ENTORNOS DE DESARROLLO

Tarea para ED03. CURSO 2022/2023.

CRISTINA CORINO LOPEZ

**-INDICE –**

[Deberás realizar un documento donde dar respuesta a los siguientes apartados 2](#_Toc123499886)

[1. Realiza un análisis de caja blanca completo del método ingresar. 2](#_Toc123499887)

[2. Realiza un análisis de caja negra, incluyendo valores límite y conjetura de errores del método retirar 4](#_Toc123499888)

[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. 5](#_Toc123499889)

[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 10](#_Toc123499890)

[5. Genera los siguientes puntos de ruptura para validar el comportamiento del método ingresar en modo depuración. 14](#_Toc123499891)

[En eclipse, pulsando el botón derecho sobre la ventana de puntos de ruptura selecciona la opción "Exportar puntos de ruptura". 15](#_Toc123499892)

[En Netbeans se realizará la muestra de puntos de ruptura mediante la captura de pantalla correspondiente. 17](#_Toc123499893)

# **- Listado de Ilustraciones -**

[Ilustración 1. Documento CCuenta - Método ingresar 2](#_Toc124356711)

[Ilustración 2. Grafo 3](#_Toc124356712)

[Ilustración 3. Documento CCuenta - Método retirar 4](#_Toc124356713)

[Ilustración 4. Instalación de Eclipse en el SO 5](#_Toc124356714)

[Ilustración 5. Clase CCuenta.java en Eclipse 6](#_Toc124356715)

[Ilustración 6. Creando JUnit Test Case 6](#_Toc124356716)

[Ilustración 7. Creando JUnit Test en JUnit 5 7](#_Toc124356717)

[Ilustración 8. Ejecución del código proporcionado JUnit 5 7](#_Toc124356718)

[Ilustración 9. Modificación del test respecto al ejercicio 1 8](#_Toc124356719)

[Ilustración 10. CCuentaTest en JUnit5 correcto 9](#_Toc124356720)

[Ilustración 11. CCuenta desde NetBeans 10](#_Toc124356721)

[Ilustración 12. CCuentaTest en NetBeans 11](#_Toc124356722)

[Ilustración 13. Añadimos las librerías necesarias 11](#_Toc124356723)

[Ilustración 14. CCuenta en NetBeans y su Test sin modificar 12](#_Toc124356724)

[Ilustración 15. Ejecución del test con los valores cambiados 13](#_Toc124356725)

[Ilustración 16. Test ejecutado correctamente en NetBeans 13](#_Toc124356726)

[Ilustración 17. Punto de parada al crear el objeto miCuenta 14](#_Toc124356727)

[Ilustración 18. Punto de parada en cada instrucción del método ingresar 14](#_Toc124356728)

[Ilustración 19. Exportación de breakpoints a un fichero 15](#_Toc124356729)

[Ilustración 20. NetBeans Breackpoints (Parte I) 17](#_Toc124356730)

[Ilustración 21. NetBeans Breakpoints (Parte II) 17](#_Toc124356731)

# **REFACTORIZACIÓN**

**Las clases deberán formar parte del paquete cuentas.**

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

**Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.**

**Encapsular los atributos de la clase CCuenta.**

**Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.**

# **GIT**

**Configurar GIT para el proyecto. Crear un repositorio público en GitHub.**

**Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.**

**Mostrar el historial de versiones para el proyecto mediante un comando desde consola.**

# **JAVADOC**

**Insertar comentarios JavaDoc en la clase CCuenta.**

**Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.**

***Criterios de puntuación. Total 10 puntos.***

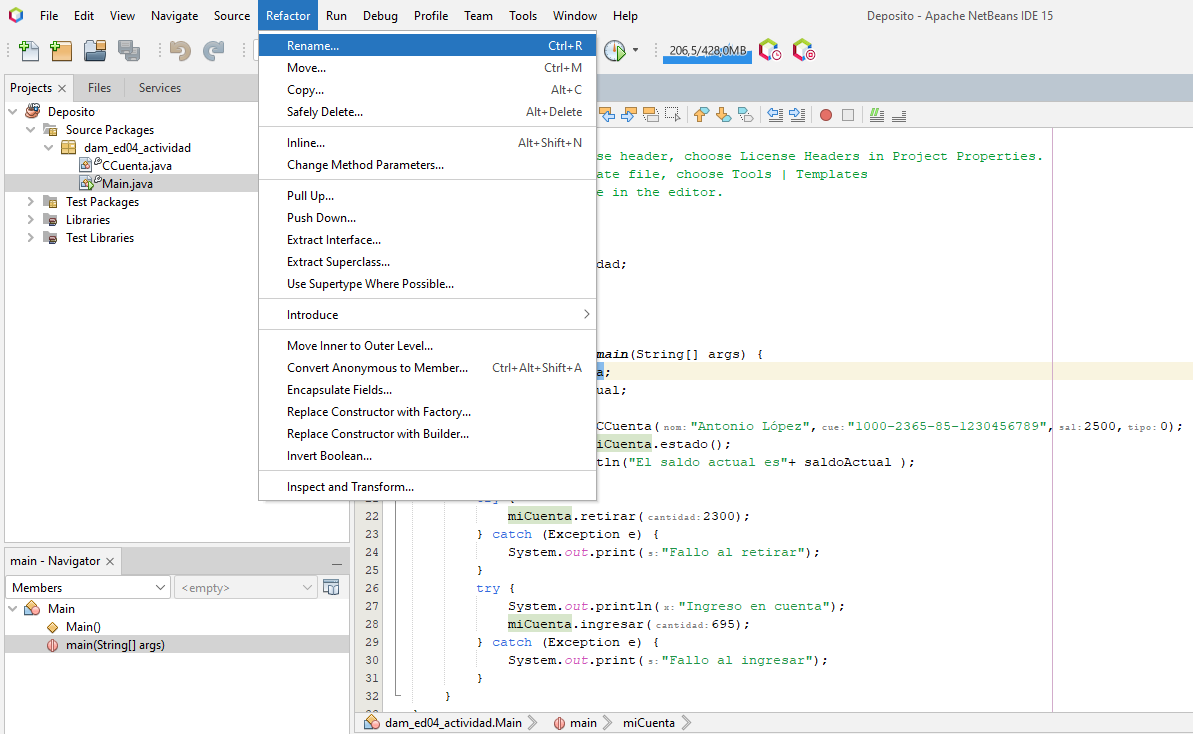
1. **Cambia el nombre de la variable "miCuenta" por "cuenta1". = 1 punto.**
2. **Introduce el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1. = 1 punto.**
3. **Encapsula los cuatro atributos de la clase CCuenta. = 1 punto.**
4. **Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float. = 1 punto.**
5. **Configurar GIT para el proyecto. Crear un repositorio público en GitHub.= 1 punto.**
6. **Realiza, al menos, una operación commit, comentando el resultado de la ejecución. = 1 punto.**
7. **Mostrar el historial de versiones para el proyecto mediante un comando desde consola.= 1 punto.**
8. **Inserta comentarios Javadoc en la clase Ccuenta. = 1 punto.**
9. **Genera documentación Javadoc para todo el proyecto. = 1 punto.**
10. **Comprueba que la documentación generada por Javadoc, abarca todos los métodos y atributos de la clase Ccuenta. = 1 punto.**

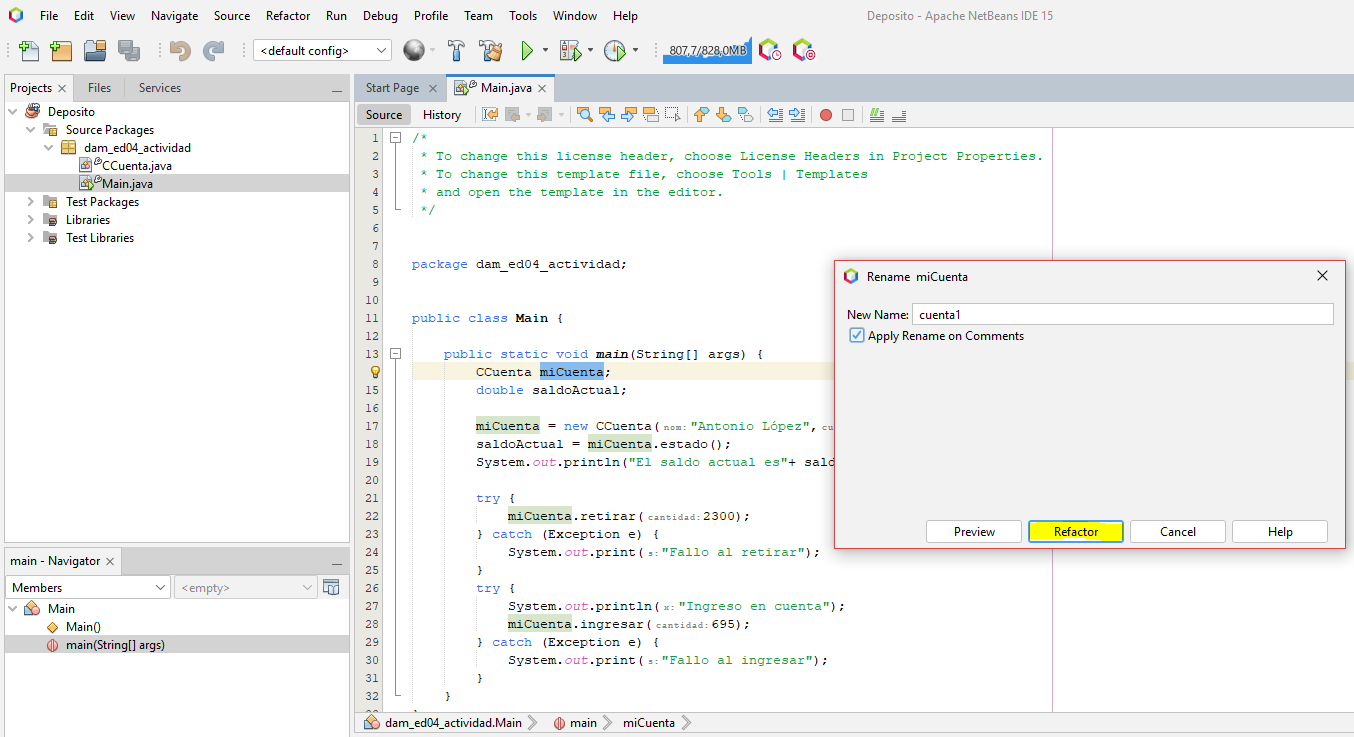
# **REFACTORIZACIÓN**

**Las clases deberán formar parte del paquete cuentas.**

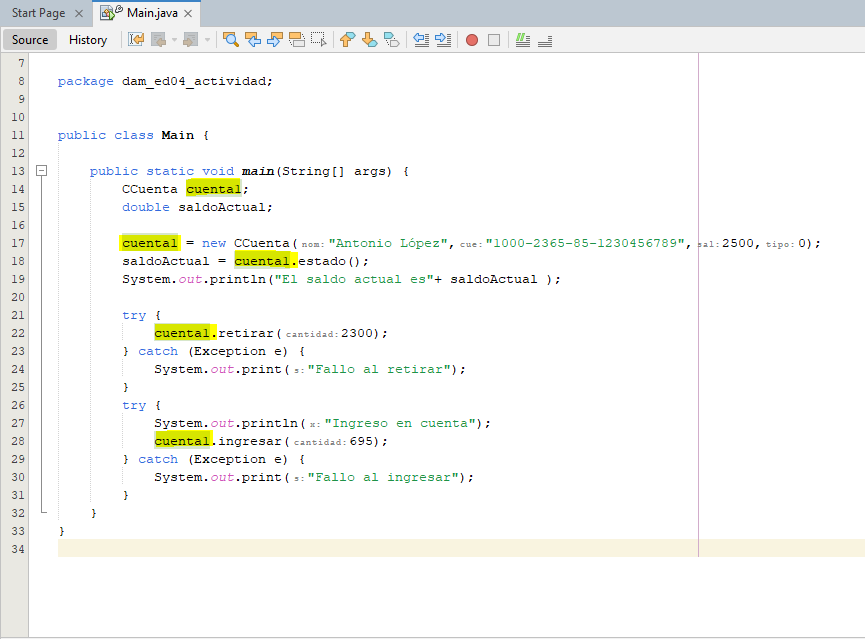
**Cambiar el nombre de la variable "miCuenta" por "cuenta1".**

Seleccionamos la variable miCuenta y desde el menú Refactor, en Rename… modificamos el nombre de la variable y clicamos en Refactor, como vemos a continuación:



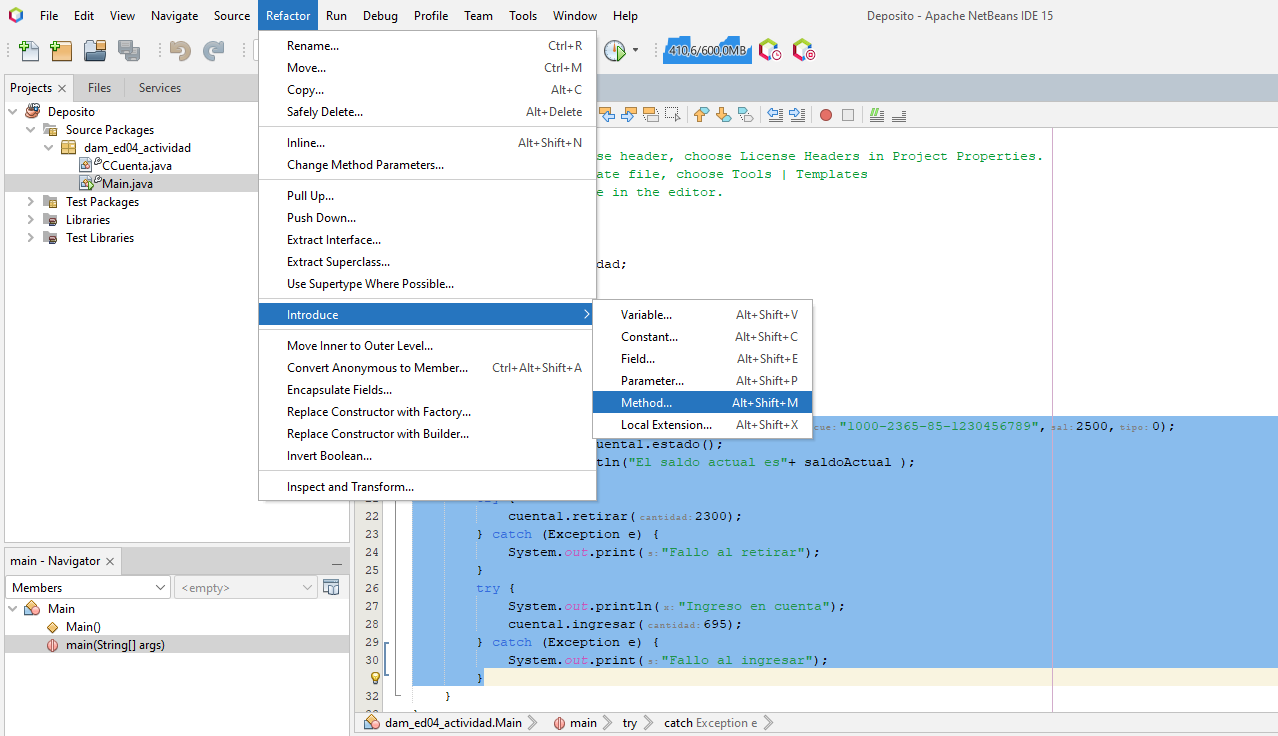


Con esto podemos ver que nos modifica lo siguiente:

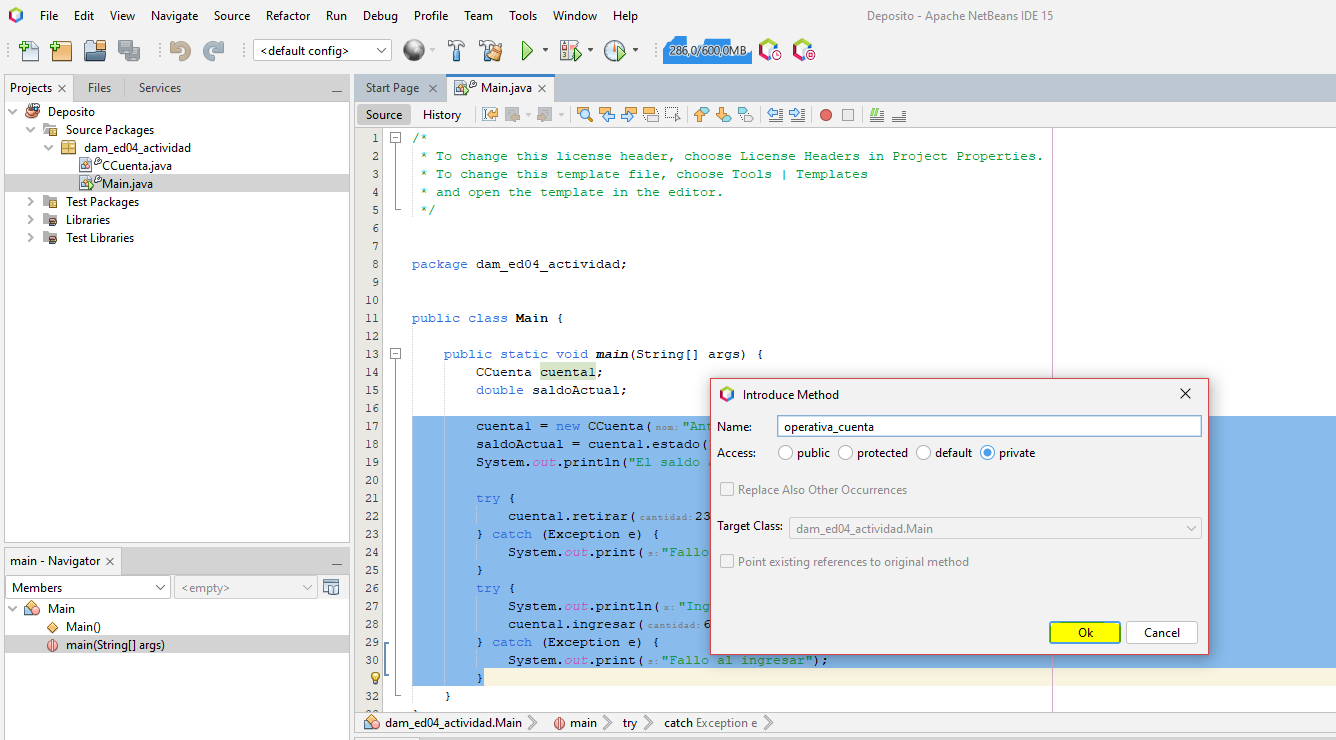


**Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.**

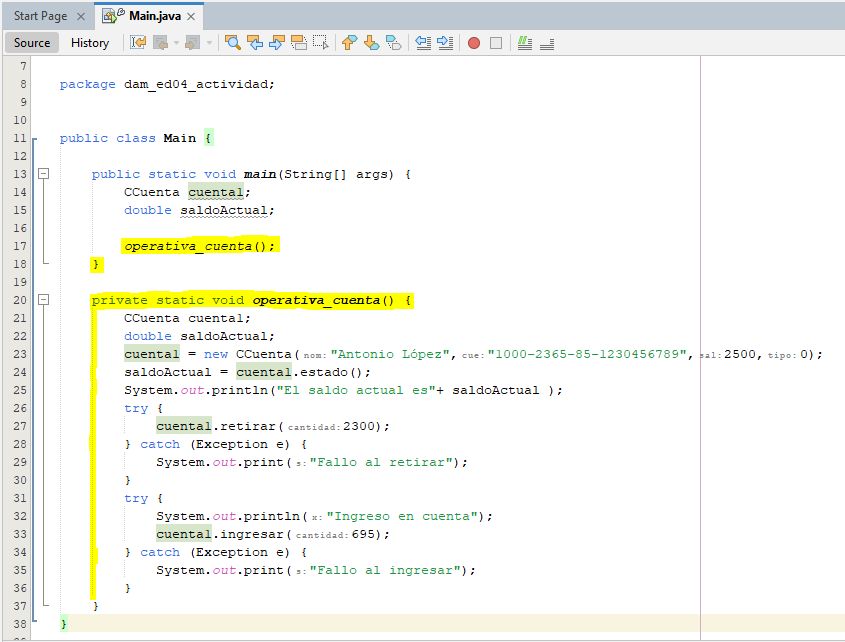
A continuación, seleccionamos las sentencias mencionadas y desde el menú Refactor, en Introduce, vamos a Method…



Como nombre indicaremos operativa\_cuenta, y tendrá acceso privado, con ello de damos a Ok:

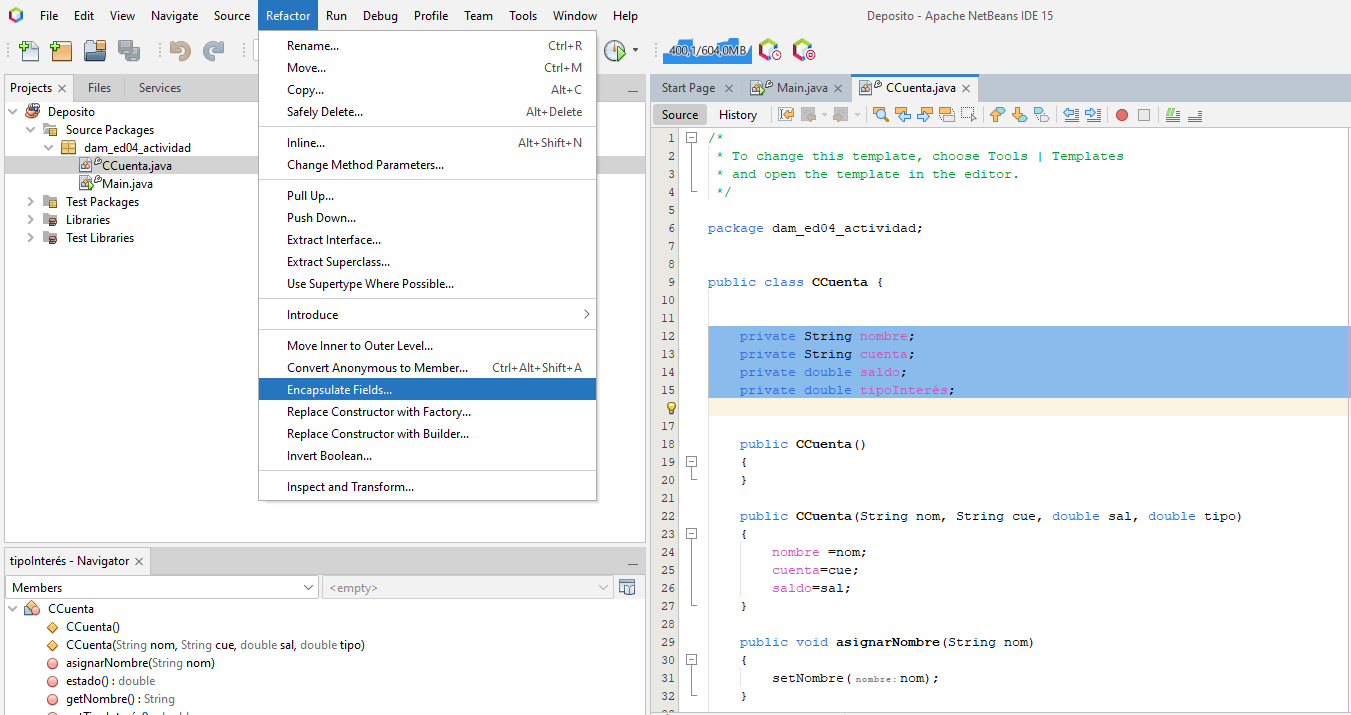


Con esto podemos ver que nos incluye lo siguiente:

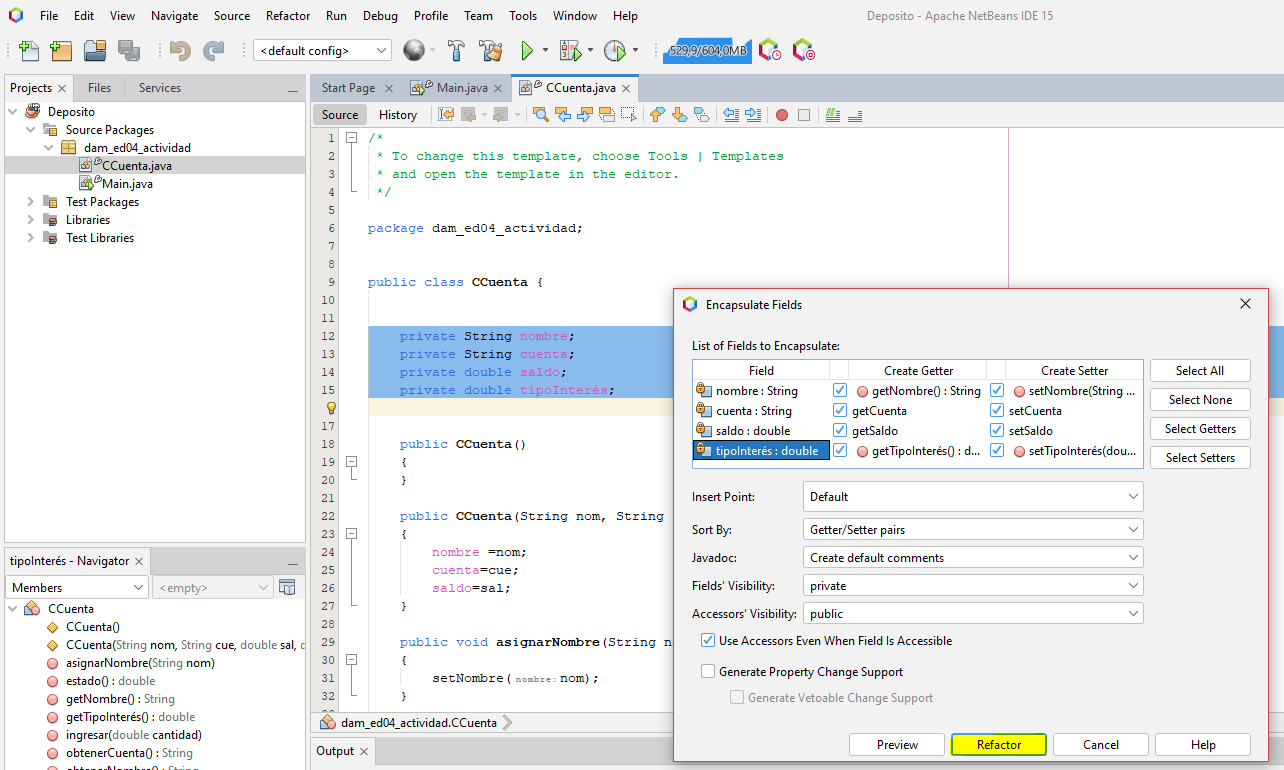


**Encapsular los atributos de la clase CCuenta.**

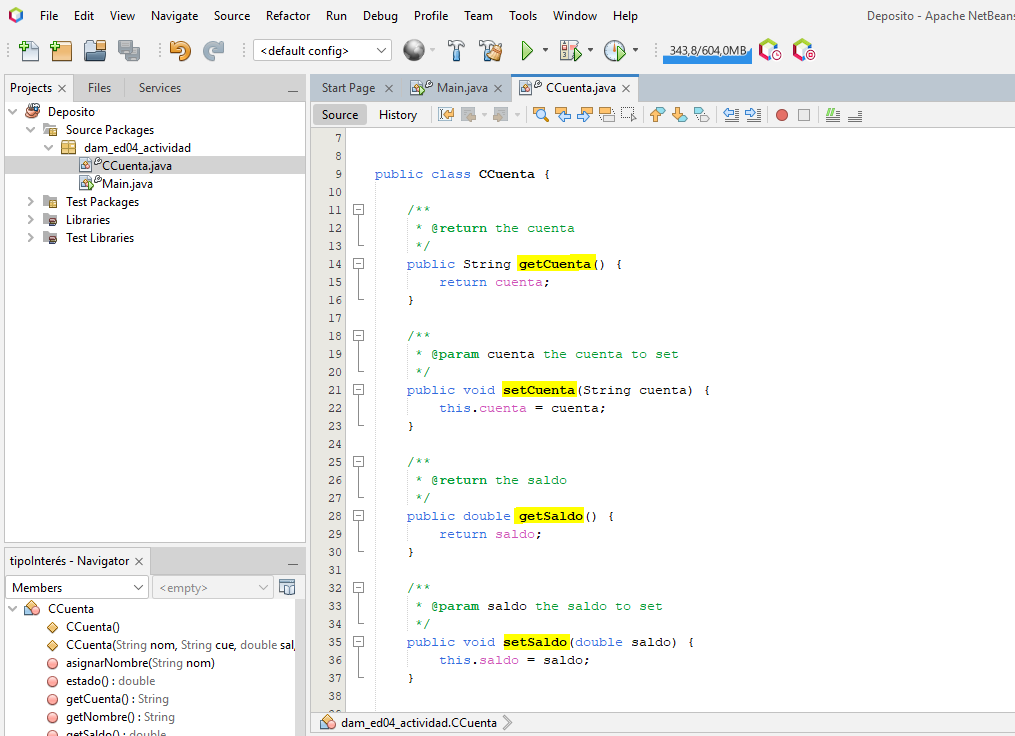
Para encapsular los atributos iremos a CCuenta.java ya que es ahí donde les podemos encontrar. Les seleccionamos y desde el menú Refactor hacemos clic en Encapsulate Fields..



Nos aparecerá la siguiente pantalla donde únicamente clicaremos sobre Refactor.

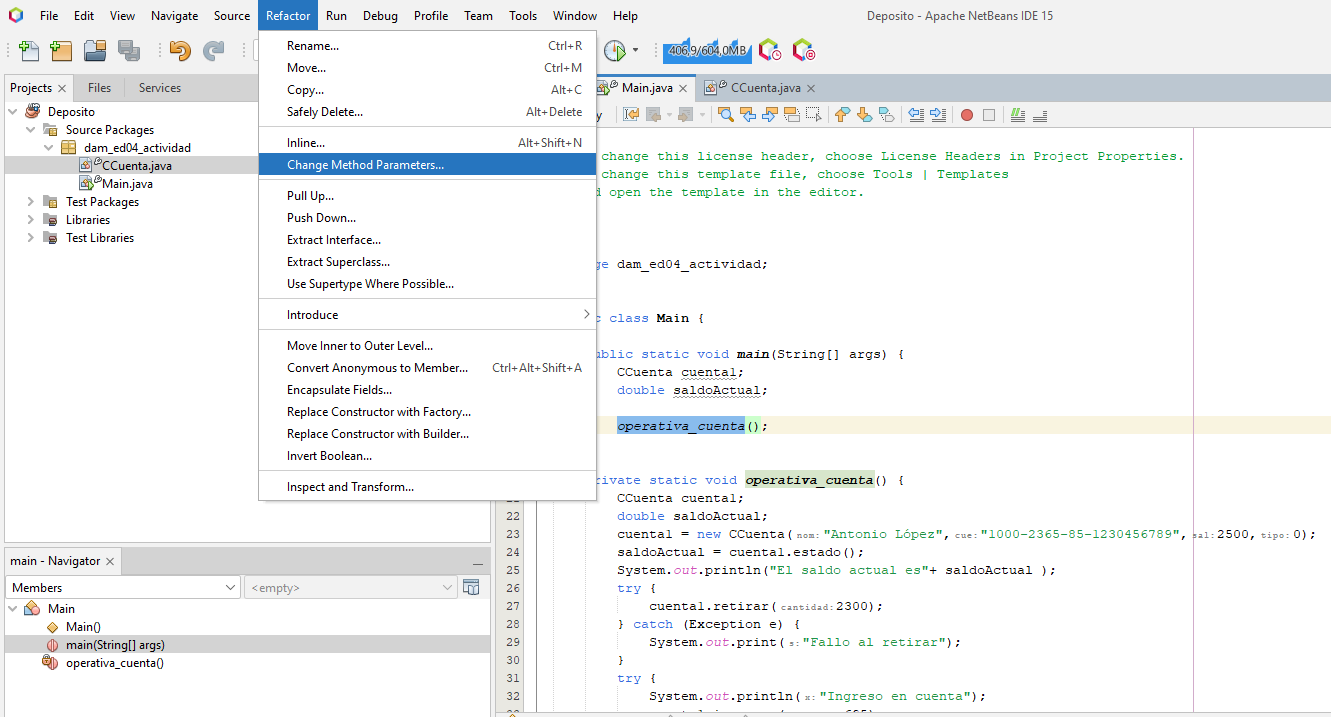


Con esto podemos ver el siguiente resultado:

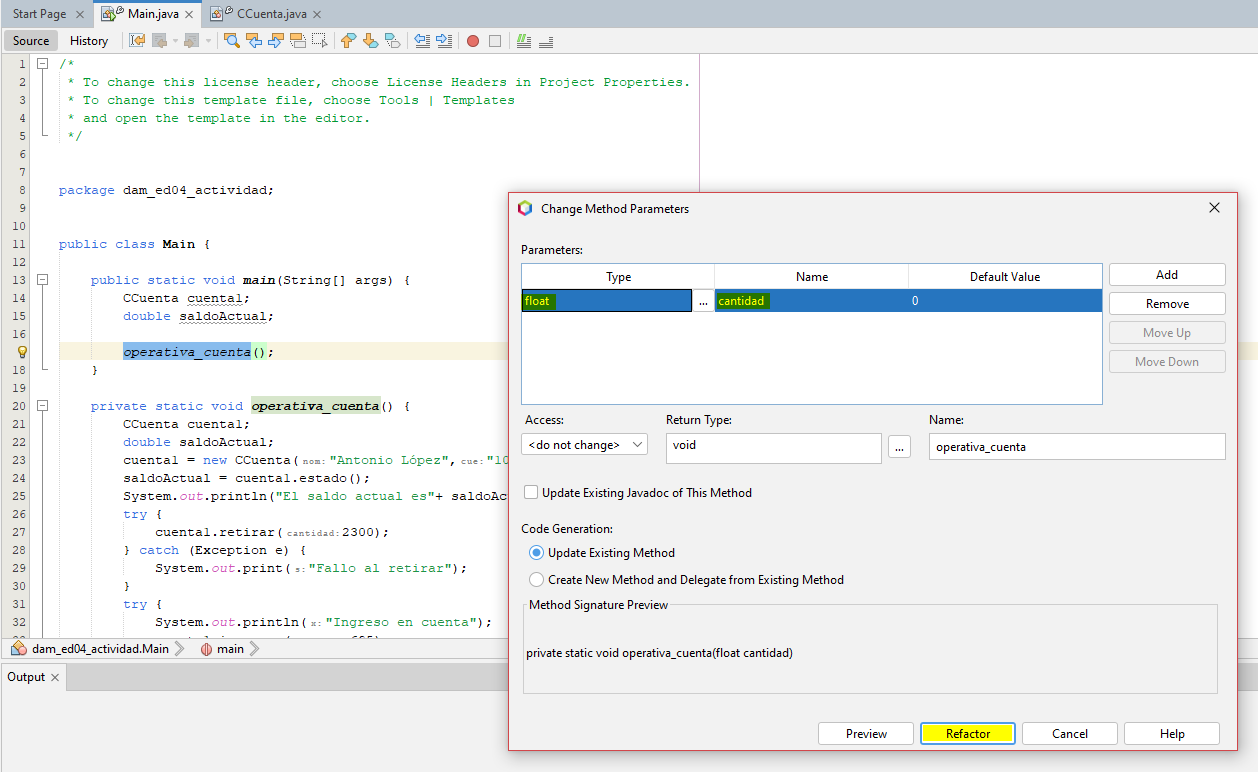


**Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.**

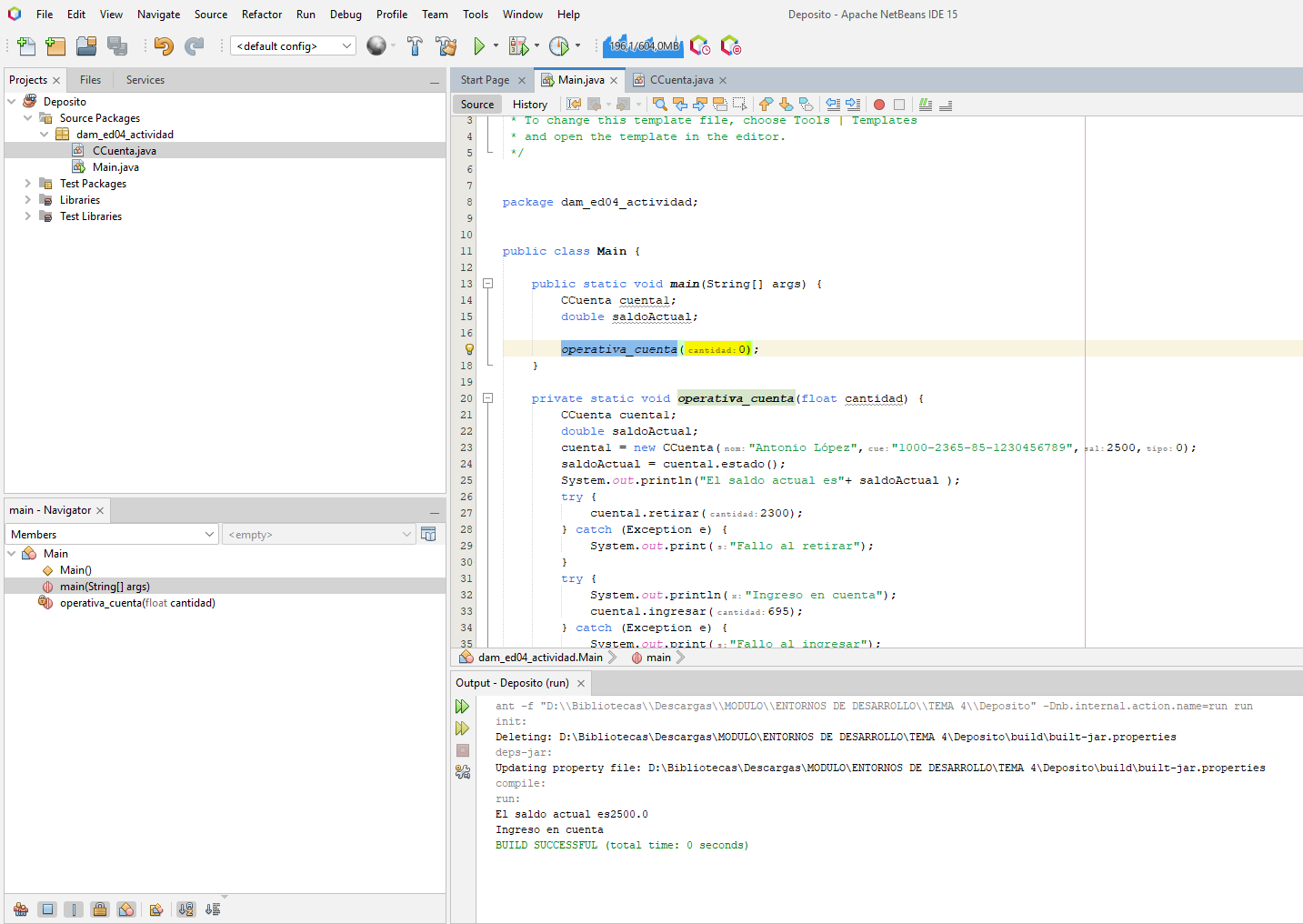
Para añadir un nuevo parámetro, seleccionamos el método y en el menú Refactor, clicamos sobre Change Method Parameters…



A continuación, creamos un parámetro a traves de add con name “cantidad” y que su type sea “float”.



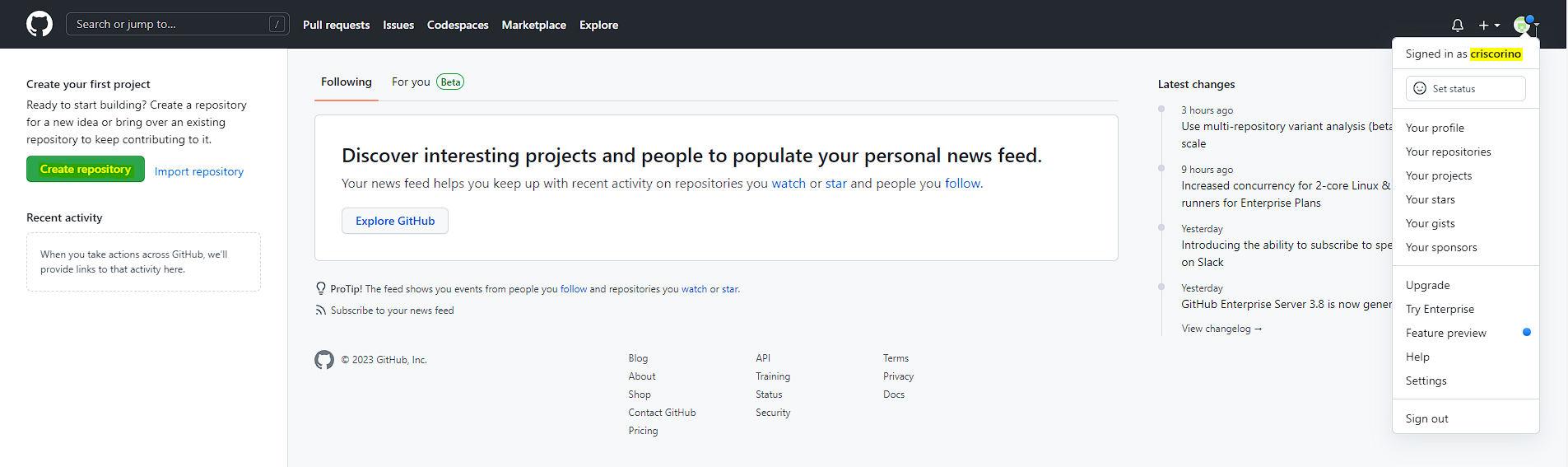
Con esto podemos ver el siguiente resultado:



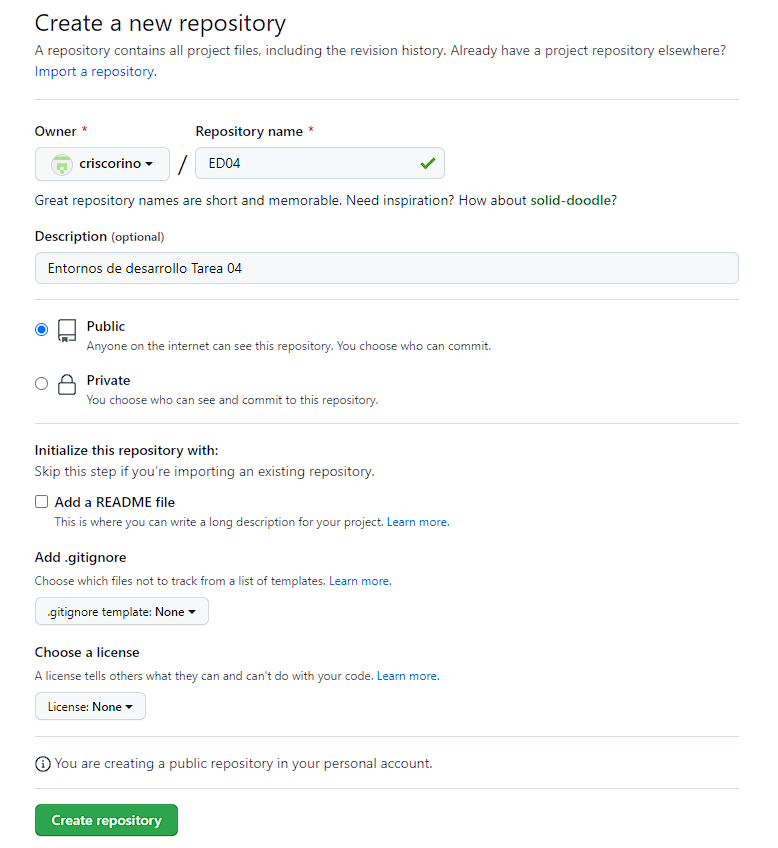
# **GIT**

**Configurar GIT para el proyecto. Crear un repositorio público en GitHub.**

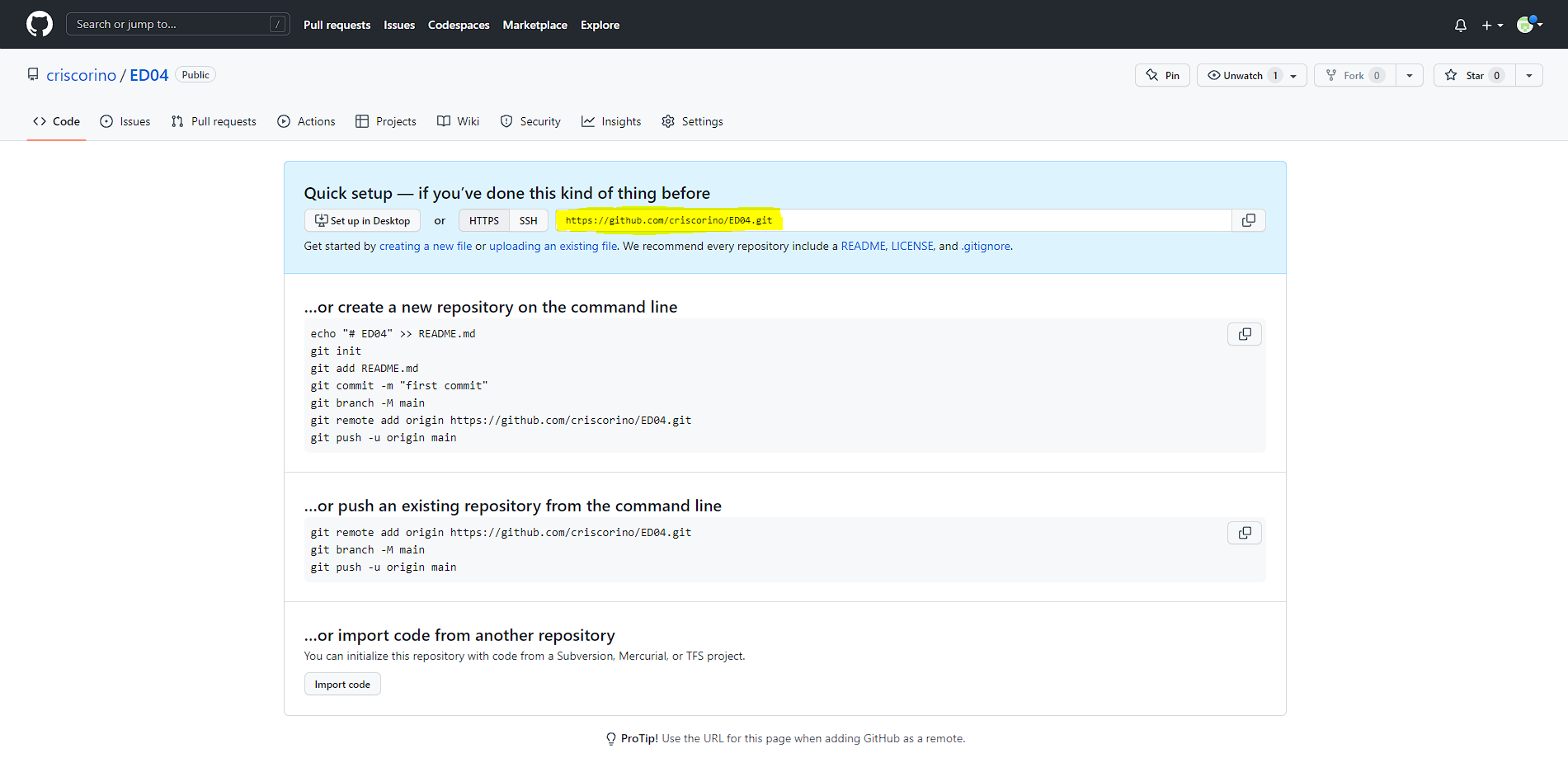
Accedemos a la página web <https://github.com/> y nos registramos. Una vez creada la cuenta podemos ver nuestro espacio donde ya estamos logeados con criscorino y a continuación hacemos clic en Create Repository.

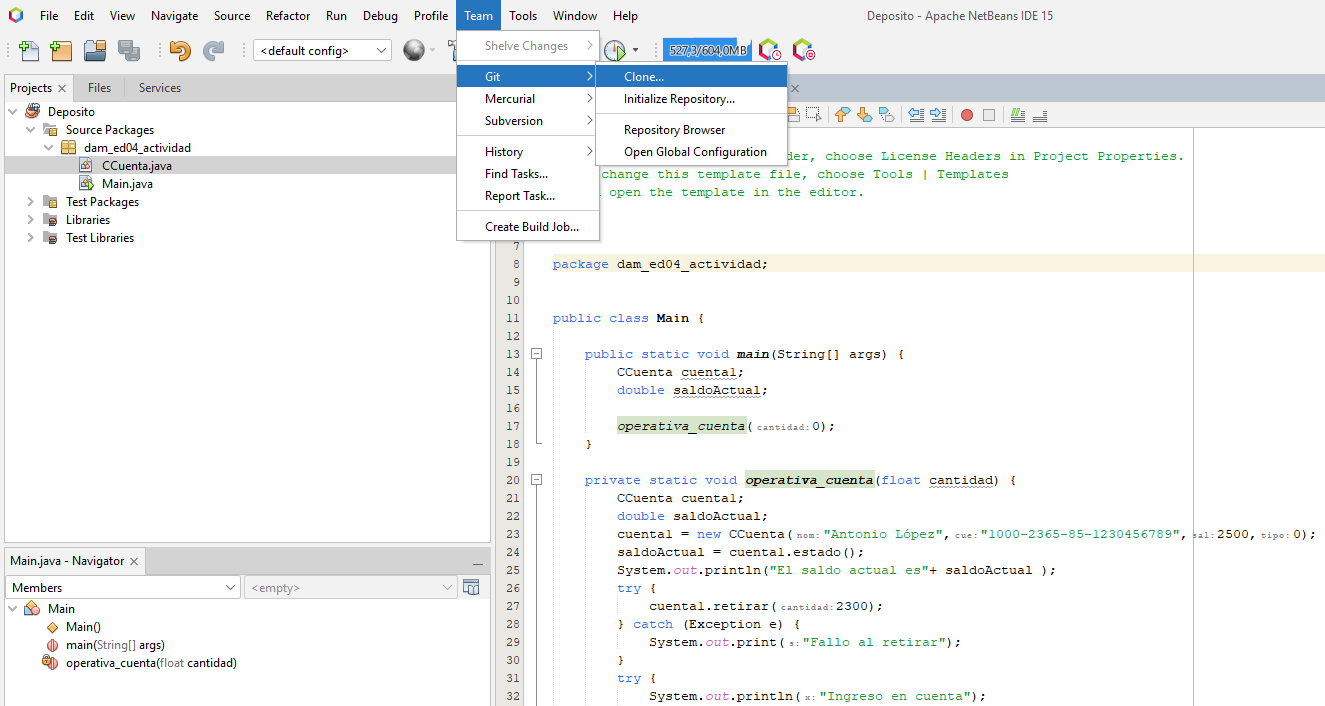


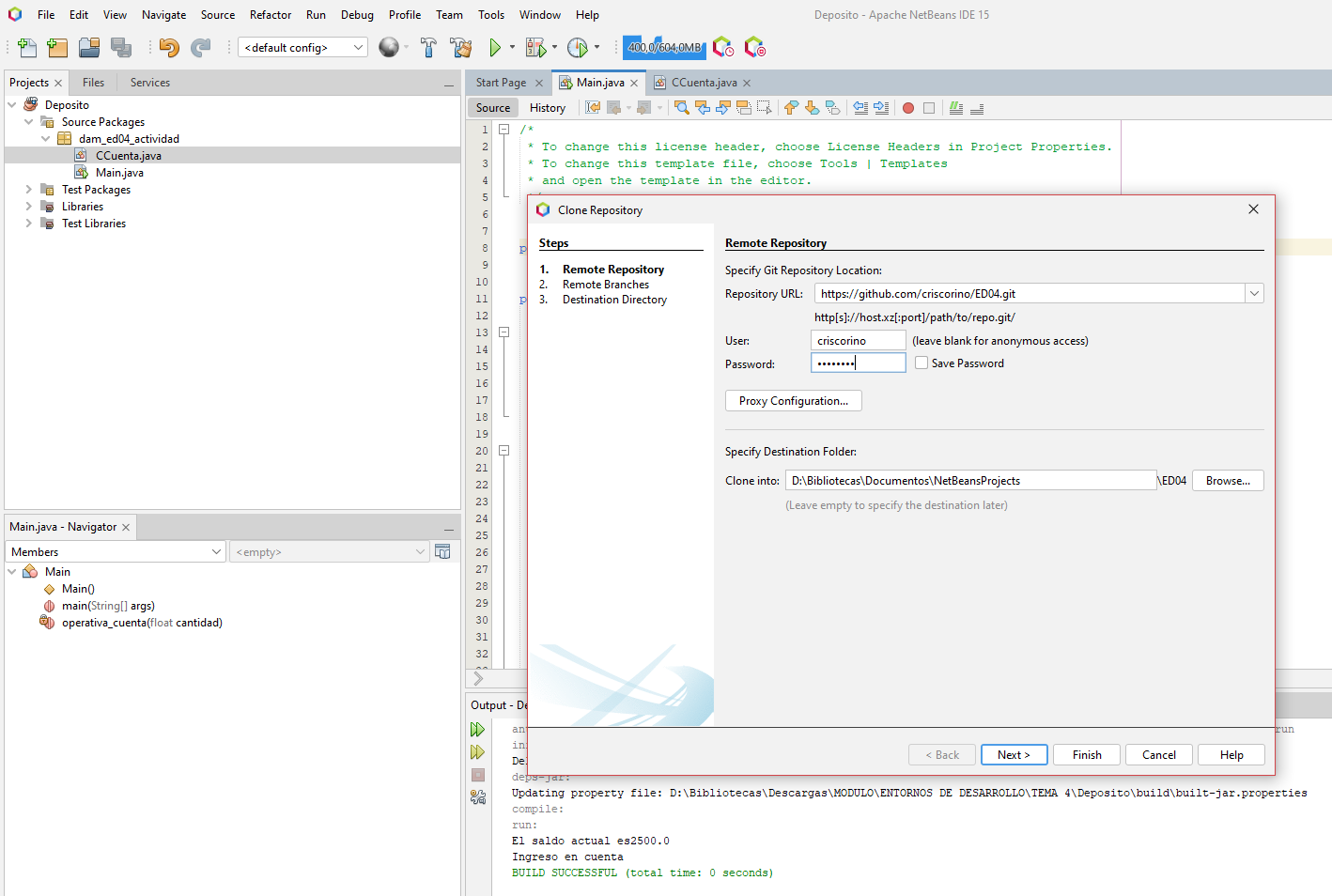
Creamos un nuevo repositorio con el nombre que queramos, en este caso ED04 haciendo referencia a la actividad y la dejamos pública.

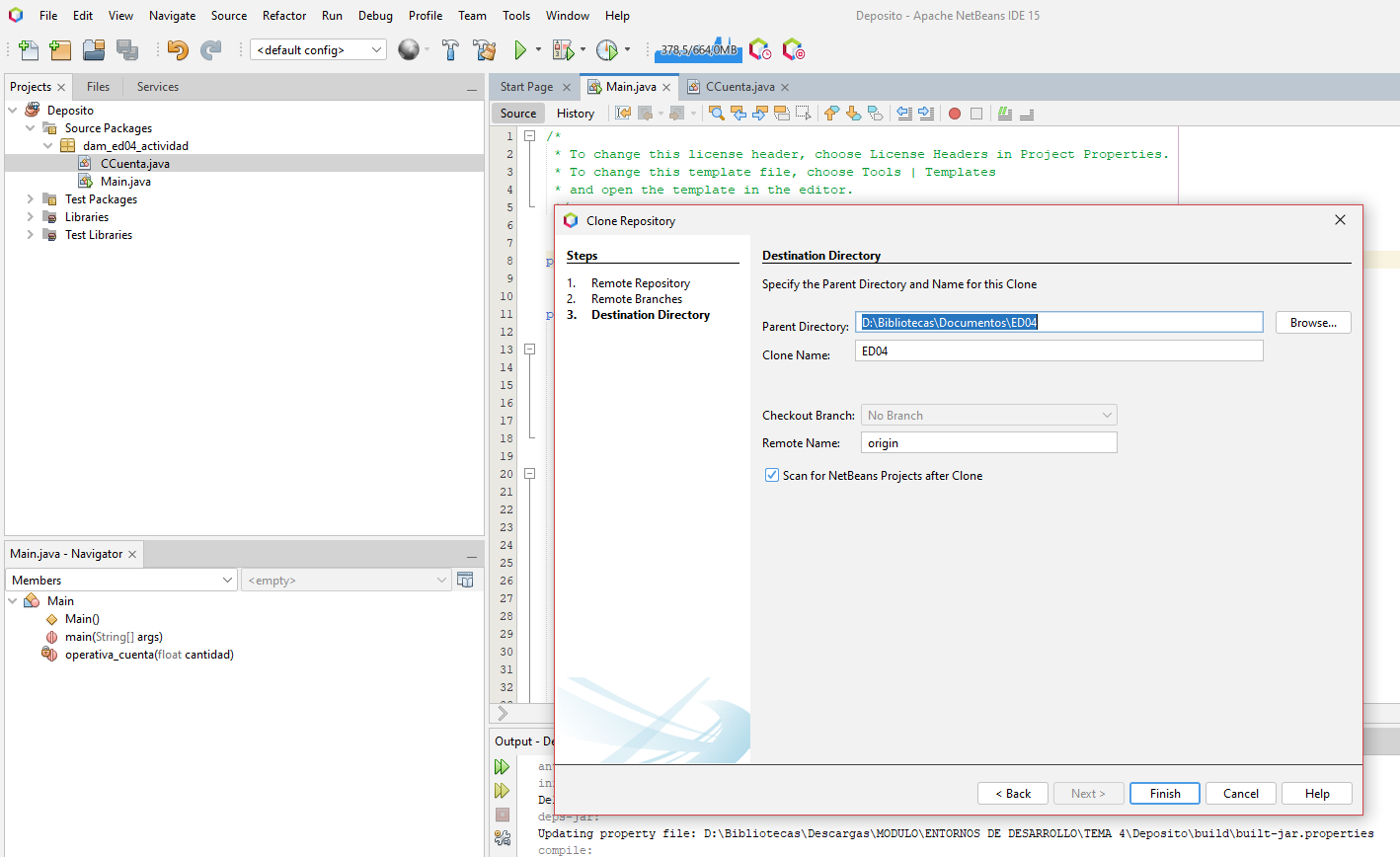


Una vez creado el repositorio se genera el siguiente código que vamos a copiar en portapapeles para su posterior utilización en netbeans:









**Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.**

**Mostrar el historial de versiones para el proyecto mediante un comando desde consola.**

# **JAVADOC**

**Insertar comentarios JavaDoc en la clase CCuenta.**

**Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.**