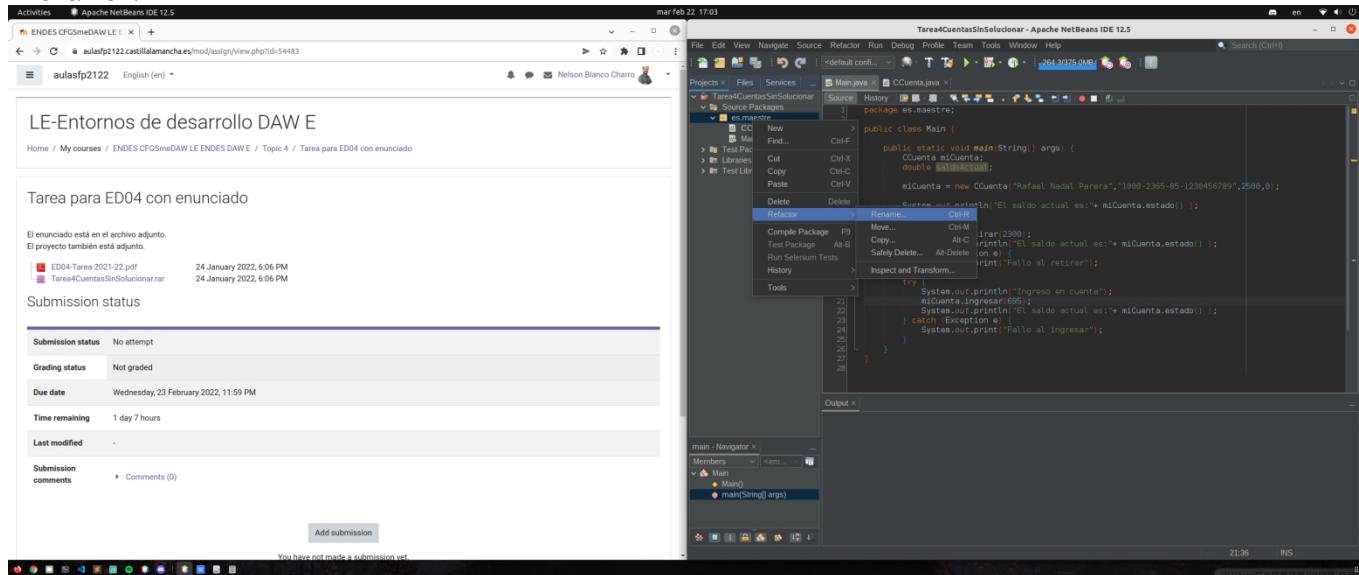


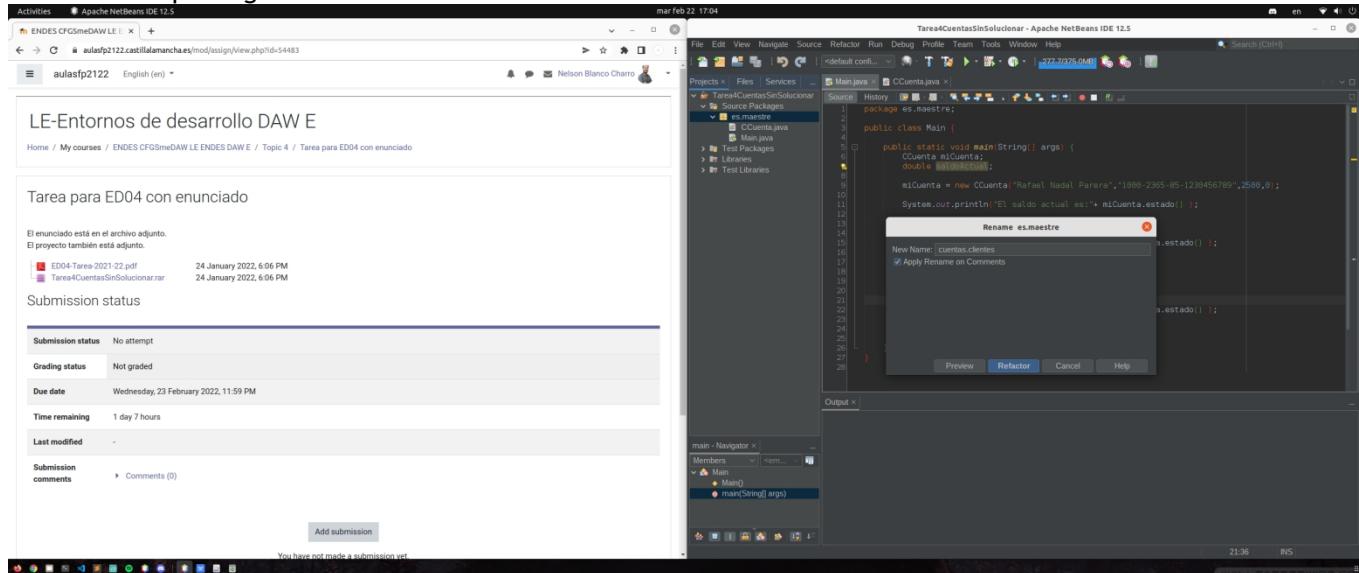
Refactorización

1. Las clases deberán formar parte del paquete cuentas.clientes en vez del actual es.maestre. (refactorizando el nombre del paquete)

Para ello pulsamos botón derecho sobre el paquete y seleccionamos la opción “Refactor” y a continuación “Rename”.

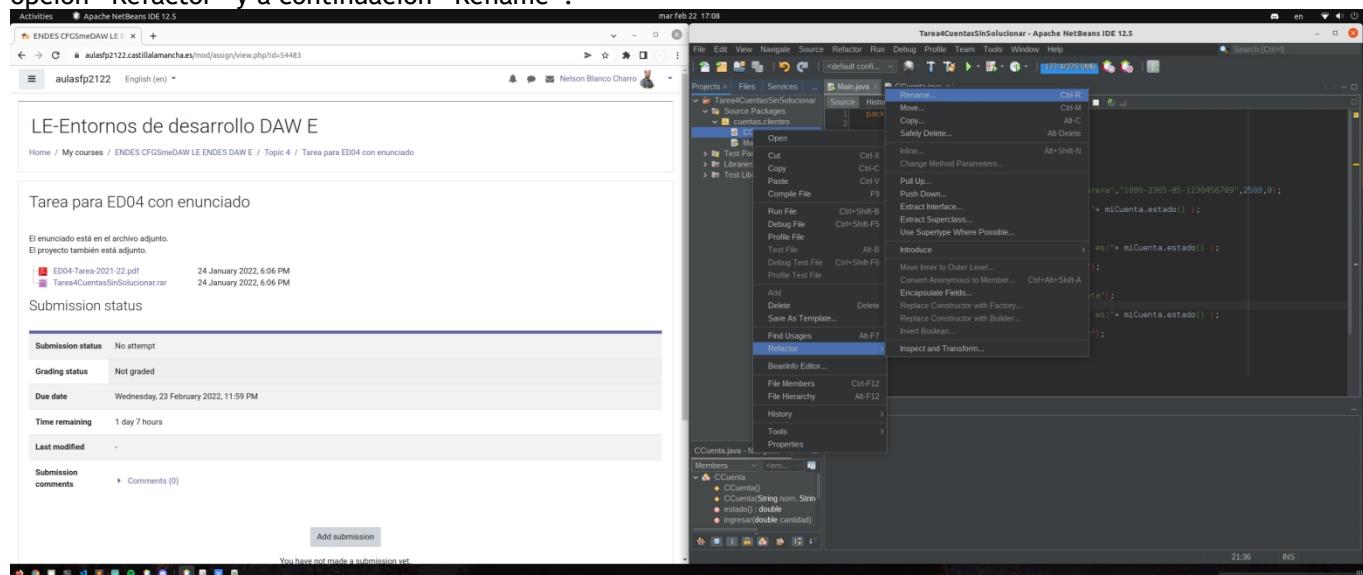


Nos aparecerá una ventana para introducir en nombre nuevo y podemos aplicar también el cambio a los comentarios que hagan referencia al nombre.

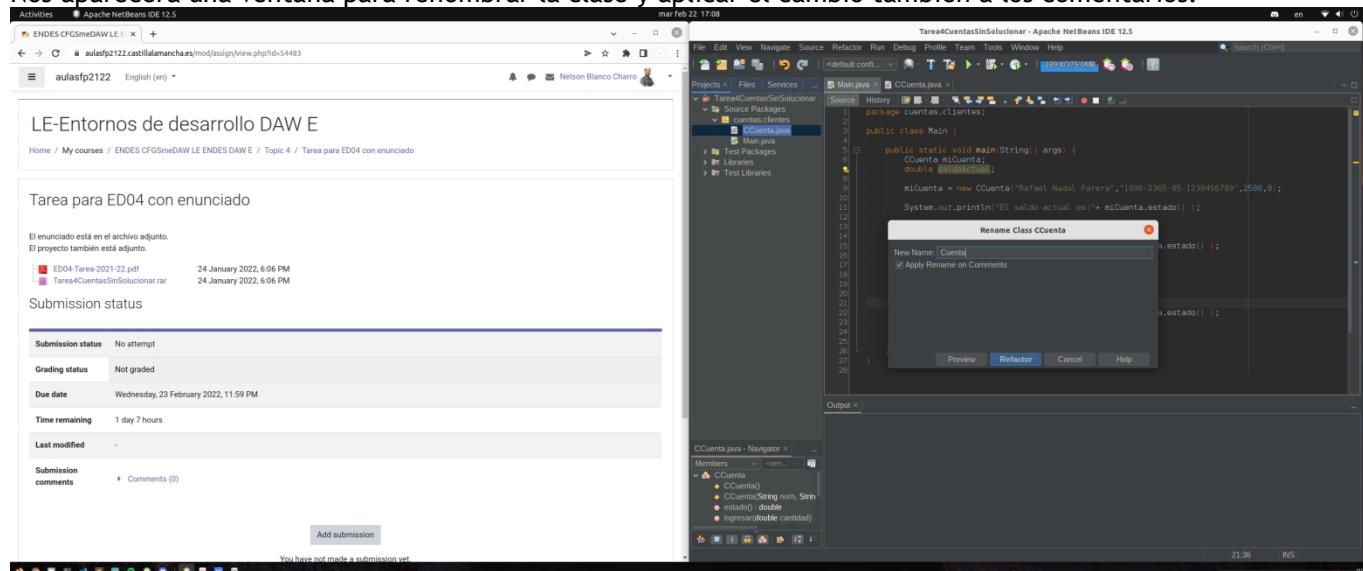


2. Renombrar la clase CCuenta por Cuenta. Cambiar el nombre de la variable "miCuenta" por "cuenta1". (refactorizando el nombre de la clase y de la variable)

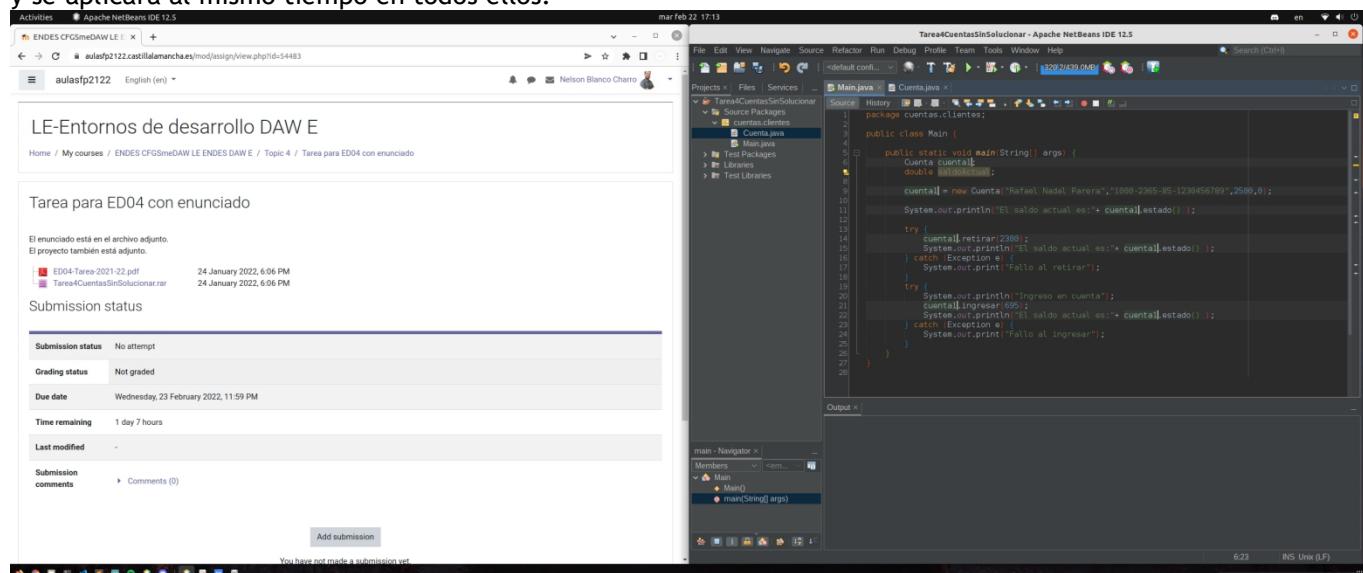
Del mismo modo que antes, podemos renombrar la clase pulsando botón derecho en la clase y seleccionando la opción “Refactor” y a continuación “Rename”.



Nos aparecerá una ventana para renombrar la clase y aplicar el cambio también a los comentarios.

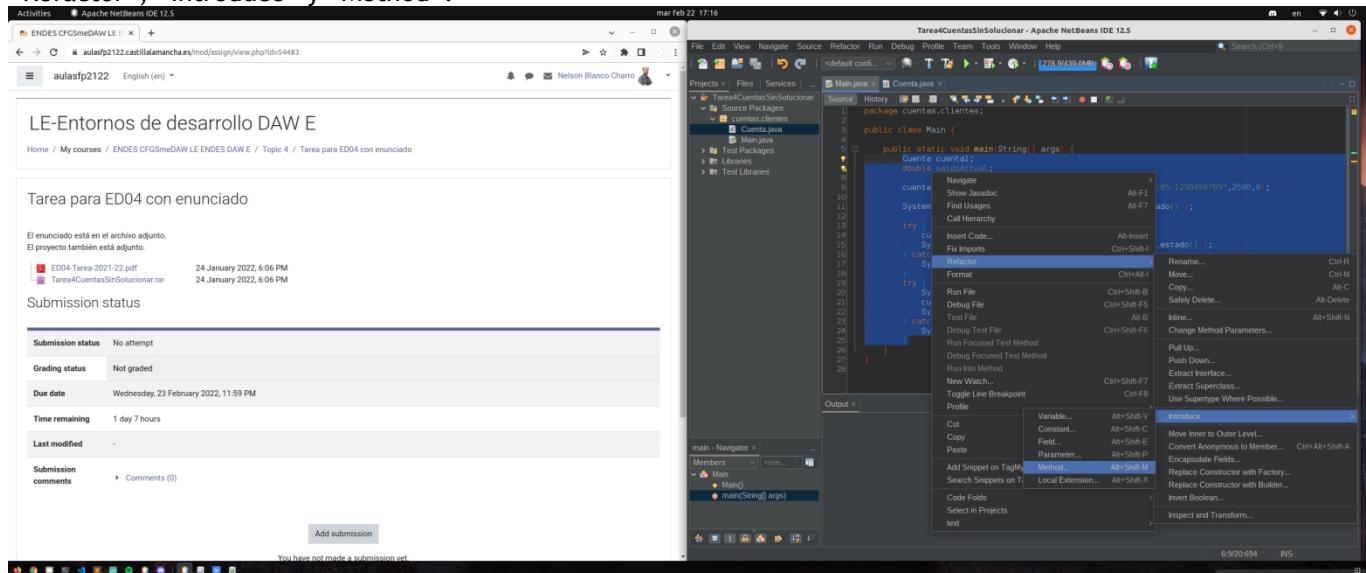


Para renombrar la variable podemos ir cambiando el nombre uno a uno, o podemos seleccionar el nombre de la variable y pulsar Control+J para ir seleccionando todos los nombres, a continuación, reescribimos el nuevo nombre y se aplicará al mismo tiempo en todos ellos.

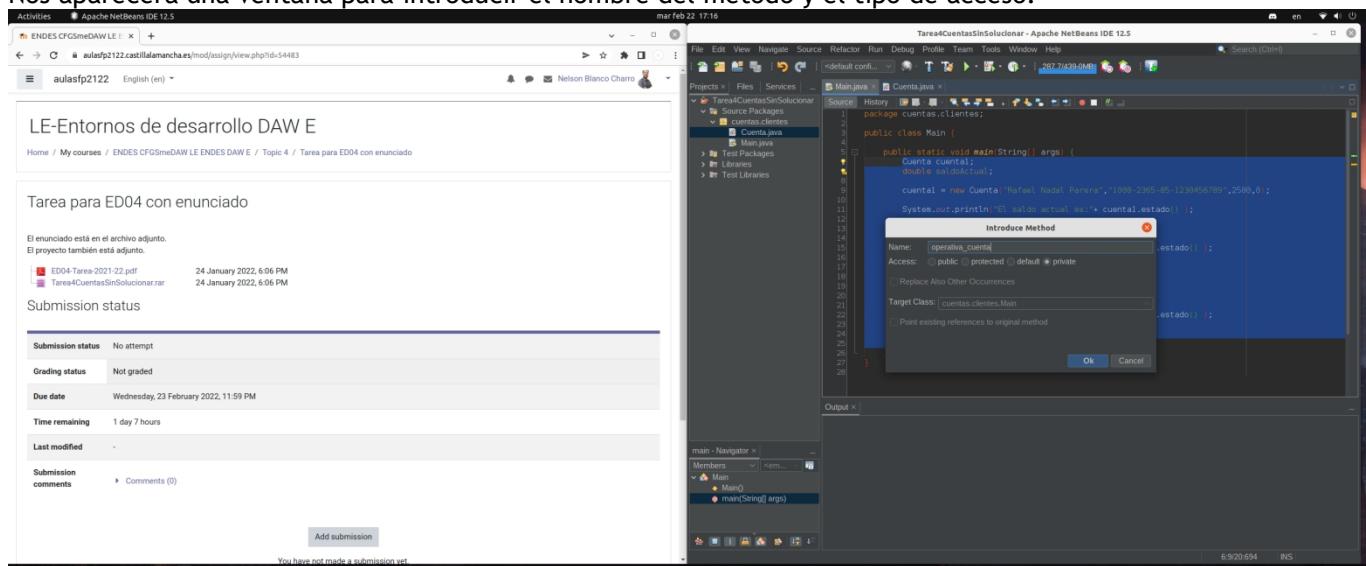


3. Introducir el método, operativa_cuenta, que englobe todas las sentencias de la clase Main. (seleccionando todo el código de dentro del método main y en Refactorizar elegir la opción Introduce-method)

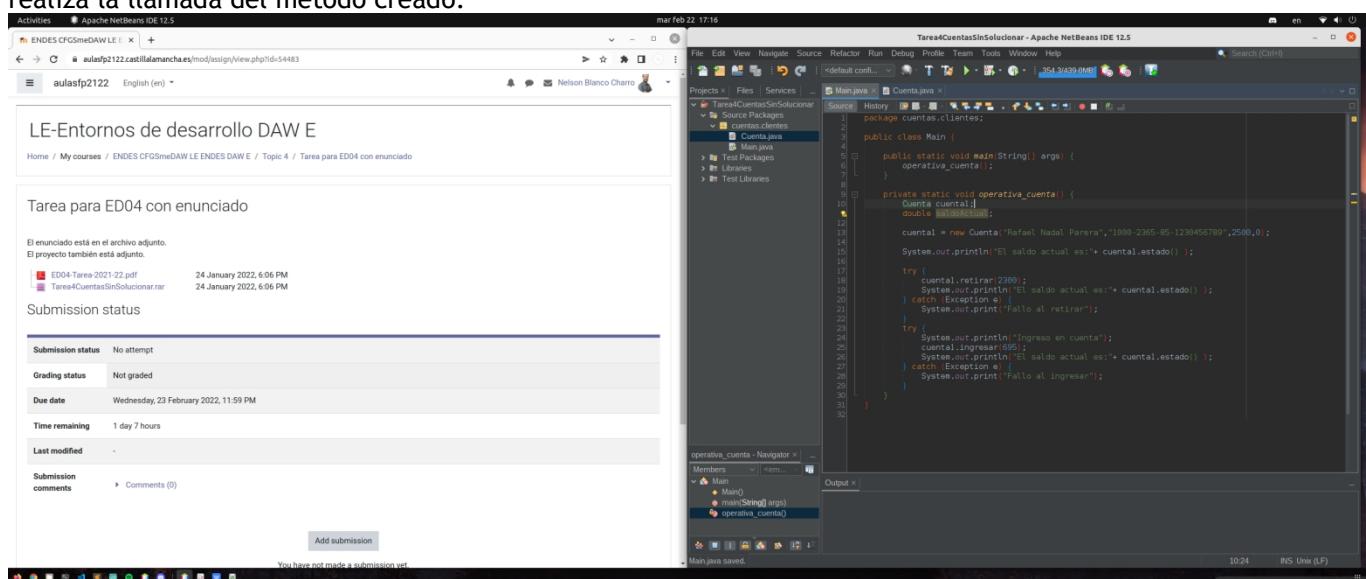
Para hacerlo seleccionamos todo el código del método Main, pulsamos botón derecho y seleccionamos las opciones “Refactor”, “Introduce” y “Method”.



Nos aparecerá una ventana para introducir el nombre del método y el tipo de acceso.

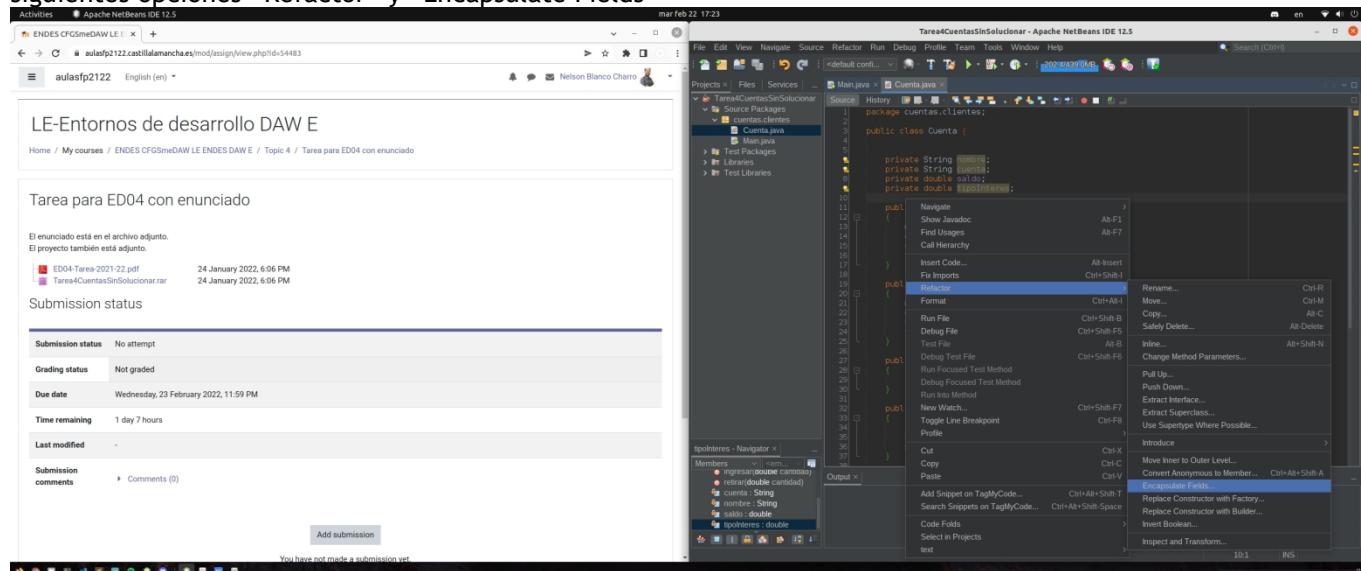


Finalmente, se habrá creado un nuevo método con el código seleccionado y en el lugar donde estaba el código se realiza la llamada del método creado.

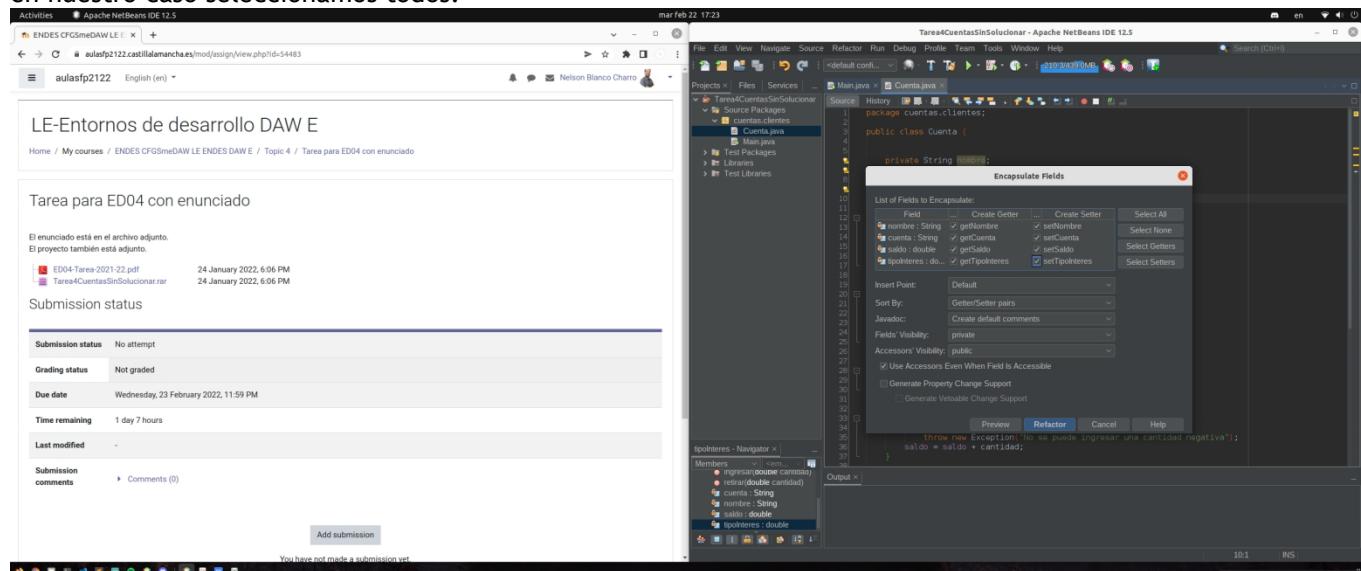


4. Encapsular los atributos de la clase Cuenta. (Crear getters y setters de manera automática)

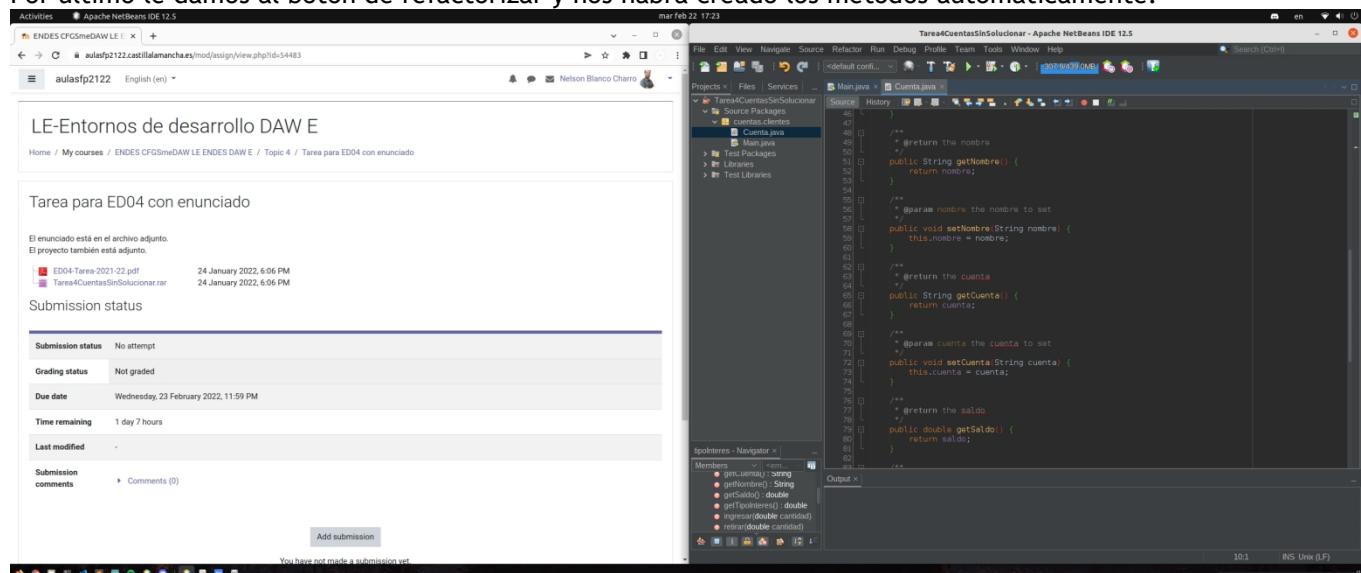
Para ello pulsamos botón derecho sobre cualquier parte de nuestro código y a continuación seleccionamos las siguientes opciones “Refactor” y “Encapsulate Fields”



Nos aparecerá una nueva ventana para seleccionar las variables que queremos crear los métodos getters y setters, en nuestro caso seleccionamos todos.

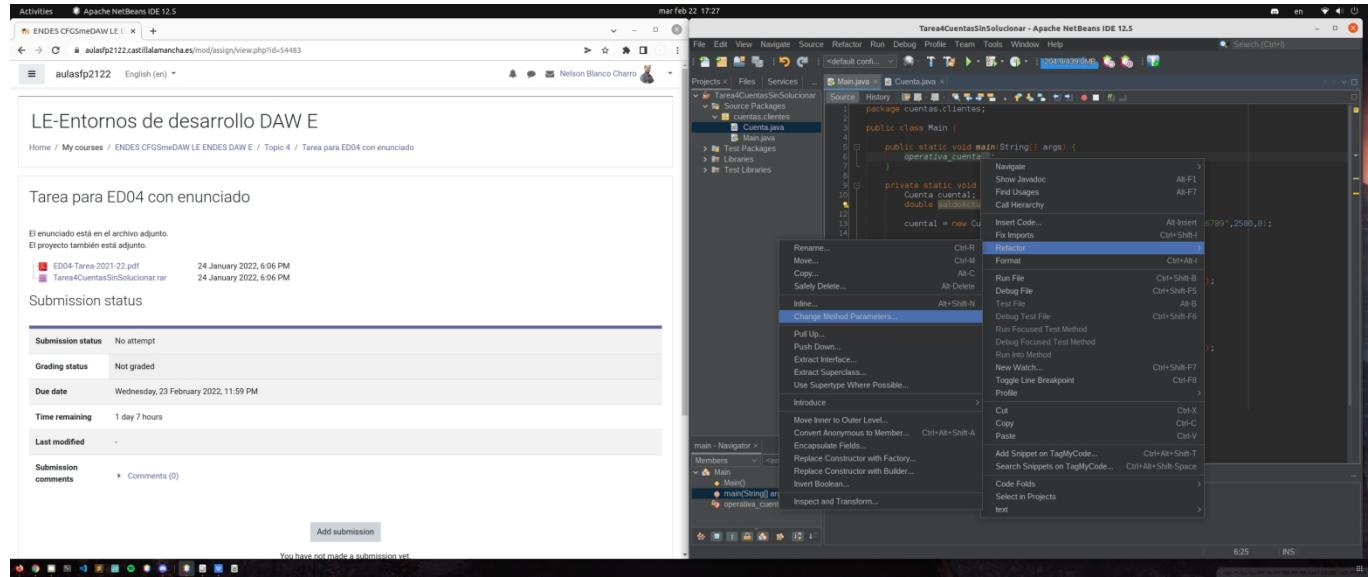


Por último le damos al botón de refactorizar y nos habrá creado los métodos automáticamente.

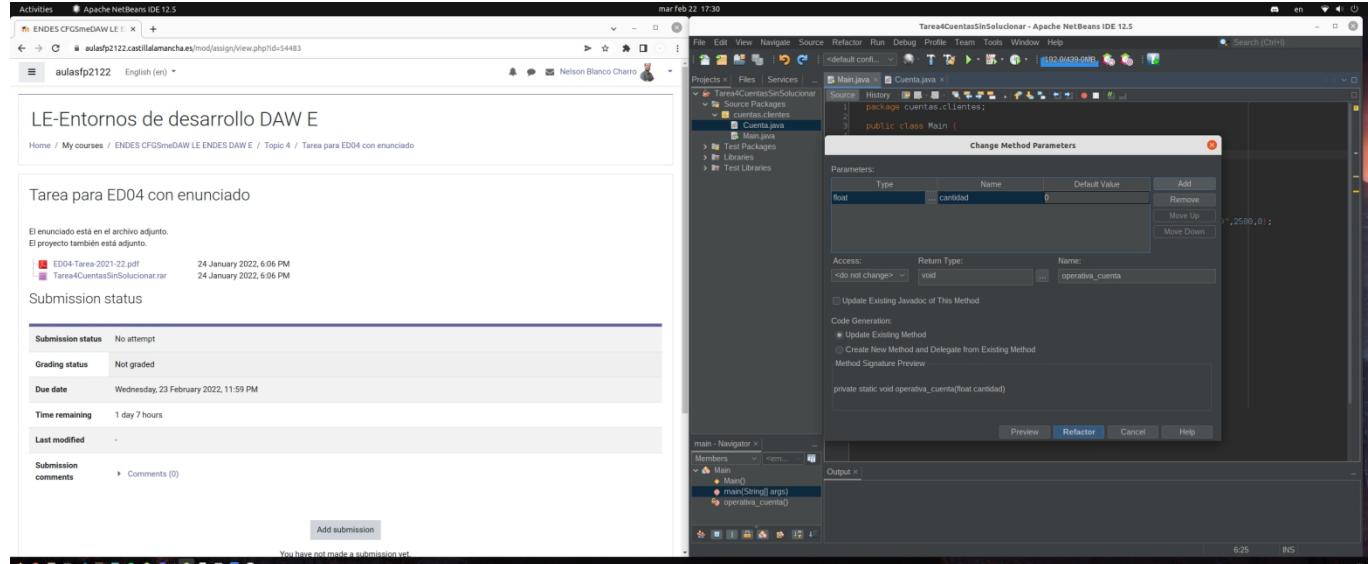


5. Añadir un nuevo parámetro al método operativa_cuenta, de nombre cantidad y de tipo float. (seleccionando la llamada al método, y eligiendo Refactor-Change Method Parameters)

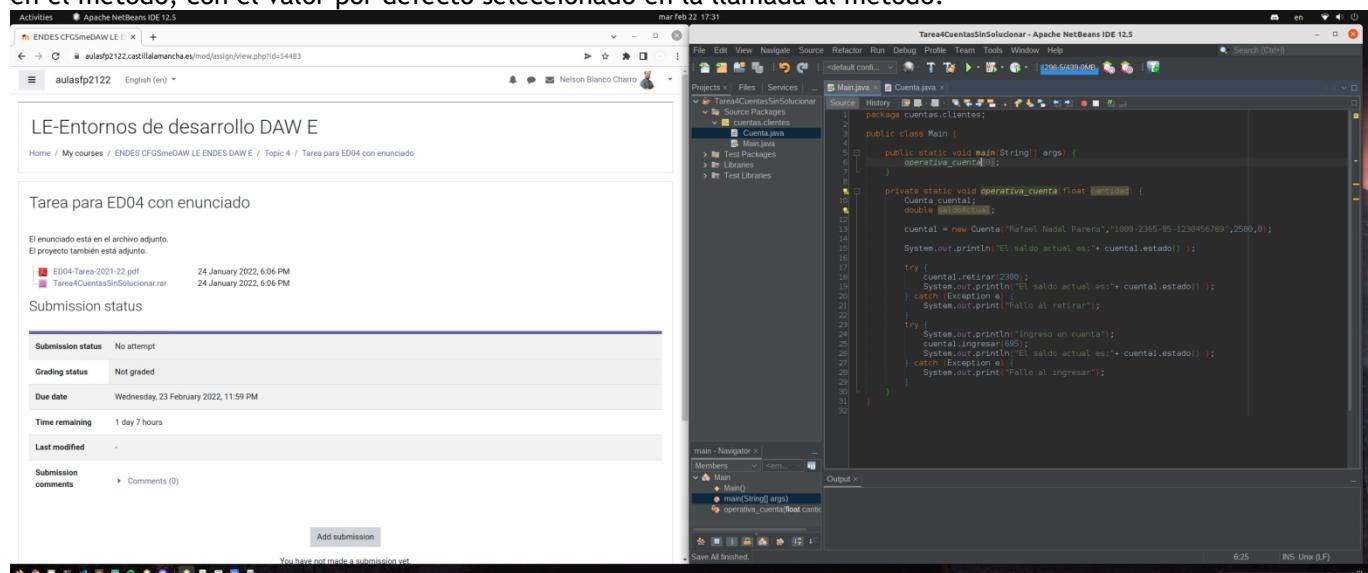
Primero le damos botón derecho a la llamada del método y seleccionamos las opciones “Refactor” y “Change Method Parameters”.



Nos aparecerá una ventana, seleccionamos la opción “Add” e introducimos el tipo, el nombre y el valor por defecto.



Una vez introducidos todos los valores pulsamos en el botón de “Refactor” y se habrá creado un nuevo parámetro en el método, con el valor por defecto seleccionado en la llamada al método.



GIT

1. Configurar GIT para el proyecto dado. Crear un repositorio público en GitHub. (se puede utilizar Netbeans o Eclipse, y el Sistema operativo que prefieras, aunque en el anexo del tema, el ejemplo está realizado para Debian y Eclipse, no te costará mucho trabajo si prefieres en otra plataforma.)

La configuración se hará en Ubuntu y Eclipse. Primero creamos el repositorio en GIT.

The screenshot shows a dual-monitor setup. On the left monitor, a web browser displays a course assignment page for 'Tarea para ED04 con enunciado'. It lists two attachments: 'ED04-Tarea-2021-22.pdf' and 'Tareas4CuentasSinSolucionar.rar'. Below the attachments is a 'Submission status' section. On the right monitor, a terminal window titled 'nelson@nelson-MacBook-Pro ~ /EclipseProjects' is open, showing the output of a 'git init' command. The terminal output includes:

```
git: Using 'master' as the name for the initial branch. This default branch name  
is subject to change. To configure the initial branch name to use in all  
branches of your new repositories, which will suppress this warning, call:  
git config --global init.defaultBranch <name>  
Names commonly chosen instead of 'master' are 'main', 'trunk' and  
'development'. The just-created branch can be renamed via this command:  
git branch -m <name>  
Initialized empty Git repository in /home/nelson/EclipseProjects/.git/  
EclipseProjects git:(master) ^Z
```

A continuación en Eclipse pulsamos botón derecho sobre nuestro proyecto y seleccionamos las opciones “Team” y “Share Project”.

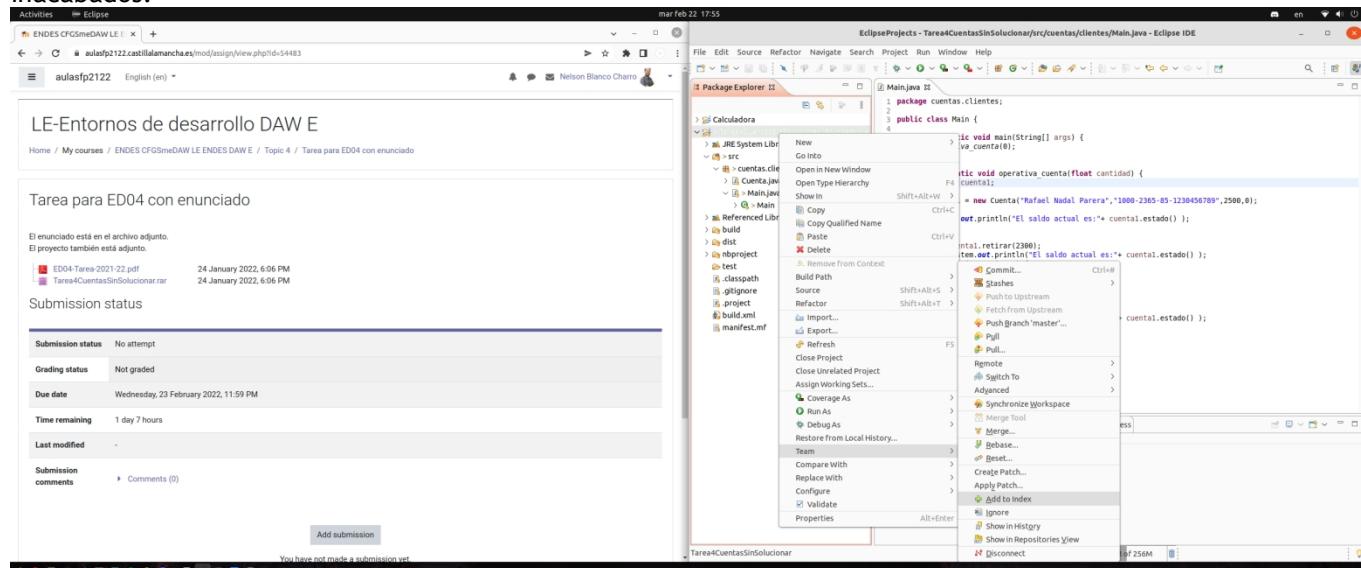
The screenshot shows the Eclipse IDE interface. A Java project named 'Tarea4CuentasSinSolucionar' is selected in the Package Explorer. A context menu is open over the project, with the 'Team' option expanded. The 'Share Project...' option is highlighted with a red box. A tooltip above the menu item says: "Share the project with others using a version and configuration management system." The main workspace shows a Java code editor with a Main.java file containing code related to bank accounts.

Nos aparecerá una ventana donde seleccionaremos el proyecto, o los proyectos, y le damos a finalizar.

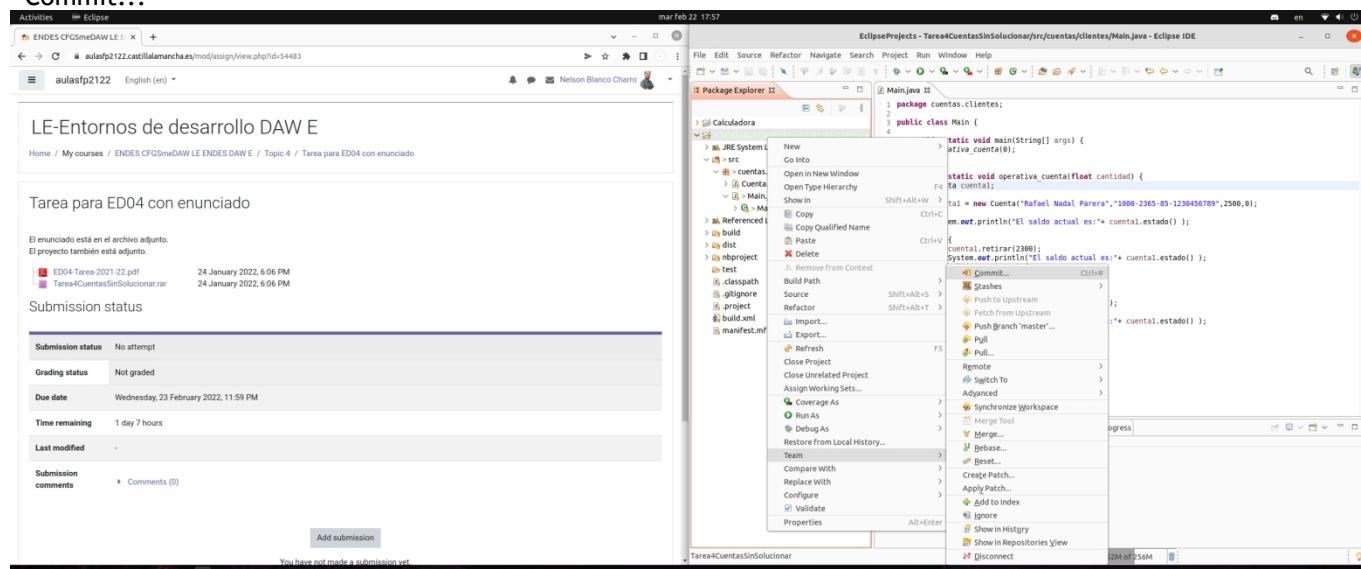
The screenshot shows the Eclipse IDE interface again. The 'Configure GIT Repository' dialog box is open, prompting the user to 'Select repository location'. The 'Project' dropdown is set to 'Tarea4CuentasSinSolucionar' and the 'Location' dropdown is set to '/home/nelson/EclipseProjects/Tarea4CuentasSinSolucionar/.git'. At the bottom of the dialog are 'Cancel' and 'Finish' buttons. The main workspace shows the same Java code editor as the previous screenshot.

2. Realizar, varias modificaciones de código y varias operaciones commit. Comentando el resultado de la ejecución.

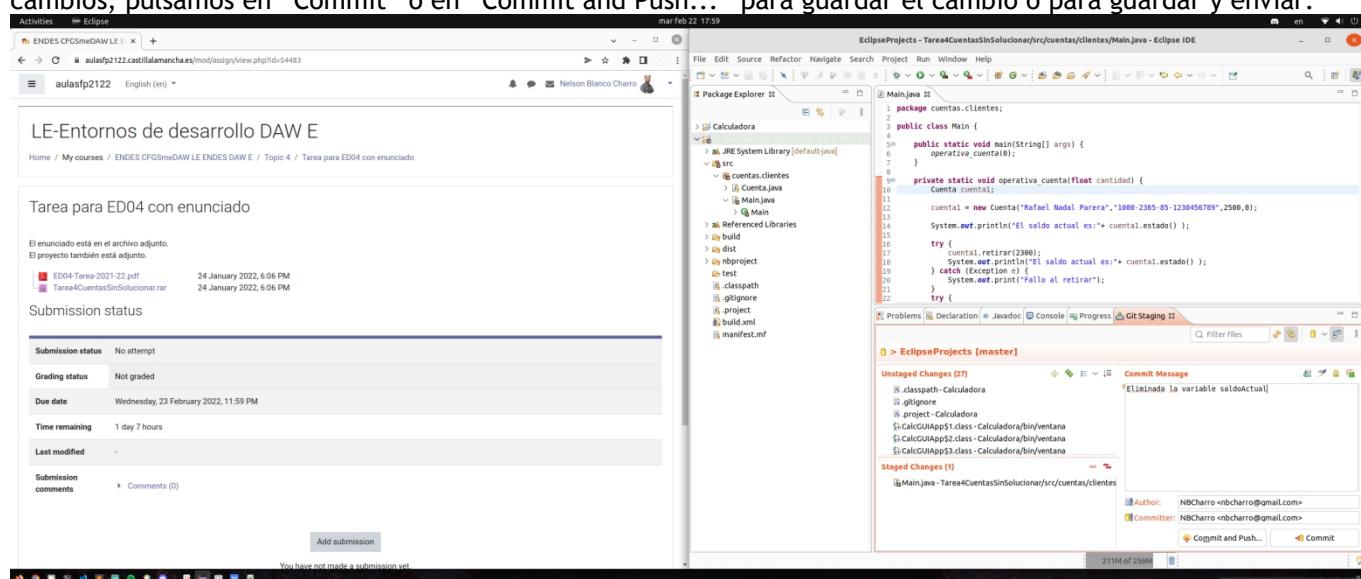
Para realizar un commit primero hay que hacer un cambio en el código, por ejemplo eliminar la variable saldoActual que no se utiliza. Después hay que añadir el cambio al index de cambios, para ello pulsamos con el botón derecho sobre el proyecto y en las opciones “Team” y “Add to Index”, esta opción sirve para indicar que cambios se guardarán en el cambio, quizás haya cambios que no queremos añadir al commit porque están incompletos.



Una vez añadido al index pulsamos nuevamente con el botón derecho en el proyecto y en las opciones “Team” y “Commit...”

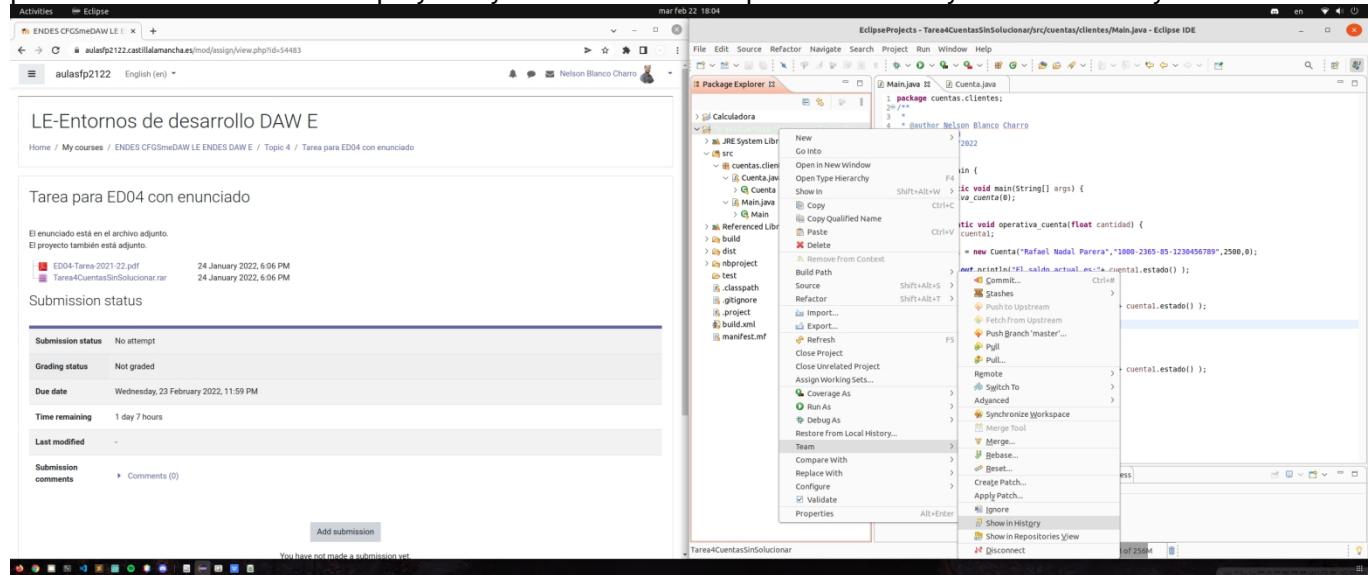


Nos aparecerá una pestaña con los cambios realizados y el comentario que queremos hacer. Una vez hechos los cambios, pulsamos en “Commit” o en “Commit and Push...” para guardar el cambio o para guardar y enviar.

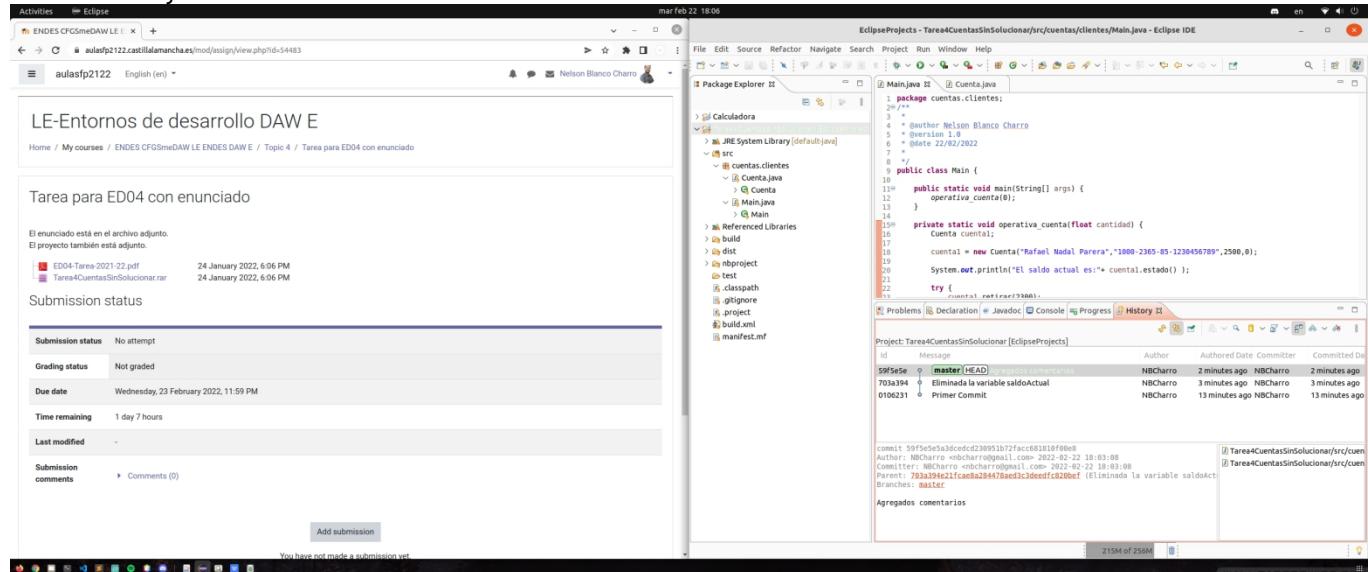


3. Mostrar el historial de versiones para el proyecto.(todo tiene que quedar debidamente documentado en el PDF)

Una vez realizados algunos cambios, añadirlos al index y hacerles commit, para ver el historial de versiones pulsamos botón derecho en el proyecto y seleccionamos las opciones “Team” y “Show in History”.



De esta forma nos aparecerá una pestaña donde nos mostrará el historial con los distintos commits, los comentarios y los archivos afectados.



JavaDoc

1. Insertar comentarios JavaDoc en la clase CCuenta y en la clase Main.

o A nivel de clase incluir al menos las etiquetas

@autor
@version
@since

o En los métodos al menos las etiquetas.

@param
@return
@throws

Comentarios en la clase Main donde se ven los comentarios de la clase y del método.

The screenshot displays the Eclipse IDE environment. On the left, the Package Explorer shows a project structure with packages like 'Calculadora' and 'JRE System Library'. Two tabs are open in the center: 'Main.java' and 'Cuenta.java'. The 'Main.java' tab contains the following code with Javadoc comments:

```
package cuentas.clientes;
/*
 * Author: Nelson Blanco Charro
 * Version: 1.0
 * Since: 22/02/2022
 */
public class Main {
    /**
     * Metodo Main
     */
    public static void main(String[] args) {
        operativa_cuenta();
    }
    /**
     * Código del metodo Main
     * @param cantidad valor a sacar
     */
    private static void operativa_cuenta(float cantidad) {
        Cuenta cuenta;
        cuenta = new Cuenta("Rafael Nadal Parera", "1000-2365-85-1230456789", 2500.0);
        System.out.println("El saldo actual es:" + cuenta.estado());
        try {
            cuenta.retirar(2300);
            System.out.println("El saldo actual es:" + cuenta.estado());
        } catch (Exception e) {
            System.out.println("Fallo al retirar");
        }
        try {
            System.out.println("Ingreso en cuenta");
            cuenta.ingresar(999);
            System.out.println("El saldo actual es:" + cuenta.estado());
        } catch (Exception e) {
            System.out.println("Fallo al ingresar");
        }
    }
}
```

The 'Cuenta.java' tab shows the definition of the 'Cuenta' class.

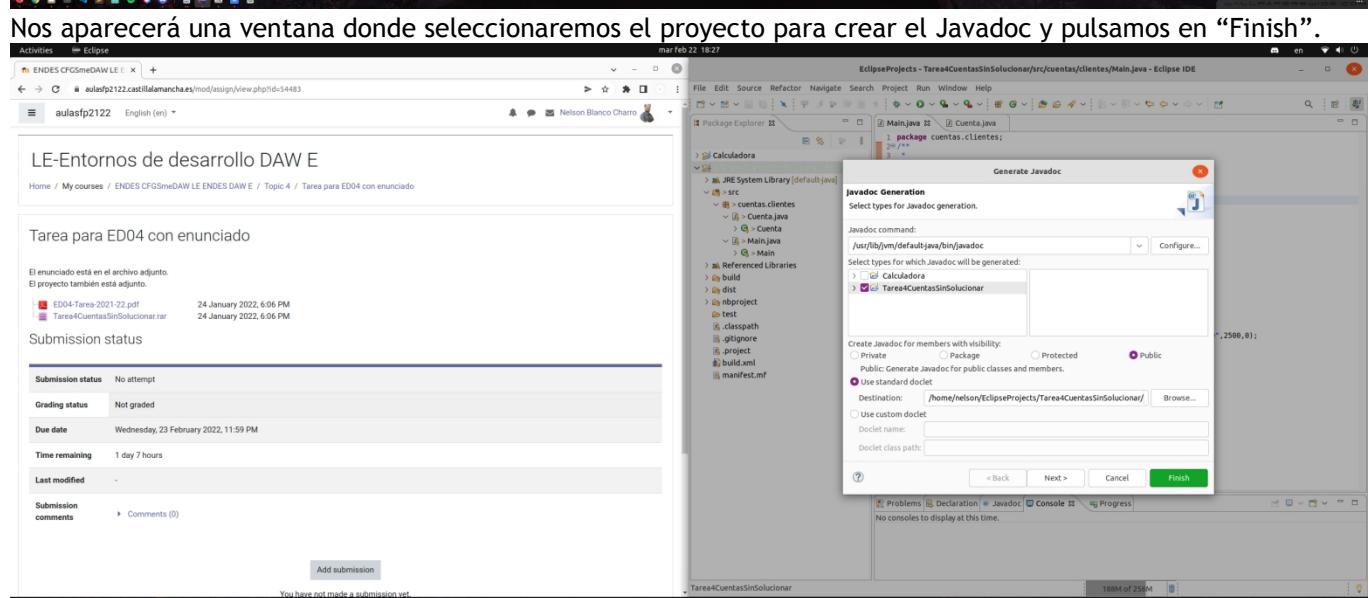
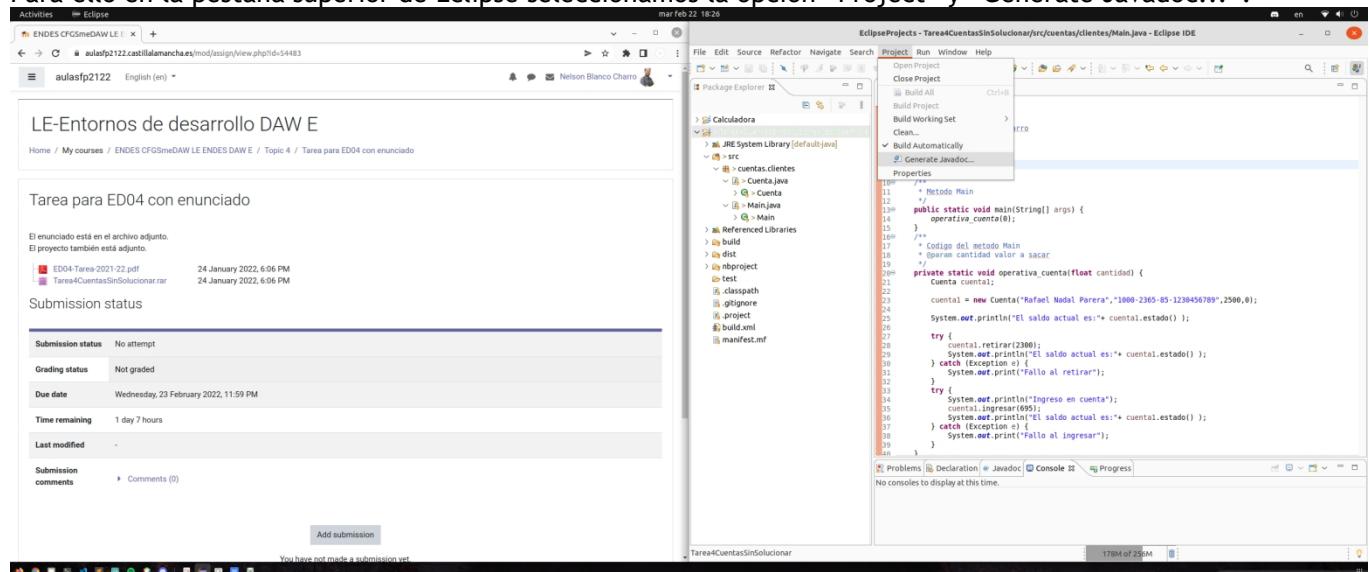
Comentarios en la clase Cuentas donde se ven los comentarios en los distintos métodos.

The screenshot displays the Eclipse IDE environment. On the left, the Package Explorer shows a project structure with packages like 'Calculadora' and 'JRE System Library'. Two tabs are open in the center: 'Main.java' and 'Cuenta.java'. The 'Main.java' tab contains the same Java code as the previous screenshot, with Javadoc comments. The 'Cuenta.java' tab shows the definition of the 'Cuenta' class, which includes methods for depositing and withdrawing money, each with its own Javadoc comments.

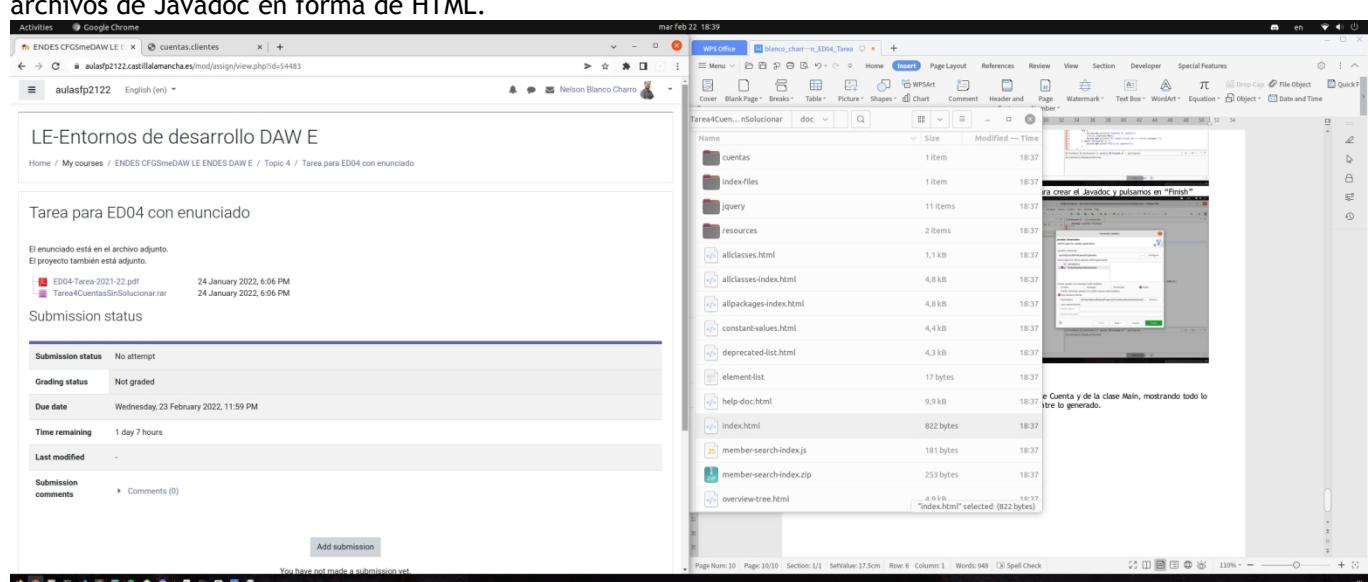
```
49    * Método público para ingresar dinero
50    * @param cantidad valor a introducir
51    * @throws Exception No se puede ingresar una cantidad negativa
52    */
53    public void ingresar(double cantidad) throws Exception {
54        if (cantidad<0)
55            throw new Exception("No se puede ingresar una cantidad negativa");
56        setSaldo(getSaldo() + cantidad);
57    }
58    /**
59     * Método público para retirar dinero
60     * @param cantidad valor a sacar
61     * @throws Exception No se puede retirar una cantidad negativa
62     * @throws Exception No hay suficiente saldo
63     */
64    public void retirar(double cantidad) throws Exception {
65        if (cantidad < 0)
66            throw new Exception("No se puede retirar una cantidad negativa");
67        if (estado()<= cantidad)
68            throw new Exception("No se hay suficiente saldo");
69        setSaldo(getSaldo() - cantidad);
70    }
71    /**
72     * Método getter del Nombre del cliente
73     * @return nombre el nombre del cliente
74     */
75    public String getNombre() {
76        return nombre;
77    }
78    /**
79     * Método setter del Nombre del cliente
80     * @param nombre nombre del cliente que cambiara
81     */
82    public void setNombre(String nombre) {
83        this.nombre = nombre;
84    }
85}
```

2. Generar documentación JavaDoc para todo el proyecto.

Para ello en la pestaña superior de Eclipse seleccionamos la opción “Project” y “Generate Javadoc...”.



Una vez finalizado se abra creado en la carpeta del proyecto una nueva carpeta llamada “doc” con los distintos archivos de Javadoc en forma de HTML.



3. Comprueba que abarca todos los métodos y atributos de la clase Cuenta y de la clase Main, mostrando todo lo generado en el PDF, dando pantallazos suficientes y navegando entre lo generado.

Cuando abramos el archivo index.html de Javadoc nos aparecerá las clases creadas.

The screenshot shows the 'cuentas.clientes' package summary page. At the top, there are tabs for 'PACKAGE', 'CLASS', 'USE', 'TREE', 'DEPRECATED', 'INDEX', and 'HELP'. Below the tabs, the package name 'cuentas.clientes' is displayed. Underneath, there are two main sections: 'Class Summary' and 'Description'. The 'Class Summary' section lists the classes 'Cuenta' and 'Main'. For each class, there is a brief description and a link to its detailed documentation. Below these sections, there are tabs for 'PACKAGE', 'CLASS', 'USE', 'TREE', 'DEPRECATED', 'INDEX', and 'HELP'. At the bottom of the page, there is a search bar labeled 'Search'.

The screenshot shows the 'Main' class summary page. At the top, there are tabs for 'PACKAGE', 'CLASS', 'USE', 'TREE', 'DEPRECATED', 'INDEX', and 'HELP'. Below the tabs, the class name 'Main' is displayed. Underneath, there is a 'Constructor Summary' section with a 'Constructors' tab. It lists the constructor 'Main()' with a brief description. Below this, there is a 'Method Summary' section with tabs for 'All Methods', 'Static Methods', and 'Concrete Methods'. It lists the static method 'main(String[] args)' with a brief description. At the bottom of the page, there is a search bar labeled 'Search'.

The screenshot shows the 'Cuenta' class summary page. At the top, there are tabs for 'PACKAGE', 'CLASS', 'USE', 'TREE', 'DEPRECATED', 'INDEX', and 'HELP'. Below the tabs, the class name 'Cuenta' is displayed. Underneath, there is a 'Constructor Summary' section with a 'Constructors' tab. It lists the constructor 'Cuenta()' with a brief description. Below this, there is a 'Method Summary' section with tabs for 'All Methods', 'Instance Methods', and 'Concrete Methods'. It lists several methods: 'estado()', 'getCuenta()', 'getNombre()', 'getSaldo()', 'getTipoInteres()', 'ingresar(double cantidad)', 'retirar(double cantidad)', 'setCuenta(String cuenta)', 'setNombre(String nombre)', 'setSaldo(double saldo)', and 'setTipoInteres(double tipoInteres)'. Each method is accompanied by a brief description. At the bottom of the page, there is a search bar labeled 'Search'.

Métodos con @param, @return y @throws.

The screenshot shows a dual-pane interface. On the left, a web browser displays assignment details for 'Tarea para ED04 con enunciado'. It includes submission status (No attempt), grading status (Not graded), due date (Wednesday, 23 February 2022, 11:59 PM), time remaining (1 day 7 hours), and a submission button. On the right, an Eclipse Java code editor shows the 'Cuenta' class with its methods: `retirar`, `getNombre`, `setNombre`, and `getCuenta`. Each method is annotated with `@param`, `@return`, and `@throws`.

```
public void retirar(double cantidad) throws Exception
Metodo publico para retirar dinero
Parameters:
cantidad - valor a sacar
Throws:
Exception - No se puede ingresar una cantidad negativa
Exception - No hay suficiente saldo

public String getNombre()
Metodo getter del Nombre del cliente
Returns:
nombre el nombre del cliente

public void setNombre(String nombre)
Metodo setter del Nombre del cliente
Parameters:
nombre - nombre del cliente que cambiara

public Cuenta(String cuenta)
Metodo setter del numero de cuenta del cliente
Parameters:
cuenta - numero de cuenta que cambiara
```

Métodos getters y setters de la clase Cuenta.

This screenshot is similar to the one above, showing the same assignment details on the left and the Eclipse Java code editor on the right. The code editor displays the `Cuenta` class with methods `setCuenta`, `getSaldo`, `setSaldo`, and `getTipoInteres`, each annotated with `@param`, `@return`, and `@throws`.

```
public void setCuenta(String cuenta)
Metodo setter del numero de cuenta del cliente
Parameters:
cuenta - numero de cuenta que cambiara

public double getSaldo()
Metodo getter del saldo del cliente
Returns:
saldo cantidad de dinero del cliente

public void setSaldo(double saldo)
Metodo setter del saldo del cliente
Parameters:
saldo - el saldo que cambiara

public double getTipoInteres()
Metodo getter del tipo de interes del cliente
```