FEB. 2018

# CONTORNOS DE DESENVOLVEMENTO

## TAREFA_04

### OPTIMIZACIÓN E DOCUMENTACIÓN_UD04

DAW 2017/18

ADRIÁN ÁLVAREZ LOIS

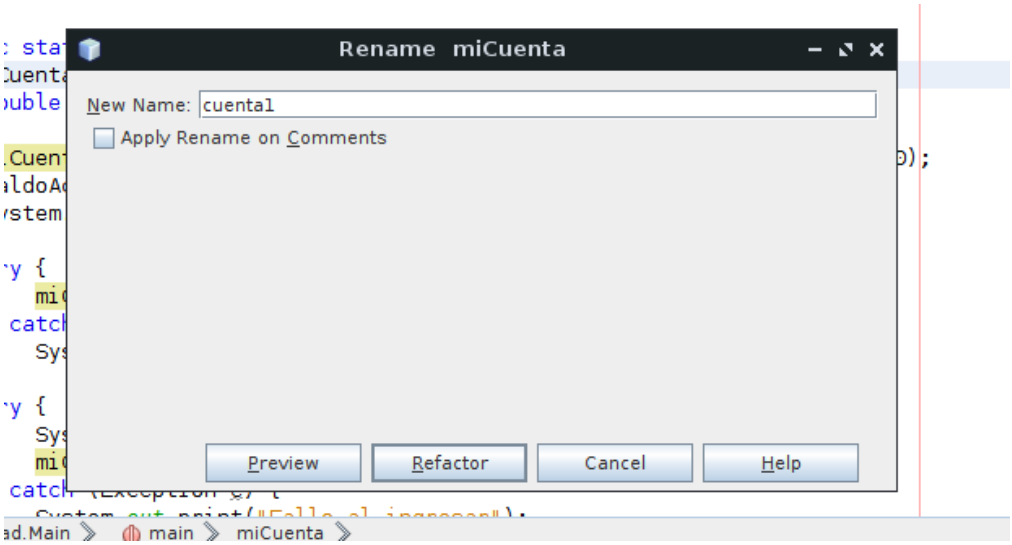
## REFACTORIZACIÓN

1. **Cambia el nombre de la variable " miCuenta " por " cuenta1 ".**

Para cambiar o nome da variable “miCuenta” situámonos no Main.java e seleccionamos a variable. Veremos como se marcan todas as variables “miCuenta” dentro do arquivo.

```
public class Main {

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

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

        try {
            miCuenta.retirar(2300);
        } catch (Exception e) {
            System.out.print("Fallo al retirar");
        }
        try {
            System.out.println("Ingreso en cuenta");
            miCuenta.ingresar(695);
        } catch (Exception e) {
```

Seguidamente botón dereito do rato no menú que se desprega buscamos Refactor -> Rename e mudamos o nome da variable a “cuenta1” e prememos no botón Refactor. Como non temos comentarios deixo sen marcar a opción Apply Rename on Comments.

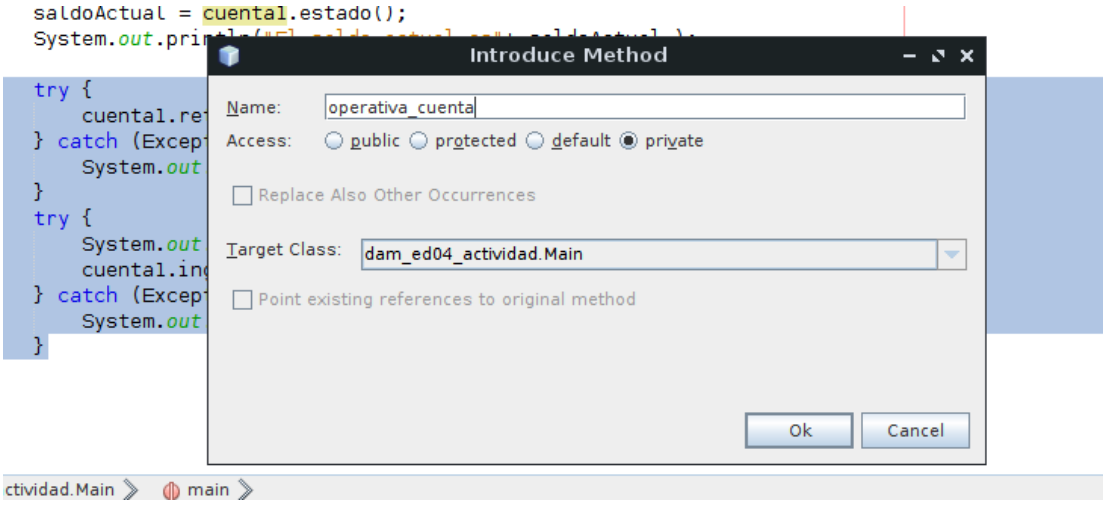


Agora podemos ver como o nome da variable mudou en todo o ficheiro.

```
public class Main {

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

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

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

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

Para introducir o método “operativa_cuenta” como se nos pide no enunciado temos que seleccionar todas as sentencias dentro da clase Main que operan co obxecto “cuenta1” é dicir todos os try-catch. Botón dereito do rato e no menú que se desprega seleccionamos Refactor -> Introduce -> Method e en nome introducimos “operativa_cuenta” e prememos OK.



O resultado da operación queda tal cual:

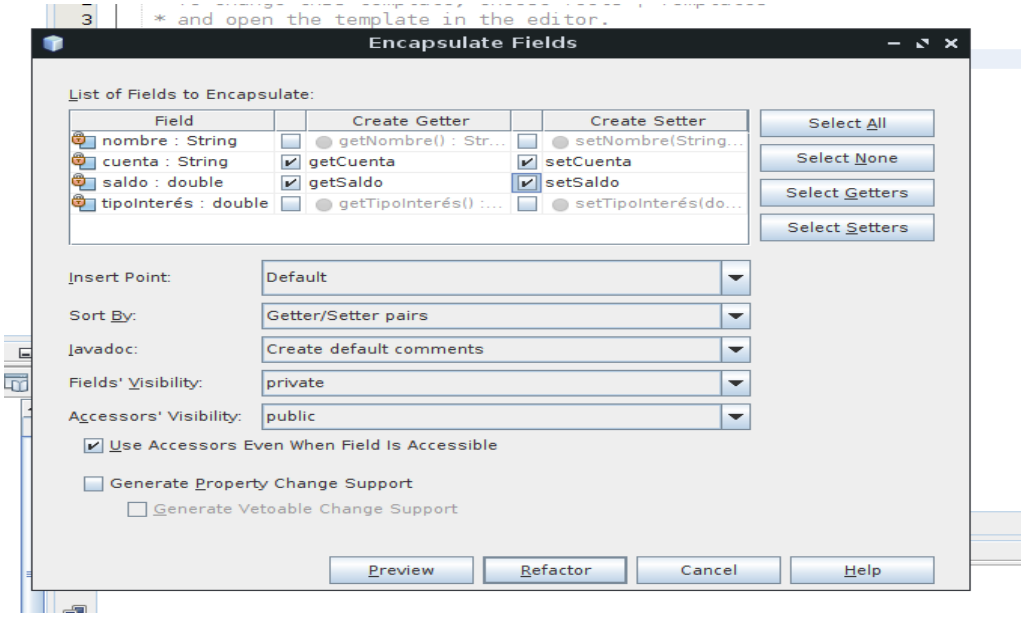
```
        System.out.println("El saldo actual es"+ saldoActual );

        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");
        }
    }
}
```

3. Encapsula los cuatro atributos de la clase CCuenta .

Para isto agora traballamos en Ccuenta.java. En materia de encapsulación de campos o que se aconsella é crear métodos getter e setter (de asignación e de consulta) para cada campo que se defins nunha clase. Cando sexa necesario acceder ou modificar o valor dun campo basta con invocar o método getter ou setter segundo nos conveña.

Os 4 atributos da clase CCuenta son nombre, cuenta, saldo e tipoInterés. De estos atributos só nombre e tipoInterés teñen declarados métodos getter e setter polo que encapsularemos os atributos cuenta e saldo.

Botón dereito Refactor -> Encapsule Fields



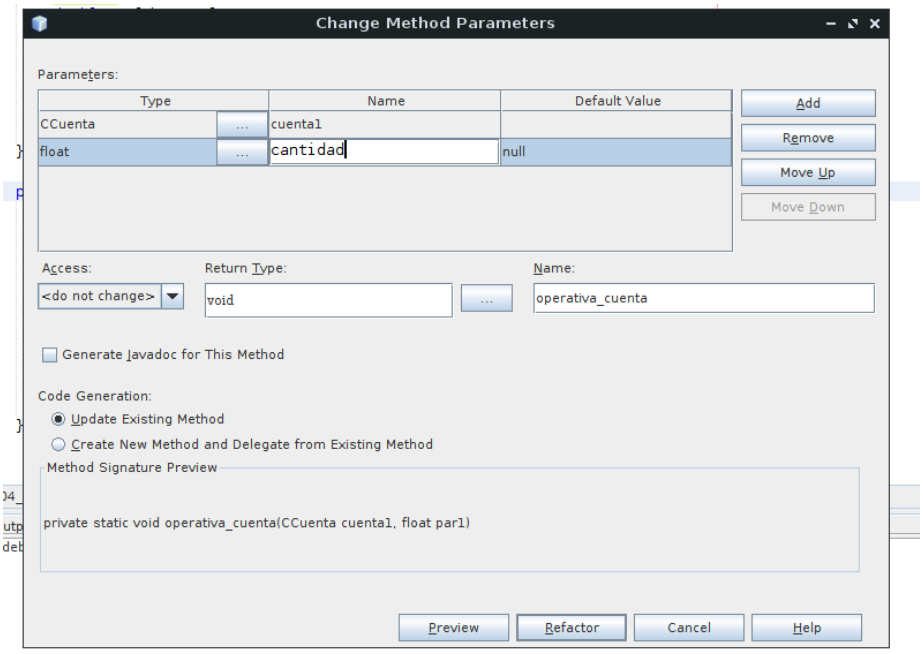
Marcamos os getters e setters que nos faltan por invocar. NetBeans pon un círculo a carón dos métodos xa declarados na clase. Prememos Refactor.

```
     * @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;
    }
```

Podemos ver como se crean os métodos dentro da clase.

**4. Añade un nuevo parámetro al método operativa_cuenta , de nombre cantidad y de tipo float .**

Volvemos a traballar sobre a clase Main.
O método operativa_cuenta só ten un parámetro que é o obxecto cuenta1 e imos inserir outro máis.
Situámonos dentro do método operativa_cuenta e clic no botón dereito e no menú seleccionamos Refactor -> Change Methods Parameters -> Add e enchemos os campos e clicamos en Refactor.



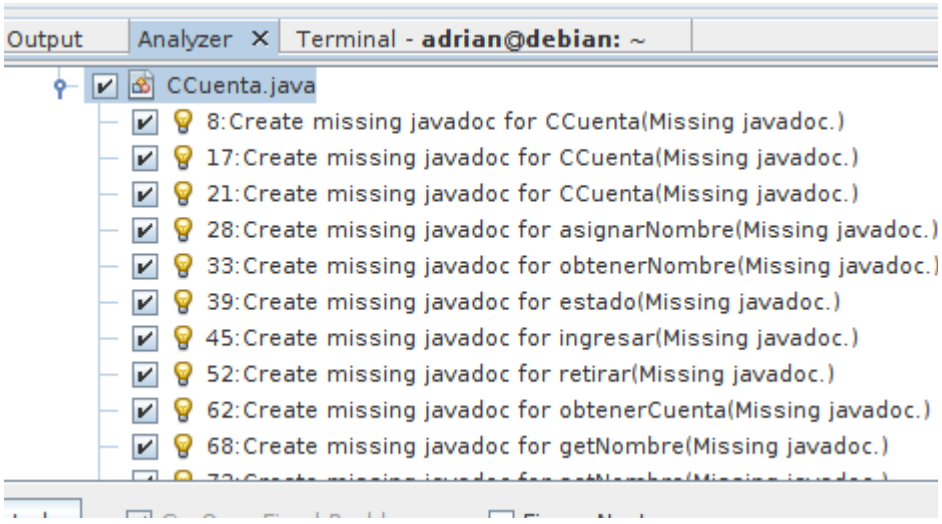
Agora vemos no código fonte como se engadiu o parámetro float cantidad ao método operativa_cuenta.

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

## JAVADOC

### 8. Inserta comentarios Javadoc en la clase CCuenta .

Na clase CCuenta seleccionamos no menú Tools -> Analyze Javadoc e veremos como se nos abre unha xanela chamada Analyzer. Nesta xanela marcamos o check Ccuenta.java para que seleccione todos os sitios onde queremos inserir comentarios e prememos o botón que di Fix Selected.



```
     * @param sal
     * @param tipo
     */
    public CCuenta(String nom, String cue, double
    {
        nombre =nom;
        cuenta=cue;
        saldo=sal;
    }

    /**
     *
     * @param nom
     */
    public void asignarNombre(String nom)
    {
        setNombre(nom);
    }

    /**
     *
     * @return
     */
d04_actividad.CCuenta
```

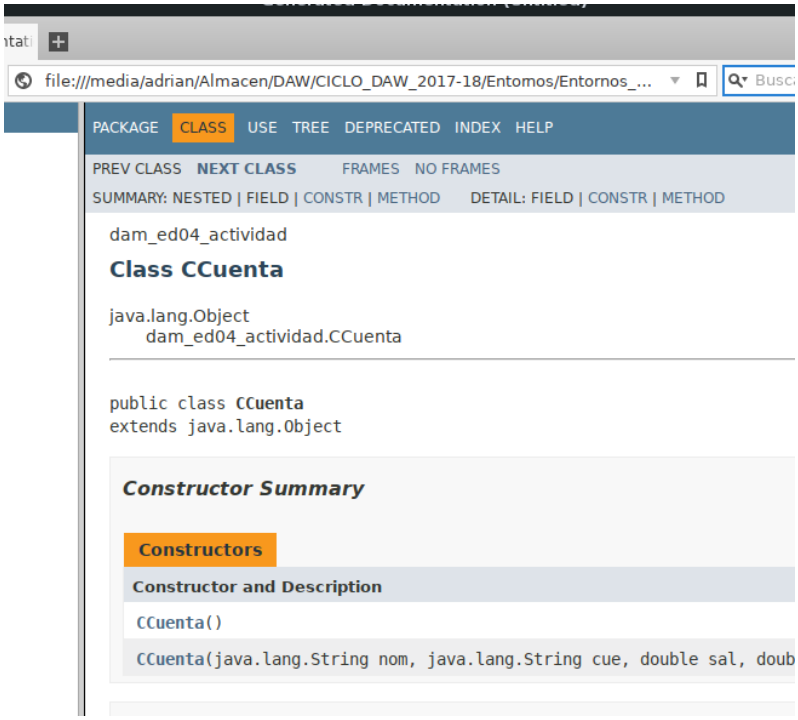
Unha vez feito isto poderemos ver no código fonte da clase os comentarios Javadoc que se caracterizan por comezar por /** e rematar por */ Estes comentarios fan referencia aos parámetros que usan os métodos, aos returns e ao author.

**9. Genera documentación Javadoc para todo el proyecto.**

No menú principal seleccionamos Run -> Generate Javadoc (proxecto).
Isto o que vai facer é abrir o navegador web onde se cargará unha páxina con toda a información relativa ao noso proxecto.
Esta páxina web será almacenada nun directorio chamado /dist/javadoc/ dentro da árbore do noso proxecto.

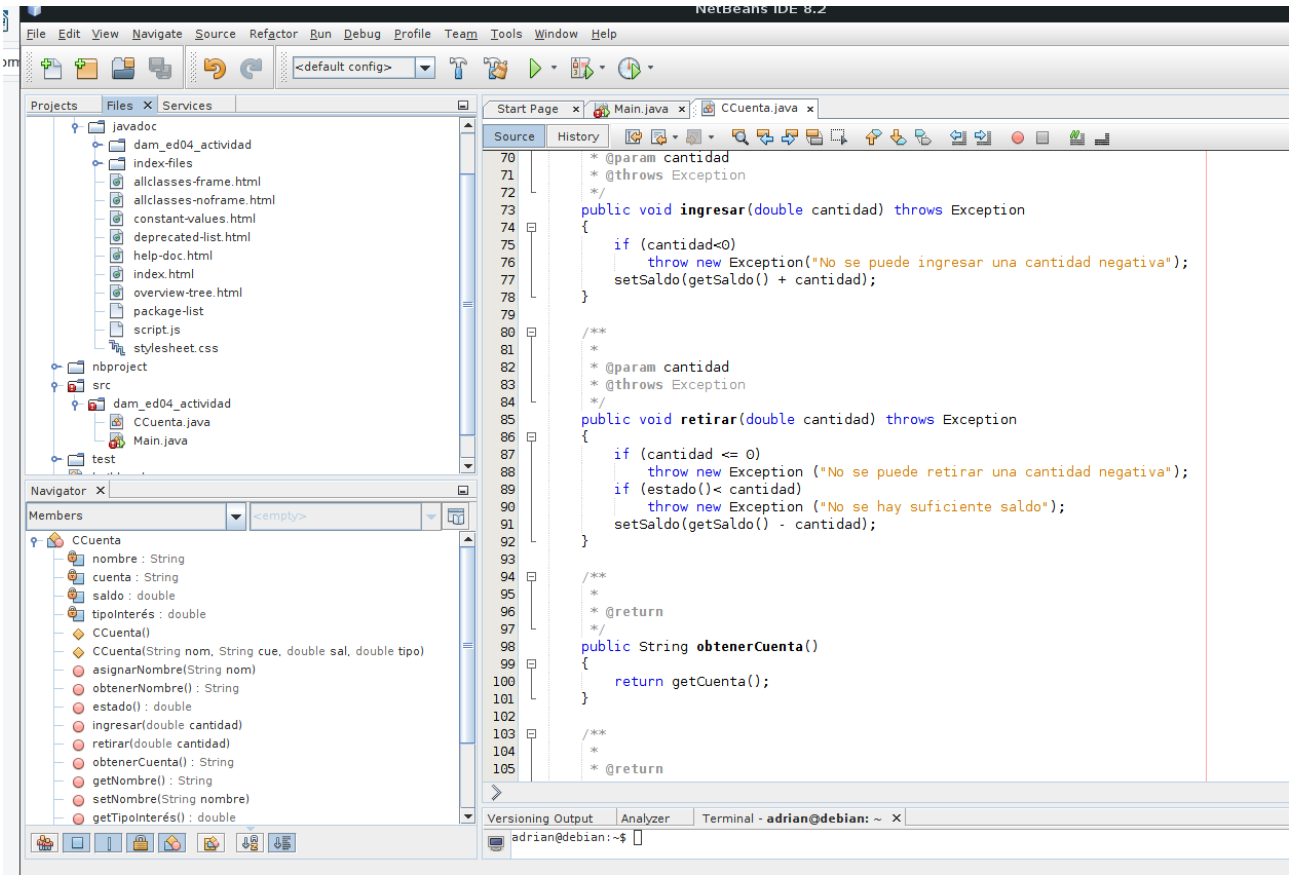
Nesta imaxe podemos ver a páxina web coa información do noso proxecto.

Tamén podemos ver os Javadoc se situamos o punteiro do mouse enriba dos diferentes elementos que empregamos no noso proxecto dende o navegador de NetBeans.



### 10. Comprueba que la documentación generada por Javadoc, abarca todos los métodos y atributos de la clase CCuenta .

Para comprobar que a documentación xerada por Javadoc abarca todos os métodos e atributos da clase Ccuenta optei por percorrer o código fonte no NetBeans e asegurarme que se incluían comentarios antes dos métodos e atributos onde aparecen etiquetas como @param, @return, @throws exception, @author...



## CVS

**5. Configura CVS para el proyecto DAW_ED04_Actividad. Crea un nuevo repositorio (deposito) en caso de no disponer de él.**

Para realizar este exercicio seguín un tutorial. Non me resultou doado. Traballaremos en Debian GNU/Linux dende a consola.

1/ Instalo o paquete cvs:
```
$ sudo apt-get install cvs
```

2/ Creo un grupo chamado cvsadmin:
```
$ sudo addgroup cvsadmin
```

3/ Engado o meu usuario (adrian) ao grupo cvsadmin:
```
$ sudo adduser adrian csadmin
```

4/ Creo un repositorio en local e indicamos onde se van colocar as variables de entorno.
```
$ export CVSROOT=/var/local/cvs
```

5/ Creamos unha carpeta CVSROOT no noso repositorio:
```
$ sudo mkdir -p $CVSROOT
```

6/ Damos permisos ao grupo cvsadmin e ao usuario adrian para poder traballar con cvs:
```
$ cd /var/local
$ sudo chown adrian cvs/
$ sudo chgrp cvsadmin cvs/
```

7/ Seguidamente vamos a gardar as variables de entorno para que manteñan o usuario editando o arquivo .profile que está no home:
```
$ sudo gedit ~/.profile
```
engadimos ao final do arquivo a seguinte sentencia: `export CVSROOT=”/var/local/cvs”`
gardamos e pechamos o editor de texto.

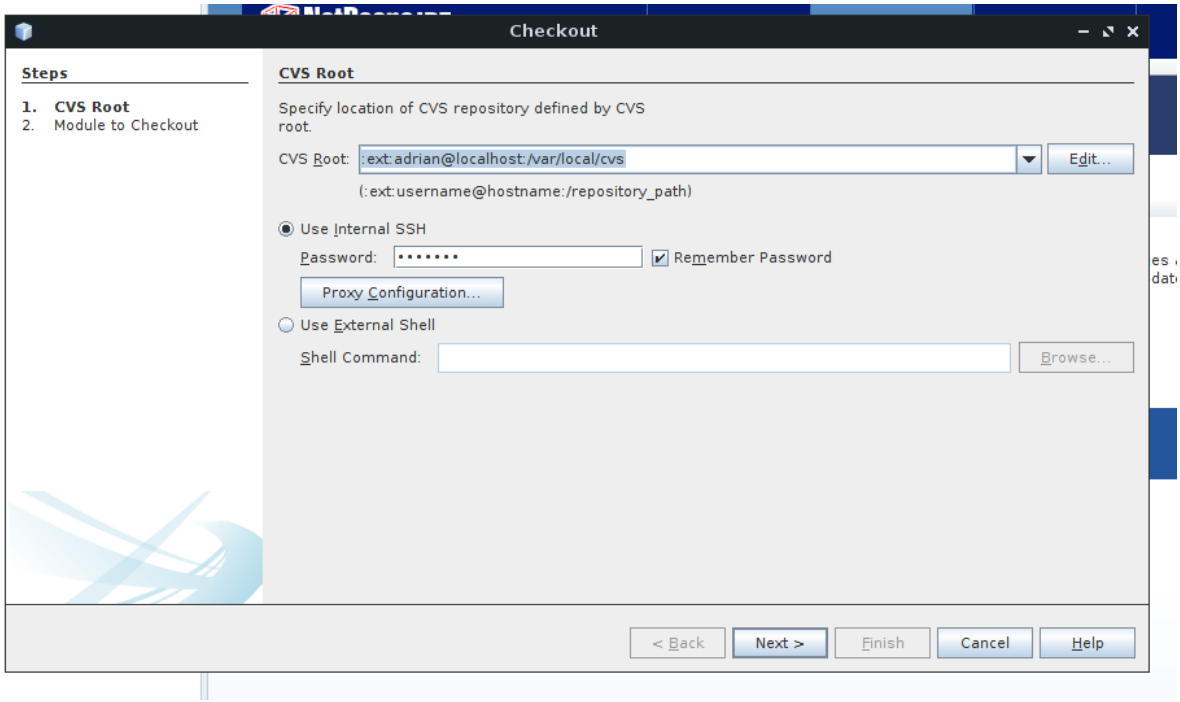
8/ Iniciamos o repositorio:
```
$ cvs init
```

9/ Agora imos crear o noso repositorio para o proxecto da tarefa. Eu por comodidade optei por renomear a carpeta DAW_ED04_Actividad como tarefa04:
navegamos até a carpeta onde se atopa o proxecto.
```
$ cd /home/adrian/Documentos/tarefa04
```
creamos o repo para tarefa04
```
$ cvs import -m "Importando Tarefa04" rep_tarefa04 tarefa04 inicio
```

10/ Instalamos un servidor ssh:
```
$ sudo apt install openssh-server ssh-import-id
```

11/ Con todo esto feito é hora de ir a NetBeans:

No menú principal seleccionamos Team -> CVS -> Checkout

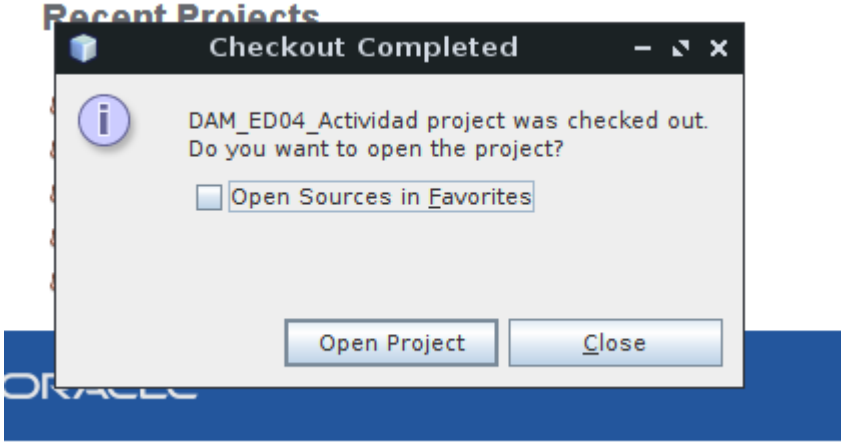



Introducimos a localización do repositorio definido con CVSROOT. O método de conexión a CVSROOT que vou empregar é ext usando o Secure Shell (SSH) polo que a localización quedaría:

```
:ext:adrian@localhost:/var/local/cvs
```

Iintroducimos tamén o meu contrasinal e prememos en Next.

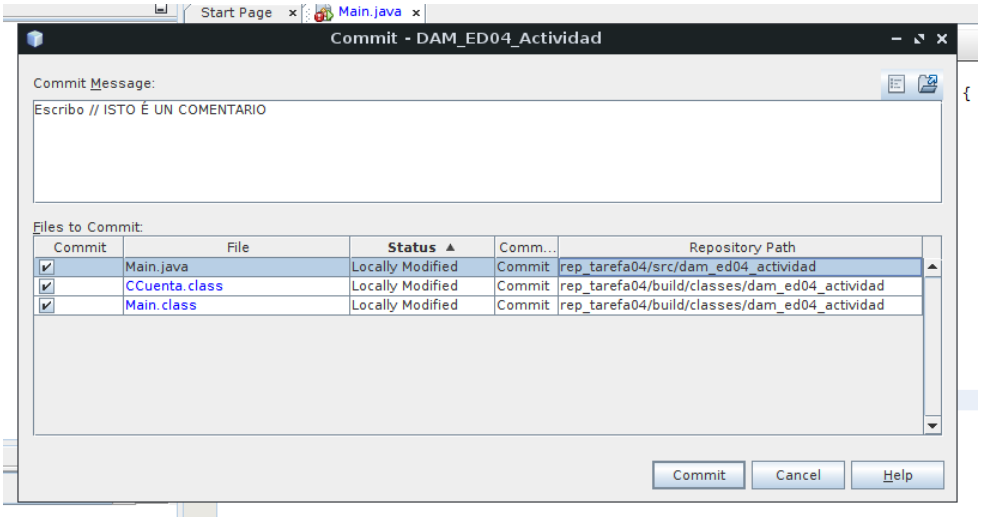
Despois pídenos que especifiquemos os Módulos, Branches e o directorio local de traballo. En módulos selecciono o proxecto dende o repo e como directorio local de traballo creo unha carpeta chamada working_folder.




O Checkout realizase con éxito. O proxecto está cargado e listo para traballar con el e subir cambios ao repositorio.

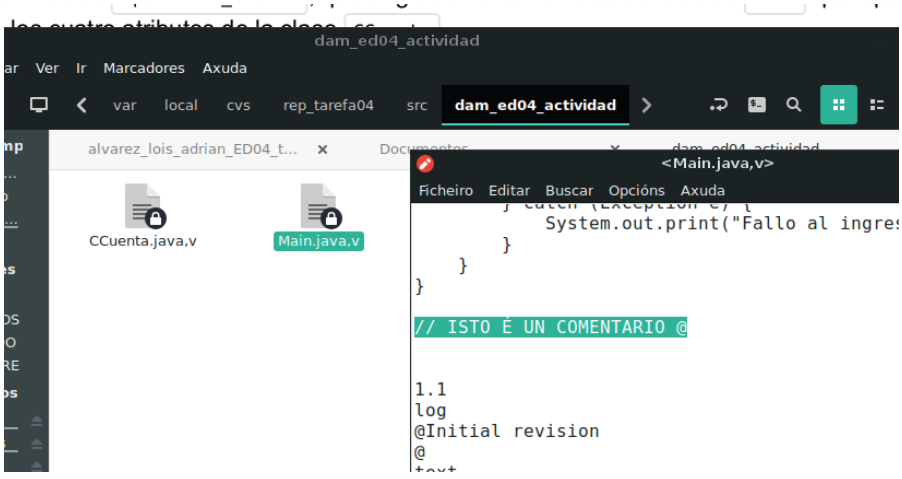
**6. Realiza, al menos, una operación commit , comentando el resultado de la ejecución.**

```
29                System.out.println("Ingreso en cuenta");
30                cuenta1.ingresar(695);
31            } catch (Exception e) {
32                System.out.print("Fallo al ingresar");
33            }
34        }
35    }
36
37    // ISTO É UN COMENTARIO
```




Na clase Main introduzo un pequeno cambio no arquivo na forma de comentario ao final do código. Despois salvo os cambios e fago un commit indo no menú principal a Team -> Commit

Para comprobar que o commit se fixo con éxito vou ao repositorio /var/local/cvs... e abro o ficheiro Main.java que se garda no repo cun editor de textos e vemos como efectivamente o comentario foi engadido.



### 7. Muestra el historial del versiones para el fichero Main.java y comenta el resultado.

No meú principal abro Team -> History -> Show History