

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.

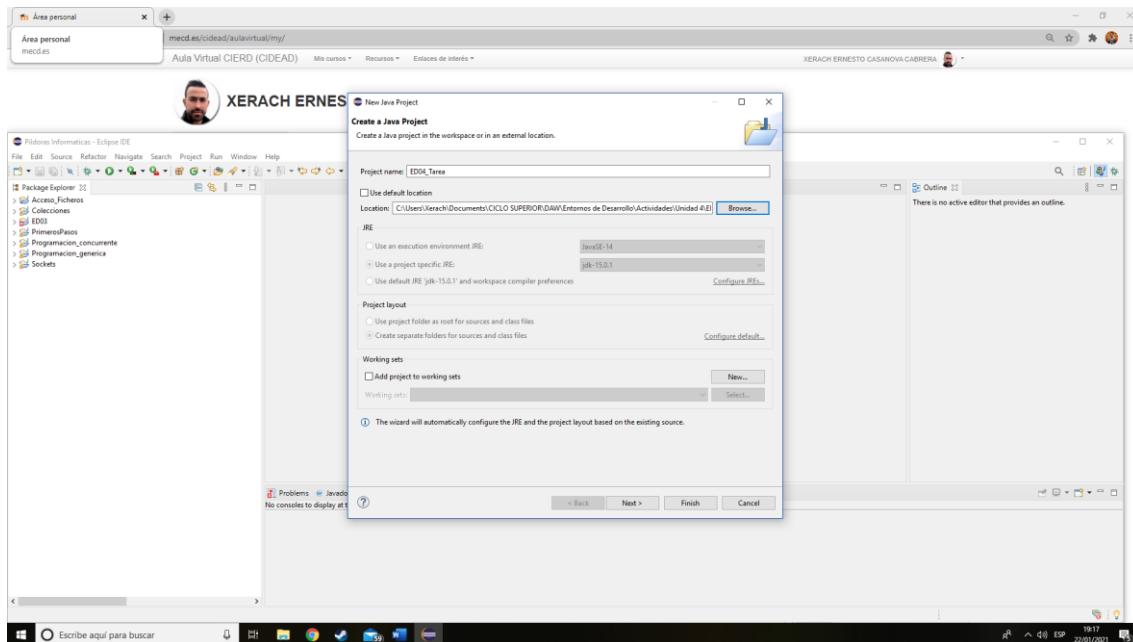
Pulsa aquí para descargar dicho proyecto ("Deposito.rar").

Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

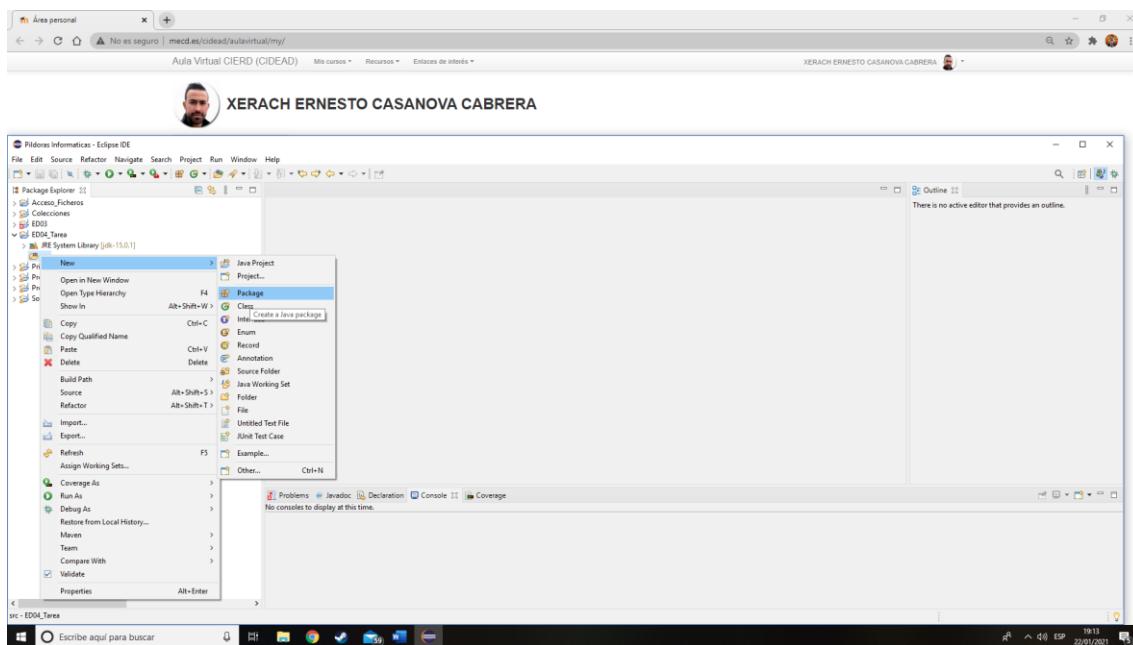
REFACTORIZACIÓN

1. Las clases deberán formar parte del paquete cuentas.

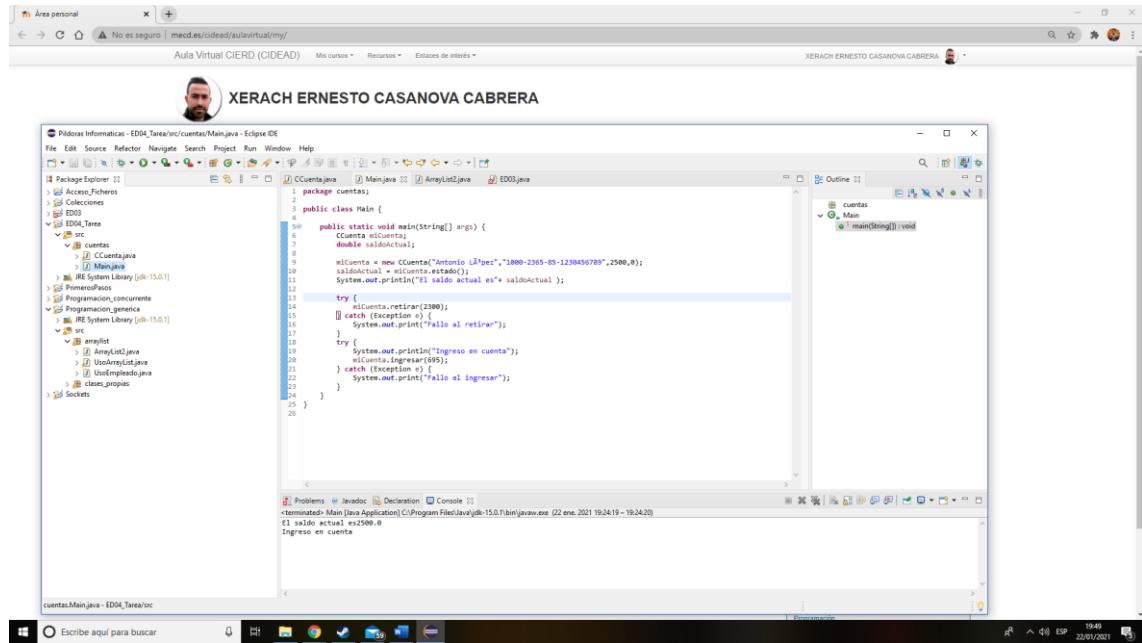
Abro Eclipse y me dirijo a File / New Project para crear un nuevo proyecto al que llamaré ED04_Tarea



Dentro del nuevo proyecto creo un nuevo paquete y lo llamo cuentas

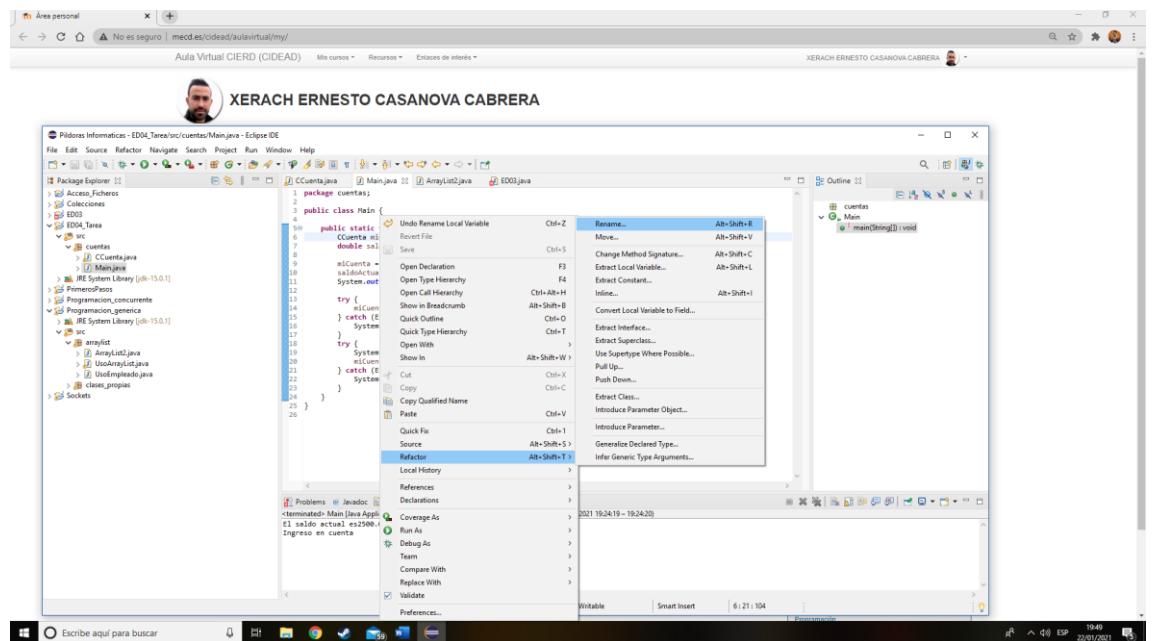


Seguidamente me dirijo a la ruta donde se encuentra el proyecto en mi ordenador y guardo los dos archivos .java en la carpeta cuentas. Saldrán automáticamente dentro del paquete cuentas en mi proyecto de Eclipse desde que refresque el explorador de proyectos. En los archivos CCuenta y Main debo añadir la instrucción package cuentas;

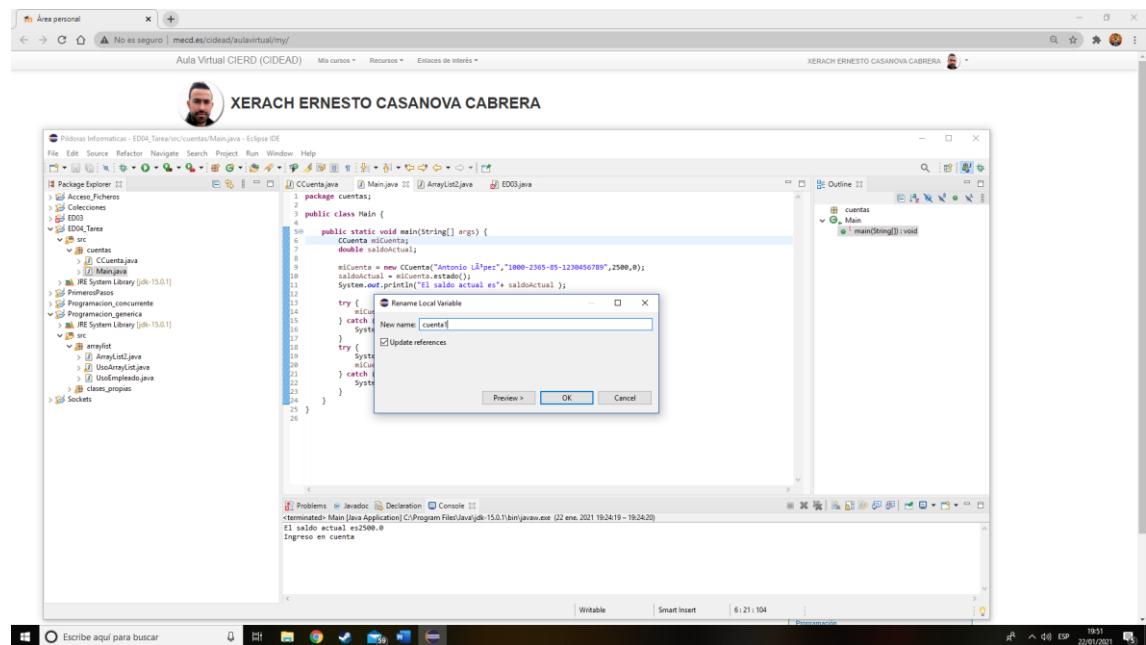


2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".

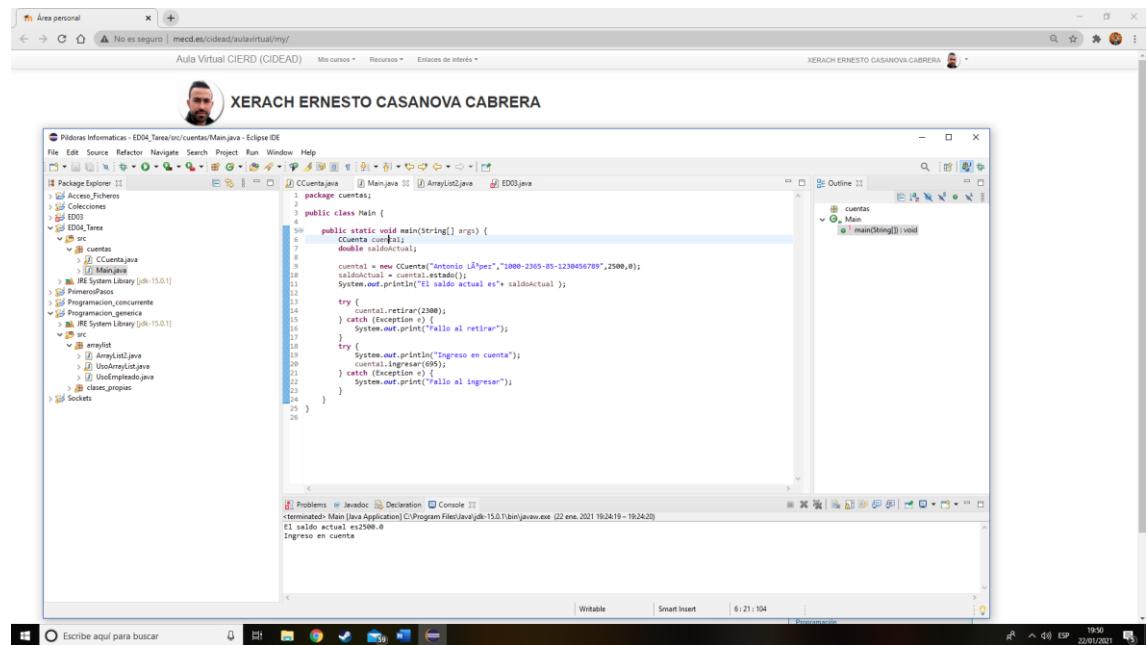
Hago click con el botón derecho en el nombre de la variable (objeto) en cualquier parte del código y seguidamente voy al menú “Refactor” y hago click en “Rename”.



Elijo el nuevo nombre de la variable y le doy a aceptar.

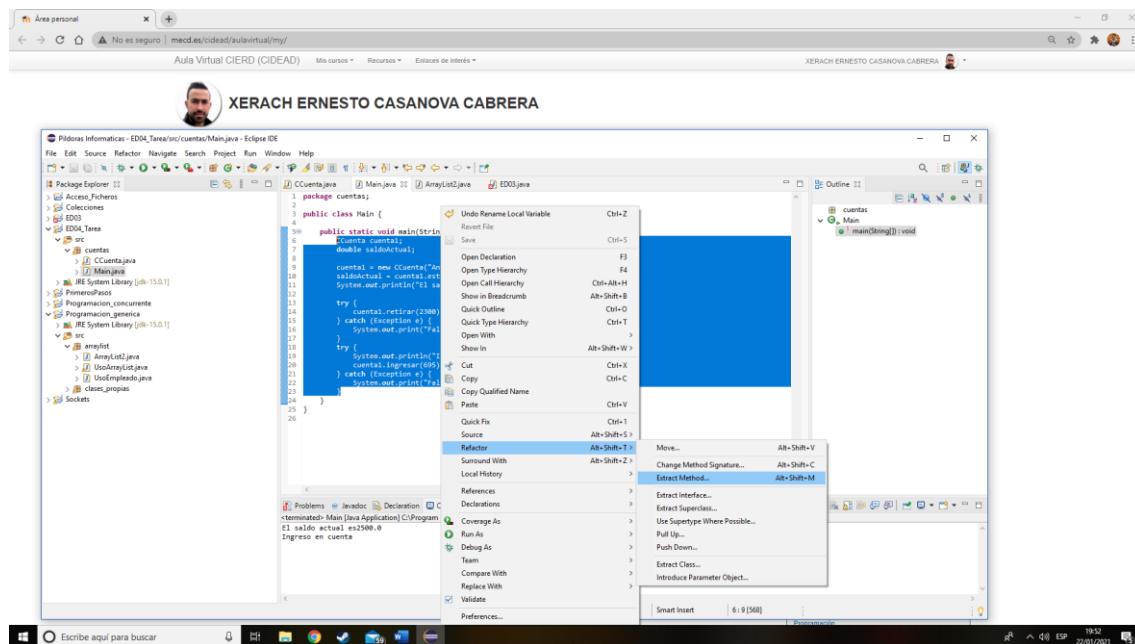


Compruebo que se ha modificado el nombre de la variable en todas y cada una de las partes del código en la que se hace referencia a ella.

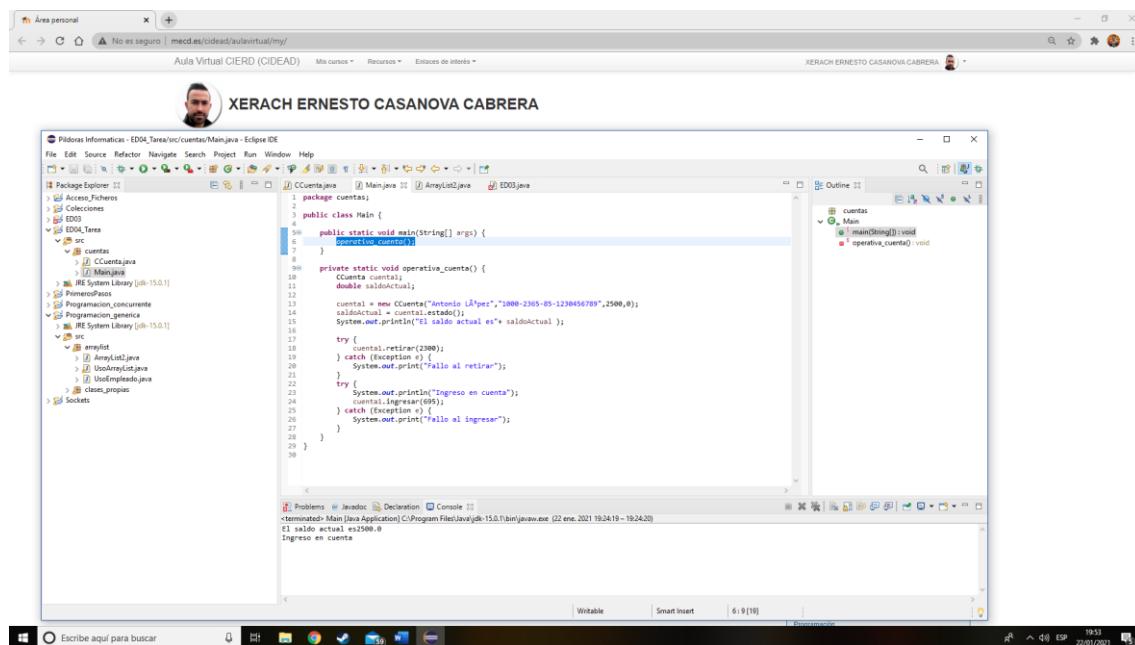


3. Introducir el método operativa_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.

Selecciono todo el bloque de código que está dentro del main y hago click derecho, me dirijo a refactor y hago click en extract method.

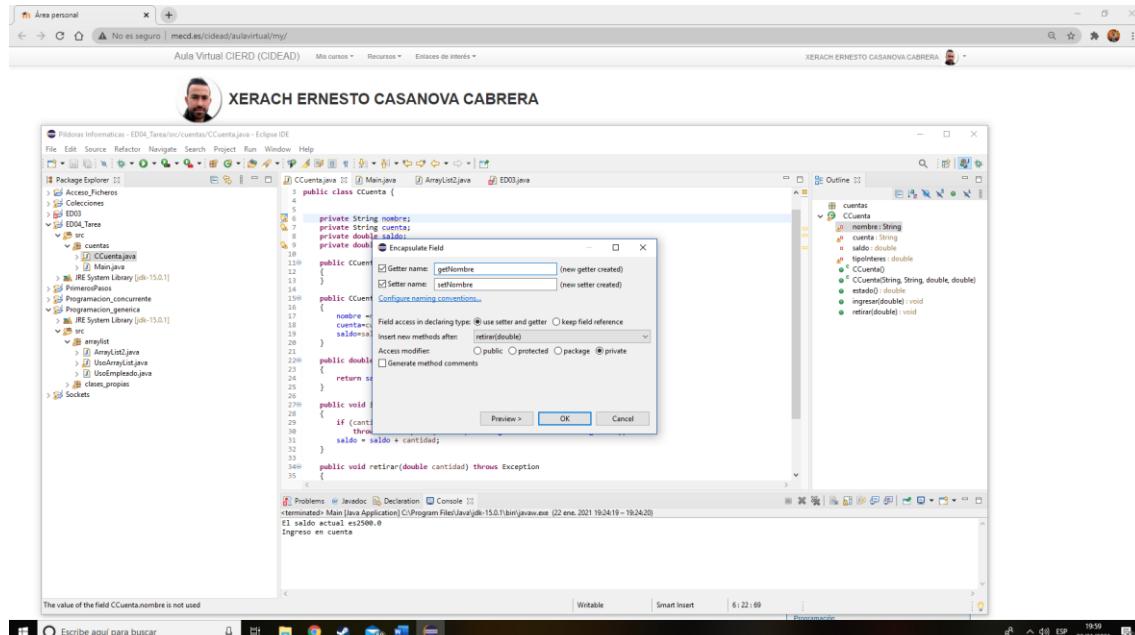
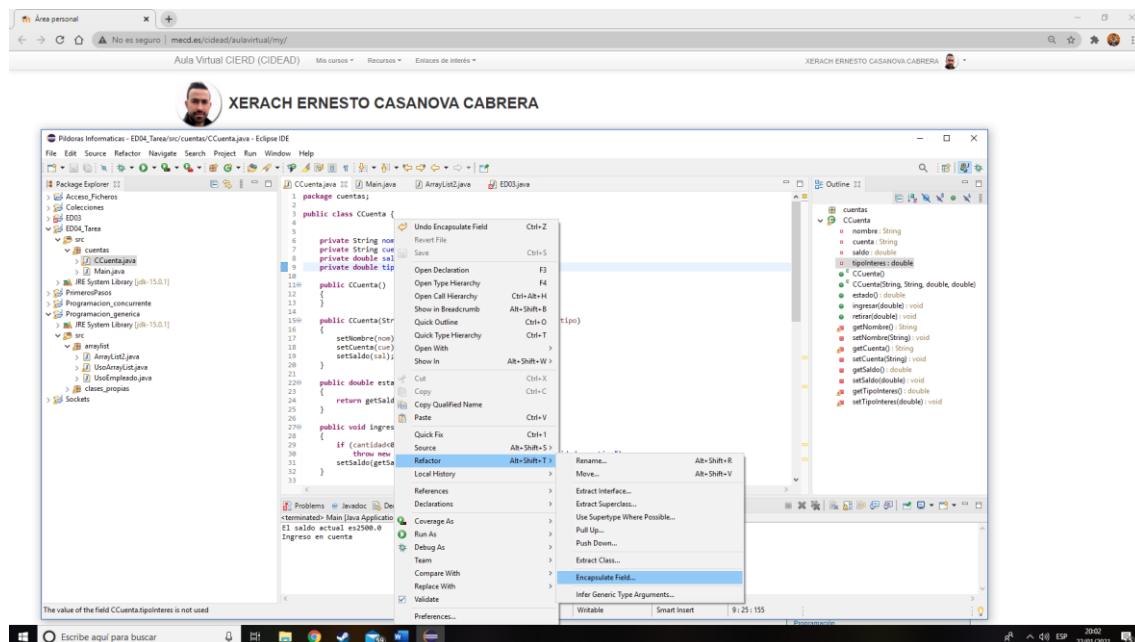


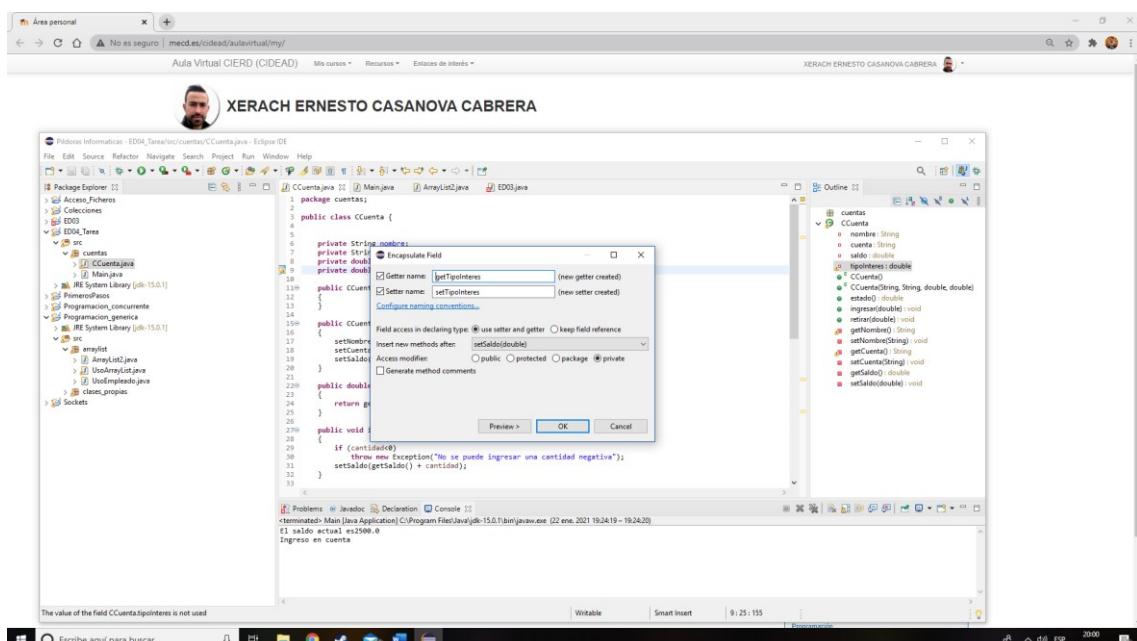
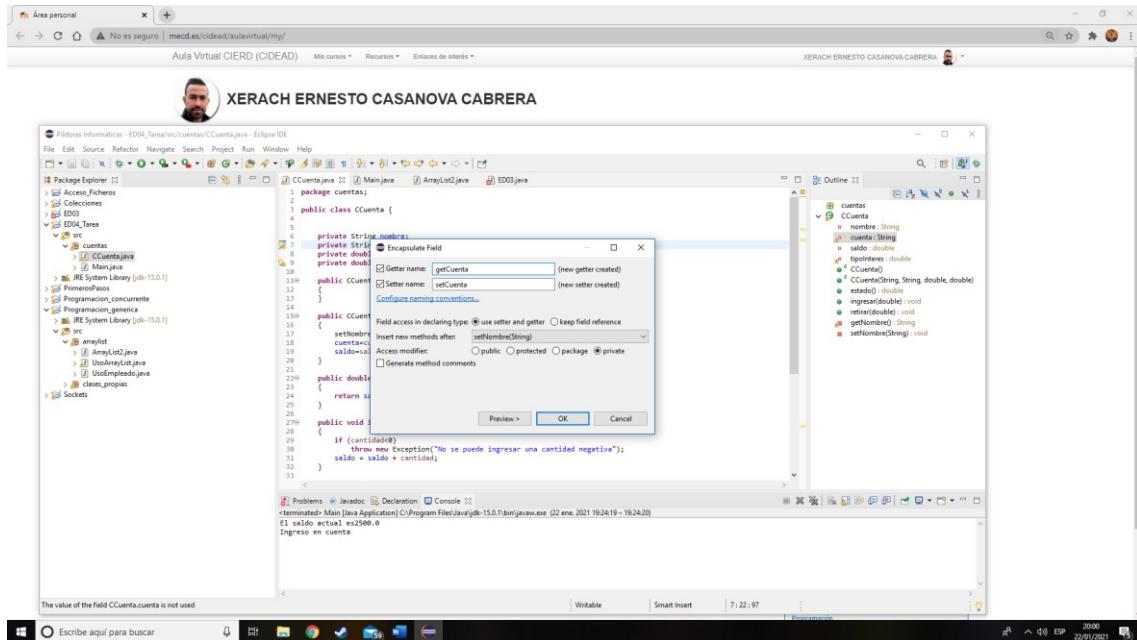
Una vez elegido el nombre del nuevo método, le doy a aceptar y compruebo que se ha realizado el cambio perfectamente.



4. Encapsular los atributos de la clase CCuenta.

Hago click derecho en cada una de los cuatro atributos de la clase CCuenta, seguidamente hago click en encapsulate field y en el cuadro de diálogo elijo en que parte del código deseo generar los métodos getter y setter





Área personal

No es seguro | mecdes/cidead/aulavirtual/my/

Aula Virtual CIERD (CIDEAD)

XERACH ERNESTO CASANOVA CABRERA

Píldoras Informáticas - E04_Tarea/src/cuentas/CCuenta.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Outline Problems Javadoc Declaration Console
cuentas Cuenta
  nombre: String
  cuenta: String
  saldo: double
  tipointeres: double
  Cuenta(String, String, double, double)
  Cuenta(double, double)
  ingresar(double) void
  retirar(double) void
  getNombre() String
  setNombre(String) void
  getSaldo() String
  setSaldo(String) void
Configure naming conventions...
Field access in declaring type: use setter and getter or keep field reference
Insert new methods after: setCuenta(String)
Access modifier: public protected package private
Generate method comments
Preview > OK Cancel

```

```

1 package cuentas;
2
3 public class CCuenta {
4
5     private String nombre;
6     private String cuenta;
7     private double saldo;
8     private double tipointeres;
9
10    public CCuenta() {
11        setNombre("Xerach");
12        setCuenta("1234567890");
13        setSaldo(1000);
14        setTipointeres(0.05);
15    }
16
17    public CCuenta(String nombre, String cuenta, double saldo, double tipointeres) {
18        setNombre(nombre);
19        setCuenta(cuenta);
20        setSaldo(saldo);
21        setTipointeres(tipointeres);
22    }
23
24    public double getSaldo() {
25        return saldo;
26    }
27
28    public void setSaldo(double cantidad) {
29        if (cantidad<0)
30            throw new Exception("No se puede ingresar una cantidad negativa");
31        saldo = saldo + cantidad;
32    }
33
34    public void ingresar(double cantidad) {
35        saldo += cantidad;
36    }
37
38    public void retirar(double cantidad) {
39        if (cantidad>saldo)
40            throw new Exception("No se puede retirar una cantidad mayor que el saldo actual");
41        saldo -= cantidad;
42    }
43
44    public String getNombre() {
45        return nombre;
46    }
47
48    public void setNombre(String nombre) {
49        this.nombre = nombre;
50    }
51
52    public String getCuenta() {
53        return cuenta;
54    }
55
56    public void setCuenta(String cuenta) {
57        this.cuenta = cuenta;
58    }
59
60    public double getSaldo() {
61        return saldo;
62    }
63
64    public void setSaldo(double saldo) {
65        this.saldo = saldo;
66    }
67
68    public double getTipointeres() {
69        return tipointeres;
70    }
71
72    public void setTipointeres(double tipointeres) {
73        this.tipointeres = tipointeres;
74    }
75
76}

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jdb-15.0.1\bin\javaw.exe (22 ene. 2021 19:24:19 - 19:24:20)

El saldo actual es:2500.0

Ingreso en cuenta

Writable Smart Insert 8:23:126

2000 22/01/2021

Finalmente compruebo que se han creado correctamente.

Área personal

No es seguro | mecdes/cidead/aulavirtual/my/

Aula Virtual CIERD (CIDEAD)

XERACH ERNESTO CASANOVA CABRERA

Píldoras Informáticas - E04_Tarea/src/cuentas/CCuenta.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Outline Problems Javadoc Declaration Console
cuentas Cuenta
  nombre: String
  cuenta: String
  saldo: double
  tipointeres: double
  Cuenta(String, String, double, double)
  Cuenta(double, double)
  ingresar(double) void
  retirar(double) void
  getNombre() String
  setNombre(String) void
  getSaldo() String
  setSaldo(String) void
The value of the field CCuenta.tipointeres is not used
Configure naming conventions...
Field access in declaring type: use setter and getter or keep field reference
Insert new methods after: setCuenta(String)
Access modifier: public protected package private
Generate method comments
Preview > OK Cancel

```

```

1 package cuentas;
2
3 public class CCuenta {
4
5     private String nombre;
6     private String cuenta;
7     private double saldo;
8     private double tipointeres;
9
10    public CCuenta() {
11        setNombre("Xerach");
12        setCuenta("1234567890");
13        setSaldo(1000);
14        setTipointeres(0.05);
15    }
16
17    public CCuenta(String nombre, String cuenta, double saldo, double tipointeres) {
18        setNombre(nombre);
19        setCuenta(cuenta);
20        setSaldo(saldo);
21        setTipointeres(tipointeres);
22    }
23
24    public double getSaldo() {
25        return saldo;
26    }
27
28    public void setSaldo(double cantidad) {
29        if (cantidad<0)
30            throw new Exception("No se puede ingresar una cantidad negativa");
31        saldo += cantidad;
32    }
33
34    public void ingresar(double cantidad) {
35        saldo += cantidad;
36    }
37
38    public void retirar(double cantidad) {
39        if (cantidad>saldo)
40            throw new Exception("No se puede retirar una cantidad mayor que el saldo actual");
41        saldo -= cantidad;
42    }
43
44    public String getNombre() {
45        return nombre;
46    }
47
48    public void setNombre(String nombre) {
49        this.nombre = nombre;
50    }
51
52    public String getCuenta() {
53        return cuenta;
54    }
55
56    public void setCuenta(String cuenta) {
57        this.cuenta = cuenta;
58    }
59
60    public double getSaldo() {
61        return saldo;
62    }
63
64    public void setSaldo(double saldo) {
65        this.saldo = saldo;
66    }
67
68    public double getTipointeres() {
69        return tipointeres;
70    }
71
72    public void setTipointeres(double tipointeres) {
73        this.tipointeres = tipointeres;
74    }
75
76}

```

Problems Javadoc Declaration Console

<terminated> Main [Java Application] C:\Program Files\Java\jdb-15.0.1\bin\javaw.exe (22 ene. 2021 19:24:19 - 19:24:20)

El saldo actual es:2500.0

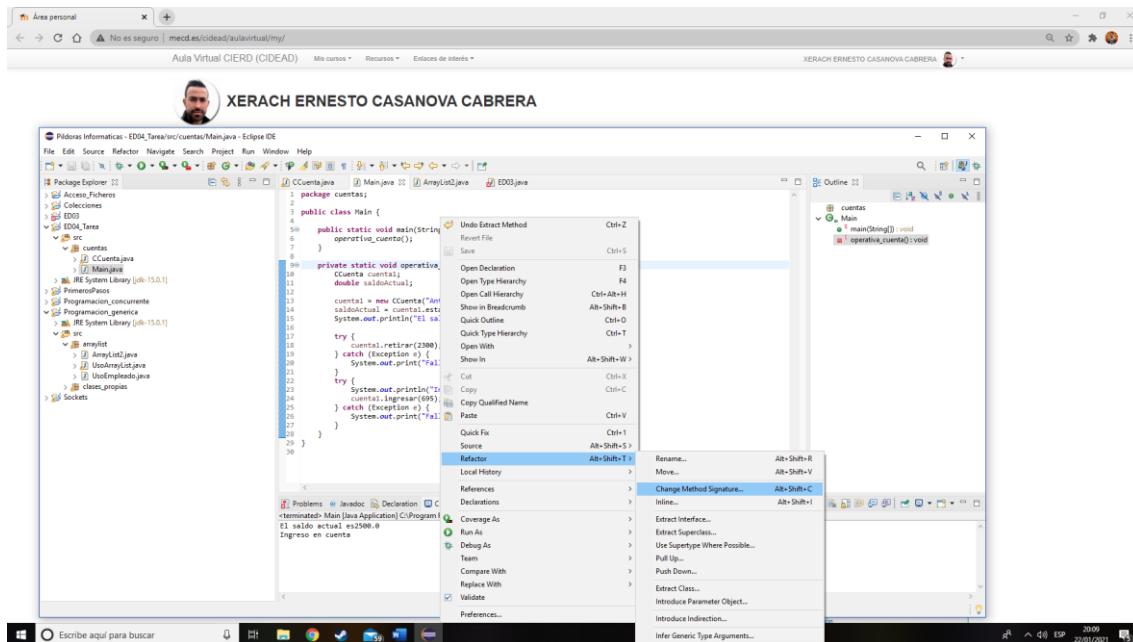
Ingreso en cuenta

Writable Smart Insert 9:25:155

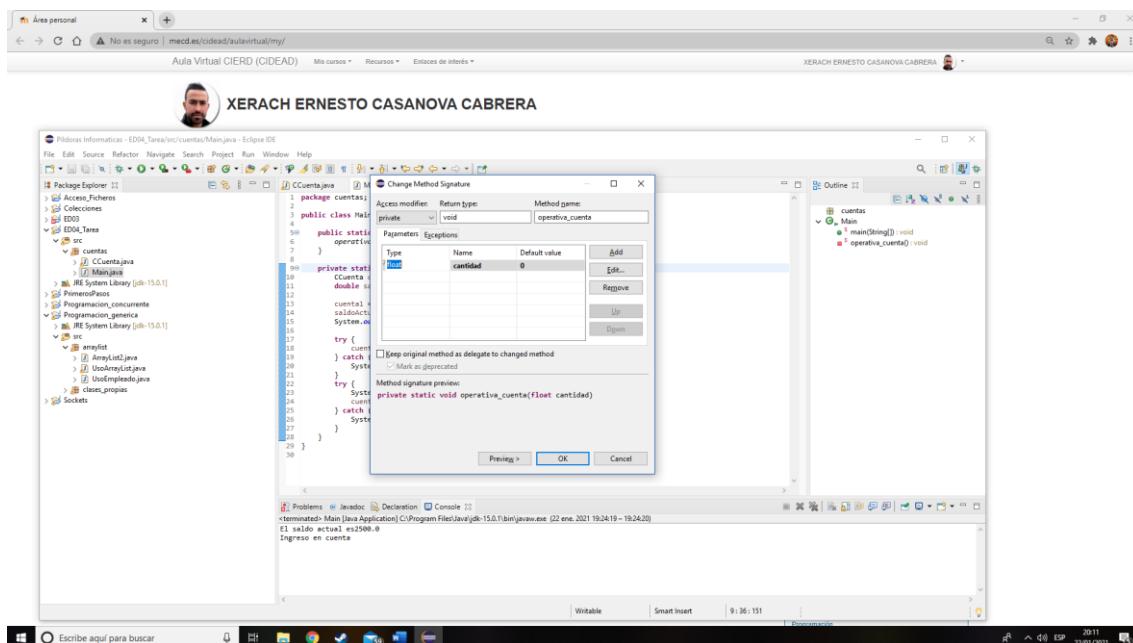
2004 22/01/2021

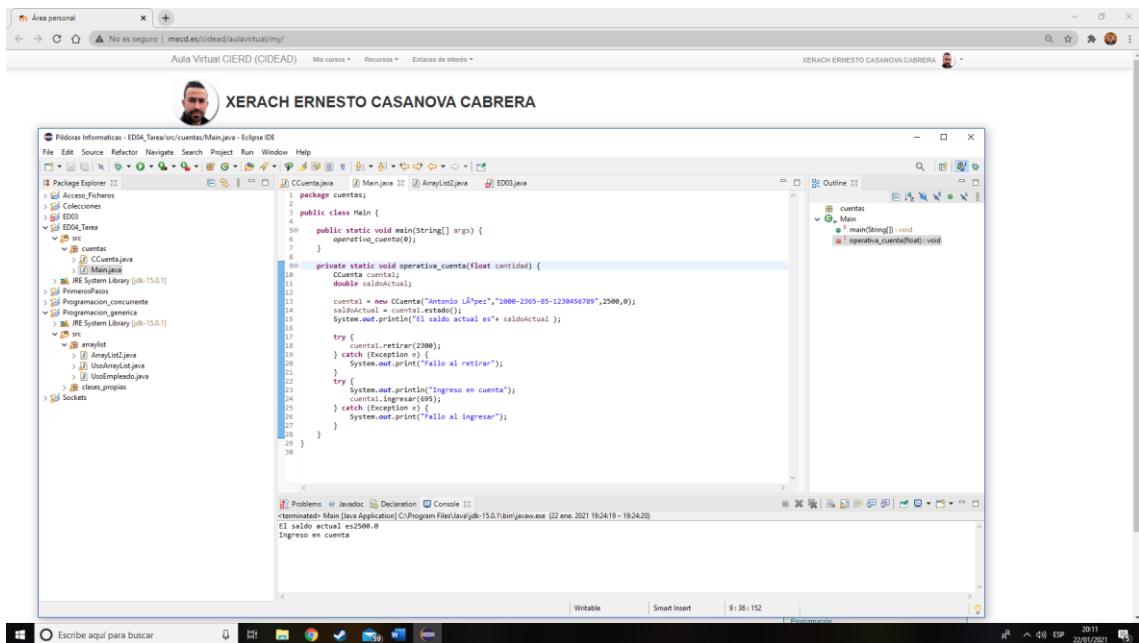
5. Añadir un nuevo parámetro al método operativa_cuenta, de nombre cantidad y de tipo float.

Hago click derecho en la cabecera del método operativa_cuenta y me dirijo a refactor/change method signature



Añado un parámetro de tipo float, de nombre cantidad y el valor por defecto que se utilizará en todas las referencias a ese método será 0.

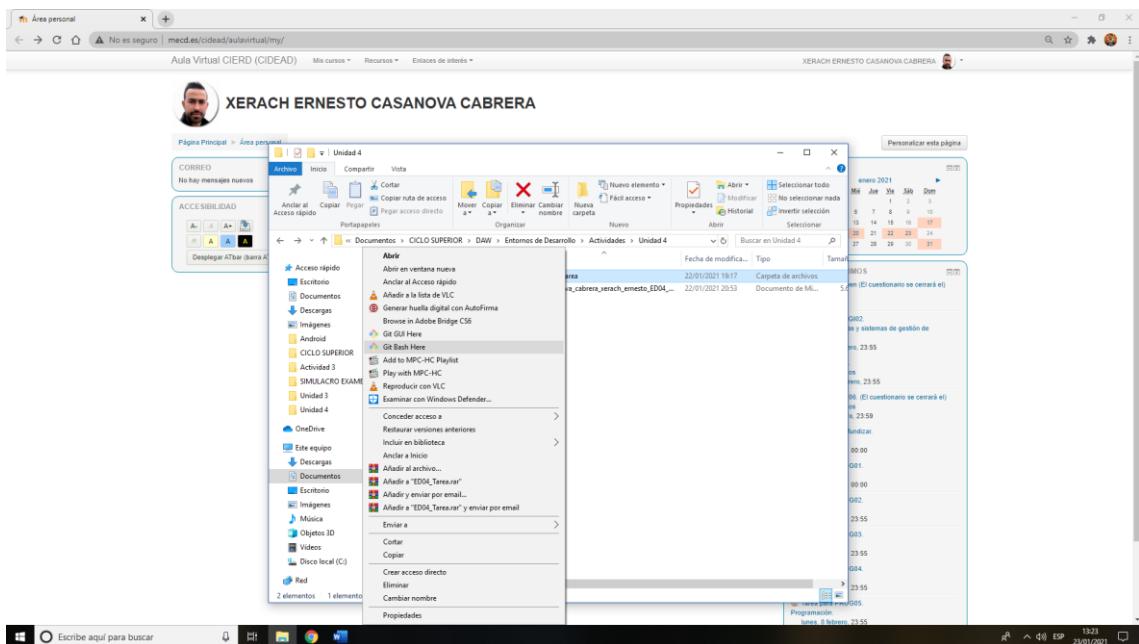




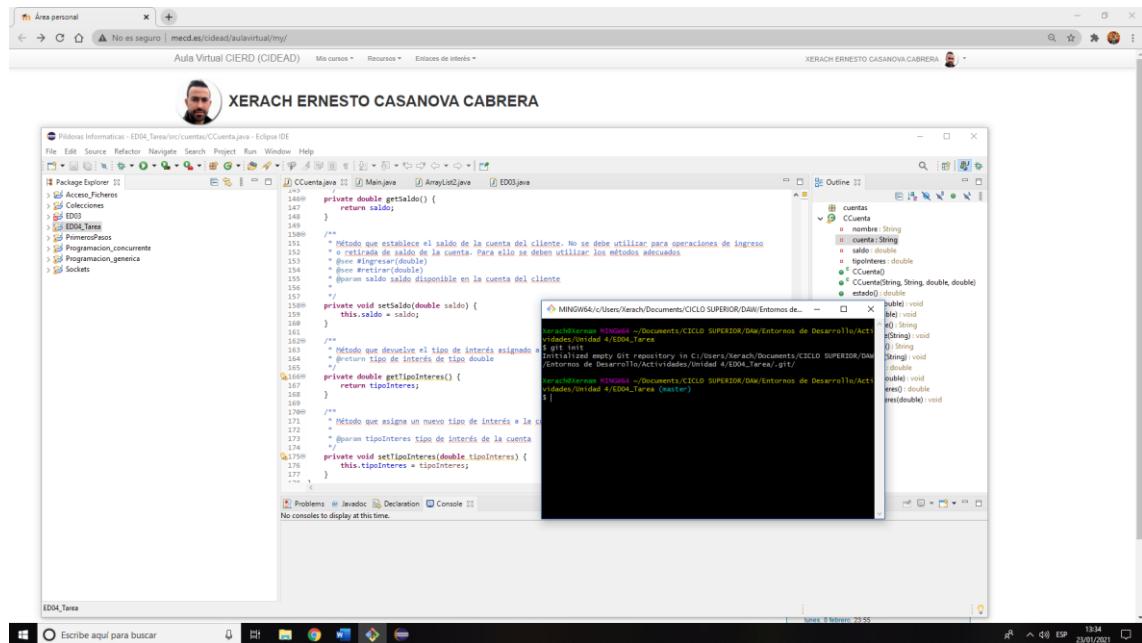
GIT

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

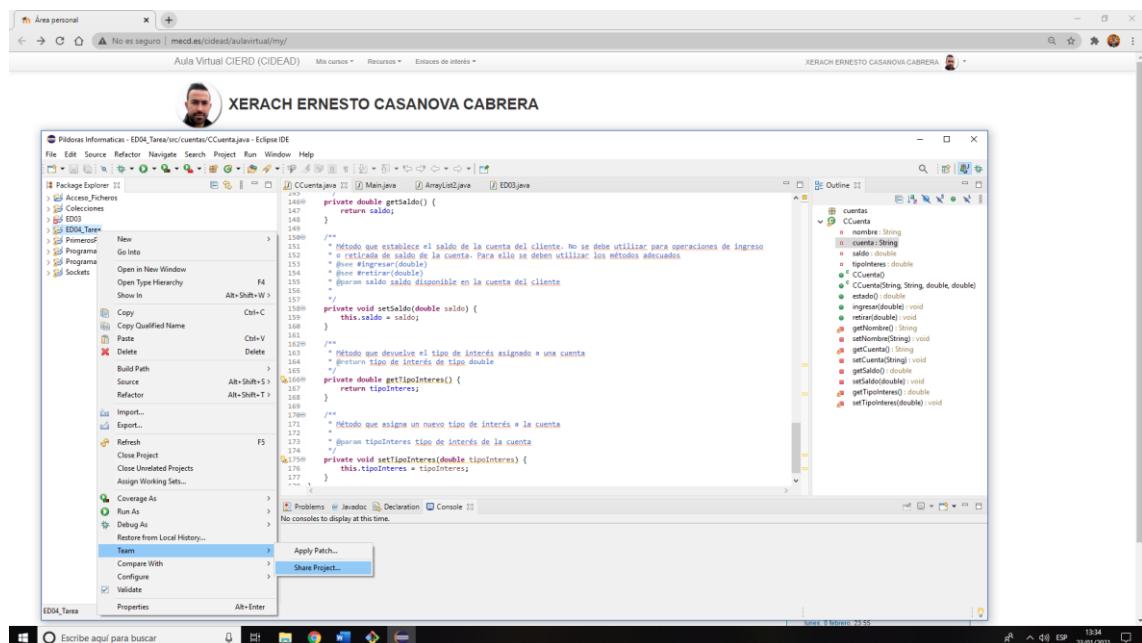
Hago click derecho encima de la carpeta del proyecto y seguidamente hago click en Git Bash here para dirigirme a la carpeta del proyecto en GIT.

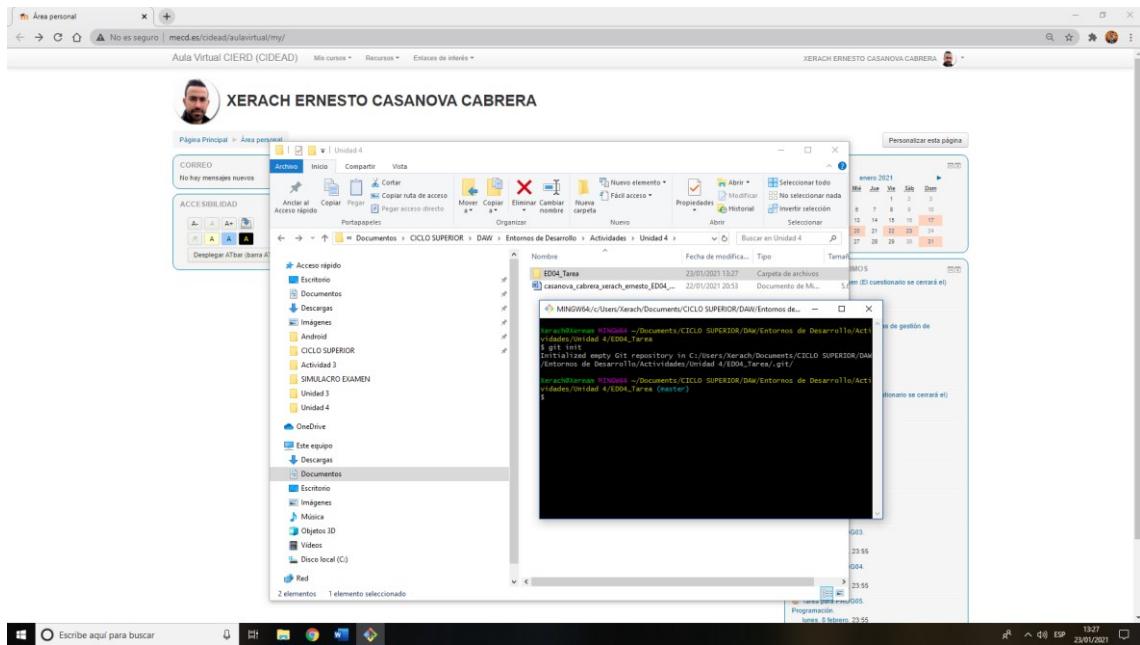
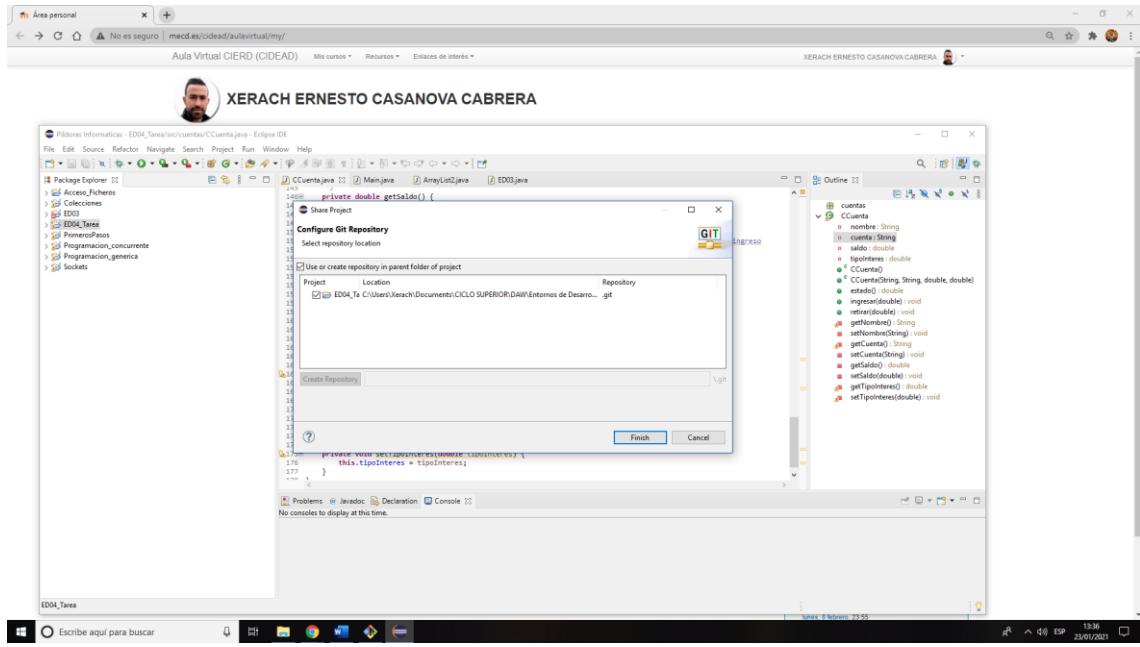


Ejecuto el comando git init para crear el repositorio.

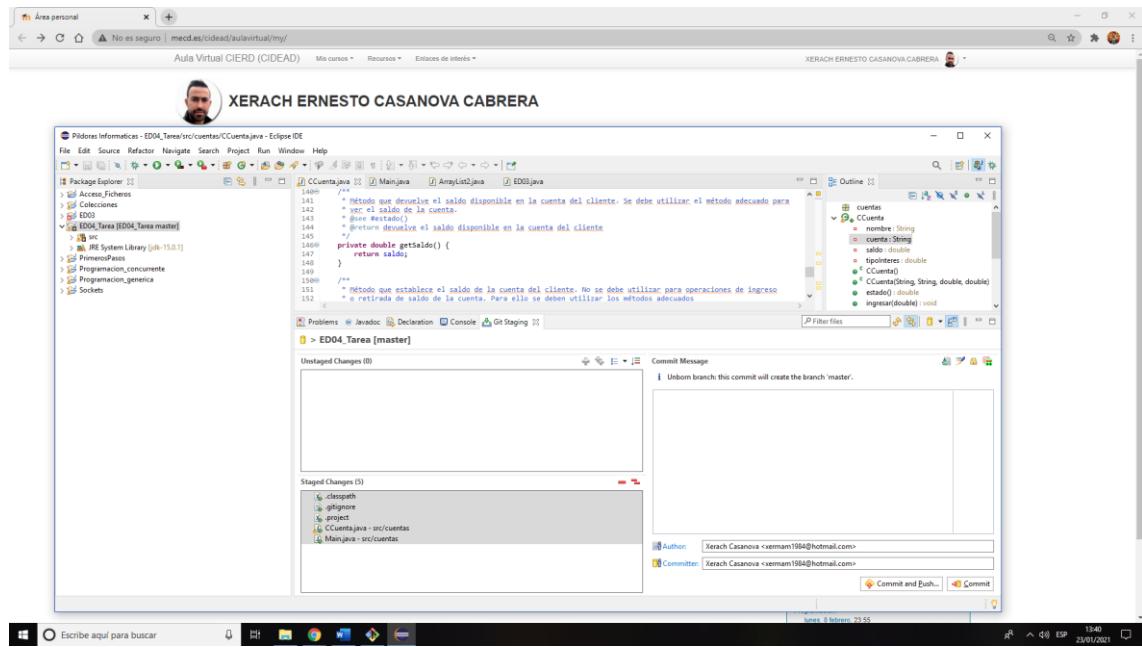


Me dirijo a eclipse y haciendo click derecho en el proyecto y seguidamente en Team/share Project.

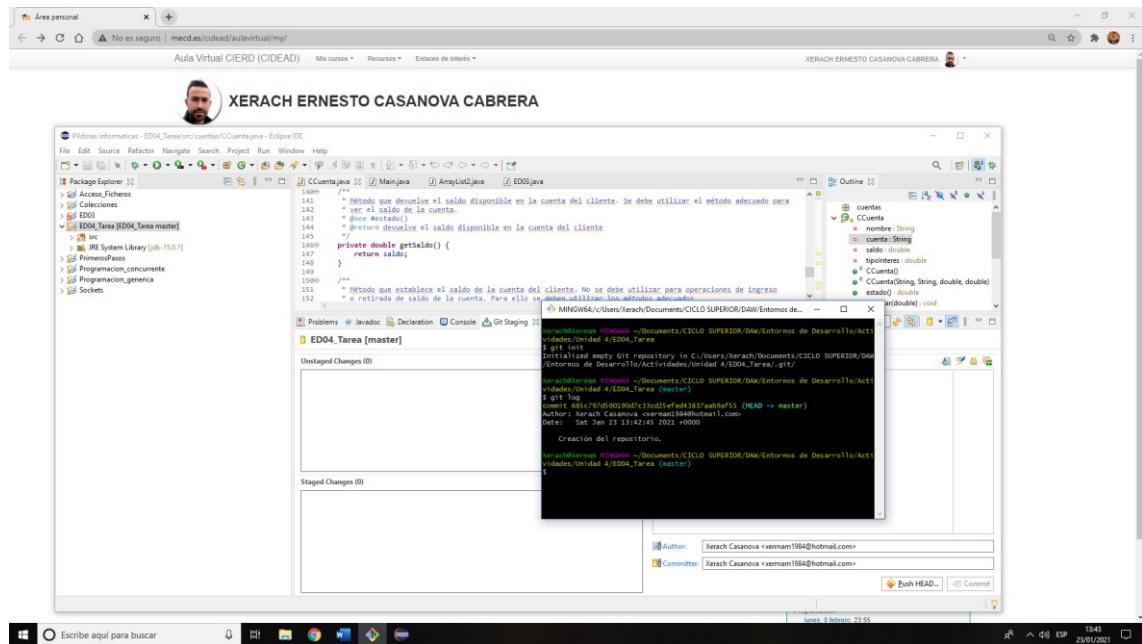




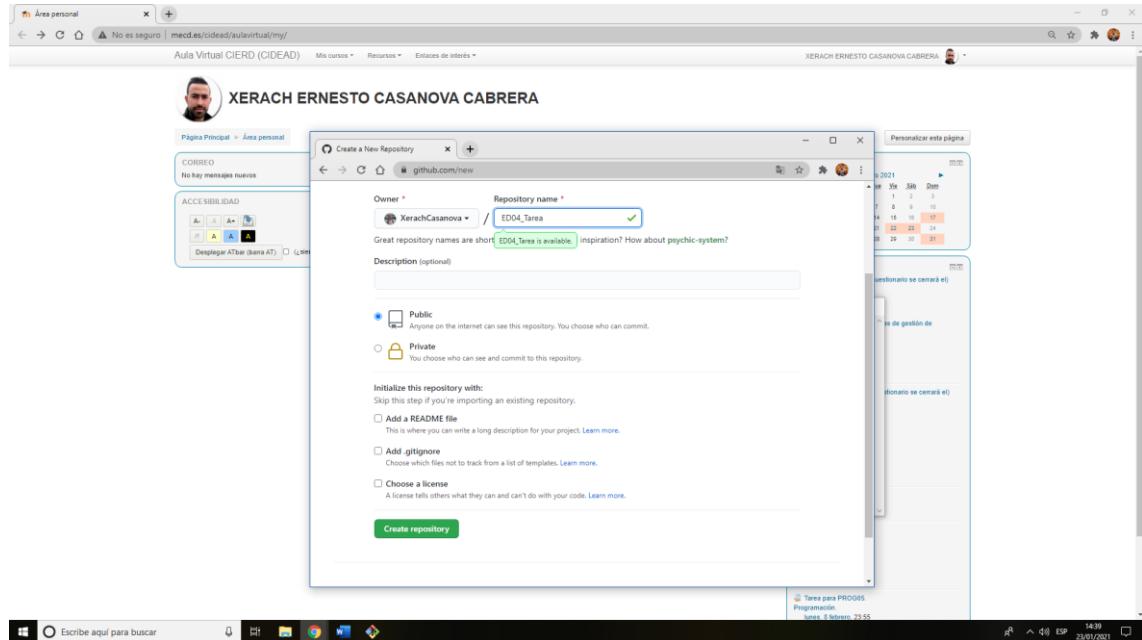
Selecciono los archivos que formarán parte del repositorio.



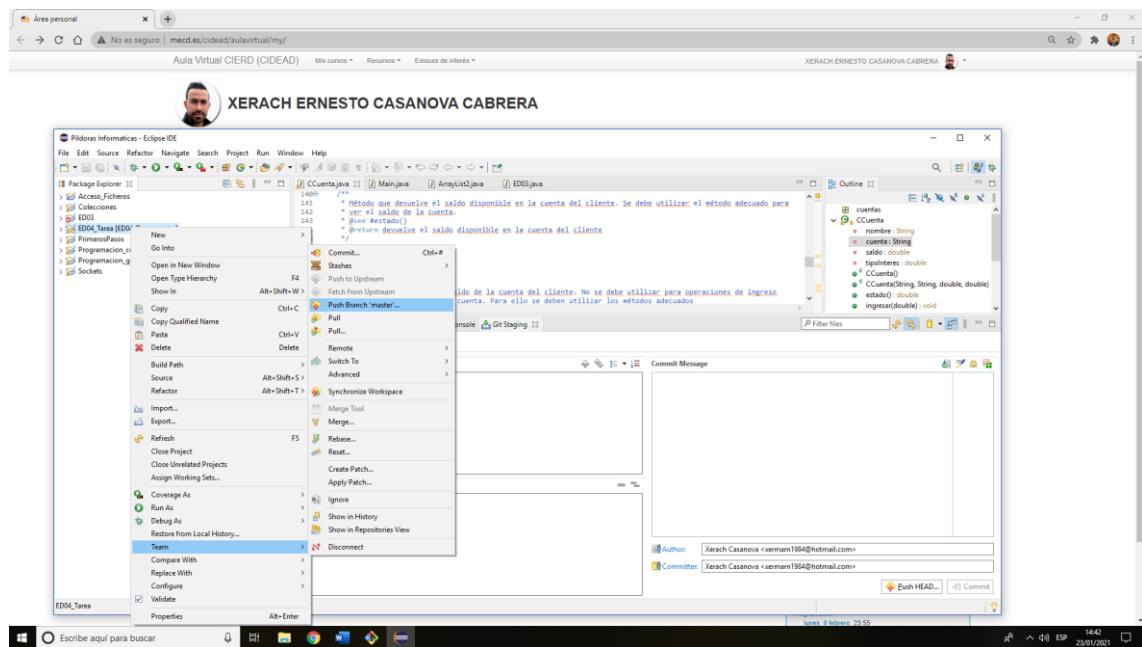
Compruebo desde consola que he lanzado el primer commit en GIT



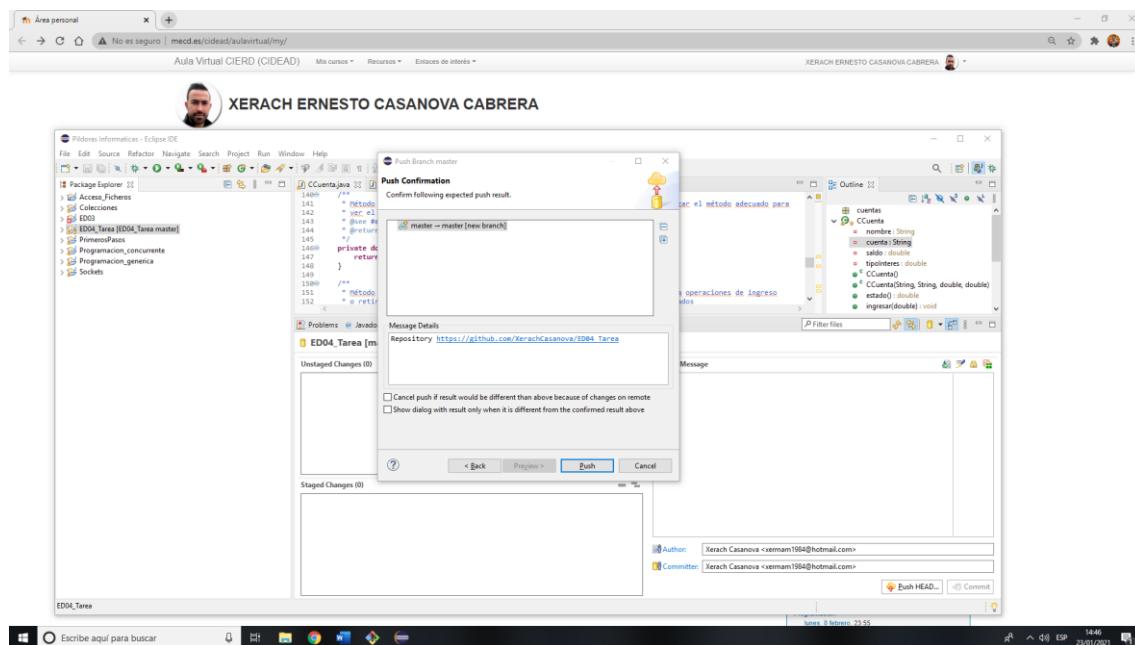
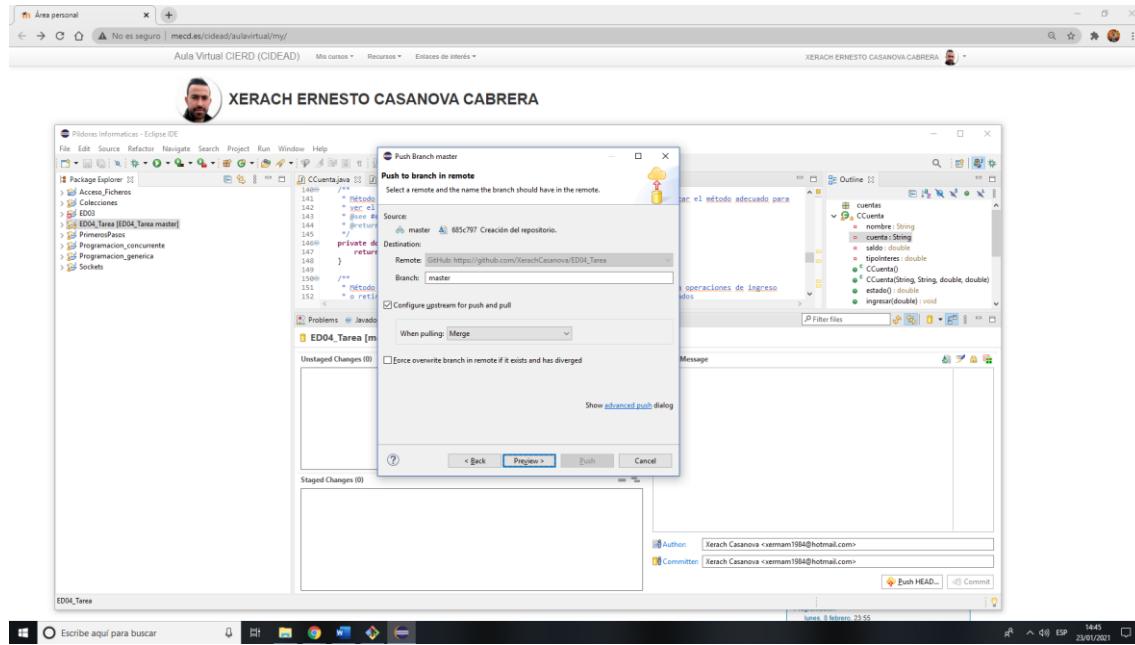
Me dirijo a GitHub y creo un nuevo repositorio. Se puede visitar en:
https://github.com/XerachCasanova/ED04_Tarea

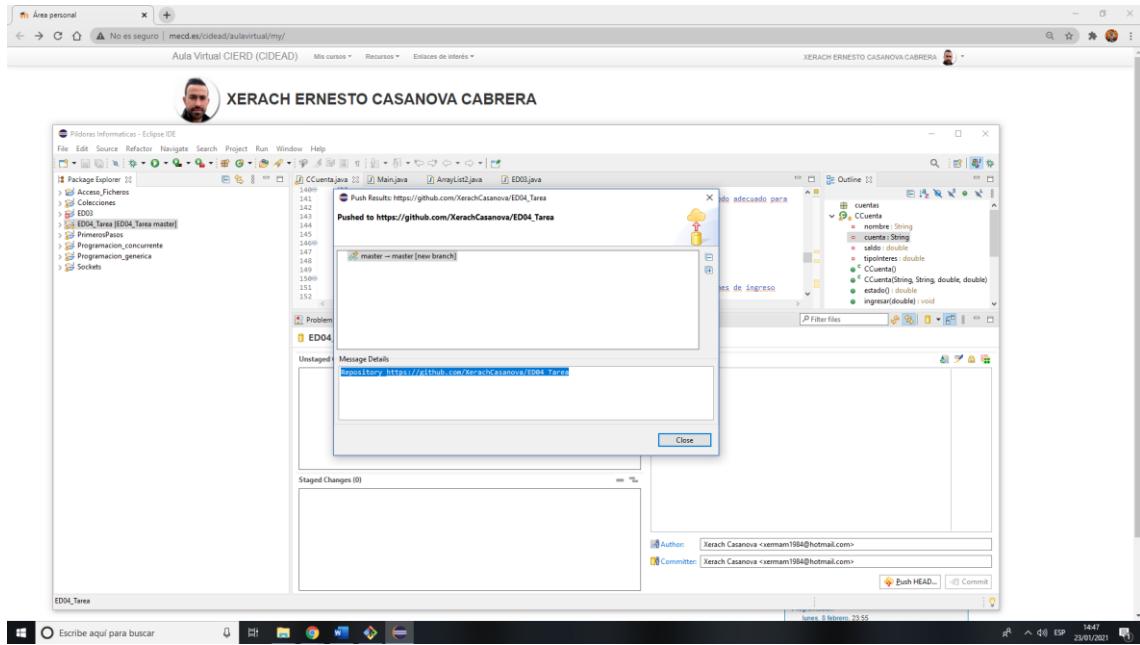


Seguidamente vuelvo a Eclipse y haciendo click derecho en el proyecto, me dirijo a Team/Push Branch Master

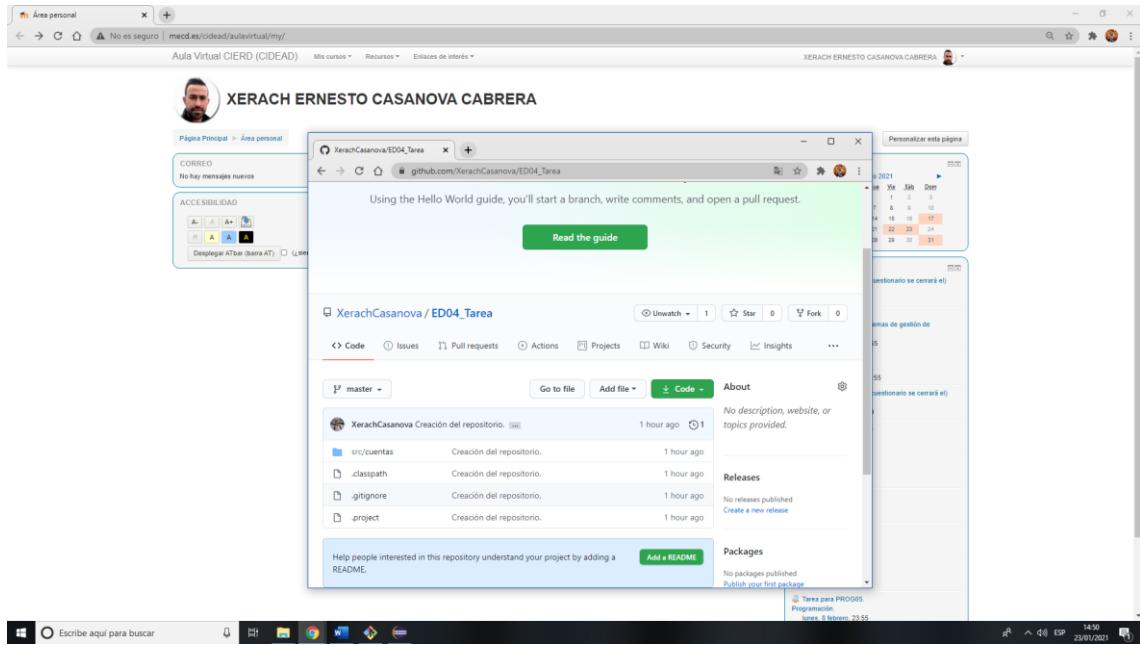


Configuro el repositorio de GitHub para sincronizarlo con el repositorio de mi proyecto.



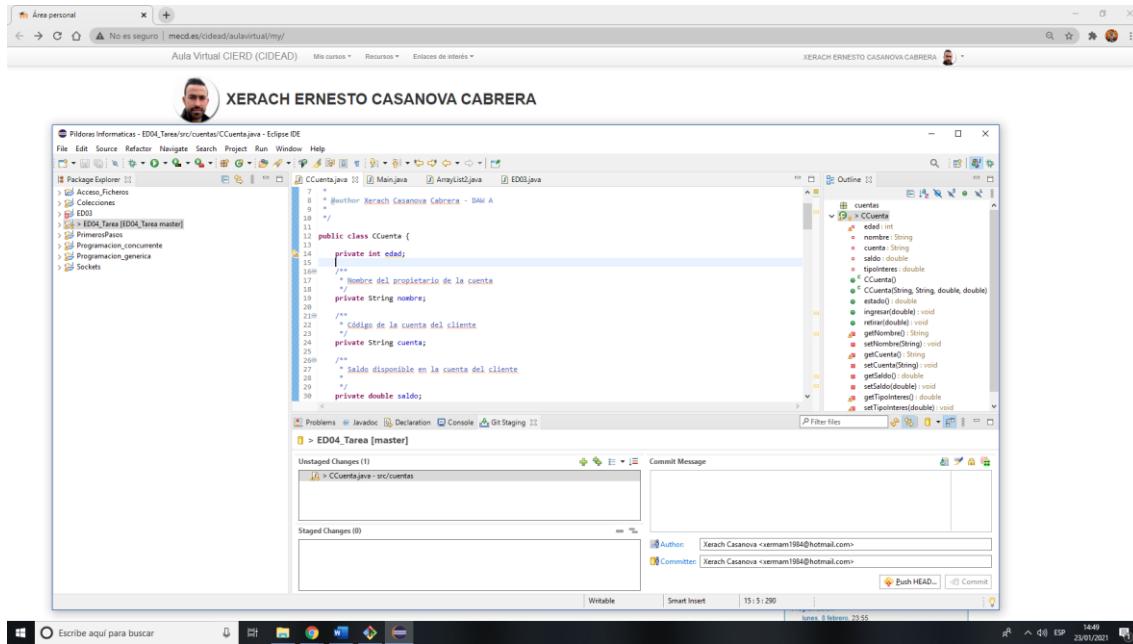


Una vez configurado me dirijo a GitHub y compruebo que se ha hecho un push de todo mi repositorio en local.

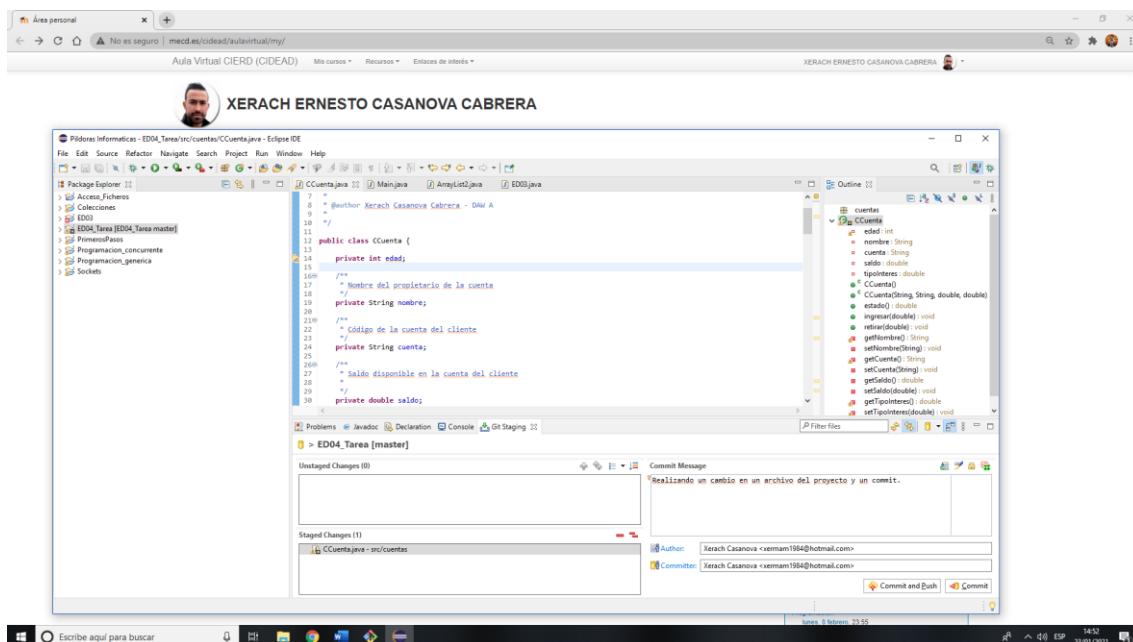


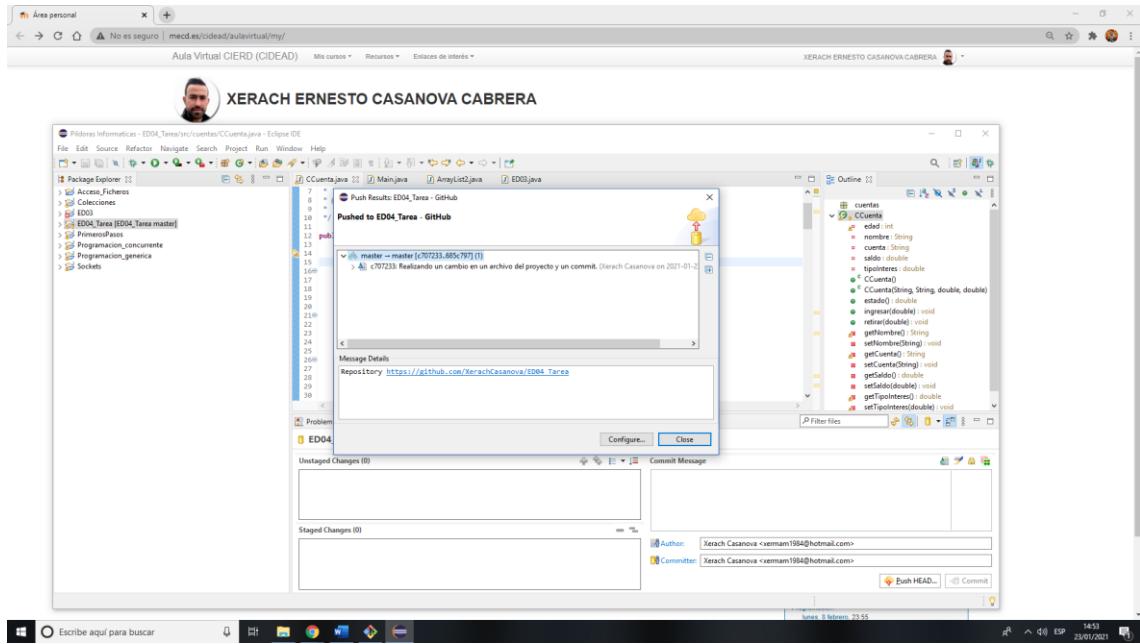
2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.

Para realizar un commit, primero hago algún cambio en cualquier fichero, he creado una variable para probar, llamada edad y he guardado cambios. Se puede observar como aparecen los ficheros modificados en la ventana inferior.



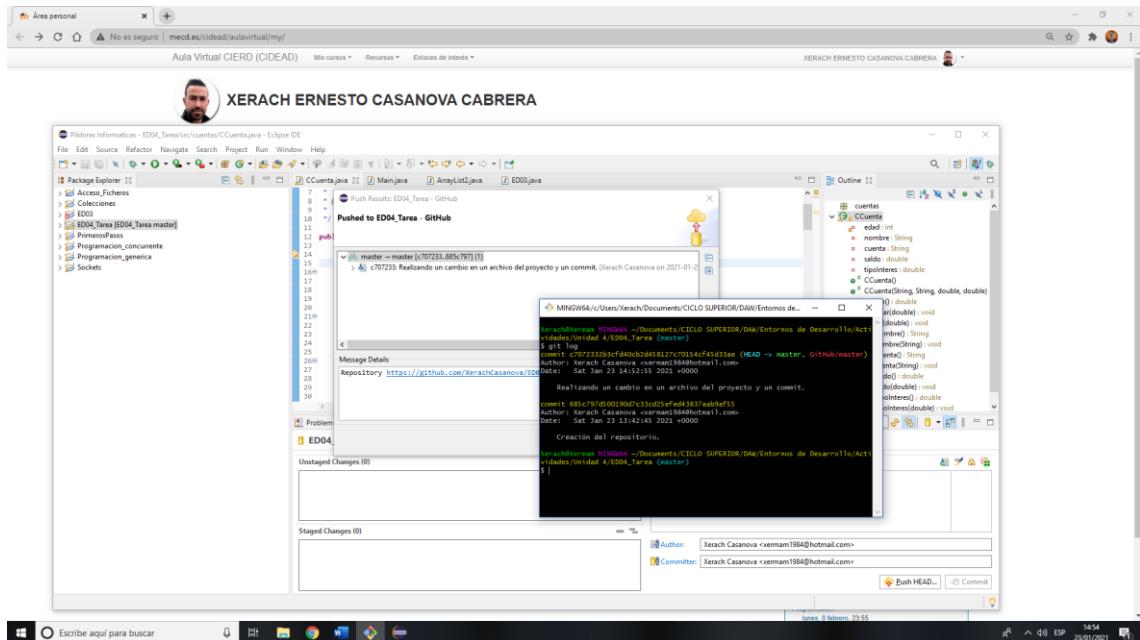
Añado el archivo para realizar un nuevo commit. En este caso, haré click en push, para además de hacer el commit, se sincronice con mi repositorio en GitHub.





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

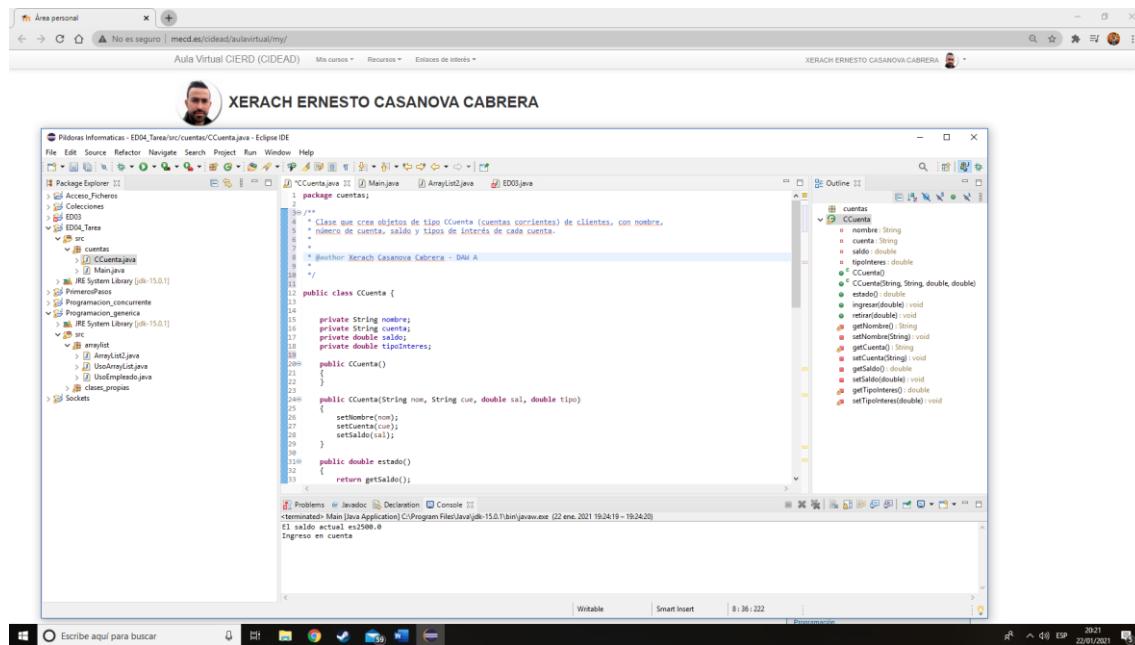
Me dirijo a la consola y ejecuto el comando git log para ver el historial de versiones del proyecto.



JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.

Genero comentario JavaDoc para la clase CCuenta.



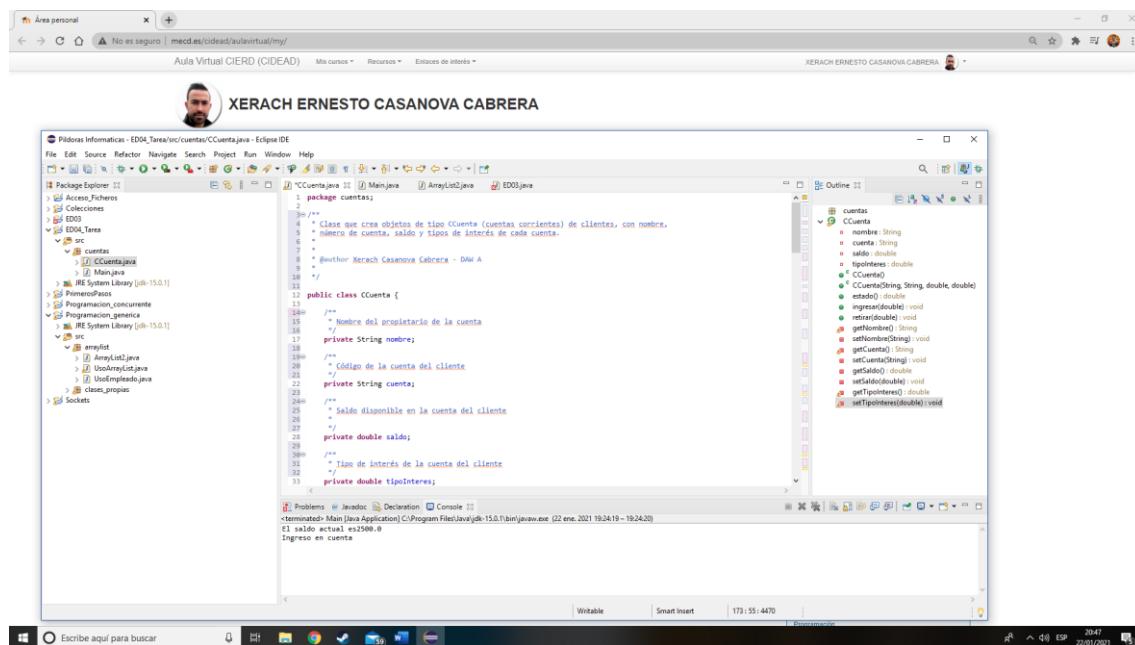
The screenshot shows the Eclipse IDE interface with the package explorer, outline, and code editor. The code editor contains the following Java code for the CCuenta class:

```
1 package cuentas;
2
3 /**
4 * Clase que crea objetos de tipo CCuenta (cuentas corrientes) de clientes, con nombre,
5 * número de cuenta, saldo y tipos de interés de cada cuenta.
6 *
7 * @author Xerach Casanova Cabrera - DAI A
8 */
9
10 public class CCuenta {
11
12     private String nombre;
13     private String cuenta;
14     private double saldo;
15     private double tipoInteres;
16
17     public CCuenta() {
18
19     }
20
21     public CCuenta(String nom, String cue, double sal, double tipo) {
22
23         setNombre(nom);
24         setCuenta(cue);
25         setSaldo(sal);
26     }
27
28     public double estados() {
29
30         return getSaldo();
31     }
32
33     /**
34      * Setters y getters
35      */
36     private void setNombre(String nom) {
37         nombre = nom;
38     }
39
40     private void setCuenta(String cue) {
41         cuenta = cue;
42     }
43
44     private void setSaldo(double sal) {
45         saldo = sal;
46     }
47
48     private void setTipoInteres(double tipo) {
49         tipoInteres = tipo;
50     }
51
52     /**
53      * Métodos adicionales
54      */
55     public void ingresar(double cantidad) {
56         saldo += cantidad;
57     }
58
59     public void retirar(double cantidad) {
60         saldo -= cantidad;
61     }
62
63     public String getNombre() {
64         return nombre;
65     }
66
67     public String getCuenta() {
68         return cuenta;
69     }
70
71     public double getSaldo() {
72         return saldo;
73     }
74
75     public double getTipoInteres() {
76         return tipoInteres;
77     }
78
79     public void setTiposInteres(double tipo) {
80         tipoInteres = tipo;
81     }
82
83     /**
84      * Método para obtener el saldo disponible en la cuenta del cliente
85      */
86     public double saldoDisponible() {
87         return saldo;
88     }
89
90     /**
91      * Método para obtener el tipo de interés de la cuenta del cliente
92      */
93     public double tipoInteres() {
94         return tipoInteres;
95     }
96 }
```

The code editor shows annotations for JavaDoc comments, such as `@author` and `@param`. The right-hand side of the interface displays the generated JavaDoc documentation for the CCuenta class, listing its attributes and methods with their descriptions.

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

Genero comentarios JavaDoc para los nombres de atributos, aunque no son necesarios y no se suelen incluir, al pedirlos el ejercicio los añado igualmente.

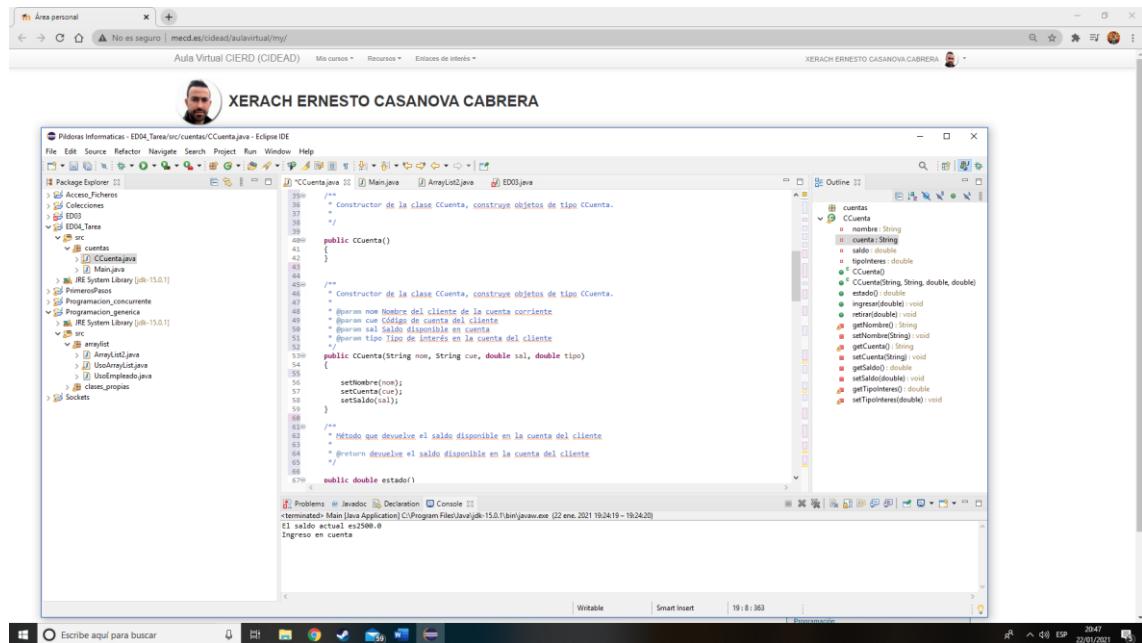


The screenshot shows the Eclipse IDE interface with the package explorer, outline, and code editor. The code editor contains the following Java code for the CCuenta class, with more detailed JavaDoc comments than in the previous screenshot:

```
1 package cuentas;
2
3 /**
4 * Clase que crea objetos de tipo CCuenta (cuentas corrientes) de clientes, con nombre,
5 * número de cuenta, saldo y tipos de interés de cada cuenta.
6 *
7 * @author Xerach Casanova Cabrera - DAI A
8 */
9
10 public class CCuenta {
11
12     /**
13      * Nombre del propietario de la cuenta
14      */
15     private String nombre;
16
17     /**
18      * Código de la cuenta del cliente
19      */
20     private String cuenta;
21
22     /**
23      * Saldo disponible en la cuenta del cliente
24      */
25     private double saldo;
26
27     /**
28      * Tipo de interés de la cuenta del cliente
29      */
30     private double tipoInteres;
31
32     /**
33      * Saldo disponible en la cuenta del cliente
34      */
35     public double saldoDisponible() {
36         return saldo;
37     }
38
39     /**
40      * Tipo de interés de la cuenta del cliente
41      */
42     public double tipoInteres() {
43         return tipoInteres;
44     }
45
46     /**
47      * Setters y getters
48      */
49     private void setNombre(String nom) {
50         nombre = nom;
51     }
52
53     private void setCuenta(String cue) {
54         cuenta = cue;
55     }
56
57     private void setSaldo(double sal) {
58         saldo = sal;
59     }
60
61     private void setTipoInteres(double tipo) {
62         tipoInteres = tipo;
63     }
64
65     /**
66      * Métodos adicionales
67      */
68     public void ingresar(double cantidad) {
69         saldo += cantidad;
70     }
71
72     public void retirar(double cantidad) {
73         saldo -= cantidad;
74     }
75
76     public String getNombre() {
77         return nombre;
78     }
79
80     public String getCuenta() {
81         return cuenta;
82     }
83
84     public double getSaldo() {
85         return saldo;
86     }
87
88     public double getTipoInteres() {
89         return tipoInteres;
90     }
91
92     public void setTiposInteres(double tipo) {
93         tipoInteres = tipo;
94     }
95 }
```

This version of the code includes detailed JavaDoc comments for each attribute and method, providing a comprehensive documentation of the class's structure and behavior.

Genero documentación JavaDoc en cada uno de los métodos añadiendo información de etiquetas en parámetros o returns en cada uno de los métodos.



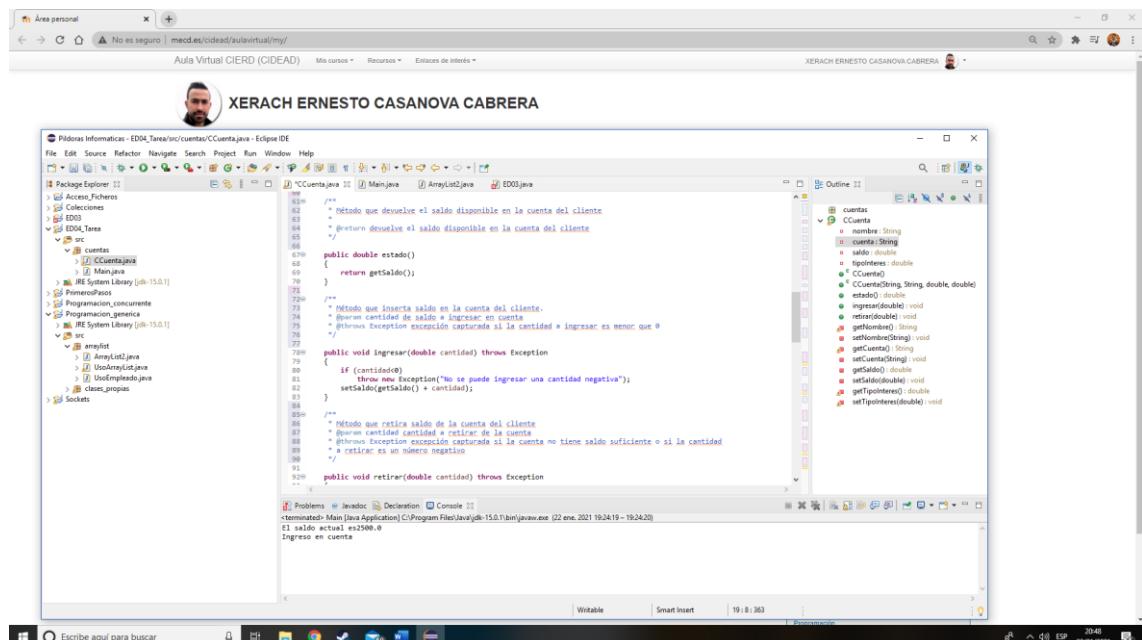
```
/*
 * Constructor de la clase CCuenta, construye objetos de tipo CCuenta.
 */
public CCuenta() {
    ...
}

/**
 * Constructor de la clase CCuenta, construye objetos de tipo CCuenta.
 *
 * @param nombre Nombre del cliente de la cuenta corriente
 * @param tipo C\u00f3digo de cuenta del cliente
 * @param tipoP Inter\u00f3n de la cuenta del cliente
 */
public CCuenta(String nom, String tipo, double sal, double tipoP) {
    ...
}

/**
 * M\u00e9todo que devuelve el saldo disponible en la cuenta del cliente
 */
public double estado() {
    ...
}

/**
 * M\u00e9todo que inserta saldo en la cuenta del cliente.
 *
 * @param cantidad cantidad de saldo a ingresar en cuenta
 * @throws Exception excepci\u00f3n capturada si la cantidad < 0
 */
public void ingresar(double cantidad) throws Exception {
    ...
}

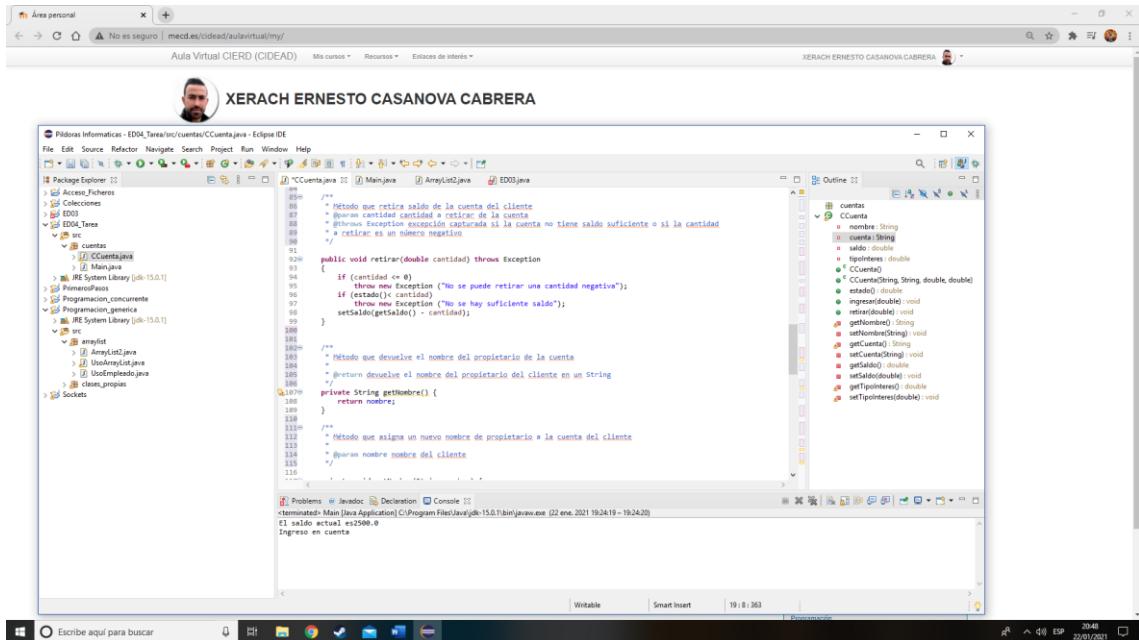
/**
 * M\u00e9todo que retira saldo de la cuenta del cliente
 *
 * @param cantidad cantidad a retirar de la cuenta
 * @throws Exception excepci\u00f3n capturada si la cuenta no tiene saldo suficiente o si la cantidad < 0
 */
public void retirar(double cantidad) throws Exception {
    ...
}
```



```
/*
 * M\u00e9todo que devuelve el saldo disponible en la cuenta del cliente
 */
public double estado() {
    ...
    return getSaldo();
}

/**
 * M\u00e9todo que inserta saldo en la cuenta del cliente.
 *
 * @param cantidad cantidad de saldo a ingresar en cuenta
 * @throws Exception excepci\u00f3n capturada si la cantidad < 0
 */
public void ingresar(double cantidad) throws Exception {
    ...
    if (cantidad < 0)
        throw new Exception("No se puede ingresar una cantidad negativa");
    setSaldo(getSaldo() + cantidad);
}

/**
 * M\u00e9todo que retira saldo de la cuenta del cliente
 *
 * @param cantidad cantidad a retirar de la cuenta
 * @throws Exception excepci\u00f3n capturada si la cuenta no tiene saldo suficiente o si la cantidad < 0
 */
public void retirar(double cantidad) throws Exception {
    ...
}
```



```
/*
 * Método que retira saldo de la cuenta del cliente
 * guaran cantidad cantidad a retirar de la cuenta
 * throws Exception excepción capturada si la cuenta no tiene saldo suficiente o si la cantidad
 * a retirar es un número negativo
 */
private void retirar(double cantidad) throws Exception {
    if (cantidad <= 0)
        throw new Exception ("No se puede retirar una cantidad negativa");
    if (cantidad > saldo)
        throw new Exception ("No se hay suficiente saldo");
    setSaldo(getSaldo() - cantidad);
}

/**
 * método que devuelve el nombre del propietario de la cuenta
 */
private String getNombre() {
    return nombre;
}

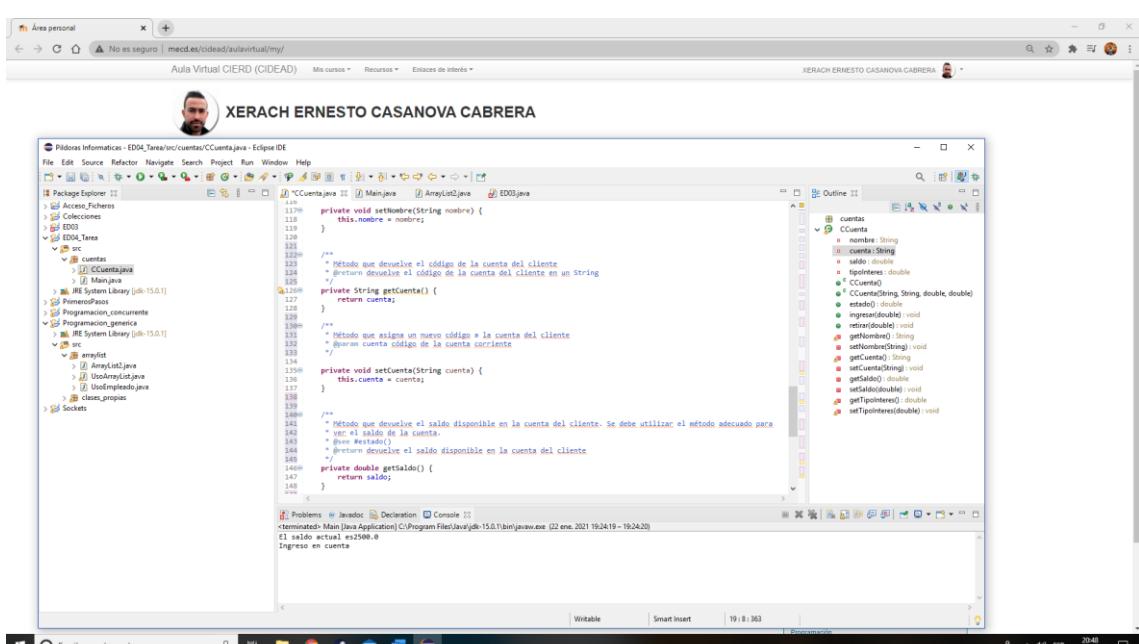
/**
 * método que asigna un nuevo nombre de propietario a la cuenta del cliente
 */
private void setNombre(String nombre) {
    this.nombre = nombre;
}

/**
 * método que devuelve el código de la cuenta del cliente
 */
private String getCuenta() {
    return cuenta;
}

/**
 * método que asigna un nuevo código a la cuenta del cliente
 */
private void setCuenta(String cuenta) {
    this.cuenta = cuenta;
}

private double getSaldo() {
    /*
     * método que devuelva el saldo disponible en la cuenta del cliente. Se debe utilizar el método adecuado para
     * ver el saldo de la cuenta.
     */
    if (estaCerrada())
        throw new IllegalStateException("La cuenta está cerrada");
    return saldo;
}

private void setSaldo(double saldo) {
    /*
     * método que establece el saldo disponible en la cuenta del cliente
     */
    if (estaCerrada())
        throw new IllegalStateException("La cuenta está cerrada");
    this.saldo = saldo;
}
```



```
/*
 * Método que retira saldo de la cuenta del cliente
 * guaran cantidad cantidad a retirar de la cuenta
 * throws Exception excepción capturada si la cuenta no tiene saldo suficiente o si la cantidad
 * a retirar es un número negativo
 */
private void retirar(double cantidad) throws Exception {
    if (cantidad <= 0)
        throw new Exception ("No se puede retirar una cantidad negativa");
    if (cantidad > saldo)
        throw new Exception ("No se hay suficiente saldo");
    setSaldo(getSaldo() - cantidad);
}

/**
 * método que devuelve el nombre del propietario de la cuenta
 */
private String getNombre() {
    return nombre;
}

/**
 * método que asigna un nuevo nombre de propietario a la cuenta del cliente
 */
private void setNombre(String nombre) {
    this.nombre = nombre;
}

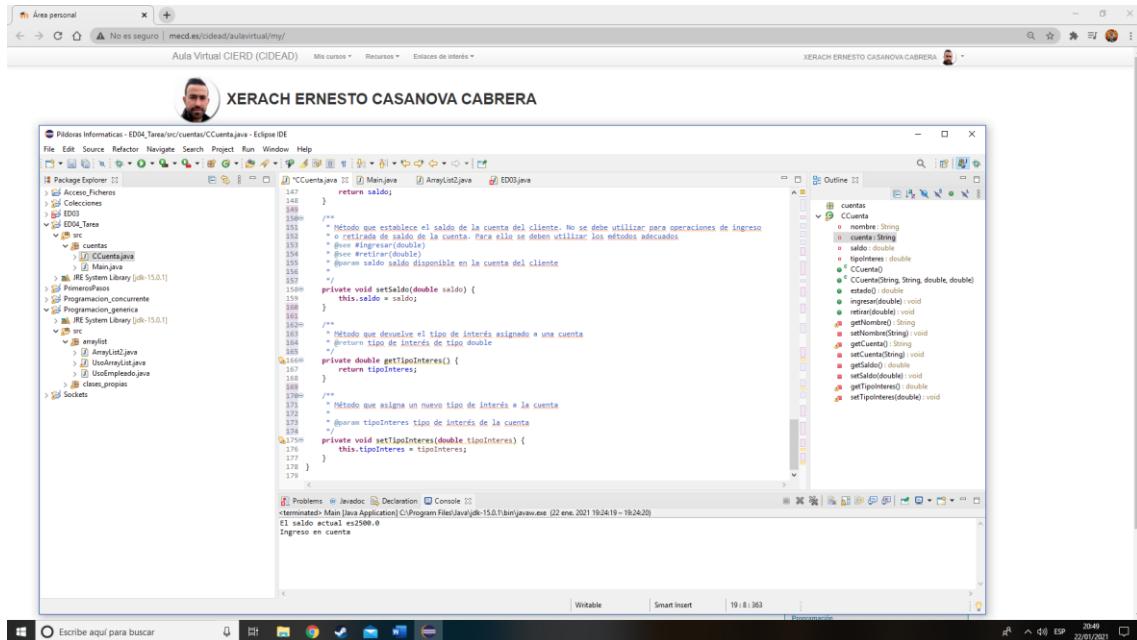
/**
 * método que devuelve el código de la cuenta del cliente
 */
private String getCuenta() {
    return cuenta;
}

/**
 * método que asigna un nuevo código a la cuenta del cliente
 */
private void setCuenta(String cuenta) {
    this.cuenta = cuenta;
}

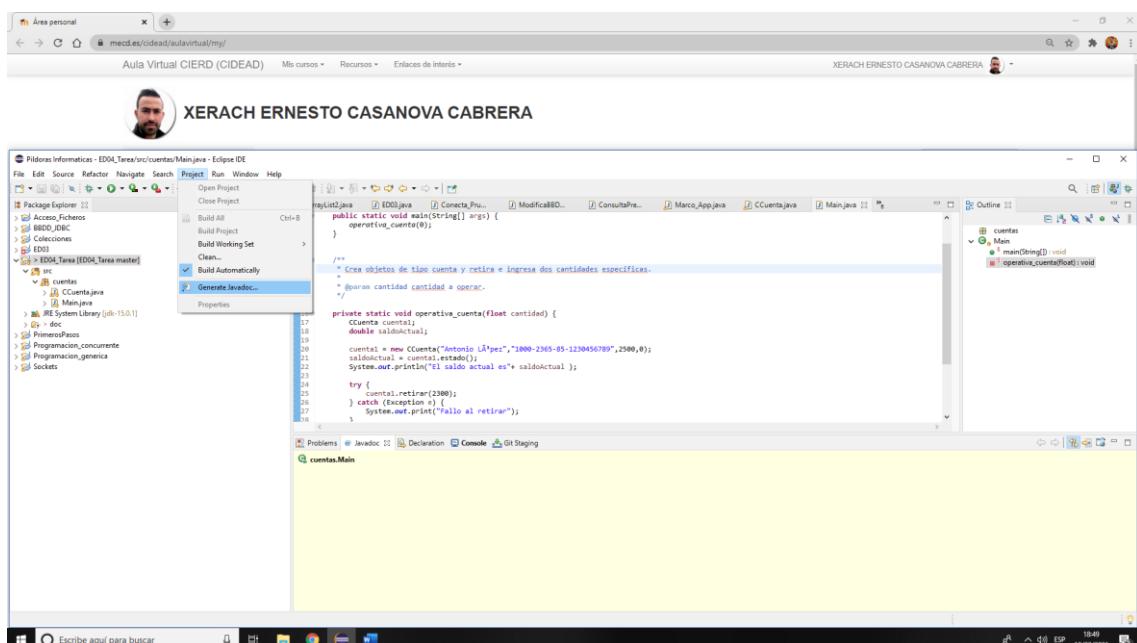
private double getSaldo() {
    /*
     * método que devuelva el saldo disponible en la cuenta del cliente. Se debe utilizar el método adecuado para
     * ver el saldo de la cuenta.
     */
    if (estaCerrada())
        throw new IllegalStateException("La cuenta está cerrada");
    return saldo;
}

private void setSaldo(double saldo) {
    /*
     * método que establece el saldo disponible en la cuenta del cliente
     */
    if (estaCerrada())
        throw new IllegalStateException("La cuenta está cerrada");
    this.saldo = saldo;
}
```

Los métodos `getSaldo` y `setSaldo` no deben usarse para hacer operaciones de inserción, retirada o visualización del saldo disponible en la cuenta, ya que existen para ello métodos implementados. Para indicar la recomendación del uso de los métodos adecuados hago referencia a ellos mediante la etiqueta `@see`.



Por último, genero el javadoc yendo al menú superior Project/generate javadoc



Y realizo un commit de todos los archivos generados por el javadoc, así como un push para subirlo a mi repositorio

