Tarea para ED04.

Detalles de la tarea de esta unidad.

Enunciado.

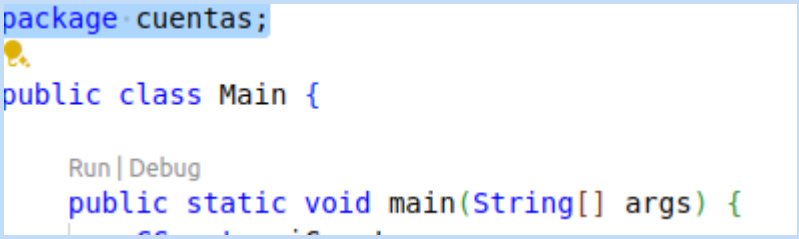
En el proyecto **Java "Deposito"**, hay definida una Clase llamada *CCuenta*, que tiene una serie de atributos y métodos. El proyecto cuenta asimismo con una Clase *Main*, donde se hace uso de la clase descrita.
Pulsa aquí para descargar dicho proyecto ("Deposito.rar").
Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

REFACTORIZACIÓN

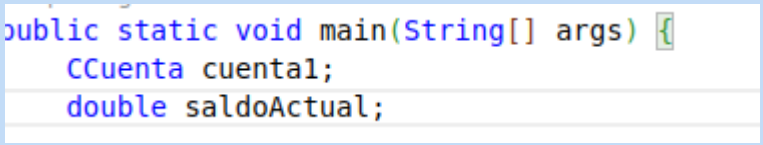
1. Las clases deberán formar parte del paquete cuentas.

   Creamos el paquete o carpeta cuentas y añadimos la palabra reservada package al inicio de cada archivo:

```
package cuentas;
```



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



Y así con todas las variables con el nombre miCuenta.

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

Movemos todo el código de dentro de Main a operativa_cuenta

```
public static void operativa_cuenta(){
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");
}
}
```

Y llamamos a este método desde el Main.

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

4. Encapsular los atributos de la clase CCuenta.

   Para encapsular, los atributos tienen que estar en private. Luego añadimos métodos setter() y getter para acceder a ellos.

```
//Paso 4: Añadimos los getters y setters para cada atributo
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;
}
```

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

```
public static void operativa_cuenta(float cantidad){
```

Y luego en la llamada al método añadimos una cantidad con la que llamar, por ejemplo 1234

```
operativa_cuenta(1234);
```

GIT

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

Ya tenía una cuenta en github, y git instalado y configurado previamente.
Creo un repositorio público que se puede consultar en https://github.com/Bobovino/ED04.

Lo he creado con el comando git init en local. Luego he guardado los cambios iniciales con git add . Y git commit –m “”. Después he creado el repositorio público con la interfaz gráfica de la web de github y pusheado el repositorio con git push.

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

   /Repositories/java/ED04$ git add .

   :~/Repositories/java/ED04$ git commit -m "Pasos 1,2 y 3"
   [main 76c090d] Pasos 1,2 y 3
   2 files changed, 32 insertions(+), 6 deletions(-)

```
:~/Repositories/java/ED04$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.10 KiB | 1.10 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Bobovino/ED04
   33569ea..76c090d  main -> main

:~/Repositories/java/ED04$
```

Aquí podemos ver un push exitoso.

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

Podemos ver el historial de versiones con el comando git log o log oneline

```
:~/Repositories/java/ED04$ git log --oneline
76c090d (HEAD -> main, origin/main, origin/HEAD) Pasos 1,2 y 3
33569ea First real commit
e78890b first commit
```

JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.
   Añadimos comentarios de tipo JavaDoc que tienen la estructura /** */ y que pueden tener las etiquetas @param @author @version @throws @return y @see. Por ejemplo:

```
/**
* Clase que representa una cuenta bancaria y sus operaciones básicas
* @author Rodrigo Martínez Tabernero
* @version 1.0
*/
```

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

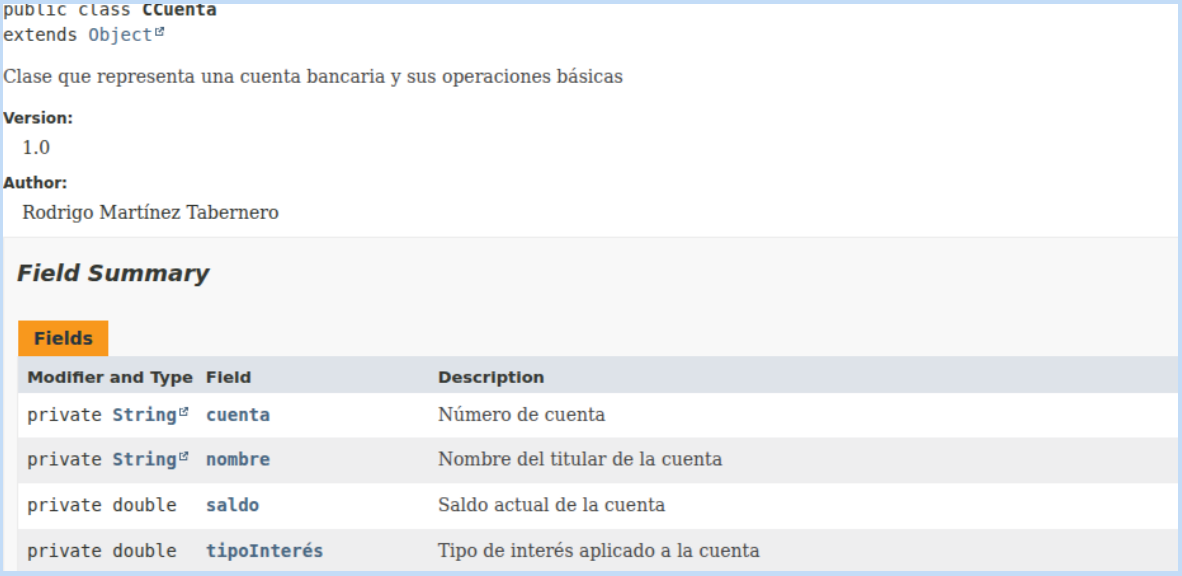
1. Generamos la documentación con el comando: Comprueba que la documentación generada por Javadoc, abarca todos los métodos y atributos de la clase Ccuenta. = 1 punto.

```
javadoc -d doc -sourcepath . -subpackages cuentas -private -author -version
```

He tenido que añadir los atributos en el comando porque no los veía en la documentación final.

Y podemos ver la documentación abriendo el index.html en el navegador

Primero vemos los atributos



public class **CCuenta**
extends Object

Clase que representa una cuenta bancaria y sus operaciones básicas

Version:
1.0

Author:
Rodrigo Martínez Tabernero

Field Summary

Fields

Modifier and Type	Field	Description
private String	cuenta	Número de cuenta
private String	nombre	Nombre del titular de la cuenta
private double	saldo	Saldo actual de la cuenta
private double	tipoInterés	Tipo de interés aplicado a la cuenta

Y luego los métodos: como no hemos usado tipoInterés, pues no le he hecho getter y setter ni tampoco los he documentado en exceso. Pero el resto abajo están.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
double	estado()	Obtiene el saldo actual
String	getCuenta()	Obtiene el número de cuenta
String	getNombre()	Obtiene el nombre del titular
void	ingresar(double cantidad)	Realiza un ingreso en la cuenta
void	retirar(double cantidad)	Realiza una retirada de la cuenta
void	setCuenta(String cuenta)	Establece el número de cuenta
void	setNombre(String nombre)	Establece el nombre del titular

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait