1. **Análise de caixa branca (llenarPruebas)**
-- Grafo en papel

Método de cálculo	Complexidade	Comentario
Número de rexións	3	Áreas A, B, C
n.aristas – n.nodos +2	11 – 10 + 2 = 3	
n.condicións + 1	2 + 1 = 3	

Camiño	Nodos
CAM1	1, 2, 3, 4, 9, 10
CAM2	1, 2, 3, 5, 6, 9, 10
CAM3	1, 2, 3, 5, 7, 8, 9, 10

2. **Análise de caixa negra (quitarPruebas)**

Tipo de dato	Descripción	Clases	Valor límite
Rango de valores de entrada	O número non poderá ser negativo nin maior a 100	Clases válidas	
		Valor entre 1 e 100	ALV1: 1 ALV2: 100
		Clases non válidas	
		Número negativo	ALV3: -1
		Maior de 100	ALV4: 101

Conxetura de erros
- CE1: non ten sentido, a nivel mundo real, engadir ou eliminar 0 litros dun depósito. Debemos comprobar que a mensaxe de erro sexa correcta
- CE2: hai que ter en mente que no canto de números se introduzan letras

3. **Junit.**

Como se nos pide que sexa para caixa branca, creamos tres probas para asegurar que pase polos 3 camiños

ID Caso de prueba	Entrada	Camiño	Saída
C1	-1	CAM1	IcodErr = 1
C2	-7	CAM2	IcodErr = 1 *
C3	2	CAM3	IcodErr = 0

* Importante: facendo as probas, comprobamos que nunca vai pasar polo camiño `CAM2`, xa que o introducires `-7`, xa valida no primeiro `if`. É por iso que comprobamos que o que devolve `iCodErr` sexa `1`: efectivamente, devolve ese número

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import deposito.CDeposito;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Iván Estévez
 */
public class CDepositoTest {

    CDeposito deposito;

    public CDepositoTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
        System.out.println("*** Iniciando prueba ***");
        deposito = new CDeposito();
    }

    @After
    public void tearDown() {
    }

    /**
     * Test C1 (CAM1)
     */
    @Test
    public void testC1() {
        double entrada = -1;
        double resultado = deposito.llenarPruebas(entrada);
        assertEquals(1, resultado, 0);
    }

    /**
     * Test C2 (CAM2*)
     */
    @Test
    public void testC2() {
        double entrada = -7;
```

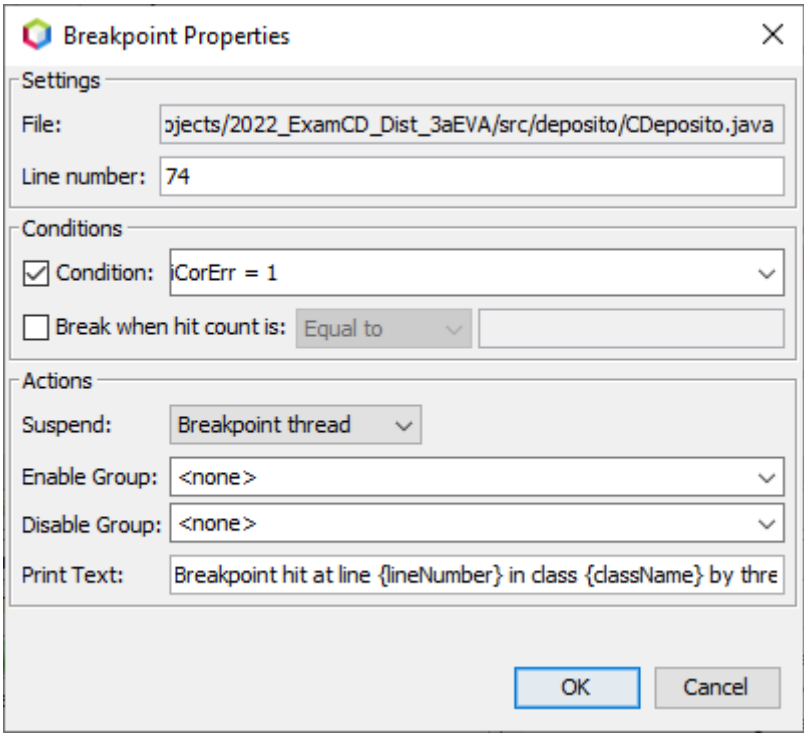
```
        double resultado = deposito.llenarPruebas(entrada);
        assertEquals(1, resultado, 0);

        /**
         * Importante: facendo as probas, comprobamos que nunca vai pasar polo
         * camiño CAM2, xa que o introducires -7, xa valida no primeiro if. É
         * por iso que comprobamos que o que devolve iCodErr sexa 1:
         * efectivamente, devolve ese número
         */
    }

    /**
     * Test C3 (CAM3)
     */
    @Test
    public void testC3() {
        double entrada = 2;
        double resultado = deposito.llenarPruebas(entrada);
        assertEquals(0, resultado, 0);
    }

}
```

4. **Puntos de interrupción**

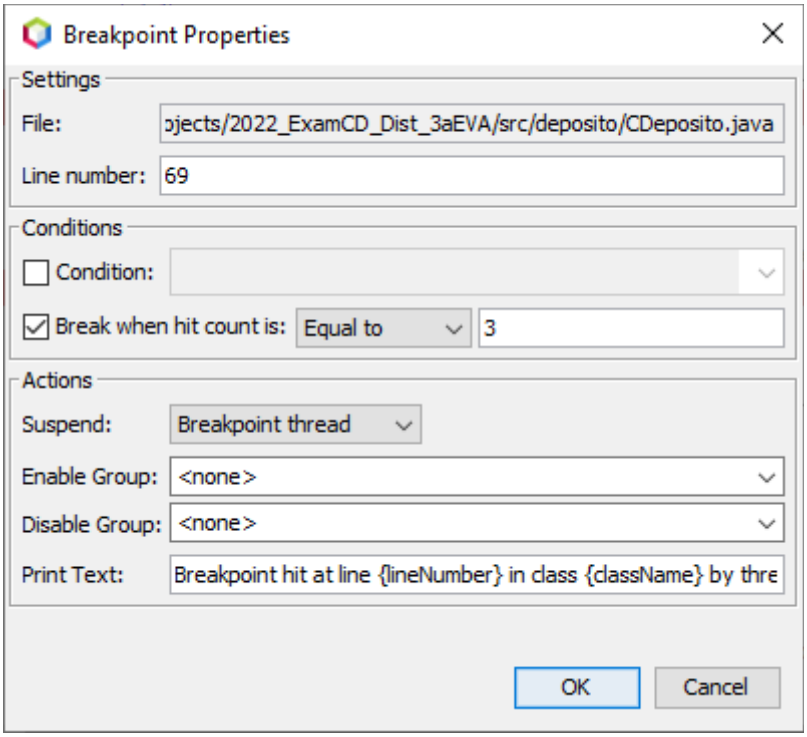
- Punto de parada sin condición al crear el objeto miDeposito en la función main.

```
13      public static void depuracion_Pruebas() throws Exception {
14          // Depuracion. Se detiene siempre
            CDeposito depositol = new CDeposito();
16
```

- Punto de parada en la instrucción return del método llenarPruebas sólo si la cantidad a ingresar es menor de 0.

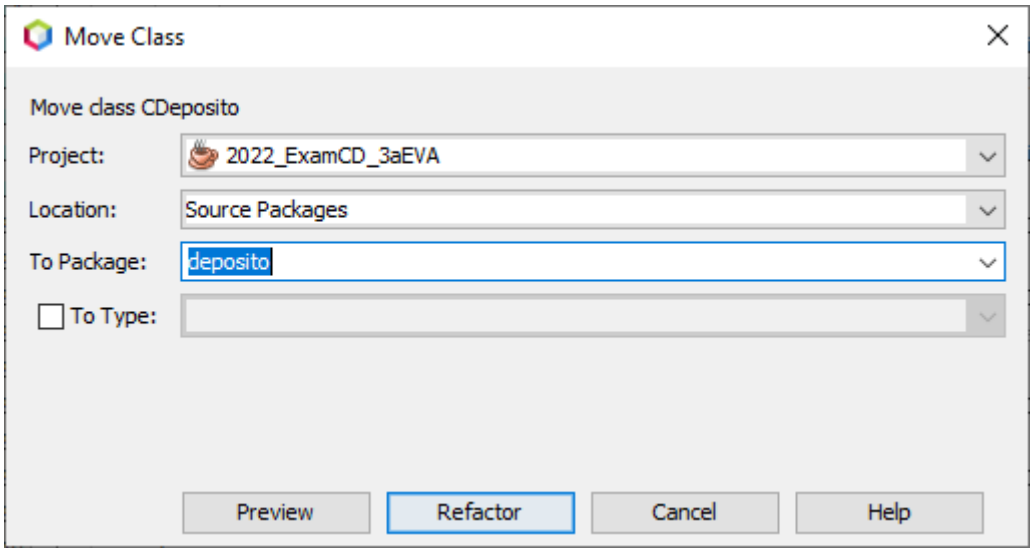


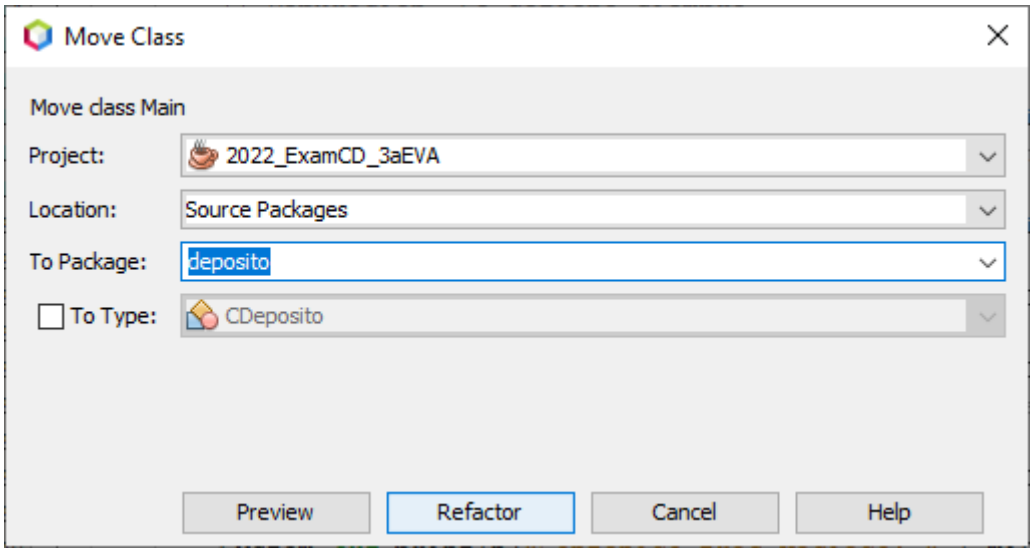
- Punto de parada en la instrucción donde se actualiza el saldo de litros, sólo deberá parar la tercera vez que sea actualizado.



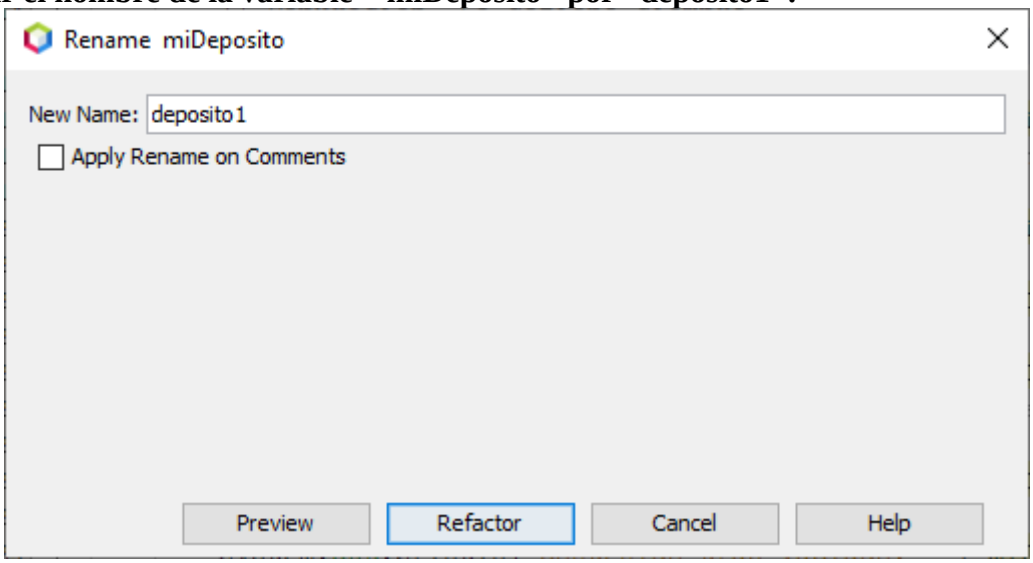
5. Refactorización

- Las clases deberán formar parte del paquete deposito.

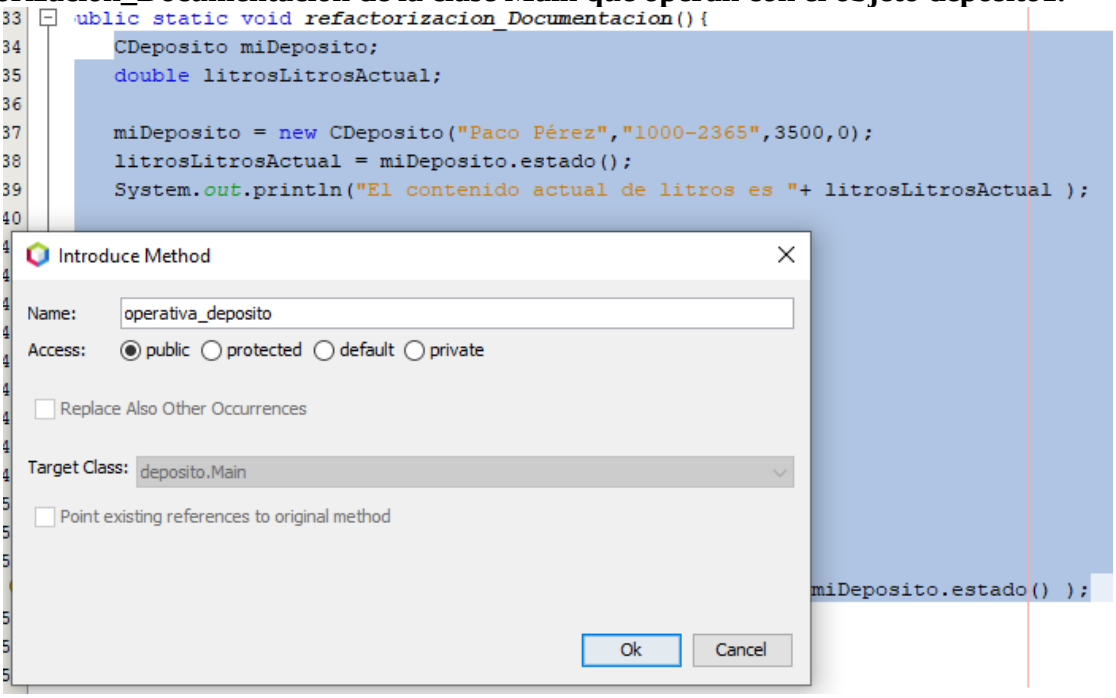




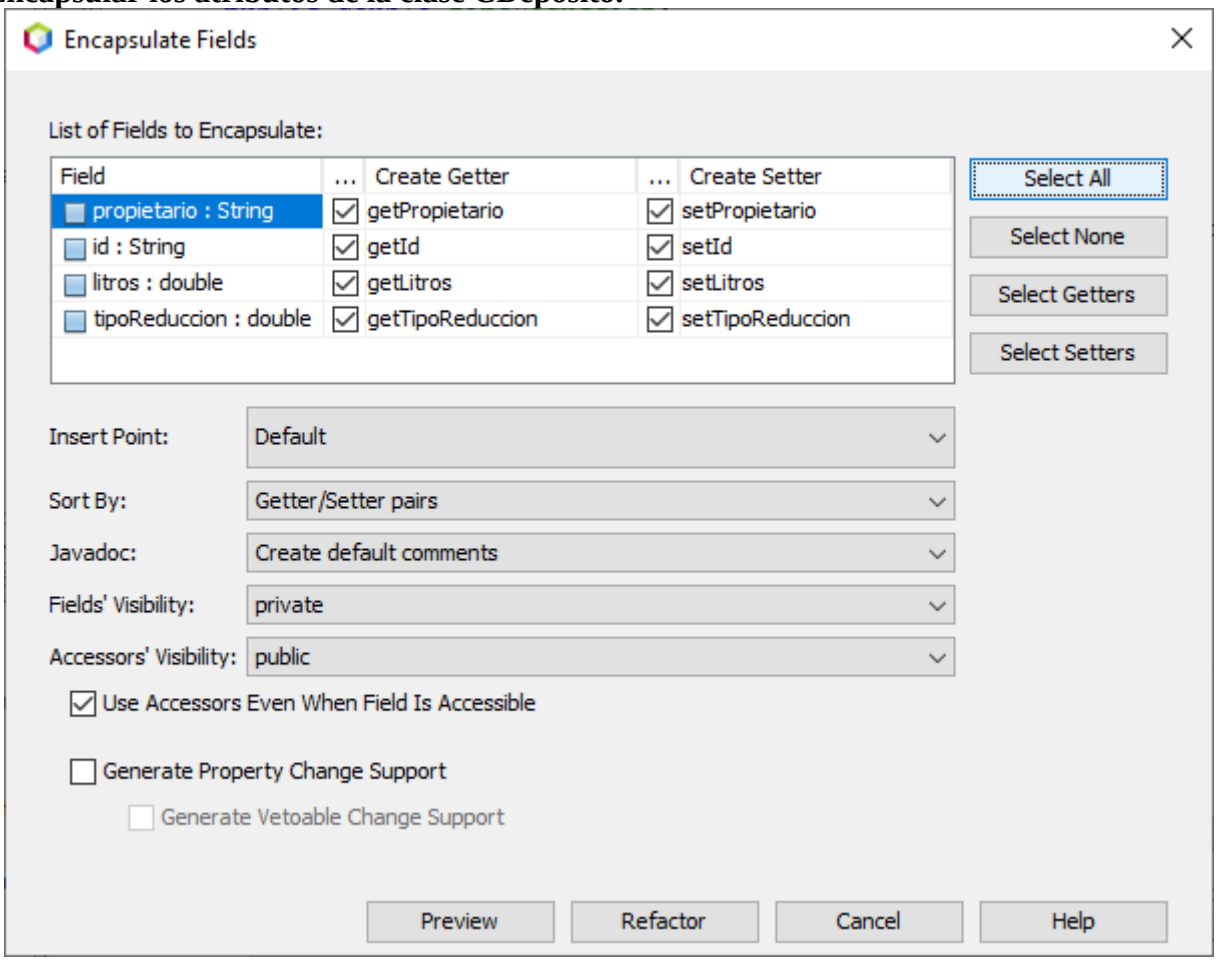
- Cambiar el nombre de la variable " miDeposito" por "deposito1".



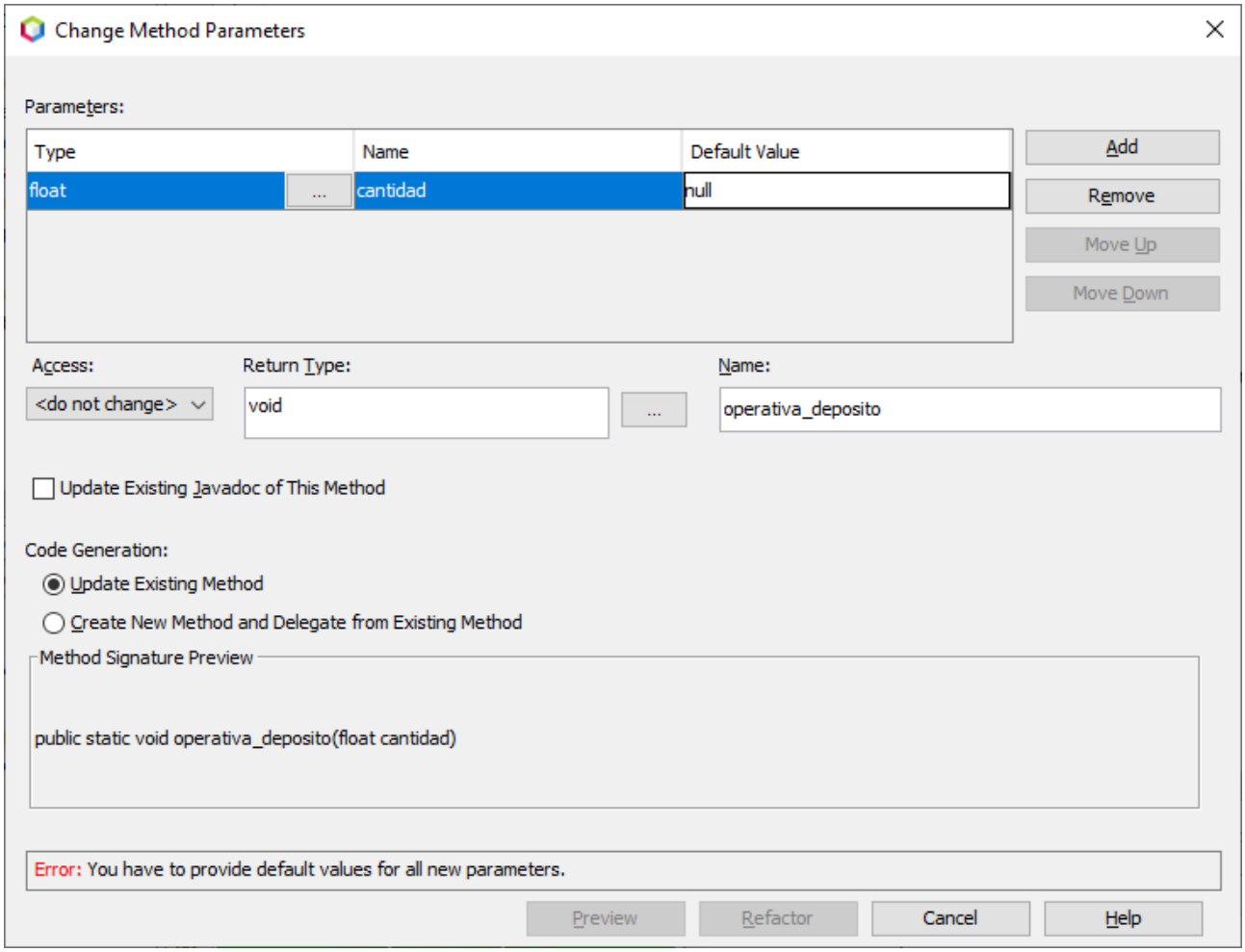
- Introducir el método operativa_deposito, que englobe las sentencias del método refactorizacion_Documentacion de la clase Main que operan con el objeto deposito1.



- Encapsular los atributos de la clase CDeposito.



- Añadir un nuevo parámetro al método operativa_deposito, de nombre cantidad y de tipo float.




```java
public static void refactorizacion_Documentacion(float cantidad) {
    operativa_deposito(cantidad);
}

public static void operativa_deposito(float cantidad) {
    CDeposito miDeposito;
    double litrosLitrosActual;

    miDeposito = new CDeposito("Paco Pérez", "1000-2365", 3500, 0)
    litrosLitrosActual = miDeposito.estado();
    System.out.println("El contenido actual de litros es " + litro

    try {
        miDeposito.quitar(cantidad);
    } catch (Exception e) {
        System.out.print("Fallo al retirar líquido");
    }
    try {
        System.out.println("Llenado en depósito");
        miDeposito.llenar(cantidad);
```

6. Javadoc

Directamente no programa