

ENTORNOS DE DESARROLLO

Tarea ED 04



11 DE FEBRERO DE 2021
MERCEDES CRUZ PORTILLO

Tarea ED 04

Enunciado.....	2
REFACTORIZACIÓN.....	2
GIT.....	10
JAVADOC.....	17
URL del repositorio GitHub.....	20

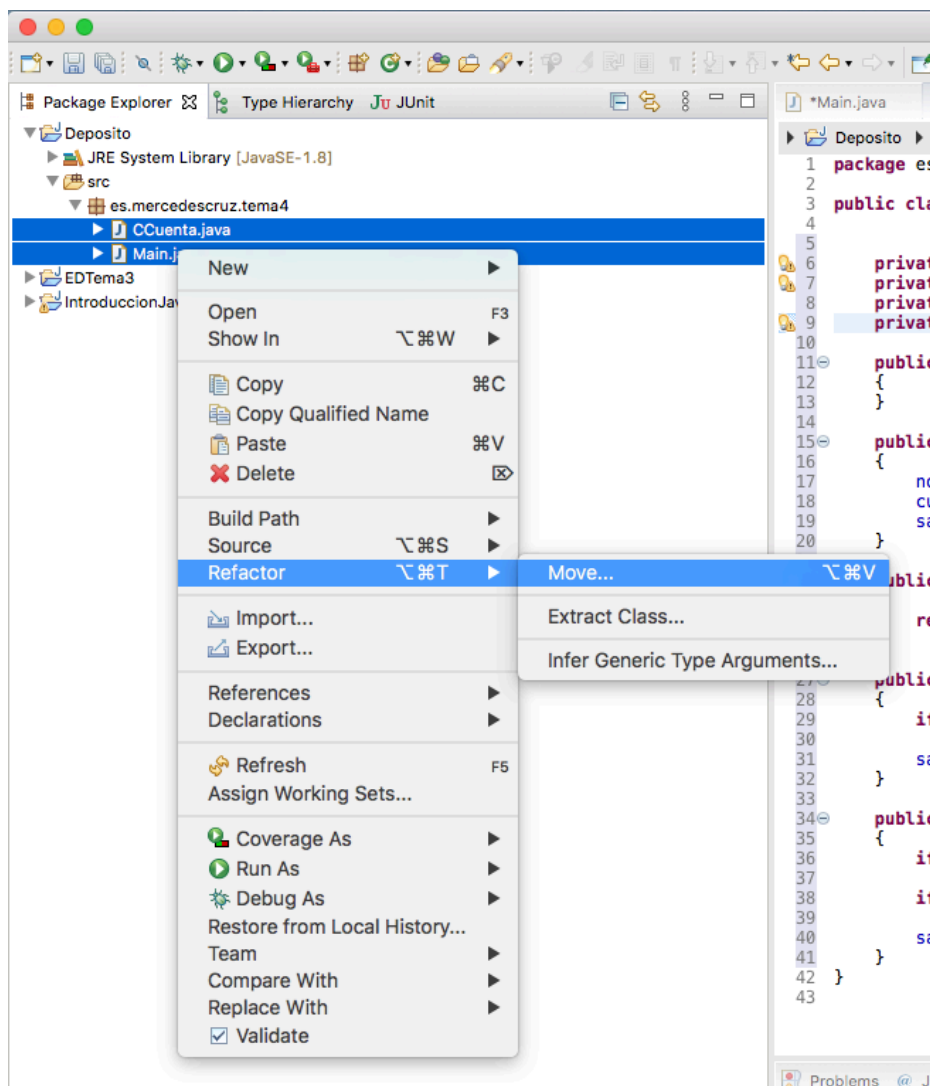
Enunciado.

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. Pula aquí para descargar dicho proyecto ("Deposito.rar"). Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

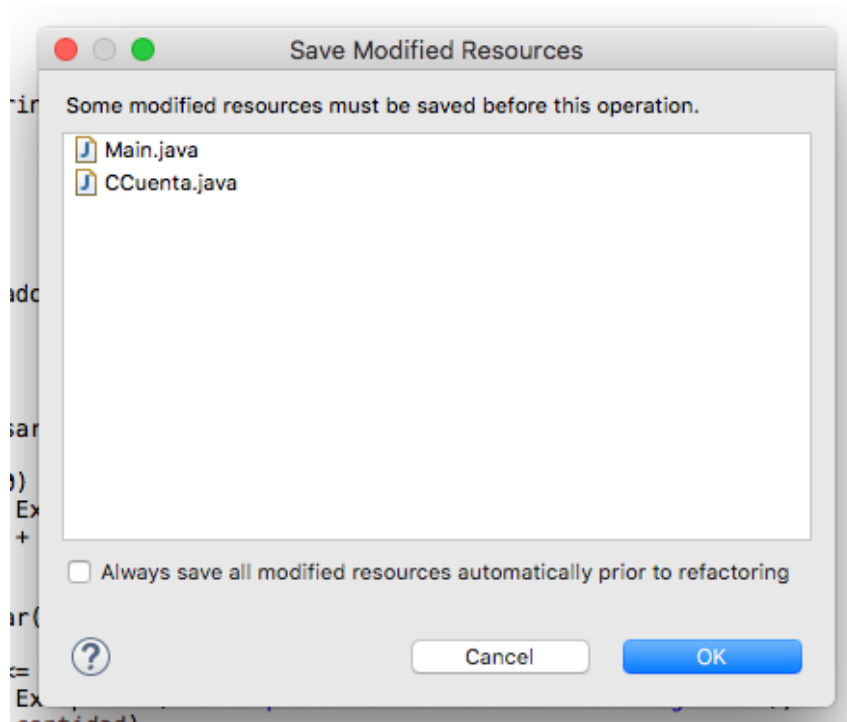
REFACTORIZACIÓN.

1. Las clases deberán formar parte del paquete cuentas.

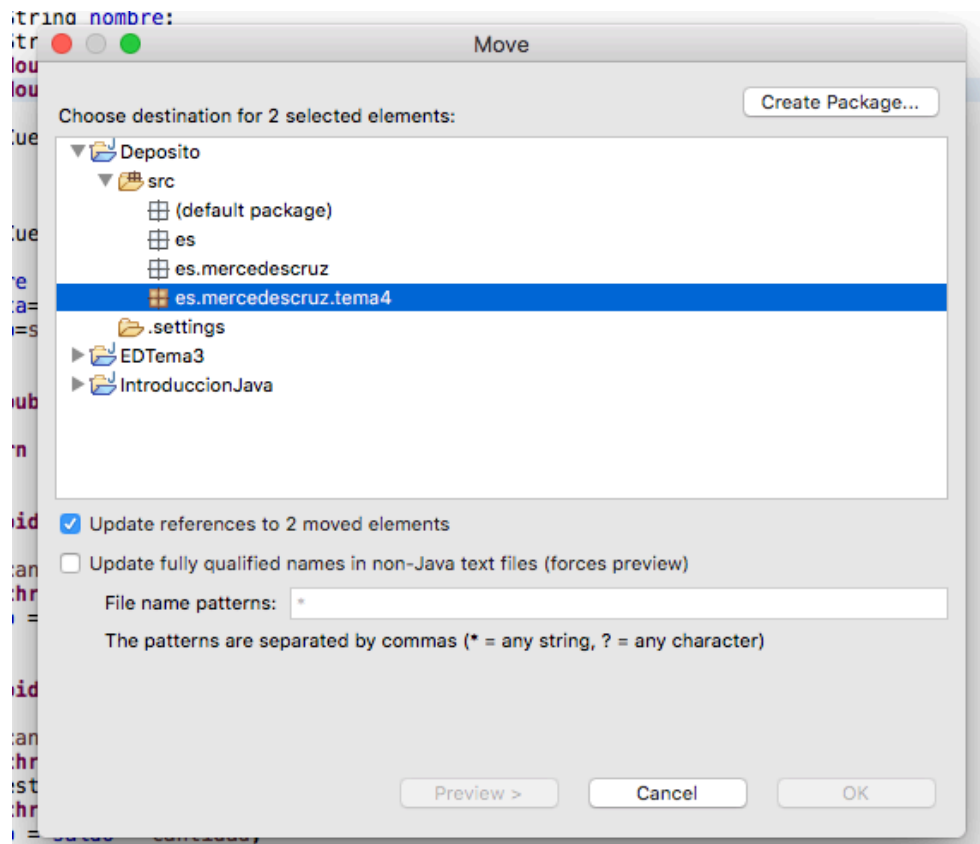
Para que las clases formen parte del paquete cuentas, he tenido que refactorizarlo. Para ello, selecciono las dos clases que quiero refactorizar, le doy a botón derecho > Refactor y selecciono Move...



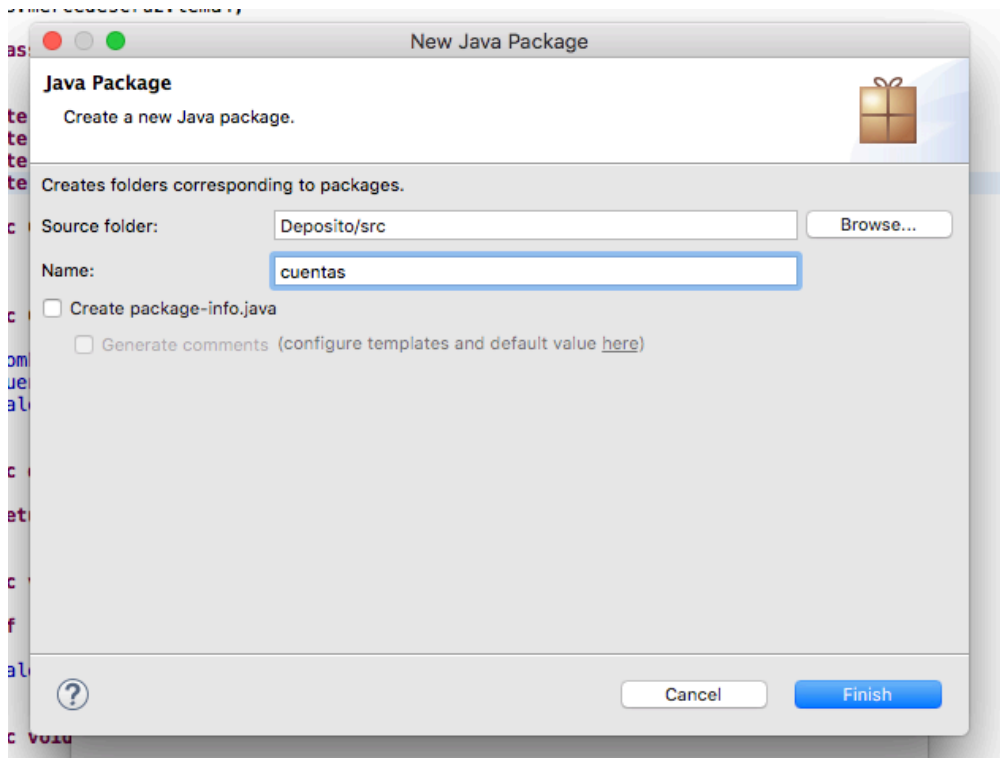
Como puede verse, he seleccionado las dos clases que queremos mover al paquete cuentas.



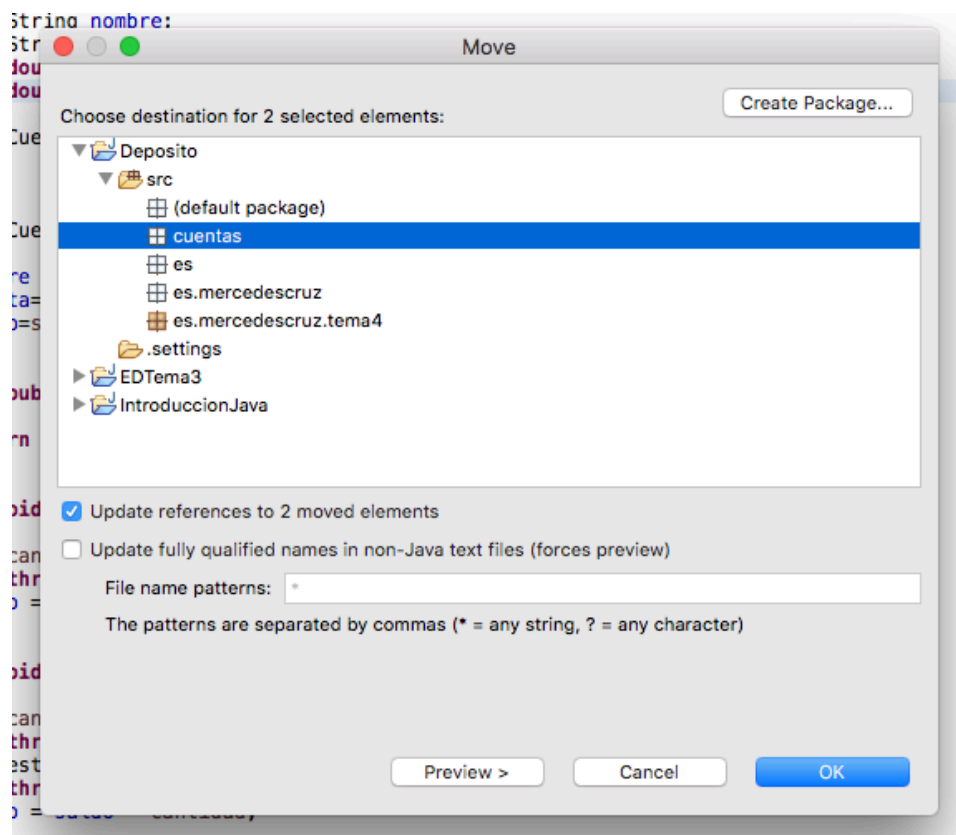
Le doy a "Create Package" para crear el paquete cuentas.



Creo el paquete cuentas.

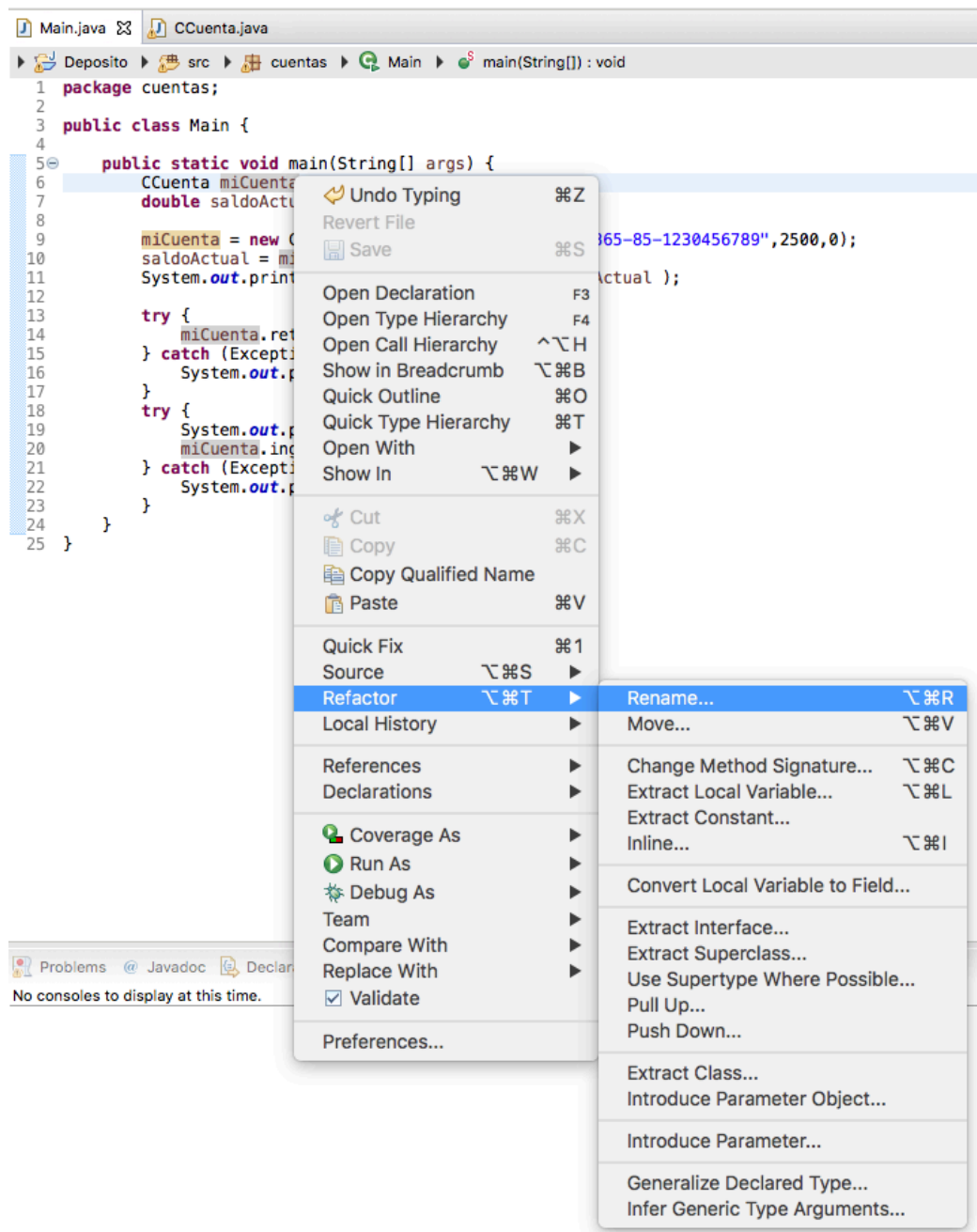


Por último, selecciono el paquete cuentas para que se muevan las clases a dicho paquete.

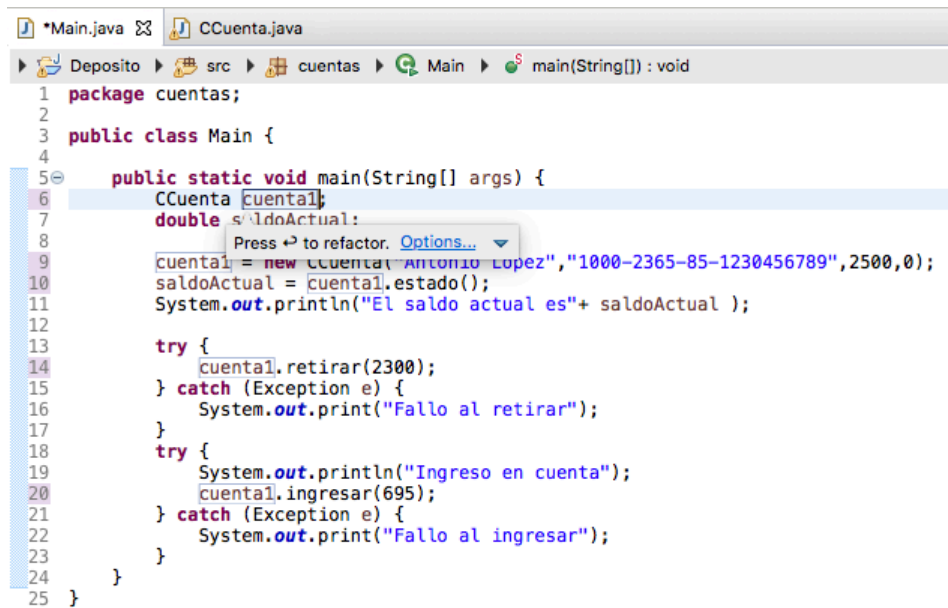


2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".

Para cambiar el nombre de la variable "miCuenta", me sitúo en el nombre de la variable, clico en el botón derecho > Refactor > Rename... y renombro la variable por "cuenta1".



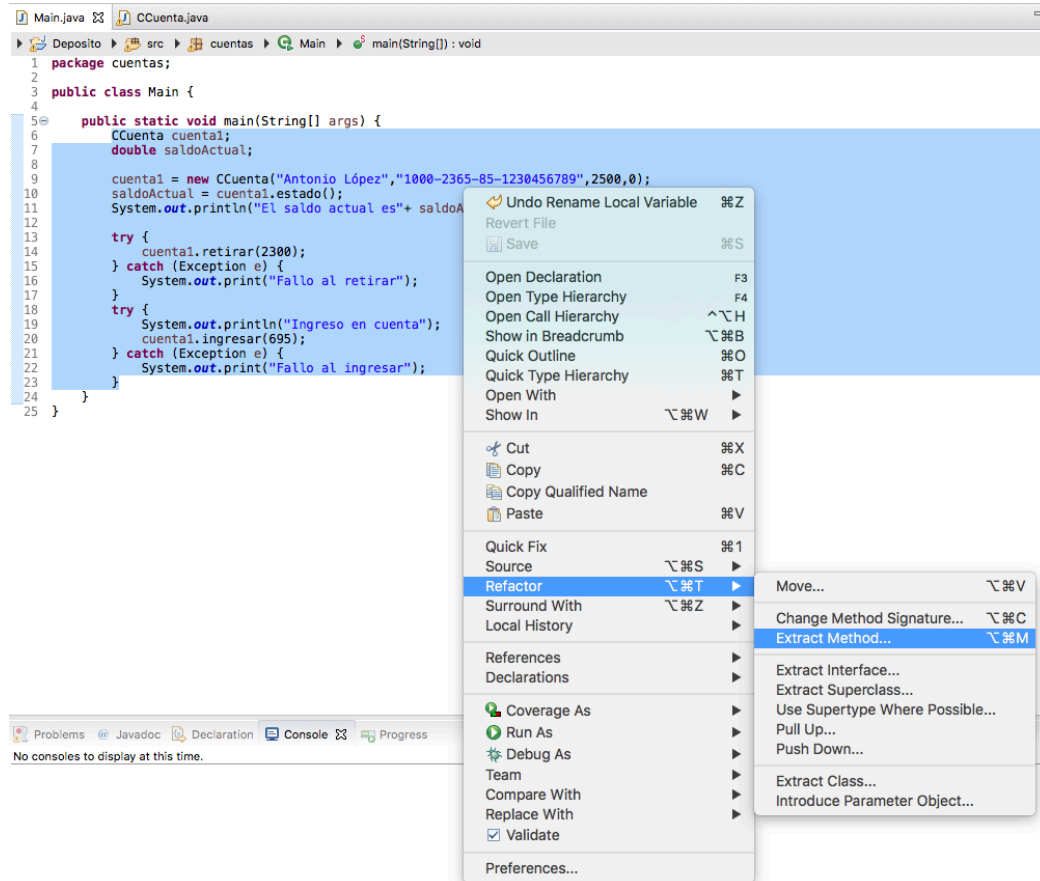
Aquí, puede verse ya cambiada. Sólo falta darle a intro.



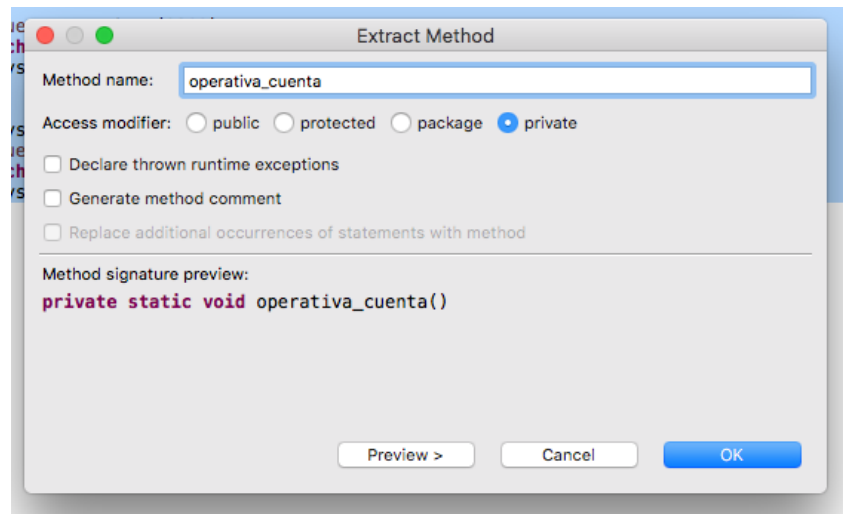
```
1 package cuentas;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         CCuenta cuenta1;
7         double saldoActual;
8         cuenta1 = new CCuenta("Antonio López", "1000-2365-85-1230456789", 2500, 0);
9         saldoActual = cuenta1.estado();
10        System.out.println("El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

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

Para introducir el método operativa_cuenta, selecciono las sentencias de la clase Main que operan con el objeto cuenta1, clico en el botón derecho > Refactor > Extract Method... y renombro la variable por "cuenta1".

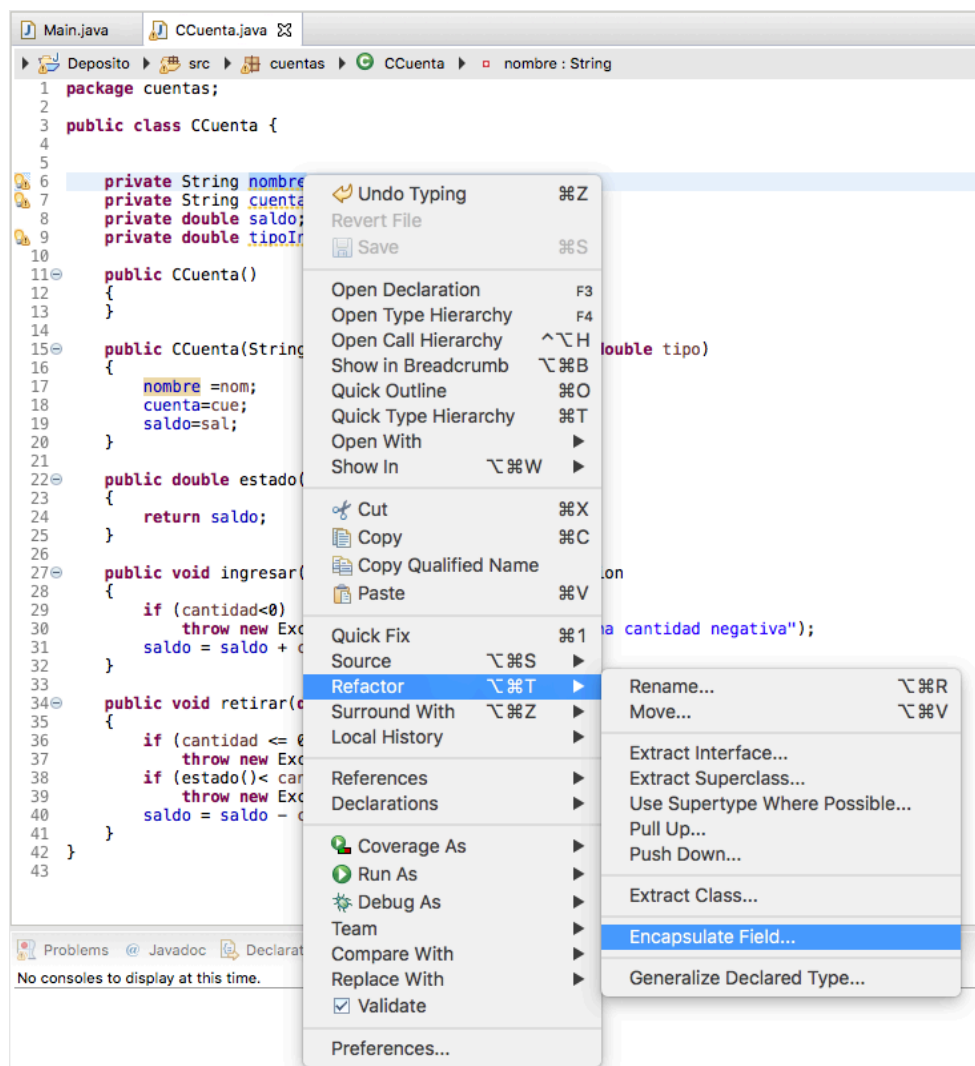


Le doy el nombre “operativa_cuenta” al nuevométodo y por último, le doy a ok.

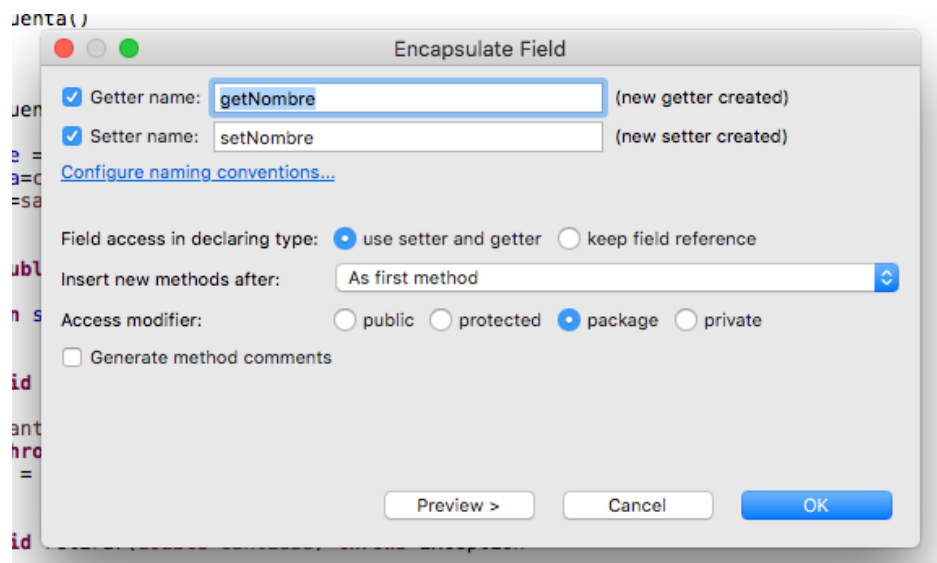


4. Encapsular los atributos de la clase CCuenta.

Para encapsular los atributos de la clase CCuenta debo seleccionar el nombre del atributo, clico en el botón derecho > Refactor > Encapsulate Field...



Después, lo selecciono “como primer método”, el modificador de acceso será de paquete y le doy a ok.



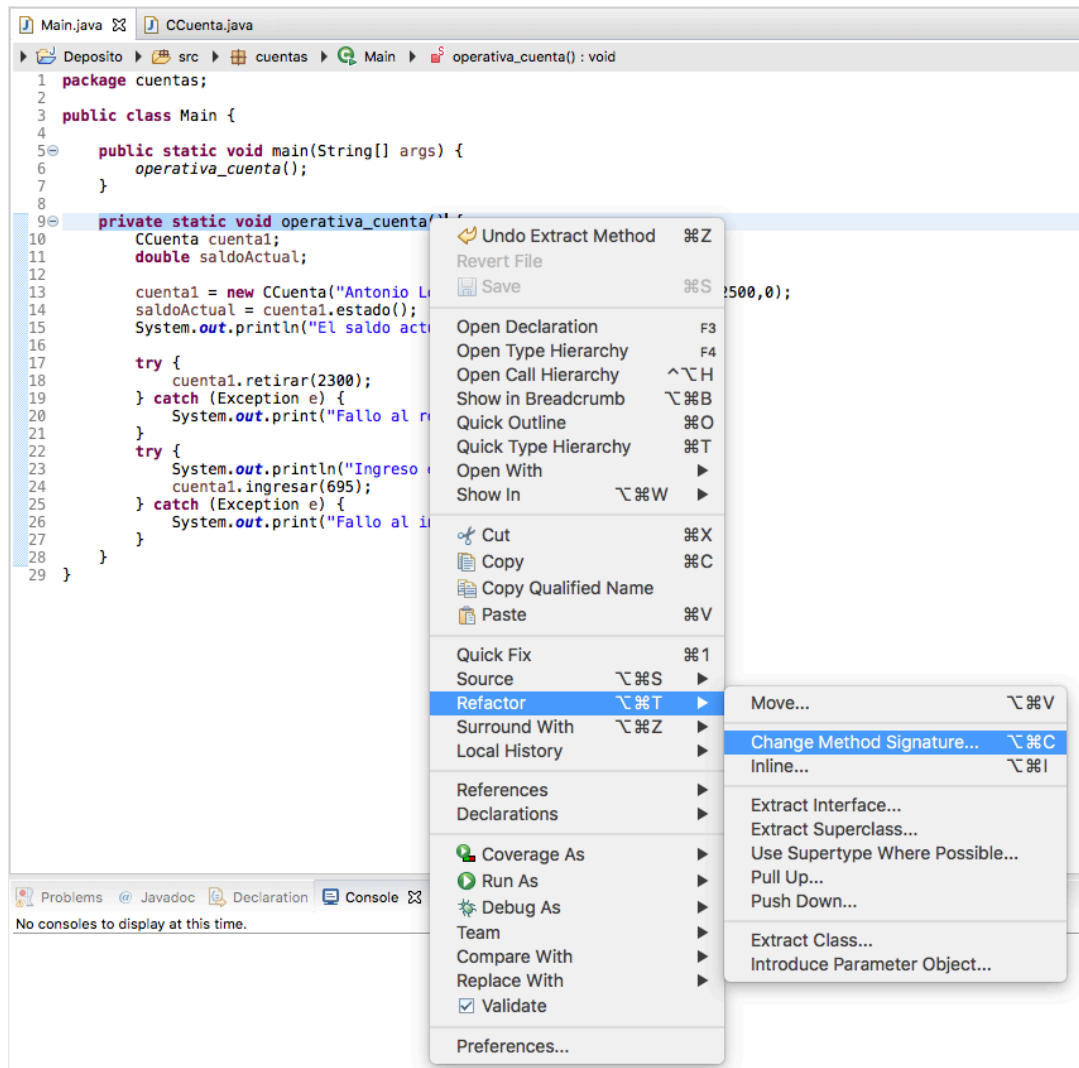
Por último, creo los demás Getters y Setter, uno detrás de otro. Es decir, primero los Getters y después los Setters. Además, el Get y Set de cada atributo también se han colocado en orden de aparición. Es decir, primero el Get y Set del atributo nombre, después el del atributo cuenta, en tercer lugar, el del atributo saldo y en cuarto lugar, el del atributo tipoInterés.

```

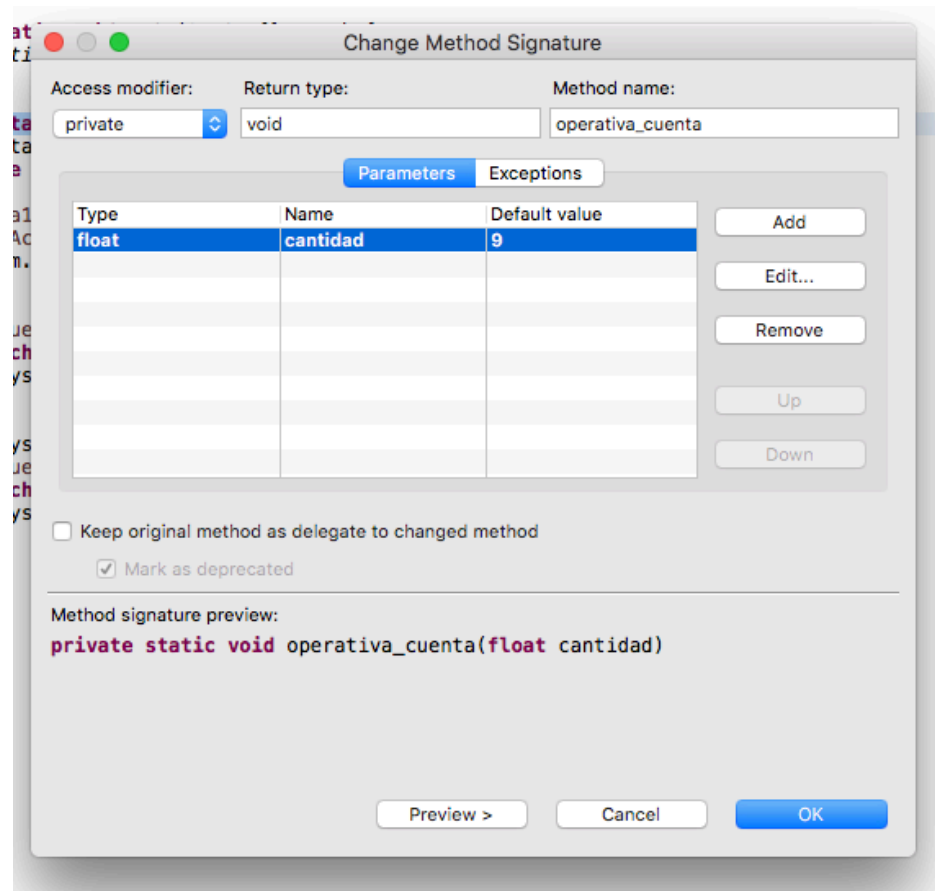
Main.java  CCuenta.java
Deposito  src  cuentas  CCuenta  CCuenta()
1  package cuentas;
2
3  public class CCuenta {
4
5
6      private String nombre;
7      private String cuenta;
8      private double saldo;
9      private double tipoInterés;
10
11     String getNombre() {
12         return nombre;
13     }
14
15     void setNombre(String nombre) {
16         this.nombre = nombre;
17     }
18
19     String getCuenta() {
20         return cuenta;
21     }
22
23     void setCuenta(String cuenta) {
24         this.cuenta = cuenta;
25     }
26
27     double getSaldo() {
28         return saldo;
29     }
30
31     void setSaldo(double saldo) {
32         this.saldo = saldo;
33     }
34
35     double getTipoInterés() {
36         return tipoInterés;
37     }
38
39     void setTipoInterés(double tipoInterés) {
40         this.tipoInterés = tipoInterés;
41     }
42
43 }
```

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

Para añadir un nuevo parámetro al método operativa_cuenta, de nombre cantidad y de tipo float, clico el botón derecho > Refactor > Change Method Signature...



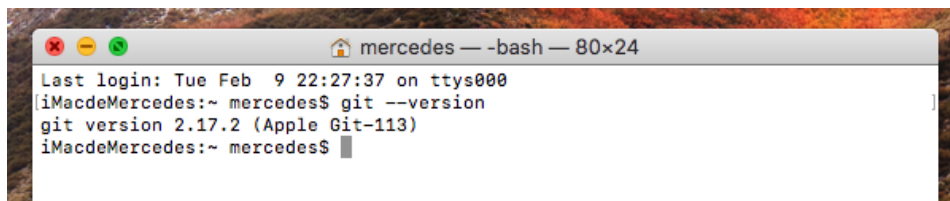
Para añadir el tipo y el nombre en el parámetro de entrada del método, selecciono "Add", clico en "Tipo" y añado el tipo "float", clico en "Nombre", añado el nombre "cantidad" y le asigno un valor "9". Por último, le doy a ok.



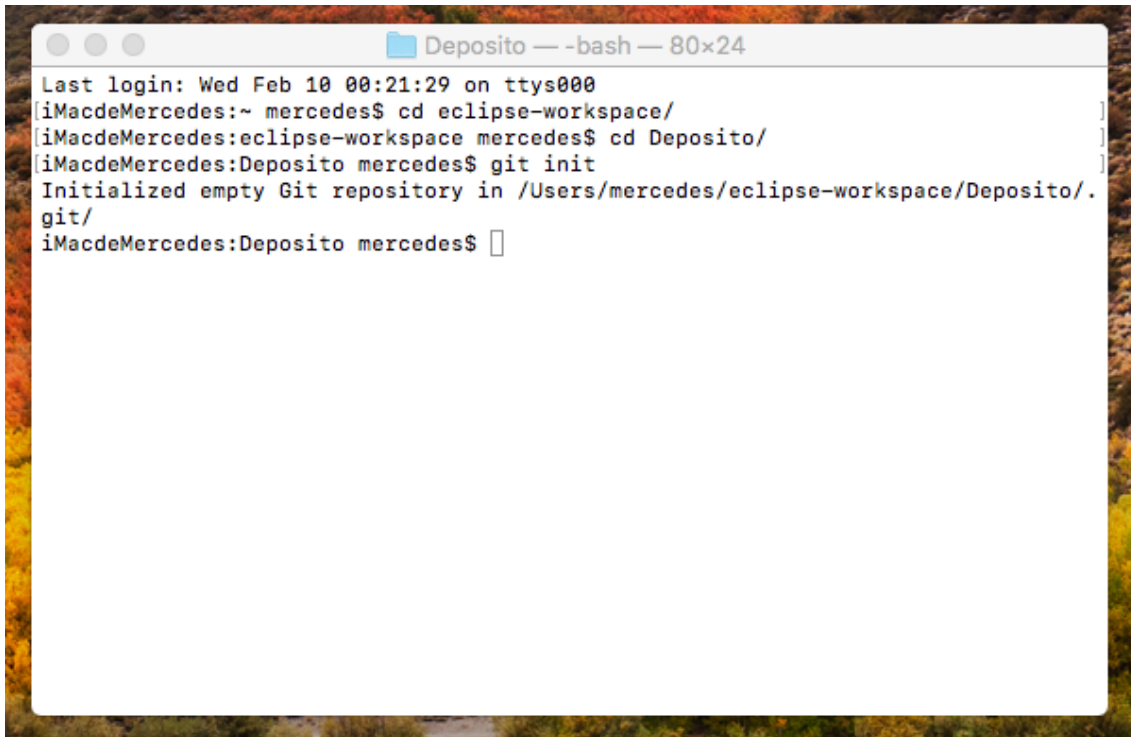
GIT.

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

Abro la Terminal.app en Mac y ejecuto el comando `git --version` para ver si ya tengo instalado GIT. Como se ve en la imagen, sí lo tengo instalado. La versión es 2.17.2 (Apple Git-113).

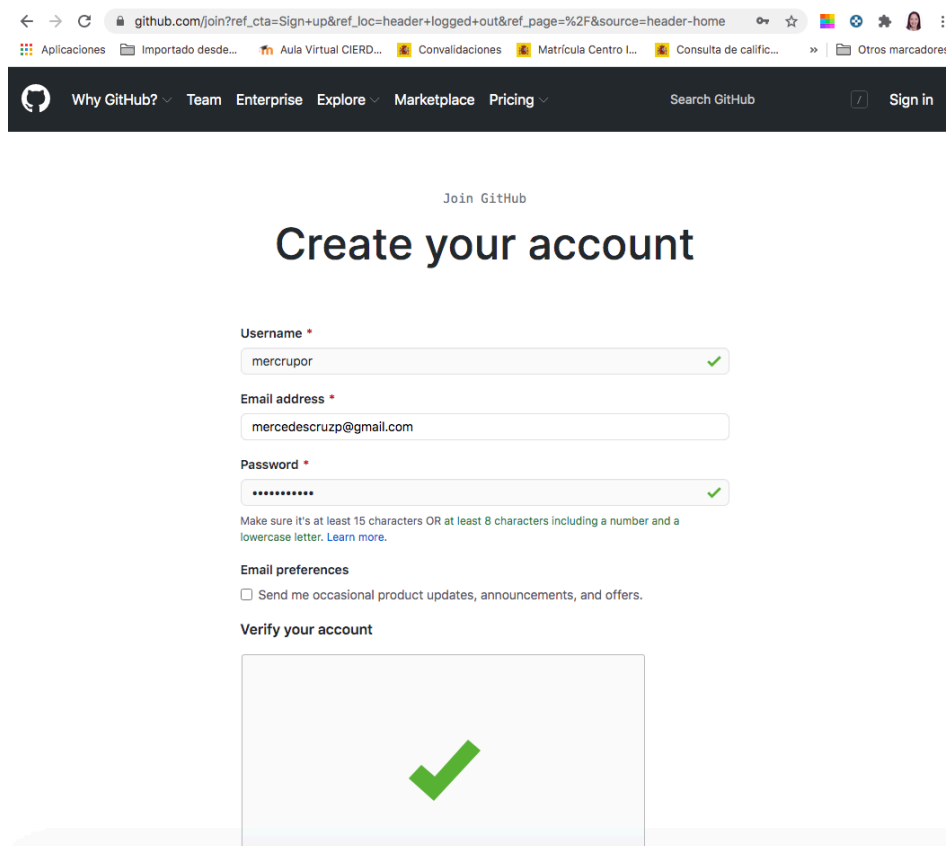


Ahora ejecuto el comando git init para crear un nuevo repositorio en mi local. El directorio, por defecto, está oculto.

A terminal window titled 'Deposito — -bash — 80x24' with a macOS-style title bar. The terminal shows the following commands and output:

```
Last login: Wed Feb 10 00:21:29 on ttys000
iMacdeMercedes:~ mercedes$ cd eclipse-workspace/
iMacdeMercedes:eclipse-workspace mercedes$ cd Deposito/
iMacdeMercedes:Deposito mercedes$ git init
Initialized empty Git repository in /Users/mercedes/eclipse-workspace/Deposito/.git/
iMacdeMercedes:Deposito mercedes$
```

Creo una cuenta en GitHub.

A screenshot of the GitHub 'Create your account' page. The page has a dark header with the GitHub logo and navigation links. The main content area is white and contains the following fields and sections:

- Join GitHub** (link)
- Create your account** (h1)
- Username *** (input field with 'mercrupor' and a green checkmark)
- Email address *** (input field with 'mercedescruzp@gmail.com')
- Password *** (input field with masked characters and a green checkmark)
- Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)
- Email preferences**
 - ☐ Send me occasional product updates, announcements, and offers.
- Verify your account**
 - A large green checkmark icon on a light gray background.

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?










Software Engineer I write code	Student I go to school
Product Manager I write specs	UX & Design I draw interfaces
Data & Analytics I write queries	Marketing & Sales I look at charts
Teacher I educate people	Other I do my own thing

How much programming experience do you have?

None I don't program at all	A little I'm new to programming
A moderate amount I'm somewhat experienced	A lot I'm very experienced

What do you plan to use GitHub for?

(Select up to 3)

 Learn to code	 Learn Git and GitHub	 Host a project (repository)
 Create a website with GitHub Pages	 Collaborating with my team	 Find and contribute to open source
 School work and student projects	 Use the GitHub API	 Other

I am interested in:

java x

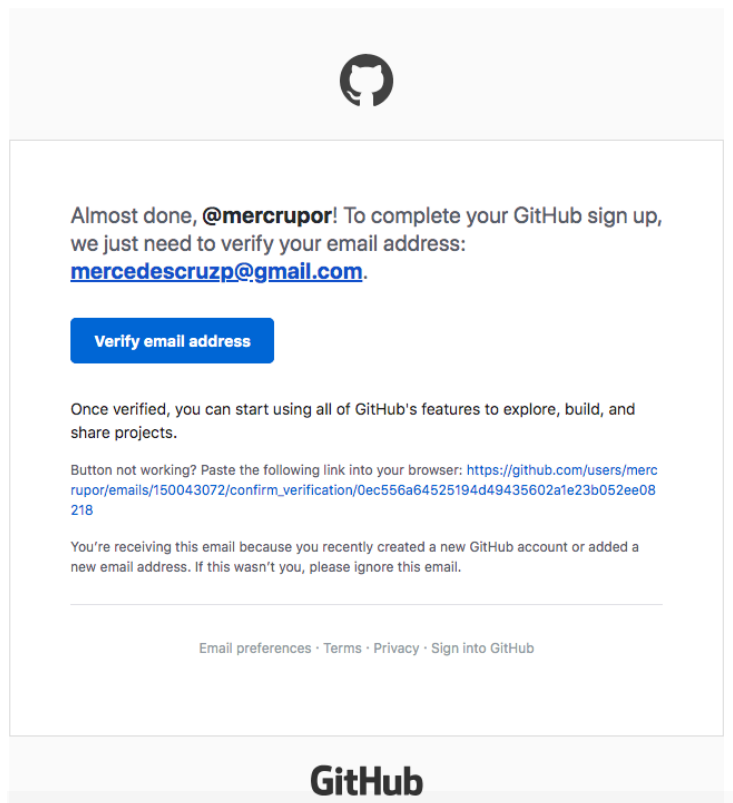
[languages, frameworks, industries]

We'll connect you with communities and projects that fit your interests.

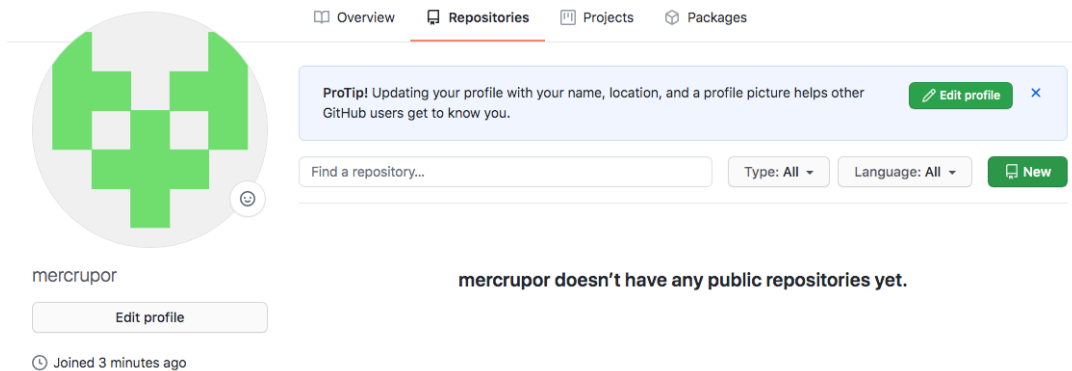
For example: `clojure` `gradescope` `jquery`

Complete setup

Después, en mi email tengo que verificar la cuenta que he creado en GitHub.



Este es mi perfil en GitHub donde aún no hay repositorios creados.



Ahora, procedo a crear un nuevo repositorio. Éste se va a llamar ED_Tema4, le doy una descripción y lo dejo como público.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

mercrupor

Repository name *

ED_Tema4

Great repository names are short and memorable. Need inspiration? How about [fictional-fiesta](#)?

Description (optional)

Repositorio público para la asignatura Entornos de Desarrollo

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Ahora ejecuto el comando git status para saber el estado de los ficheros. Dicho estado dice que estoy en la rama master y que aún no hay commits. Git ha detectado que hay ficheros sin registrar en mi máquina. Es decir, que no están guardados en el repositorio. Por lo tanto, Git ignora los ficheros .DS_Store, .classpath, .project, .settings/, bin/ y src/, pero te da la posibilidad de trackearlos (track).

```
iMacdeMercedes:Deposito mercedes$ git init
Initialized empty Git repository in /Users/mercedes/eclipse-workspace/Deposito/.git/
iMacdeMercedes:Deposito mercedes$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .DS_Store
        .classpath
        .project
        .settings/
        bin/
        src/

nothing added to commit but untracked files present (use "git add" to track)
iMacdeMercedes:Deposito mercedes$ ls -la
total 32
drwxr-xr-x  9 mercedes  staff   288 10 feb 00:39 .
drwxr-xr-x  7 mercedes  staff   224  9 feb 22:40 ..
-rw-r--r--@ 1 mercedes  staff  6148 10 feb 00:38 .DS_Store
-rw-r--r--  1 mercedes  staff   295  9 feb 22:40 .classpath
drwxr-xr-x  9 mercedes  staff   288 10 feb 00:46 .git
-rw-r--r--  1 mercedes  staff   367  9 feb 22:40 .project
drwxr-xr-x  3 mercedes  staff    96  9 feb 22:40 .settings
drwxr-xr-x  4 mercedes  staff   128  9 feb 22:58 bin
drwxr-xr-x  4 mercedes  staff   128  9 feb 22:57 src
iMacdeMercedes:Deposito mercedes$
```

Gracias a git add los trackea (track), es decir, hace que Git observe los ficheros, los selecciona para incluirlos en el repositorio. He seleccionado los ficheros .classpath, .project, .settings/ y src/.

```
iMacdeMercedes:Deposito mercedes$ git add .project
iMacdeMercedes:Deposito mercedes$ git add src/
iMacdeMercedes:Deposito mercedes$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   .classpath
    new file:   .project
    new file:   src/cuentas/CCuenta.java
    new file:   src/cuentas/Main.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .DS_Store
    .settings/
    bin/

iMacdeMercedes:Deposito mercedes$ git add .settings/
iMacdeMercedes:Deposito mercedes$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   .classpath
    new file:   .project
    new file:   .settings/org.eclipse.jdt.core.prefs
    new file:   src/cuentas/CCuenta.java
    new file:   src/cuentas/Main.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .DS_Store
    bin/

iMacdeMercedes:Deposito mercedes$
```

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

Ahora, creo mi primer commit. La opción -m significa mensaje que en este caso, ha sido “mi primer commit”.

```
iMacdeMercedes:Deposito mercedes$ git commit -m "mi primer commit"
[master (root-commit) fc5378b] mi primer commit
Committer: Mercedes Cruz Portillo <mercedes@iMacdeMercedes.lan>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

5 files changed, 137 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 src/cuentas/CCuenta.java
create mode 100644 src/cuentas/Main.java
iMacdeMercedes:Deposito mercedes$
```


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

Para mostrar el historial de versiones del proyecto mediante un comando desde consola he usado el comando git log. Además, he usado el comando git diff para mostrar los cambios de los ficheros que se han modificado. En este caso, el fichero modificado ha sido "src/cuentas/Main.java".

```
iMacdeMercedes:Deposito mercedes$ git log
commit fc5378bacb32d6ceab1028dbae384dad03770943 (HEAD -> master)
Author: Mercedes Cruz Portillo <mercedes@iMacdeMercedes.lan>
Date:   Wed Feb 10 01:05:53 2021 +0100

    mi primer commit

iMacdeMercedes:Deposito mercedes$ git diff
diff --git a/src/cuentas/Main.java b/src/cuentas/Main.java
index da3cbb2..544a73d 100644
--- a/src/cuentas/Main.java
+++ b/src/cuentas/Main.java
@@ -23,7 +23,7 @@ public class Main {
     System.out.println("Ingreso en cuenta");
     cuenta1.ingresar(695);
     } catch (Exception e) {
-        System.out.print("Fallo al ingresar");
+        System.out.print("Fallo al ingresar");jhggfgh
     }
 }
}
\ No newline at end of file
iMacdeMercedes:Deposito mercedes$
```

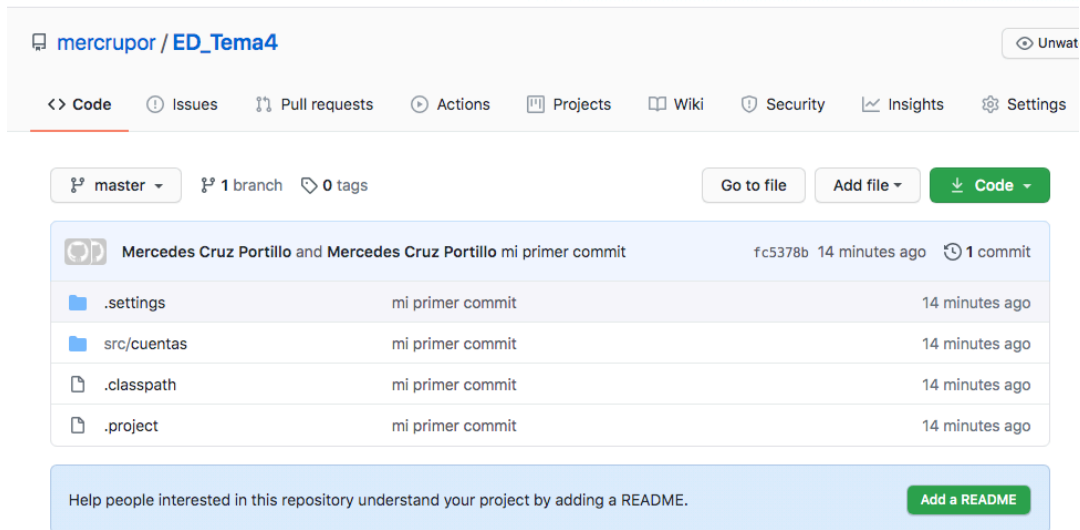
Con el comando git remote add lo que estoy haciendo es vincular mi repositorio local con el repositorio remoto ubicado en GitHub, al que he llamado origin. El comando completo es git remote add origin https://github.com/mercrupor/ED_Tema4.git

```
iMacdeMercedes:Deposito mercedes$ git remote add origin https://github.com/mercrupor/ED_Tema4.git
iMacdeMercedes:Deposito mercedes$
```

Una vez vinculado, ya puedo hacer el push, es decir, subir los commits. He usado el comando "git push -u origin master". La opción -u vincula la rama directamente a master de origin para que los futuros pull sean más sencillos.

```
iMacdeMercedes:Deposito mercedes$ git push -u origin master
Username for 'https://github.com': mercrupor
[Password for 'https://mercrupor@github.com':
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/mercrupor/ED_Tema4.git/'
iMacdeMercedes:Deposito mercedes$ git push -u origin master
Username for 'https://github.com': mercrupor
[Password for 'https://mercrupor@github.com':
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 1.81 KiB | 925.00 KiB/s, done.
Total 10 (delta 0), reused 0 (delta 0)
To https://github.com/mercrupor/ED_Tema4.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
iMacdeMercedes:Deposito mercedes$
```

Por último, observamos que en GitHub tenemos el commit “mi primer commit” con los ficheros que seleccioné anteriormente.



JAVADOC.

1. Insertar comentarios Javadoc en la clase CCuenta.

Como se ve en las siguientes imágenes, he insertado los comentarios Javadoc en la clase CCuenta.

```
1 package cuentas;
2
3 /**
4  * Clase que representa la cuenta de un usuario.
5  *
6  * @author mercedes cruz portillo
7  */
8 public class CCuenta {
9
10     /**
11      * Almacena el nombre del usuario.
12      */
13     private String nombre;
14
15     /**
16      * Almacena la cuenta del usuario.
17      */
18     private String cuenta;
19
20     /**
21      * Almacena el saldo de la cuenta del usuario.
22      */
23     private double saldo;
24
25     /**
26      * Almacena el tipo de interés.
27      */
28     private double tipoInteres;
29
30     /**
31      * Método que devuelve el nombre del usuario.
32      *
33      * @return
34      */
35     String getNombre() {
36         return nombre;
37     }
38
39     /**
40      * Método que actualiza el nombre del usuario.
41      *
42      * @param nombre a actualizar
43      */
44     void setNombre(String nombre) {
45         this.nombre = nombre;
46     }
47 }
```

```

Main.java  CCuenta.java
Deposito  src  cuentas  CCuenta
1 package cuentas;
2
3 /**
4  * Clase que representa la cuenta de un usuario.
5  *
6  * @author mercedes cruz portillo
7  */
8 public class CCuenta {
9
10     /**
11      * Almacena el nombre del usuario.
12      */
13     private String nombre;
14
15     /**
16      * Almacena la cuenta del usuario.
17      */
18     private String cuenta;
19
20     /**
21      * Almacena el saldo de la cuenta del usuario.
22      */
23     private double saldo;
24
25     /**
26      * Almacena el tipo de interés.
27      */
28     private double tipoInteres;
29
30     /**
31      * Método que devuelve el nombre del usuario.
32      *
33      * @return
34      */
35     String getNombre() {
36         return nombre;
37     }
38
39     /**
40      * Método que actualiza el nombre del usuario.
41      *
42      * @param nombre a actualizar
43      */
44     void setNombre(String nombre) {
45         this.nombre = nombre;

```

```

Main.java  CCuenta.java
Deposito  src  cuentas  CCuenta
47
48     /**
49      * Método que devuelve la cuenta del usuario.
50      *
51      * @return la cuenta del usuario
52      */
53     String getCuenta() {
54         return cuenta;
55     }
56
57     /**
58      * Método que actualiza la cuenta del usuario.
59      *
60      * @param cuenta a actualizar
61      */
62     void setCuenta(String cuenta) {
63         this.cuenta = cuenta;
64     }
65
66     /**
67      * Método que devuelve el saldo de la cuenta del usuario.
68      *
69      * @return el saldo de la cuenta del usuario
70      */
71     double getSaldo() {
72         return saldo;
73     }
74
75     /**
76      * Método que actualiza el saldo de la cuenta del usuario.
77      *
78      * @param saldo a actualizar
79      */
80     void setSaldo(double saldo) {
81         this.saldo = saldo;
82     }
83
84     /**
85      * Método que devuelve el tipo de interés.
86      *
87      * @return el tipo de interés
88      */
89     double getTipoInterés() {
90         return tipoInteres;
91     }

```

```

Main.java CCuenta.java
Deposito src cuentas CCuenta
93 /**
94  * Método que actualiza el tipo de interés.
95  *
96  * @param tipoInteres a actualizar
97  */
98 void setTipoInterés(double tipoInteres) {
99     this.tipoInteres = tipoInteres;
100 }
101
102 /**
103  * Constructor de la clase.
104  */
105 public CCuenta() {
106 }
107
108 /**
109  * Constructor sobrecargado para inicializarse con un nombre, cuenta y saldo.
110  *
111  * @param nom el nombre
112  * @param cue la cuenta
113  * @param sal el saldo
114  * @param tipo el tipo de interés (que se ignora)
115  */
116 public CCuenta(String nom, String cue, double sal, double tipo) {
117     setNombre(nom);
118     setCuenta(cue);
119     setSaldo(sal);
120 }
121
122 /**
123  * Método que devuelve el saldo de la cuenta del usuario.
124  *
125  * @return el saldo de la cuenta del usuario.
126  */
127 public double estado() {
128     return getSaldo();
129 }
130
131 /**
132  * Método que se ejecuta cuando el usuario intenta ingresar saldo en la cuenta.
133  *
134  * @param cantidad a ingresar
135  * @throws Exception lanza una excepción cuando el usuario mete una cantidad
136  *         negativa
137  */

```

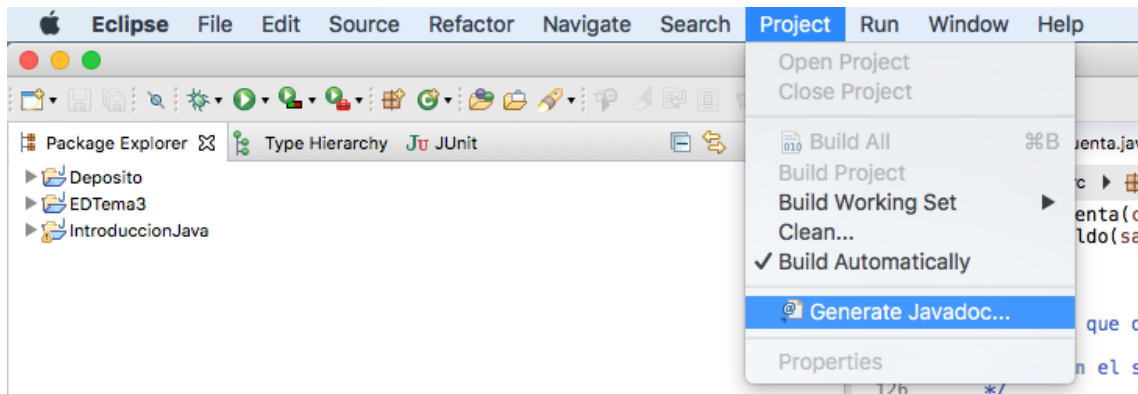
```

Main.java CCuenta.java
Deposito src cuentas CCuenta
118     setCuenta(cue);
119     setSaldo(sal);
120 }
121
122 /**
123  * Método que devuelve el saldo de la cuenta del usuario.
124  *
125  * @return el saldo de la cuenta del usuario.
126  */
127 public double estado() {
128     return getSaldo();
129 }
130
131 /**
132  * Método que se ejecuta cuando el usuario intenta ingresar saldo en la cuenta.
133  *
134  * @param cantidad a ingresar
135  * @throws Exception lanza una excepción cuando el usuario mete una cantidad
136  *         negativa
137  */
138 public void ingresar(double cantidad) throws Exception {
139     if (cantidad < 0)
140         throw new Exception("No se puede ingresar una cantidad negativa.");
141     setSaldo(getSaldo() + cantidad);
142 }
143
144 /**
145  * Método que se ejecuta cuando el usuario intenta retirar saldo de la cuenta.
146  *
147  * @param cantidad a retirar
148  * @throws Exception lanza dos excepciones. En el primer caso, lanza una
149  *         excepción cuando el usuario intenta retirar de la cuenta
150  *         una cantidad negativa. En el segundo caso, lanza una
151  *         excepción cuando el usuario intenta retirar de la cuenta
152  *         una cantidad en el que no hay saldo.
153  */
154 public void retirar(double cantidad) throws Exception {
155     if (cantidad <= 0)
156         throw new Exception("No se puede retirar una cantidad negativa.");
157     if (estado() < cantidad)
158         throw new Exception("No hay suficiente saldo.");
159     setSaldo(getSaldo() - cantidad);
160 }
161 }
162

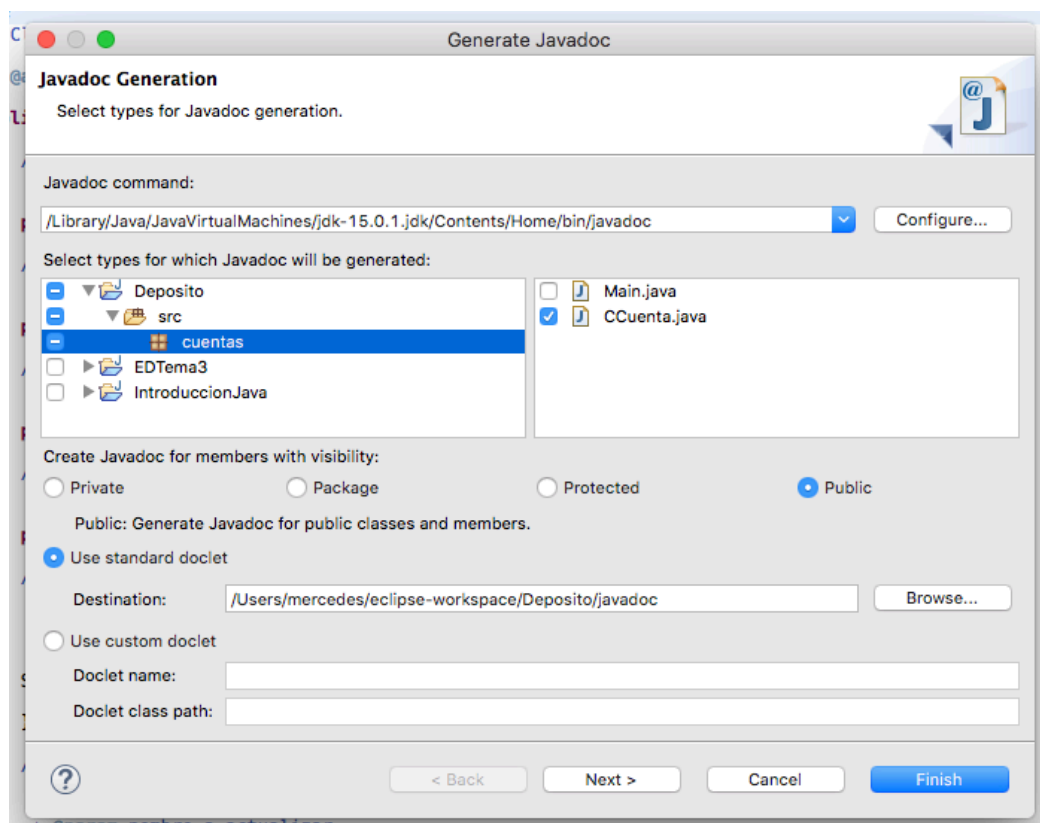
```

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

Una vez documentado el proyecto, procedo a generar la documentación Javadoc. Para ello, me dirijo al menú Project y selecciono la opción Generar Javadoc.



Aparecerá una ventana, selecciono el proyecto al que le quiero generar la documentación Javadoc. Selecciono Deposito > src > cuentas > CCuentas. Además, elijo la carpeta de destino donde guardar todos los ficheros. En este caso, he creado un nuevo directorio llamado "javadoc".



URL del repositorio GitHub.

https://github.com/mercrupor/ED_Tema4