Tarea para ED04.

Detalles de la tarea de esta unidad.

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.

Pulsa aquí para descargar dicho proyecto ("Deposito.rar").

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

REFACTORIZACIÓN

1. Las clases deberán formar parte del paquete cuentas.
2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".
3. Introducir el método operativa_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.
4. Encapsular los atributos de la clase CCuenta.
5. Añadir un nuevo parámetro al método operativa_cuenta, de nombre cantidad y de tipo float.

GIT

1. Configurar GIT para el proyecto. Crear un repositorio público en GitHub.
2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.
3. Mostrar el historial de versiones para el proyecto mediante un comando desde consola.

JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.
2. 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.

Los criterios de puntuación serán los siguientes:

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.

Recursos necesarios para realizar la Tarea.

Ordenador con el **IDE que se vaya a usar.**
Proyecto **Java** "deposito" disponible en este enlace.
Conexión a Internet si precisas la instalación de **GIT** o trabajas con **GitHub**.

Consejos y recomendaciones.

Se pretende que realices las operaciones de refactorización, control de versiones y documentación de una aplicación. Se parte de la base, de que has adquirido conocimientos suficientes de programación.

Indicaciones de entrega.

Una vez realizada la tarea elaborarás un único documento donde figure la URL del repositorio github (contendrá el proyecto con el código refactorizado y con la documentación generada con javadoc) y una imagen con la respuesta a la actividad 8. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_SIGxx_Tarea.pdf

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna **Begoña Sánchez Mañas para la cuarta unidad del MP de ED**, debería nombrar esta tarea como...

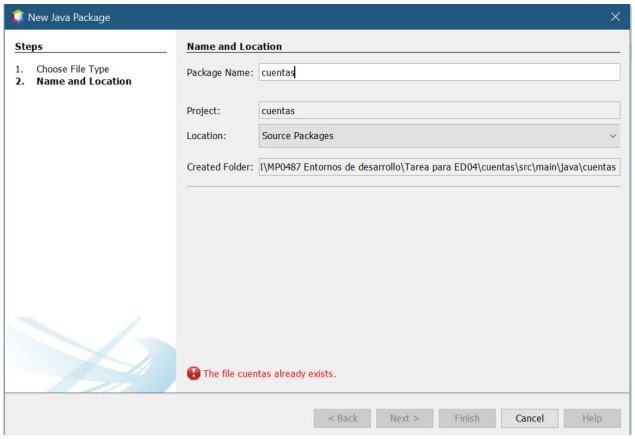
sanchez_manas_begona_ED04_Tarea.pdf

Estado da entrega

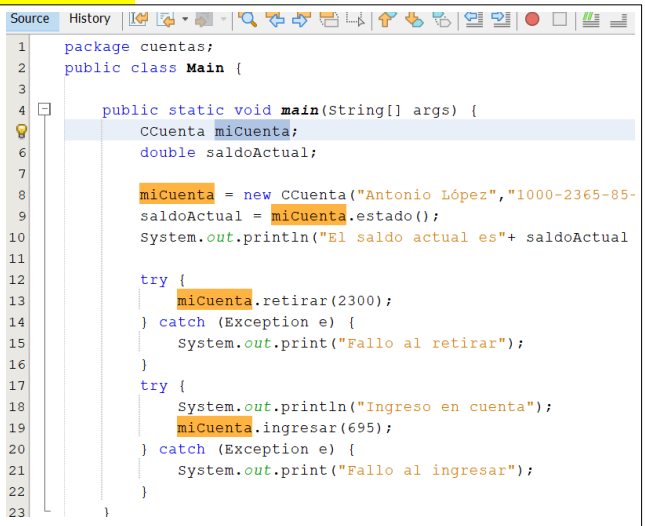
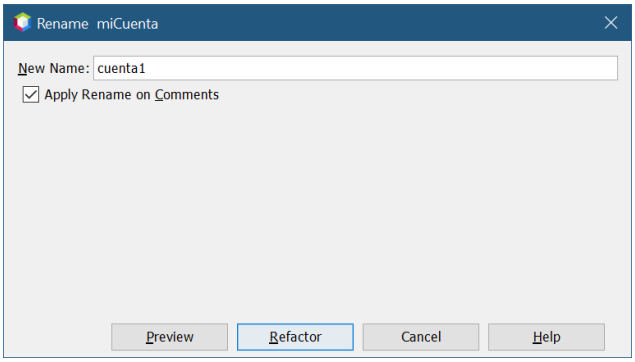
Estado da entrega	Sen intentos
Estado das cualificacións	Sen cualificar
Data límite	Domingo, 13 de Marzo de 2022, 23:59
Tempo restante	11 días 5 horas
Última modificación	-
Comentarios a entrega	Comentarios (0)

REFACTORIZACIÓN

Las clases deberán formar parte del paquete cuentas.

1- Creamos el paquete “cuentas” dentro del proyecto en Netbeans	
2- Añadimos manualmente “package cuentas” al inicio de ambos archivos *.java	
3- Movemos los archivos *.java al paquete “cuentas”	

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

1- Seleccionamos el término a refactorizar por sustitución, en este caso, “miCuenta”	
En Netbeans pulsamos Refactor> Rename. Un cuadro de diálogo nos pedirá el nuevo nombre para el objeto. En este caso indicamos “cuenta1” como se solicita en el enunciado y pulsamos “Refactor”	

Netbeans realiza todas las sustituciones necesarias.

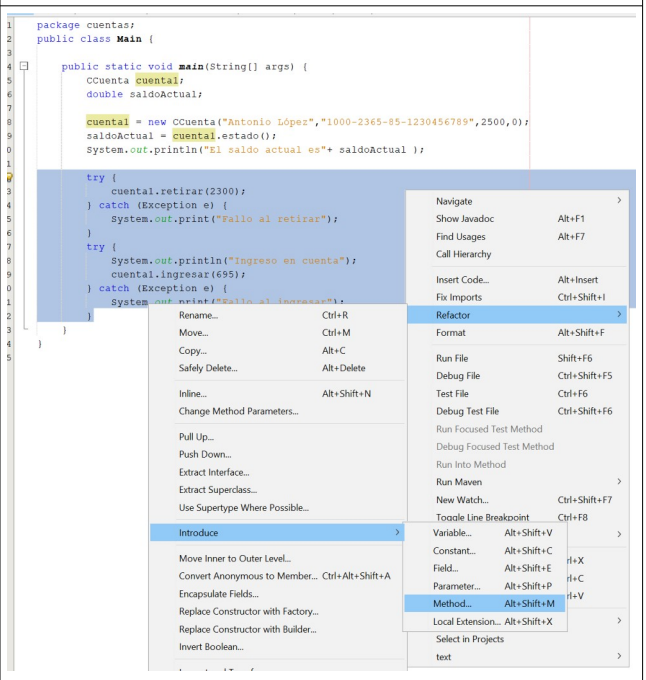
```java
package cuentas;
public class Main {

    public static void main(String[] args) {
        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta("Antonio López","1000-2365-85-1230456
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es"+ saldoActual );

        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");
        }
    }
```

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

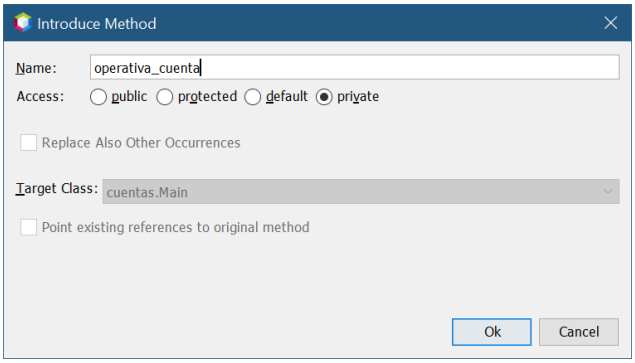
Seleccionamos las sentencias de la clase Main que operan con el objeto cuenta1. En este caso se trata de los Try / Catch.

```java
package cuentas;
public class Main {

    public static void main(String[] args) {
        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta("Antonio López","1000-2365-85-1230456789",2500,0);
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es"+ saldoActual );

        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");
        }
    }
}
```

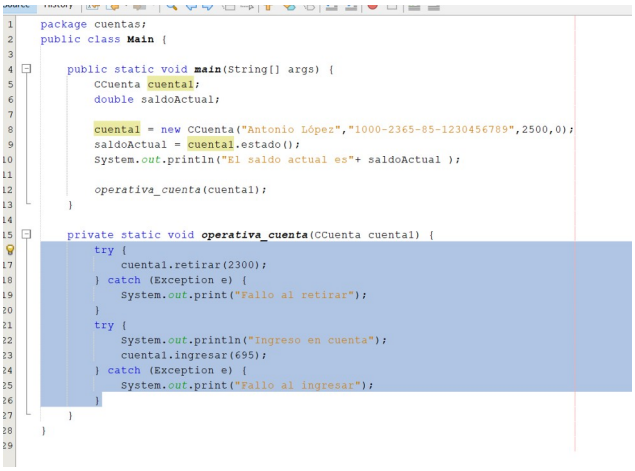
Seleccionamos Botón derecho > Refactor > Introduce > Method



Nos solicita un Nombre. Indicamos “operativa_cuenta” como nos solicita el enunciado.



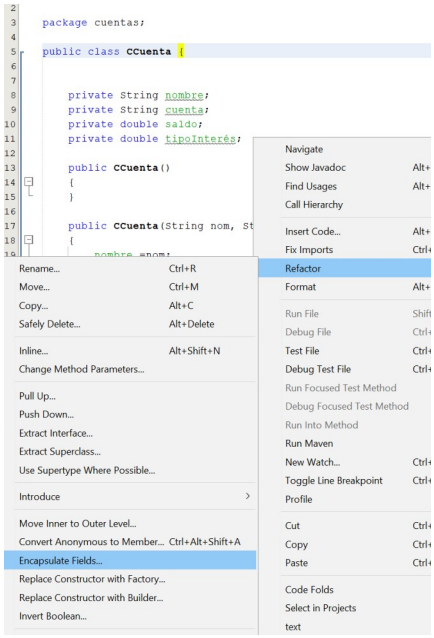
Comprobamos que ahora los try/catch han sido contenidos dentro de un nuevo método.



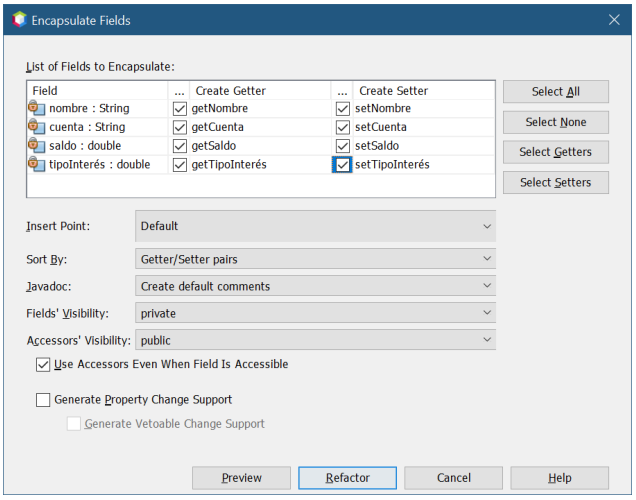
```
package cuentas;
public class Main {

    public static void main(String[] args) {
        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta("Antonio López","1000-2365-85-1230456789",2500,0);
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es"+ saldoActual );

        operativa_cuenta(cuenta1);
    }

    private static void operativa_cuenta(CCuenta cuenta1) {
        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");
        }
    }
}
```

Encapsular los atributos de la clase CCuenta.,

Para encapsular (que sólo sean accesibles mediante métodos set y set) los atributos de la clase Cccuenta, procedemos a pulsar:
Botón derecho > Refactor > Encapsulate Fields



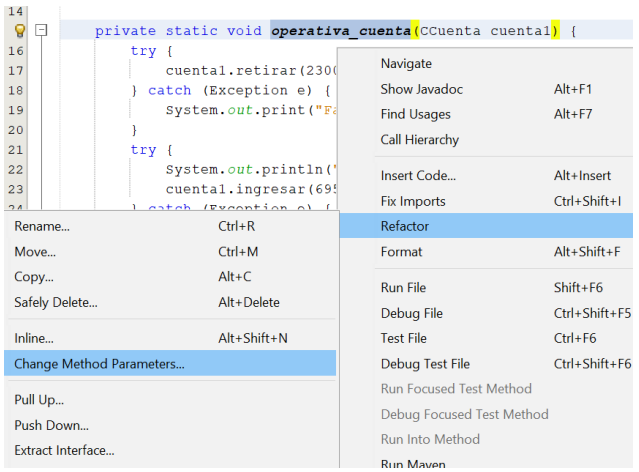
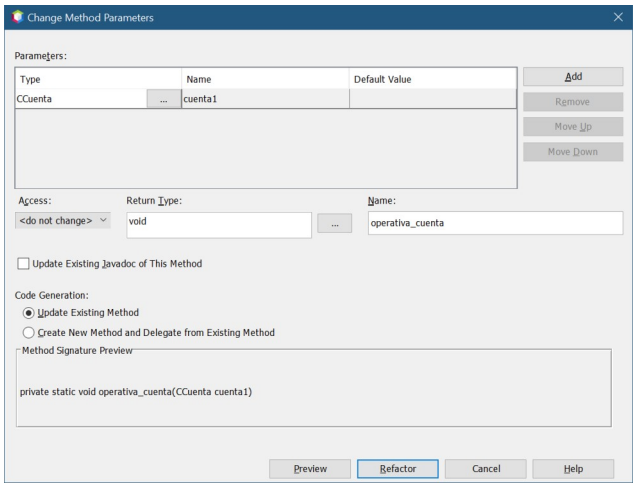
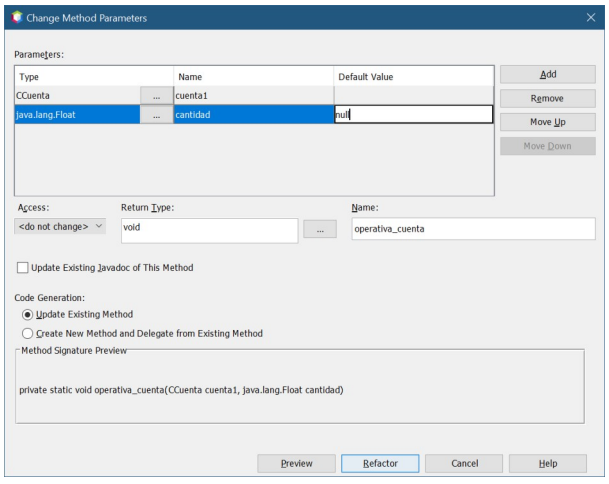
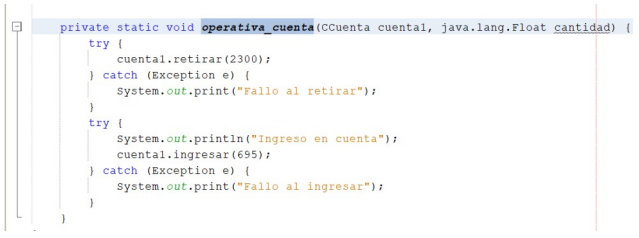
En el cuadro de diálogo nos pregunta qué métodos deben crearse, indicamos los get y set para todos los atributos (select all) y pulsamos “Refactor”.



Se crean todos los get y set.

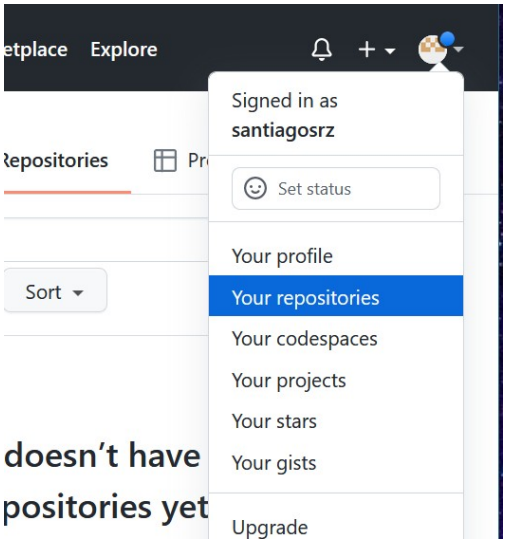
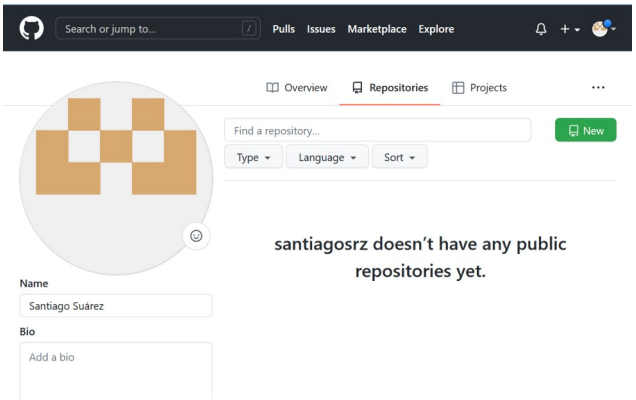
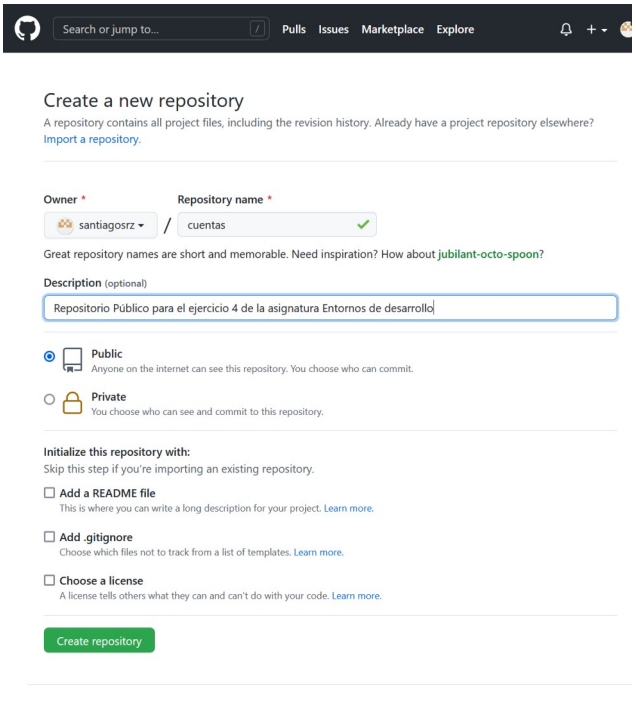
```
43      }
44
45      /**
46       * @return the nombre
47       */
48      public String getNombre() {
49          return nombre;
50      }
51      /**
52       * @param nombre the nombre to set
53       */
54      public void setNombre(String nombre) {
55          this.nombre = nombre;
56      }
57      /**
58       * @return the cuenta
59       */
60      public String getCuenta() {
61          return cuenta;
62      }
63      /**
64       * @param cuenta the cuenta to set
65       */
66      public void setCuenta(String cuenta) {
67          this.cuenta = cuenta;
68      }
69      /**
70       * @return the saldo
71       */
72      public double getSaldo() {
73          return saldo;
74      }
75      /**
76       * @param saldo the saldo to set
77       */
78      public void setSaldo(double saldo) {
79          this.saldo = saldo;
80      }
81      /**
82       * @return the tipoInterés
83       */
84      public double getTipoInterés() {
85          return tipoInterés;
86      }
87      /**
88       * @param tipoInterés the tipoInterés to set
89       */
90      public void setTipoInterés(double tipoInterés) {
91          this.tipoInterés = tipoInterés;
92      }
93  }
```

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

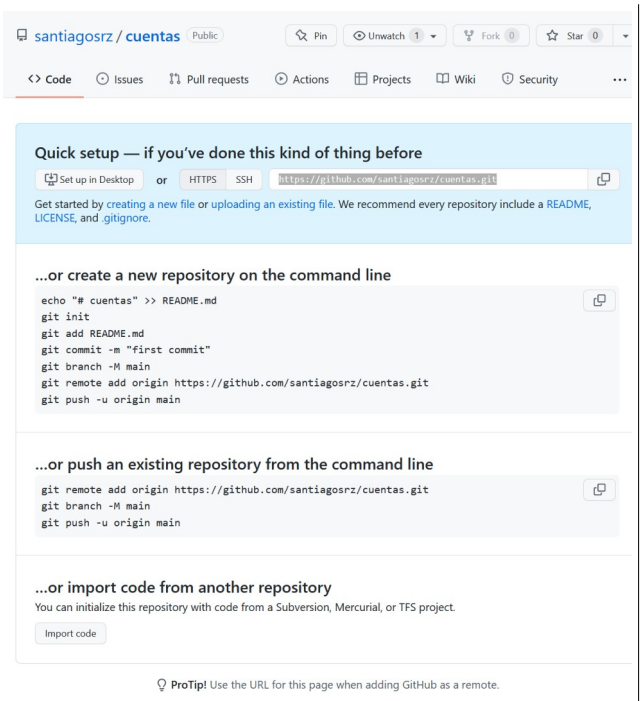
Seleccionamos “operativa_cuenta” y pulsamos botón derecho > Refactor > Change Method Parameters...	
Nos muestra un cuadro de diálogo en el que aparecen los parámetros actuales de método, en este caso “cuenta1” del tipo “CCuenta”	
Pulsamos “add”. En type seleccionamos “float” y en nombre indicamos “cantidad” como se nos solicita en el enunciado. Una vez cubierto, pulsamos “Refactor”	
Comprobamos que la cabecera del método operativa_cuenta ahora se menciona el parámetro float “cantidad”.	

GIT

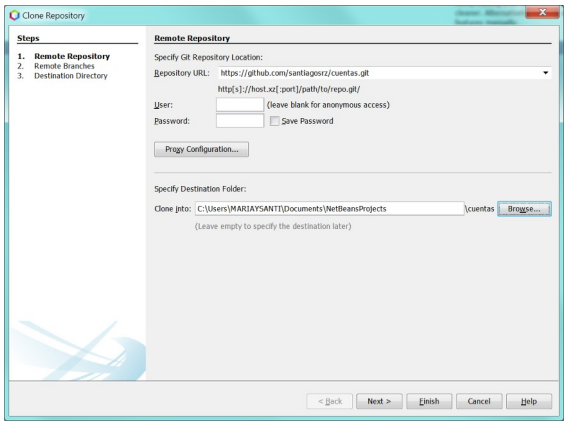
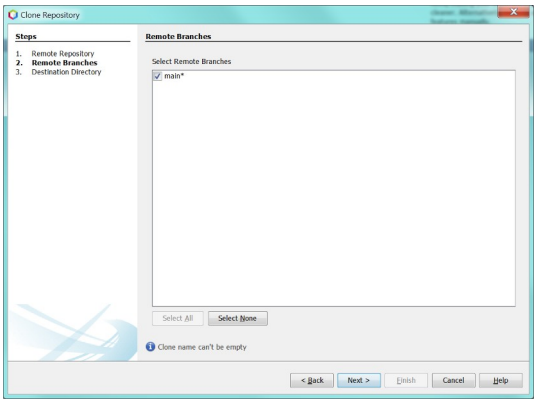
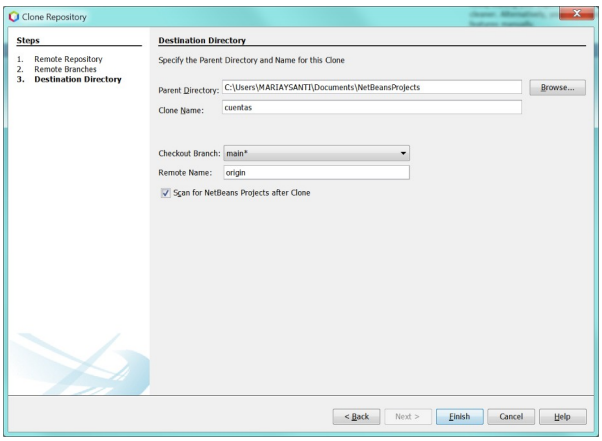
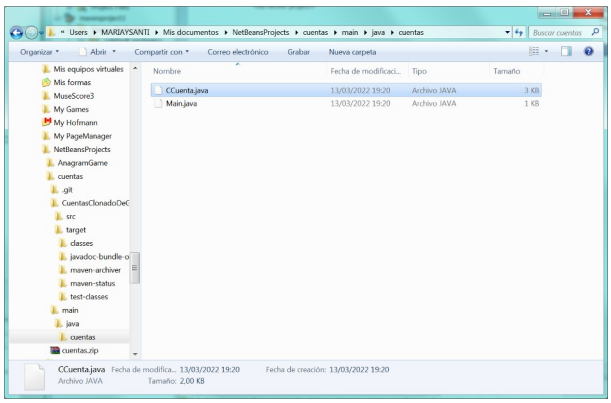
Configurar GIT para el proyecto. Crear un repositorio público en GitHub.

Desde la página principal de GitHub pulsamos el botón del perfil en la esquina superior derecha y vamos a “Your repositories”	
Pulsamos “New”	
En el formulario indicamos la el nombre, la descripción e indicamos que debe ser público.	

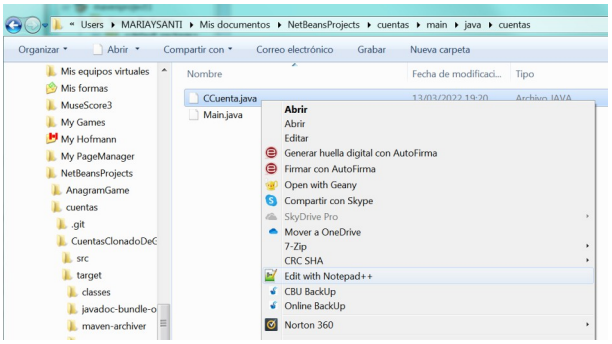
Finalmente nos da el resumen de la creación del git. En este caso: https://github.com/santiagosrz/cuentas.git



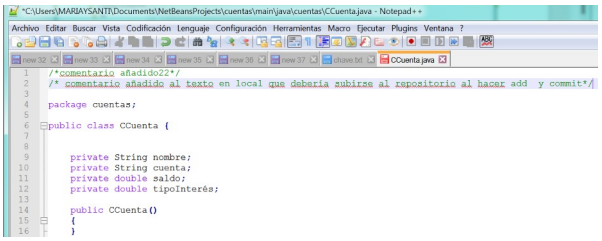
Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.

En primer lugar sincronizaremos la información en el git con la local, ya que hemos subido los archivos. Indicamos URL para el git y la carpeta local donde lo vamos a clonar.	
Nos pregunta la pregunta la rama remota del git que queremos clonar a la máquina local. En este caso main	
Finalmente nos pregunta la ruta, y carpeta de lo clonado.	
Comprobamos que efectivamente se ha clonado a la carpeta local. C:\Users\MARIAYSANTI\Documents\NetBeansProjects\cuentas\main\java\cuentas	

Editamos el archivo con Notepad++

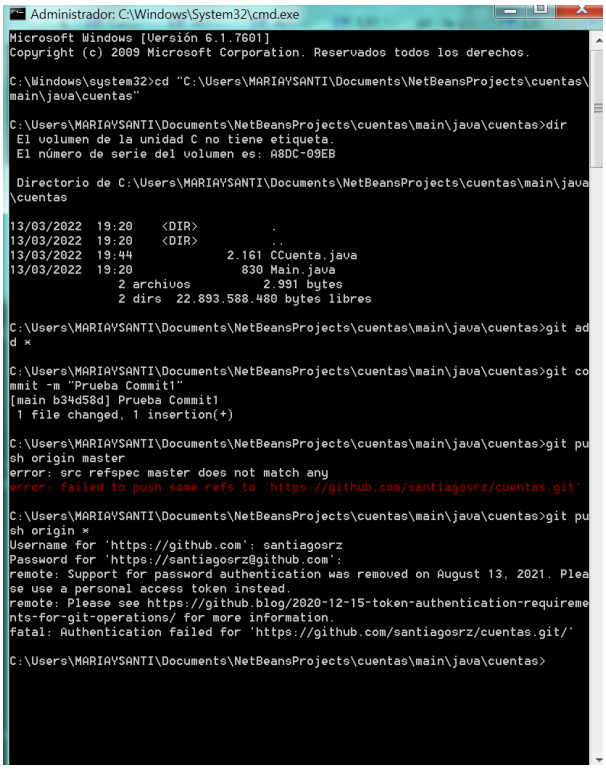


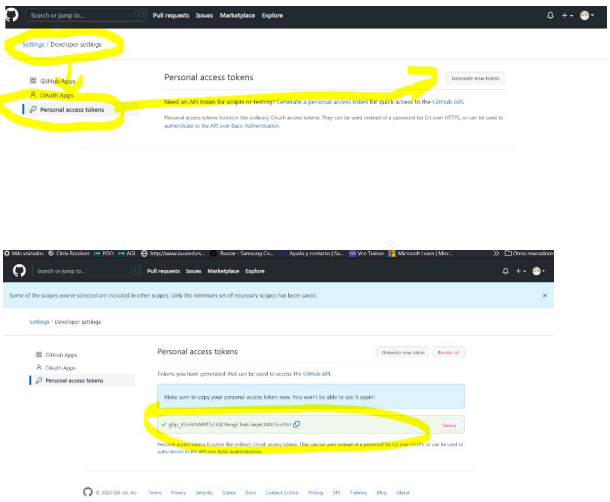
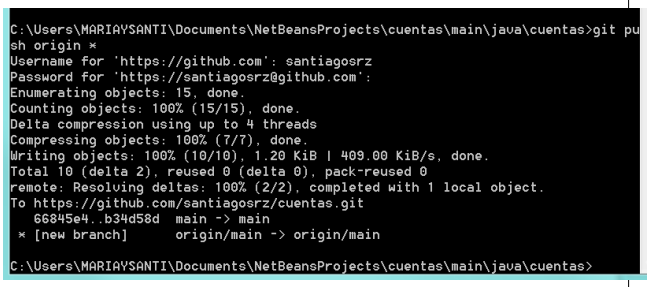
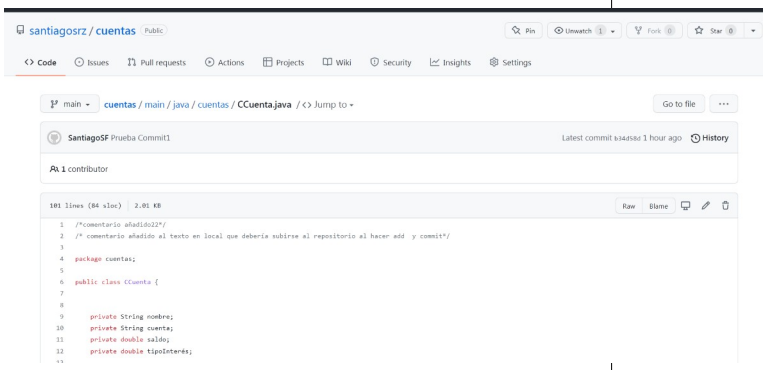
Añadimos un comentario que debería aparecer en el repositorio tras hacer add yh commit



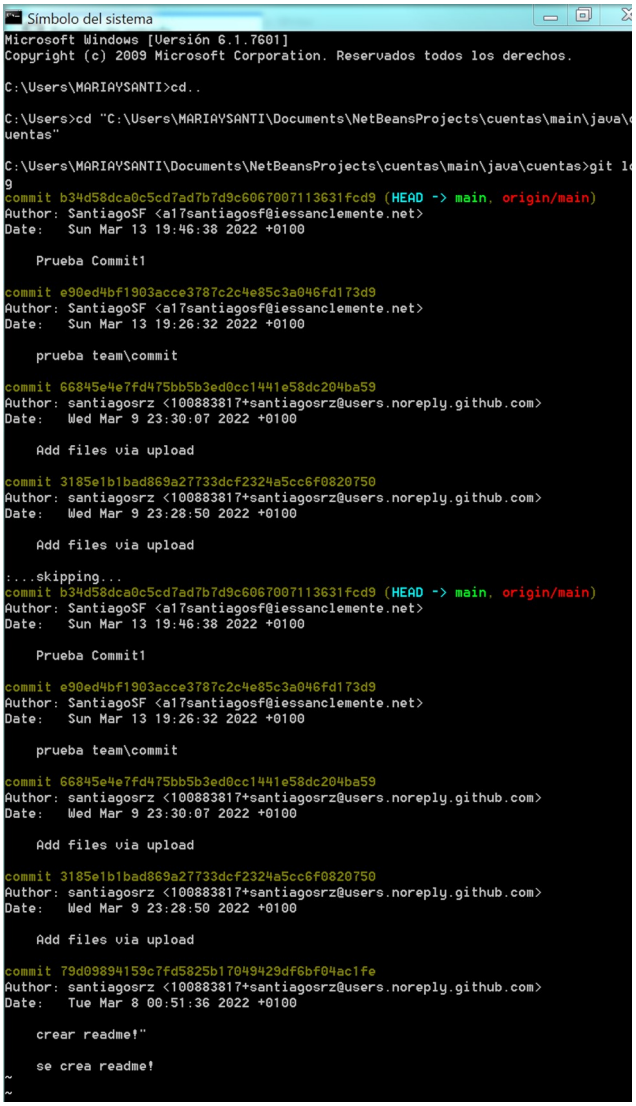
Ejecutamos
git add * > Se añaden todos los cambios en todos los archivos del proyecto .
git commit –m “Prueba Commit1” >Git detecta los cambios realizados y los añade en la bd local para subir. Nos informe del numero de archivos e inserciones realizadas.
git push origin * > se solicita que se ejecute el acceso al repositorio git y se ejecuten los cambios en remoto que se han almacenado en la bd local a través del commit.

Aquí dio error. Al introducir usuario y contraseña desde la línea de comando nos informa que desde el 13/08/2021 no se admite validación con usuario y contraseña normal, sino que como contraseña debe usarse un token especialmente creado con unos permisos determinados y con una validez dada.



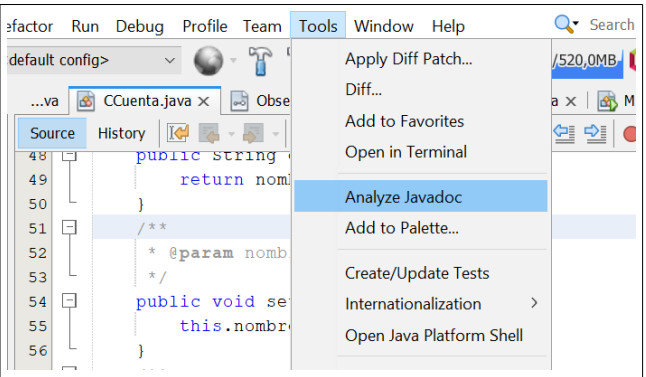
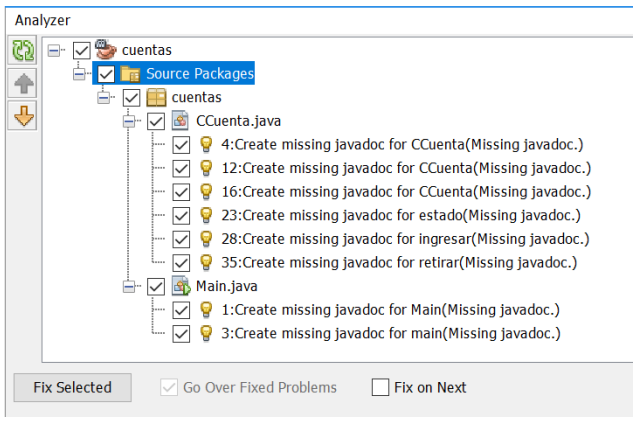
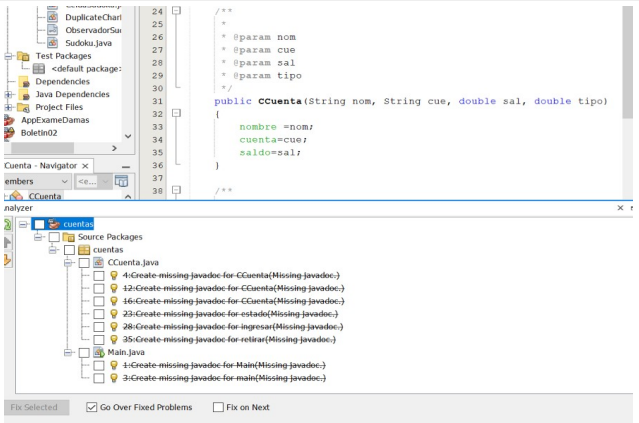
Creamos el token .	
Volvemos a hacer el Push, pero introduciendo esta vez el token. Se realizan los cambios que añadimos en el último commit.	
Comprobamos que el comentario de prueba ya está en la versión del repositorio git.	

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

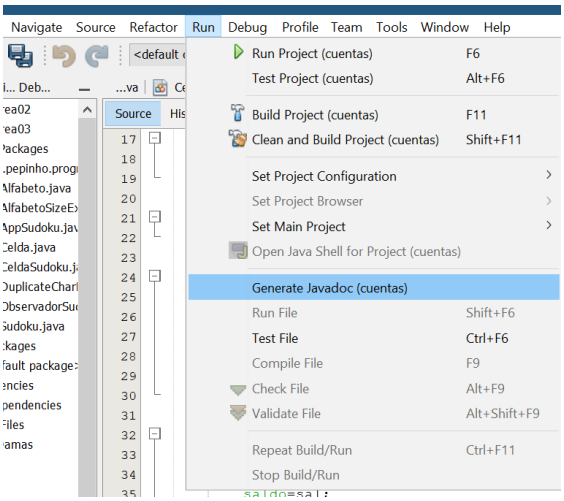
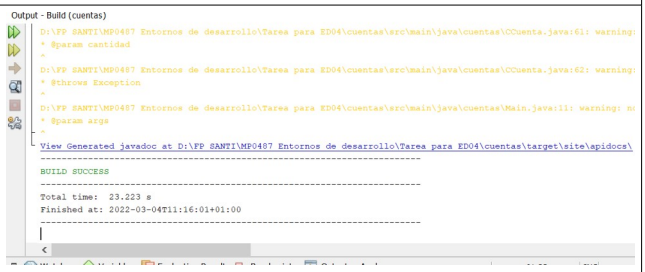
Para conseguier el historial de cambios del git, solicitamos : **git log** Nos dará el historial de cambios realizados en el proyecto.	

JAVADOC

Insertar comentarios JavaDoc en la clase CCuenta.

Para generar el Javadoc, en primer lugar seleccionamos el alcance (en nuestro caso el paquete “cuentas”) y luego vamos a Tools > Analyze Javadoc	
En la ventana “Analyzer” que se nos presenta, aparecen todos los objetos que no tienen Javadoc creados, así como la línea donde deberían ir. Seleccionamos todo y pulsamos “Fix Selected”.	
Se añaden los comentarios Javadoc a los archivos java y en el analizador ahora aparecen tachados, conforme ya se han realizado. Podemos comprobar en el archivo que se han realizado los comentarios Javadoc.	

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

Para generar el informe Javadoc comprobamos que tenemos seleccionado el paquete cuentas que es el que queremos analizar. Pulsamos Run > Generate Javadoc	
Se genera dentro del la carpeta del proyecto una carpeta cuentas (al ser el paquete analizado) \ target\site\apidocs	
En esa ruta encontramos index.html donde se resume lo documentado.