

Tarea para ED04.



Julio Antonio Sanmartino Rodríguez

Tarea para ED04.

Detalles de la tarea de esta unidad.

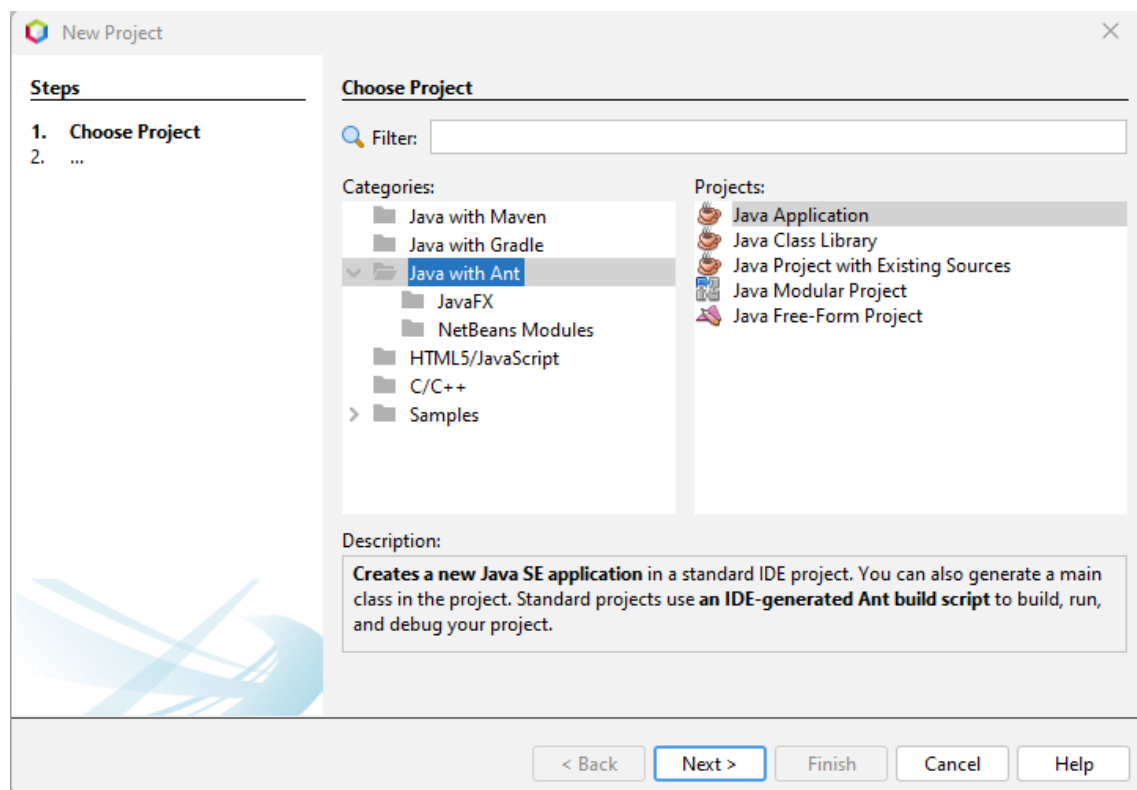
Enunciado.

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

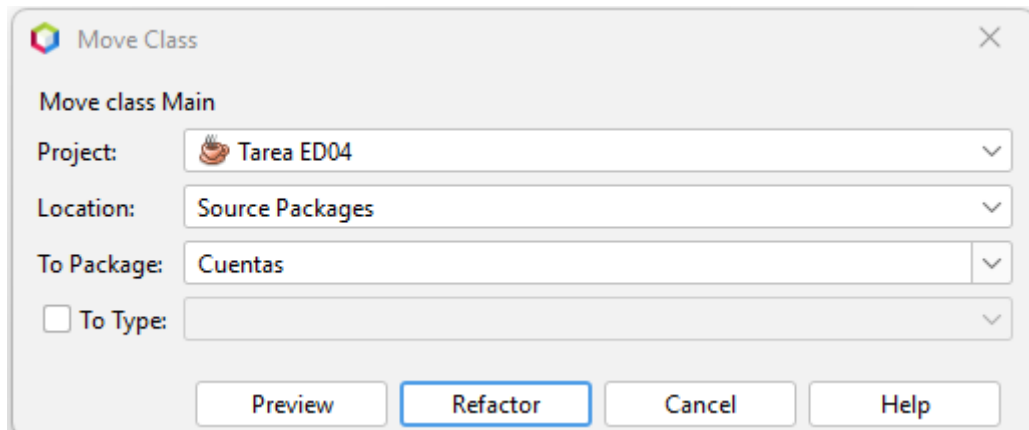
REFACTORIZACIÓN

1. Las clases deberán formar parte del paquete cuentas.

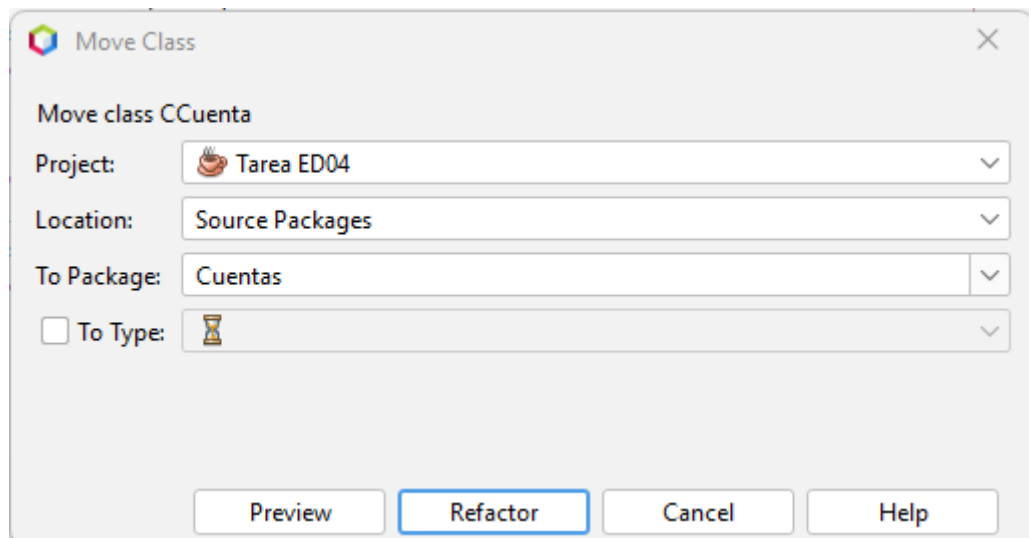
En primer lugar, creamos un nuevo proyecto en NetBeans de nombre Tarea ED04:



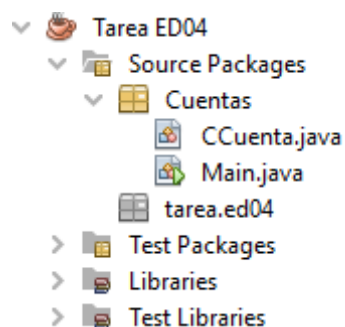
Una vez creado importamos las clases Main y CCuenta del archivo deposito.rar. Para mover la clase Main desde el paquete por defecto a uno denominado Cuentas, hacemos clic con el segundo botón del ratón sobre la clase, luego hacemos clic en Refactorizar > Mover:



Hacemos lo mismo con la clase CCuenta:

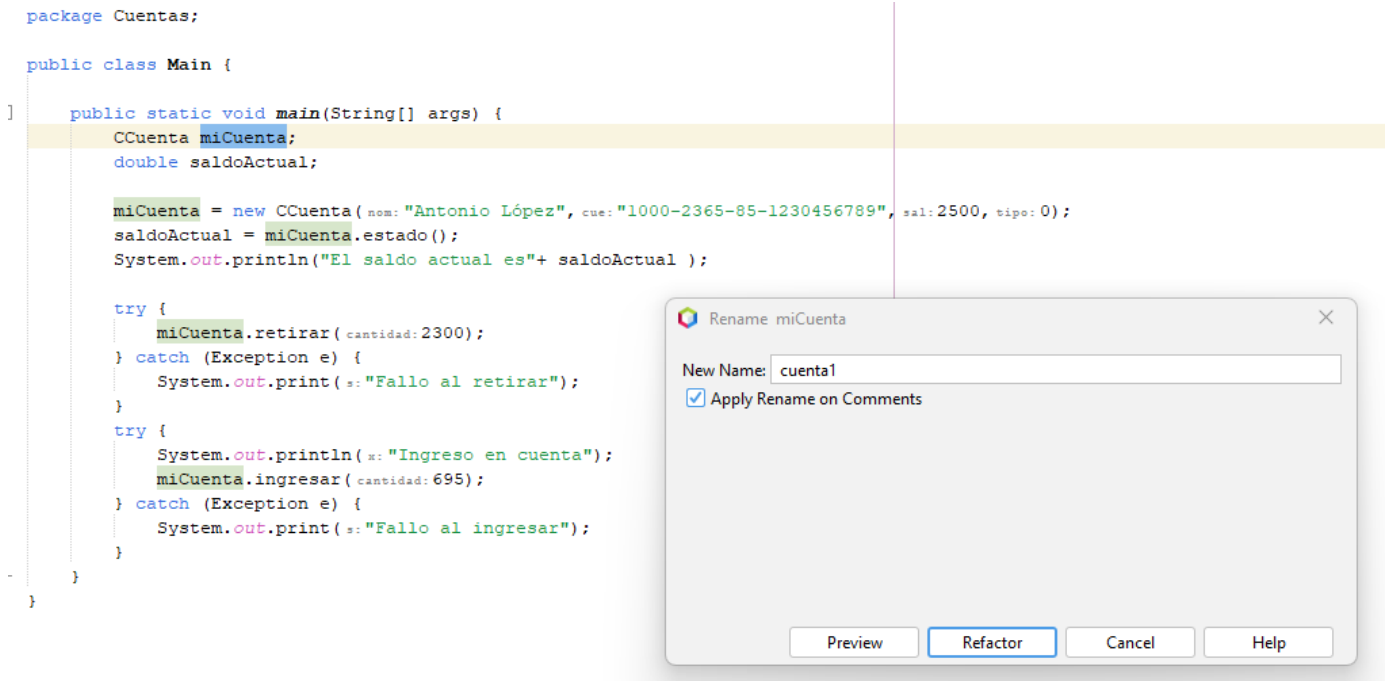


Las dos clases han sido movidas al paquete Cuentas. Esto implica que se actualicen todas las referencias a las clases en todo el código fuente:

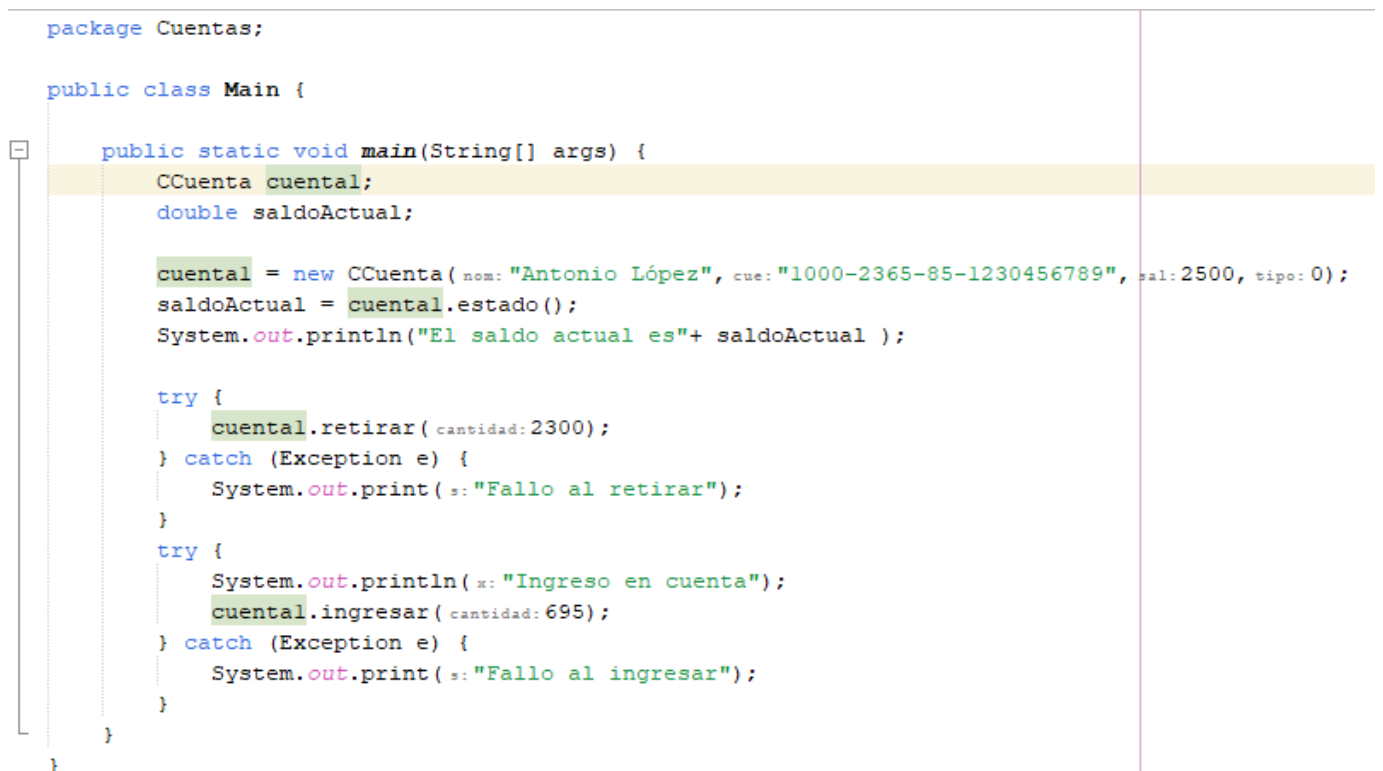


2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".

Para cambiar el nombre de la variable miCuenta por cuenta1, lo marcamos, botón derecho del ratón y Refactor > Rename:



Comprobamos que, donde antes decía miCuenta, ahora aparece cuenta1:



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

El objetivo es sustituir bloques de código por un método. De esta forma, cada vez que queramos acudir a ese bloque bastará con invocar al método. Para ello creamos, en el método `Main`, una sentencia llamada `operativa_cuenta()`;

```
package Cuentas;

public class Main {

    public static void main(String[] args) {

        operativa_cuenta();

        CCuenta cuental;
        double saldoActual;

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

        try {
            cuental.retirar( cantidad: 2300 );
        } catch (Exception e) {
            System.out.print( s: "Fallo al retirar" );
        }

        try {
            System.out.println( x: "Ingreso en cuenta" );
            cuental.ingresar( cantidad: 695 );
        } catch (Exception e) {
            System.out.print( s: "Fallo al ingresar" );
        }

    }

}
```

A continuación, seleccionamos la sentencia anterior, vamos al menú `Refactor > Introduce > Method` y creamos un método público de nombre `operativa_cuenta` y que engloba todas las sentencias de la clase `Main` que operan con el objeto `cuenta1`:

```

package Cuentas;

public class Main {

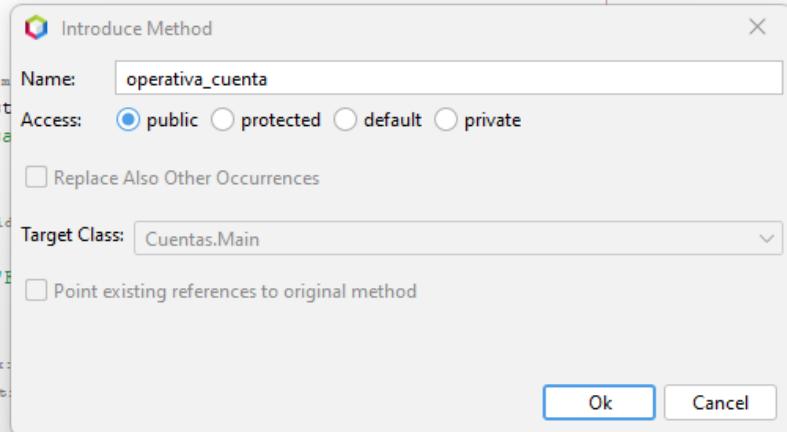
    public static void main(String[] args) {

        operativa_cuenta();
        CCuenta cuental;
        double saldoActual;

        cuental = new CCuenta( nom
        saldoActual = cuental.est
        System.out.println("El sa

        try {
            cuental.retirar( cantid
        } catch (Exception e) {
            System.out.print( s:"E
        }
        try {
            System.out.println( x:
            cuental.ingresar( cant
        } catch (Exception e) {
            System.out.print( s:"Fallo al ingresar");
        }
    }
}

```



Introduce Method

Name:

Access: ☒ public ☐ protected ☐ default ☐ private

☐ Replace Also Other Occurrences

Target Class:

☐ Point existing references to original method

Ok Cancel

```

package Cuentas;

public class Main {

    public static void main(String[] args) {
        operativa_cuenta();
    }

    public static void operativa_cuenta() {

        CCuenta cuental;
        double saldoActual;

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

        try {
            cuental.retirar( cantidad: 2300);
        } catch (Exception e) {
            System.out.print( s:"Fallo al retirar");
        }
        try {
            System.out.println( x: "Ingreso en cuenta");
            cuental.ingresar( cantidad: 695);
        } catch (Exception e) {
            System.out.print( s:"Fallo al ingresar");
        }
    }
}

```

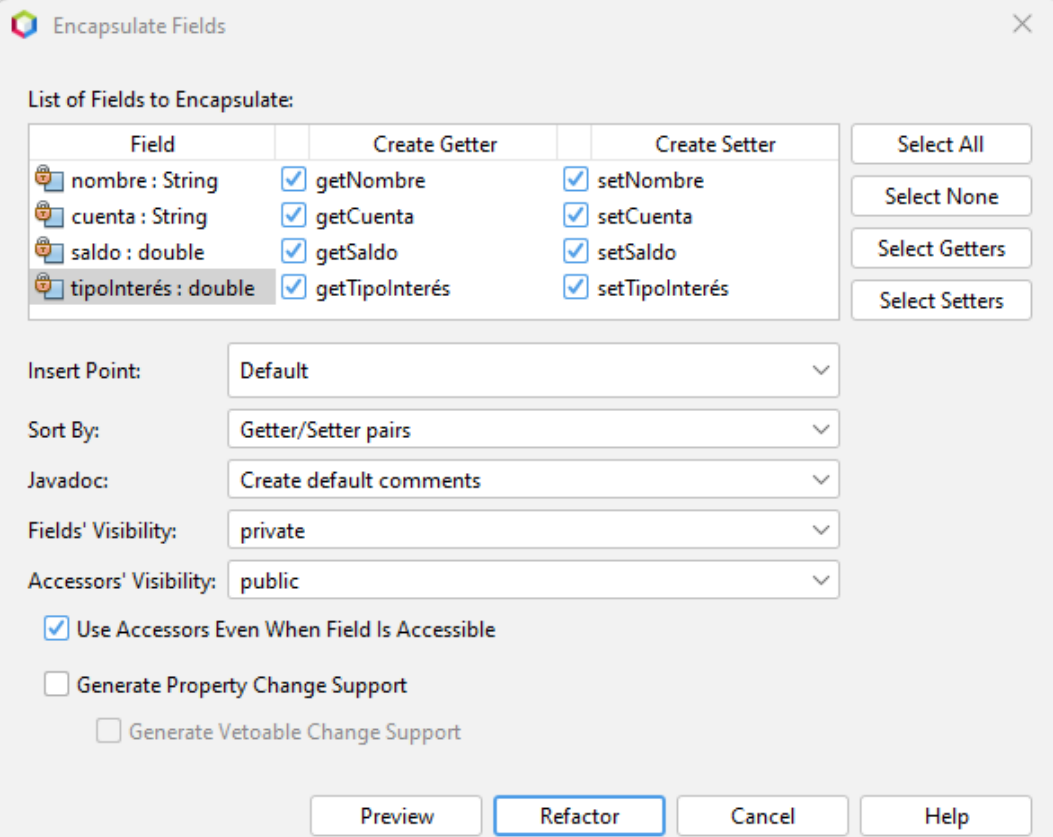
4. Encapsular los atributos de la clase CCuenta.

Abrimos la clase CCuenta. Seleccionamos todos los atributos, vamos a Refactor > Encapsulate Fields y los ordenamos por pares Getter/Setter:

```
package Cuentas;
```

```
public class CCuenta {
```

```
    private String nombre;  
    private String cuenta;  
    private double saldo;  
    private double tipoInterés;
```



Lo que estamos haciendo es crear métodos de asignación y de consulta (getters y setters) para los distintos campos o atributos de la clase:

```
package Cuentas;
```

```
public class CCuenta {
```

```
    /**
     * @return the nombre
     */
    public String getNombre() {
        return nombre;
    }
```

```
    /**
     * @param nombre the nombre to set
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```
    /**
     * @return the cuenta
     */
    public String getCuenta() {
        return cuenta;
    }
```

```
    /**
     * @param cuenta the cuenta to set
     */
    public void setCuenta(String cuenta) {
        this.cuenta = cuenta;
    }
```

```
    /**
     * @return the saldo
     */
    public double getSaldo() {
        return saldo;
    }
```

```
    /**
     * @param saldo the saldo to set
     */
    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }
```

```
    /**
     * @return the tipoInterés
     */
    public double getTipoInterés() {
        return tipoInterés;
    }
```

```
    /**
     * @param tipoInterés the tipoInterés to set
     */
    public void setTipoInterés(double tipoInterés) {
        this.tipoInterés = tipoInterés;
    }
```


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

Vamos a la clase Main y escribimos la sentencia cantidad;:

```
package Cuentas;

public class Main {

    public static void main(String[] args) {
        operativa_cuenta();
    }

    public static void operativa_cuenta() {

        CCuenta cuental;
        double saldoActual;
        cantidad;

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

        try {
            cuental.retirar(cantidad: 2300);
        } catch (Exception e) {
            System.out.print(s: "Fallo al retirar");
        }

        try {
            System.out.println(x: "Ingreso en cuenta");
            cuental.ingresar(cantidad: 695);
        } catch (Exception e) {
            System.out.print(s: "Fallo al ingresar");
        }

    }

}
```

Luego la marcamos, hacemos clic con el botón derecho del ratón en Refactor > Change Method Parameters. Esto permite añadir parámetros a un método y cambiar los modificadores de acceso. Seleccionamos tipo float, nombre cantidad y refactorizamos.

```

package Cuentas;

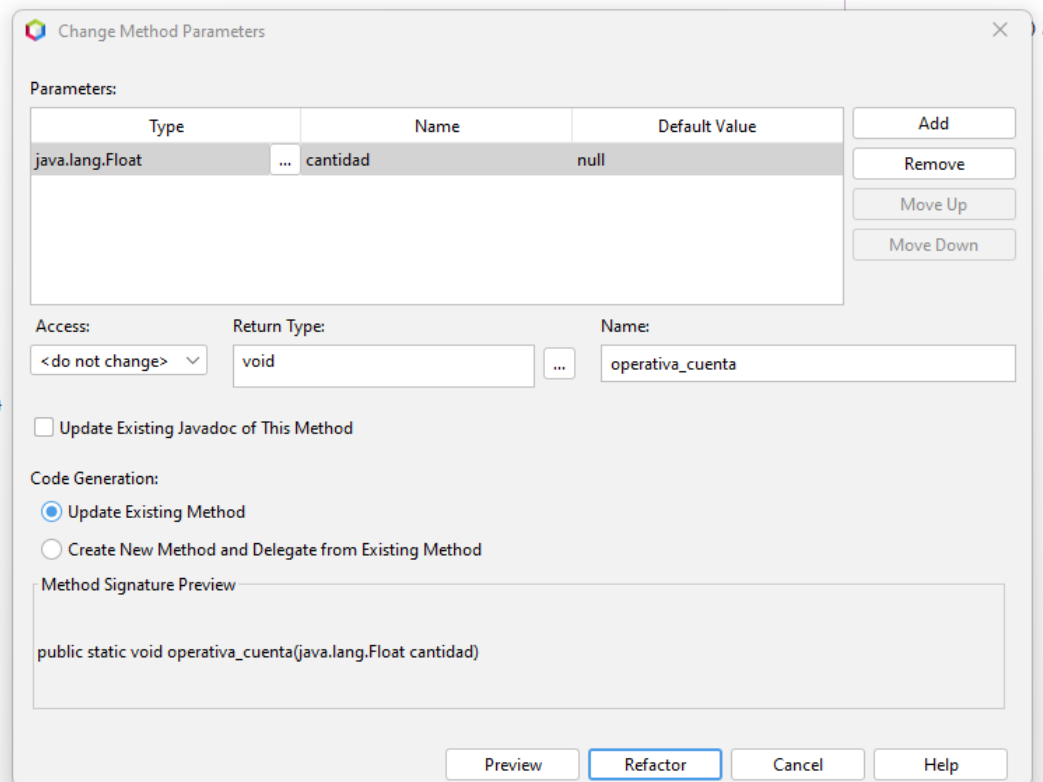
public class Main {

    public static void main(String[] args) {
        operativa_cuenta();
    }

    public static void operativa_cuenta() {

        CCuenta cuental;
        double saldoActual;
        cantidad;
    }
}

```



Finalmente, eliminamos la sentencia cantidad; escrita anteriormente:

```

package Cuentas;

public class Main {

    public static void main(String[] args) {
        operativa_cuenta( cantidad:null);
    }

    public static void operativa_cuenta(java.lang.Float cantidad) {

        CCuenta cuental;
        double saldoActual;

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

        try {
            cuental.retirar( cantidad: 2300);
        } catch (Exception e) {
            System.out.print( s: "Fallo al retirar");
        }
        try {
            System.out.println( x: "Ingreso en cuenta");
            cuental.ingresar( cantidad: 695);
        } catch (Exception e) {
            System.out.print( s: "Fallo al ingresar");
        }
    }
}

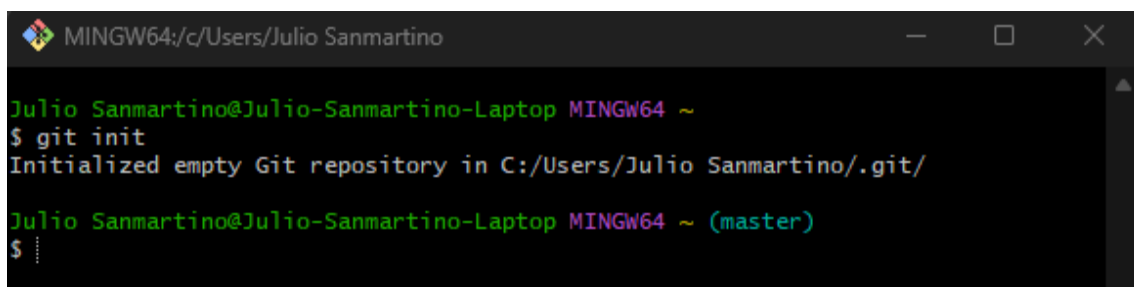
```

GIT

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

Vamos a crear un repositorio público en GitHub. En primer lugar, descargamos el software Git, en su versión 2.39.1-64-bit para Windows. Git es una herramienta de control de versiones muy popular por ser de código abierto y disponible para múltiples plataformas. Una vez instalado lo abrimos.

Para crear un nuevo repositorio, usaremos el comando `git init`. Al ejecutar este comando, se creará un repositorio principal vacío en nuestro directorio de trabajo actual:



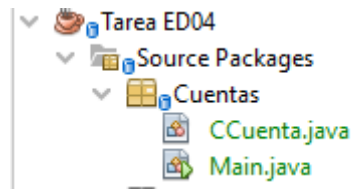
```

MINGW64/c/Users/Julio Sanmartino
Julio Sanmartino@Julio-Sanmartino-Laptop MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/Julio Sanmartino/.git/
Julio Sanmartino@Julio-Sanmartino-Laptop MINGW64 ~ (master)
$

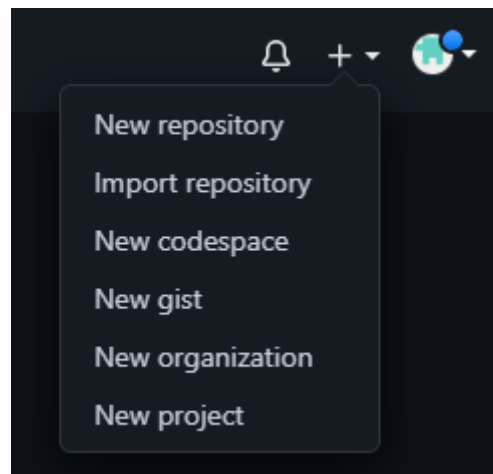
```

Otra opción es, desde NetBeans ir al menú Team > Git > Connect.

A continuación, en NetBeans podemos ver que aparecen una serie de pequeños iconos azules en el proyecto y paquetes. Además, los archivos que contiene el proyecto aparecen en verde, lo que indica que son archivos que aún no se han añadido al nuevo repositorio local Git.



De forma paralela, vamos a GitHub. Tras registrarnos, seleccionamos la opción New repository:




Le damos un nombre y lo hacemos público:

Create a new repository

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

Owner *

Repository name *

 JulioSanmartino ▾


/

ED ✓


Great repository names are short and memorable. Need inspiration? How about [animated-carnival?](#)

Description (optional)

Repositorio para Entornos de desarrollo

☒  **Public**

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

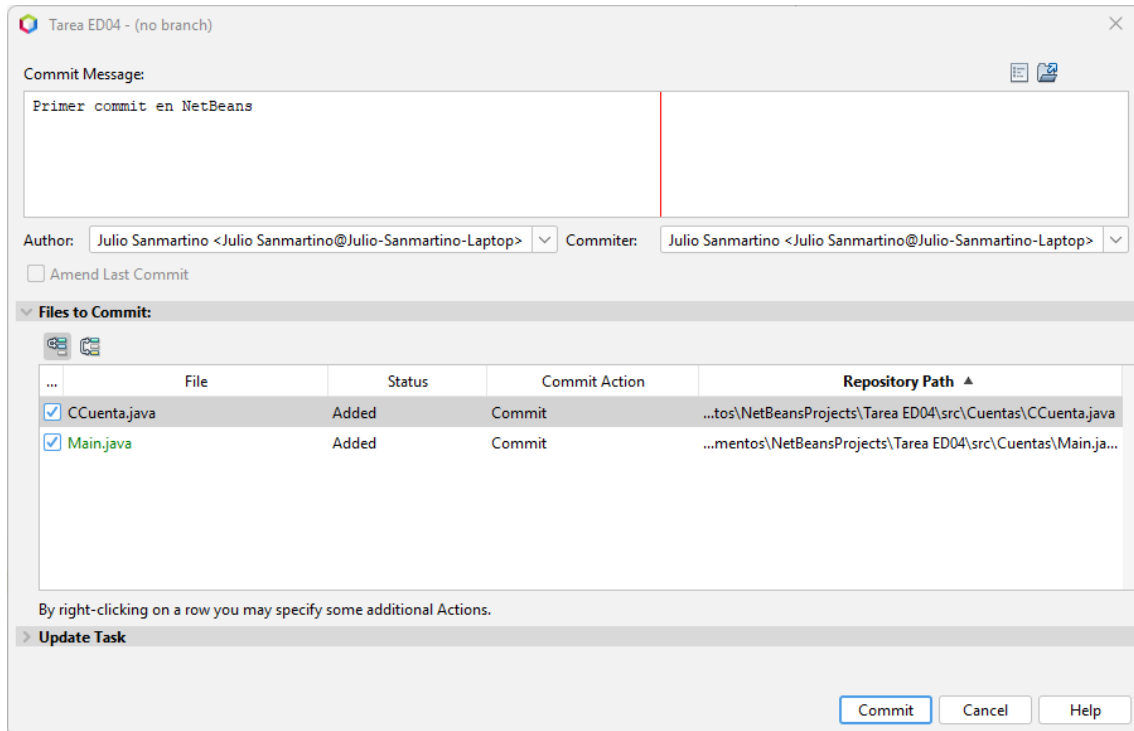
☐  **Private**

You choose who can see and commit to this repository.

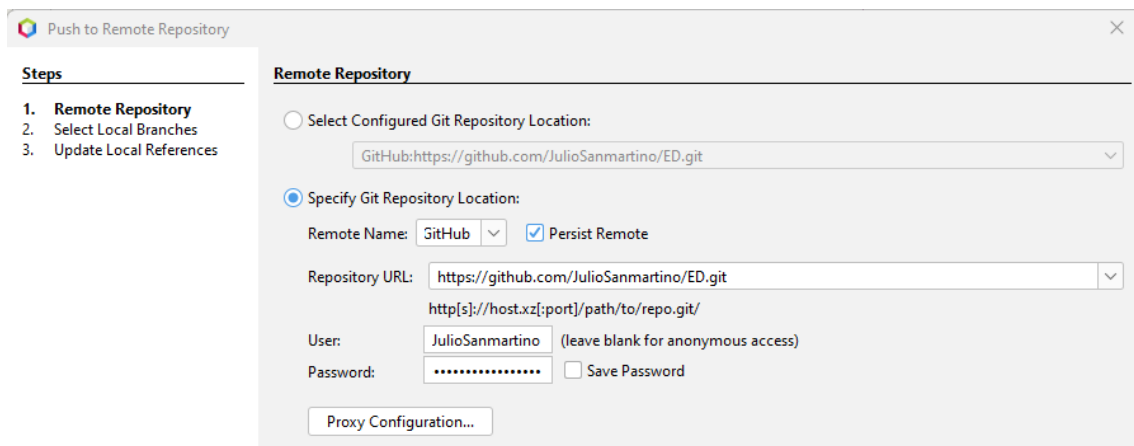
2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.

Un commit es un volcado de la información del área de trabajo al repositorio local. Previo al commit, habrá que seleccionar los ficheros del proyecto candidatos a formar parte del repositorio (Unstaged => Staged). Para ello, hacemos clic sobre ellos con el botón derecho del ratón Git > Add.

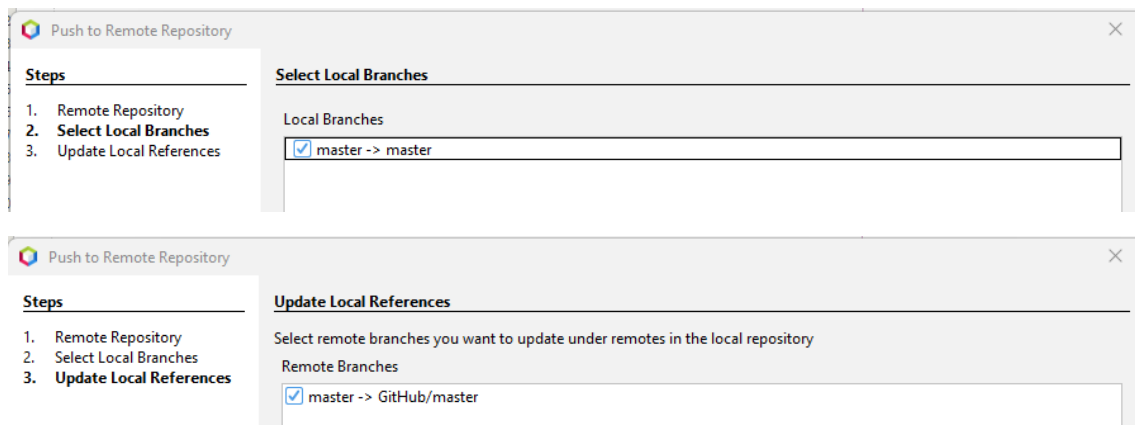
A continuación, seleccionamos el proyecto y vamos a menú Team > Commit:



Introducimos un mensaje y pulsamos el botón Commit. Para subir el proyecto al repositorio público creado en NetBeans lo seleccionamos y vamos al menú Team > Remote > Push. Completamos los campos indicando la URL del repositorio de GitHub:



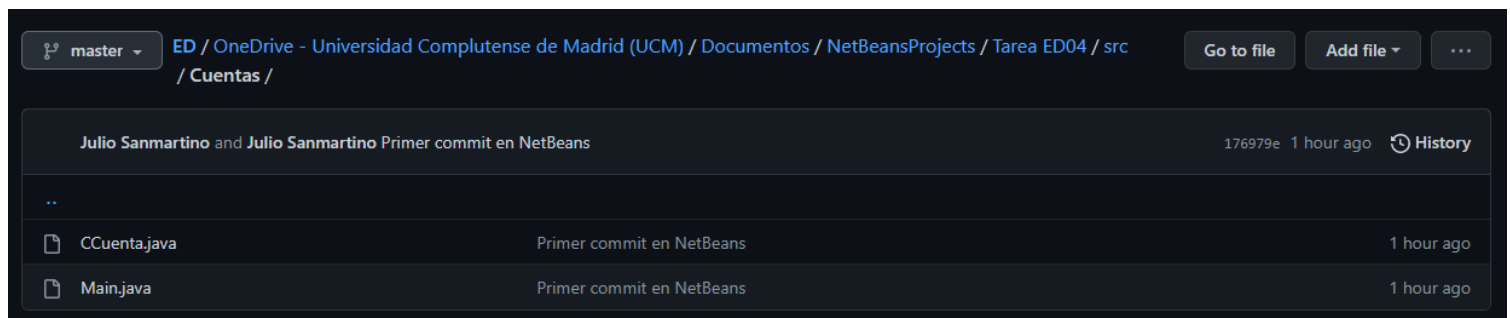
Seleccionamos la rama master:



Y finalmente hacemos clic en push. Otra forma de hacerlo es a través de la consola de Git, escribiendo `git push --set-upstream GitHub master` y permitiendo el acceso:

```
Julio Sanmartino@Julio-Sanmartino-Laptop MINGW64 ~ (master)
$ git push --set-upstream GitHub master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (10/10), 1.38 KiB | 141.00 KiB/s, done.
Total 10 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JulioSanmartino/ED.git
 * [new branch]      master -> master
branch 'master' set up to track 'GitHub/master'.
```

El resultado aparece en la página web, dentro de nuestro servidor público:



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

Simplemente escribimos `git log` y nos mostrará todos los commit realizados. En este caso sólo uno.

```
Julio Sanmartino@Julio-Sanmartino-Laptop MINGW64 ~ (master)
$ git log
commit 176979eec9b896a148bf97cfef043a35e0c98c09 (HEAD -> master)
Author: Julio Sanmartino <Julio Sanmartino@Julio-Sanmartino-Laptop>
Date: Thu Feb 9 12:14:15 2023 +0100

Primer commit en NetBeans
```

JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.

Al documentar una clase se debe incluir:

- a) Nombre de la clase, descripción general, número de versión, nombre de autores.
- b) Documentación de cada constructor o método (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

Insertamos los más comunes: autor, versión, return, param y exception/throws. Siempre con el siguiente formato: `/** comentario */`.

```

package Cuentas;

/**
 *
 * @author Julio Sanmartino
 * @version Primera versión 09/02/2023
 */
public class CCuenta {

    /**
     * @return the nombre. Describe el valor devuelto de un método.
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * @param nombre the nombre to set. Describe el parámetro del método
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    /**
     * @return the cuenta
     */
    public String getCuenta() {
        return cuenta;
    }

    /**
     * @param cuenta the cuenta to set
     */
    public void setCuenta(String cuenta) {
        this.cuenta = cuenta;
    }

    /**
     * @return the saldo
     */
    public double getSaldo() {
        return saldo;
    }
}

```



```

/**
 * @param saldo the saldo to set
 */
public void setSaldo(double saldo) {
    this.saldo = saldo;
}

/**
 * @return the tipoInterés
 */
public double getTipoInterés() {
    return tipoInterés;
}

/**
 * @param tipoInterés the tipoInterés to set
 */
public void setTipoInterés(double tipoInterés) {
    this.tipoInterés = tipoInterés;
}

private String nombre;
private String cuenta;
private double saldo;
private double tipoInterés;

public CCuenta()
{
}

public CCuenta(String nom, String cue, double sal, double tipo)
{
    nombre =nom;
    cuenta=cue;
    saldo=sal;
}

```

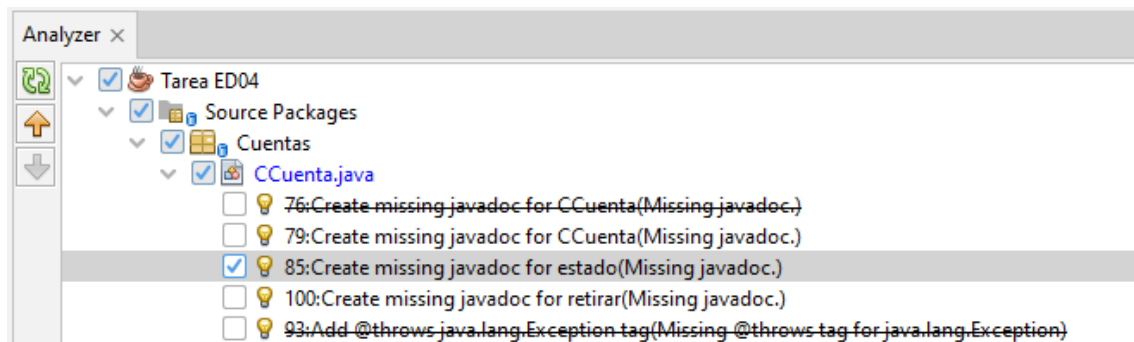
```

/**
 * @param cantidad: cantidad de dinero a ingresar en la cuenta
 * @throws Exception. Devuelve la excepción: No se puede ingresar una cantidad negativa cuando sea necesario
 */
public void ingresar(double cantidad) throws Exception {
    if (cantidad < 0) {
        throw new Exception( message: "No se puede ingresar una cantidad negativa");
    }
    setSaldo(getSaldo() + cantidad);
}

public void retirar(double cantidad) throws Exception {
    if (cantidad <= 0) {
        throw new Exception( message: "No se puede retirar una cantidad negativa");
    }
    if (estado() < cantidad) {
        throw new Exception( message: "No se hay suficiente saldo");
    }
    setSaldo(getSaldo() - cantidad);
}
}

```

Si seleccionamos la clase que queremos comentar y vamos al menú Tools > Analyze Javadoc. Nos mostrará todos aquellos métodos, parámetros y constructores que, según la herramienta faltarían sin comentar. Pudiendo incluirlos marcando la línea y pulsando el botón Fix:



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

Una vez insertados todos los comentarios Javadoc, seleccionamos el proyecto vamos al menú Run > Generate Javadoc.

Se crea la documentación de javadoc, que podemos visualizar online:

Nombre	Fecha de modificación	Tipo
legal	09/02/2023 16:13	Carpeta de archivos
resources	09/02/2023 16:13	Carpeta de archivos
script-dir	09/02/2023 16:13	Carpeta de archivos
allclasses-index	09/02/2023 16:13	Archivo de origen HTML
allpackages-index	09/02/2023 16:13	Archivo de origen HTML
copy	09/02/2023 16:13	Microsoft Edge HTML Document
element-list	09/02/2023 16:13	Archivo
help-doc	09/02/2023 16:13	Archivo de origen HTML
index	09/02/2023 16:13	Archivo de origen HTML
jquery-ui.overrides	09/02/2023 16:13	Archivo de origen CSS
member-search-index.js	09/02/2023 16:13	JSFile
module-search-index.js	09/02/2023 16:13	JSFile
overview-tree	09/02/2023 16:13	Archivo de origen HTML

All Classes and Interfaces

Classes	
Class	Description
CCuenta	
Main	

Package Cuentas

Class CCuenta

java.lang.Object[Ⓔ]
Cuentas.CCuenta

```
public class CCuenta  
extends ObjectⒺ
```

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
double	estado()		
String [Ⓔ]	getCuenta()		
String [Ⓔ]	getNombre()		
double	getSaldo()		
double	getTipoInterés()		
void	ingresar(double cantidad)		
void	retirar(double cantidad)		
void	setCuenta(String [Ⓔ] cuenta)		
void	setNombre(String [Ⓔ] nombre)		

Y que subimos al repositorio público de GitHub, cuya ruta es:

<https://github.com/JulioSanmartino/ED.git>