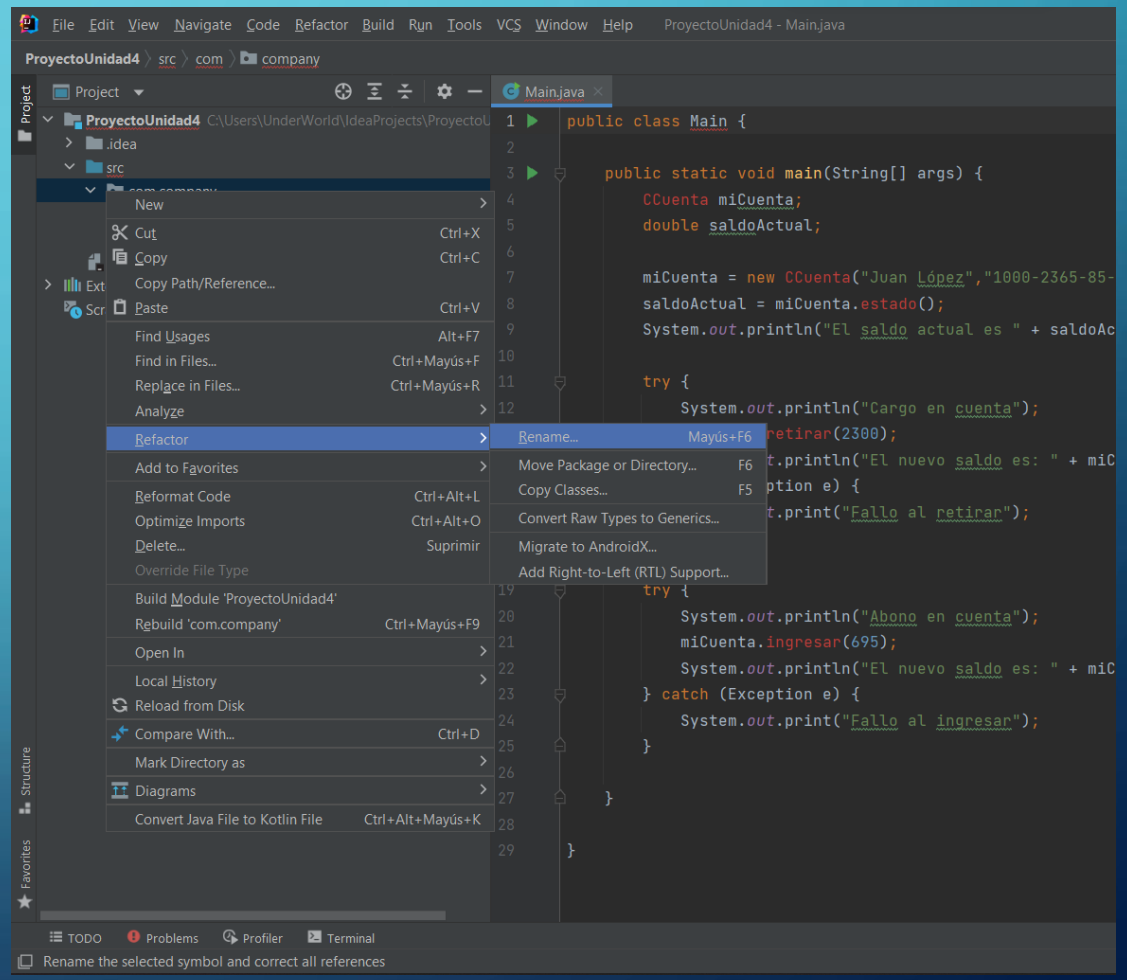
PRÁCTICA FINAL UT4: OPTIMIZACIÓN Y DOCUMENTACIÓN.

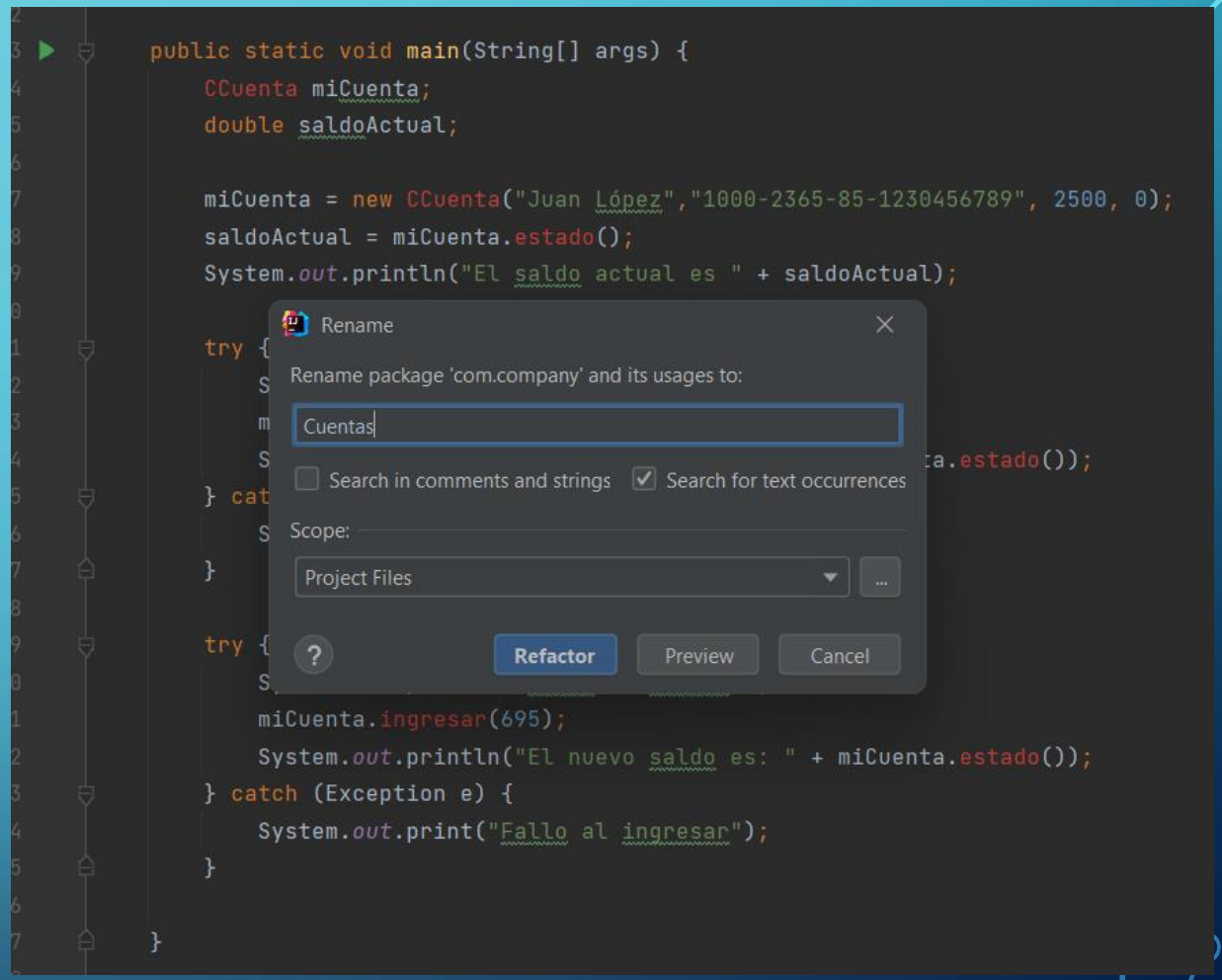
FABIO YAMUZA GONZALEZ

REFACTORIZAR CODIGO

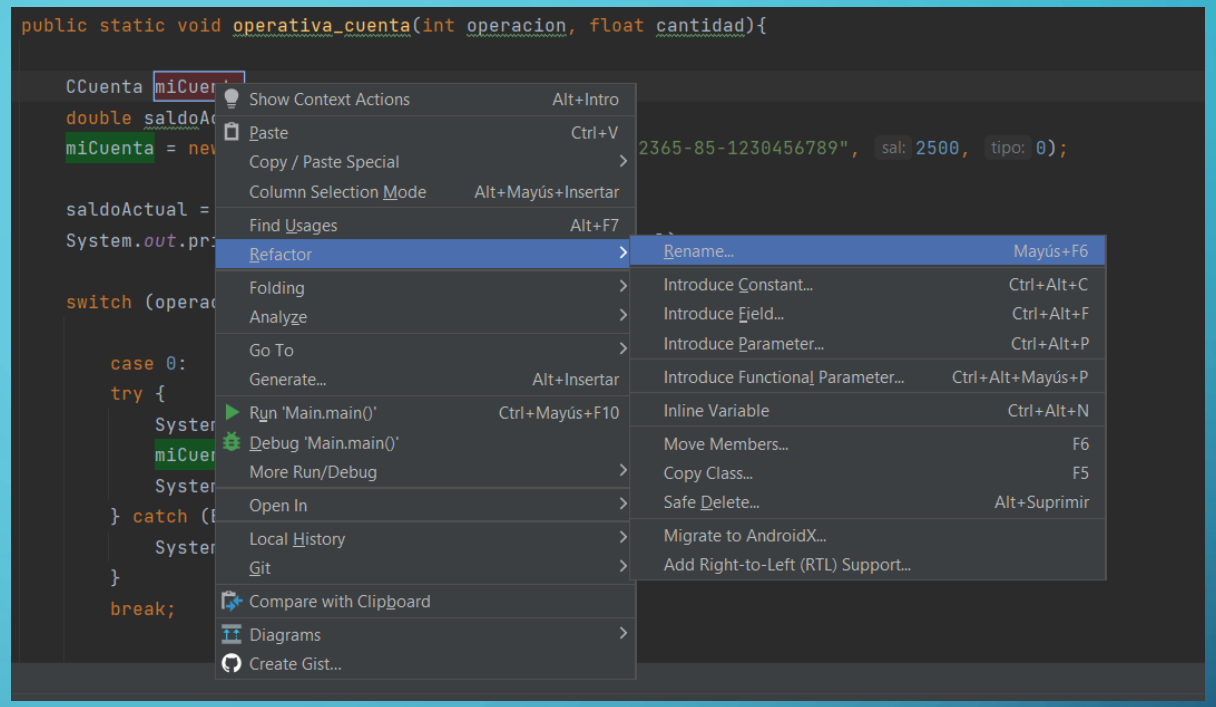
- Se han seguido una serie de pasos para refactorizar el código de la aplicación por defecto que venia incluida en el trabajo de la UT4, lo primero que se ha hecho es cambiar el nombre del paquete en el que viene el main y la clase Ccuenta

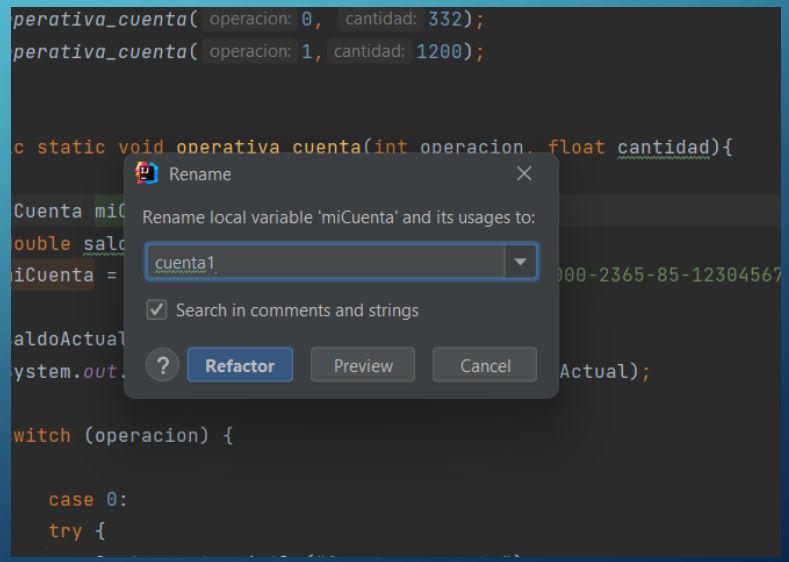


- En la imagen anterior nos dirigimos a la izquierda del entorno y buscamos entre las carpetas del proyecto el paquete contenedor del main y el Ccuenta hacemos click derecho en el paquete y seleccionamos refactor nos saldrá otro menú emergente y esta vez damos click en rename…
- Escribimos el nuevo nombre del paquete clickeamos en refactor y ya estaría refactorizado el nombre del paquete en el proyecto entero



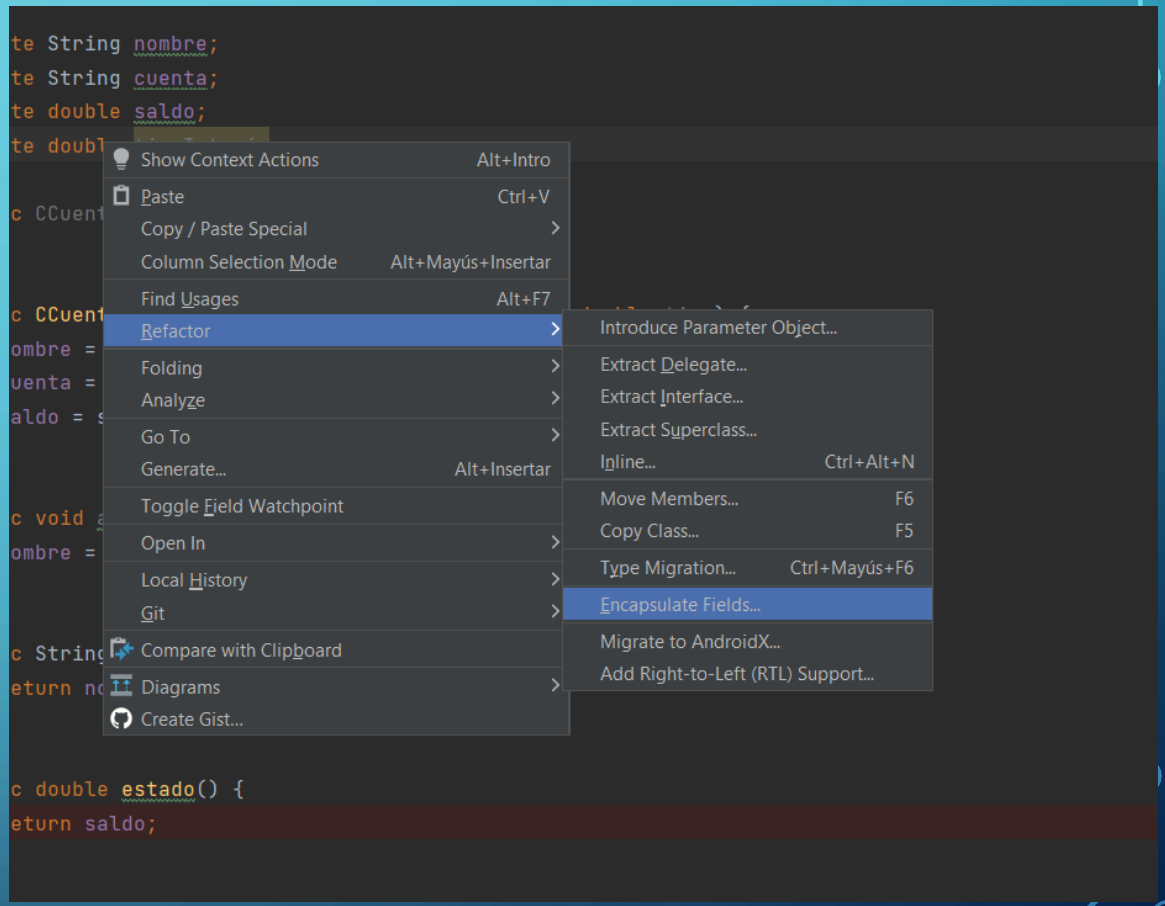
- Ahora vamos refactorizar el nombre de una variable del main y utilizaremos el mismo método mostrado en el punto anterior click derecho en la variable que queremos renombrar, click en refactor, click en rename, añadimos el nombre nuevo y click de nuevo en refactor





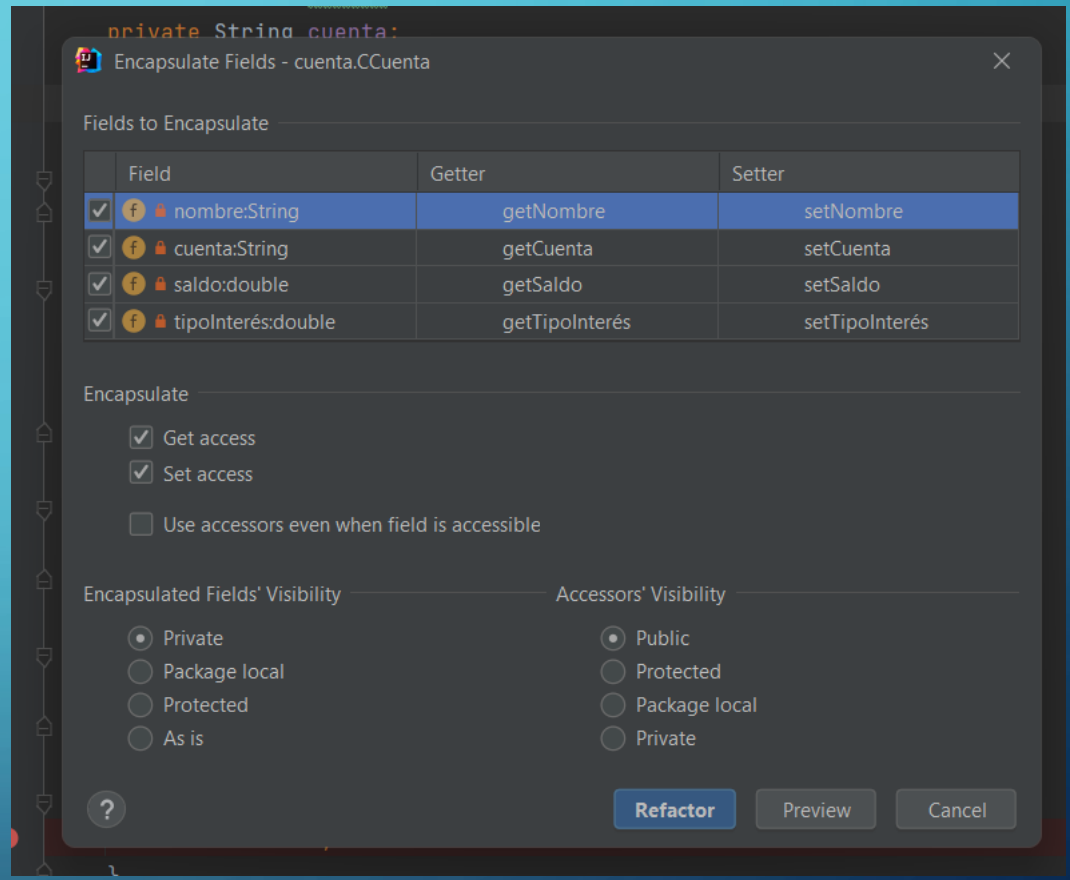
- Los siguientes dos pasos de refactorización al no usar la opción refactor no se han realizado capturas ya que se ha hecho de manera manual en el mismo main (estos cambios se podrán ver directamente en el repositorio master creado en mi cuenta github), los cambios han sido la creación de un método llamado operativa_cuenta, que englobe las sentencias de la clase main que operan con el objeto cuenta1.
- A este método se le parametrizaran dos variables una de tipo entero llamada operación y otra de tipo float llamada float, este método se utilizara para comprobar tanto el ingreso a una cuenta como retirada de dinero.

- El ultimo punto de refactorización es el encapsulamiento de los atributos de la cuenta CCuenta, se vuelto a usar la opción refactor, nos dirigimos a donde están declaramos los parámetros damos click derecho, refactor, y click en este, saldrá un submenú y escogeremos la opción encapsulate fields.



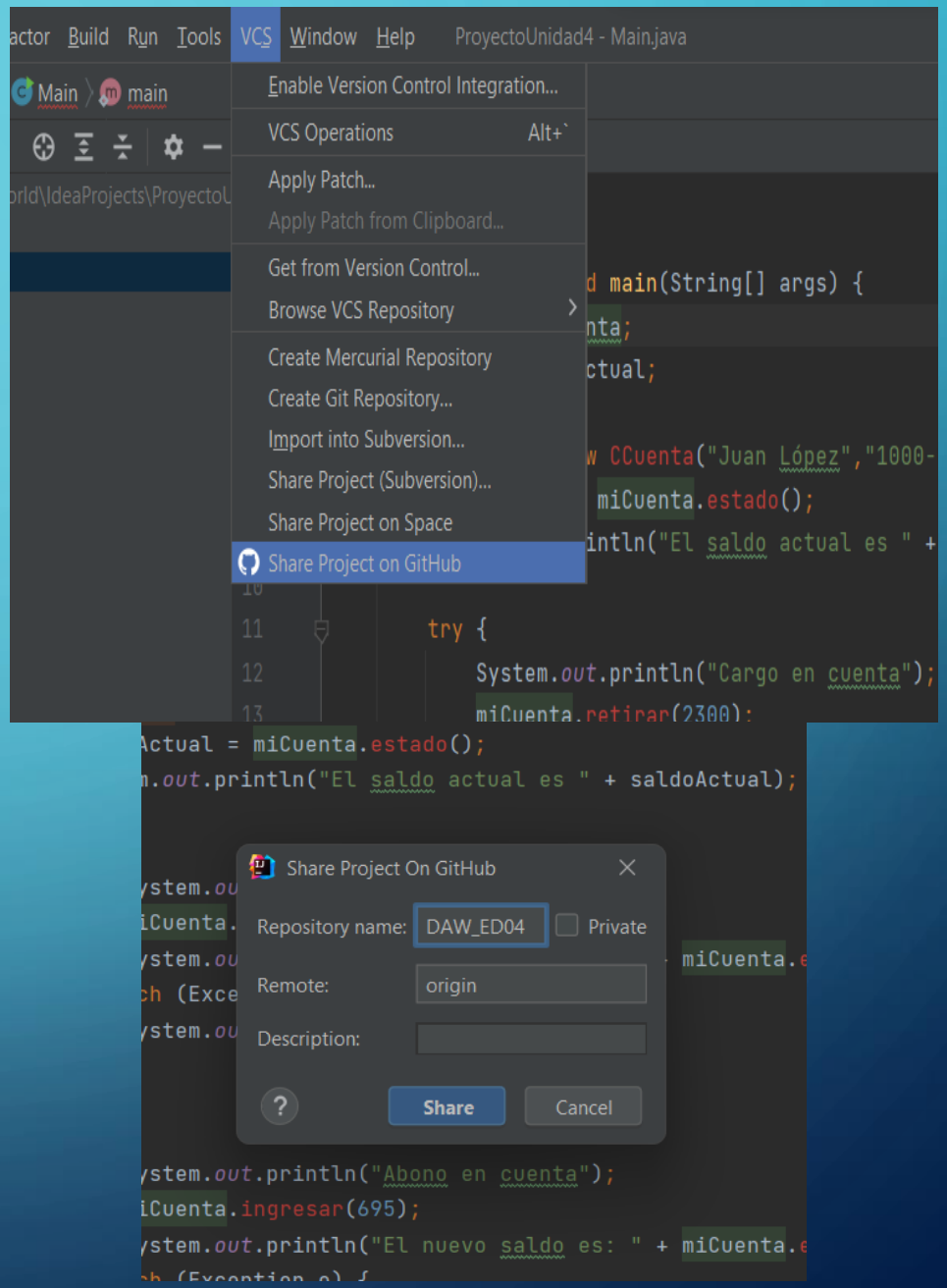
Se nos abrirá un menú en el cual marcaremos todos los atributo de la clase y en el apartado encapsulate marcaremos get y set pero desmarcaremos Use accesors even when fields is accessible ya que este nos modificara el constructor parametrizado que tenemos ya creado, una vez encapsulado los atributos borraremos los siguientes métodos ya que seria redundante tenerlos.

```
public void asignarNombre(String nom) {
    nombre = nom;
}

public String obtenerNombre() {
    return nombre;
}

public double estado() {
    return saldo;
}

public String obtenerCuenta() {
    return cuenta;
}
```

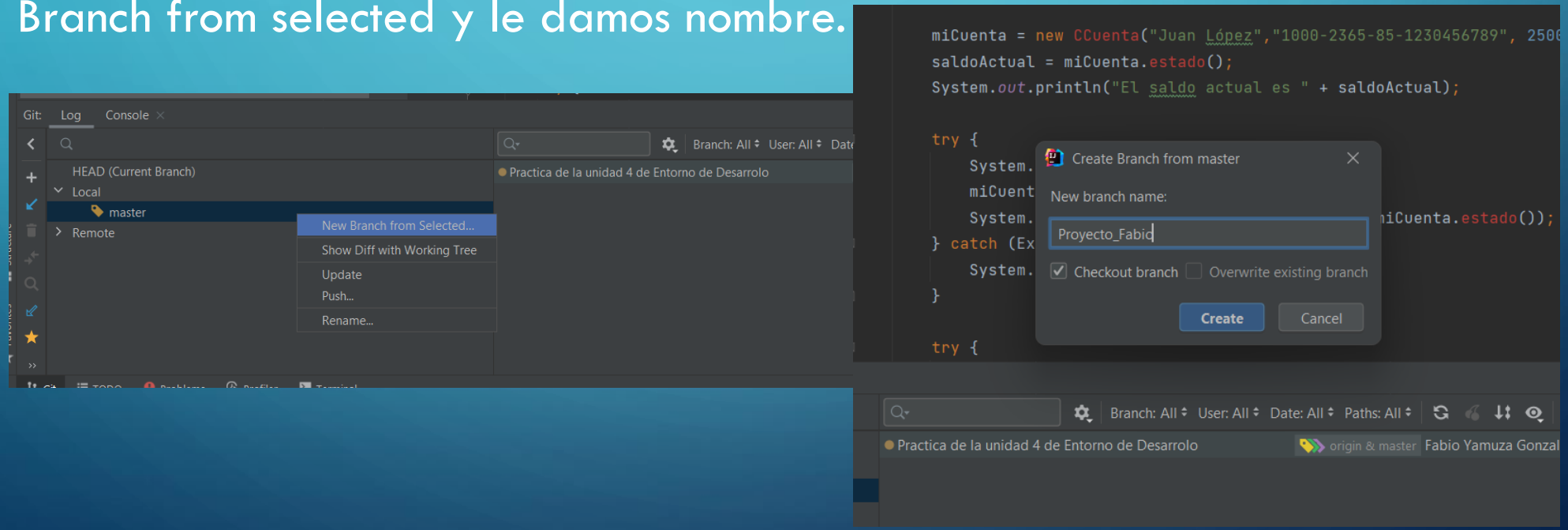


GIT

- El primer paso es crear un repositorio con este proyecto en git, escogeremos entre los menus superiores el que pone VCS y escogeremos share Project on GitHub, nos saltara una venta en la cual tendrmeos que darle nombre al repositorio y daremos click en share.

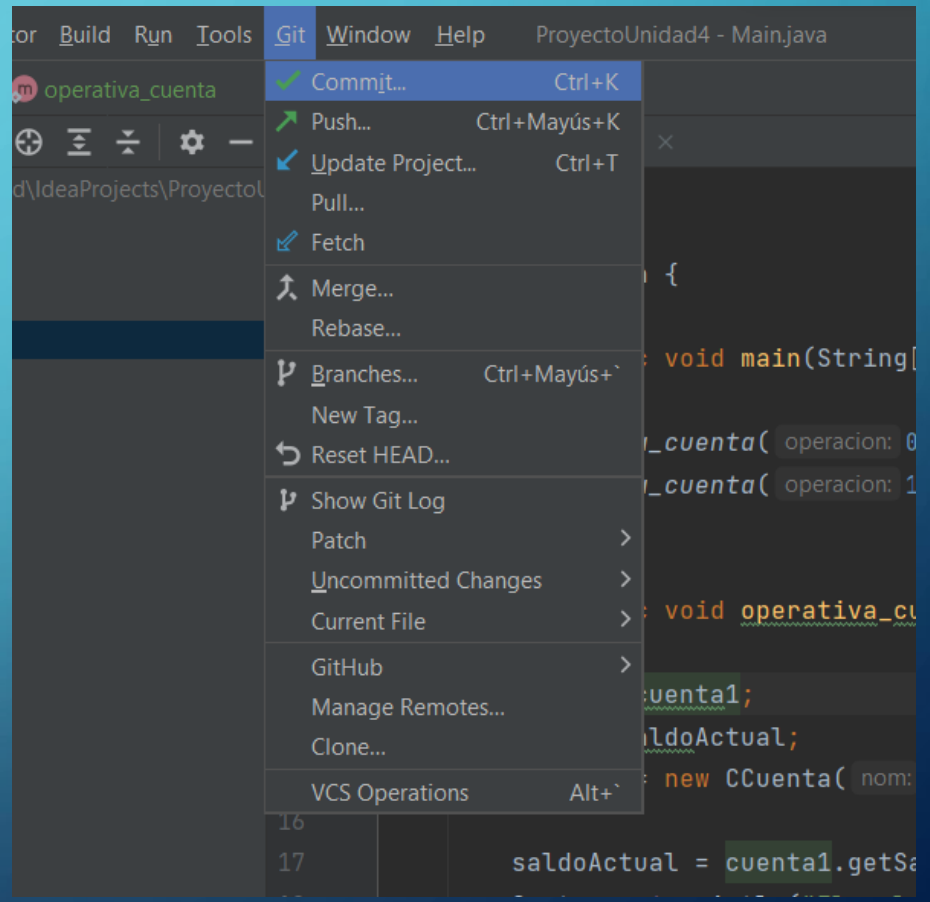


- Ahora vamos a crear una rama secudaria, en esta rama es donde vamos a realizar todos los cambios que se han explicado en el primera apartado del proyecto en el de refactorización, nos dirigimos a la esquina inferior derecha y clickeamos en git, damos click derecho en la rama master y escogemos new Branch from selected y le damos nombre.

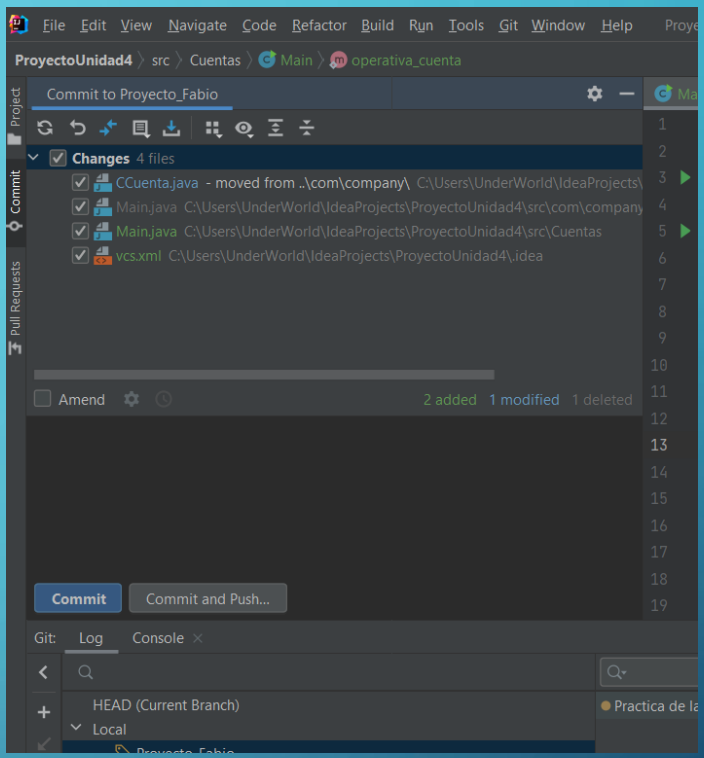


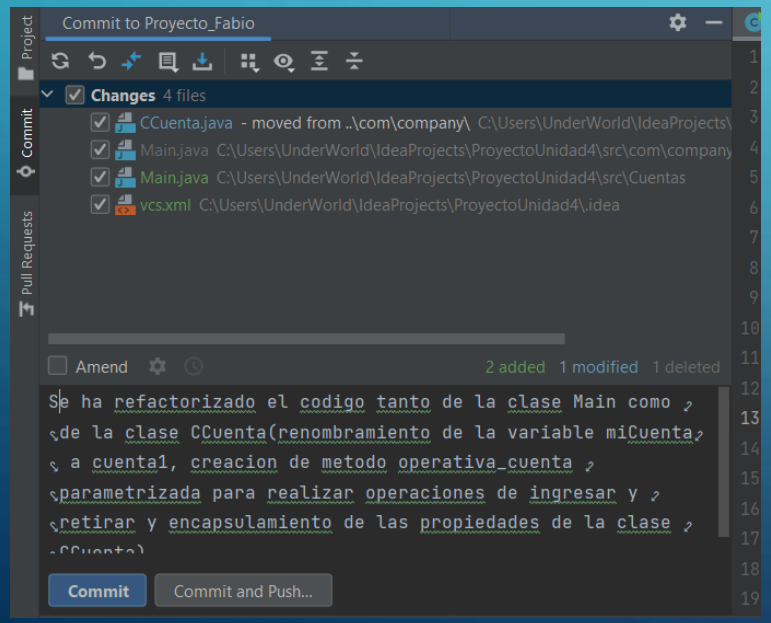
COMMIT

- En este apartado vamos a hacer un commit de los cambios de refactorización en nuestra rama secundaria danto un breve resumen comentado de los cambios realizados, click en la opción git del entorno y escogemos el que nos pone commit.



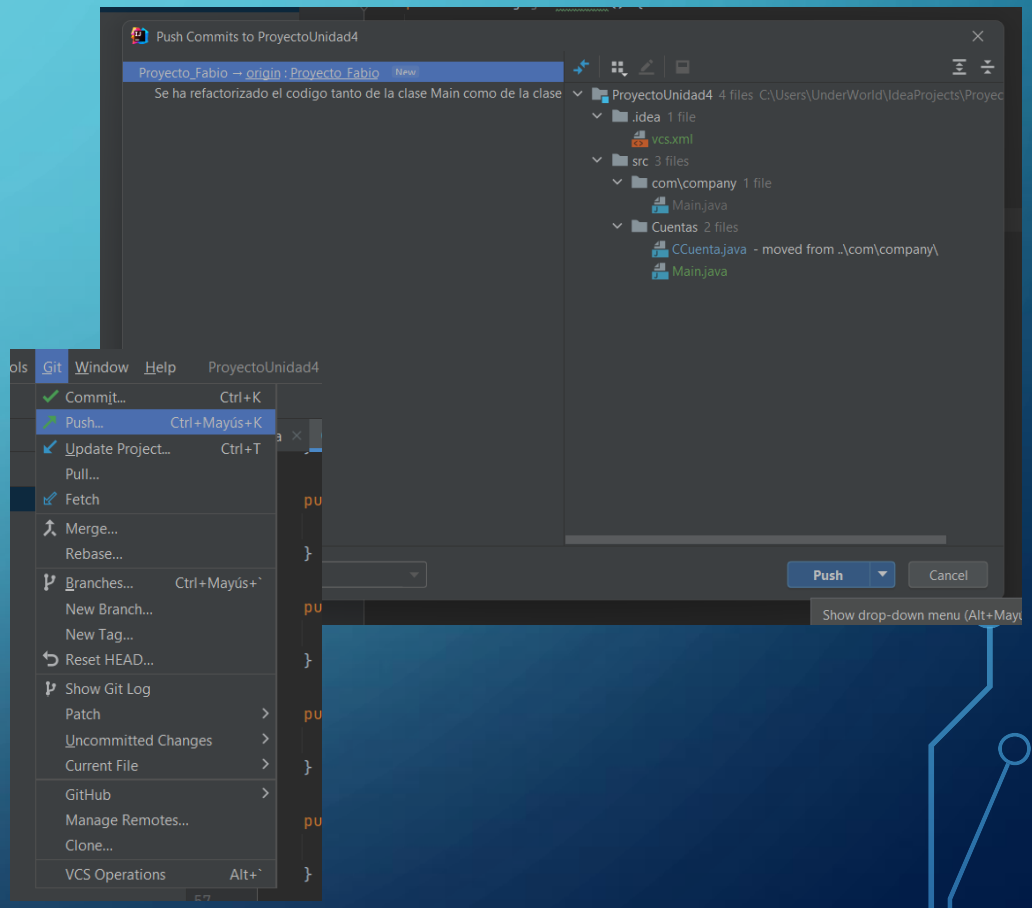
- Se seleccionan los archivos del proyecto de los que se quieren hacer el commit y en el cuadro de texto de abajo es donde se debe escribir la documentación de los cambios.



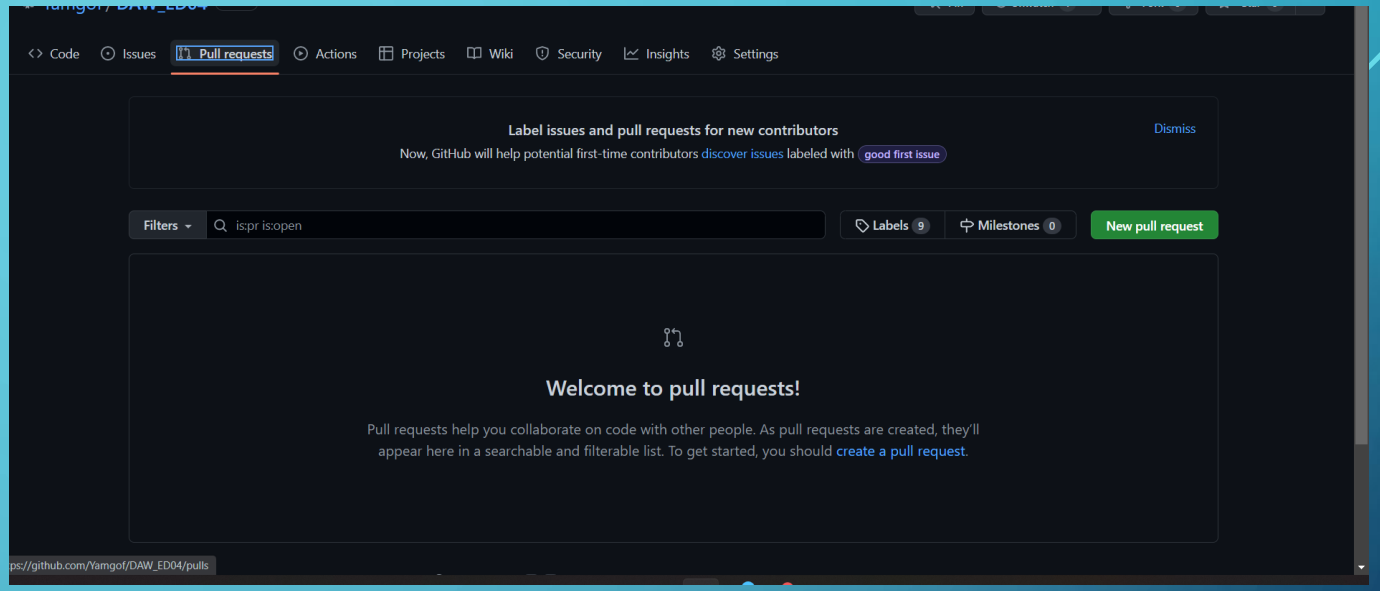


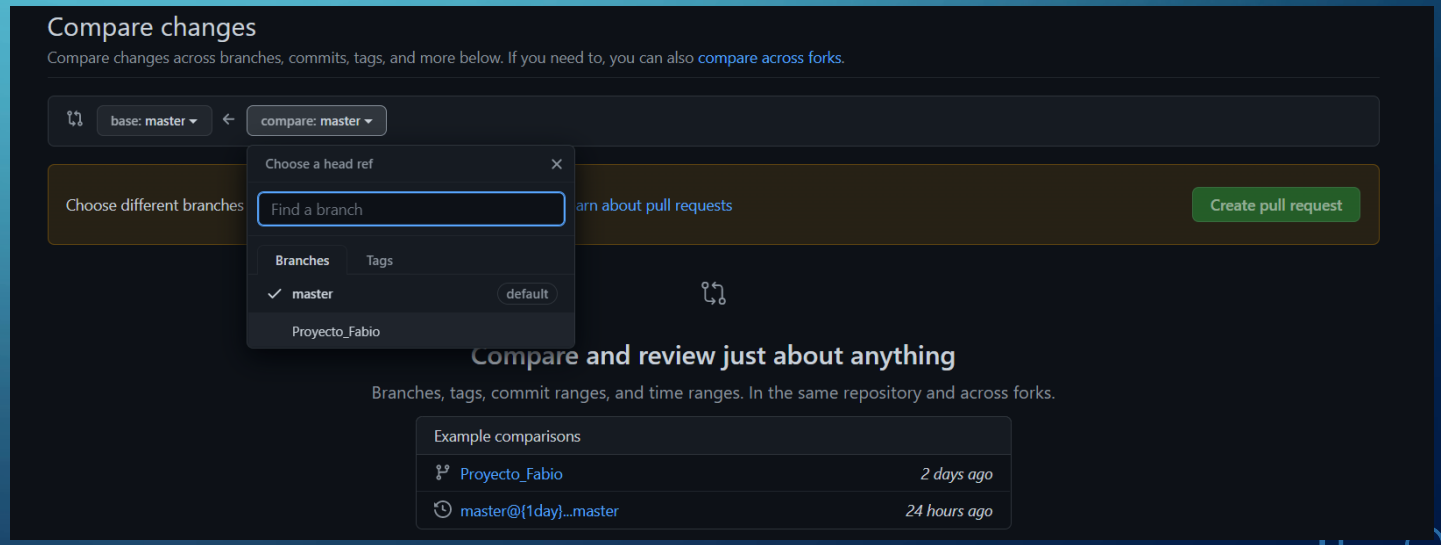
DE RAMA SECUNDARIA A PRINCIPAL

Ahora justo debajo de donde hicimos el commit tenemos la opción de push con la cual subiremos nuestros cambios de la rama secundaria a nuestra rama secundaria que tenemos en remoto.
El siguente paso si será hacer un pull request en el cual subiremos nuestros cambios de la rama secundaria a la rama principal pero esta vez si haremos esta operación desde el mismo github para manejar directamente nuestras ramas que están en remoto

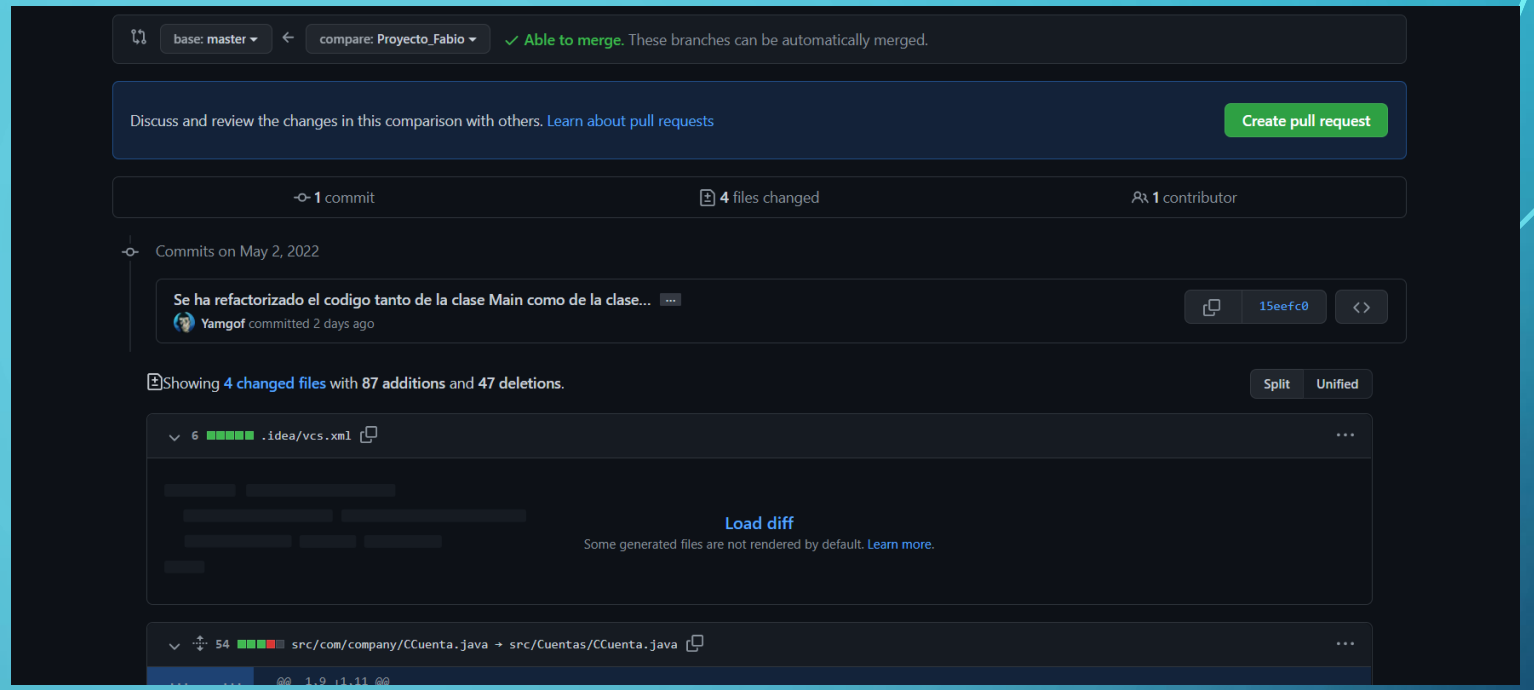


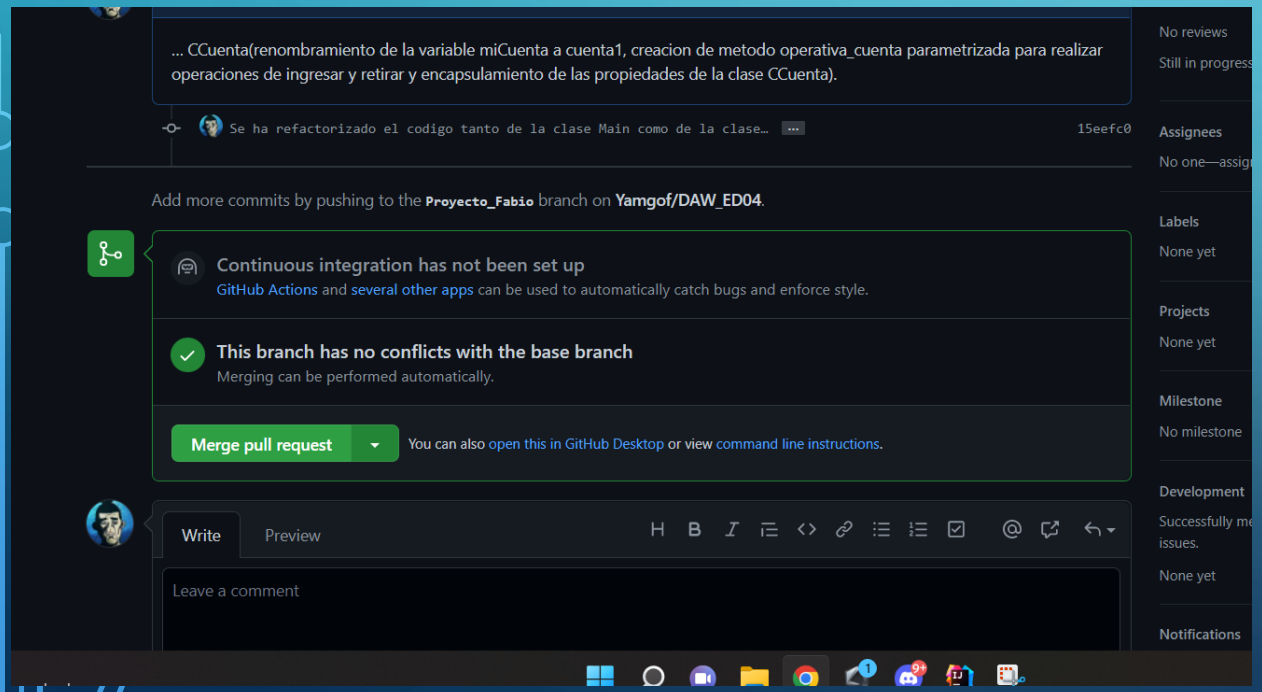
- Dentro de la pagina de github abrimos nuestro repositorio y escogemos la acción pull request, una vez seleccionada tendremos que especificar de que rama a que rama haremos el pull, y se ha cogido la rama secundaria remota que aplique cambios a la rama master.





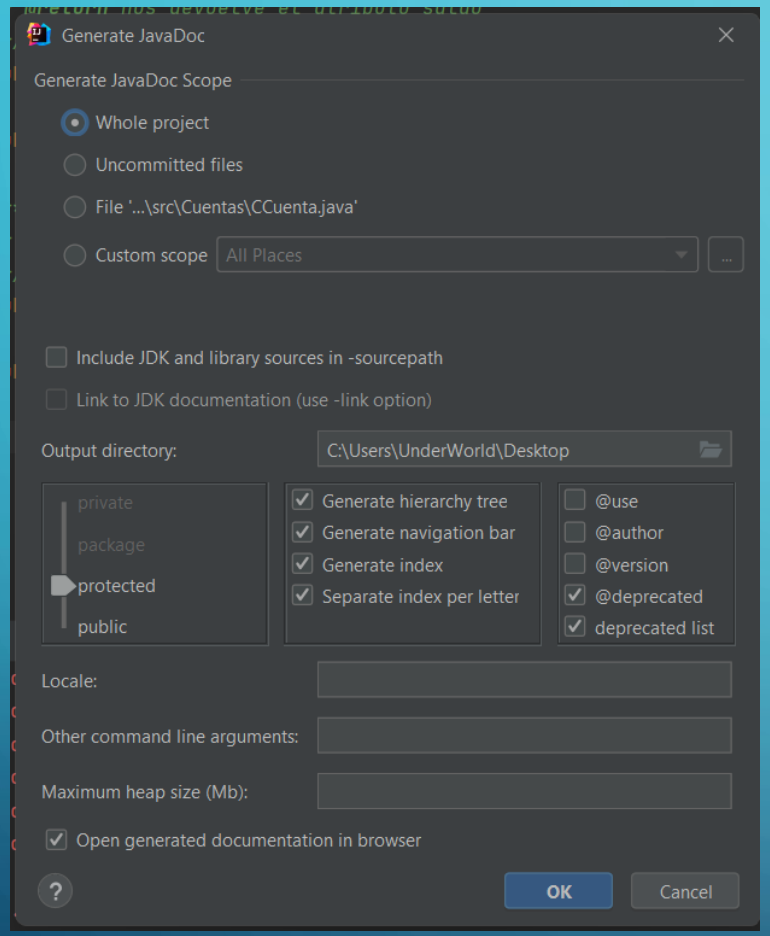
- Una vez escogido las ramas le daremos a create pull request, veremos el comentario del commit para saber los cambios que vamos a hacer pull una vez hecho esto solo tendremos que mergear los cambios.

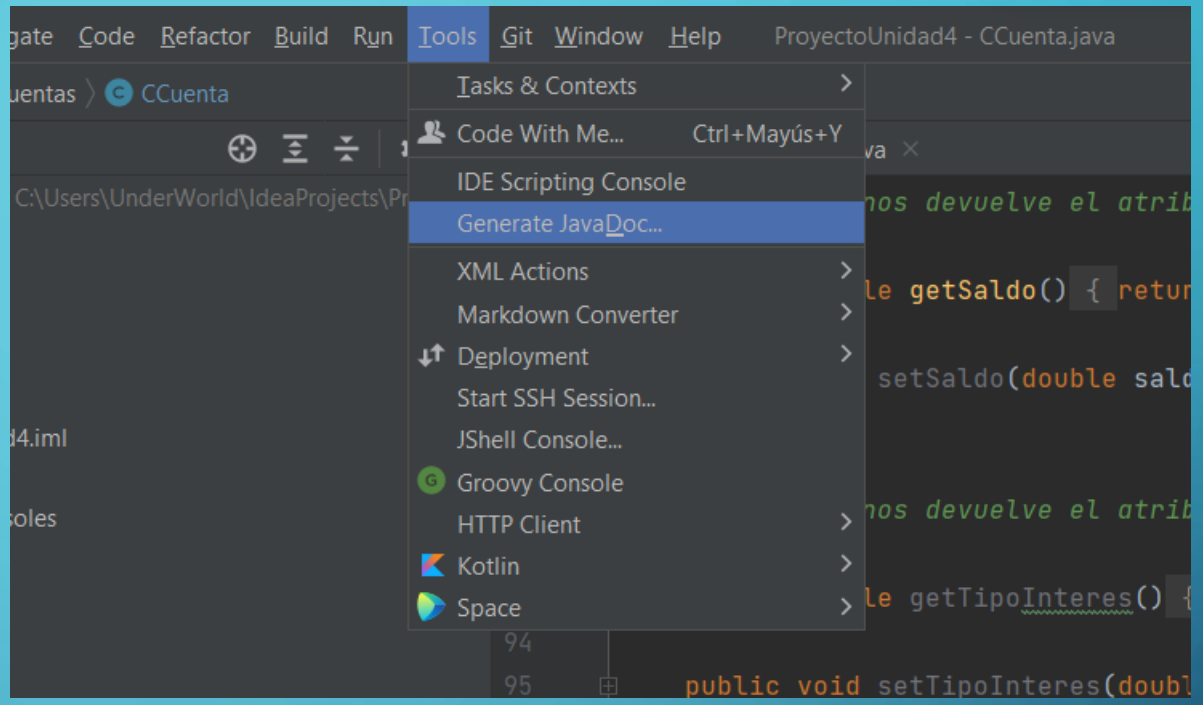


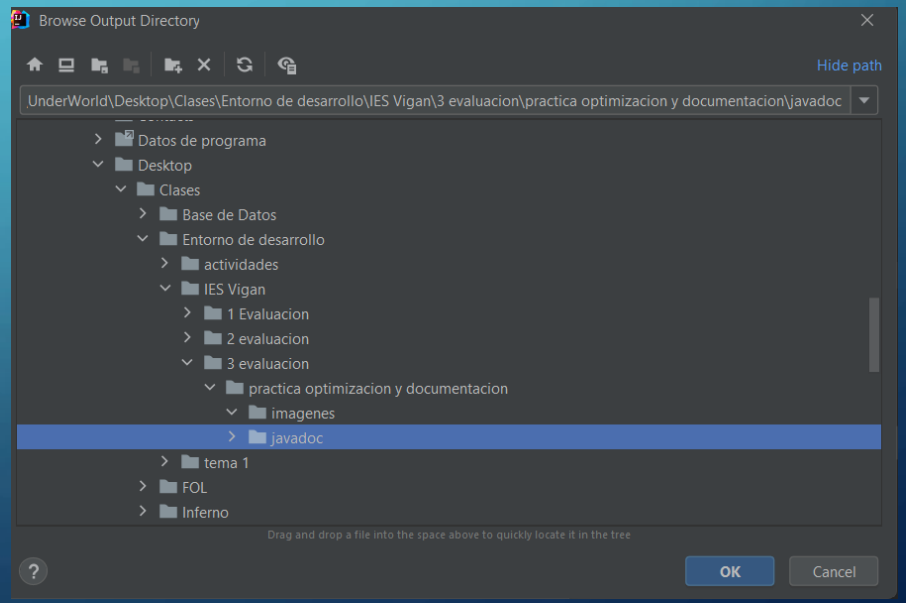


JAVADOC

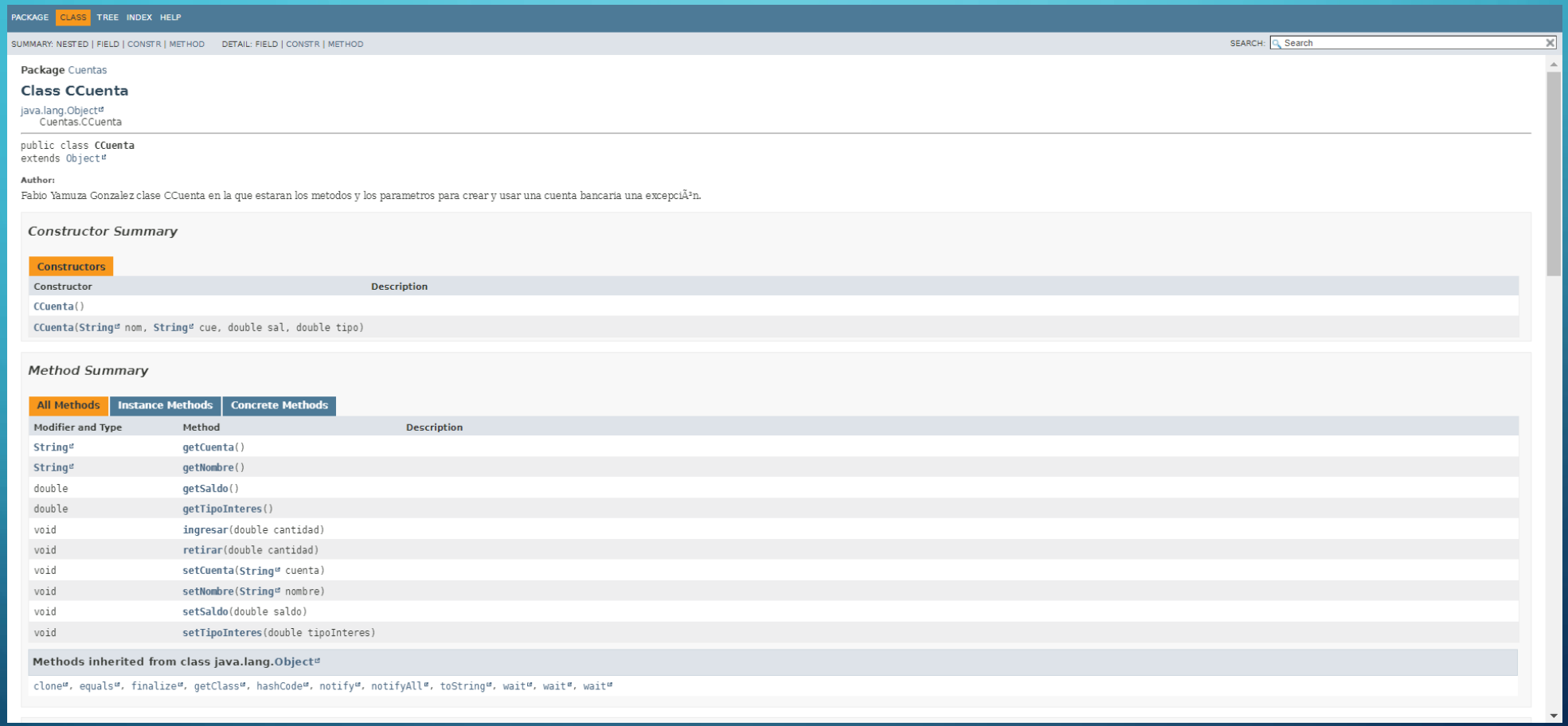
- Ahora vamos a generar un archivo javadoc, pero antes vamos a agregar una serie de comentarios en la que vamos a definir el nombre del autor y explicaremos la función de cada parámetro que le pasamos a los métodos y en que influye y explicaremos que va a devolver los métodos que nos esten devolviendo algo. Cuando tengamos esto nos dirigimos a tools y escogeremos generate javadoc, nos saldrá una ventana emergente en la que tendremos que poner que queremos que muestre el javadoc y donde queremos guardarlo en la siguiente diapositiva estarán las imágenes del proceso.


Generate JavaDoc
Generate JavaDoc Scope
Whole project
Uncommitted files
File '...\src\Cuentas\CCuenta.java'
Custom scope
All Places
Include JDK and library sources in -sourcepath
Link to JDK documentation (use -link option)
Output directory:
C:\Users\UnderWorld\Desktop
private
package
protected
public
Generate hierarchy tree
Generate navigation bar
Generate index
Separate index per letter
@use
@author
@version
@deprecated
deprecated list
Locale:
Other command line arguments:
Maximum heap size (Mb):
Open generated documentation in browser
OK
Cancel


Code
Refactor
Build
Run
Tools
Git
Window
Help
ProyectoUnidad4 - CCuenta.java
CCuenta
Tasks & Contexts
Code With Me... Ctrl+Mayús+Y
IDE Scripting Console
Generate JavaDoc...
XML Actions
Markdown Converter
Deployment
Start SSH Session...
JShell Console...
Groovy Console
HTTP Client
Kotlin
Space
C:\Users\UnderWorld\IdeaProjects\Pr
d4.iml
oles
94
95
public void setTipoInteres(doubl


Browse Output Directory
Hide path
UnderWorld\Desktop\Clases\Entorno de desarrollo\IES Vigan\3 evaluacion\practica optimizacion y documentacion\javadoc
Datos de programa
Desktop
Clases
Base de Datos
Entorno de desarrollo
actividades
IES Vigan
1 Evaluacion
2 evaluacion
3 evaluacion
practica optimizacion y documentacion
imagenes
javadoc
tema 1
FOL
Inferno
Drag and drop a file into the space above to quickly locate it in the tree
OK
Cancel

CAPTURAS DEL JAVADOC



PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package Cuentas

Class CCuenta

java.lang.Object
  Cuentas.CCuenta

public class **CCuenta**
extends Object

Author:
Fabio Yamuza Gonzalez clase CCuenta en la que estaran los metodos y los parametros para crear y usar una cuenta bancaria una excepciÃ³n.

Constructor Summary

Constructors

Constructor	Description
CCuenta()	
CCuenta(String nom, String cue, double sal, double tipo)	

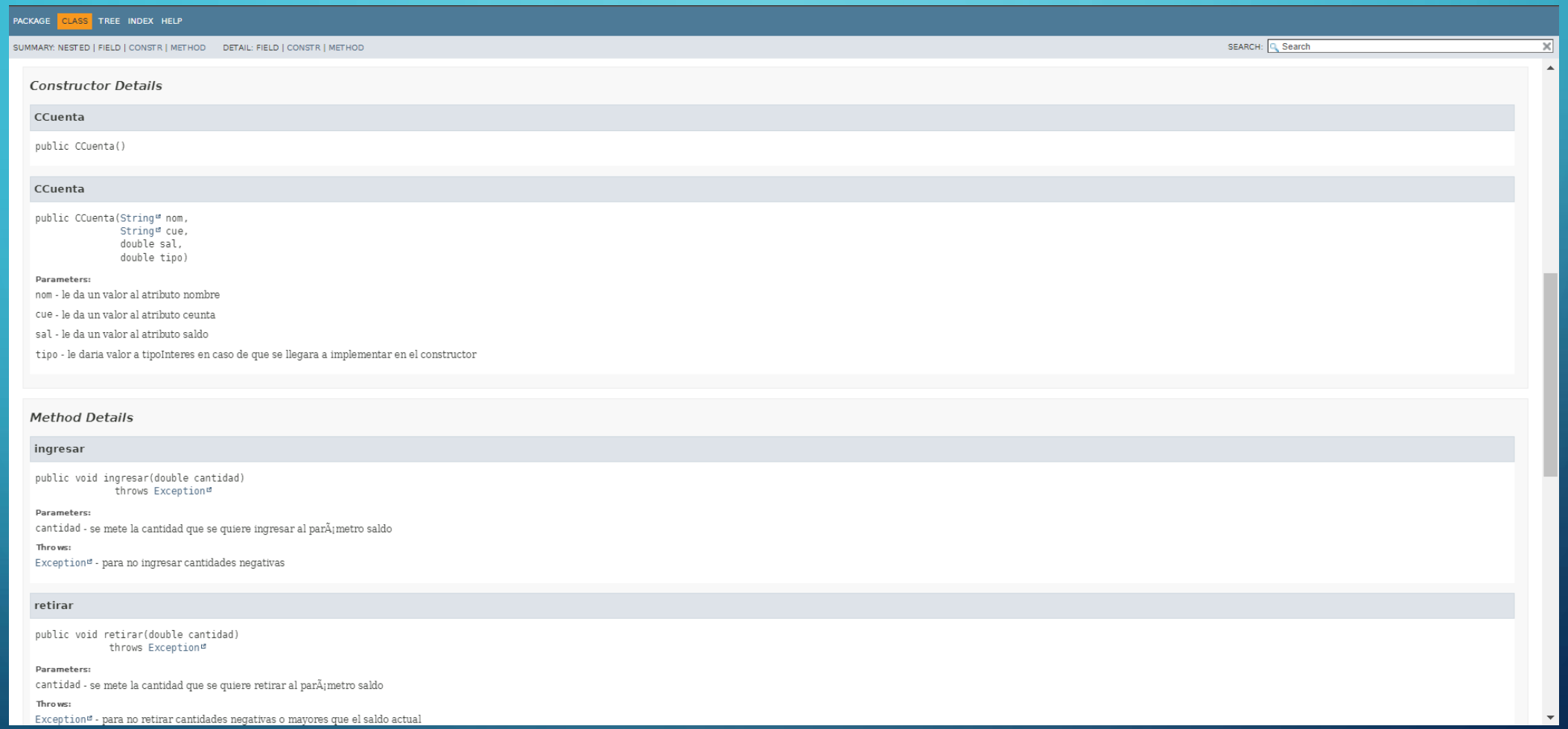
Method Summary

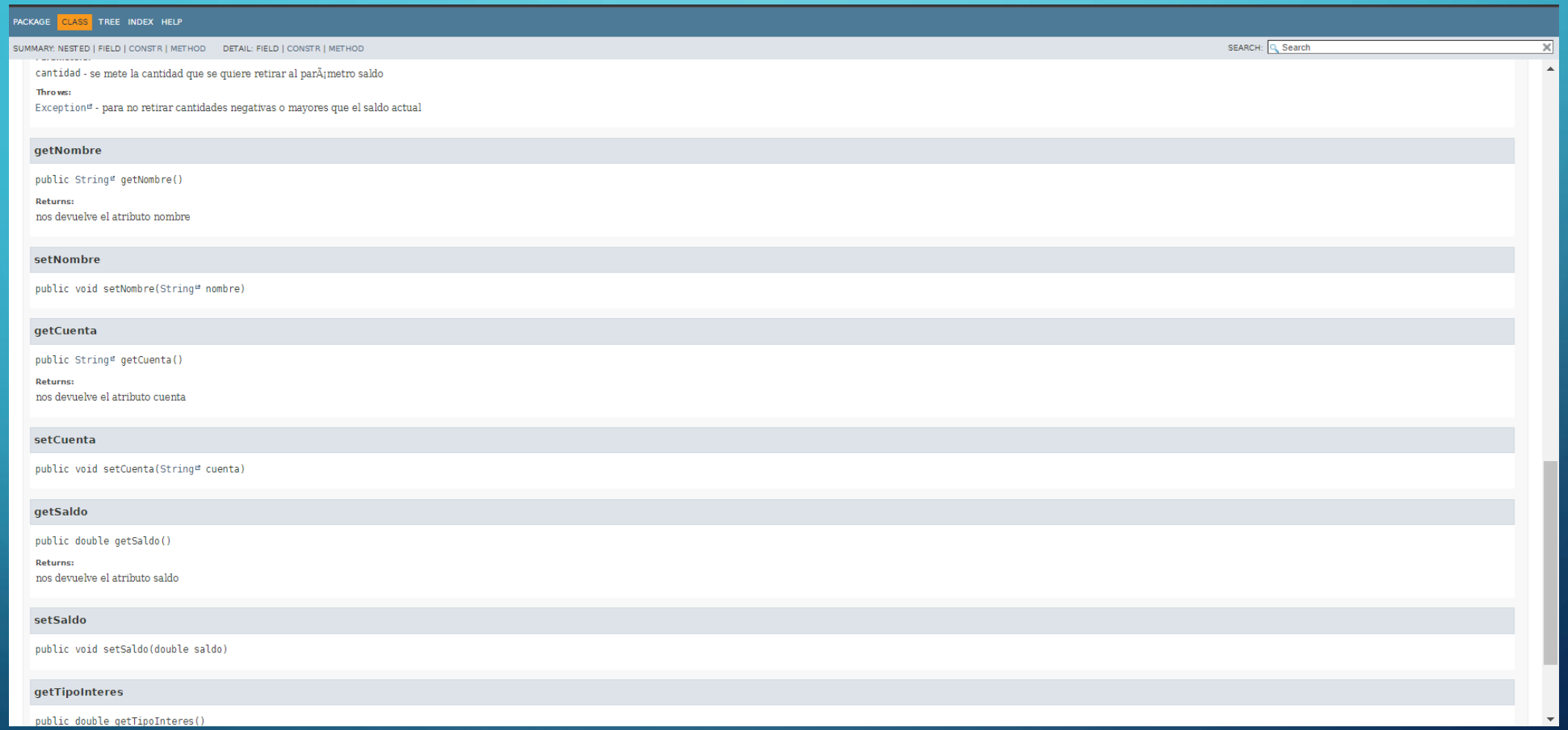
All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
String	getCuenta()	
String	getNombre()	
double	getSaldo()	
double	getTipoInteres()	
void	ingresar(double cantidad)	
void	retirar(double cantidad)	
void	setCuenta(String cuenta)	
void	setNombre(String nombre)	
void	setSaldo(double saldo)	
void	setTipoInteres(double tipoInteres)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait


PACKAGE CLASS TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
SEARCH:
Search
Constructor Details
CCuenta
public CCuenta()
CCuenta
public CCuenta(String nom,
String cue,
double sal,
double tipo)
Parameters:
nom - le da un valor al atributo nombre
cue - le da un valor al atributo ceunta
sal - le da un valor al atributo saldo
tipo - le daria valor a tipoInteres en caso de que se llegara a implementar en el constructor
Method Details
ingresar
public void ingresar(double cantidad)
throws Exception
Parameters:
cantidad - se mete la cantidad que se quiere ingresar al parÃ¡metro saldo
Throws:
Exception - para no ingresar cantidades negativas
retirar
public void retirar(double cantidad)
throws Exception
Parameters:
cantidad - se mete la cantidad que se quiere retirar al parÃ¡metro saldo
Throws:
Exception - para no retirar cantidades negativas o mayores que el saldo actual


PACKAGE CLASS TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH: Search
cantidad - se mete la cantidad que se quiere retirar al parÃ¡metro saldo
Throws:
Exception - para no retirar cantidades negativas o mayores que el saldo actual
getNombre
public String getNombre()
Returns:
nos devuelve el atributo nombre
setNombre
public void setNombre(String nombre)
getCuenta
public String getCuenta()
Returns:
nos devuelve el atributo cuenta
setCuenta
public void setCuenta(String cuenta)
getSaldo
public double getSaldo()
Returns:
nos devuelve el atributo saldo
setSaldo
public void setSaldo(double saldo)
getTipoInteres
public double getTipoInteres()