

1

New Java Project

Create a Java Project

Discouraged module name. By convention, module names usually start with a lowercase letter

Project name: Deposito

☒ Use default location

Location: C:\Users\Hugo JB\eclipse-workspace\Deposito [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-21 [Configure JREs...](#)

☐ Use a project specific JRE: jdk-21

☐ Use default JRE 'jdk-21' and workspace compiler preferences

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

Module

☒ Create module-info.java file

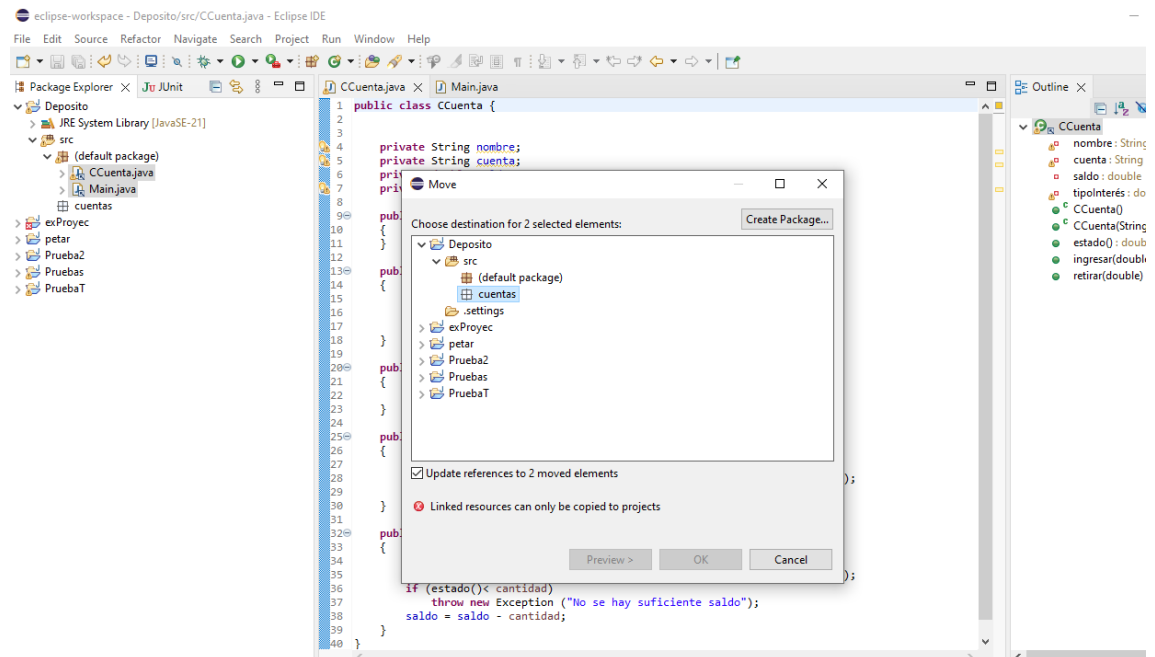
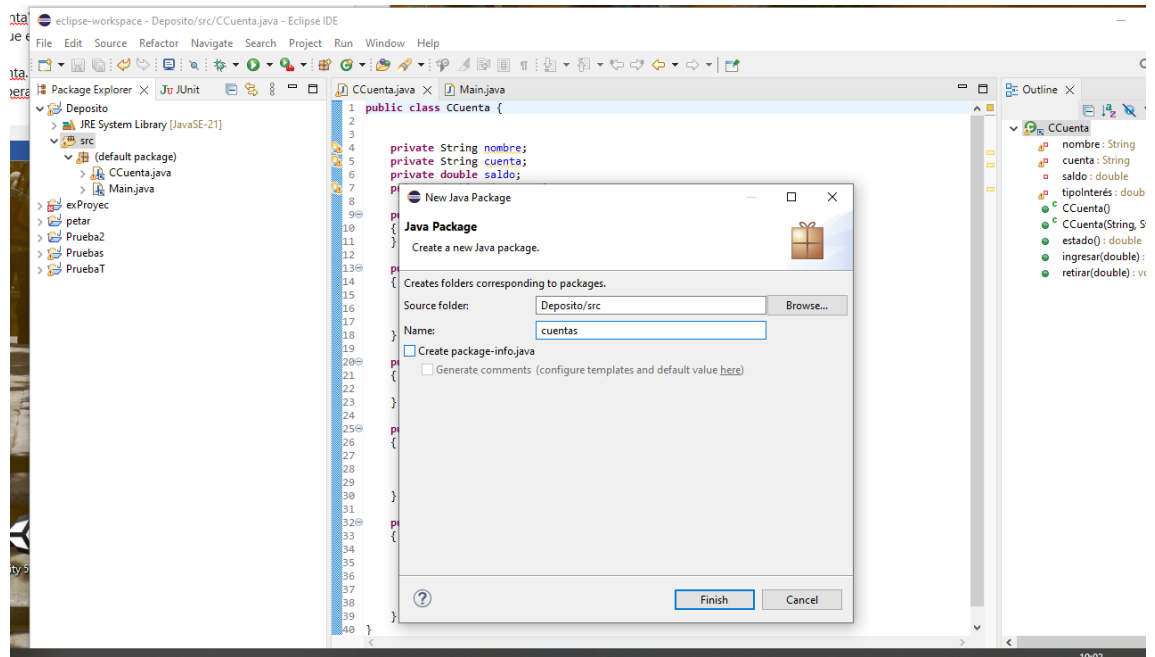
Module name:

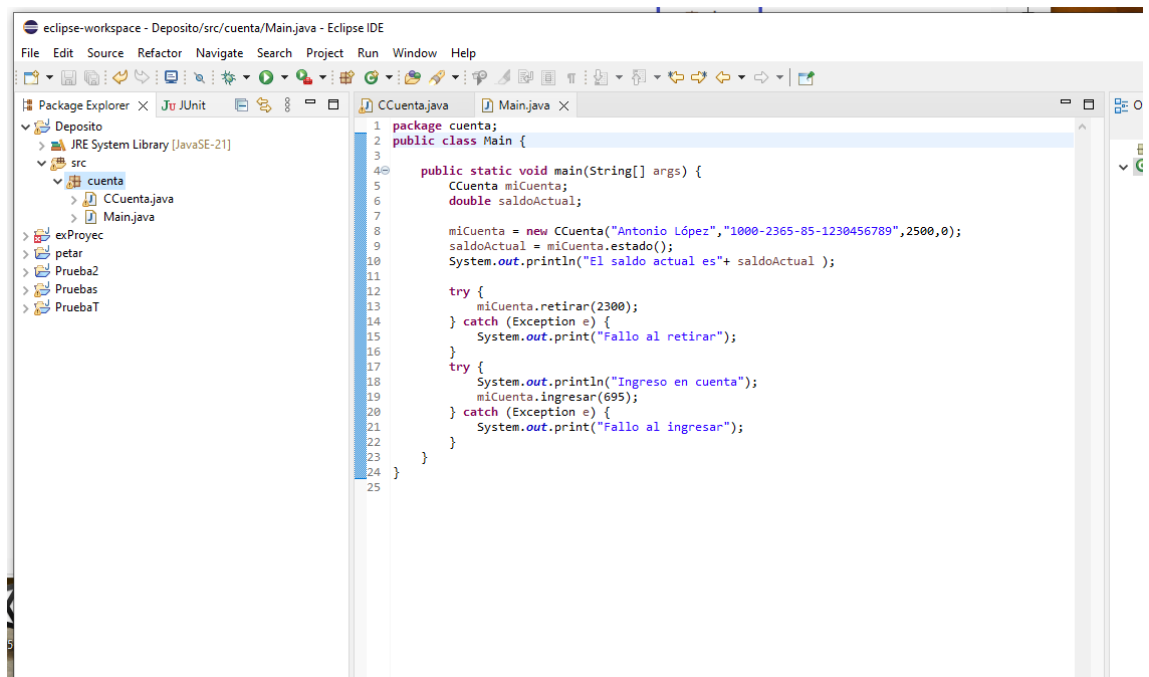
☒ Generate comments

module name will be "Deposito" (if no module is specified, then project name will be used as module name)

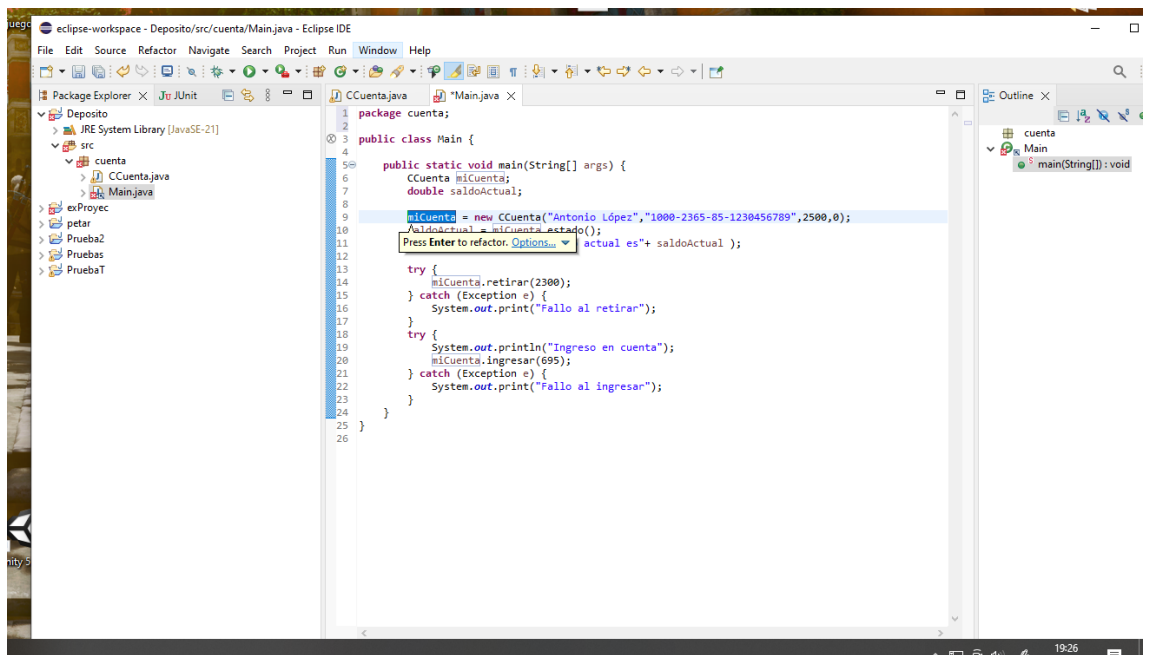
[?](#) [< Back](#) [Next >](#) **Finish** [Cancel](#)

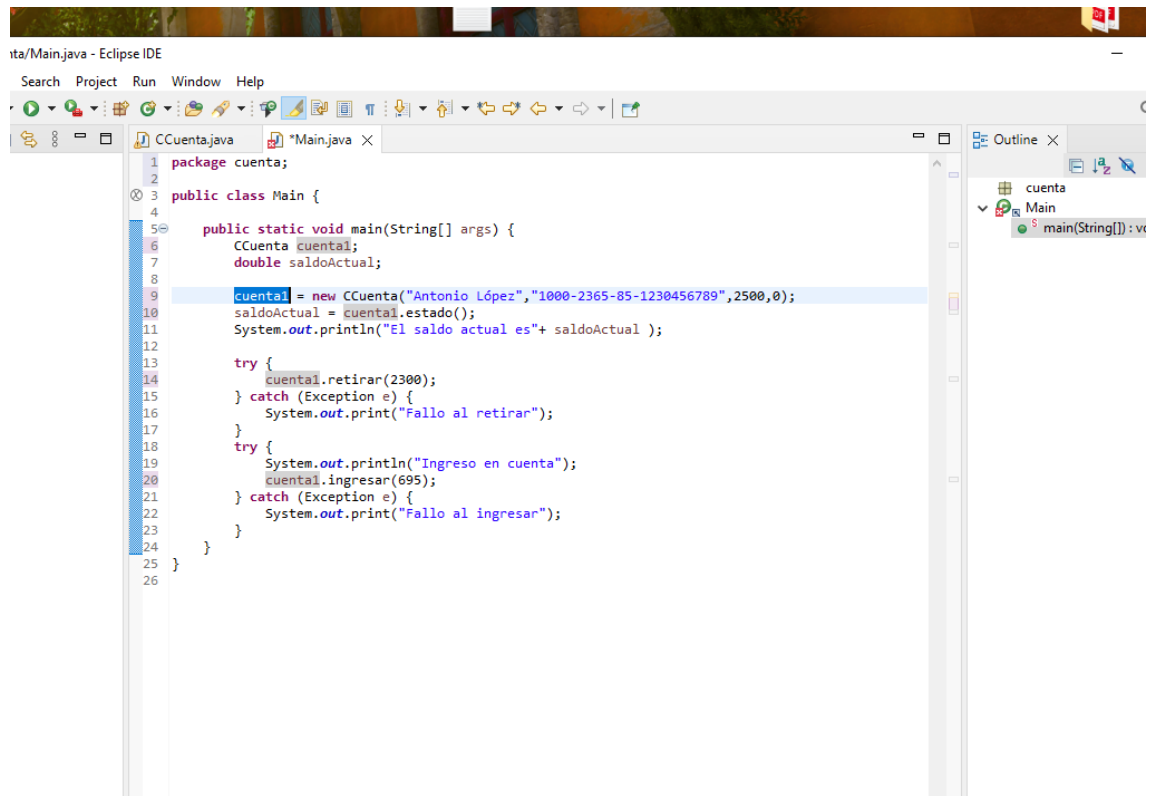
1. Las clases deberán formar parte del paquete cuentas.





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





The screenshot shows the Eclipse IDE with a Java project. The main editor displays the code for `Main.java` in the `cuenta` package. The code defines a `Main` class with a `main` method. Inside `main`, a `CCuenta` object named `cuenta1` is created with specific details. The current balance is retrieved and printed. Then, two operations are performed: a withdrawal of 2300 and a deposit of 695, each with a try-catch block for exceptions. The final balance is printed again. The Outline view on the right shows the project structure with `cuenta` and `Main` class.

```
1 package cuenta;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         CCuenta cuenta1;
7         double saldoActual;
8
9         cuenta1 = new CCuenta("Antonio López", "1000-2365-85-1230456789", 2500, 0);
10        saldoActual = cuenta1.estado();
11        System.out.println("El saldo actual es"+ saldoActual );
12
13        try {
14            cuenta1.retirar(2300);
15        } catch (Exception e) {
16            System.out.print("Fallo al retirar");
17        }
18        try {
19            System.out.println("Ingreso en cuenta");
20            cuenta1.ingresar(695);
21        } catch (Exception e) {
22            System.out.print("Fallo al ingresar");
23        }
24    }
25 }
26
```

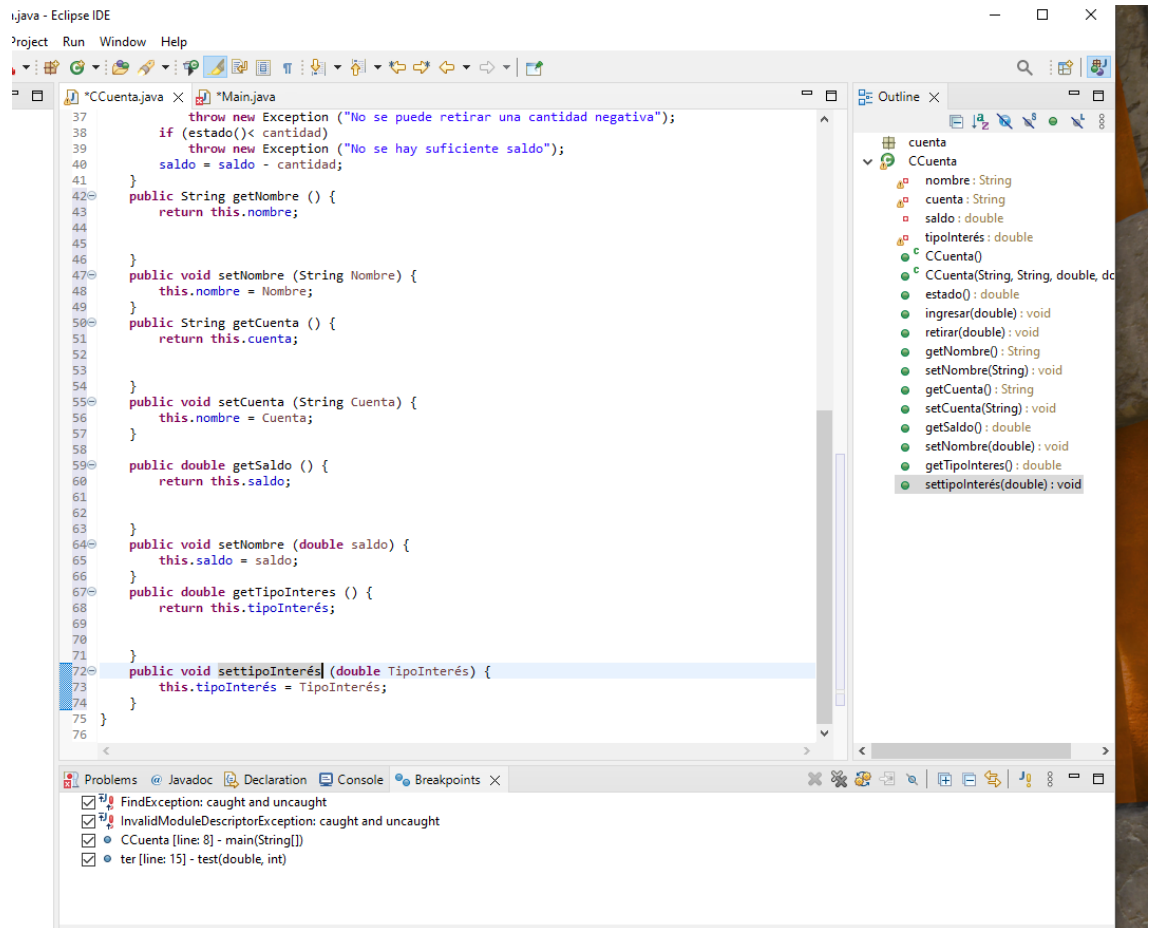
3. Introducir el método `operativa_cuenta`, que englobe las sentencias de la clase `Main` que operan con el objeto `cuenta1`.

```
1 package cuenta;
2
3 public class Main {
4     CCuenta cuenta1;
5     double saldoActual;
6     cuenta1 = new CCuenta("Antonio López", "1000-2365-85-1230456789", 2500, 0);
7     public static void main(String[] args) {
8         Main obj1 = new Main ();
9         obj1.operativa_cuenta ();
10
11     }
12 }
13
14
15 System.out.println("El saldo actual es"+ saldoActual );
16
17 try {
18     cuenta1.retirar(2300);
19 } catch (Exception e) {
20     System.out.print("Fallo al retirar");
21 }
22 try {
23     System.out.println("Ingreso en cuenta");
24     cuenta1.ingresar(695);
25 } catch (Exception e) {
26     System.out.print("Fallo al ingresar");
27 }
28
29 }
30 }
31
```

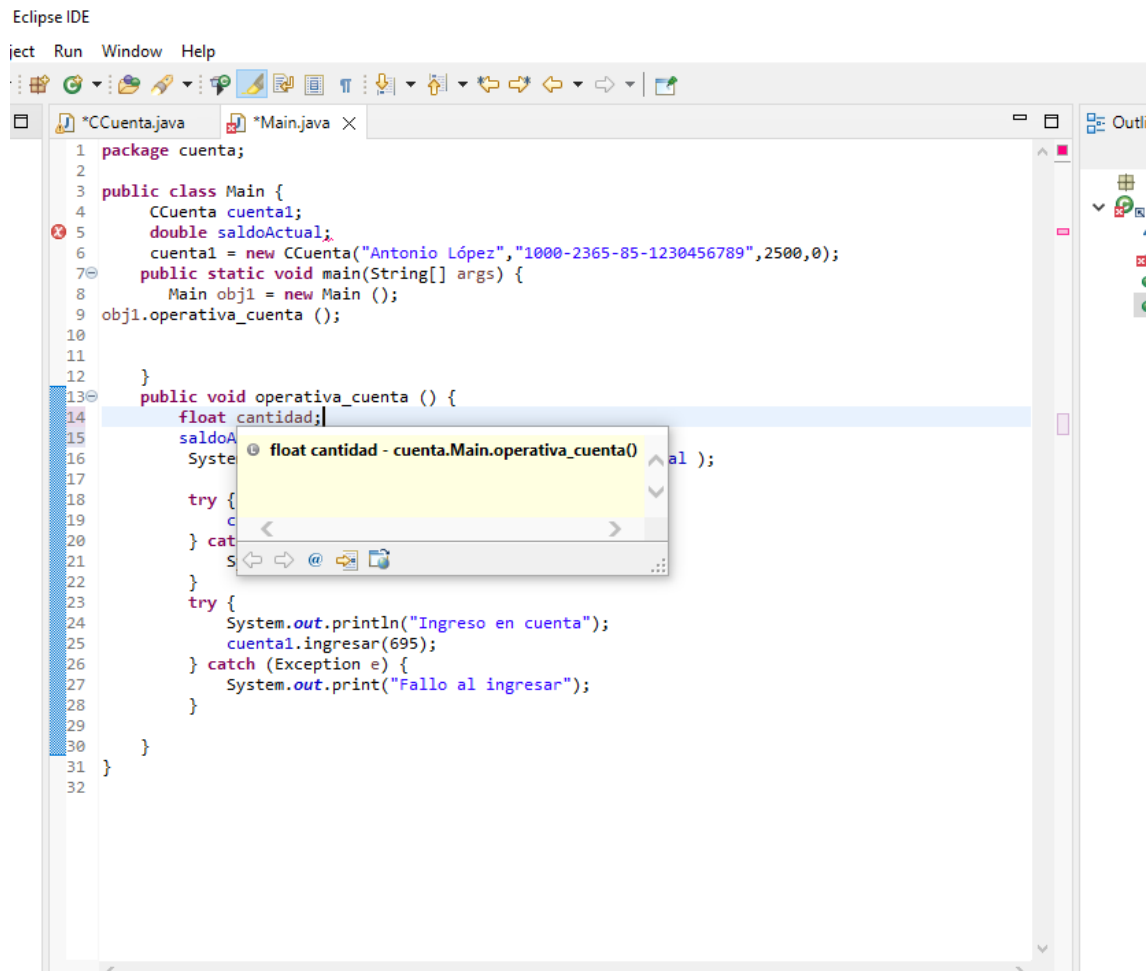
void cuenta.Main.operativa_cuenta()

Press 'F2' for focus

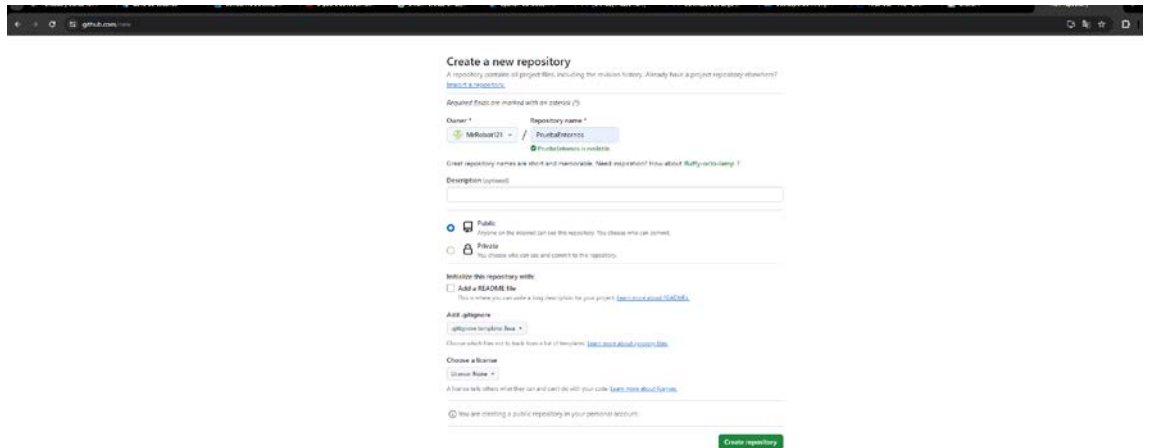
4. Encapsular los atributos de la clase CCuenta.



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



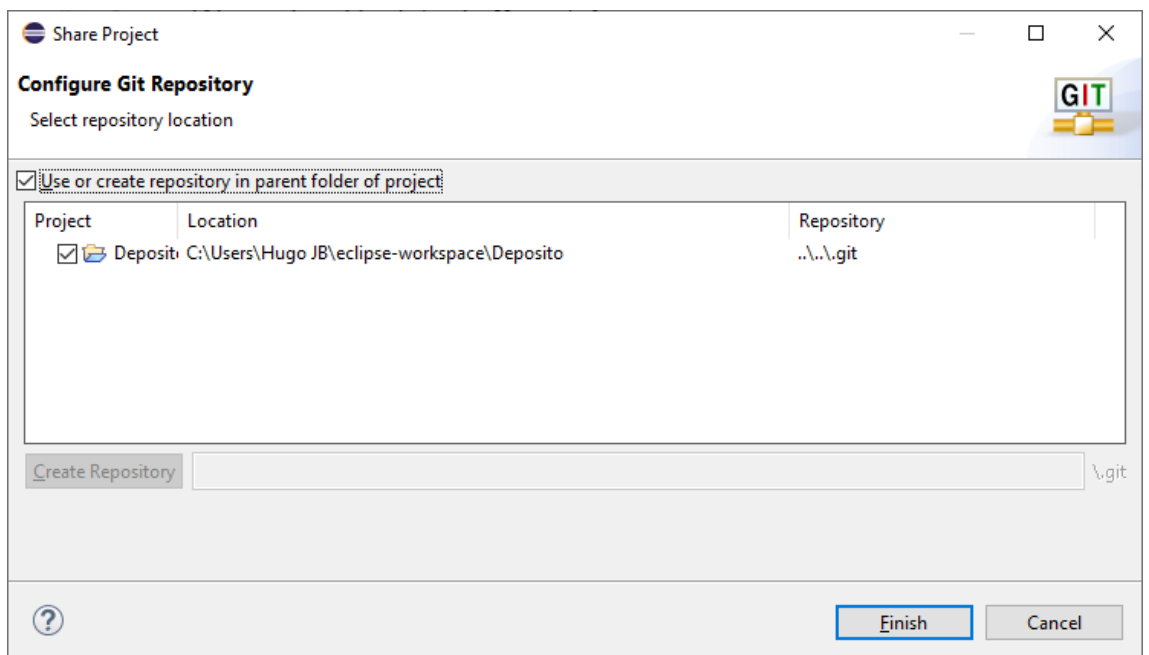
1. Configurar GIT para el proyecto. Crear un repositorio público en GitHub.



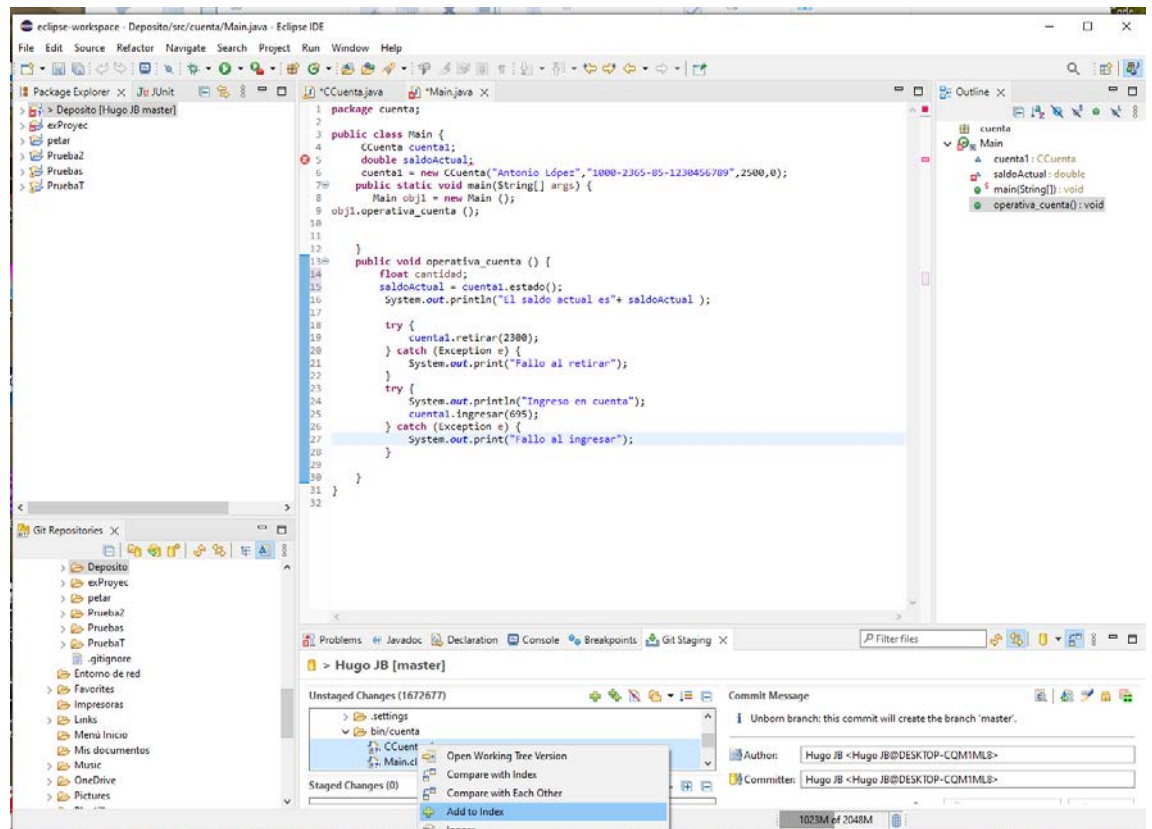
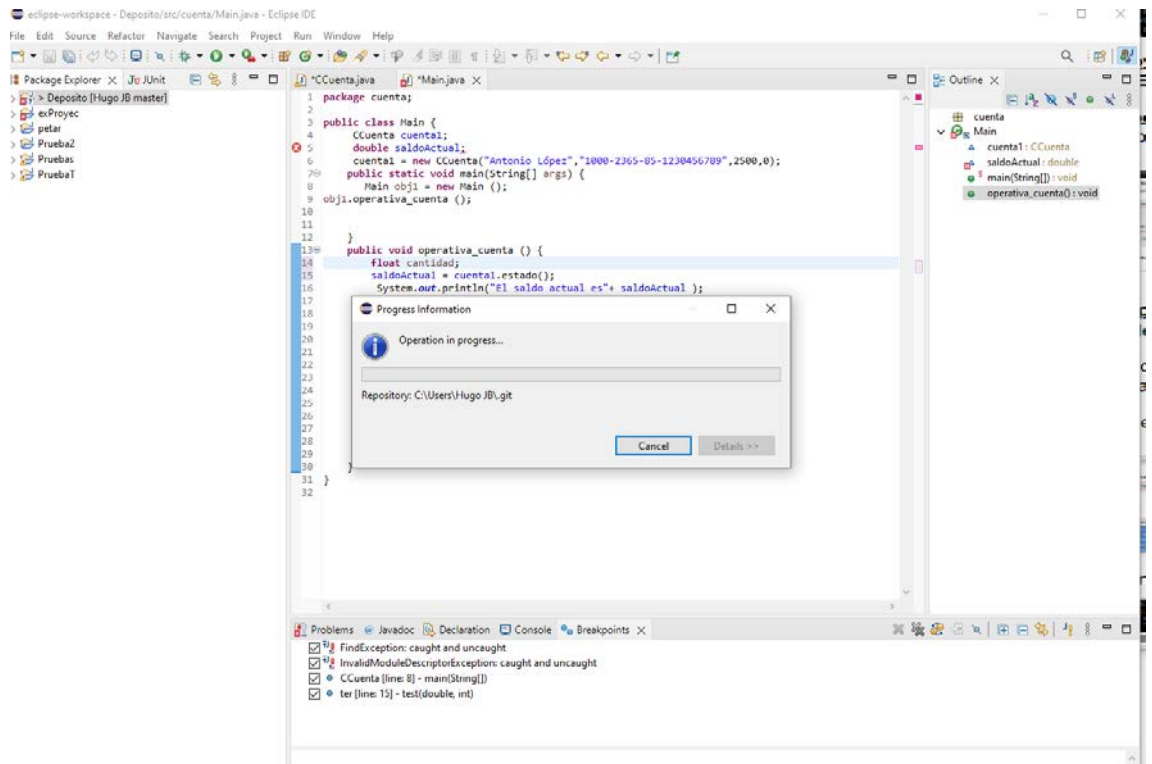
```
help [-dms] [pattern ...] { COMMANDS ; }

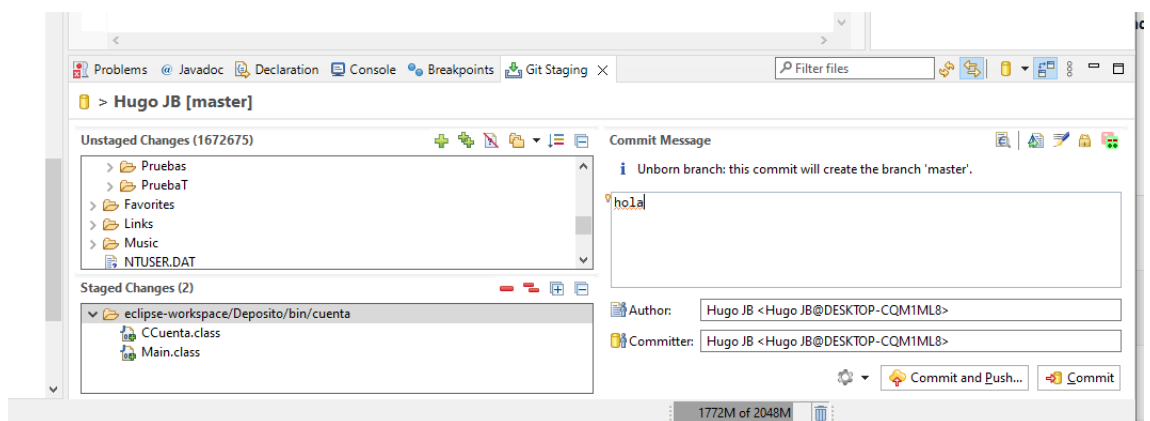
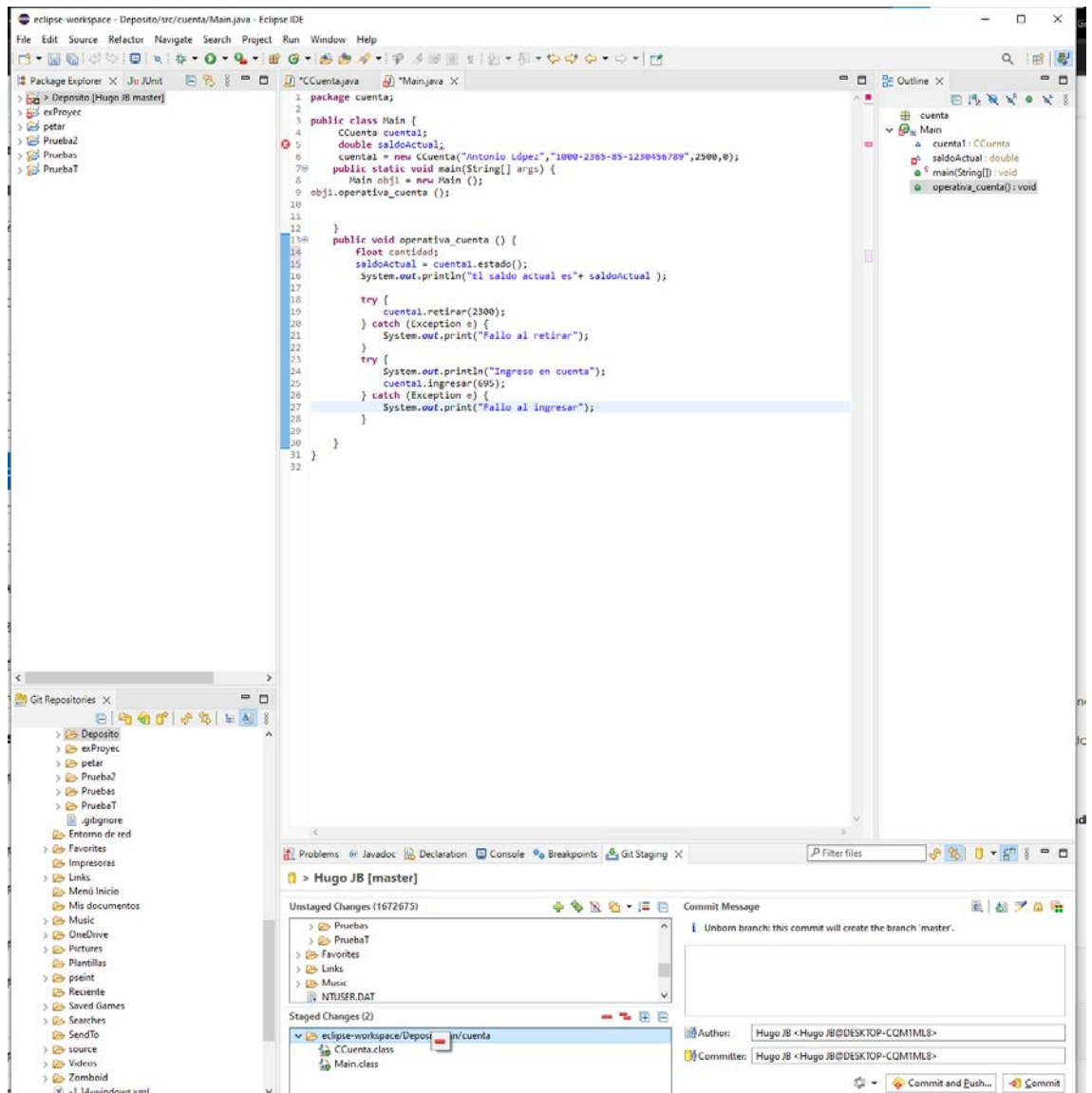
Hugo JB@DESKTOP-CQM1ML8 MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/Hugo JB/.git/

Hugo JB@DESKTOP-CQM1ML8 MINGW64 ~ (master)
$ |
```



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





```

fatal: your current branch 'master' does not have any commits yet

Hugo JB@DESKTOP-CQM1ML8 MINGW64 ~ (master)
$ git log
commit dadba5d48645924cfff152191a449090d2c34885 (HEAD -> master)
Author: Hugo JB <Hugo JB@DESKTOP-CQM1ML8>
Date: Tue Feb 13 20:56:55 2024 +0100

    hola

Hugo JB@DESKTOP-CQM1ML8 MINGW64 ~ (master)
$

```

Dice la fecha .Autor. Menssaje

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

```

Hugo JB@DESKTOP-CQM1ML8 MINGW64 ~ (master)
$ git show
commit dadba5d48645924cfff152191a449090d2c34885 (HEAD -> master)
Author: Hugo JB <Hugo JB@DESKTOP-CQM1ML8>
Date: Tue Feb 13 20:56:55 2024 +0100

    hola

diff --git a/eclipse-workspace/Deposito/bin/cuenta/CCuenta.class b/eclipse-workspace/Deposito/bin/cuenta/CCuenta.class
new file mode 100644
index 0000000..7b10777
Binary files /dev/null and b/eclipse-workspace/Deposito/bin/cuenta/CCuenta.class differ
diff --git a/eclipse-workspace/Deposito/bin/cuenta/Main.class b/eclipse-workspace/Deposito/bin/cuenta/Main.class
new file mode 100644
index 0000000..ed79a2d
Binary files /dev/null and b/eclipse-workspace/Deposito/bin/cuenta/Main.class differ

Hugo JB@DESKTOP-CQM1ML8 MINGW64 ~ (master)
$

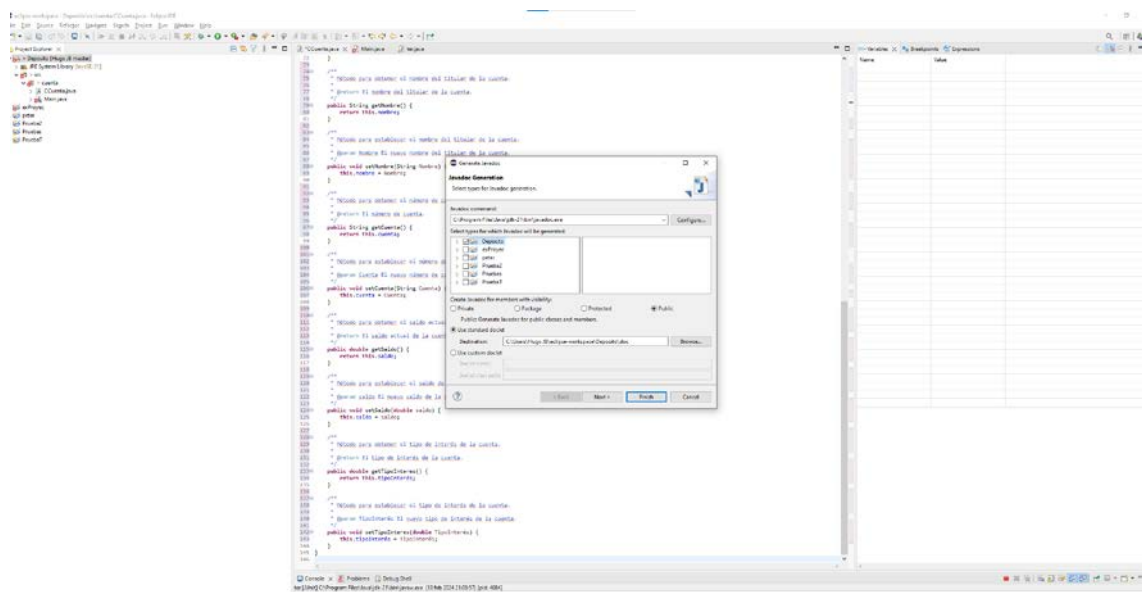
```

1. Insertar comentarios JavaDoc en la clase CCuenta.

```
1 package cuenta;
2
3 /**
4  * La clase CCuenta representa una cuenta bancaria con funcionalidades básicas.
5  * Permite realizar operaciones como ingresar, retirar y consultar el saldo.
6  * También proporciona métodos para acceder y modificar la información de la cuenta.
7  *
8  * @author hugg
9  * @version 1.0
10 */
11 public class CCuenta {
12
13     private String nombre;
14     private String cuenta;
15     private double saldo;
16     private double tipoInterés;
17
18     /**
19      * Constructor predeterminado de la clase CCuenta.
20      */
21     public CCuenta() {
22     }
23
24     /**
25      * Constructor con parámetros para inicializar la cuenta con valores específicos.
26      *
27      * @param nom El nombre del titular de la cuenta.
28      * @param cue El número de cuenta.
29      * @param sal El saldo inicial.
30      * @param tipo El tipo de interés.
31      */
32     public CCuenta(String nom, String cue, double sal, double tipo) {
33         nombre = nom;
34         cuenta = cue;
35         saldo = sal;
36         tipoInterés = tipo;
37     }
38
39     /**
40      * Método para obtener el estado actual del saldo.
41      *
42      * @return El saldo actual de la cuenta.
43      */
44     public double estado() {
45         return saldo;
46     }
47
48     /**
49      * Permite ingresar una cantidad en la cuenta.
50      *
51      * @param cantidad la cantidad a ingresar.
52      * @throws Exception si se intenta ingresar una cantidad negativa.
53      */
54     public void ingresar(double cantidad) throws Exception {
55         if (cantidad < 0)
56             throw new Exception("No se puede ingresar una cantidad negativa");
57         saldo = saldo + cantidad;
58     }
59
60     /**
61      * Permite retirar una cantidad de la cuenta.
62      *
63      * @param cantidad la cantidad a retirar.
64      * @throws Exception si se intenta retirar una cantidad negativa o si no hay suficiente saldo.
65      */
66     public void retirar(double cantidad) throws Exception {
67         if (cantidad <= 0)
68             throw new Exception("No se puede retirar una cantidad negativa");
69         if (estado() < cantidad)
70             throw new Exception("No se hay suficiente saldo");
71         saldo = saldo - cantidad;
72     }
73
74     /**
75      * Método para obtener el nombre del titular de la cuenta.
76      */
77 }
```

```
80         return this.nombre;
81     }
82
83     /**
84      * Método para establecer el nombre del titular de la cuenta.
85      *
86      * @param Nombre El nuevo nombre del titular de la cuenta.
87      */
88     public void setNombre(String Nombre) {
89         this.nombre = Nombre;
90     }
91
92     /**
93      * Método para obtener el número de cuenta.
94      *
95      * @return El número de cuenta.
96      */
97     public String getCuenta() {
98         return this.cuenta;
99     }
100
101     /**
102      * Método para establecer el número de cuenta.
103      *
104      * @param Cuenta El nuevo número de cuenta.
105      */
106     public void setCuenta(String Cuenta) {
107         this.cuenta = Cuenta;
108     }
109
110     /**
111      * Método para obtener el saldo actual de la cuenta.
112      *
113      * @return El saldo actual de la cuenta.
114      */
115     public double getSaldo() {
116         return this.saldo;
117     }
118
119     /**
120      * Método para establecer el saldo de la cuenta.
121      *
122      * @param saldo El nuevo saldo de la cuenta.
123      */
124     public void setSaldo(double saldo) {
125         this.saldo = saldo;
126     }
127
128     /**
129      * Método para obtener el tipo de interés de la cuenta.
130      *
131      * @return El tipo de interés de la cuenta.
132      */
133     public double getTipoInteres() {
134         return this.tipoInteres;
135     }
136
137     /**
138      * Método para establecer el tipo de interés de la cuenta.
139      *
140      * @param TipoInteres El nuevo tipo de interés de la cuenta.
141      */
142     public void setTipoInteres(double TipoInteres) {
143         this.tipoInteres = TipoInteres;
144     }
145 }
146
```

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



Resource

API

View

Tree

Notes

Help

Summary: Methods: Public Constructors: Methods: Details: Private Constructors: Methods: Methods: 1/1

getCuenta

public String getCuenta()
Método para obtener el número de cuenta.
Returns:
El número de cuenta.

setCuenta

public void setCuenta(String cuenta)
Método para establecer el número de cuenta.
Parameters:
cuenta - El nuevo número de cuenta.

getSaldo

public double getSaldo()
Método para obtener el saldo actual de la cuenta.
Returns:
El saldo actual de la cuenta.

setSaldo

public void setSaldo(double saldo)
Método para establecer el saldo de la cuenta.
Parameters:
saldo - El nuevo saldo de la cuenta.

getTipoInteres

public double getTipoInteres()
Método para obtener el tipo de interés de la cuenta.
Returns:
El tipo de interés de la cuenta.

setTipoInteres

public void setTipoInteres(double tipoInteres)
Método para establecer el tipo de interés de la cuenta.
Parameters:
TipoInteres - El nuevo tipo de interés de la cuenta.

Resource

API

View

Tree

Notes

Help

Summary: Methods: Public Constructors: Methods: Details: Private Constructors: Methods: Methods: 1/1

Constructors

Constructor

Description

CCuenta()

Constructor predeterminado de la clase CCuenta.

CCuenta(String nom, String cur, double sal, double tipo)

Constructor con parámetros para inicializar la cuenta con valores específicos.

Method Summary

Method and Type

Method

Description

double

estado()

Método para obtener el estado actual del saldo.

String

getCuenta()

Método para obtener el número de cuenta.

String

getNombre()

Método para obtener el nombre del titular de la cuenta.

double

getSaldo()

Método para obtener el saldo actual de la cuenta.

double

getTipoInteres()

Método para obtener el tipo de interés de la cuenta.

void

ingresar(double cantidad)

Permite ingresar una cantidad en la cuenta.

void

retirar(double cantidad)

Permite retirar una cantidad de la cuenta.

void

setCuenta(String cuenta)

Método para establecer el número de cuenta.

void

setNombre(String nombre)

Método para establecer el nombre del titular de la cuenta.

void

setSaldo(double saldo)

Método para establecer el saldo de la cuenta.

void

setTipoInteres(double TipoInteres)

Método para establecer el tipo de interés de la cuenta.

Methods inherited from class java.lang.Object

equals(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(), wait(), wait()

Constructor Details

CCuenta

public CCuenta()
Constructor predeterminado de la clase CCuenta.

CCuenta

public CCuenta(String nom, String cur, double sal, double tipo)
Constructor con parámetros para inicializar la cuenta con valores específicos.
Parameters:
nom - El nombre del titular de la cuenta.