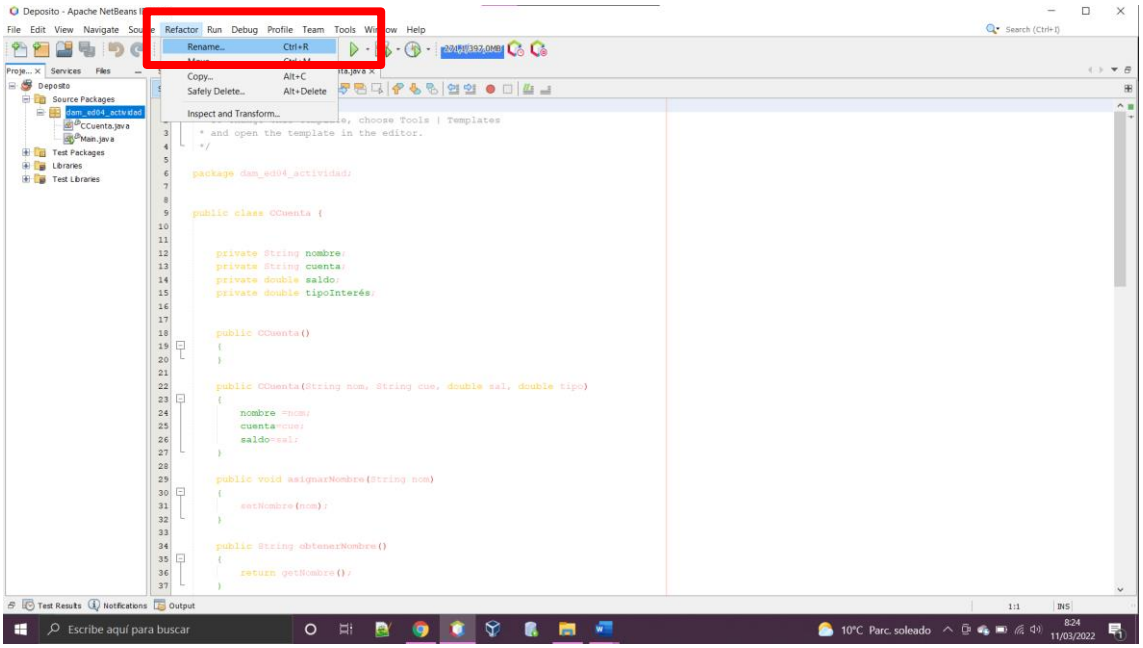
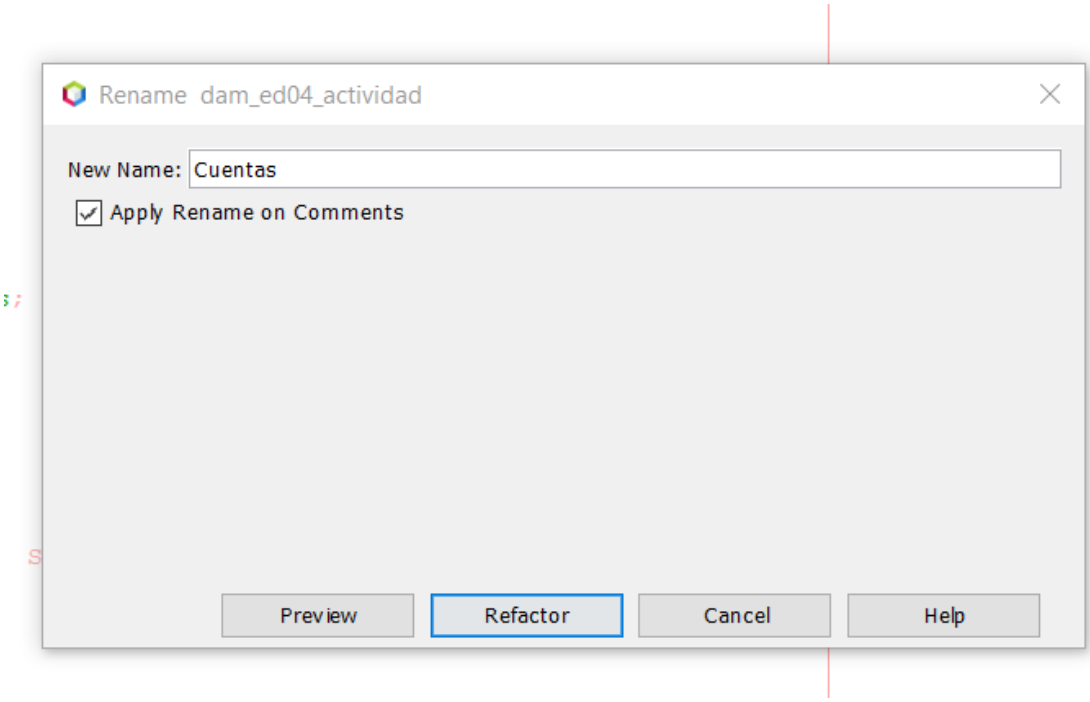
1. Las clases deberán formar parte del paquete cuentas.

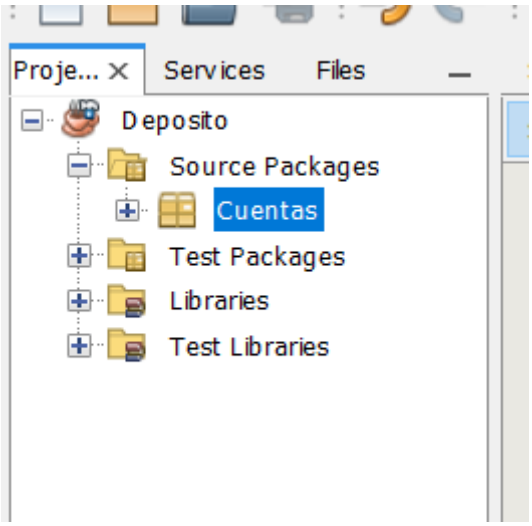
Para renombrar el paquete a Cuentas, acudimos al menú Refactor, con el paquete que queremos renombrar seleccionado, y pulsamos en Rename o utilizamos Ctrl + R:



Escribimos el nuevo nombre y aplicamos pulsando Refactor:



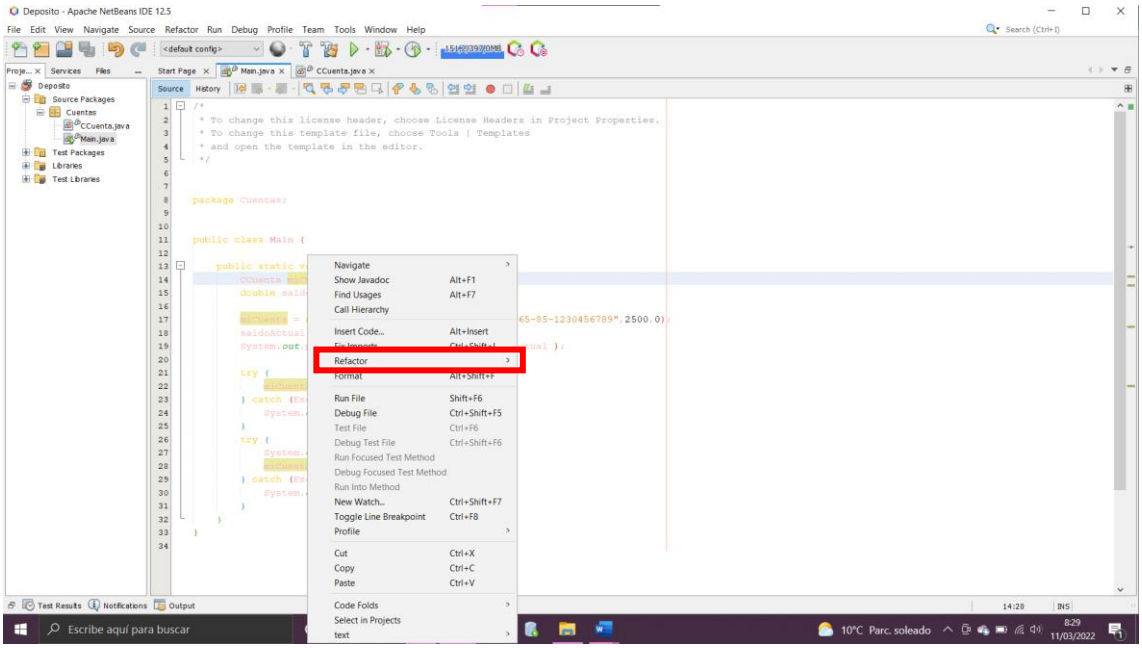
El nombre del paquete se ha modificado y al igual que en las clases:



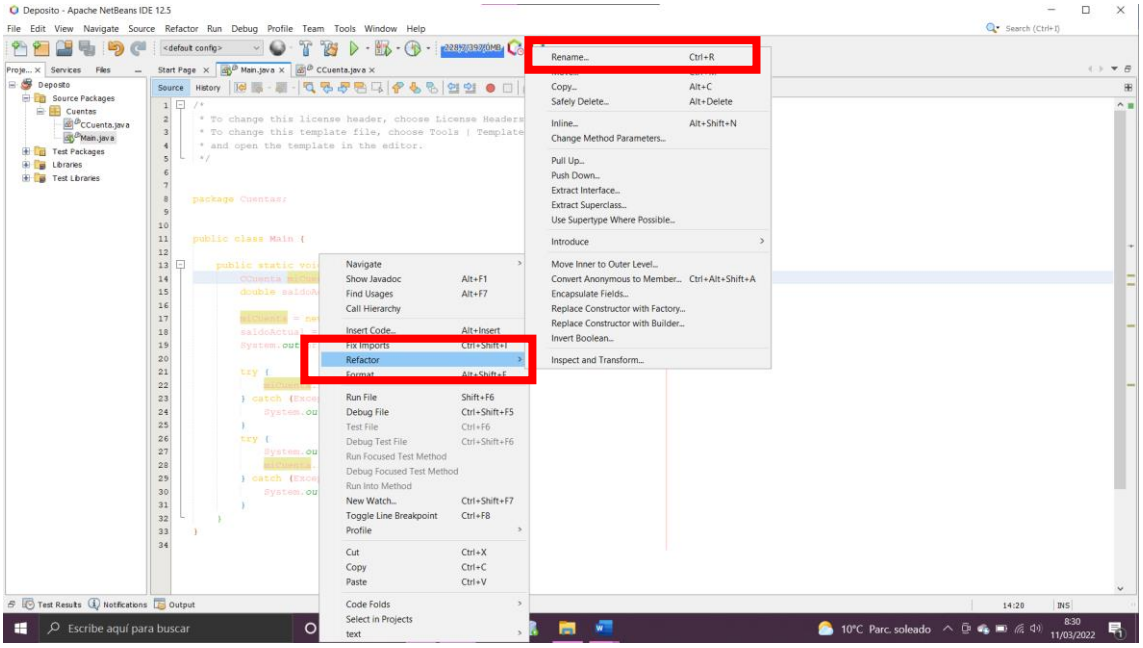

```
4   */
5
6   package Cuentas;
7
8
9   public class CCuenta {
10
11
```

2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".

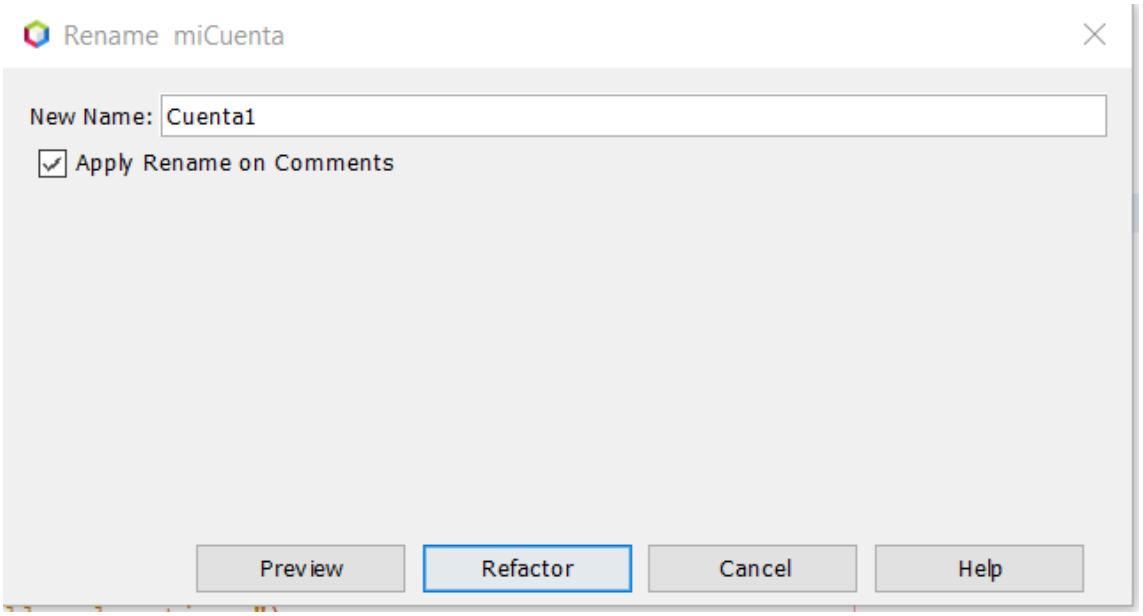
Para cambiar el nombre de la variable, tan solo pulsamos el botón derecho sobre el nombre de la variable. En el menú que nos aparece, acudimos a Refactor:



En el menú de Refactor pulsamos en Rename o, como en el caso anterior, podemos usar directamente Ctrl + R:



Escribimos el nuevo nombre de la variable y aplicamos pulsando Refactor:

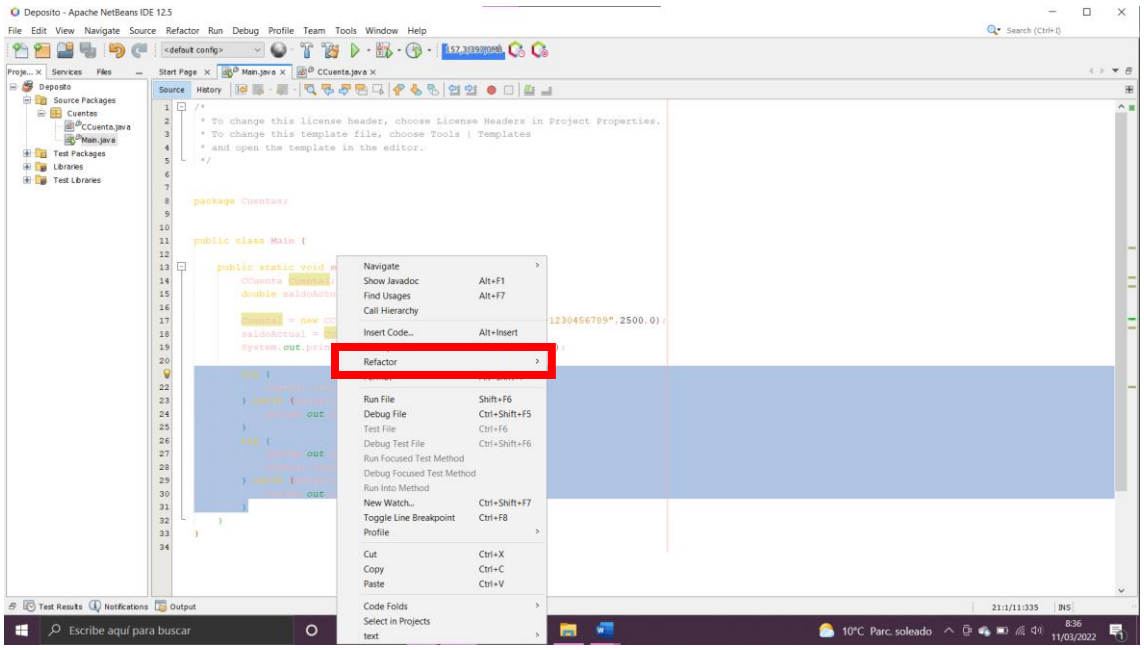


Se ha cambiado el nombre de la variable en todo el código:

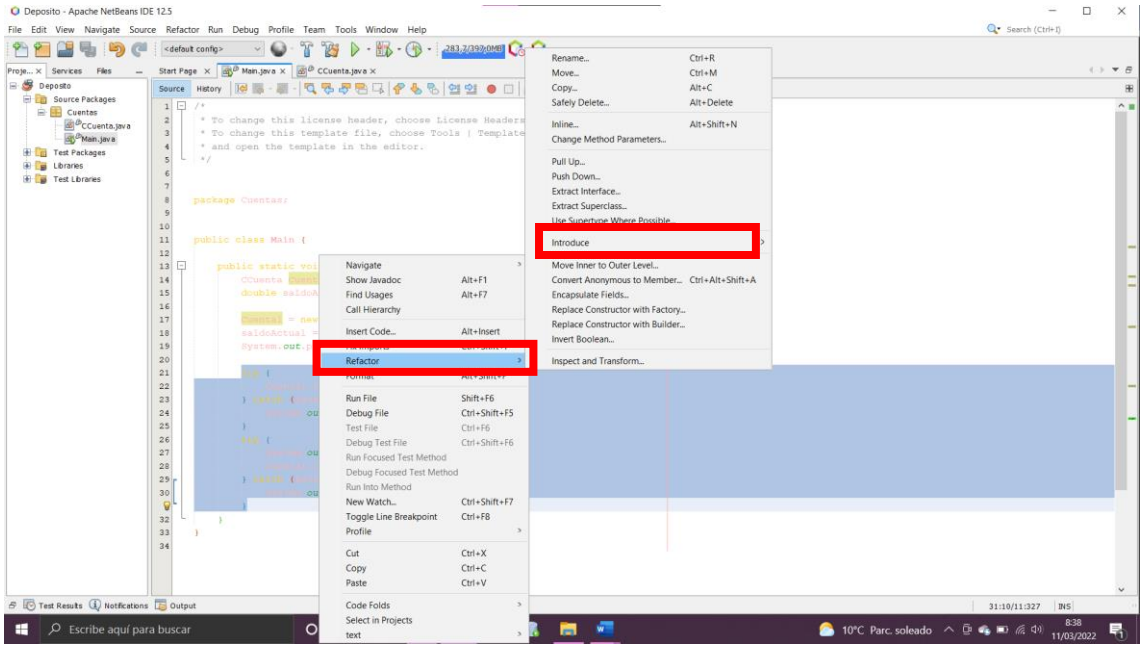
```
10
11  public class Main {
12
13      public static void main(String[] args) {
14          CCuenta Cuenta1;
15          double saldoActual;
16
17          Cuenta1 = new CCuenta("Antonio López","1000-2365-85-1230456789",2500,0);
18          saldoActual = Cuenta1.estado();
19          System.out.println("El saldo actual es"+ saldoActual );
20
21          try {
22              Cuenta1.retirar(2300);
23          } catch (Exception e) {
24              System.out.print("Fallo al retirar");
25          }
26          try {
27              System.out.println("Ingreso en cuenta");
28              Cuenta1.ingresar(695);
29          } catch (Exception e) {
30              System.out.print("Fallo al ingresar");
31          }
32      }
33  }
34
```

3. Introducir el método operativa_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.

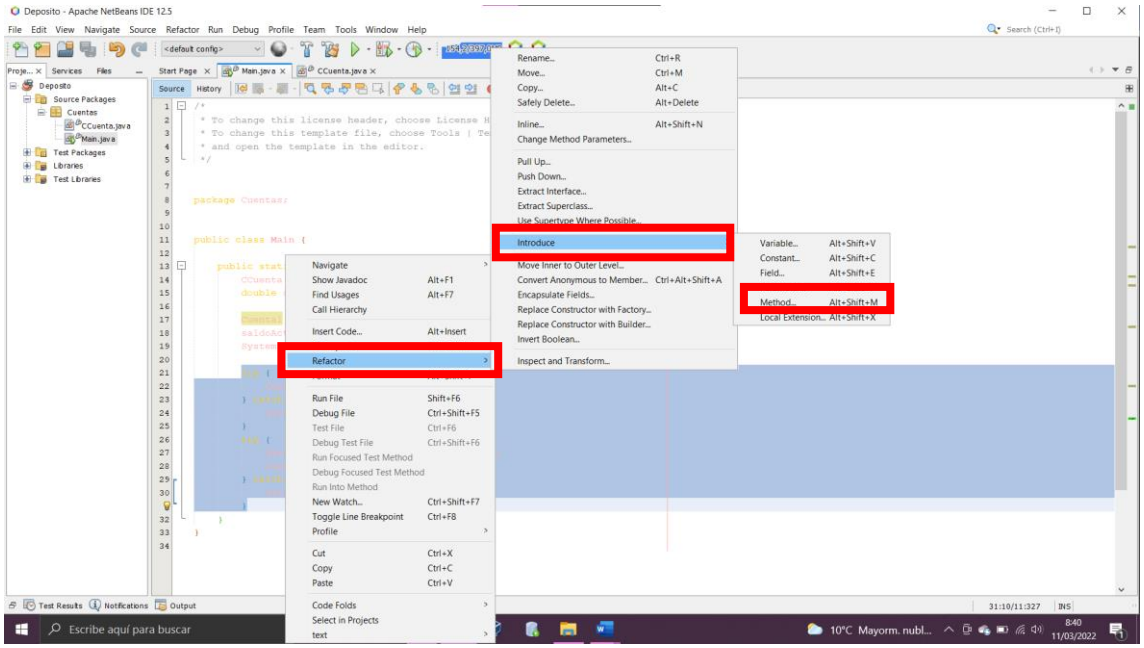
Para introducir las sentencias en un nuevo método, seleccionamos las sentencias que queremos incluir y pulsamos de nuevo en el menú Refactor:



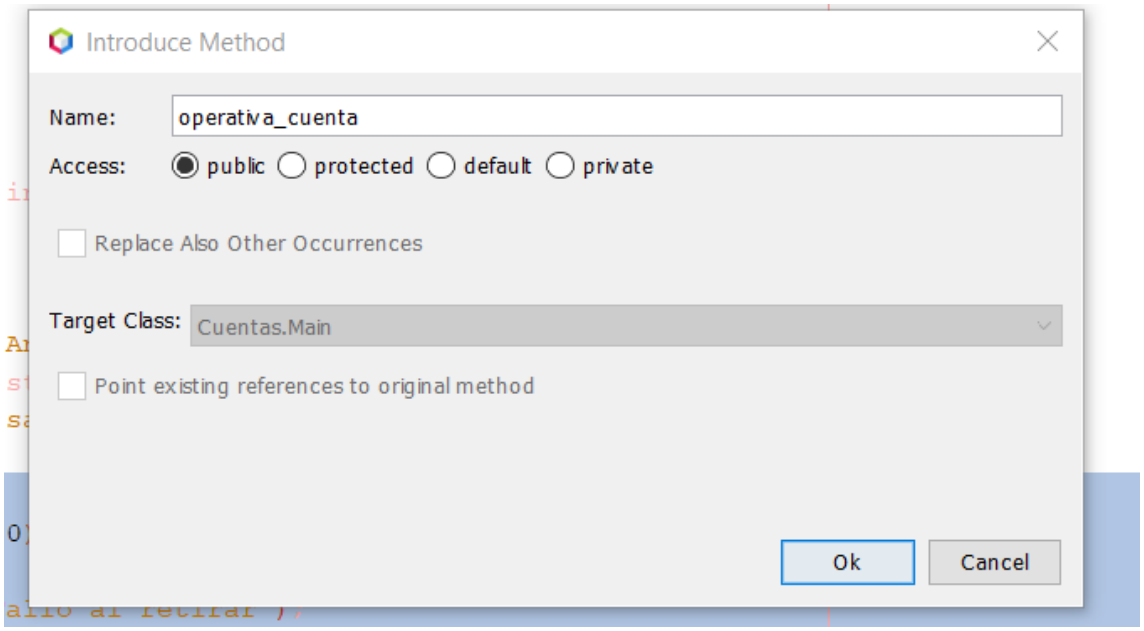
En ese menú, seleccionamos Introduce:



Finalmente pulsamos sobre Method o usamos Alt + Shift + M:



Escogemos el nombre del método, en este caso operativa_cuenta, y el acceso, en este caso, público:

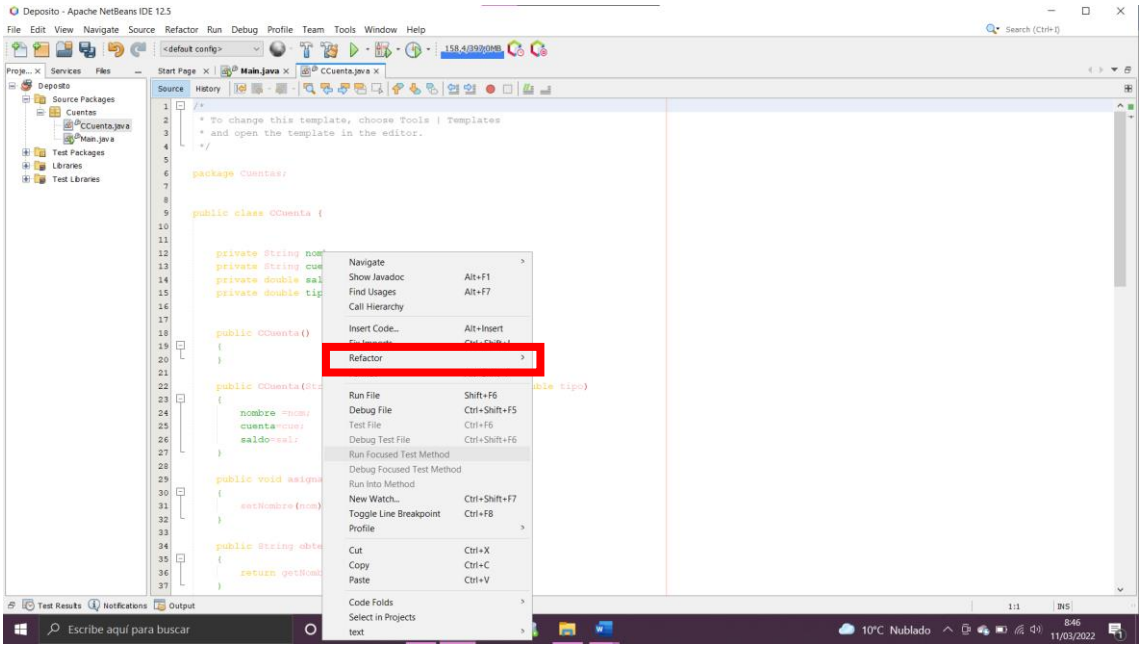


Se ha creado el método operativa_cuenta:

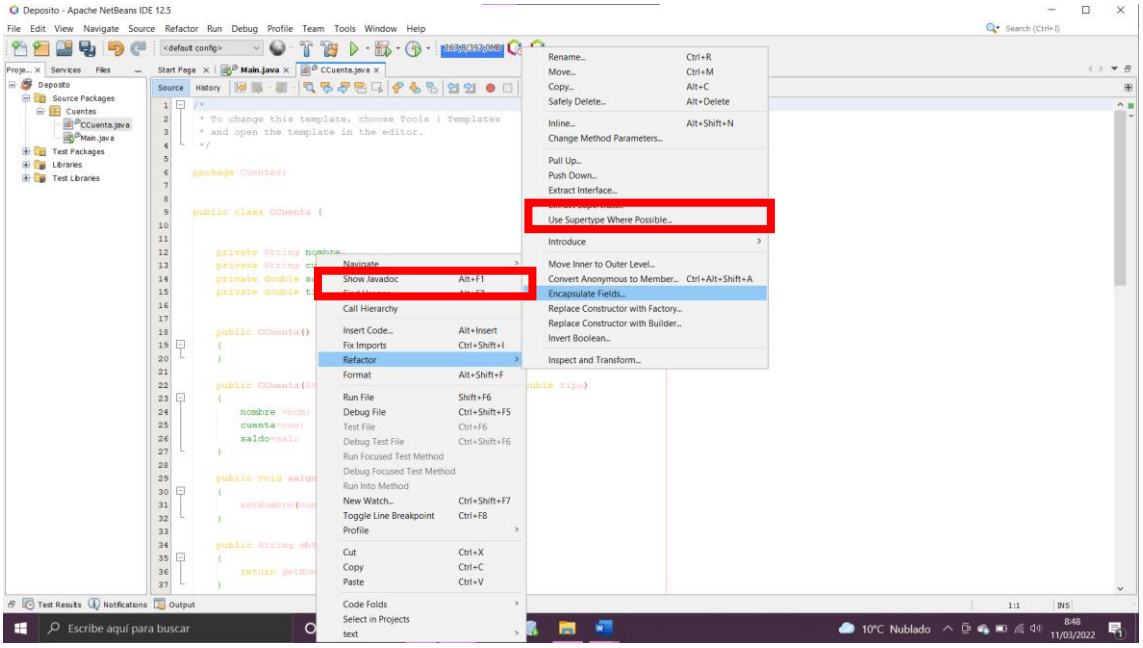
```
3
4       public static void operativa_cuenta(CCuenta Cuenta1) {
5           try {
6               Cuenta1.retirar(2300);
7           } catch (Exception e) {
8               System.out.print("Fallo al retirar");
9           }
0           try {
1               System.out.println("Ingreso en cuenta");
2               Cuenta1.ingresar(695);
3           } catch (Exception e) {
4               System.out.print("Fallo al ingresar");
5           }
6       }
```

4. Encapsular los atributos de la clase CCuenta.

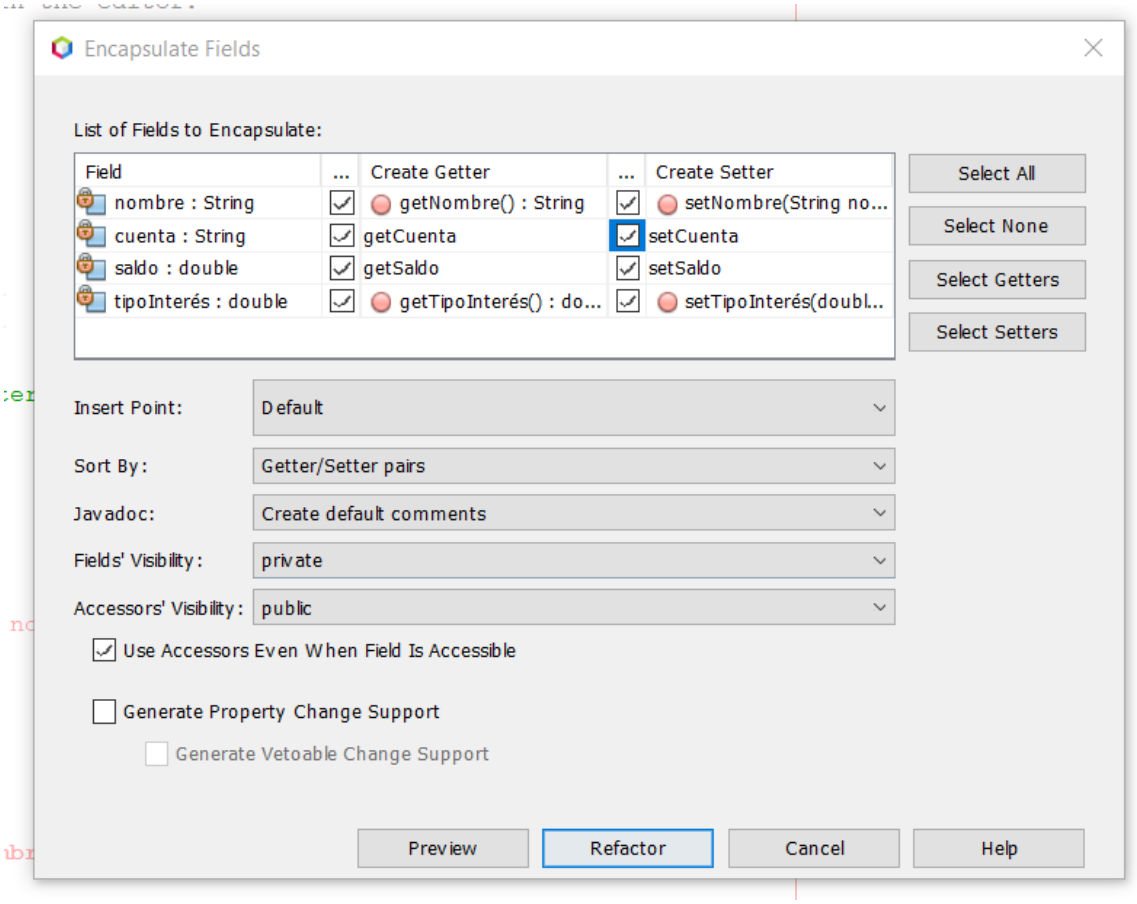
Para encapsular los atributos de la clase CCuenta, pulsamos el botón derechopy en el menú acudimos a Refactor:



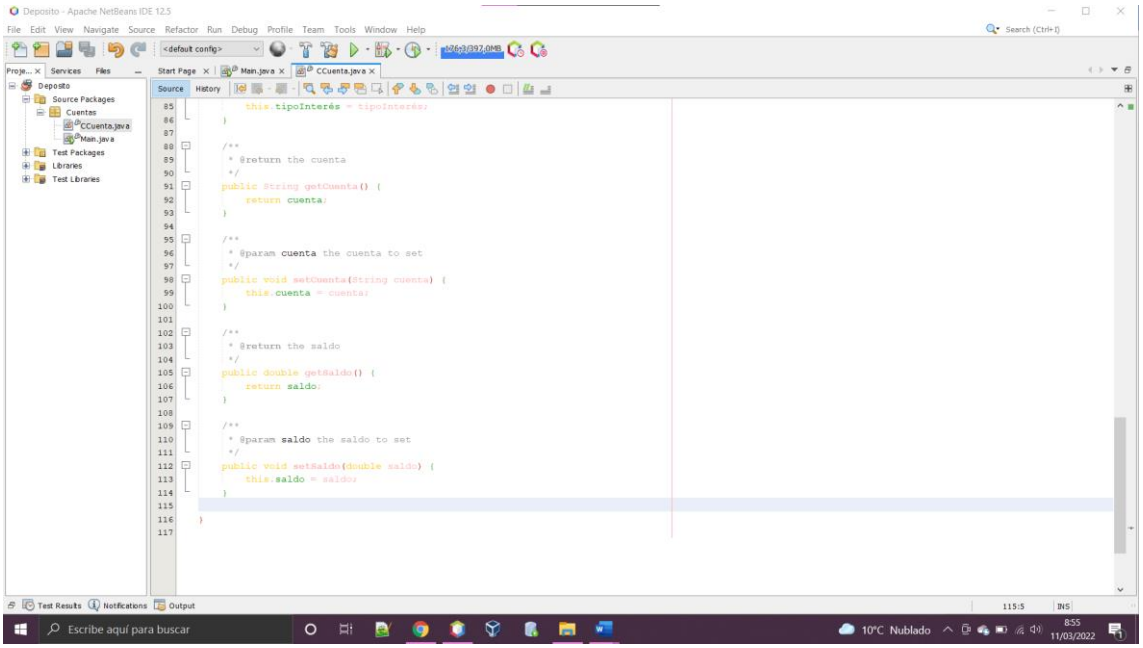
En el menú de Refactor, pulsamos en Encapsulate fields:



Seleccionamos todos los atributos y todos los métodos getter/setter y les establecemos en acceso privado:

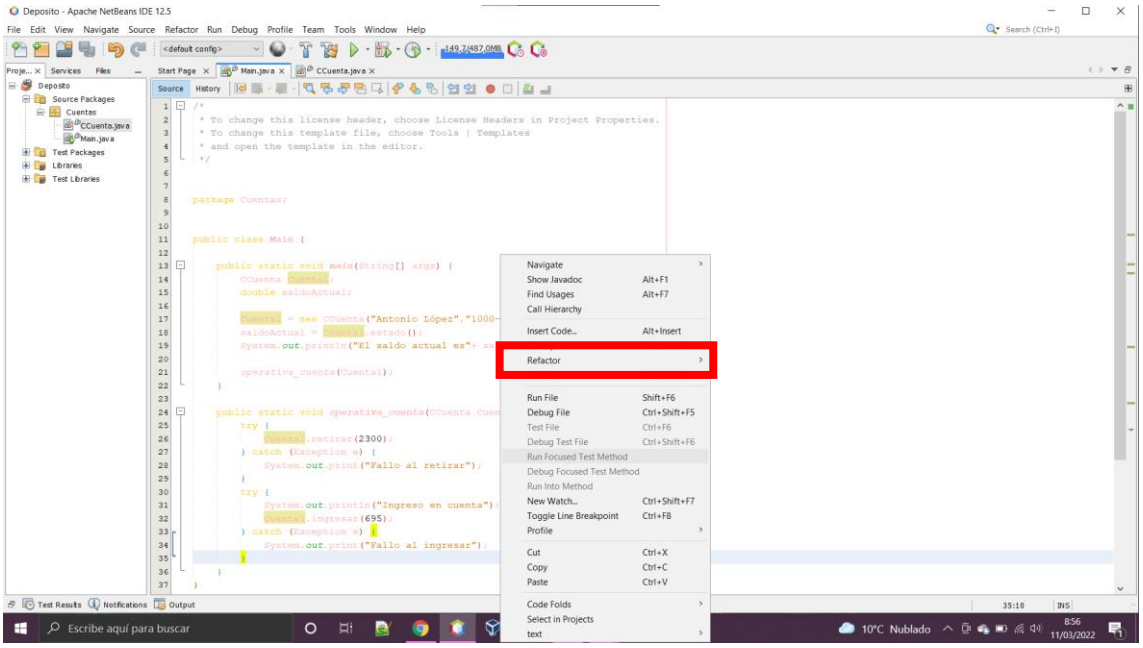


Pulsamos Refactor para aplicar los cambios:

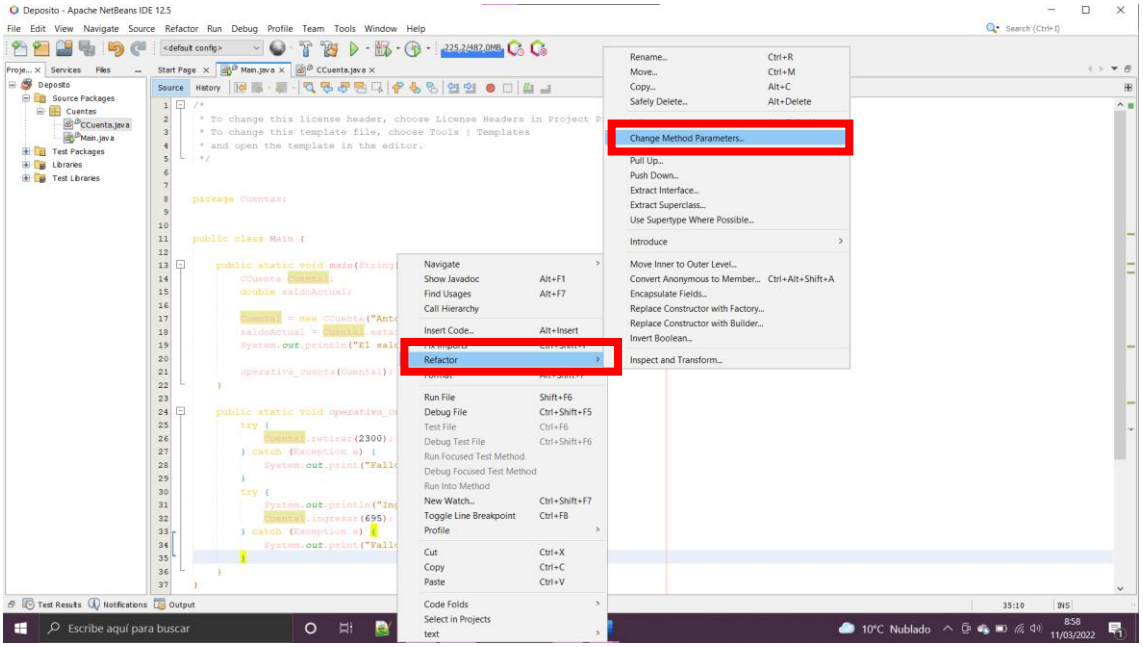


5. Añadir un nuevo parámetro al método operativa_cuenta, de nombre cantidad y de tipo float.

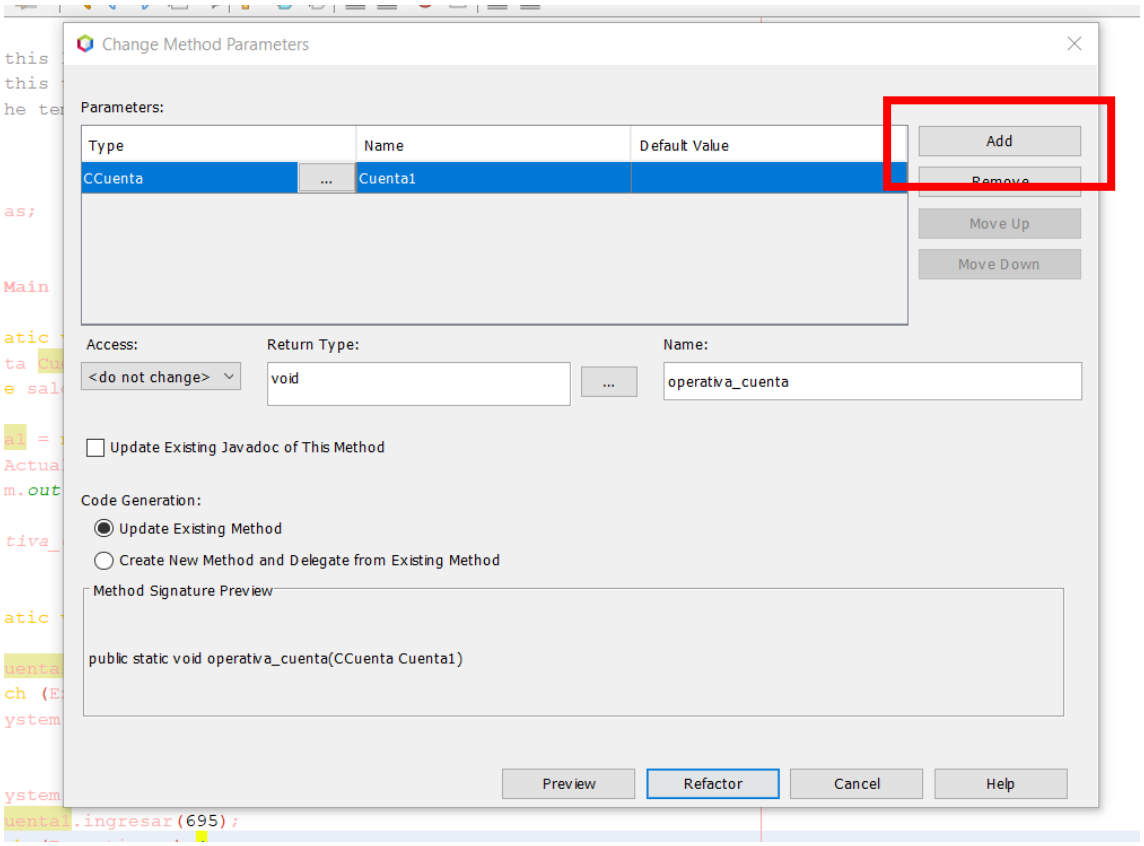
Para añadir el nuevo parámetro, volvemos al menú Refactor:



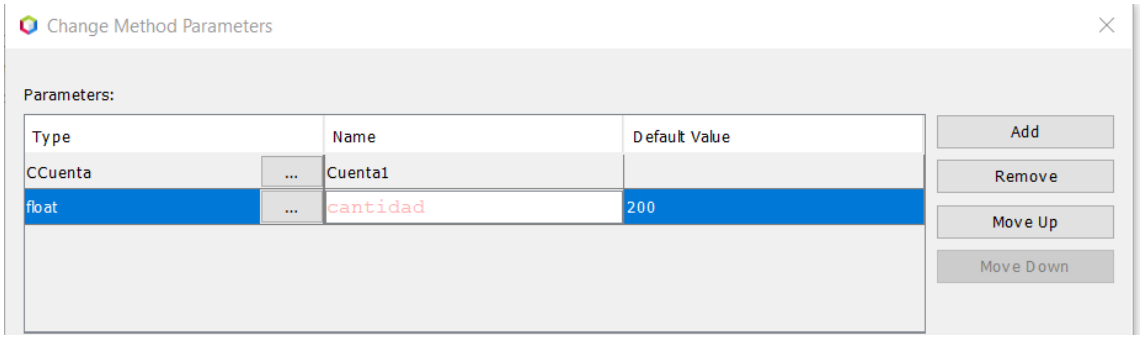
En el menú Refactor, pulsamos sobre Change Method Parameters:



En el menú que nos aparece pulsamos sobre Add:



Escribimos el tipo de variable, su nombre y su valor por defecto:



Pulsamos Refactor para aplicar los cambios porque no queremos realizar ningún cambio más:

```
public static void operativa_cuenta(CCuenta Cuenta1, float cantidad) {
    try {
        Cuenta1.retirar(2300);
    } catch (Exception e) {
        System.out.print("Fallo al retirar");
    }
    try {
        System.out.println("Ingreso en cuenta");
        Cuenta1.ingresar(695);
    } catch (Exception e) {
        System.out.print("Fallo al ingresar");
    }
}
```