

TÍTULO DE LA PRÁCTICA

Nombre: Jesús David León Da Trindade

Curso: 1º de Ciclo Superior de Diseño de Aplicaciones Web.

ÍNDICE

- Introducción
- Objetivos
- Material empleado
- Desarrollo
- Conclusiones

Introducción.

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.

Objetivos.

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.

Material empleado.

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**.

Desarrollo.


























1. Las clases deberán formar parte del paquete cuentas.

New Java Package

Steps	Name and Location
1. Choose File Type	Package Name: <input type="text" value="Cuentas"/>
2. Name and Location	Project: <input type="text" value="Main"/>
	Location: <input type="text" value="Source Packages"/>
	Created Folder: <input type="text" value="ome/david/NetBeansProjects/Main/src/main/java/Cuentas"/>

< Back Next > **Finish** Cancel Help

Projects x

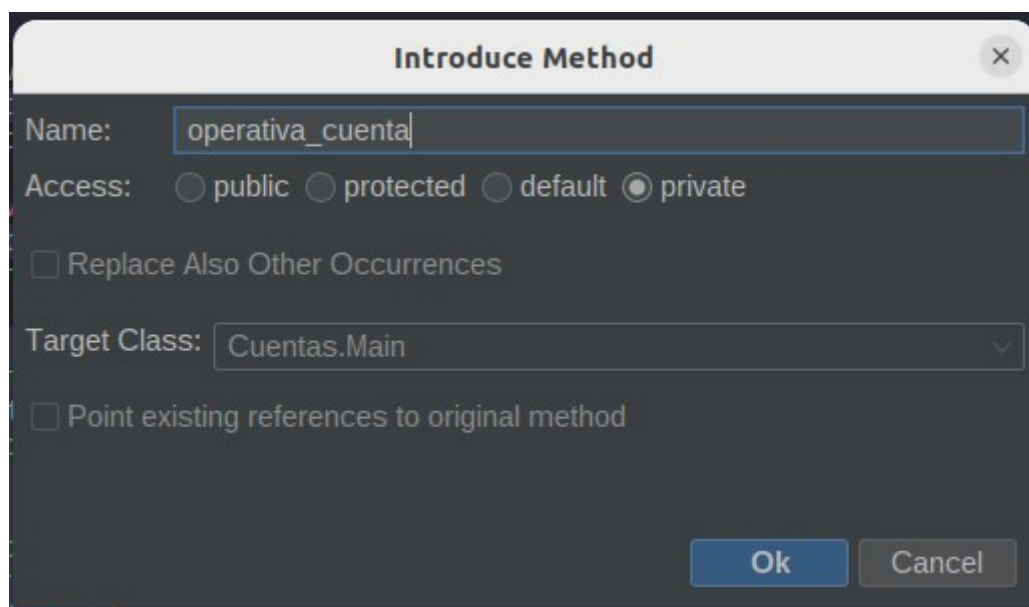
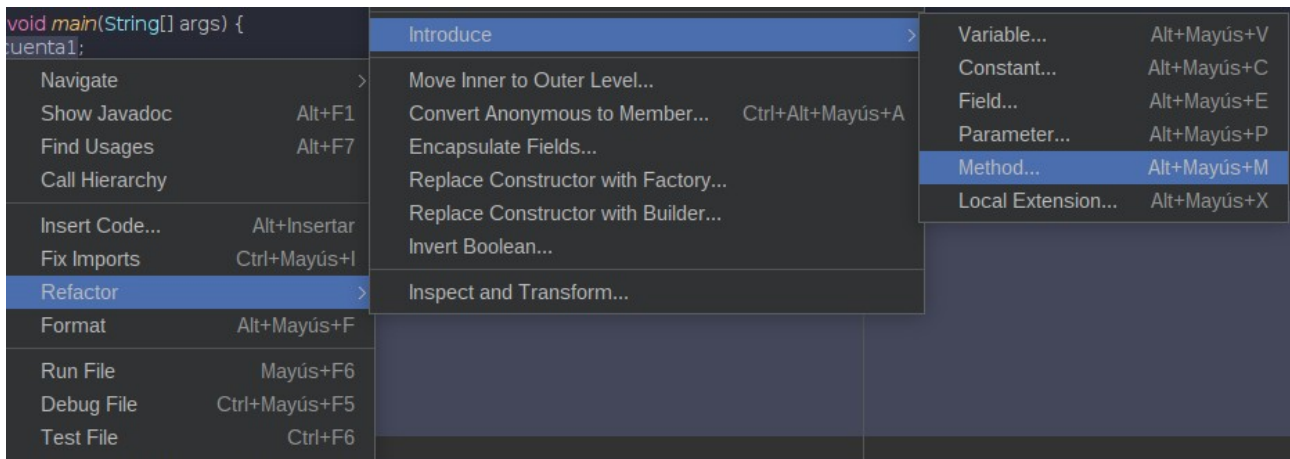
- >  ahorcado
- >  Calculadora
- >  EjercicioFicheroDDR2
- >  EjercicioFicheroDDR3
- >  EjercicioFicherosDDR
- >  EjercicioFicherosDDR1
- >  ExamenDiscoDuro
- >  ExamenMain
- >  ExamenMain3
- ✓  Main
 - ✓  Source Packages
 - ✓  Cuentas
 -  CCuenta.java
 -  Main.java
 - >  Test Packages
 - >  Dependencies
 - >  Java Dependencies
 - >  Project Files
 - >  MainExamen2
 - >  MainInstituto
 - >  RaicesDuroRoer
 - >  SimulacroAyo4_GPT
 - >  SimulacroExamen
 - >  SimulacroExamen1
 - >  TiendaEnLinea

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

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to cha
3  */
4
5  package Cuentas;
6
7  import Cuentas.CCuenta;
8
9  /**
10   *
11   * @author david
12   */
13  public class Main {
14
15      public static void main(String[] args) {
16          CCuenta cuenta1;
17          double saldoActual;
18
19          cuenta1 = new CCuenta(nom: "Antonio López", cue: "1000-2365-85-1230456789",
20          saldoActual = cuenta1.estado();
21          System.out.println("El saldo actual es"+ saldoActual );
22
23          try {
24              cuenta1.retirar(cantidad: 2300);
25          } catch (Exception e) {
26              System.out.print(s: "Fallo al retirar");
27          }
28
29          try {
30              System.out.println(x: "Ingreso en cuenta");
31              cuenta1.ingresar(cantidad: 695);
32          } catch (Exception e) {
33              System.out.print(s: "Fallo al ingresar");
34          }
35      }
36  }
```

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

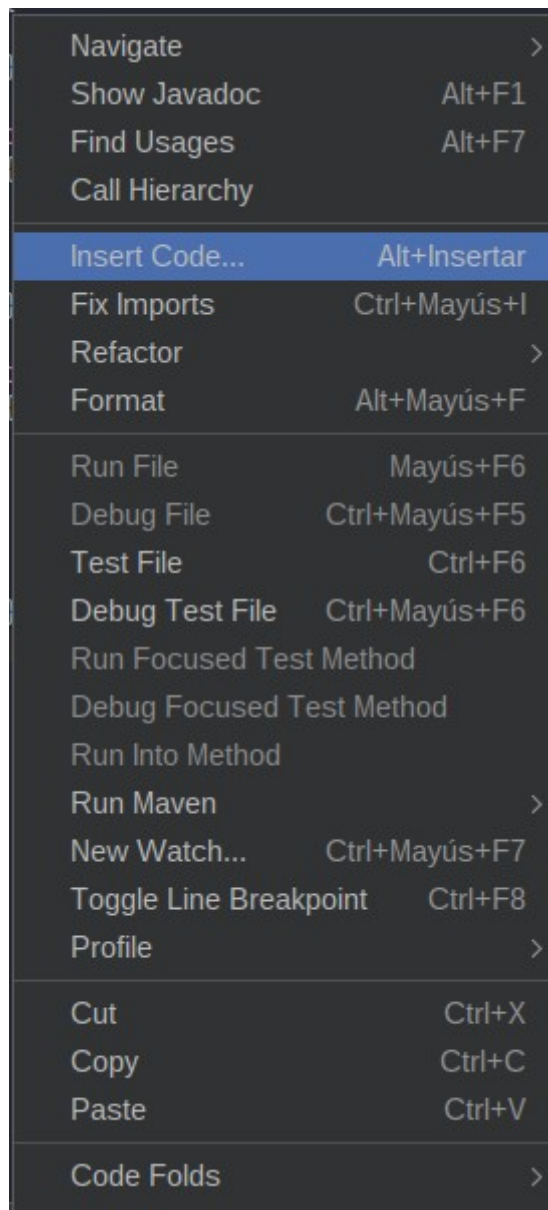
Para añadir el método `operativa_cuenta` tendremos que seleccionar los dos bloques del try-catch del código, botón derecho refactor/introduce/method y lo llamamos `operativa_cuenta`.



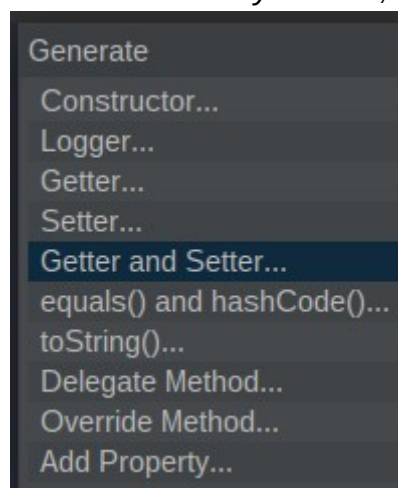
Una vez finalizado este proceso nos crea un método.

4.Encapsular los atributos de la clase CCuenta.

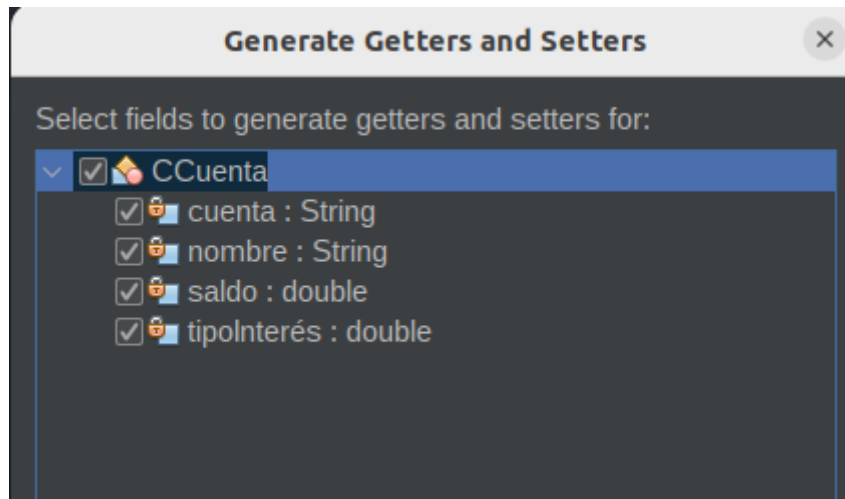
Para encapsular los atributos de la clase cuenta tendremos que dirigirnos a la clase cuenta dentro de netbeans, seguidamente iremos a insert code.



Dentro de este apartado seleccionamos Getters y Setters, como en la siguiente imagen.



Y por ultimo seleccionamos todos los atributos de la clase cuenta.



```
public String getNombre() {  
    return nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public String getCuenta() {  
    return cuenta;  
}  
  
public void setCuenta(String cuenta) {  
    this.cuenta = cuenta;  
}  
  
public double getSaldo() {  
    return saldo;  
}  
  
public void setSaldo(double saldo) {  
    this.saldo = saldo;  
}  
  
public double getTipointerés() {  
    return tipointerés;  
}  
  
public void setTipointerés(double tipointerés) {  
    this.tipointerés = tipointerés;  
}
```

De esta manera estamos encapsulando todos los atributos de la clase, ahora solo podremos acceder a los atributos de la clase por medio de los getters y setters.

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

```
*/
*/
public class Main {

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

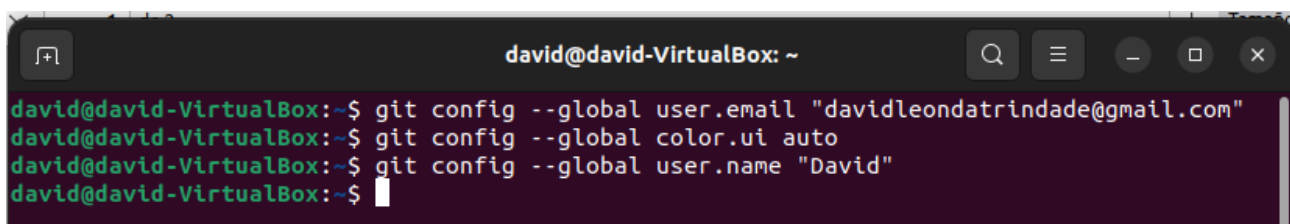
        cuenta1 = new CCuenta(nom: "Antonio López", cue: "1000-2365-85-1230456789", sal: 2500, tipo: 0);
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es" + saldoActual);

        operativa_cuenta(cuenta1, cantidad: 2.90f);
    }

    private static void operativa_cuenta(CCuenta cuenta1, float cantidad) {
        try {
            cuenta1.retirar(cantidad: 2300);
        } catch (Exception e) {
            System.out.print(s: "Fallo al retirar");
        }
        try {
            System.out.println(x: "Ingreso en cuenta");
            cuenta1.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.



```
david@david-VirtualBox: ~
david@david-VirtualBox:~$ git config --global user.email "davidleondatrindade@gmail.com"
david@david-VirtualBox:~$ git config --global color.ui auto
david@david-VirtualBox:~$ git config --global user.name "David"
david@david-VirtualBox:~$
```



For you Beta Following


<> Start writing code

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

portu628 /

☒  **Public**
Anyone on the internet can see this repository

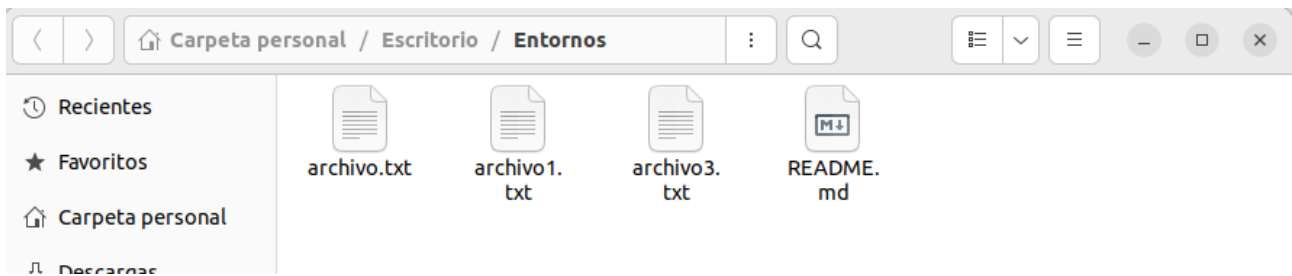
☐  **Private**
You choose who can see and commit to this repository

Create a new repository

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

Después de configurar nuestra clave publica, procedemos a clonar el repositorio de Github con el comando `git clone` y la dirección que nos sale en el repositorio, si todo ha ido bien obtendremos una imagen como la siguiente y nos aparecerá el repositorio en nuestro escritorio o en la carpeta que hayamos decidido en mi caso el escritorio para visualizarlo mejor.

```
 david@david-VirtualBox:~/Escritorio$ git clone git@github.com:portu628/Entornos.
git
Clonando en 'Entornos'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 1), reused 6 (delta 1), pack-reused 0
Recibiendo objetos: 100% (8/8), listo.
Resolviendo deltas: 100% (1/1), listo.
david@david-VirtualBox:~/Escritorio$
```



Tengo varios documentos por que he estado haciendo pruebas para entender el funcionamiento.

Una vez dentro del repositorio con un comando `cd Entornos` vamos a crear un archivo, por ejemplo `archivo4.txt`

```
david@david-VirtualBox:~/Escritorio$ cd Entornos
david@david-VirtualBox:~/Escritorio/Entornos$ touch archivo4.txt
david@david-VirtualBox:~/Escritorio/Entornos$ ls
archivo1.txt  archivo3.txt  archivo4.txt  archivo.txt  README.md
david@david-VirtualBox:~/Escritorio/Entornos$
```

Una vez echo esto vamos a lanzar el comando `git status` para que el propio programa nos muestre que la rama `main` a recibido cambios para poder seguir tenemos que usar el comando `git add` y el nombre del fichero creado o un `.` Para todos los ficheros creados nuevos.

```
git add para hacerlos siguientes)
david@david-VirtualBox:~/Escritorio/Entornos$ git add archivo4.txt
david@david-VirtualBox:~/Escritorio/Entornos$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
nuevos archivos: archivo4.txt

david@david-VirtualBox:~/Escritorio/Entornos$
```

Ahora el propio git nos dice que el archivo esta listo para confirmar los cambio para esto tendremos que usar el comando `git commit -m "aquí la descripcion"`.

```
david@david-VirtualBox:~/Escritorio/Entornos$ git commit -m "Creando archivo4"
[main 88b16a2] Creando archivo4
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 archivo4.txt
david@david-VirtualBox:~/Escritorio/Entornos$
```

```

david@david-VirtualBox:~/Escritorio/Entornos$ git log
commit 88b16a25e7509a56d2f19502371237a2c5b98854 (HEAD -> main)
Author: David <davidleondatrindade@gmail.com>
Date:   Wed May 10 10:26:13 2023 +0200

    Creando archivo4

commit 29d22cc1c26f603cca2deb13685ba6f2b2fd3055 (origin/main, origin/HEAD)
Author: David <davidleondatrindade@gmail.com>
Date:   Tue May 9 12:26:28 2023 +0100

    Creo el archivo3.txt

commit b1d3b7f1994c1c883b47a15b68b9009816d06e28
Author: portu628 <131296455+portu628@users.noreply.github.com>
Date:   Tue May 9 12:24:31 2023 +0100

    Create README.md

commit 253d8f04ba3837e7532537e9d09dc1aa7b305eef
Author: David <davidleondatrindade@gmail.com>
Date:   Tue May 9 12:19:28 2023 +0100

    Creando dos archivos
:...skipping...
commit 88b16a25e7509a56d2f19502371237a2c5b98854 (HEAD -> main)

```


Aquí tenemos un pequeño historial en terminal de la creación de archivos en git. Ahora vamos a hacer un push y actualizar nuestro repositorio con los cambios creados en local.

```


david@david-VirtualBox:~/Escritorio/Entornos$ git push git@github.com:portu628/Entornos.git
Enumerando objetos: 3, listo.
Contando objetos: 100% (3/3), listo.
Compresión delta usando hasta 16 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (2/2), 234 bytes | 234.00 KiB/s, listo.
Total 2 (delta 1), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:portu628/Entornos.git
   29d22cc..88b16a2  main -> main
david@david-VirtualBox:~/Escritorio/Entornos$

```


Ahora veremos que en la pagina web de github tambien nos aparecera el archivo creado en local archivo4.txt.



main




1 branch





0 tags






Go to file

Add file

 Code

 **portu628** Creando archivo4

88b16a2 7 minutes ago  4 commits

	README.md	Create README.md	yesterday
	archivo.txt	Creando dos archivos	yesterday
	archivo1.txt	Creando dos archivos	yesterday
	archivo3.txt	Creo el archivo3.txt	yesterday
	archivo4.txt	Creando archivo4	7 minutes ago

JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.

```
private double tipoInterés;

public CCuenta()
{
}

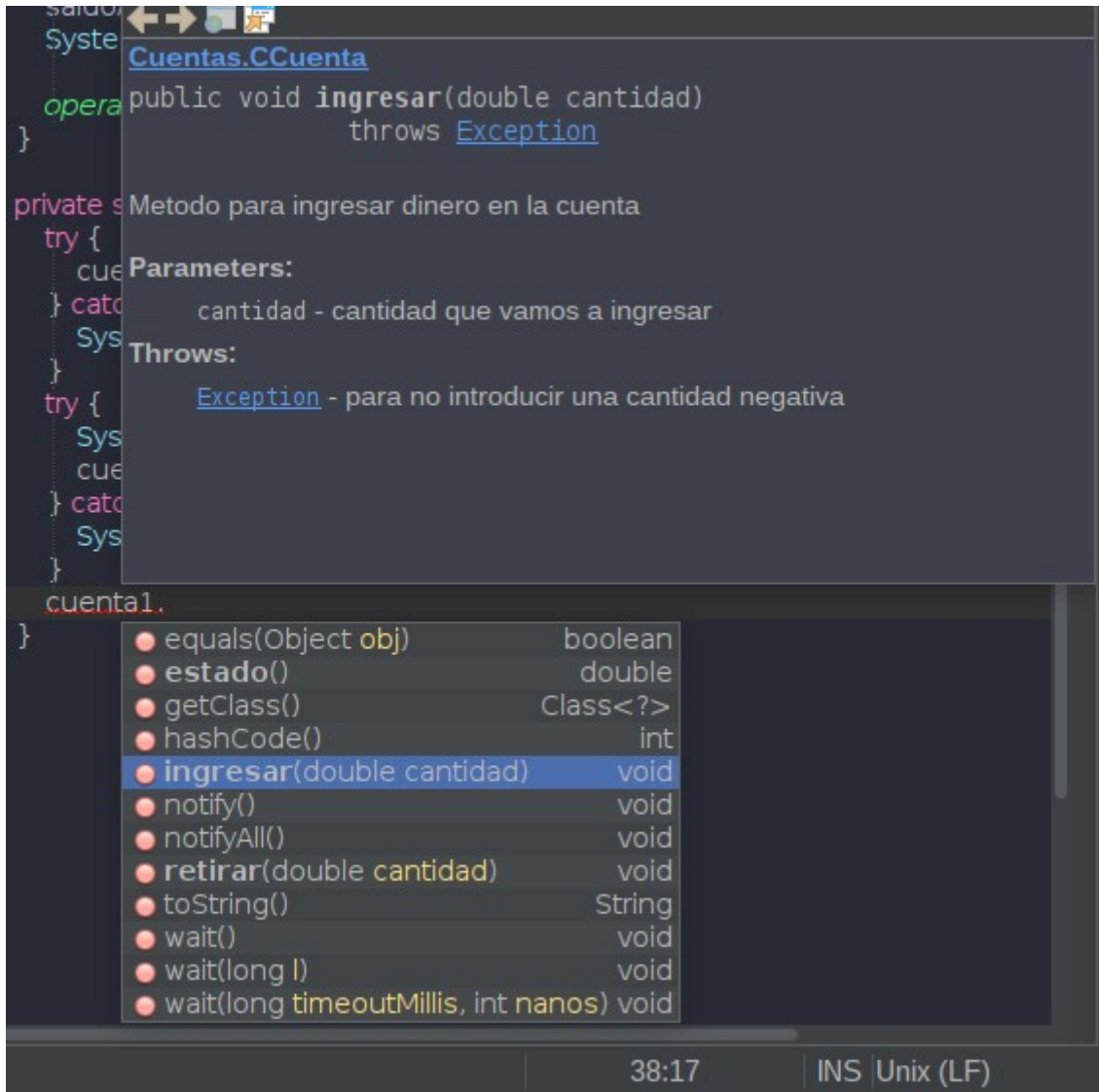
public CCuenta(String nom, String cue, double sal, double tipo)
{
    nombre = nom;
    cuenta = cue;
    saldo = sal;
}

/**Metodo que devuelve el saldo de la cuenta
 * @return <code>saldo</code> saldo disponible
 */
public double estado()
{
    return saldo;
}

/**Metodo para ingresar dinero en la cuenta
 * @param cantidad cantidad que vamos a ingresar
 * @throws Exception para no introducir una cantidad negativa
 */
public void ingresar(double cantidad) throws Exception
{
    if (cantidad < 0)
        throw new Exception("No se puede ingresar una cantidad negativa");
    saldo = saldo + cantidad;
}

/**Metodo para retirar dinero de la cuenta
 * @param cantidad cantidad que vamos a retirar
 * @throws Exception una excepcion para no introducir cantidades negativas
 * o retirar mas del dinero disponible en la cuenta.
 */
public void retirar(double cantidad) throws Exception
{
    if (cantidad <= 0)
        throw new Exception("No se puede retirar una cantidad negativa");
    if (estado() < cantidad)
        throw new Exception("No se hay suficiente saldo");
    saldo = saldo - cantidad;
}
}
```


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



Aquí tenemos la documentación creada en el método `ingresar` de la clase `Cuenta`.

Conclusiones.

Las conclusiones de esta práctica son muy interesantes, en la primera parte de la práctica realizamos varias acciones con el entorno de desarrollo NetBeans que desconocíamos hasta el momento. La segunda parte de Git personalmente me ha parecido más interesante, hemos aprendido comandos básicos para poder empezar a

usar git y github, en la tercera parte de la practica hemos introducido comentarios javadoc al código, acción sumamente importante para en un futuro trabajar en proyectos comunes y saber que hacen los métodos de otros programadores simplemente dejar bien documentado el texto.