

## Enunciado.

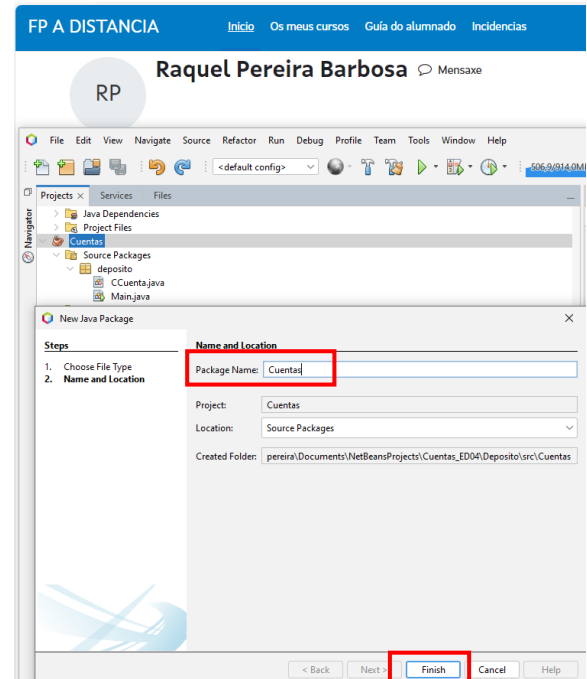
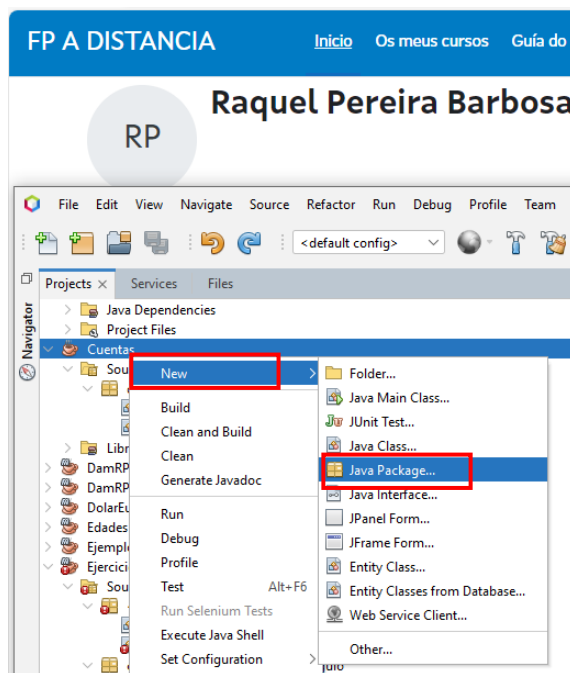
En el proyecto Java "Deposito", hay definida una Clase llamada CCuenta, que tiene una serie de atributos y métodos. El proyecto cuenta asimismo con una Clase Main, donde se hace uso de la clase descrita.

Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

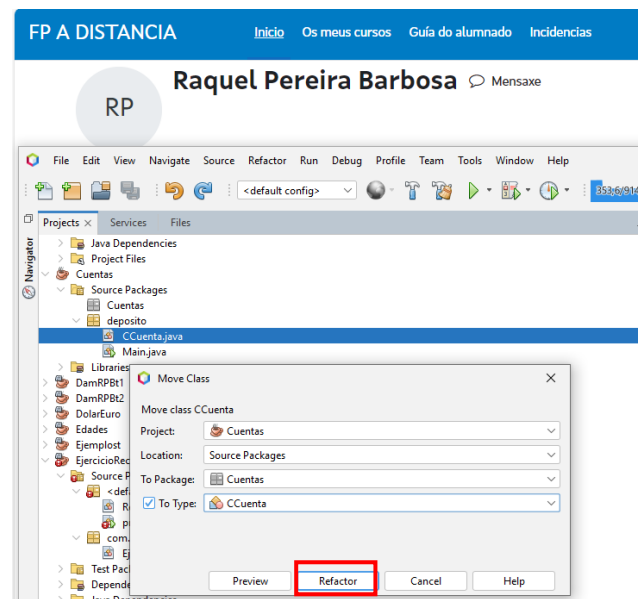
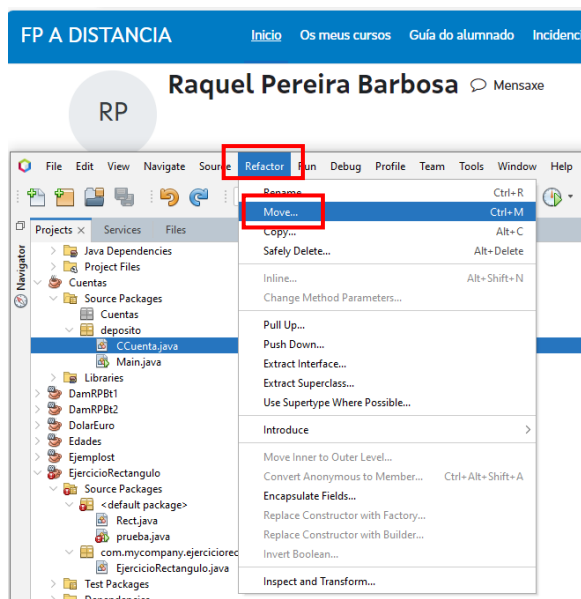
## REFACTORIZACIÓN

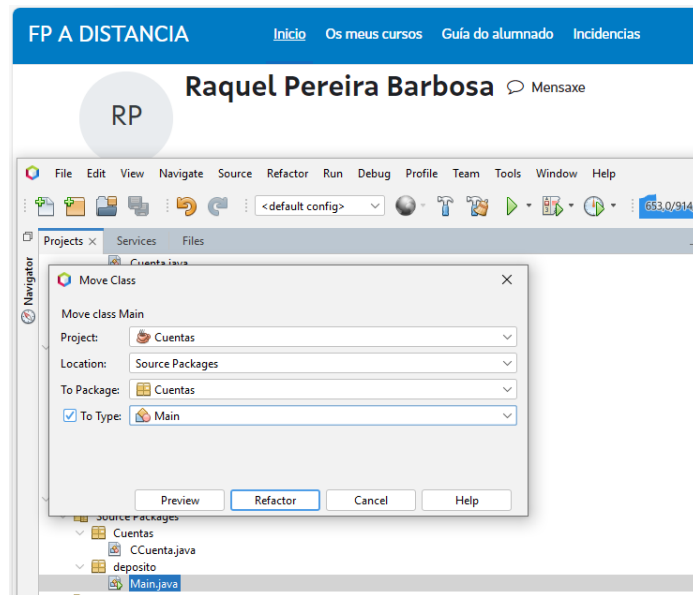
1. Las clases deberán formar parte del paquete cuentas.

Crearemos un nuevo paquete llamado cuentas pulsando sobre el proyecto botón derecho-Nuevo Paquete y lo llamaremos Cuentas

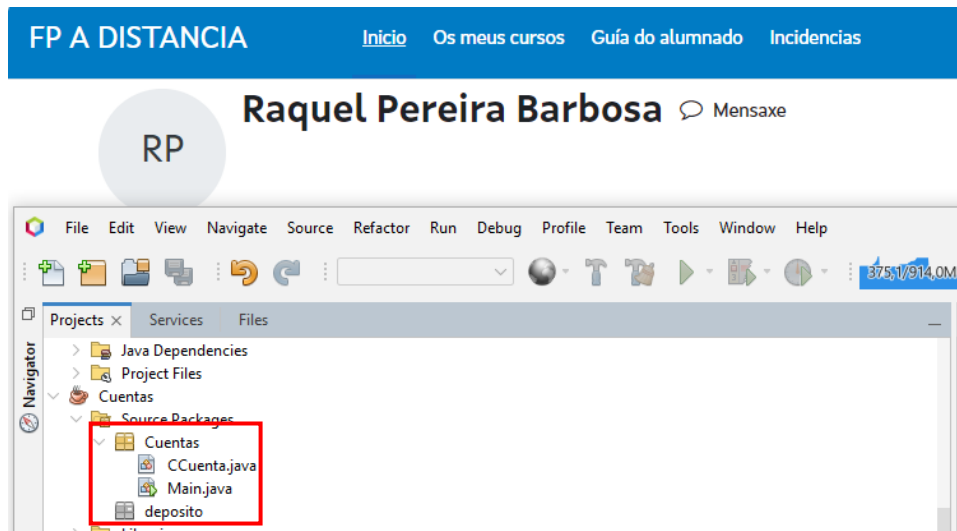


Con refactorización movemos las clases existentes al paquete Cuentas y lo repetimos para ambas clases

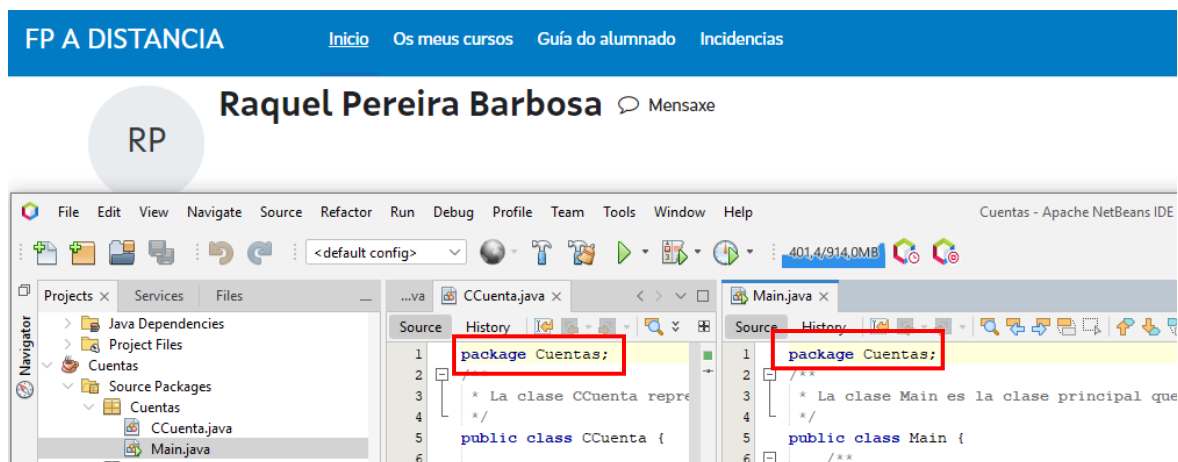




Vemos que se han movido al paquete cuentas y que depósito está vacío

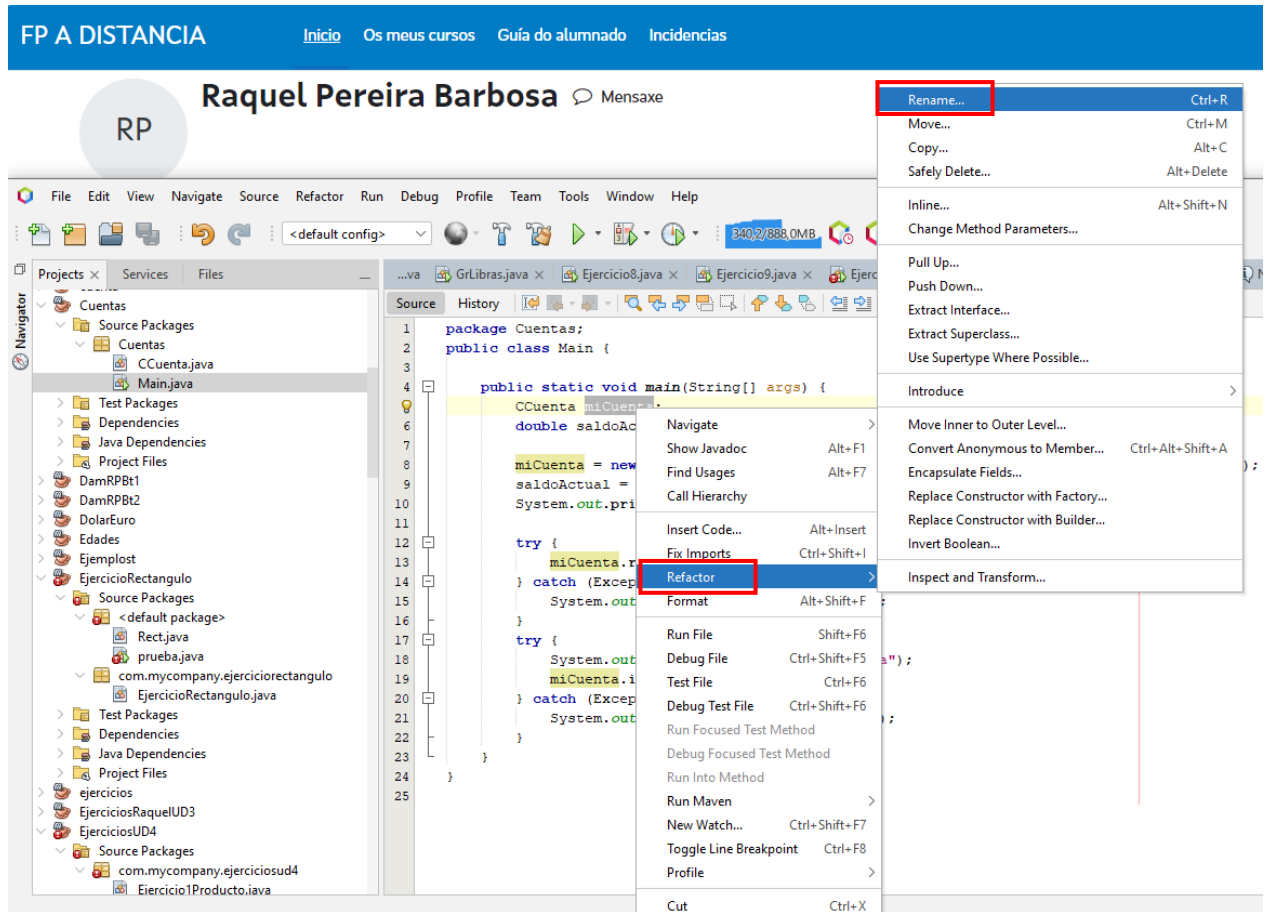


También podemos observar que al refactorizar el paquete ya se cambia automático en el código:

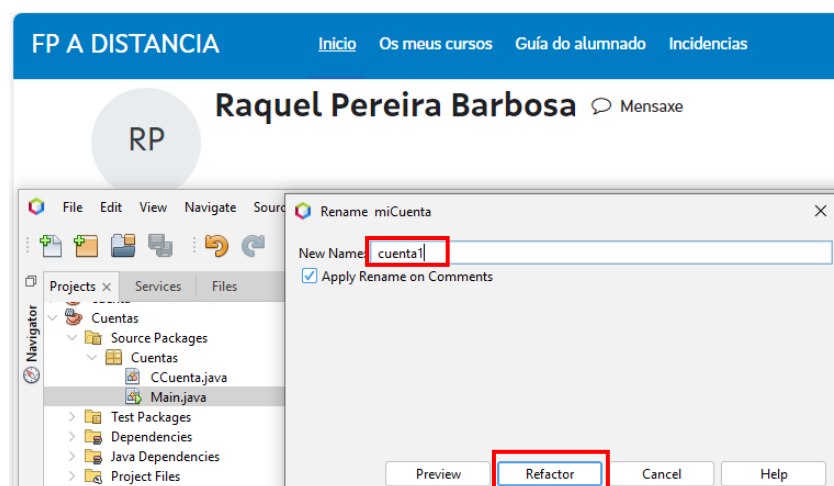


2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".

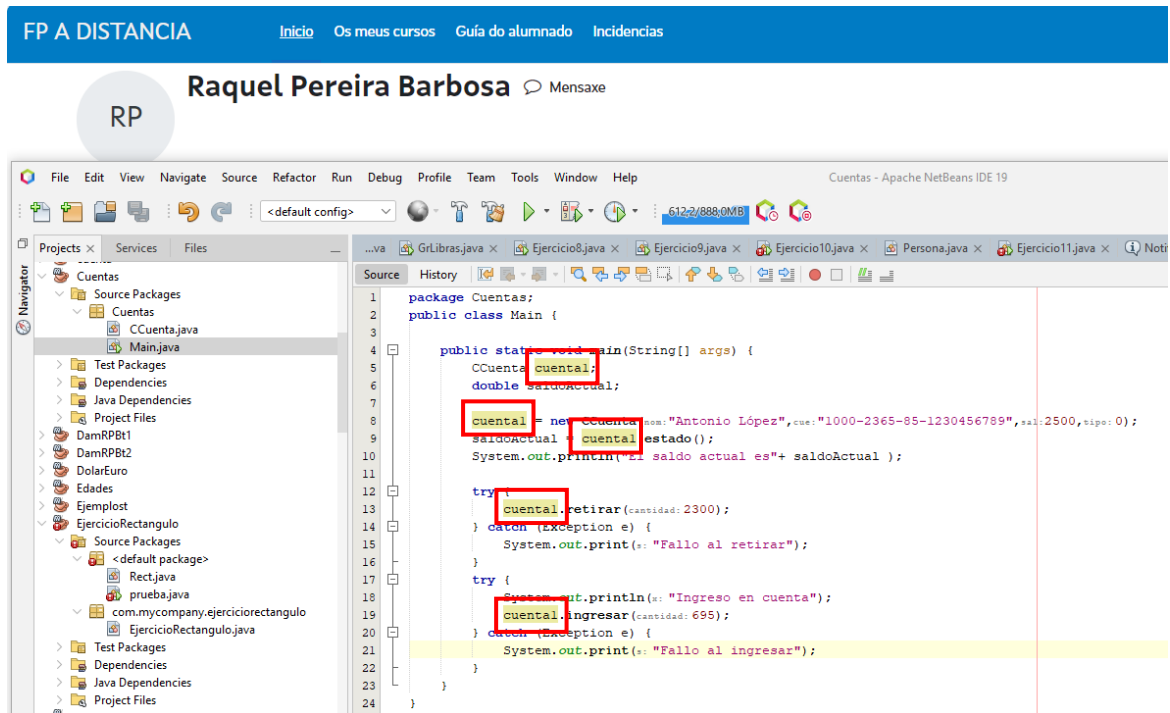
Para hacer este cambio marcaremos el primer miCuenta y pulsaremos botón derecho, ahí iremos a Refactorizar - Renombrar:



A continuación, ponemos el nuevo nombre "cuenta1" y le damos a refactorizar:

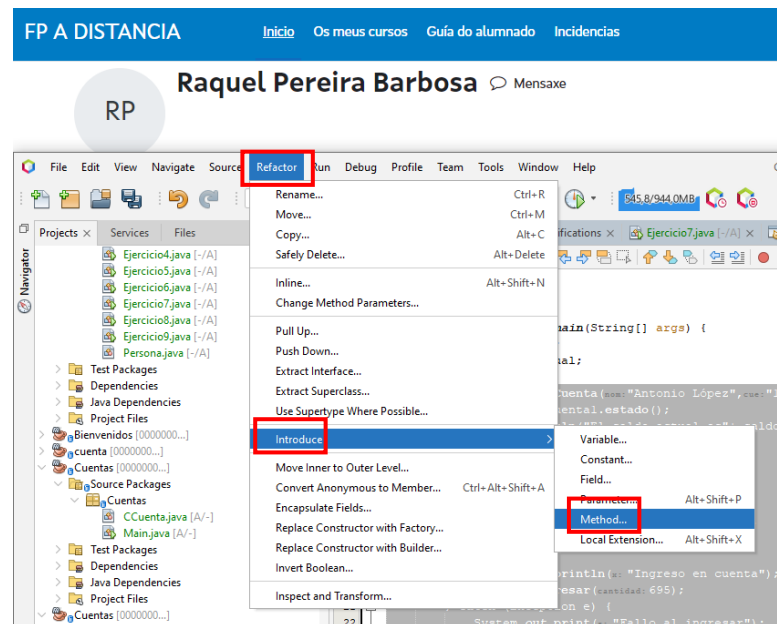


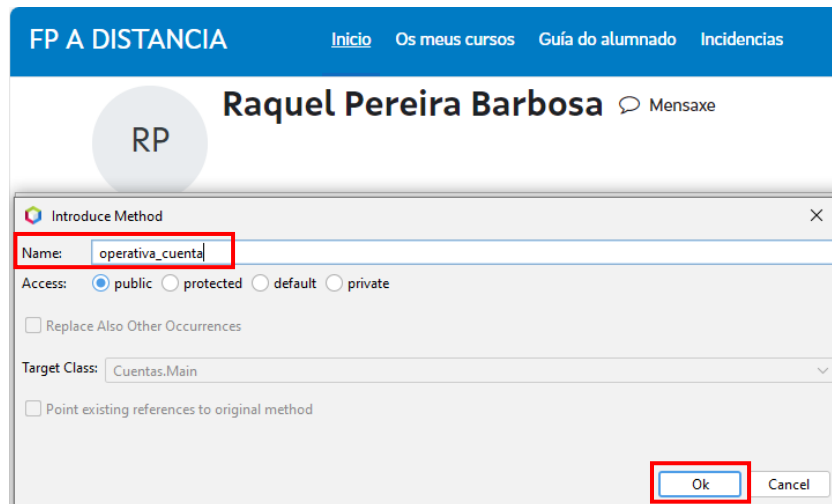
Y con esto nos cambia todos los nombres de la variable a la vez



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

Para esta parte seleccionaremos la parte del código donde queremos introducir el método y le damos a refactorizar/introducir/método y le ponemos el nombre

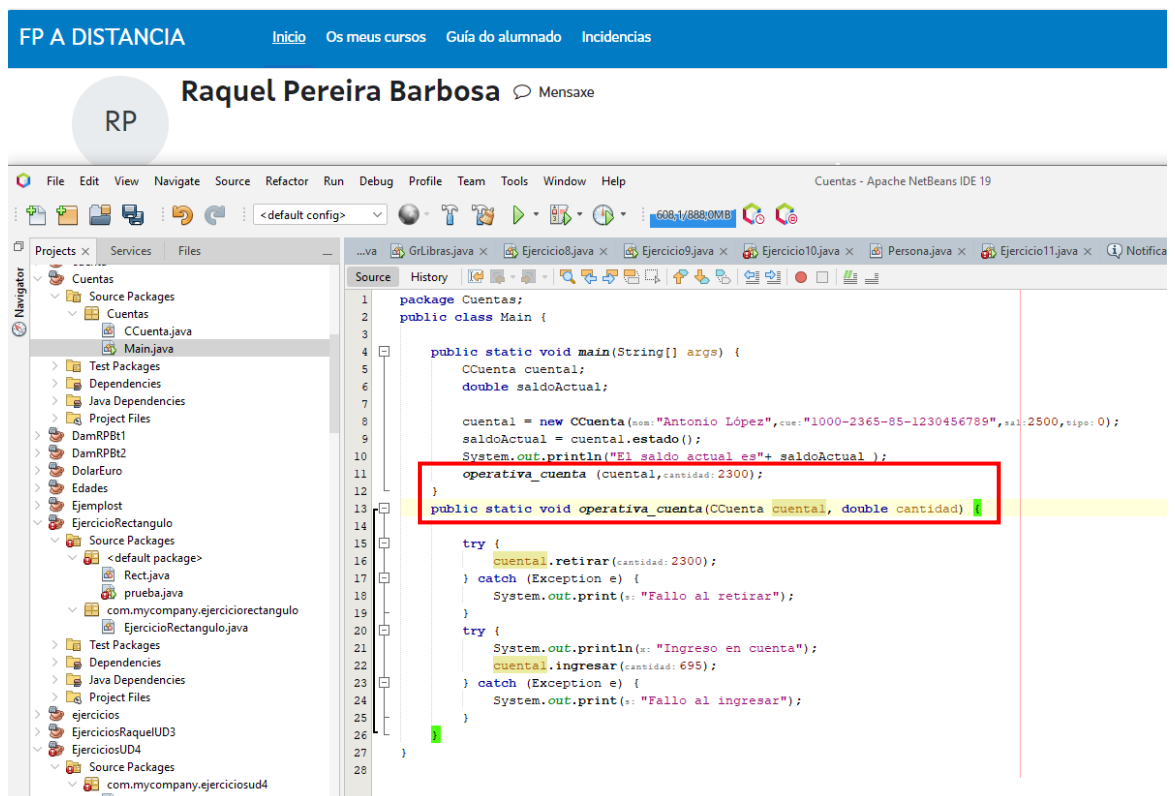




Nos ha creado lo solicitado

Igualmente, también en este caso que no es mucho código podríamos hacerlo a mano para ajustarlo a nuestro gusto. Para esto en este código, se ha introducido el método `operativa_cuenta` que recibe como parámetros una instancia de `CCuenta` llamada `cuenta` y un valor `cantidad` de tipo `double`.

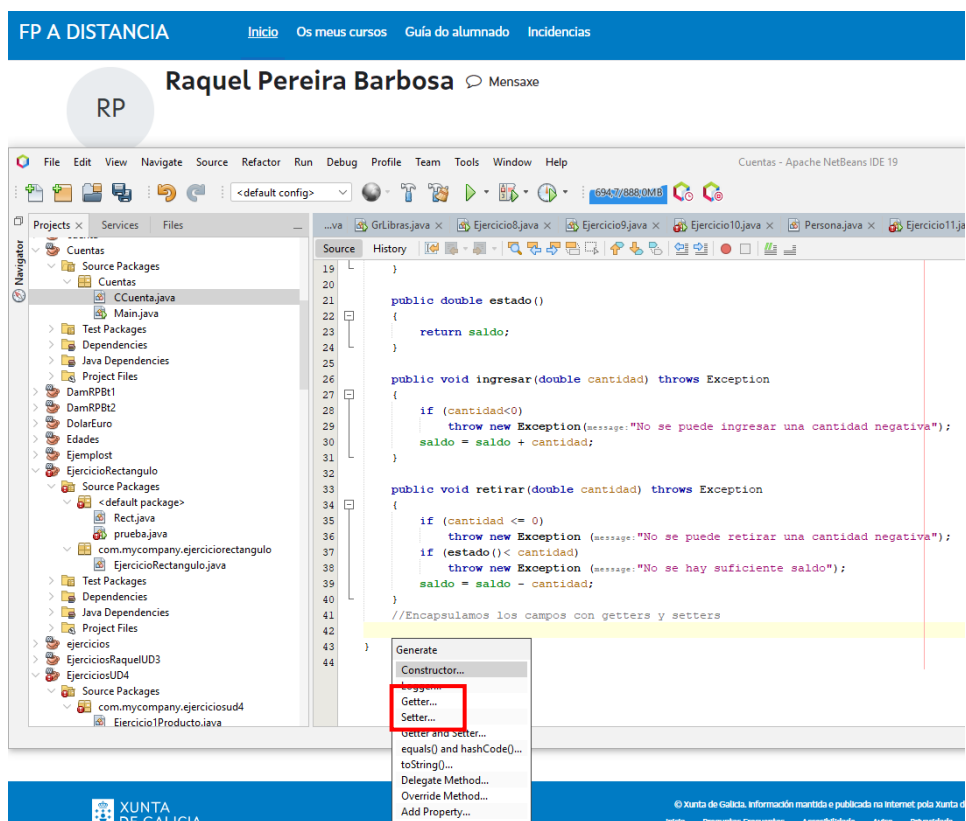
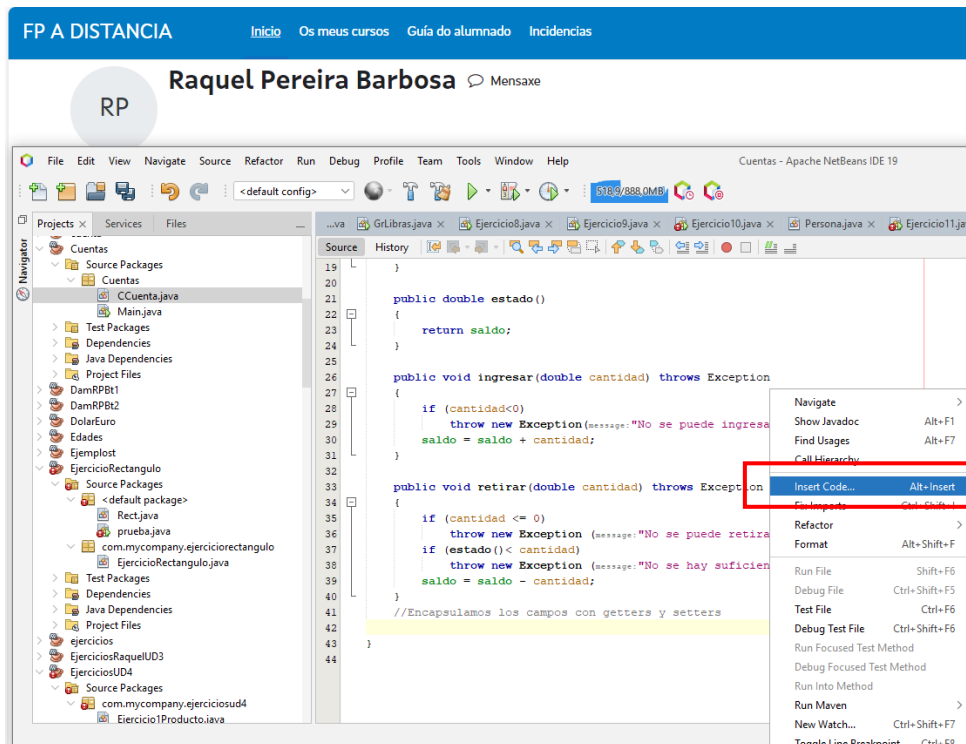
Luego, en el método `main`, se invoca `operativa_cuenta` pasando la instancia `cuenta1` y la cantidad `2300`.



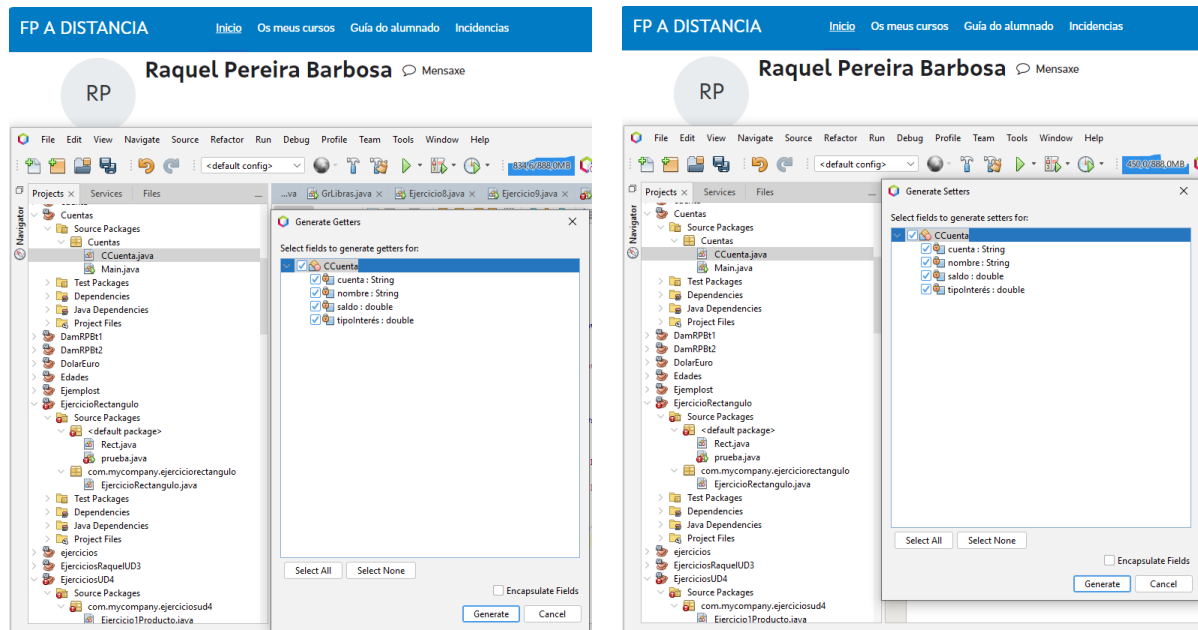
#### 4. Encapsular los atributos de la clase CCuenta.

Para esto crearemos los métodos de asignación y de consulta getters y setters para los campos de la clase. Hay dos formas de realizarlo:

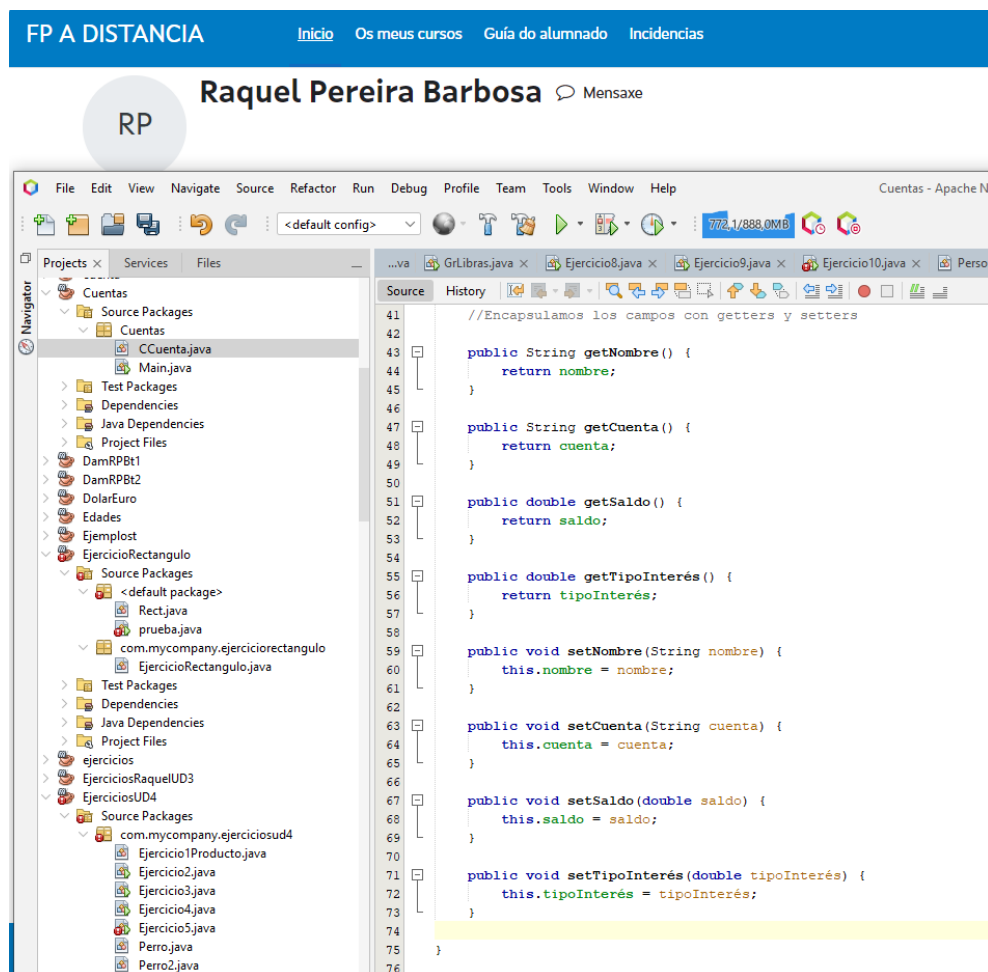
1.-Lo haremos de forma automática, nos posicionaremos donde queremos incluirlo, botón derecho, añadir código:



Seleccionamos getter y setter y lo aplicamos a toda CCuenta y le damos a generar

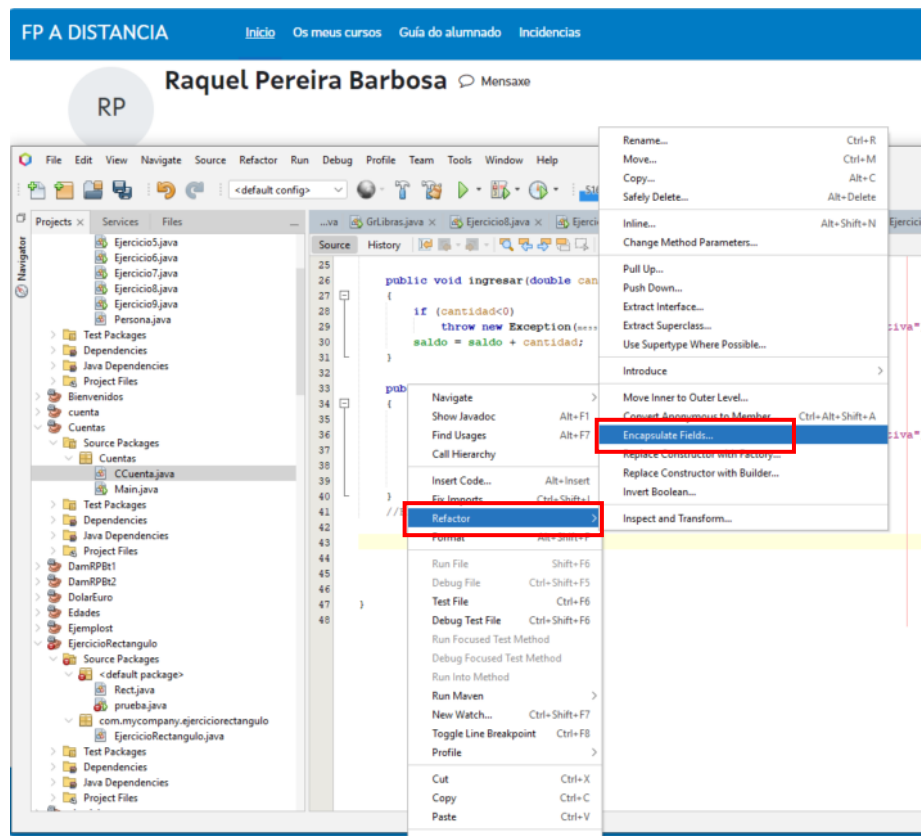


Esto generará todo el código automáticamente:

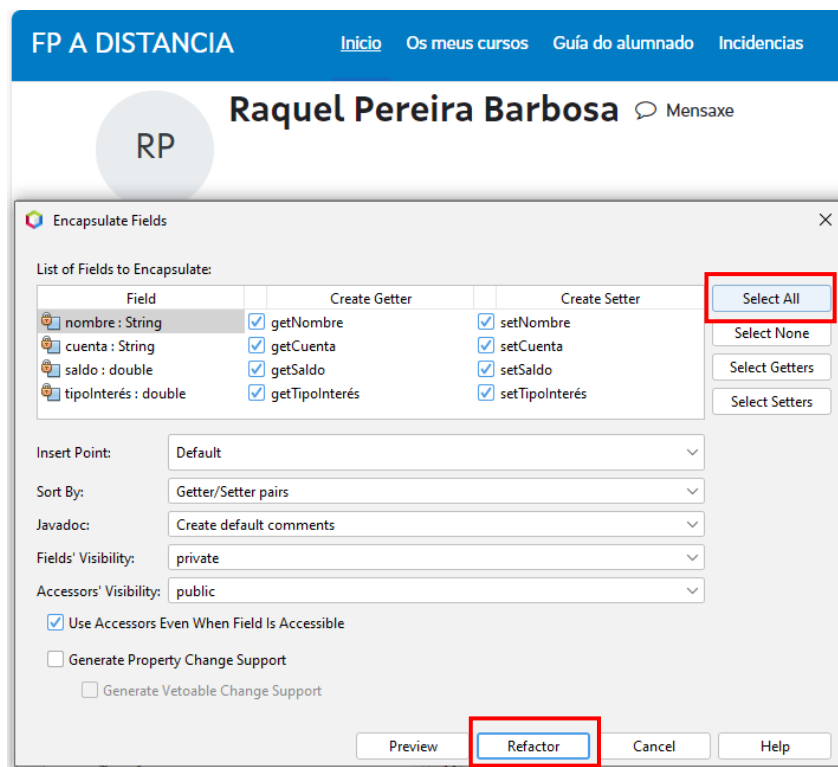




2.-Otra manera de hacerlo es a través de la opción de refactorización, nos posicionamos donde queremos ubicarlo y botón derecho refactorizar, encapsular campos:

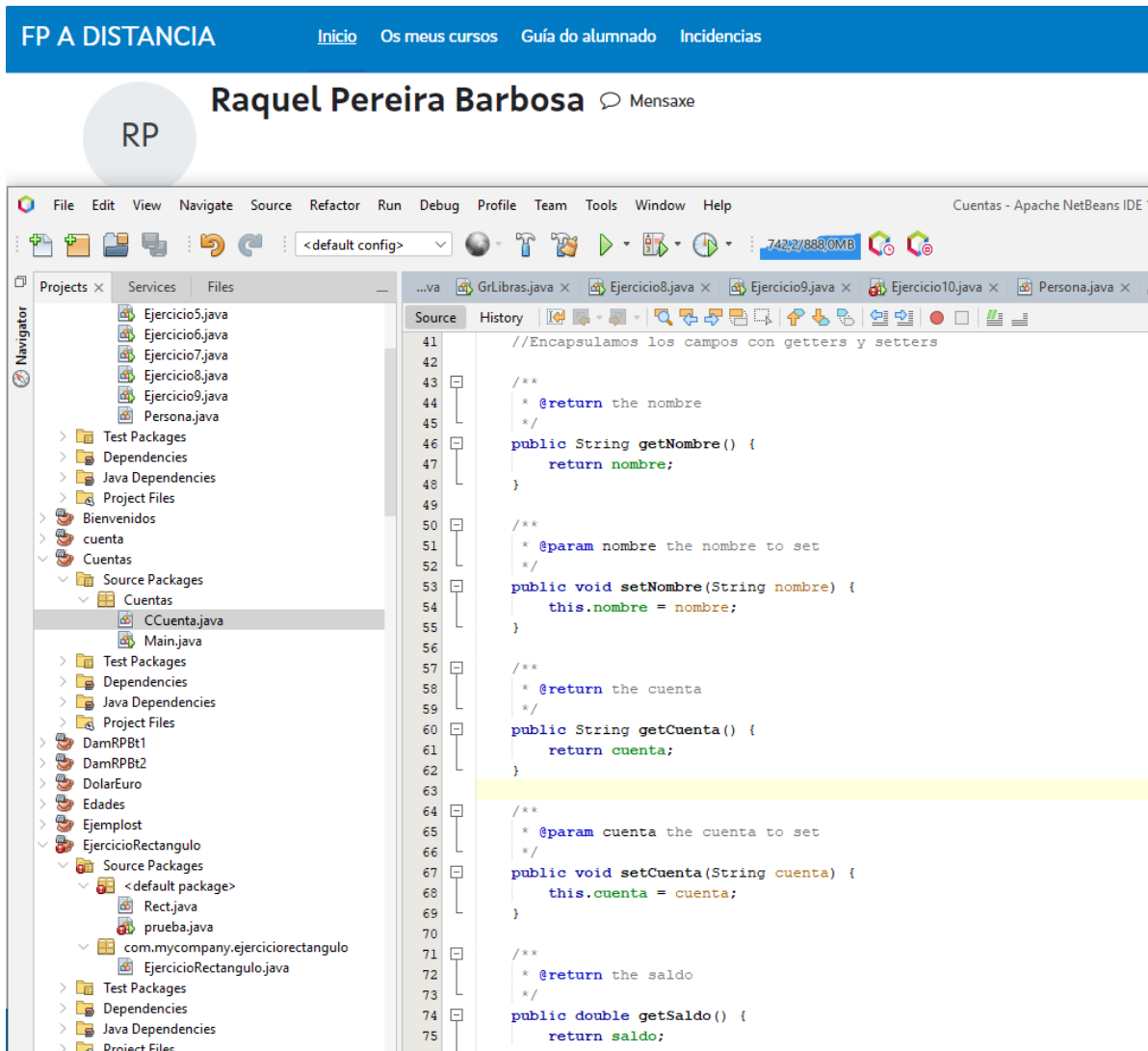


Seleccionamos todo y le damos a refactorizar:





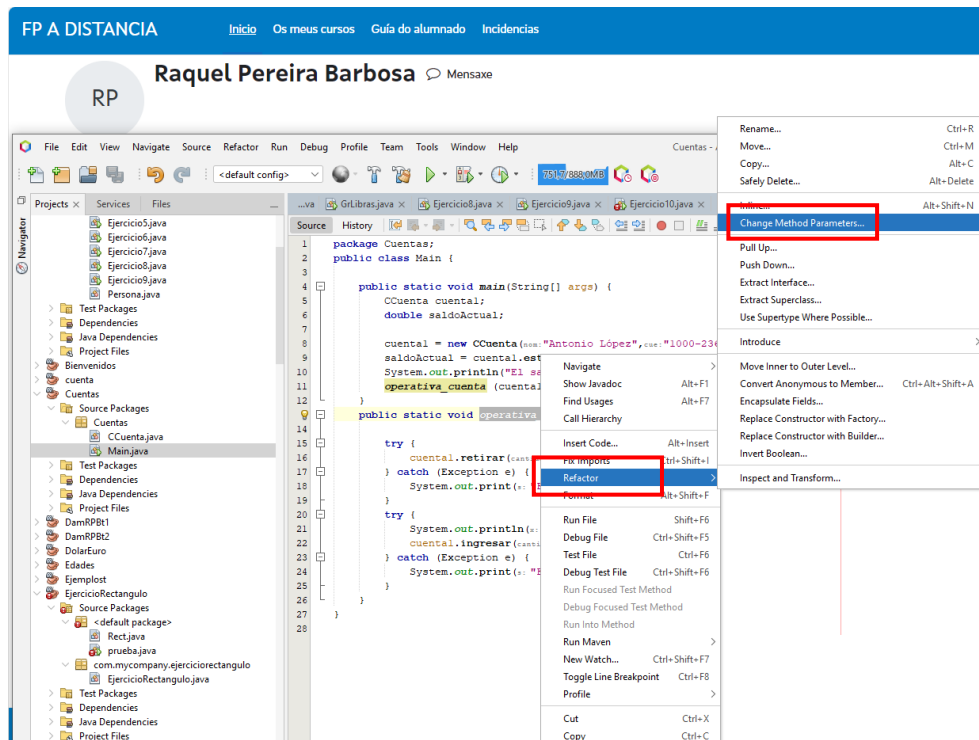
De esta manera nos insertará comentarios para poder entender el código:



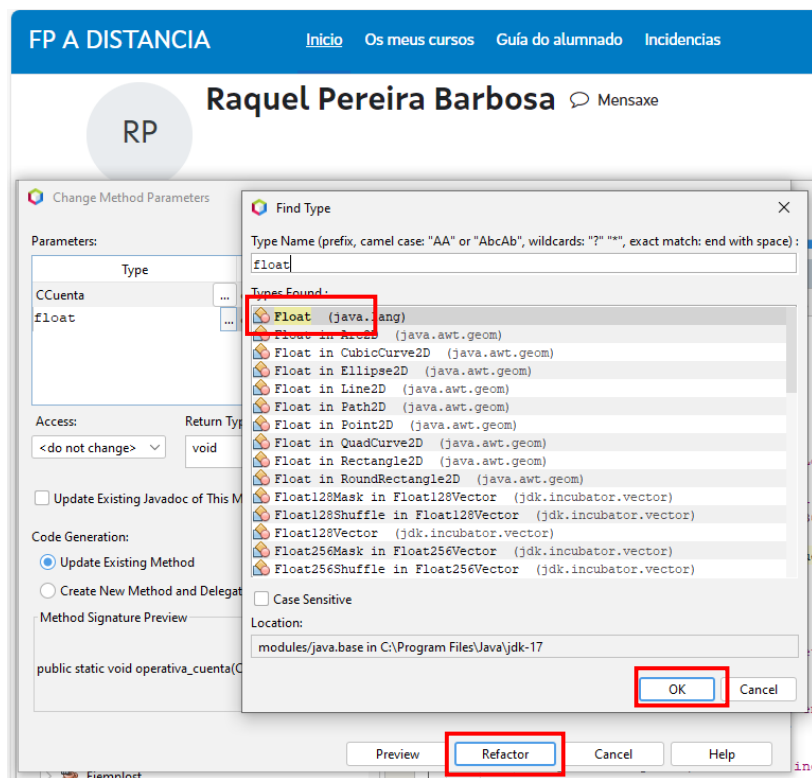
(Esto es una muestra ya que el código es más largo y no entra en un solo pantallazo)

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

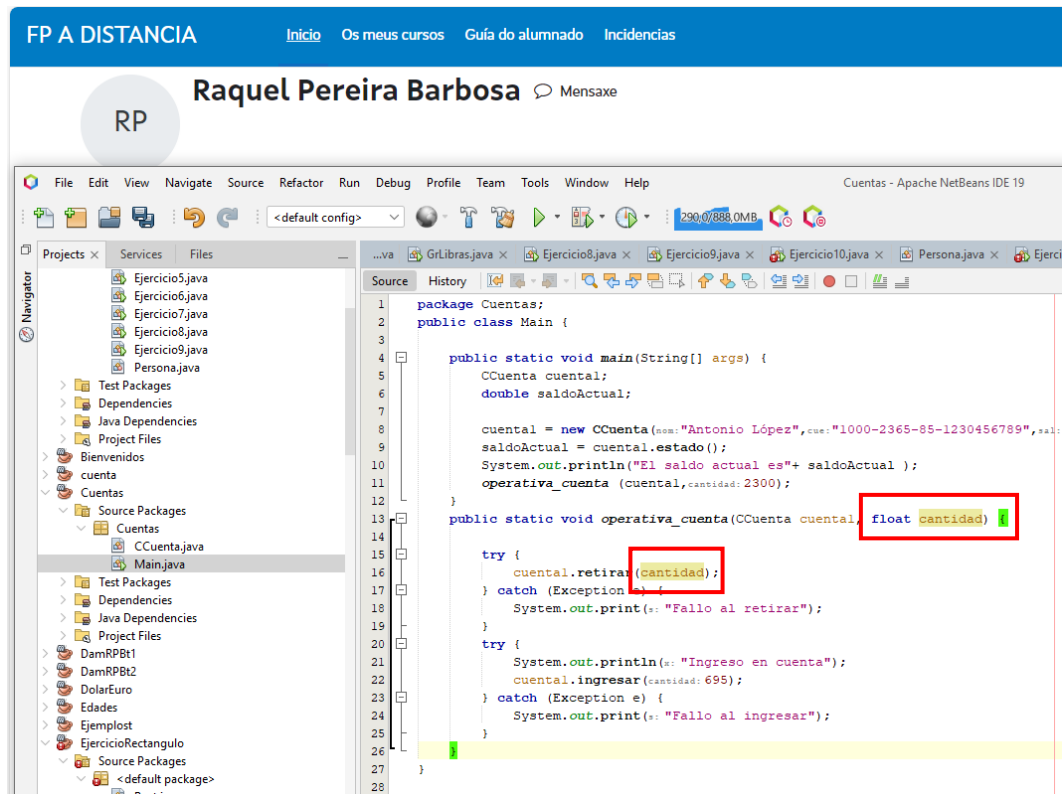
Para hacer este cambio en vez de hacerlo directamente sobre el código también lo haremos a través de la opción de refactorizar. Seleccionamos en nombre del método "operativa\_cuenta", botón derecho refactorizar, cambiar los parámetros del método:



Cambiamos el double por float y refactorizamos



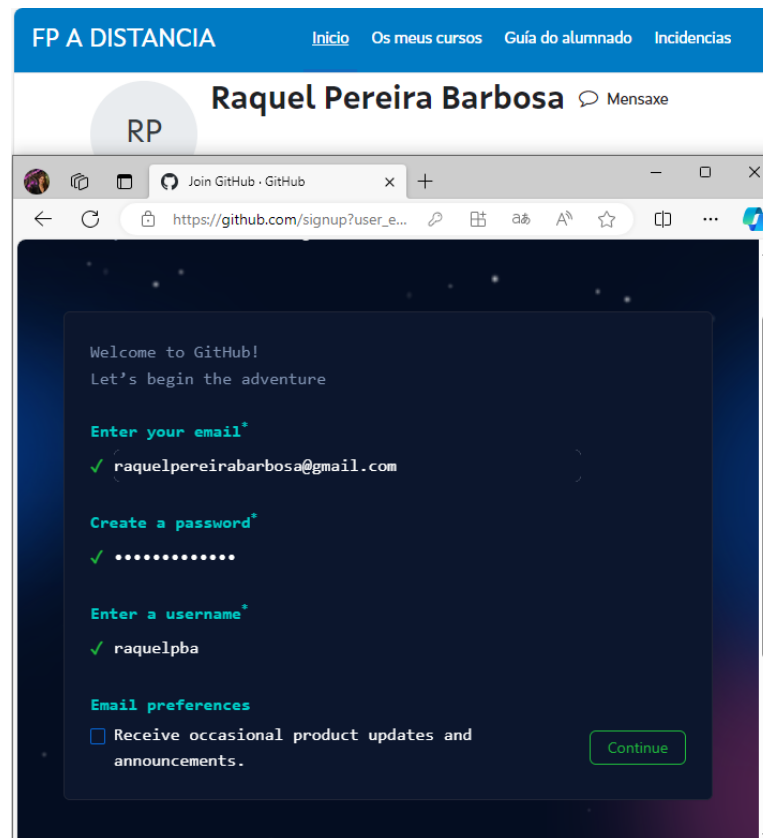
Ahora actualizamos la referencia al parámetro



## GIT

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

Para crear un repositorio en GitHub, primero daremos de alta una cuenta:



Una vez que ya tenemos una cuenta creamos un repositorio que llamaremos cuentas como nuestro proyecto

FP A DISTANCIA Inicio Os meus cursos Guía do alumnado Incidencias

Raquel Pereira Barbosa Mensaxe

New repository

https://github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \* raquelpba

Repository name \* Cuentas

Great repository names are short and memorable. Need inspiration? How about [jubilant-octo-rotary-phone](#)?

Description (optional) Tarea UD4 Entornos

☐ Public  
Anyone on the internet can see this repository. You choose who can commit.

☒ Private  
You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore  
.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license  
License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

🔒 You are creating a private repository in your personal account.

Create repository

Una vez creado copiamos la URL generada <https://github.com/raquelpba/Cuentas.git> (es privada, me hace falta el usuario de github del profesor para que tenga acceso)

FP A DISTANCIA Inicio Os meus cursos Guía do alumnado Incidencias

Raquel Pereira Barbosa Mensaxe

raquelpba / Cuentas

Code Issues Pull requests Actions Projects Security Insights Settings

Cuentas Private

Unwatch 1 Fork 0 Star 0

Set up GitHub Copilot  
Use GitHub's AI pair programmer to autocomplete suggestions as you code.  
Get started with GitHub Copilot

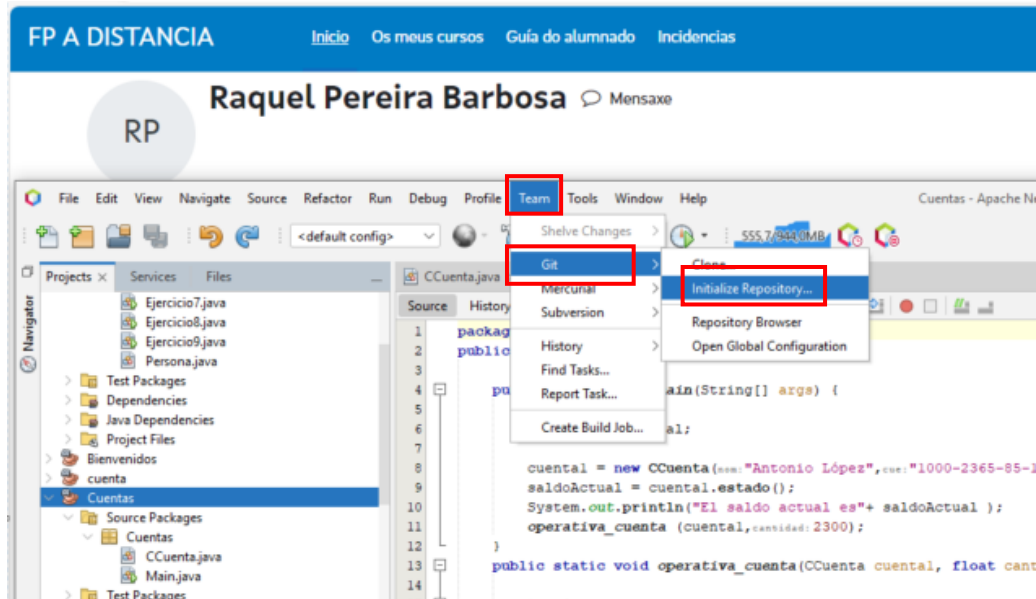
Add collaborators to this repository  
Search for people using their GitHub username or email address.  
Invite collaborators

Quick setup — if you've done this kind of thing before

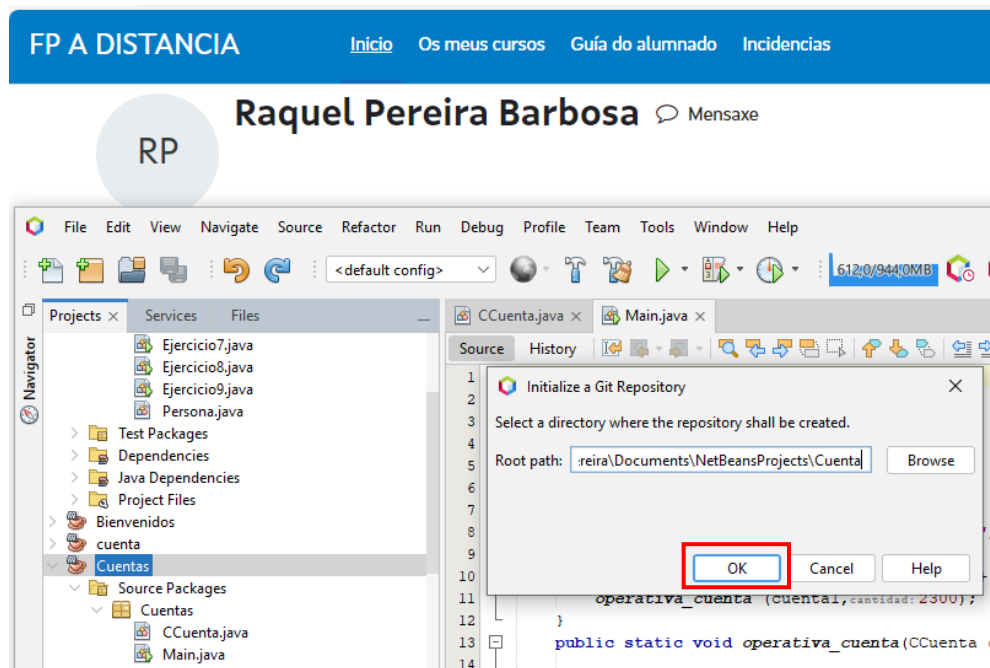
Set up in Desktop or HTTPS SSH <https://github.com/raquelpba/Cuentas.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

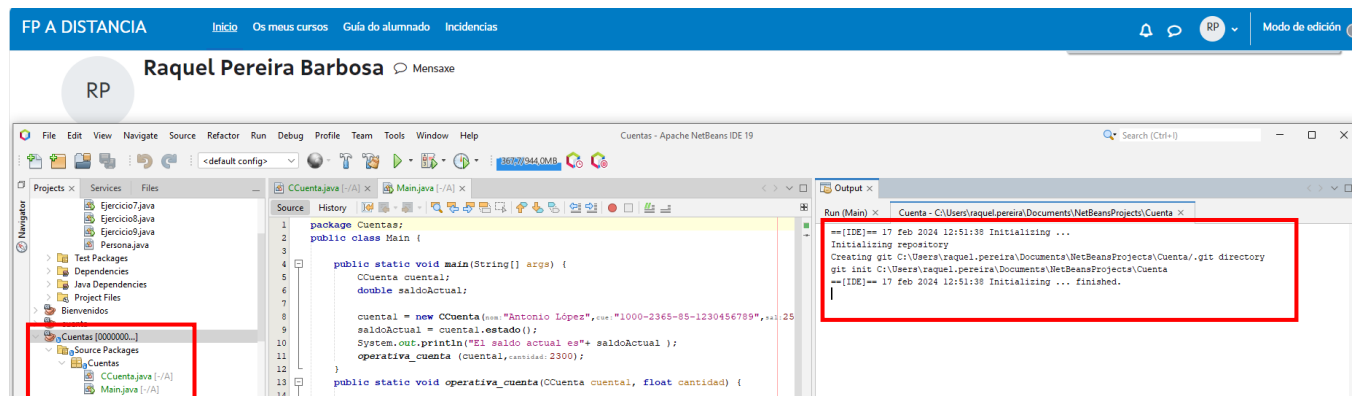
Ahora nos vamos a Netbeans y vamos a Team-Git-Inicializar repositorio



Aquí nos sale la ubicación de nuestro proyecto y es la ubicación que mantendremos así que le decimos ok



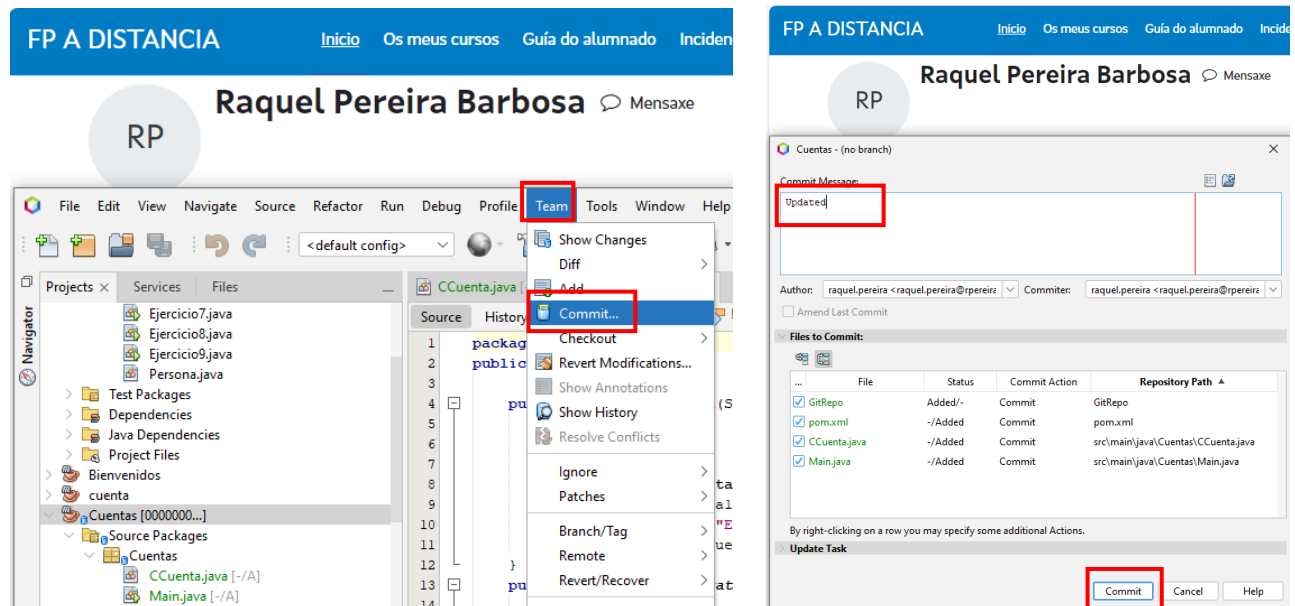
Podemos comprobar que nos ha creado el repositorio en el panel de salida. También nos fijamos que salen con el icono del repositorio nuestro proyecto.



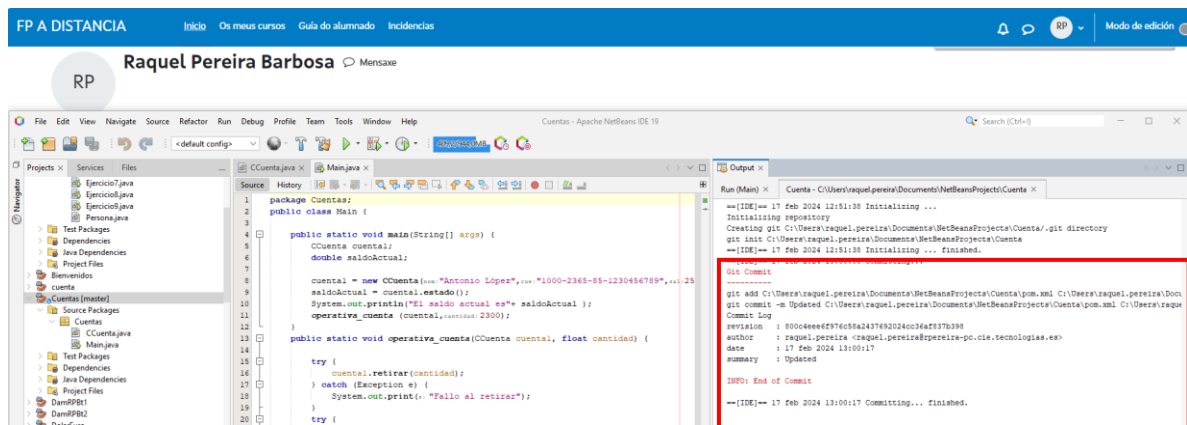
Con esto nos queda configurado GIT para el proyecto y creado el repositorio público

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

Ahora nos dirigiremos a Team, commit para indicar un mensaje para que se puedan identificar los cambios antes de subirlos al repositorio

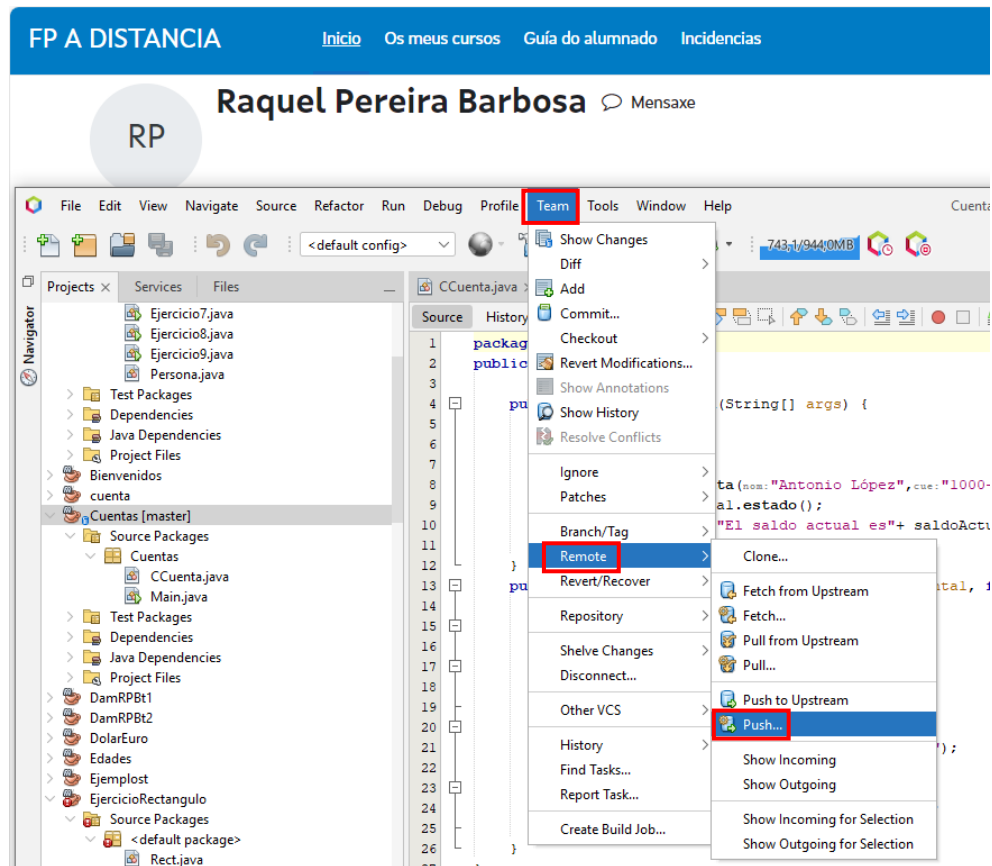


Podemos ver en el panel de salida el inicio y el fin del commit



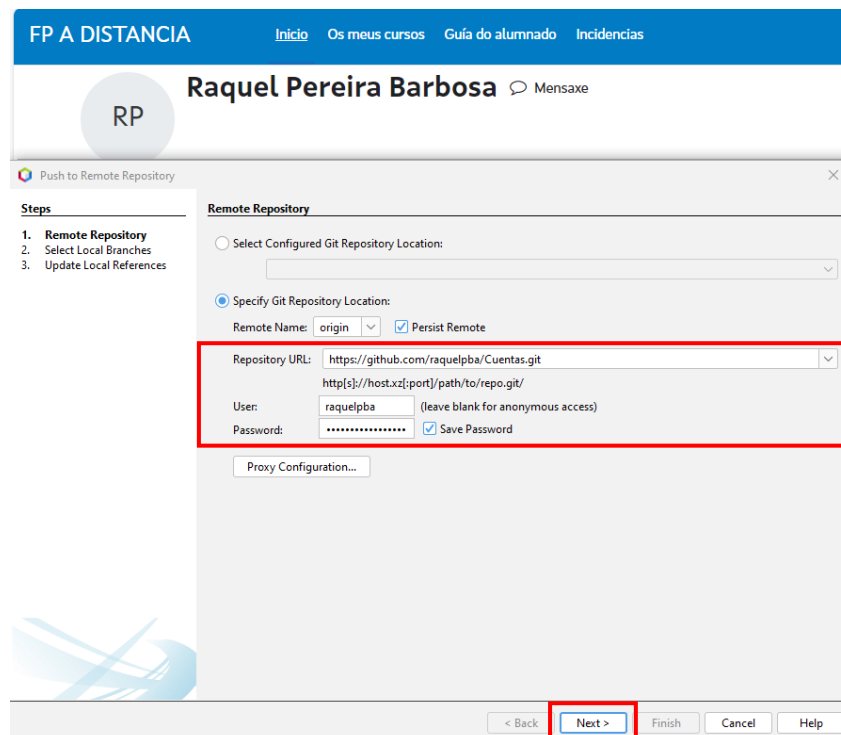


Para que esto tenga sentido subiremos los cambios al repositorio, para eso vamos a Team, Remoto, Push

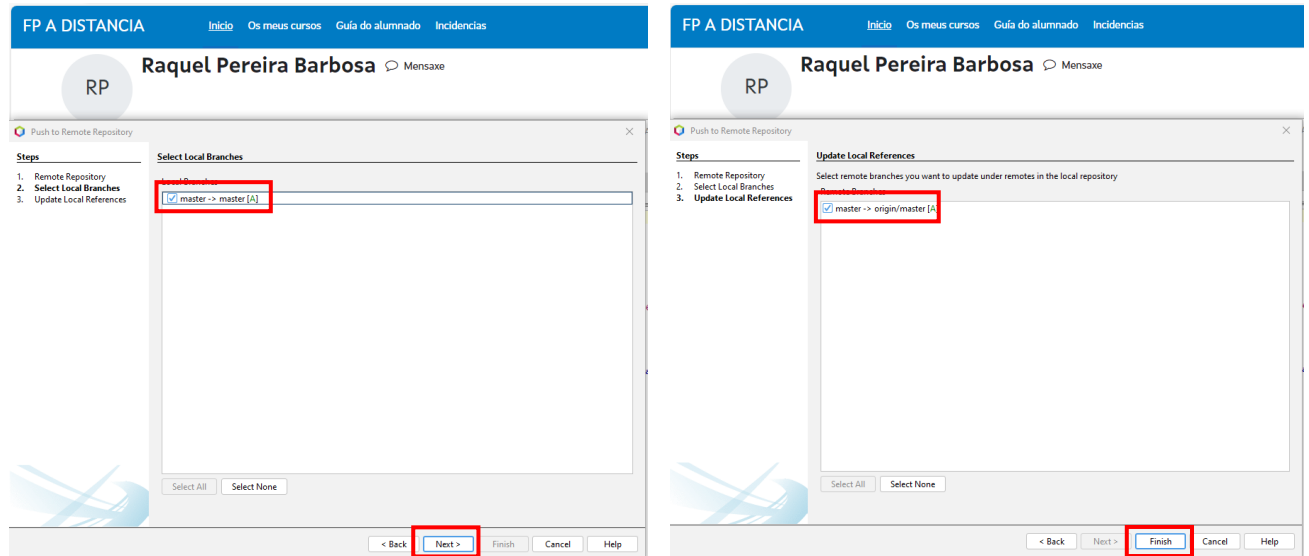


Aquí nos pedirá la URL de nuestro repositorio y nos solicitará nuestras credenciales.

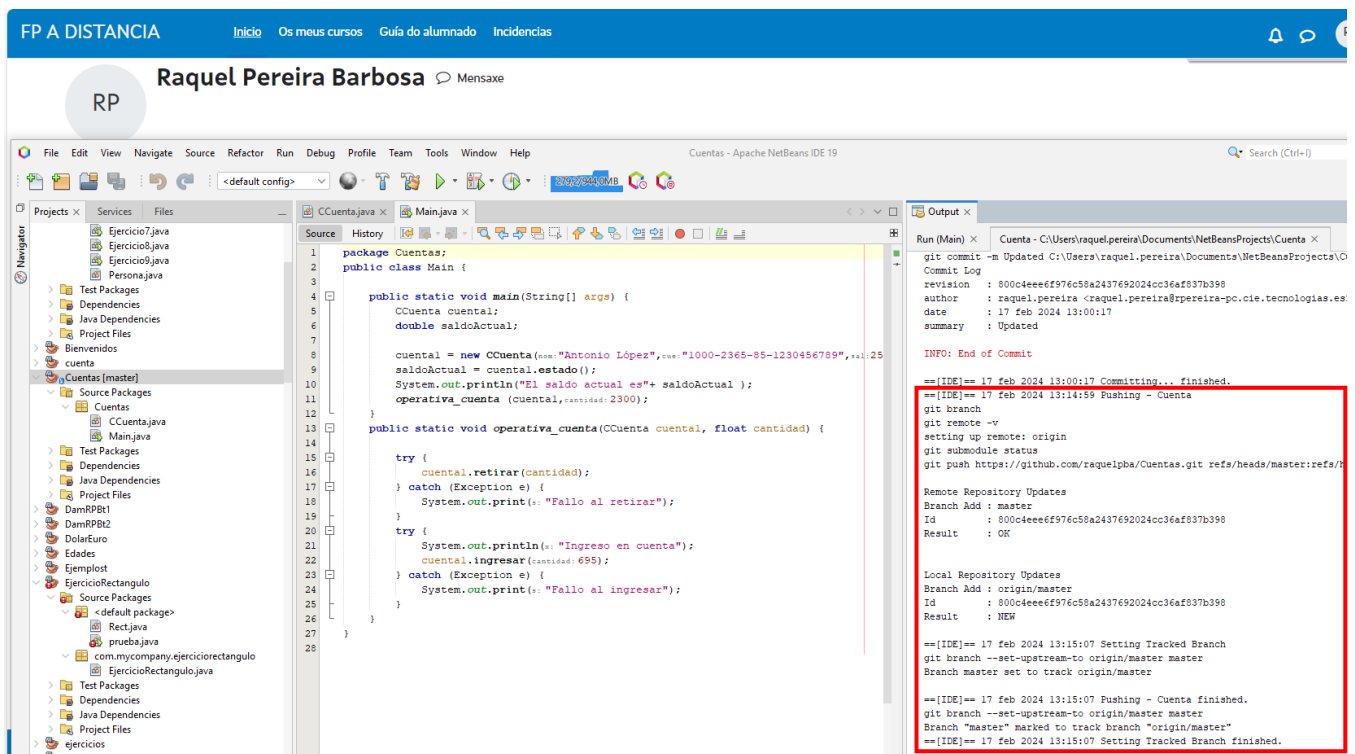
OJO!!!! En las credenciales no valdrá con nuestra contraseña de github, sino que hay que generar un token de seguridad.



Seleccionamos nuestra rama master y continuamos, actualizamos la rama y finalizamos

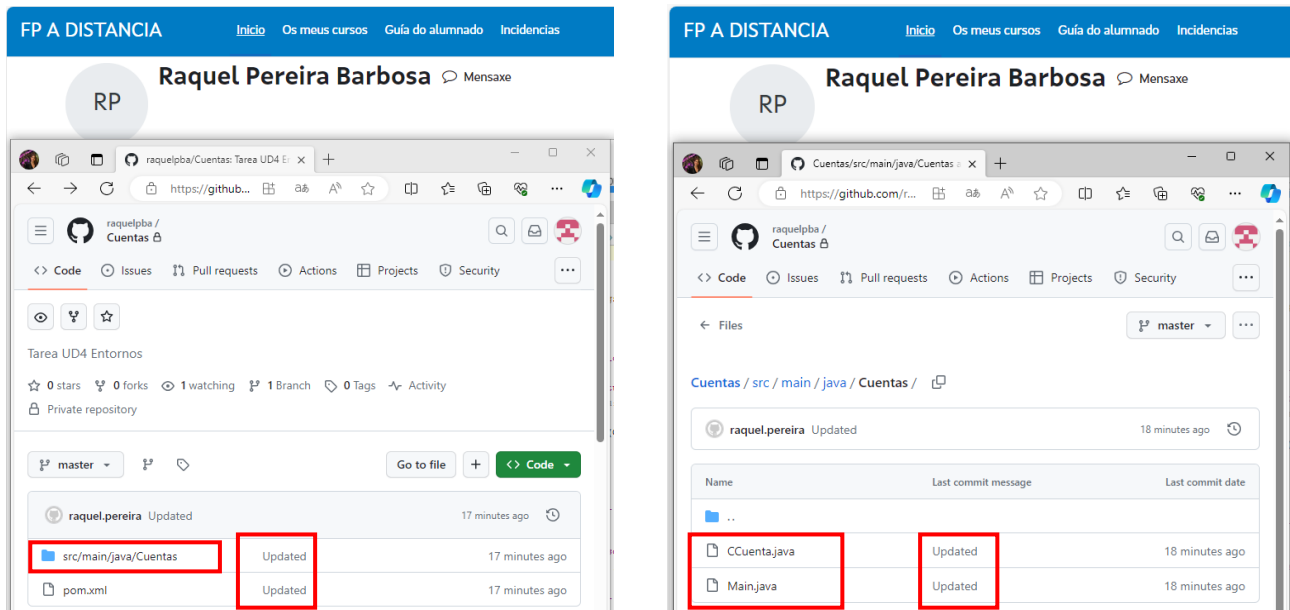


Ahora esperamos a que salga el mensaje en el panel de salida de que se ha subido correctamente

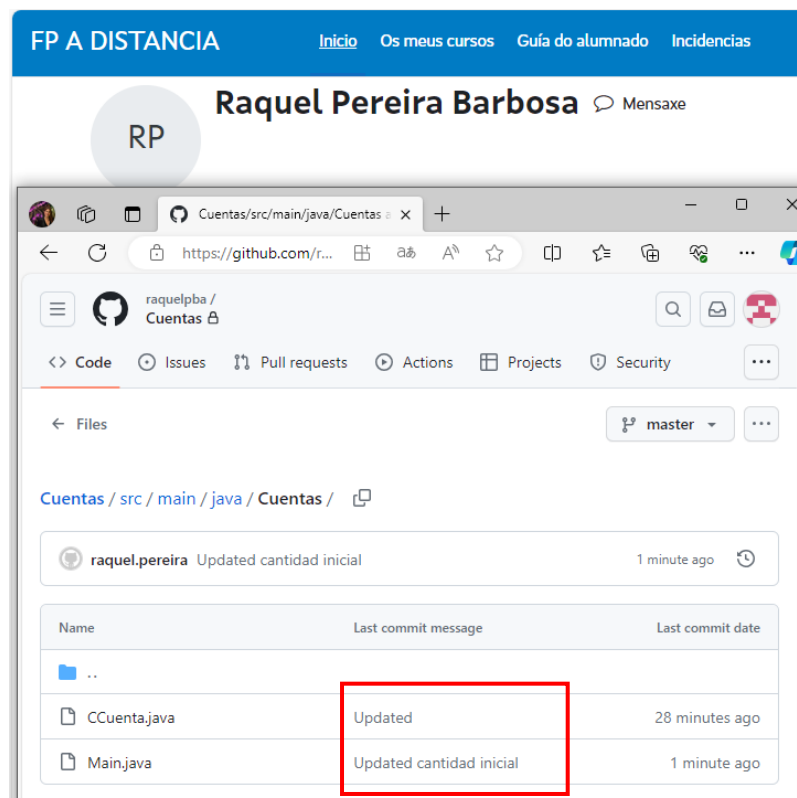


Ahora comprobaremos en nuestro Github que nuestro proyecto está subido, y vemos que así es

Nos fijamos que el mensaje que pusimos en commit aparece en la columna "Last commit message"

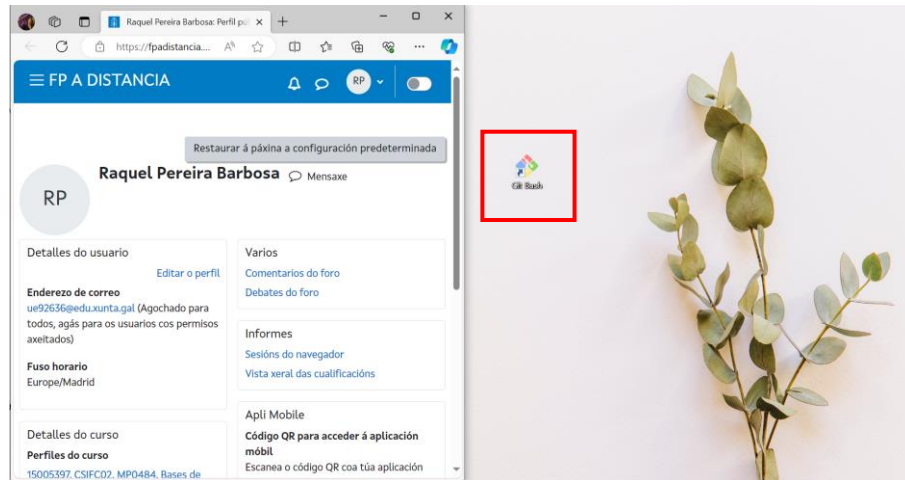


Hacemos otro commit y push para hacer una prueba de que nos indica el cambio



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

Para esto hemos instalado GIT durante la revisión de la unidad:



Lo abrimos y nos ubicamos en nuestra carpeta donde está el proyecto y donde está nuestro repositorio local:

1. Sincronizamos nuestro repositorio remoto con el local, para ello hacemos un git pull
2. Para obtener el historial de versiones una vez sincronizado utilizaremos el comando git log

Y con esto podemos ver todos los commits que hemos realizado:

```

FP A DISTANCIA Inicio Os meus cursos Guía do alumnado Incidencias

Raquel Pereira Barbosa Mensaxe

MINGW64:/C:/Users/raquel.pereira/Documents/NetBeansProjects/Cuentas

raquel.pereira@rpereira-pc MINGW64 /C:/Users/raquel.pereira/Documents/NetBeansProjects/Cuentas (master)
$ git pull
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 17 (delta 4), reused 16 (delta 3), pack-reused 0
Unpacking objects: 100% (17/17), 2.09 KiB | 33.00 KiB/s, done.
From https://github.com/raquelpba/Cuentas
* [new branch]      master    -> origin/master

raquel.pereira@rpereira-pc MINGW64 /C:/Users/raquel.pereira/Documents/NetBeansProjects/Cuentas (master)
$ git log
commit 1240ce3abcc25a08a90828030e8f0454afdf1605 (HEAD -> master, origin/master)
Author: raquel.pereira <raquel.pereira@rpereira-pc.cie.tecnologias.es>
Date: Sat Feb 17 13:27:30 2024 +0100

    Updated cantidad inicial

commit 9f6db6ec0a9954a8d324b16294047922fe9ccf5
Author: raquel.pereira <raquel.pereira@rpereira-pc.cie.tecnologias.es>
Date: Sat Feb 17 13:26:49 2024 +0100

    Updated cantidad

commit 800c4eee6f976c58a2437692024cc36af837b398
Author: raquel.pereira <raquel.pereira@rpereira-pc.cie.tecnologias.es>
Date: Sat Feb 17 13:00:17 2024 +0100

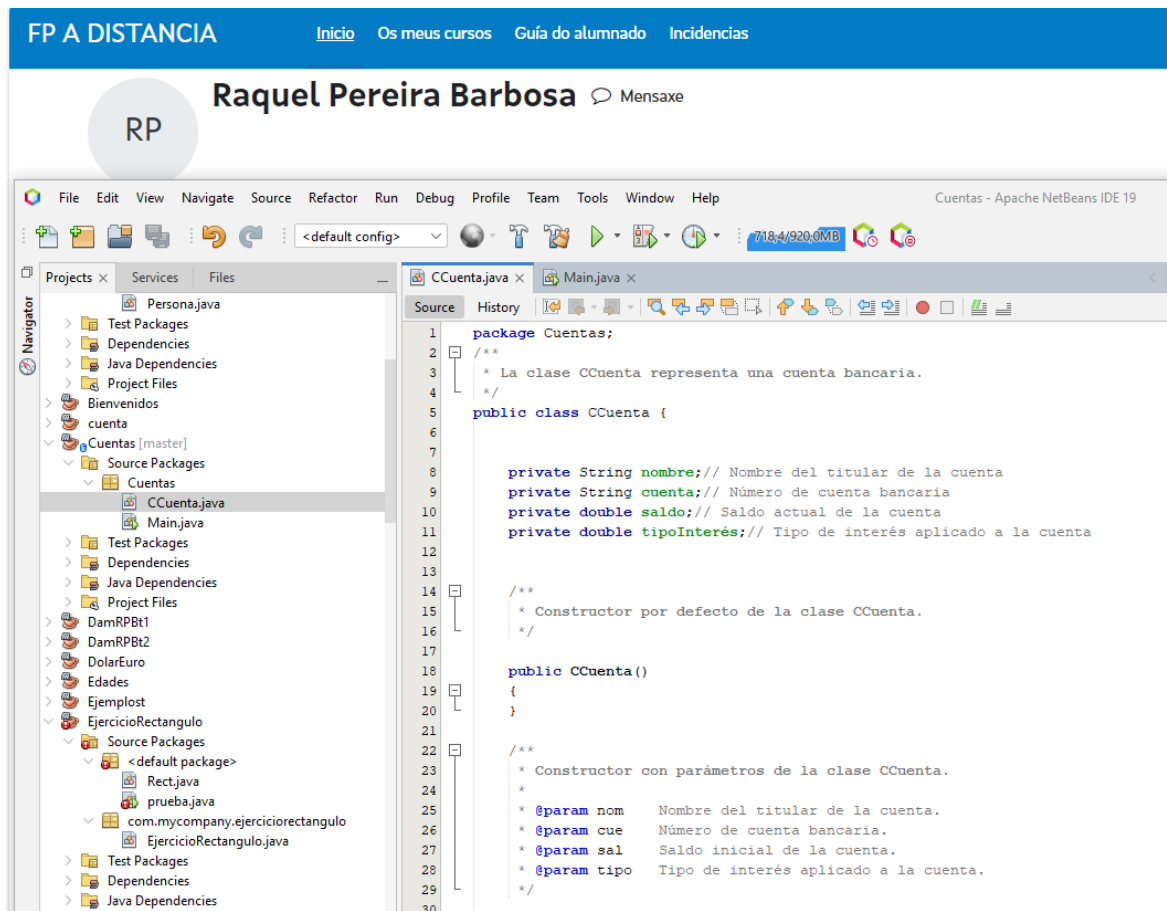
    Updated

```

## JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.

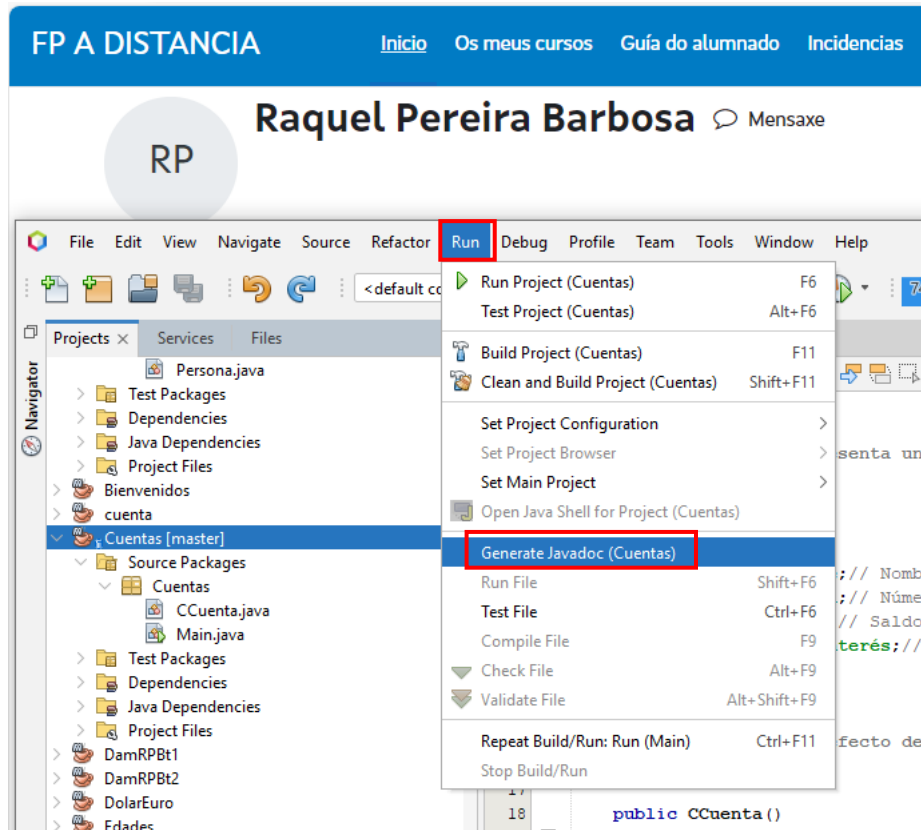
Añadimos algún comentario a mayores de los generados en la refactorización de getters y setters, y generamos comentarios para documentar parámetros de métodos y constructores.



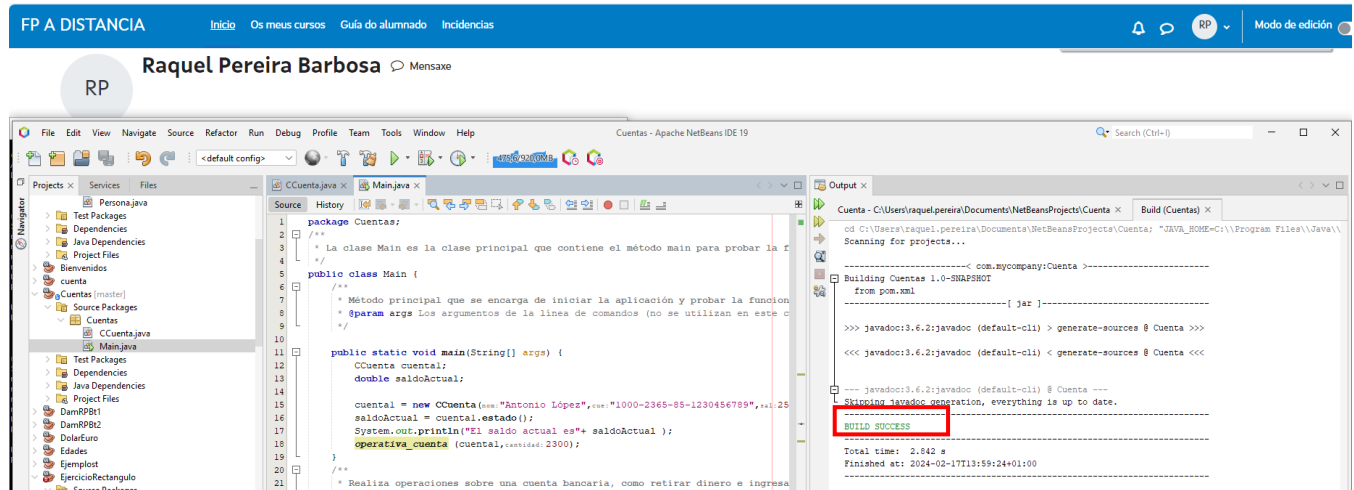
(Como adjuntaremos el proyecto a la tarea ahí se podrán ver todos los comentarios)

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

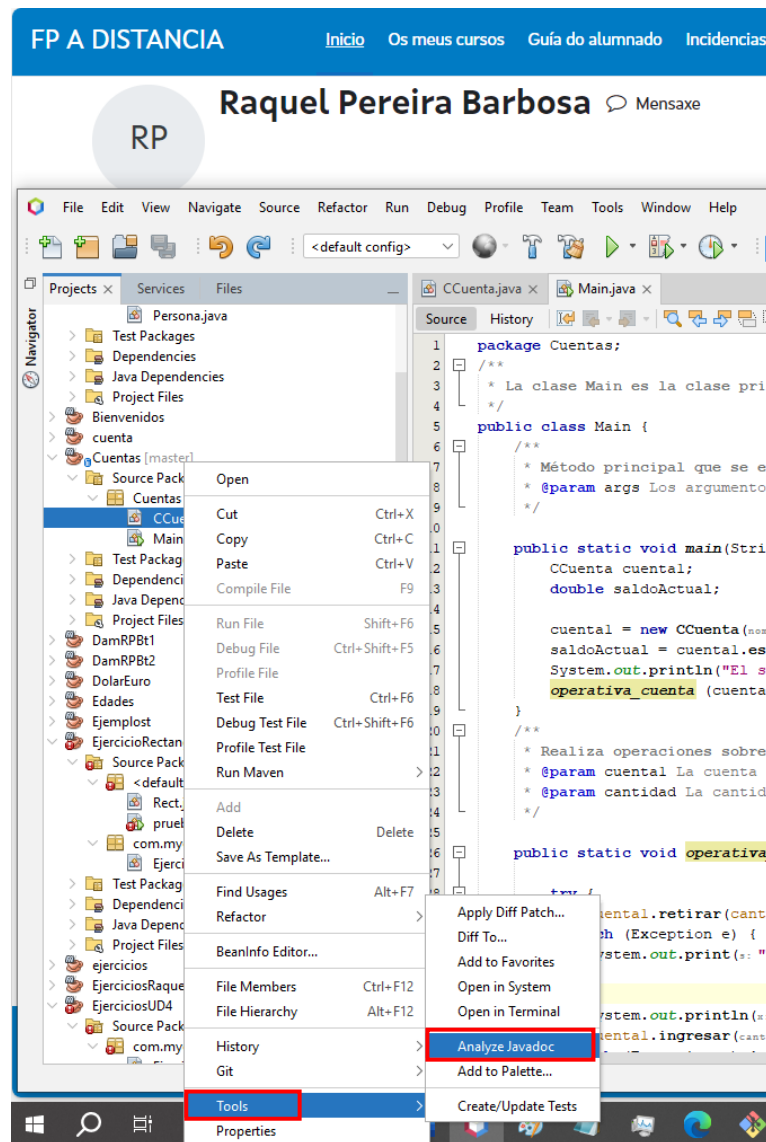
Para generar el Javadoc nos dirigimos a la pestaña Run. Generar Javadoc



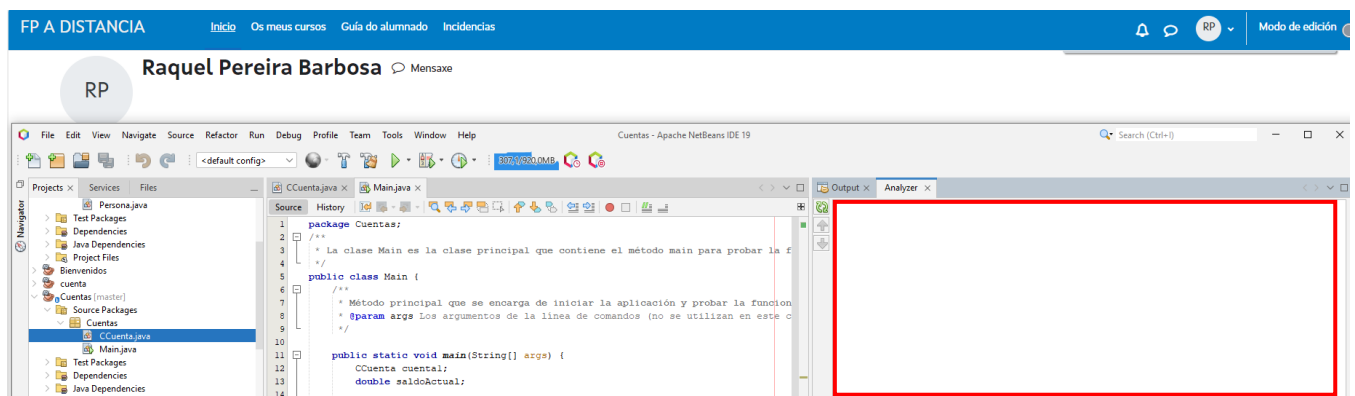
Vemos que abarca todos los métodos y atributos de la clase cuenta ya que lo genera sin ningún error:



Además, si vamos a nuestro fichero CCuentas y analizamos el Javadoc no nos da ninguna opción que nos falte, por tanto, está todo incluido:

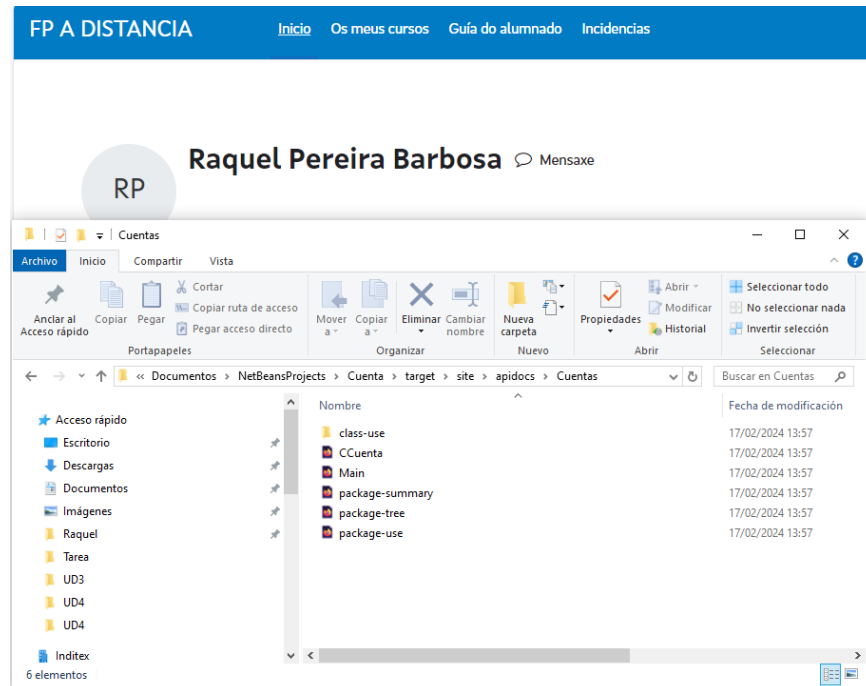


No da opciones de cambio por lo que está todo correcto

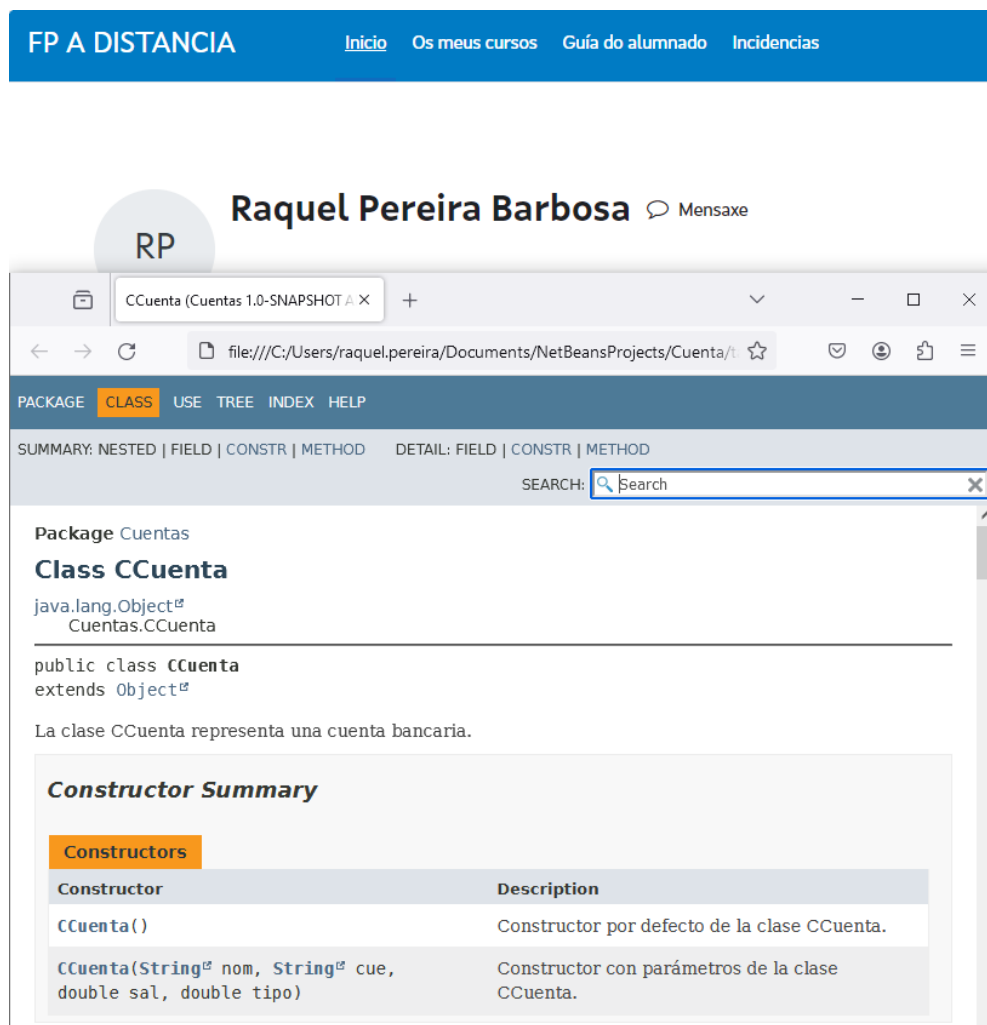




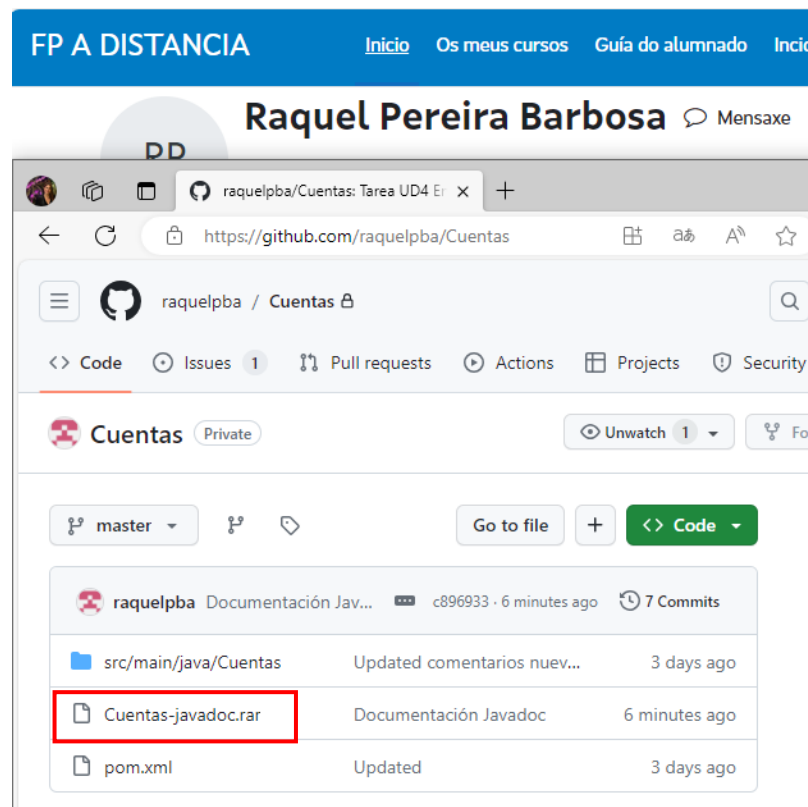
Nuestro Javadoc lo podemos encontrar en la carpeta que ha generado netbeans para nuestro proyecto dentro de apidocs



Lo abrimos y vemos que abre perfectamente, lo adjuntamos en esta entrega y lo podemos encontrar también en nuestro GitHub



Nuestro JavaDoc lo hemos subido a GitHub en un comprimido con todo lo necesario para que se pueda visualizar correctamente:



Hemos añadido una issue para explicarlo correctamente:

