

ciclo: [DAM]

MÓDULO DE [PROGRAMACIÓN]

[Tarea Nº 07]

Alumno:

[Juan Carlos Filter Martín]

[15456141A]

**Contenido**

[1. Documentos que se adjuntan a este informe. 3](#__RefHeading___Toc72_2776016478)

[2. RA07\_g) Se han realizado programas que implementen y utilicen jerarquías de clases. 3](#__RefHeading___Toc74_2776016478)

[A) Crear proyecto en NetBeans denominado "CuentasBancarias" 3](#__RefHeading___Toc76_2776016478)

[B) Dentro de dicho proyecto, crear un paquete denominado "modeloBancario" 3](#__RefHeading___Toc78_2776016478)

[C) Dentro del paquete "modeloBancario", crear una clase denominada Cliente, que modele los distintos clientes del banco que tienen una cuenta asociada para almacenar su dinero. 4](#__RefHeading___Toc80_2776016478)

[I. Las características clase Cliente 5](#__RefHeading___Toc82_2776016478)

[3. RA07\_b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos. 7](#__RefHeading___Toc74_2776016478_Copia_1)

[A) Dentro del paquete *"modeloBancario"*, crear una clase abstracta denominada "Cuenta", que modele las distintas cuentas bancarias que mantiene la entidad financiera donde los clientes depositan su dinero. 7](#__RefHeading___Toc99_2776016478)

[I. Las características clase Cuenta 8](#__RefHeading___Toc82_2776016478_Copia_1)

[4. RA07\_c) Se ha reconocido la incidencia de los constructores en la herencia. 11](#__RefHeading___Toc74_2776016478_Copia_1_)

[A) Clases padre y clases hijas, deben de tener implementados los constructores. 11](#__RefHeading___Toc567_3356961770)

[5. RA07\_d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase. 12](#__RefHeading___Toc74_2776016478_Copia_11)

[A) Dentro del paquete "modeloBancario", crear una clase denominada "CuentaCorriente", que herede de la clase “Cuenta”, que modele un tipo de cuenta con un interés fijo del 1.5%. 12](#__RefHeading___Toc590_3356961770)

[I. Las características clase CuentaCorriente 13](#__RefHeading___Toc82_2776016478_Copia_1_)

[Dentro del paquete "modeloBancario", crear una clase denominada "CuentaAhorro", que herede de la clase “Cuenta”, que modele un tipo de cuenta con un interés variable y un saldo mínimo necesario. 14](#__RefHeading___Toc592_3356961770)

[I. Las características clase CuentaAhorro 15](#__RefHeading___Toc82_2776016478_Copia_11)

[6. RA07\_e) Se han diseñado y aplicado jerarquías de clases. 17](#__RefHeading___Toc74_2776016478_Copia_12)

[A) Crear un entorno que importe el paquete "modeloBancario" y que permita probar las distintas clases Cuentas implementadas. 17](#__RefHeading___Toc594_3356961770)

[B) Probar clases mediante la clase main “CuentasBancarias” 17](#__RefHeading___Toc2507_2712339160)

[I. CuentaCorriente 17](#__RefHeading___Toc2509_2712339160)

[II. CuentaAhorro 21](#__RefHeading___Toc2511_2712339160)

[7. RA07\_f) Se han probado y depurado las jerarquías de clases. 25](#__RefHeading___Toc74_2776016478_Copia_13)

[Se han adjuntado captura sobre el funcionamiento de la aplicación y se han realizado comentarios en el código. 25](#__RefHeading___Toc2533_2712339160)

[Cliente 25](#__RefHeading___Toc2523_2712339160)

[Cuenta (Abstract) 26](#__RefHeading___Toc2525_2712339160)

[CuentaCorriente (hereda de cuenta) 27](#__RefHeading___Toc2527_2712339160)

[CuentaAhorro (hereda de cuenta) 28](#__RefHeading___Toc2529_2712339160)

[CuentasBancarias (main class) 29](#__RefHeading___Toc2531_2712339160)

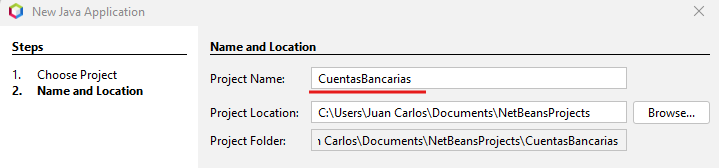
# Documentos que se adjuntan a este informe.

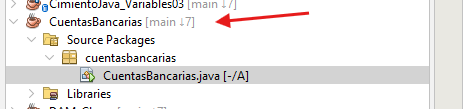
A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Proyecto java “CuentasBancarias”.

# RA07\_g) Se han realizado programas que implementen y utilicen jerarquías de clases.

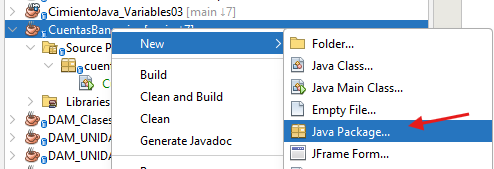
## Crear proyecto en NetBeans denominado "CuentasBancarias"



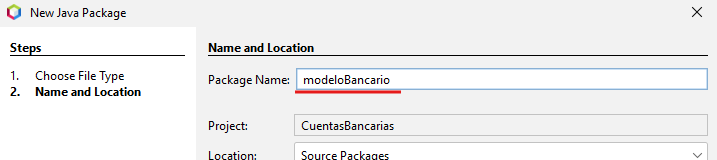


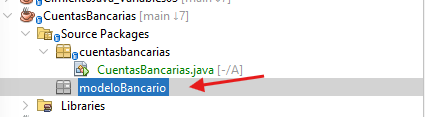
## Dentro de dicho proyecto, crear un paquete denominado "modeloBancario"

Botón derecho sobre el proyecto > New > **Java Package**



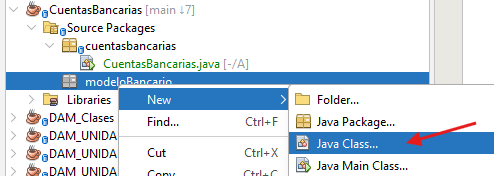
Le asignamos el nombre **modeloBancario**



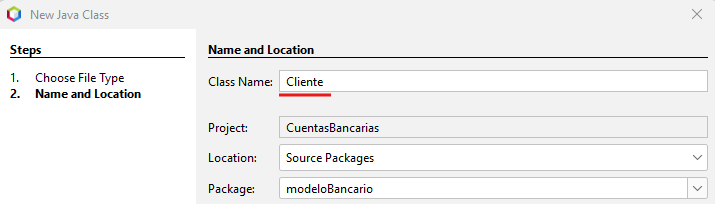


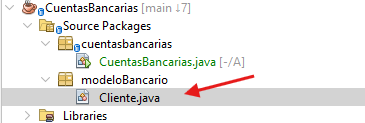
## Dentro del paquete "modeloBancario", crear una clase denominada Cliente, que modele los distintos clientes del banco que tienen una cuenta asociada para almacenar su dinero.

Botón derecho sobre el paquete “modeloBancario” > New > **Java Class**



Le asignamos el nombre **Cliente***(dentro del paquete modeloBancario)*





### Las características clase Cliente

**Atributos: *Los atributos con visibilidad privada.***

***private****: Para que sean con visibilidad privada se debe indicar mediante* ***private***

* **idCliente:** número entero que representa el identificador único del cliente dentro del banco.



private int idCliente private int idCliente

* **nombre:** cadena de caracteres que representa el nombre del cliente del banco.



private String nombre private String nombre

* **direccion**: cadena de caracteres que representa la dirección donde vive el cliente del banco.



private String direccion private String direccion

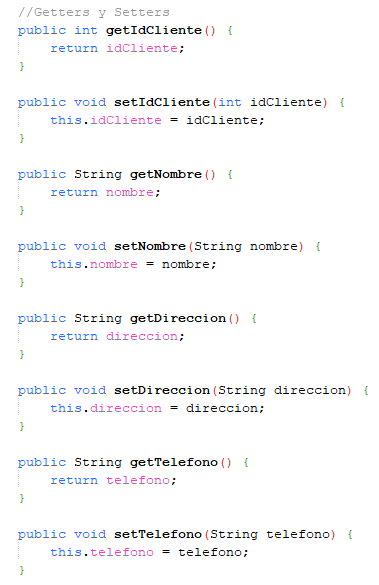
* **telefono**: cadena de caracteres que representa el teléfono que permite contactar con el cliente.



private String telefono Private String telefono

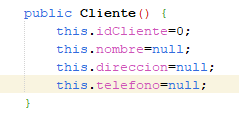
**Métodos L*os métodos con visibilidad pública.***

***public****: Para que sean con visibilidad pública se debe indicar mediante* ***public***

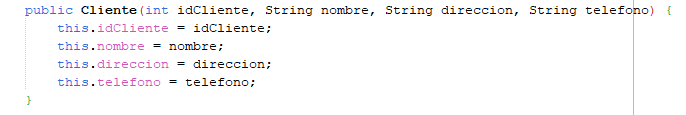


setT

***Constructor sin parámetros: constructor que inicializa todos los atributos de tipo cadenas de caracteres al valor null y los números enteros a 0.***

******

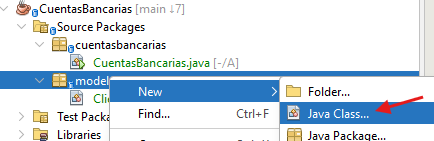
***Constructor con parámetros:*** ***constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes.***

******

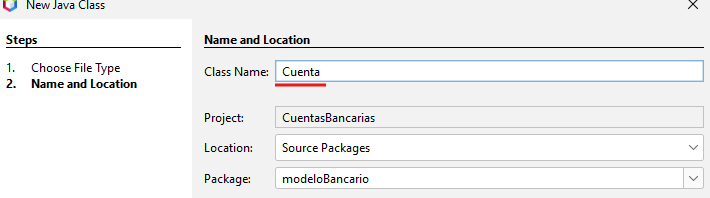
# RA07\_b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos.

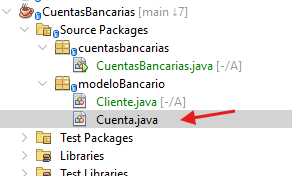
## Dentro del paquete ***"modeloBancario"***, crear una clase abstracta denominada "Cuenta", que modele las distintas cuentas bancarias que mantiene la entidad financiera donde los clientes depositan su dinero.

Botón derecho sobre el paquete “modeloBancario” > New > **Java Class**



Le asignamos el nombre **Cuenta***(dentro del paquete modeloBancario)*

**





### Las características clase Cuenta

**Atributos: *Los atributos con visibilidad protegida.***

***protected*** *: Para que sean con visibilidad protegida se debe indicar mediante* ***protected.***

* **numeroDeCuenta:** número entero que representa el identificador único asociado a cada una de las cuentas del banco.



protected int numeroDeCuenta protected int numeroDeCuenta

* **saldo**: número real que representa la cantidad de dinero almacenado en dicha cuenta.



protected double saldo protected double saldo

* **titular**: atributo de tipo Cliente que representa la persona que está asociada a dicha cuenta.

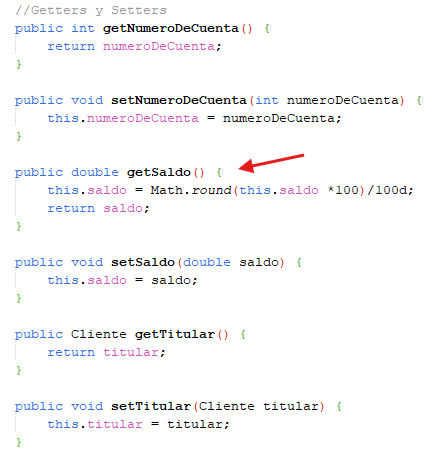


protected Cliente titular

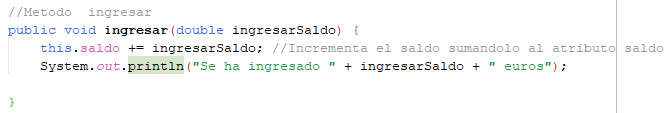
protected Cliente titular

**Métodos Los métodos con visibilidad pública.**

***public****: Para que sean con visibilidad pública se debe indicar mediante* ***public***

**El método getSaldo redondea el saldo a 2 decimales y devuelve saldo** getNumeroDeCuentasetNumeroDeCugetSaldsetSaldo getTitul setTitu

**ingresar**: recibe un parámetro entero que representa la cantidad que se desea ingresar en la cuenta. El método incrementará el saldo en la cantidad recibida como parámetro.



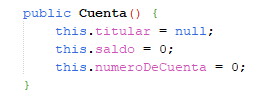
**retirar**: método abstracto que permitirá sacar una cantidad de la cuenta (si hay saldo disponible para ello), no se implementará ya que dependerá del tipo de cuenta, por tanto su implementación recaerá en las clases hijas.



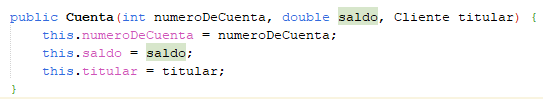
**actualizarSaldo**: método abstracto que actualizará el saldo de la cuenta, dependiendo del tipo de interés de cada una de las cuenta, por tanto su implementación recaerá en las clases hijas.

****

**Constructor sin parámetros: constructor que inicializa el cliente titular de la cuenta a null, y el saldo y el número de cuenta a cero.**



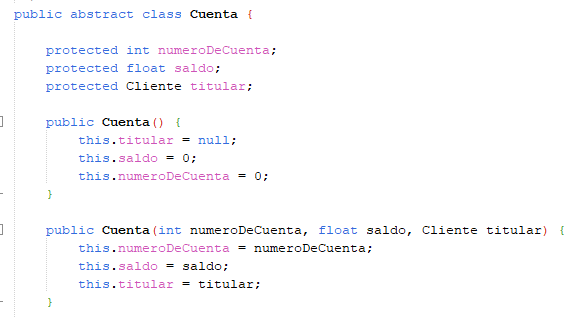
**Constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes.**



# RA07\_c) Se ha reconocido la incidencia de los constructores en la herencia.

## Clases padre y clases hijas, deben de tener implementados los constructores.

**→ Clase padre Cuenta** *(Abstracta)*

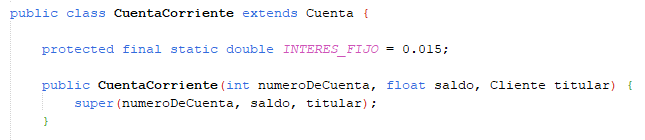


Constructor sin parámetros

Constructor con parámetros

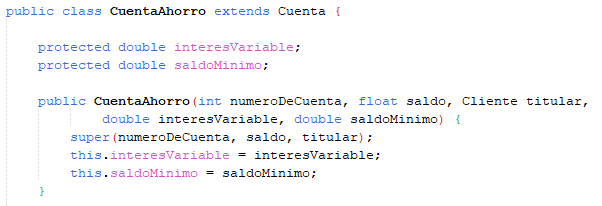
Constructor con parámetros

**→ Clase hija CuentaCorriente**



**→ Clase hija CuentaAhorro**

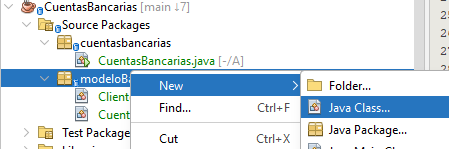
Constructor con parámetros



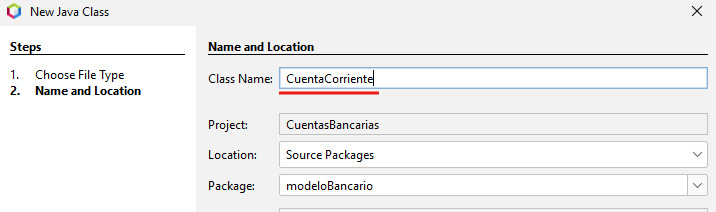
# RA07\_d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase.

## Dentro del paquete "modeloBancario", crear una clase denominada "CuentaCorriente", que herede de la clase “Cuenta”, que modele un tipo de cuenta con un interés fijo del 1.5%.

Botón derecho sobre el proyecto > New > **Java Class**



Le asignamos el nombre **CuentaCorriente***(dentro del paquete modeloBancario)*



**Hereda de la clase Cuenta**

public class CuentaCorriente extends Cuenta public class CuentaCorriente extends Cuenta



### Las características clase CuentaCorriente

**Atributos Los atributos de la clase deben tener visibilidad protegida.**

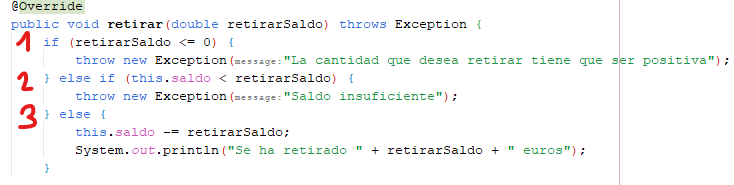
***protected*** *: Para que sean con visibilidad protegida se debe indicar mediante* ***protected.***

**interesFijo:** constante real cuyo valor es 0.015.



**Métodos** **Los métodos deben tener visibilidad pública.**

**Métodos abstracto retirar:**

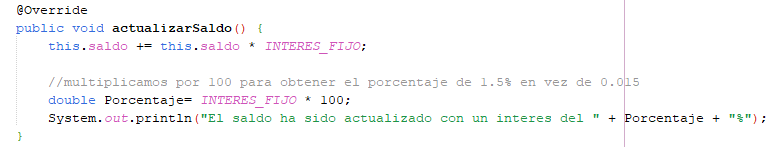


1 – Si retirarSaldo(indicado por parámetros) es 0 o <: *El saldo a retirar debe ser positivo.*

2 – Si saldo es menor a retirarSaldo: *Insuficiente saldo.*

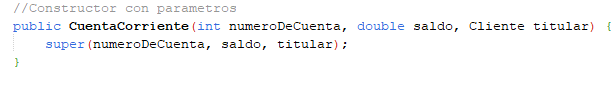
3 – Si no se cumple lo anterior entonces resta rertirarSaldo al saldo de la cuenta.

**Métodos abstracto actualizarSaldo.**

****

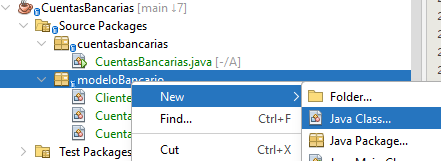
1 – Actualizamos el saldo de la cuenta añadiéndole el interés fijo.

**Constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes. Dicho constructor debe hacer uso del constructor de la clase padre “Cuenta”.**

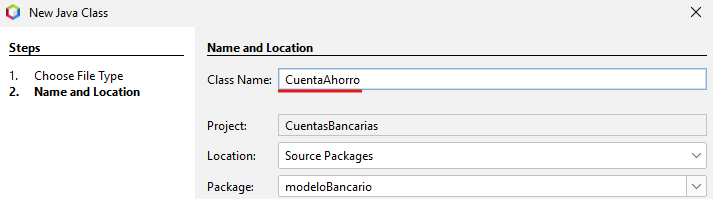


## Dentro del paquete "modeloBancario", crear una clase denominada "CuentaAhorro", que herede de la clase “Cuenta”, que modele un tipo de cuenta con un interés variable y un saldo mínimo necesario.

Botón derecho sobre el proyecto > New > **Java Class**



Le asignamos el nombre **CuentaAhorro***(dentro del paquete modeloBancario)*



**Hereda de la clase Cuenta**

public class CuentaAhorro extends Cuenta

public class CuentaAhorro extends Cuenta



### Las características clase CuentaAhorro

**Atributos Los atributos deben tener visibilidad protegida**

***protected*** *: Para que sean con visibilidad protegida se debe indicar mediante* ***protected****.*

**interesVariable:** número real que representa el tipo de interés que se aplica a la cuenta.

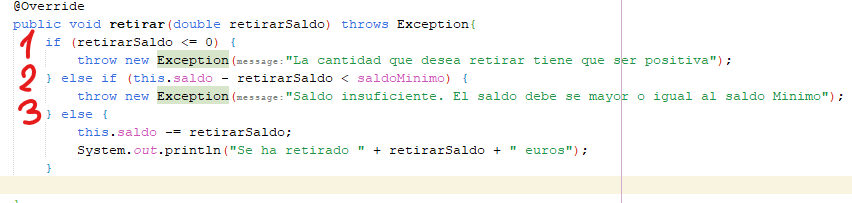


**saldoMinimo:** número real que representa el dinero mínimo que debe haber en la cuenta.



**Métodos (Todos los métodos de la clase CuentaAhorro deben tener visibilidad pública)**

**Métodos abstracto retirar:**

****

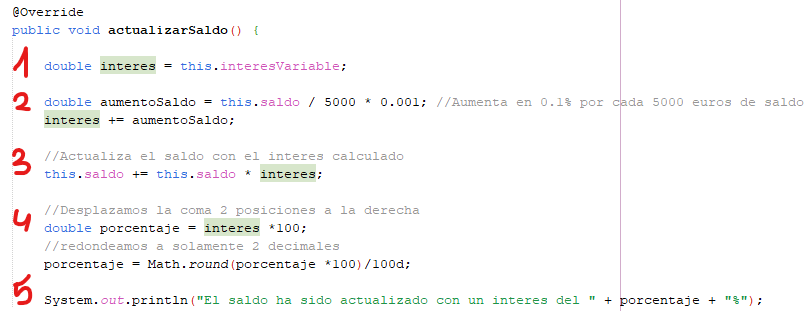
1 – Si retirarSaldo(indicado por parámetros) es 0 o <: *El saldo a retirar debe ser positivo.*

2 – Si saldo se le resta retirarSaldo y es menor al saldoMinimo: *Insuficiente saldo*

*(el saldosiempre debe ser mayor o igual al saldoMinimo)*

3 – Si no se cumple lo anterior entonces resta rertirarSaldo al saldo de la cuenta.

**Métodos abstracto actualizarSaldo.**



1 – Almacenamos el valor de interesVariable en interes

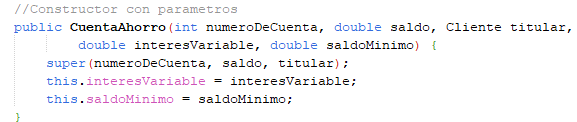
2 – Por cada 5000 aumentamos en 0.1% y almacenamos ese porcentaje en aumentoSaldo e indicamos que aumentoSaldo se sume al interes.

3 – Aplicamos el interes al saldo.

4 – multiplicamos por 100 el interes para obtener el % sin 0 a la izquierda y posteriormente redondeamos a 2 decimales.

5 – Mostramos por pantalla.

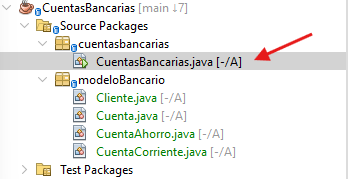
**Constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes. Dicho constructor debe hacer uso del constructor de la clase padre “Cuenta”.**



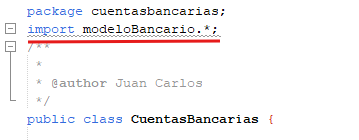
# RA07\_e) Se han diseñado y aplicado jerarquías de clases.

## Crear un entorno que importe el paquete "modeloBancario" y que permita probar las distintas clases Cuentas implementadas.

Esta clase está situada en el mismo proyecto pero en diferente paquete.



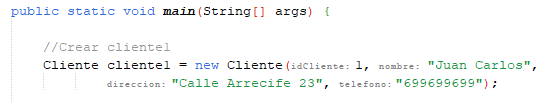
Entonces debemos importar el paquete modeloBancario a esta clase



## Probar clases mediante la clase main “CuentasBancarias”

### CuentaCorriente

* Creamos un **objeto** de la **clase** **cliente** Con parámetros *(Registramos 1 cliente)*



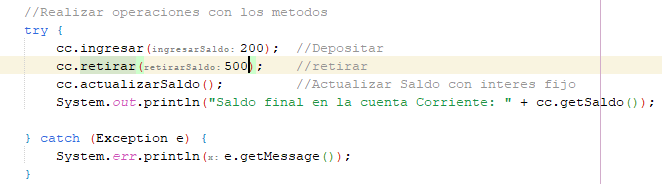
* Creamos un **objeto** de la **clase** **CuentaCorriente**

****

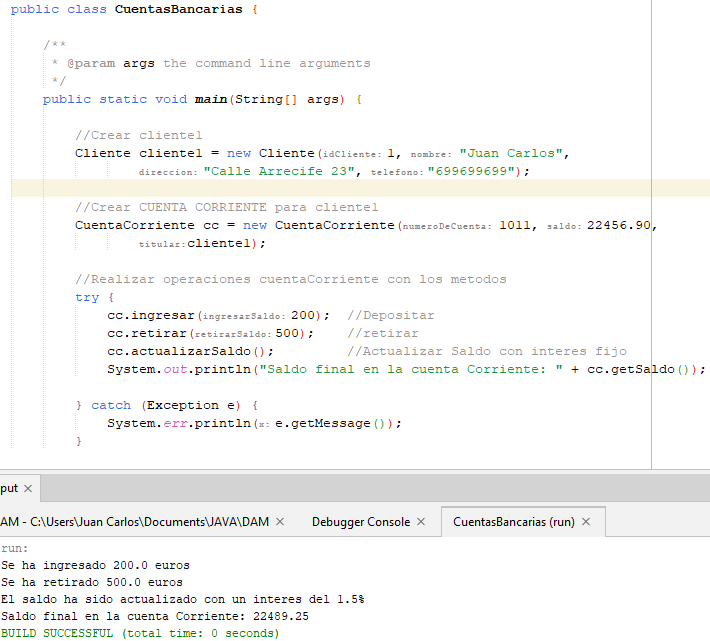
**1 – Prueba con éxito**

Mediante el objeto cuenta corriente “cc” le indicamos el método para ingresar, retirar y actualizar el saldo

Mostrando este mediante un System.out.println junto al método getSalgo que se encuentra en la clase cliente



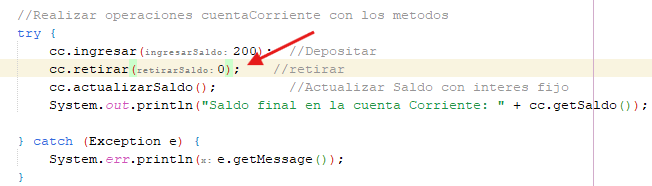
Resultado por pantalla:



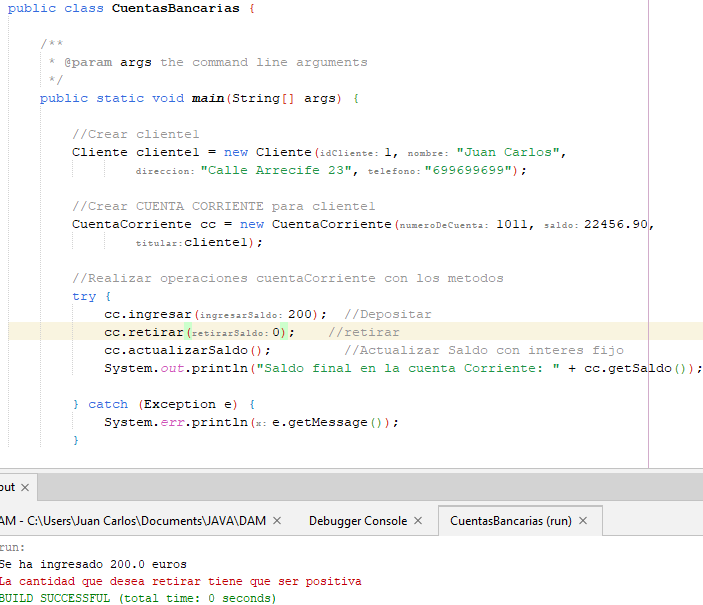
**2 – Prueba sin éxito**

En el método retirar: al indicarle un valor de 0 o menor, mandará un mensaje de error





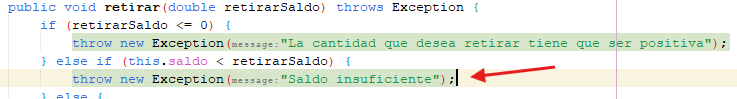
Resultado por pantalla:

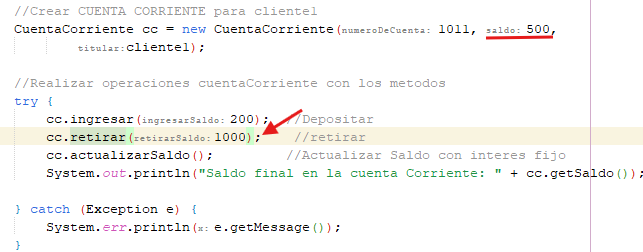


**3 – Prueba sin éxito**

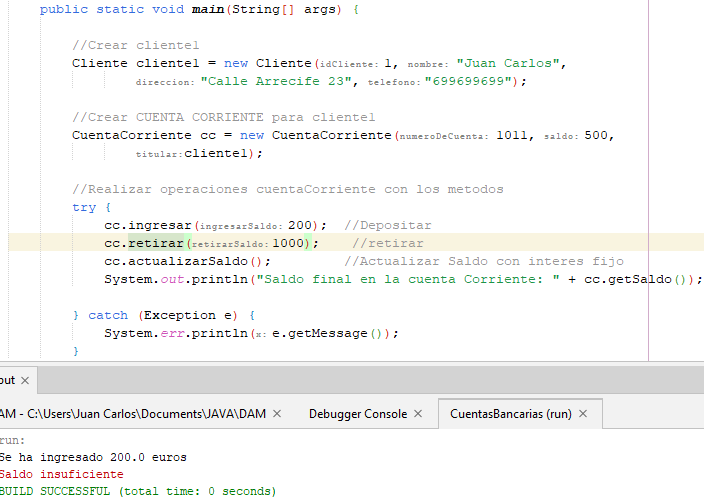
En el método retirar: al indicarle un valor mayor al saldo que tiene en la cuenta, mandará un mensaje de error

Se le ha indicado que el saldo en la cuenta sea de 500 y posteriormente se ha intentado retirar 1000



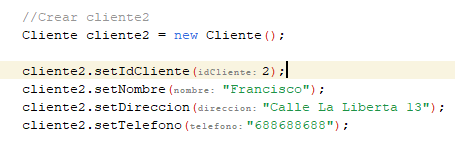


Resultado por pantalla:



### CuentaAhorro

* Creamos un **objeto** de la **clase** **cliente** Sin parámetros *(Registramos 1 cliente)*
* Y asignamos los atributos mediante los Setters



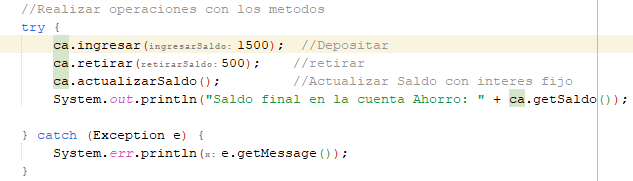
* Creamos un **objeto** de la **clase** **CuentaAhorro**

****

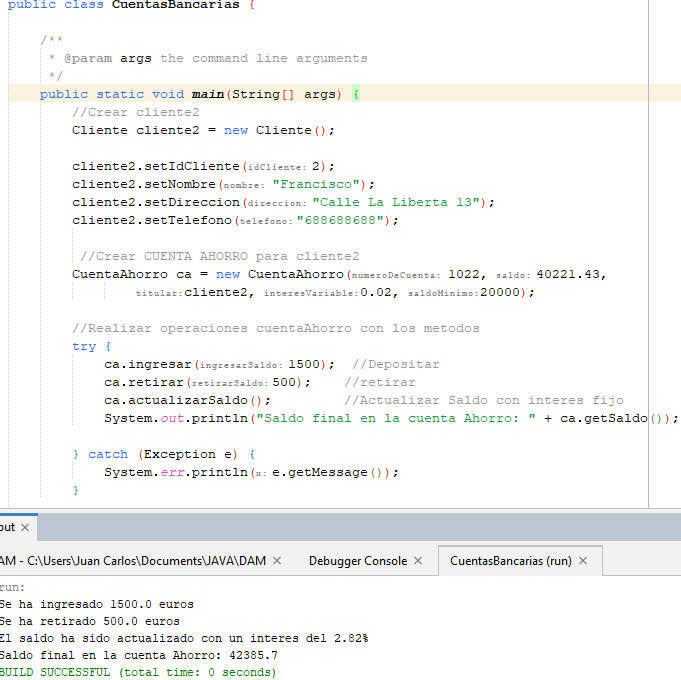
**1 – Prueba con éxito**

Mediante el objeto cuenta ahorro “ca” le indicamos el método para ingresar, retirar y actualizar el saldo

Mostrando este mediante un System.out.println junto al método getSalgo que se encuentra en la clase cliente

****

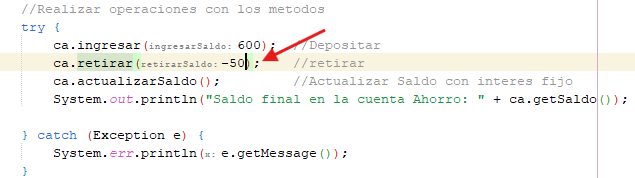
**Resultado por pantalla:**

****

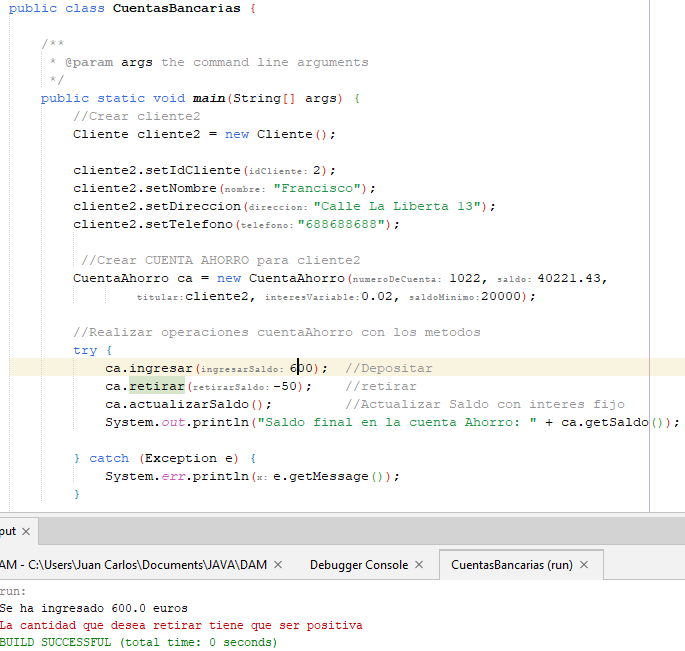
**2 – Prueba sin éxito**

En el método retirar: al indicarle un valor de 0 o menor, mandará un mensaje de error

****

****

**Resultado por pantalla:**

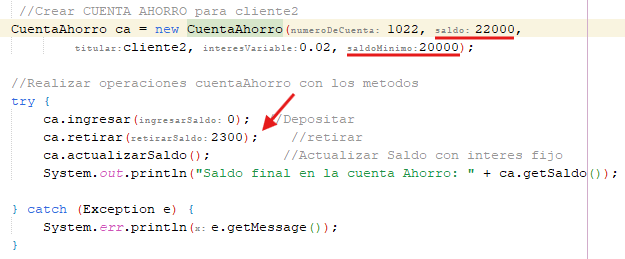
****

**3 – Prueba sin éxito**

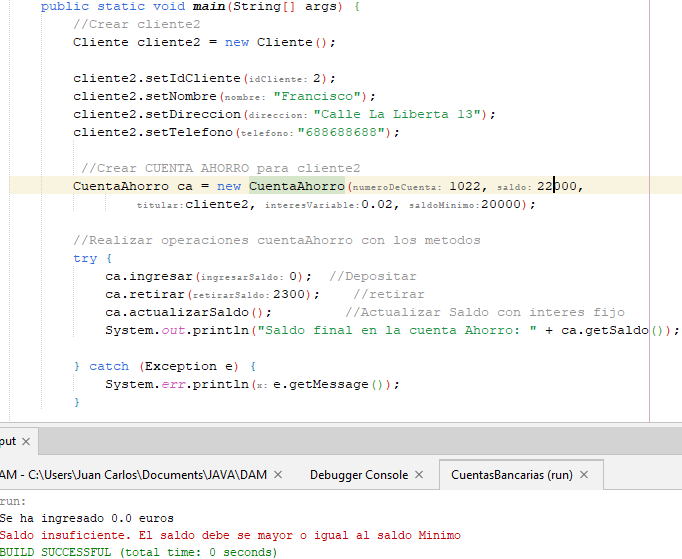
En el método retirar: Si se le resta el valor de retirar saldo al saldo y este es menor o igual al saldo mínimo, mandará un mensaje de error

Se le ha indicado que el saldo en la cuenta sea de 22000 con un saldo mínimo de 20000 y posteriormente se ha intentado retirar 2300.

El dinero restante sería 19000 entonces mandará mensaje de error



**Resultado por pantalla:**

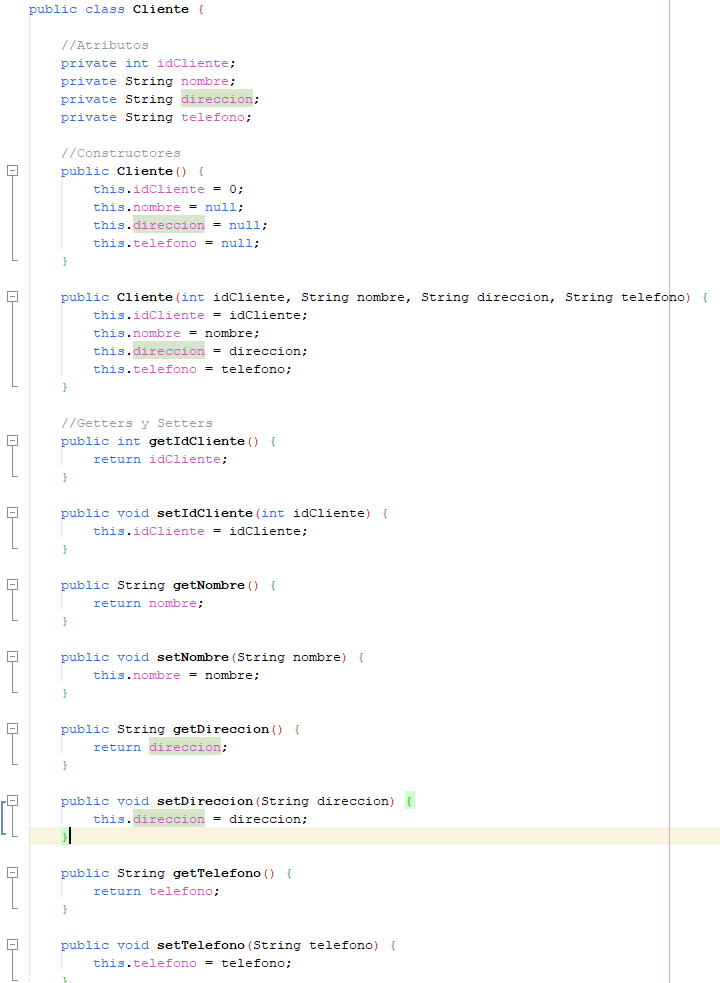
****

# RA07\_f) Se han probado y depurado las jerarquías de clases.

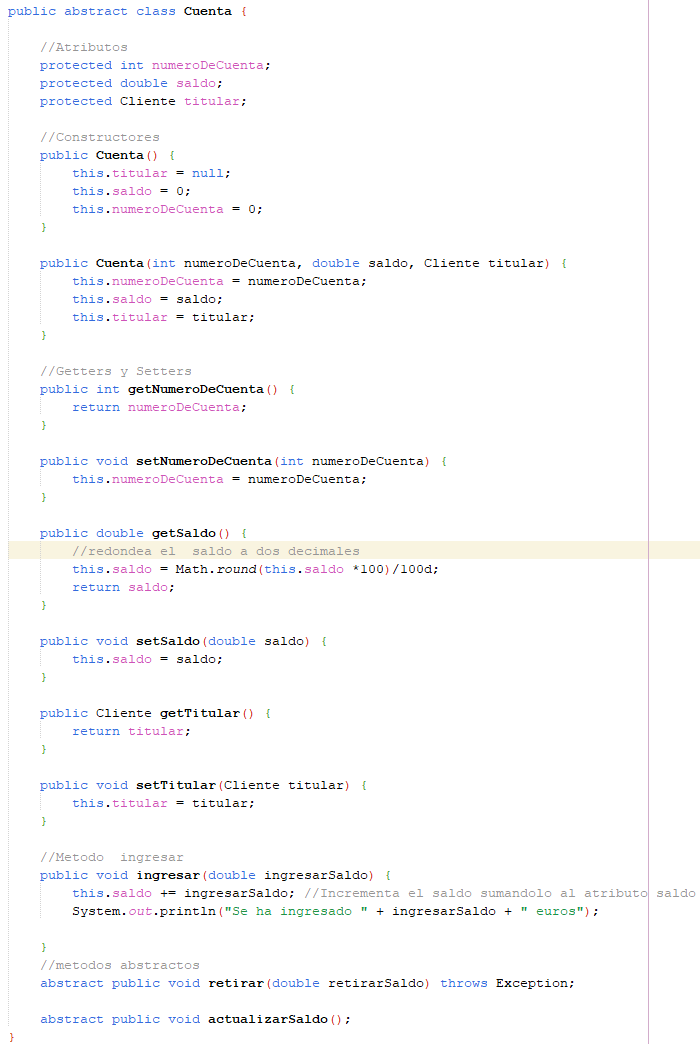
## Se han adjuntado captura sobre el funcionamiento de la aplicación y se han realizado comentarios en el código.

Para finalizar se van a dejar capturas de cada clase de la aplicación

## Cliente



## Cuenta (Abstract)



## CuentaCorriente (hereda de cuenta)

## 

## CuentaAhorro (hereda de cuenta)



## CuentasBancarias (main class)

