main/java/com/example/ccuentaproyecto/CCuenta.java

Line number: 58

Conditions:

- Condition: cantidad < 0
- Break when hit count is: Equal to
- Actions: Suspend: Breakpoint thread
- Enable Group: <none>
- Disable Group: <none>
- Print Text: Breakpoint hit at line (lineNumber) in class (className) by th

```

public static void main(String[] args) {
    // Depuración. Se detiene siempre
    CCuenta miCuenta = new CCuenta();

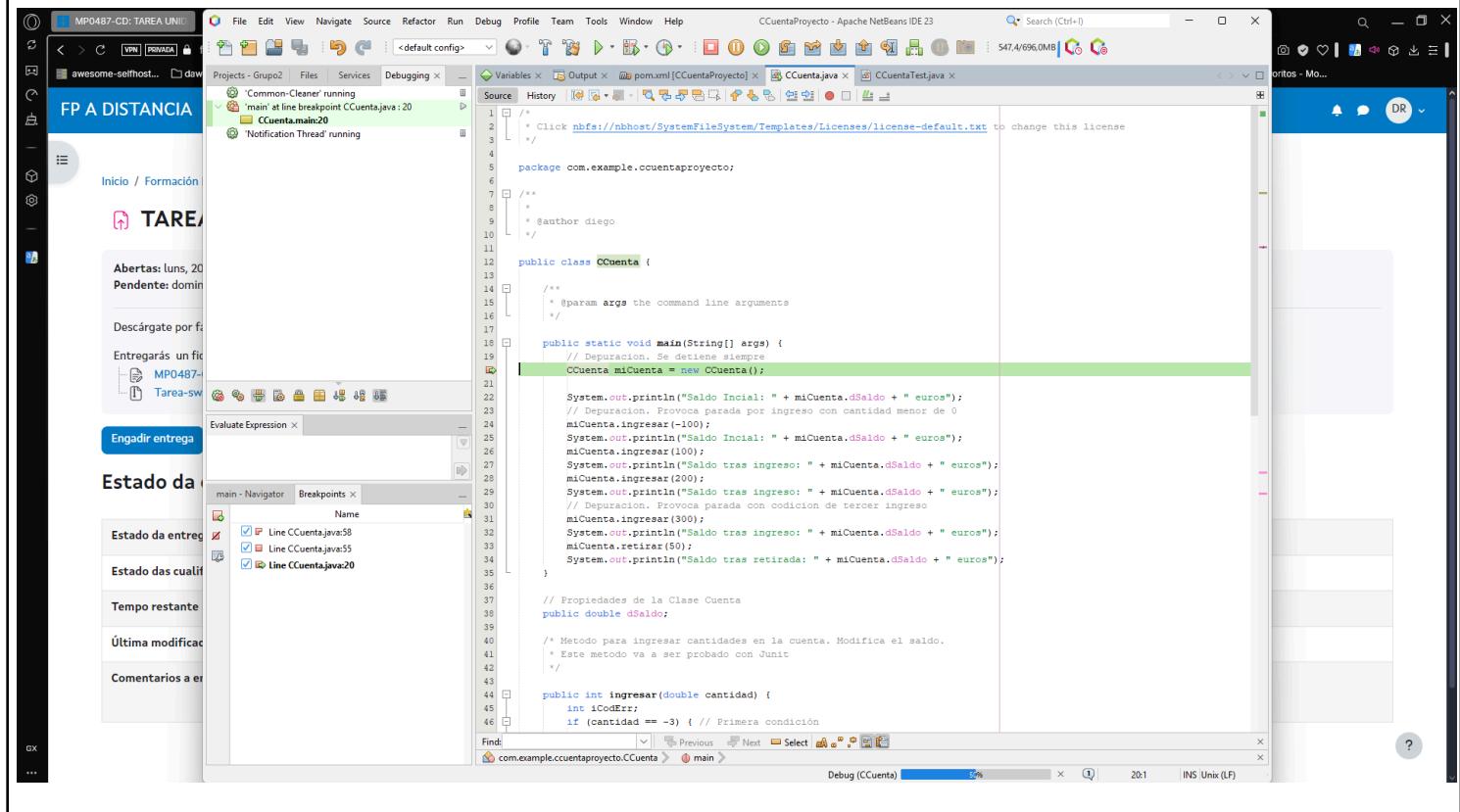
    System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
    // Depuración. Provoca parada por ingreso con cantidad menor de 0
    miCuenta.ingresar(-100);
    System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
    miCuenta.ingresar(100);

    if (cantidad < 0) {
        System.out.println("Saldo: " + miCuenta.dSaldo + " euros");
        iCodErr = 2;
    } else if (cantidad < 0) { // Segunda condición
        System.out.println("No se puede ingresar una cantidad negativa");
        iCodErr = 1;
    } else {
        dSaldo += cantidad;
        iCodErr = 0;
    }
    return iCodErr;
}

/* Método para retirar cantidades en la cuenta. Modifica el saldo.
 * Este método va a ser probado con Junit
 */
public int retirar(double cantidad) {
    int iCodErr;
    if (cantidad == -3) { // Primera condición
        System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
        miCuenta.retirar(50);
        System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
    }
}

```

Ejecutar CCuenta.java en modo depuración (Debug File).



Breakpoints

Name
Line CCuenta.java:58
Line CCuenta.java:55
Line CCuenta.java:20

```

public class CCuenta {
    /**
     * @param args the command line arguments
     */

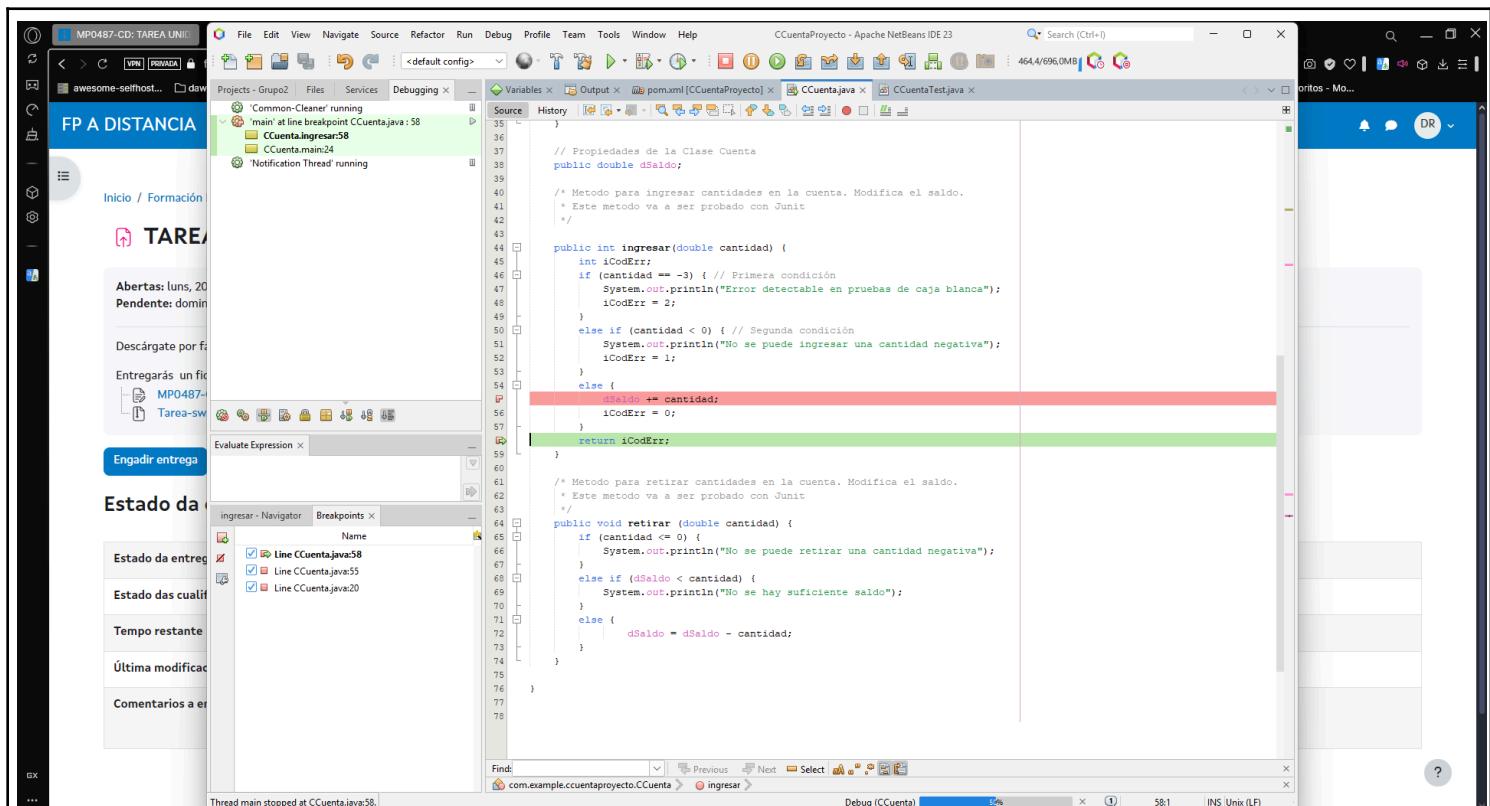
    public static void main(String[] args) {
        // Depuración. Se detiene siempre
        CCuenta miCuenta = new CCuenta();

        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
        // Depuración. Provoca parada por ingreso con cantidad menor de 0
        miCuenta.ingresar(-100);
        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
        miCuenta.ingresar(100);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        miCuenta.ingresar(200);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        // Depuración. Provoca parada con codición de tercer ingreso
        miCuenta.ingresar(300);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        miCuenta.retirar(50);
        System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");

        // Propiedades de la Clase Cuenta
        public double dSaldo;

        /* Método para ingresar cantidades en la cuenta. Modifica el saldo.
         * Este método va a ser probado con Junit
         */
        public int ingresar(double cantidad) {
            int iCodErr;
            if (cantidad == -3) { // Primera condición
                System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
                miCuenta.ingresar(100);
                System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
            }
        }
    }
}

```



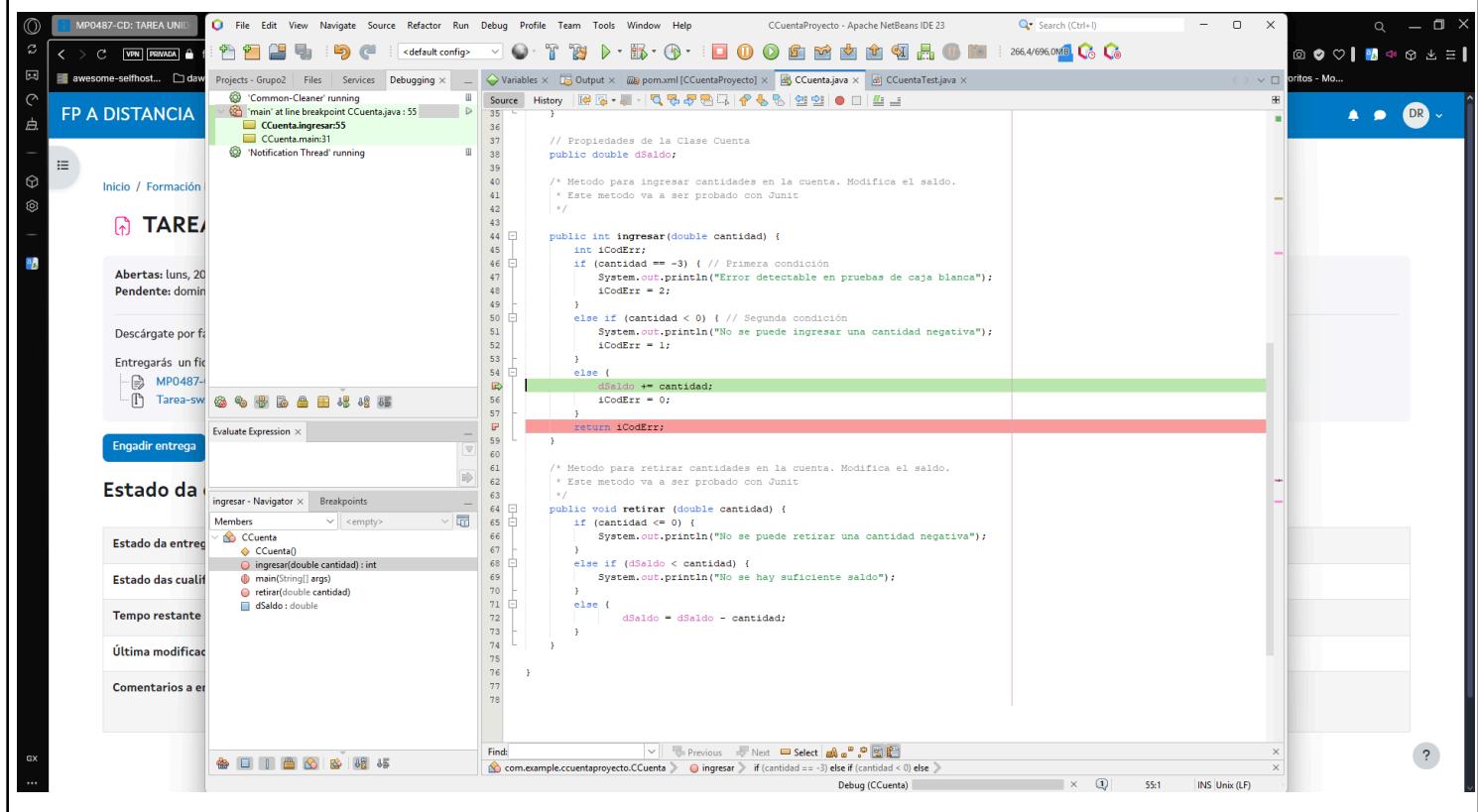
```

    // Propiedades de la Clase Cuenta
    public double dSaldo;

    /* 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 == -3) { // Primera condición
            System.out.println("Error detectable en pruebas de caja blanca");
            iCodErr = 2;
        }
        else if (cantidad < 0) { // Segunda condición
            System.out.println("No se puede ingresar una cantidad negativa");
            iCodErr = 1;
        }
        else {
            dSaldo += cantidad;
            iCodErr = 0;
        }
        return iCodErr;
    }

    /* Metodo para retirar cantidades en la cuenta. Modifica el saldo.
     * Este metodo va a ser probado con Junit
     */
    public void retirar (double cantidad) {
        if (cantidad < 0) {
            System.out.println("No se puede retirar una cantidad negativa");
        }
        else if (dSaldo < cantidad) {
            System.out.println("No se hay suficiente saldo");
        }
        else {
            dSaldo -= cantidad;
        }
    }
}

```



```

    // Propiedades de la Clase Cuenta
    public double dSaldo;

    /* 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 == -3) { // Primera condición
            System.out.println("Error detectable en pruebas de caja blanca");
            iCodErr = 2;
        }
        else if (cantidad < 0) { // Segunda condición
            System.out.println("No se puede ingresar una cantidad negativa");
            iCodErr = 1;
        }
        else {
            dSaldo += cantidad;
            iCodErr = 0;
        }
        return iCodErr;
    }

    /* Metodo para retirar cantidades en la cuenta. Modifica el saldo.
     * Este metodo va a ser probado con Junit
     */
    public void retirar (double cantidad) {
        if (cantidad < 0) {
            System.out.println("No se puede retirar una cantidad negativa");
        }
        else if (dSaldo < cantidad) {
            System.out.println("No se hay suficiente saldo");
        }
        else {
            dSaldo -= cantidad;
        }
    }
}

```

Información del fichero del puntos de ruptura (breakpoints)

No encontré como hacerlo, lo hice manualmente

# Breakpoints para CCuenta.java

```
file:///C:/Users/diego/Documents/NetBeansProjects/CCuentaProyecto/src/main/java/com/example/ccuentaproyecto/CCuenta.java
line=20
enabled=true
```

```
file:///C:/Users/diego/Documents/NetBeansProjects/CCuentaProyecto/src/main/java/com/example/ccuentaproyecto/CCuenta.java
line=55
enabled=true
hitcount=3
```

```
file:///C:/Users/diego/Documents/NetBeansProjects/CCuentaProyecto/src/main/java/com/example/ccuentaproyecto/CCuenta.java
line=58
enabled=true
condition=cantidad < 0
```

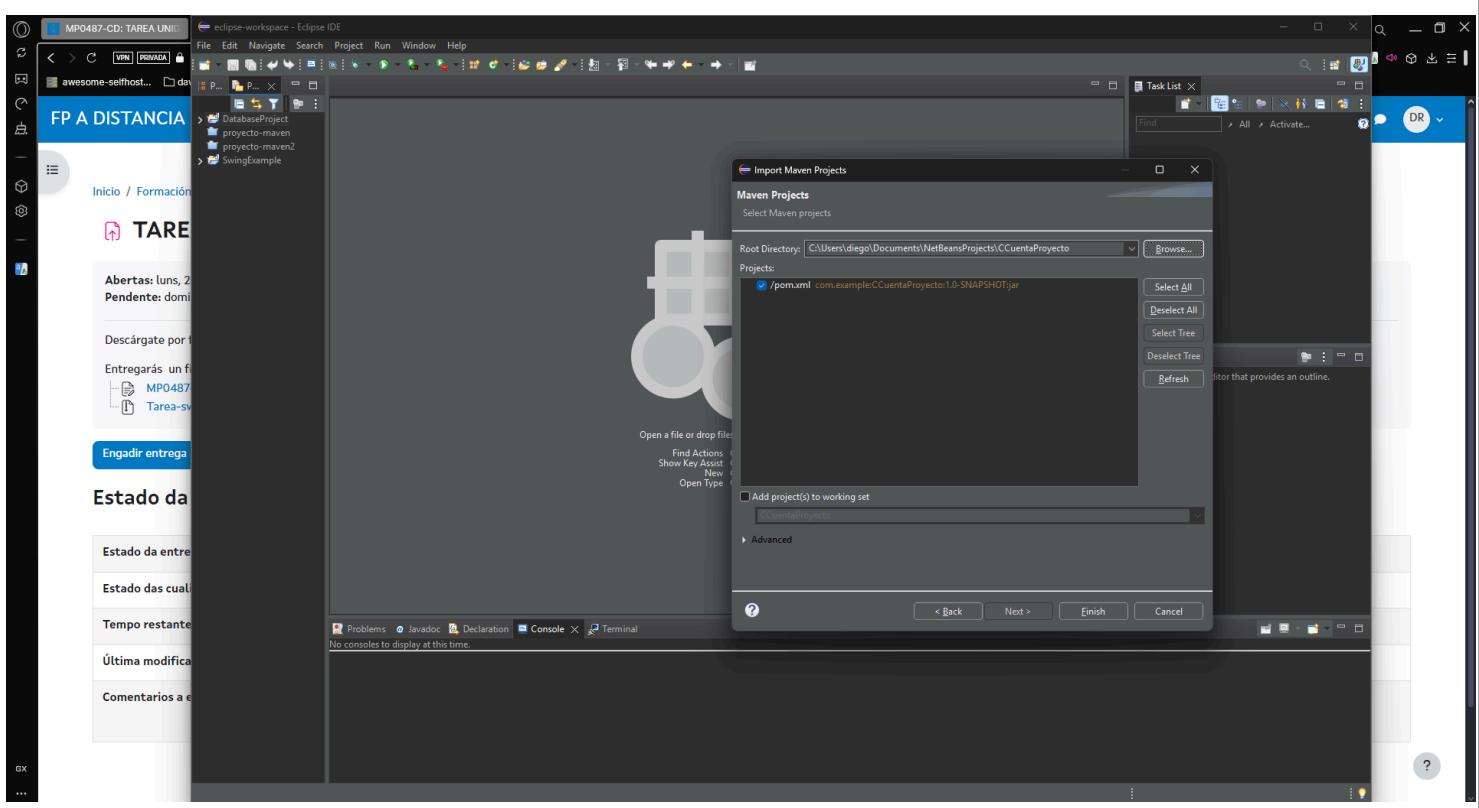
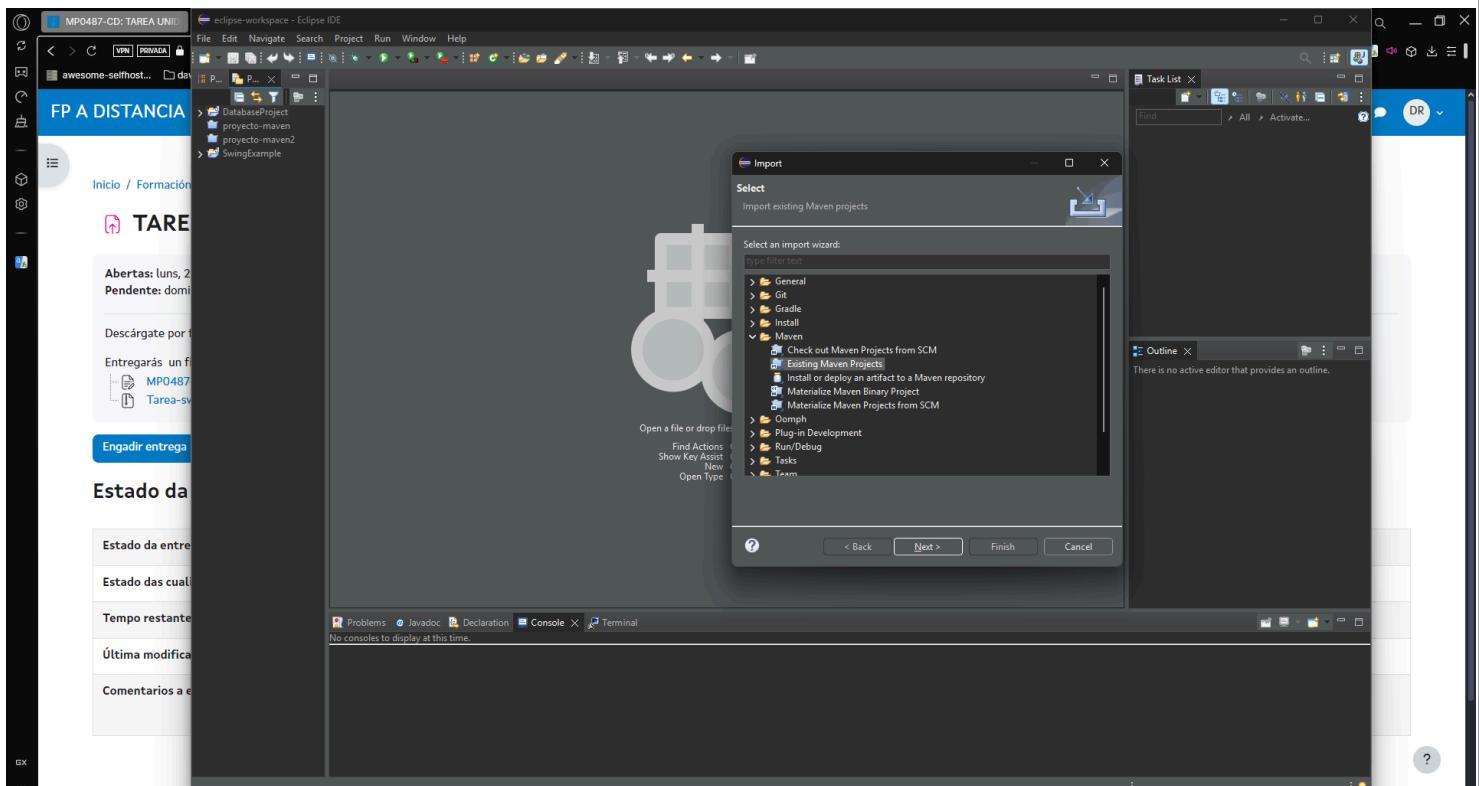
## PASOS 3 y 4: SOLUCIÓN EN ECLIPSE

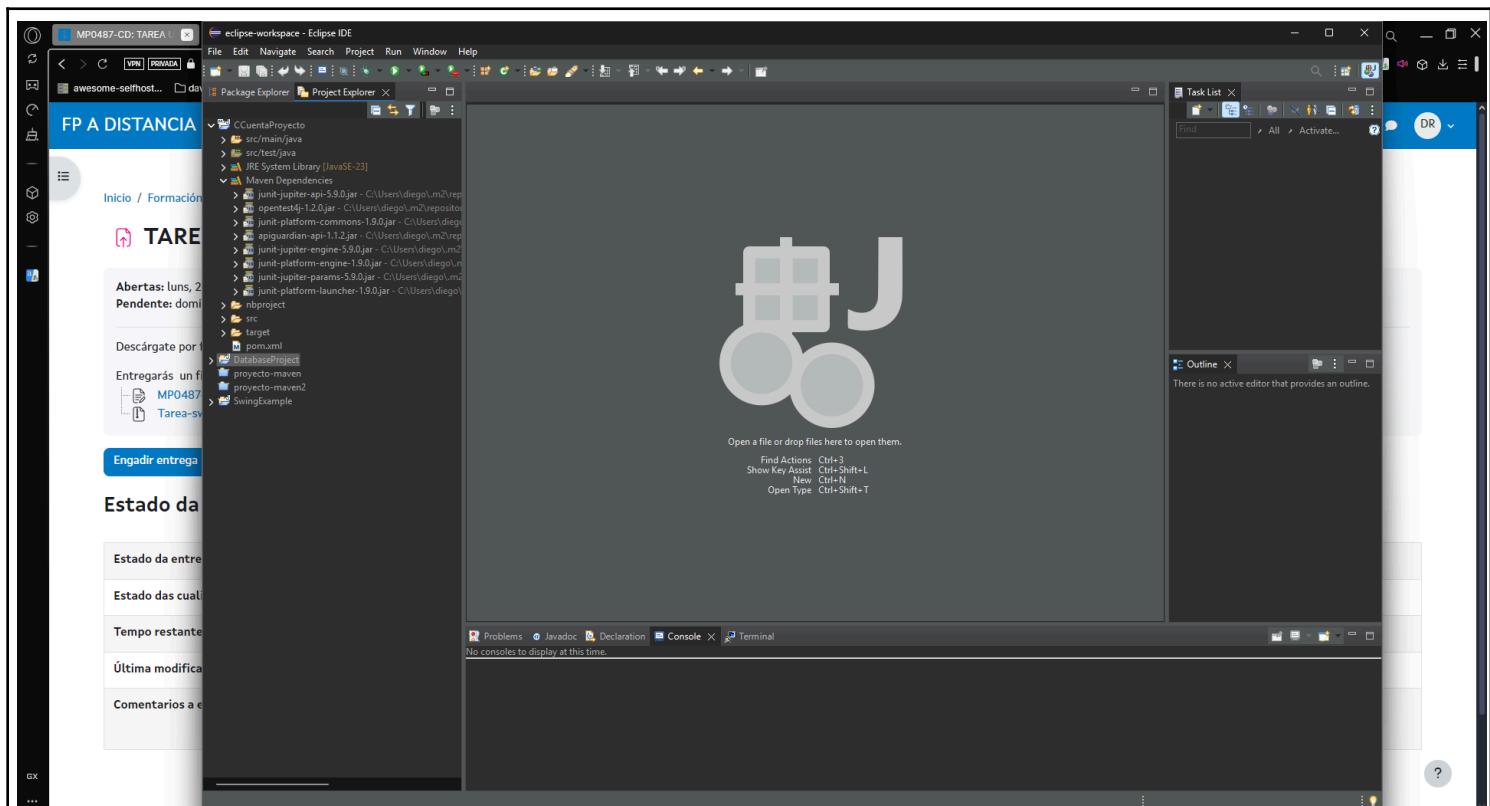
### 3. Crea la clase CCuentaTest del tipo Caso de prueba JUnit

Capturas de pantalla ilustrando el proceso detallado

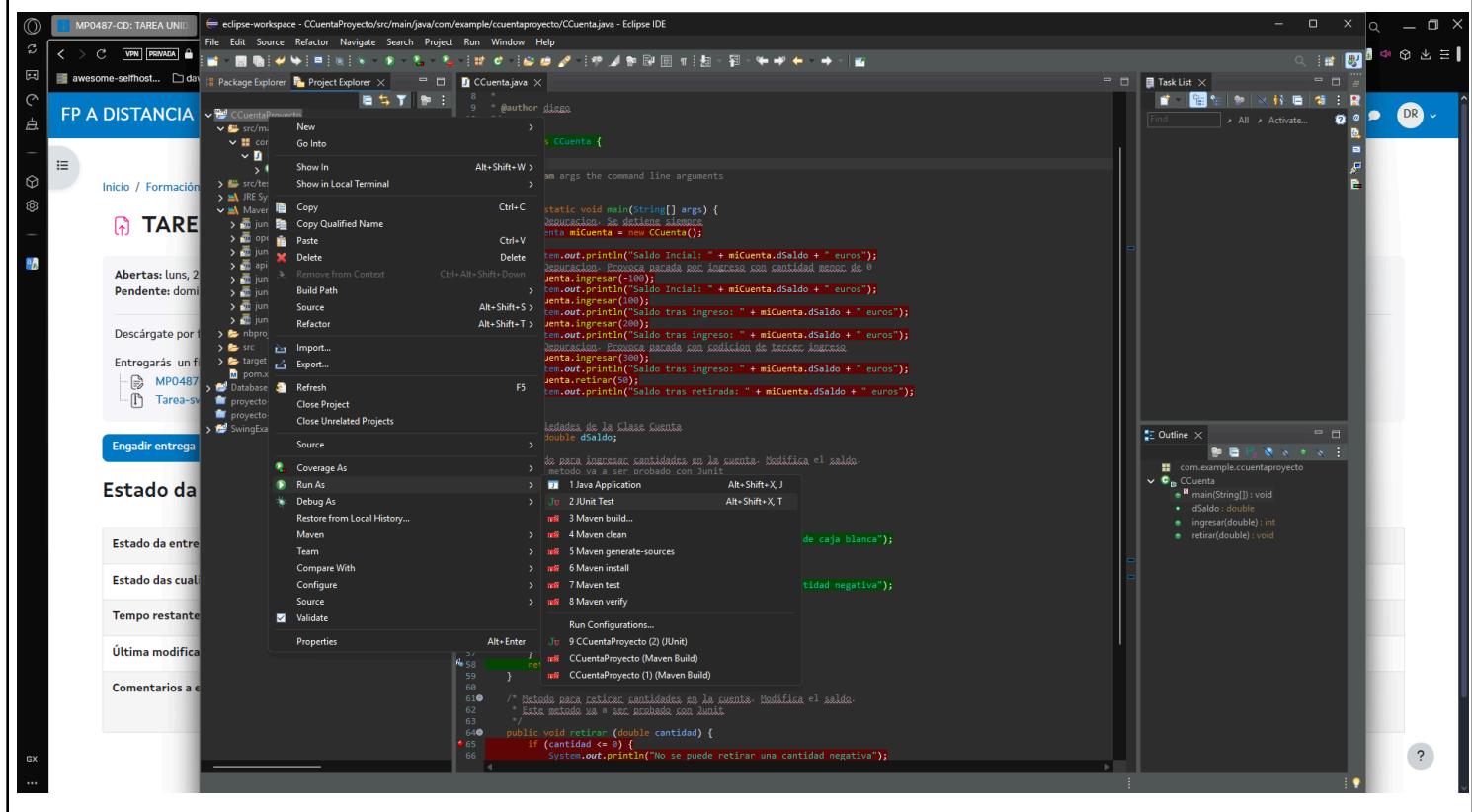
Voy a importar directamente el proyecto que ya he creado en Netbeans ya que la lógica es la misma:

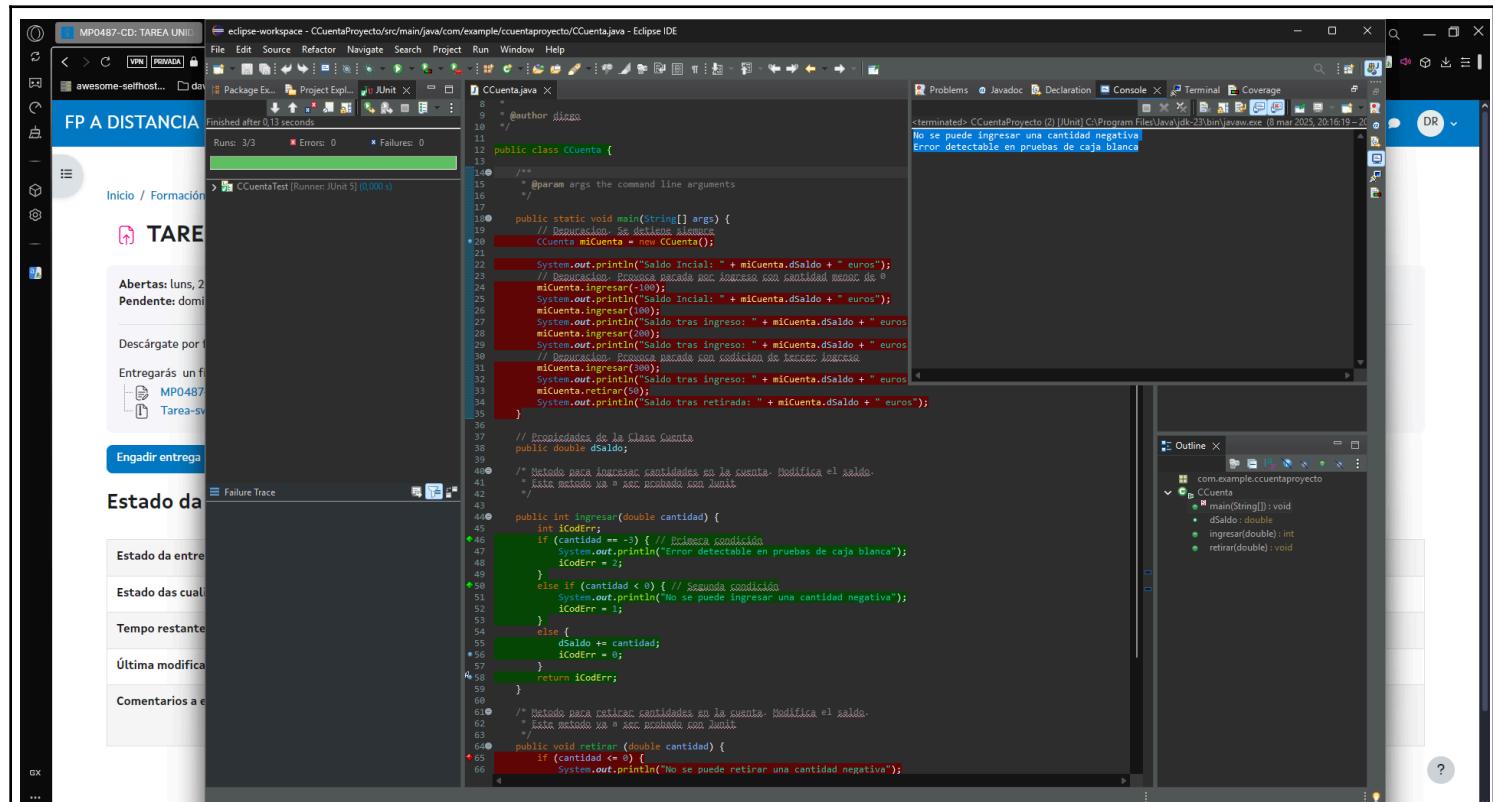
File → Import → Maven → Existing Maven Projects.





Ejecuto las pruebas unitarias.





```


1 /**
2  * @param args the command line arguments
3 */
4 public static void main(String[] args) {
5     // Recuración: se inicializa el objeto
6     CCuenta miCuenta = new CCuenta();
7
8     System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
9     // Recuración: Excepción lanzada con la cantidad menor de 0
10    miCuenta.ingresar(-100);
11    System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
12    miCuenta.ingresar(0);
13    System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
14    miCuenta.ingresar(200);
15    System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
16
17    // Recuración: Excepción lanzada con cantidad de saldo menor
18    miCuenta.retirar(300);
19    System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
20    miCuenta.retirar(50);
21    System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
22
23 }
24
25 /**
26  * Propiedades de la Clase Cuenta
27 */
28 public double dSaldo;
29
30 /**
31  * Método para ingresar cantidades en la cuenta. Modifica el saldo.
32  * Este método no a excepciones con null.
33 */
34 public int ingresar(double cantidad) {
35     int iCodErr;
36
37     if (cantidad == -3) { // Primera condición
38         System.out.println("Error detectable en pruebas de caja blanca");
39         iCodErr = 2;
40     }
41     else if (cantidad < 0) { // Segunda condición
42         System.out.println("No se puede ingresar una cantidad negativa");
43         iCodErr = 1;
44     }
45     else {
46         dSaldo += cantidad;
47         iCodErr = 0;
48     }
49
50     return iCodErr;
51 }
52
53 /**
54  * Método para retirar cantidades en la cuenta. Modifica el saldo.
55  * Este método no a excepciones con null.
56 */
57 public void retirar (double cantidad) {
58     if (cantidad < 0) {
59         System.out.println("No se puede retirar una cantidad negativa");
60     }
61 }
62
63 /**
64  * Recuración: se inicializa el objeto
65 */
66


```

Para cantidad = -10:

El método detecta cantidad < 0 e imprime No se puede ingresar una cantidad negativa → retorna código 1.

Para cantidad = -3:

El método detecta cantidad == -3 e imprime Error detectable... → retorna código 2.

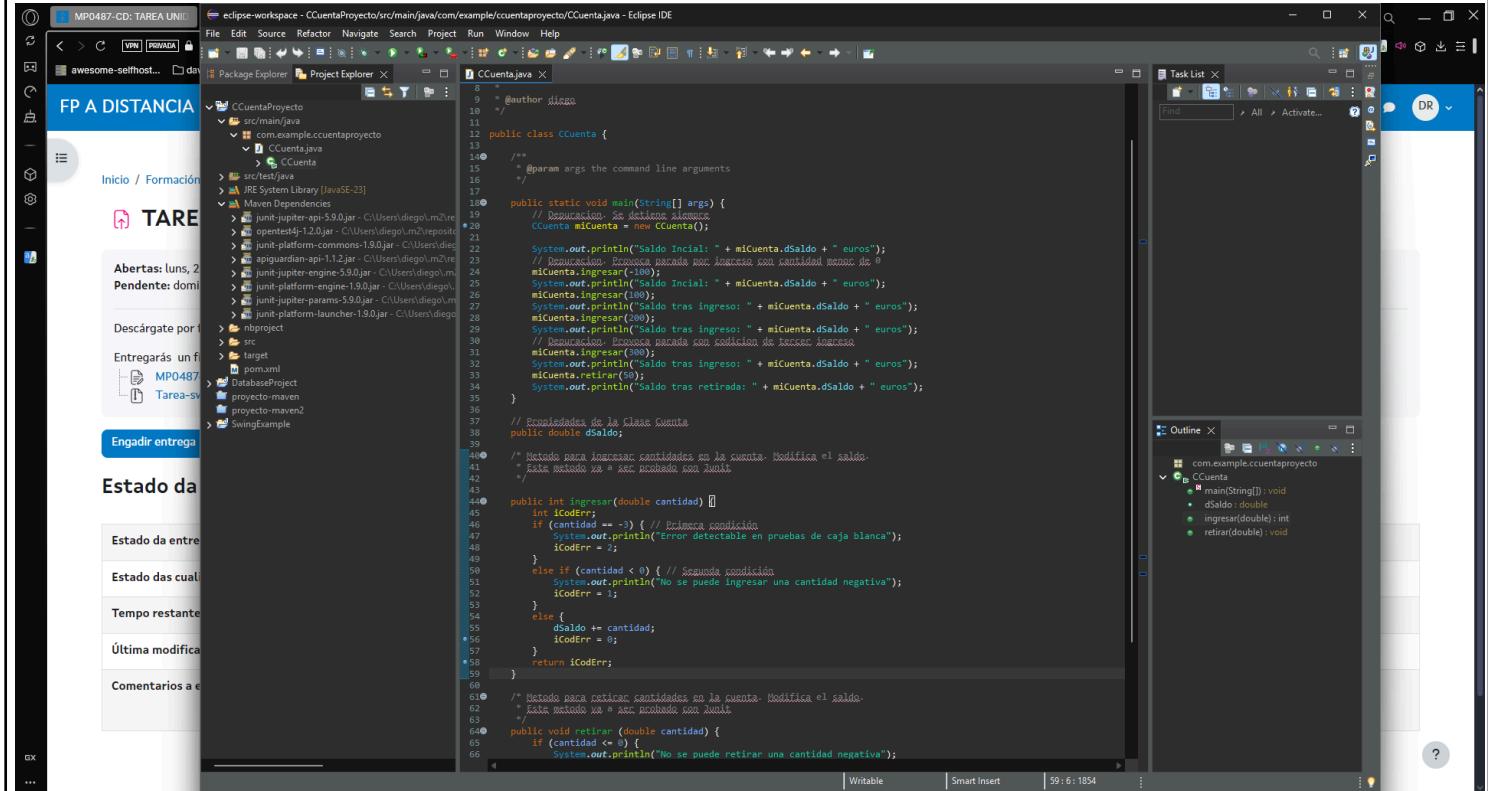
Para cantidad = 10:

No hay mensajes (caso exitoso).

#### 4. Genera los siguientes puntos de ruptura para validar el comportamiento del método ingresar...

Capturas de pantalla ilustrando el proceso detallado

Añado los breakpoints en las mismas líneas que en Netbeans. También configuro las propiedades de los dos últimos de la misma forma (condición y Hit Count).



```

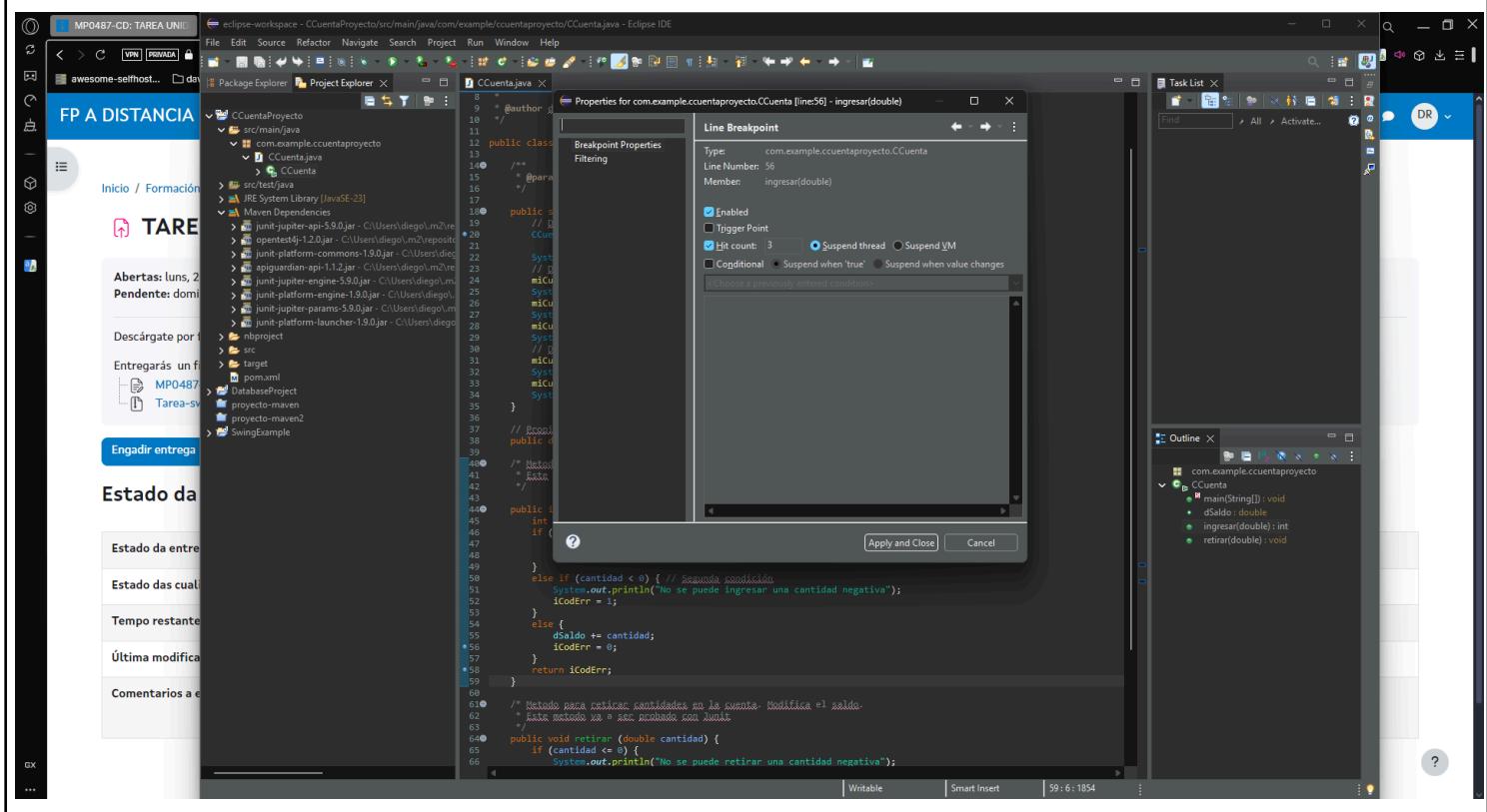
public class CCuenta {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Inicializa la cuenta con un saldo inicial
        CCuenta miCuenta = new CCuenta();
        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
        miCuenta.ingresar(-100);
        System.out.println("Saldo Inicial: " + miCuenta.dSaldo + " euros");
        miCuenta.retirar(50);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        miCuenta.ingresar(200);
        System.out.println("Saldo tras ingreso: " + miCuenta.dSaldo + " euros");
        miCuenta.retirar(150);
        System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
    }

    // Propiedades de la Clase Cuenta
    public double dSaldo;
}

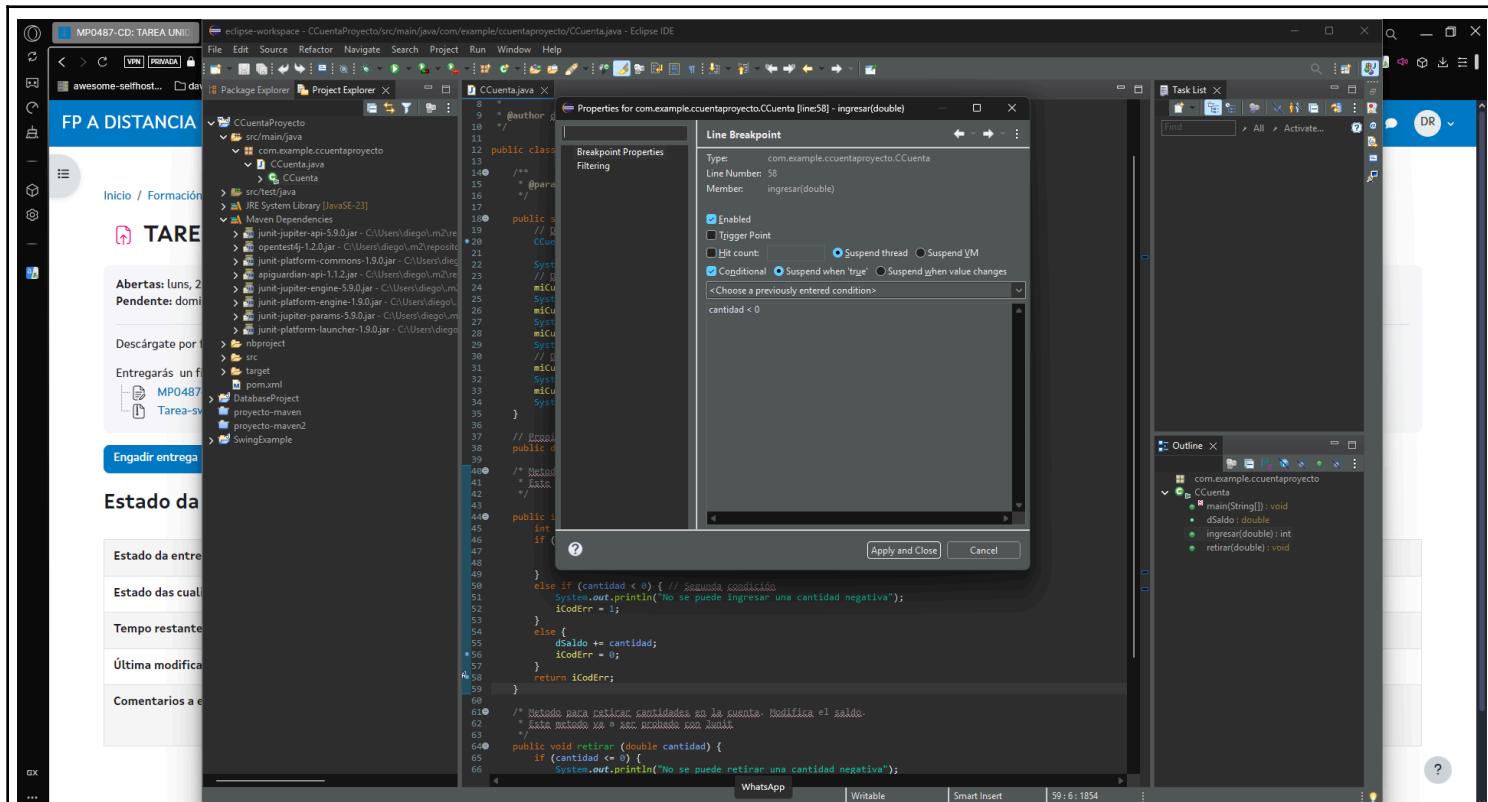
/* Método para ingresar cantidades en la cuenta. Modifica el saldo.
   Esta método no es recomendado para clientes.
*/
public int ingresar(double cantidad) {
    int iCodErr;
    if (cantidad == -3) { // Primera condición
        System.out.println("Error detectable en pruebas de caja blanca");
        iCodErr = 2;
    } else if (cantidad < 0) { // Segunda condición
        System.out.println("No se puede ingresar una cantidad negativa");
        iCodErr = 1;
    } else {
        dSaldo += cantidad;
        iCodErr = 0;
    }
    return iCodErr;
}

/* Método para retirar cantidades en la cuenta. Modifica el saldo.
   Esta método no es recomendado para clientes.
*/
public void retirar (double cantidad) {
    if (cantidad <= 0) {
        System.out.println("No se puede retirar una cantidad negativa");
    }
}

```

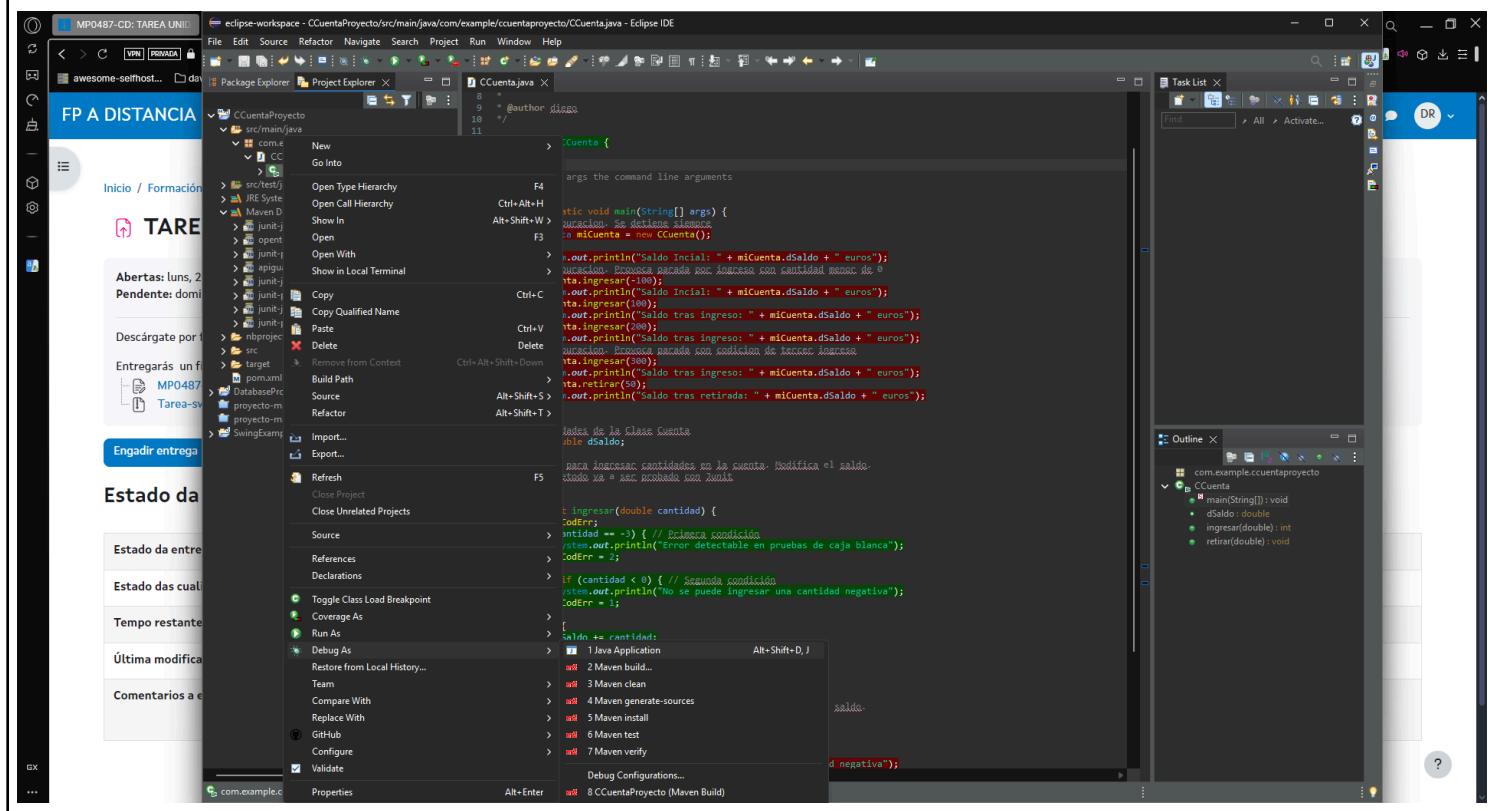


Type:	com.example.cuentaproyecto.CCuenta
Line Number:	56
Member:	ingresar(double)
Enabled:	<input checked="" type="checkbox"/>
Trigger Point:	<input type="checkbox"/> Breakpoint <input checked="" type="radio"/> Hit count: 3 <input type="radio"/> Suspend thread <input type="radio"/> Suspend VM <input type="checkbox"/> Conditional <input type="checkbox"/> Suspend when 'true' <input type="checkbox"/> Suspend when value changes



Iniciar la depuración.

Clic derecho sobre la clase > Debug as > Java Application. Para saltar entre breakpoints pulsar el botón Resume.



**Eclipse IDE - CCuenta.java**

```


1 * Click http://www.eclipse.org/SystemFileSystem/Templates/Licenses/license-default.txt to change this license.
2 * Click http://maven.apache.org/LICENSE-2.0.html to change this license.
3 * 
4 * Copyright 2025 by díazgo
5 * 
6 * Licensed under the Apache License, Version 2.0 (the "License");
7 * you may not use this file except in compliance with the License.
8 * You may obtain a copy of the License at
9 * 
10 *     http://www.apache.org/licenses/LICENSE-2.0
11 * 
12 * Unless required by applicable law or agreed to in writing, software
13 * distributed under the License is distributed on an "AS IS" BASIS,
14 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 * See the License for the specific language governing permissions and
16 * limitations under the License.
17 * 
18 * Contributors:
19 *     díazgo - initial API and implementation
20 * 

```

**Variables**

Name	Value
no method return value	
args	String[0] (id=20)

**Console**

```

CCuenta [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (8 mar 2025, 20:22:40 elapsed: 0:00:48) [pid: 2823]
Saldo Inicial: 0.0 euros

```

**Eclipse IDE - CCuenta.java**

```


33 miCuenta.retirar(50);
34 System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
35 
36 // Propiedades de la Clase Cuenta
37 public double dSaldo;
38 
39 /* Método para ingresar cantidades en la cuenta. Modifica el saldo.
40 * Esta método no es a ser ejecutado con null
41 */
42 
43 public int ingresar(double cantidad) {
44     int iCodErr;
45 
46     if (cantidad == -3) { // Primera condición
47         System.out.println("Error detectable en pruebas de caja blanca");
48         iCodErr = 2;
49     } 
50     else if (cantidad < 0) { // Segunda condición
51         System.out.println("No se puede ingresar una cantidad negativa");
52         iCodErr = 1;
53     }
54     else {
55         dSaldo += cantidad;
56         iCodErr = 0;
57     }
58     return iCodErr;
59 }
60 
61 /* Método para retirar cantidades en la cuenta. Modifica el saldo.
62 * Esta método no es a ser ejecutado con null
63 */
64 public void retirar(double cantidad) {
65     if (cantidad <= 0) {
66         System.out.println("No se puede retirar una cantidad negativa");
67     } 
68     else if (dSaldo < cantidad) {
69         System.out.println("No se hay suficiente saldo");
70     }
71     else {
72         dSaldo = dSaldo - cantidad;
73     }
74 }
75 

```

**Variables**

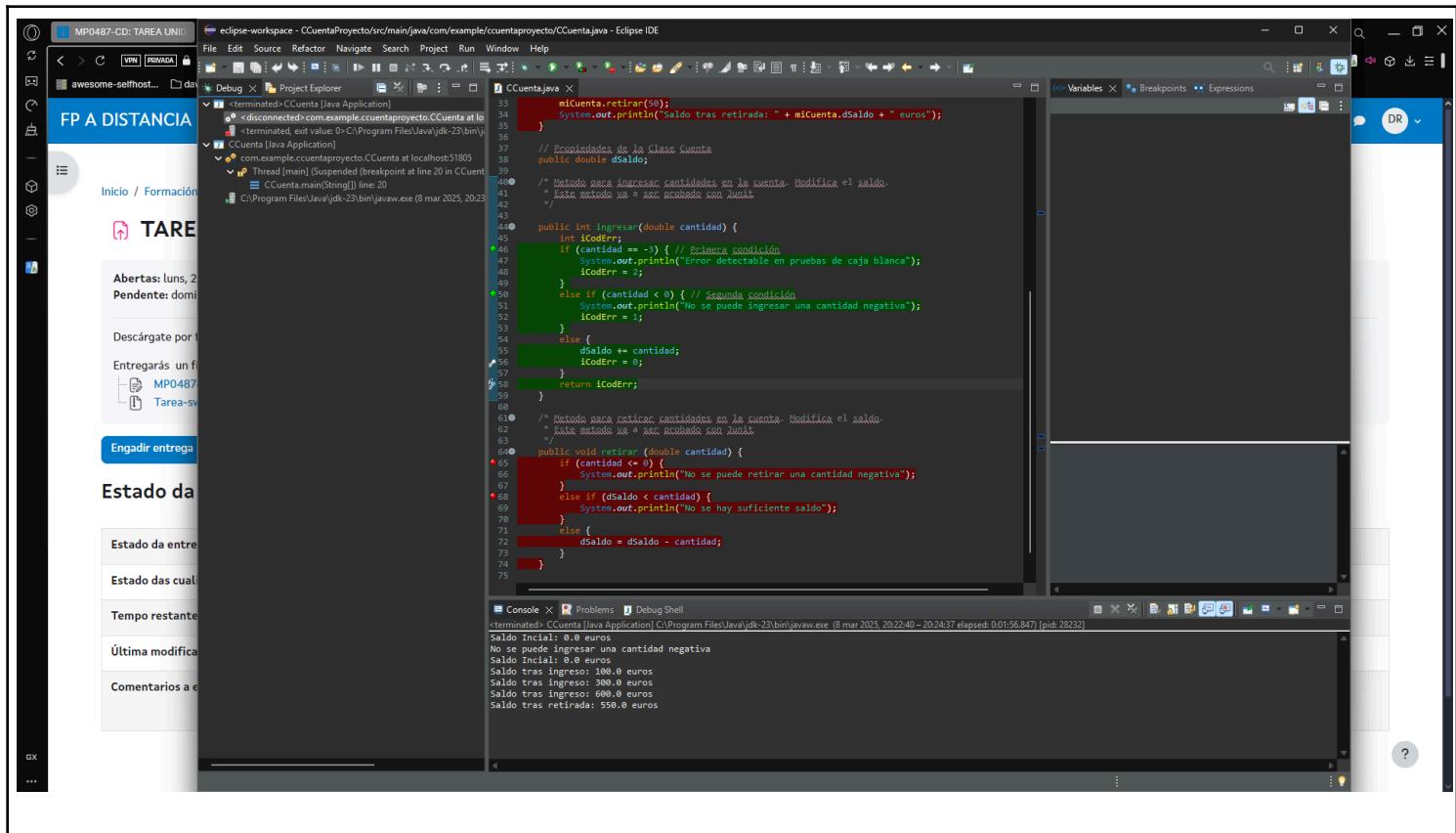
Name	Value
no method return value	
this	CCuenta (id=23)
cantidad	-100.0
iCodErr	1

**Console**

```

CCuenta [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (8 mar 2025, 20:22:40 elapsed: 0:01:39) [pid: 2823]
Saldo Inicial: 0.0 euros
No se puede ingresar una cantidad negativa

```



```

33     miCuenta.retirar(50);
34     System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
35 }
36
37 // Modificación de la Clase Cuenta
38 public double dSaldo;
39
40 /* Método para ingresar cantidades en la cuenta. Modifica el saldo.
41 * Esta método no a este resultado son Añadir
42 */
43
44 public int ingresar(double cantidad) {
45     int iCodeErr;
46     if (cantidad == ->) { // Primera condición
47         System.out.println("Error detectable en pruebas de caja blanca");
48         iCodeErr = 2;
49     }
50     else if (cantidad < 0) { // Segunda condición
51         System.out.println("No se puede ingresar una cantidad negativa");
52         iCodeErr = 1;
53     }
54     else {
55         dSaldo += cantidad;
56         iCodeErr = 0;
57     }
58     return iCodeErr;
59 }
60
61 /* Método para retirar cantidades en la cuenta. Modifica el saldo.
62 * Esta método no a este resultado son Añadir
63 */
64 public void retirar (double cantidad) {
65     if (cantidad <= 0) {
66         System.out.println("No se puede retirar una cantidad negativa");
67     }
68     else if (dSaldo < cantidad) {
69         System.out.println("No se hay suficiente saldo");
70     }
71     else {
72         dSaldo = dSaldo - cantidad;
73     }
74 }
75

```

Console output:

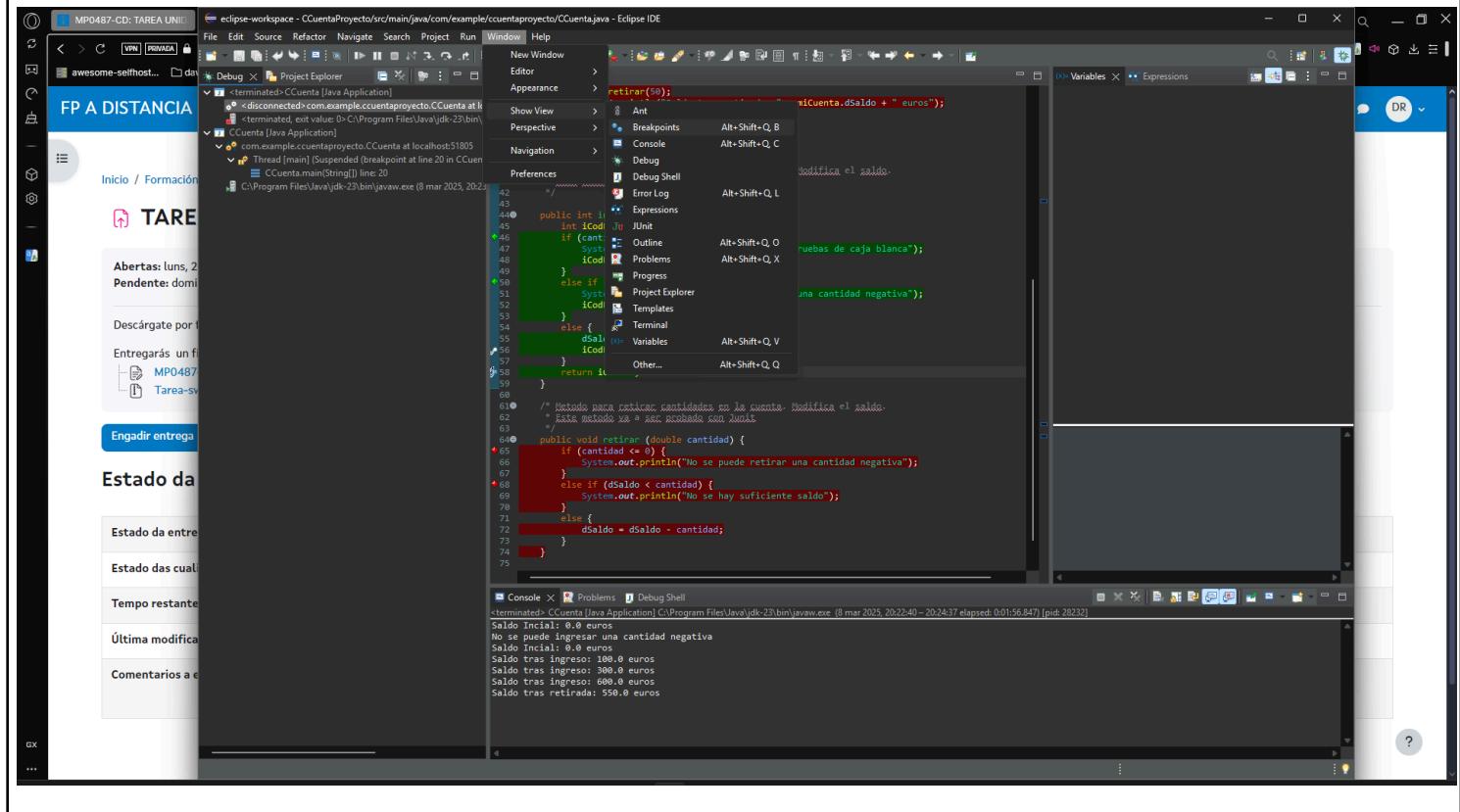
```

Saldo inicial: 0.0 euros
No se puede ingresar una cantidad negativa
Saldo Inicial: 0.0 euros
Saldo tras ingreso: 100.0 euros
Saldo tras ingreso: 300.0 euros
Saldo tras ingreso: 600.0 euros
Saldo tras retirada: 550.0 euros

```

### Información del fichero de los puntos de ruptura (breakpoints)

Acceder a Window > Show View > Breakpoints, seleccionar los que queremos exportar, clic derecho y pulsar en Export Breakpoints.



New Window

Show View

Perspective

Navigation

Preferences

IntelliJ IDEA

Breakpoints

Alt+Shift+Q, B

Console

Debug Shell

Error Log

Alt+Shift+Q, L

JUnit

Outline

Alt+Shift+Q, O

Problems

Alt+Shift+Q, X

Progress

Project Explorer

Alt+Shift+Q, P

Templates

Terminal

Variables

Alt+Shift+Q, V

Other...

Alt+Shift+Q, Q

```

42     /*
43      * Modificación de la Clase Cuenta
44      * Esta método no a este resultado son Añadir
45      */
46     public int ingresar(double cantidad) {
47         int iCodeErr;
48         if (cantidad == ->) { // Primera condición
49             System.out.println("Error detectable en pruebas de caja blanca");
50             iCodeErr = 2;
51         }
52         else if (cantidad < 0) { // Segunda condición
53             System.out.println("No se puede ingresar una cantidad negativa");
54             iCodeErr = 1;
55         }
56         else {
57             dSaldo += cantidad;
58             iCodeErr = 0;
59         }
60         return iCodeErr;
61     }
62
63     /* Método para retirar cantidades en la cuenta. Modifica el saldo.
64     * Esta método no a este resultado son Añadir
65     */
66     public void retirar (double cantidad) {
67         if (cantidad <= 0) {
68             System.out.println("No se puede retirar una cantidad negativa");
69         }
70         else if (dSaldo < cantidad) {
71             System.out.println("No se hay suficiente saldo");
72         }
73         else {
74             dSaldo = dSaldo - cantidad;
75         }
76     }
77

```

Console output:

```

Saldo inicial: 0.0 euros
No se puede ingresar una cantidad negativa
Saldo Inicial: 0.0 euros
Saldo tras ingreso: 100.0 euros
Saldo tras ingreso: 300.0 euros
Saldo tras ingreso: 600.0 euros
Saldo tras retirada: 550.0 euros

```

Eclipse IDE - CCuentaProyecto/src/main/java/com/example/cuentaproyecto/CCuenta.java - Eclipse IDE

```

33     miCuenta.retirar(50);
34     System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
35 }
36
37 // Encapsulada de la Clase Cuenta
38 public double dSaldo;
39
40 /* Método para ingresar cantidades en la cuenta. Modifica el saldo.
41 * Esta método no es así llamado con Atributo
42 */
43
44 public int ingresar(double cantidad) {
45     int iCodErr = 0;
46     if (cantidad == ->) { // Revisa condición
47         System.out.println("Error detectable en pruebas de caja blanca");
48         iCodErr = 2;
49     }
50     else if (cantidad < 0) { // Segunda condición
51         System.out.println("No se puede ingresar una cantidad negativa");
52         iCodErr = 1;
53     }
54     else {
55         dSaldo += cantidad;
56         iCodErr = 0;
57     }
58     return iCodErr;
59 }
60
61 /* Método para retirar cantidades en la cuenta. Modifica el saldo.
62 * Esta método no es así llamado con Atributo
63 */
64 public void retirar (double cantidad) {
65     if (cantidad <= 0) {
66         System.out.println("No se puede retirar una cantidad negativa");
67     }
68     else if (dSaldo < cantidad) {
69         System.out.println("No se hay suficiente saldo");
70     }
71     else {
72         dSaldo = dSaldo - cantidad;
73     }
74 }
75

```

Variables Breakpoints Expressions

- CCuenta [line: 20] - main(String[])
- CCuenta [line: 56] [hit count: 3] - ingresar(double)
- CCuenta [line: 58] [conditional] - retirar(double)

No details to display for the current selection.

Debug Project Explorer Problems Debug Shell

Saldo Inicial: 0 euros  
 No se puede ingresar una cantidad negativa  
 Saldo Inicial: 0 euros  
 Saldo tras ingreso: 100.0 euros  
 Saldo tras ingreso: 300.0 euros  
 Saldo tras ingreso: 600.0 euros  
 Saldo tras retirada: 550.0 euros

Eclipse IDE - CCuentaProyecto/src/main/java/com/example/cuentaproyecto/CCuenta.java - Eclipse IDE

```

33     miCuenta.retirar(50);
34     System.out.println("Saldo tras retirada: " + miCuenta.dSaldo + " euros");
35 }
36
37 // Propiedad dSaldo
38 // Export Breakpoints
39 // Método para ingresar cantidades en la cuenta. Modifica el saldo.
40 // Esta método no es así llamado con Atributo
41
42 public int ingresar(double cantidad) {
43     int iCodErr = 0;
44     if (cantidad == ->) { // Revisa condición
45         System.out.println("Error detectable en pruebas de caja blanca");
46         iCodErr = 2;
47     }
48     else if (cantidad < 0) { // Segunda condición
49         System.out.println("No se puede ingresar una cantidad negativa");
50         iCodErr = 1;
51     }
52     else {
53         dSaldo += cantidad;
54         iCodErr = 0;
55     }
56     return iCodErr;
57 }
58
59 /* Método para retirar cantidades en la cuenta. Modifica el saldo.
60 * Esta método no es así llamado con Atributo
61 */
62 public void retirar (double cantidad) {
63     if (cantidad <= 0) {
64         System.out.println("No se puede retirar una cantidad negativa");
65     }
66     else if (dSaldo < cantidad) {
67         System.out.println("No se hay suficiente saldo");
68     }
69     else {
70         dSaldo = dSaldo - cantidad;
71     }
72 }
73
74

```

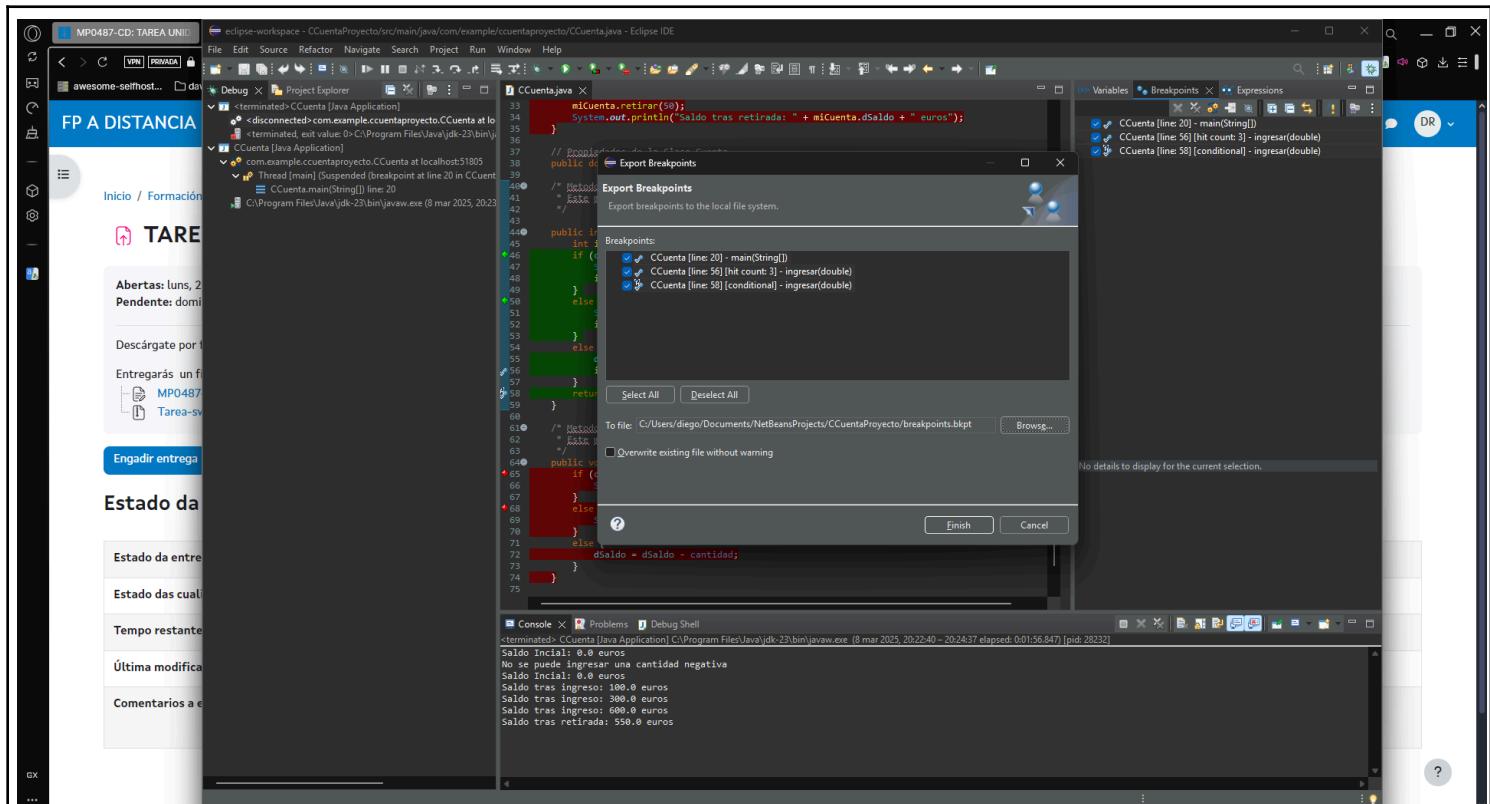
Variables Breakpoints Expressions

- CCuenta [line: 20] - main(String[])
- CCuenta [line: 56] [hit count: 3] - ingresar(double)
- CCuenta [line: 58] [conditional] - retirar(double)

No details to display for the current selection.

Debug Project Explorer Problems Debug Shell

Saldo Inicial: 0 euros  
 No se puede ingresar una cantidad negativa  
 Saldo Inicial: 0 euros  
 Saldo tras ingreso: 100.0 euros  
 Saldo tras ingreso: 300.0 euros  
 Saldo tras ingreso: 600.0 euros  
 Saldo tras retirada: 550.0 euros



Revisar el contenido del fichero exportado.

```
<?xml version="1.0" encoding="UTF-8"?>
<breakpoints>
<breakpoint enabled="true" persistant="true" registered="true">
<resource path="/CCuentaProyecto/src/main/java/com/example/cccuentaproyecto/CCuenta.java" type="1"/>
<marker charStart="352" lineNumber="20" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="352"/>
<attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="--CCuentaProyecto/src/main/java=optional=/true=/maven.pomderived=true=/&lt;com.example.cccuentaproyecto(CCuenta.java[CCuenta])</attrib>
<attrib name="charEnd" value="393"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
<attrib name="message" value="Line breakpoint:CCuenta [line: 20] - main(String[])"/>
<attrib name="org.eclipse.jdt.debug.core.installCount" value="1"/>
<attrib name="org.eclipse.debug.core.id" value="org.eclipse.jdt.debug"/>
<attrib name="org.eclipse.debug.core.typeName" value="com.example.cccuentaproyecto.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="/CCuentaProyecto/src/main/java/com/example/cccuentaproyecto/CCuenta.java" type="1"/>
<marker charStart="1796" lineNumber="56" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="1796"/>
<attrib name="org.eclipse.jdt.debug.core.suspendPolicy" value="2"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="--CCuentaProyecto/src/main/java=optional=/true=/maven.pomderived=true=/&lt;com.example.cccuentaproyecto(CCuenta.java[CCuenta])</attrib>
<attrib name="org.eclipse.jdt.debug.core.hitCount" value="3"/>
<attrib name="charEnd" value="1814"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
<attrib name="org.eclipse.jdt.debug.core.expired" value="false"/>
<attrib name="message" value="Line breakpoint:CCuenta [line: 56] [hit count: 3] - ingresar(double)"/>
<attrib name="org.eclipse.jdt.debug.core.installCount" value="1"/>
<attrib name="org.eclipse.debug.core.id" value="org.eclipse.jdt.debug"/>
<attrib name="org.eclipse.jdt.debug.core.typeName" value="com.example.cccuentaproyecto.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="/CCuentaProyecto/src/main/java/com/example/cccuentaproyecto/CCuenta.java" type="1"/>
<marker charStart="1825" lineNumber="58" type="org.eclipse.jdt.debug.javaLineBreakpointMarker">
<attrib name="charStart" value="1825"/>
<attrib name="org.eclipse.jdt.debug.core.conditionEnabled" value="true"/>
<attrib name="org.eclipse.jdt.debug.core.condition" value="cantidad &lt; 0"/>
<attrib name="org.eclipse.jdt.debug.ui.JAVA_ELEMENT_HANDLE_ID" value="--CCuentaProyecto/src/main/java=optional=/true=/maven.pomderived=true=/&lt;com.example.cccuentaproyecto(CCuenta.java[CCuenta])</attrib>
<attrib name="charEnd" value="1848"/>
<attrib name="org.eclipse.debug.core.enabled" value="true"/>
</marker>
</breakpoint>

```

**Criterios de puntuación. Total 10 puntos**

Los criterios de puntuación serán los siguientes.

Pruebas de caja blanca. 2,5 puntos.

Pruebas de caja negra. 2,5 puntos.

JUnit en 2,5 puntos.

Depuración. 2,5 puntos.