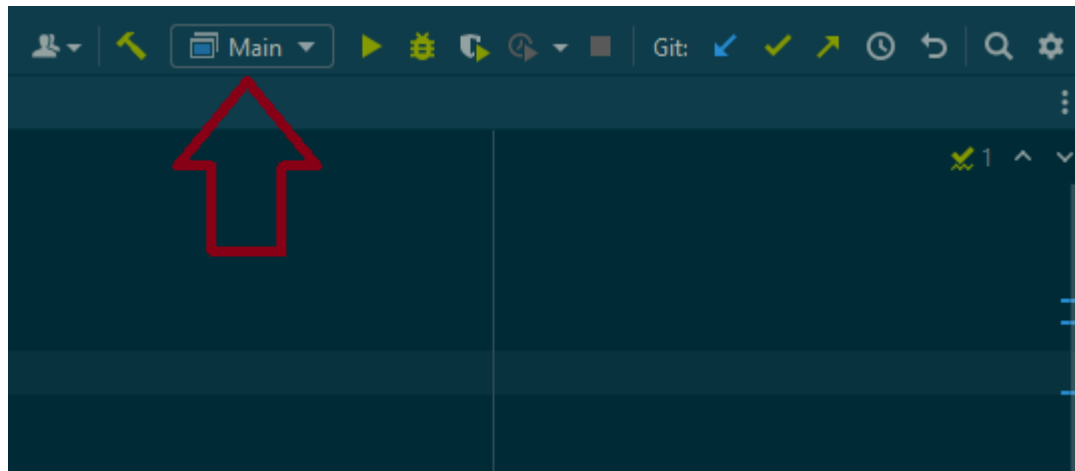


# **Actividad 4: Entornos de Desarrollo.**

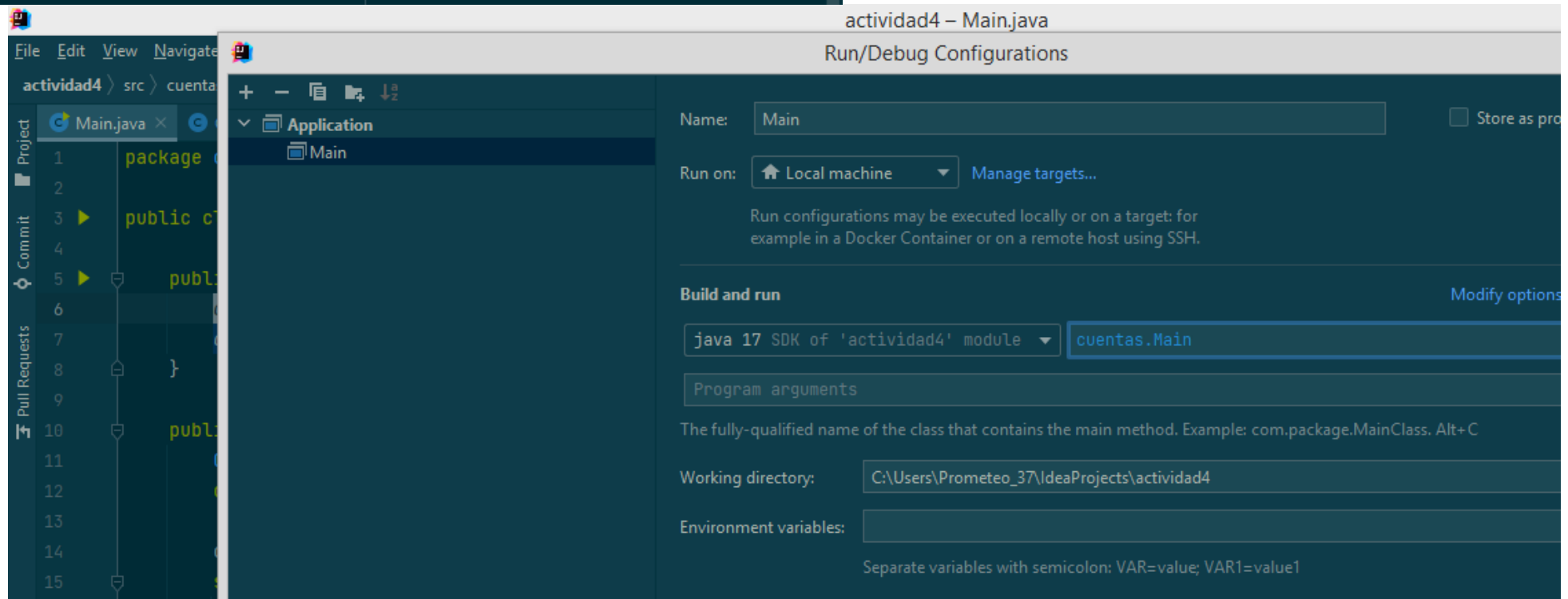
Israel David Ramos Caldera.

## **APARTADO 1: REFACTORIZACIÓN.**

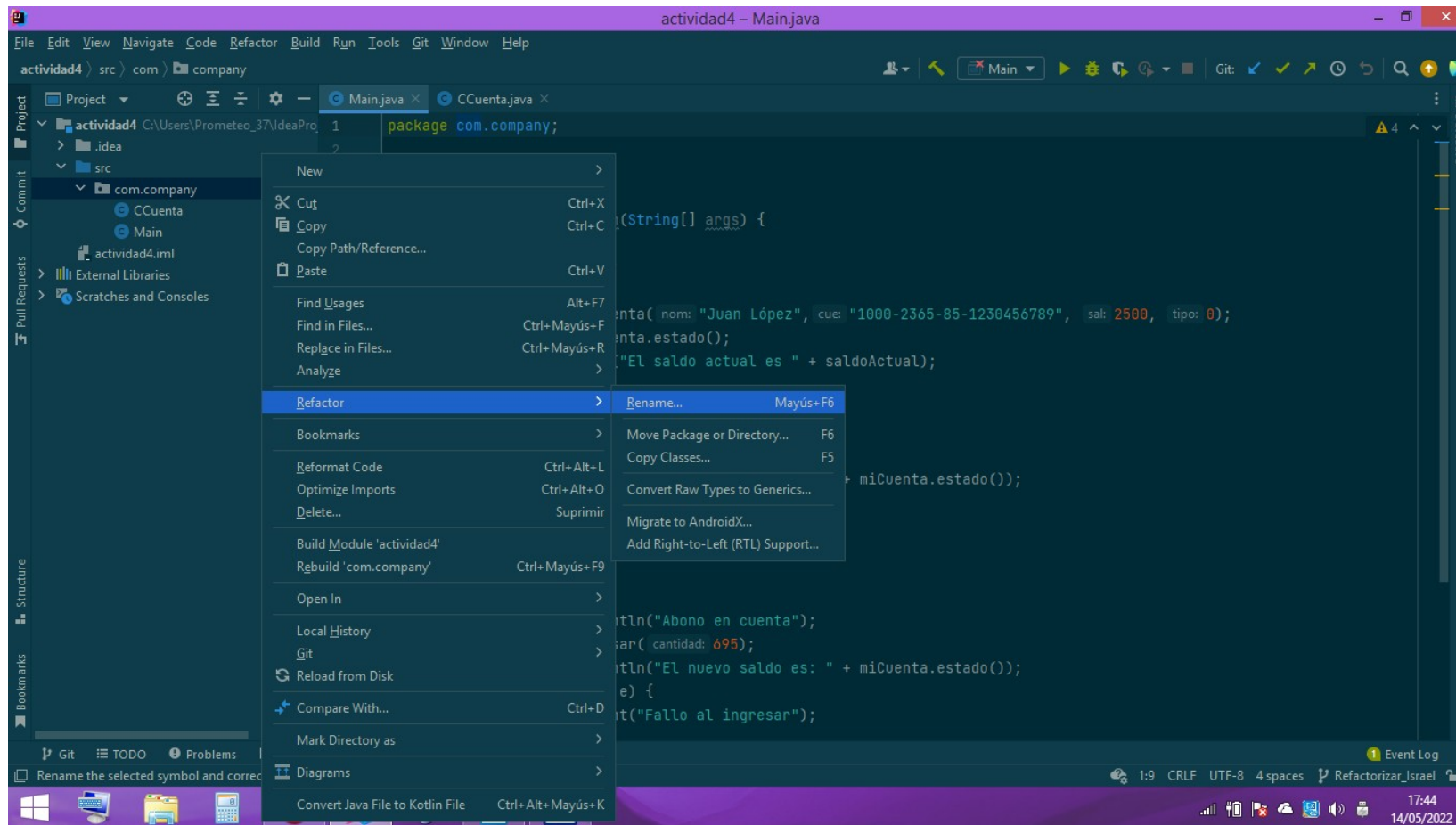
Punto 1: Las clases deberán formar parte del paquete “*cuentas*”.



En el menú desplegable seleccionamos Main para que se nos abra la ventana mostrada debajo. Sustituímos el nombre del paquete. En este caso com.company por cuentas.

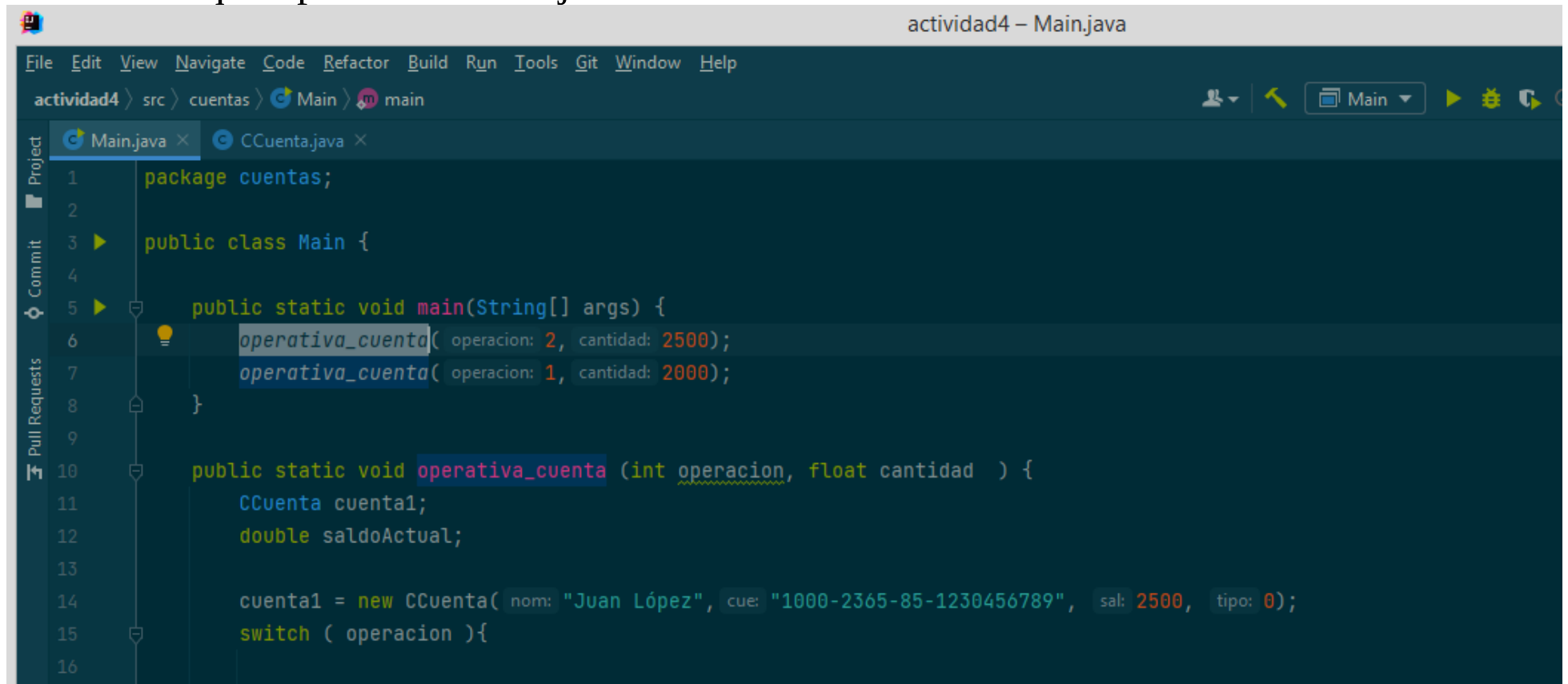


## Punto 2: Cambiar el nombre de la variable “*miCuenta*” por “*cuenta1*”:



Seleccionamos la variable a refactorizar y con un clic derecho nos dirigimos directamente a la opción “*Refactor*” para introducir el nuevo nombre. Realizamos el cambio: *micuenta* por *cuenta1* en la opción *RENAME*.

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



The screenshot shows an IDE window titled "actividad4 - Main.java". The code is in the `src > cuentas > Main` package. The `Main` class contains a `main` method and a new static method `operativa_cuenta`. The `main` method calls `operativa_cuenta` twice with different parameters. The `operativa_cuenta` method is implemented with a `switch` statement and a `CCuenta` object.

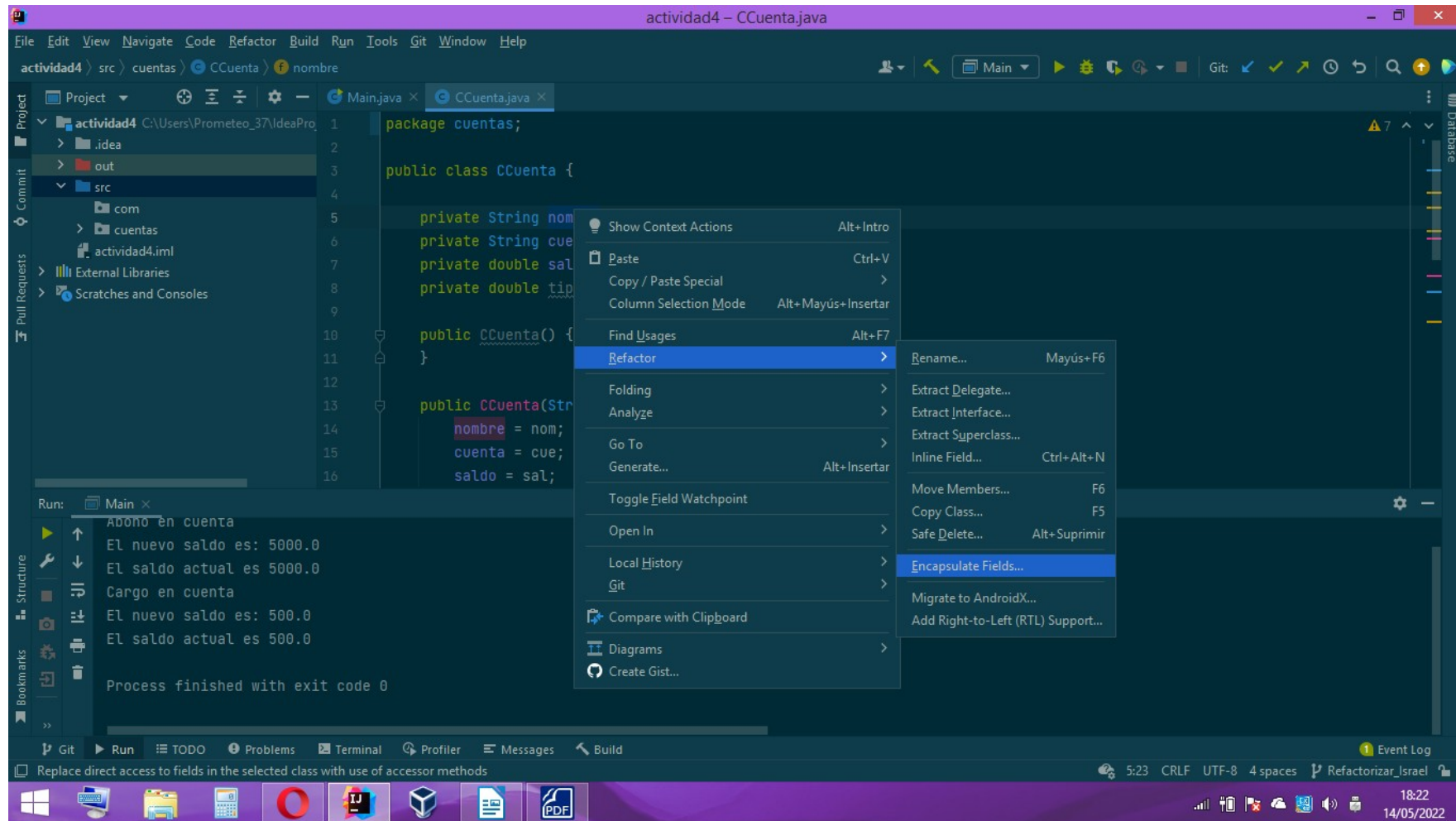
```
1 package cuentas;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         operativa_cuenta( operacion: 2, cantidad: 2500);
7         operativa_cuenta( operacion: 1, cantidad: 2000);
8     }
9
10    public static void operativa_cuenta (int operacion, float cantidad ) {
11        CCuenta cuenta1;
12        double saldoActual;
13
14        cuenta1 = new CCuenta( nom: "Juan López", cue: "1000-2365-85-1230456789", sal: 2500, tipo: 0);
15        switch ( operacion ){
16
```

**Punto 4:** Añadir dos nuevos parámetros al método *operativa\_cuenta*, el primero tendrá el nombre de *operación* y de tipo entero (nos indicará el tipo de operación a realizar) y el segundo tendrá el nombre *cantidad* y de tipo float.



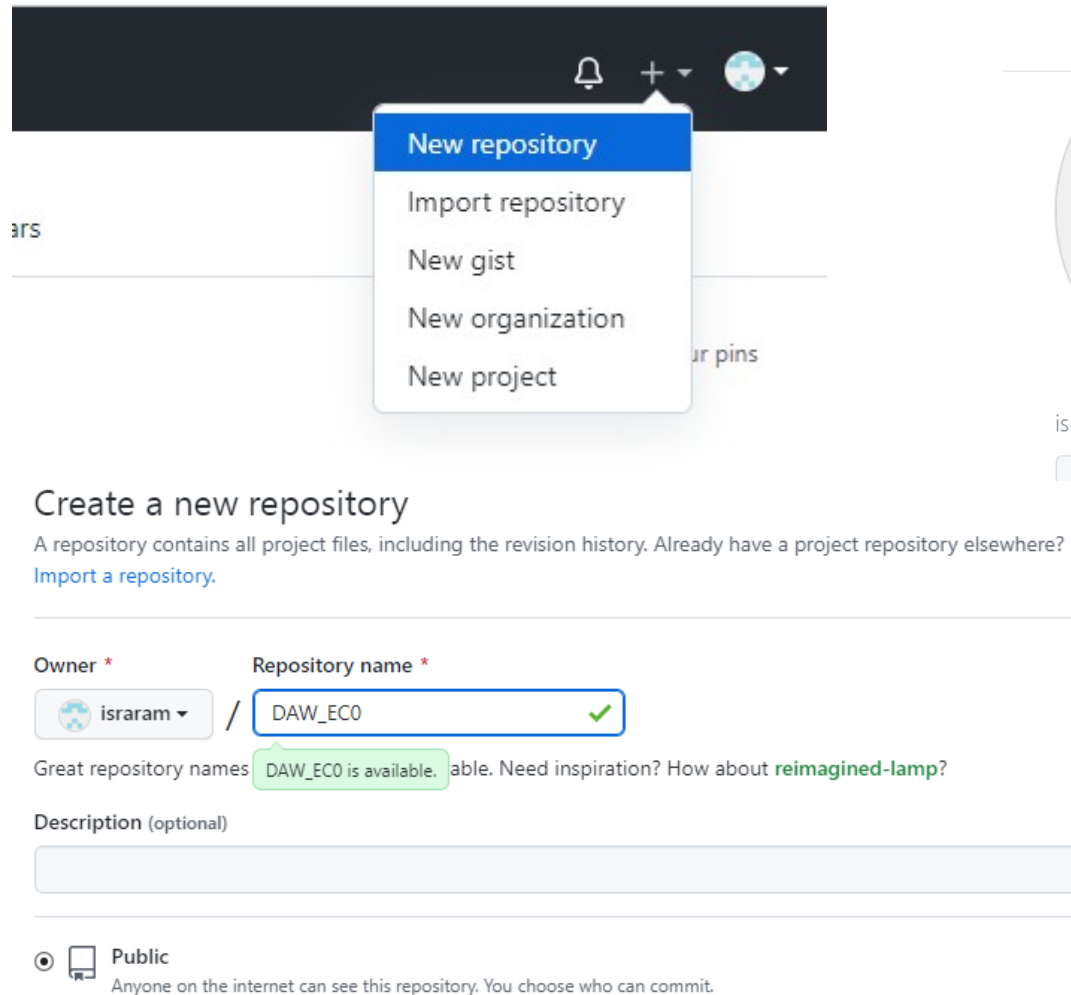
```
actividad4 – Main.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
actividad4 > src > cuentas > Main > main
Main.java x CCuenta.java x
1 package cuentas;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         operativa_cuenta( operacion: 2, cantidad: 2500);
7         operativa_cuenta( operacion: 1, cantidad: 2000);
8     }
9
10    public static void operativa_cuenta (int operacion, float cantidad ) {
11        CCuenta cuenta1;
12        double saldoActual;
13
14        cuenta1 = new CCuenta( nom: "Juan López", cue: "1000-2365-85-1230456789", sal: 2500, tipo: 0);
15        switch ( operacion ){
16
```

## Punto 5: Encapsular los atributos de la clase *cuenta*:



Seleccionamos los atributos y con un clic derecho nos dirigimos a *Refactor/Encapsulate fields*.

**GIT: PUNTO 1:**Configurar GIT para el proyecto. Crear un repositorio público en GitHub para este proyecto llamado DAW\_EC04.



The screenshot shows the 'Create a new repository' form on GitHub. At the top, there's a navigation bar with a search bar and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, a user profile section for 'israram' is visible, showing a profile picture, a bio, and a '5 contributions in the last year' calendar. The main form has a title 'Create a new repository' and a subtitle 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. The form fields include 'Owner' (set to 'israram'), 'Repository name' (set to 'DAW\_EC0' with a green checkmark), and 'Description (optional)'. Below the description field, there's a 'Public' radio button selected, with a note: 'Anyone on the internet can see this repository. You choose who can commit.'

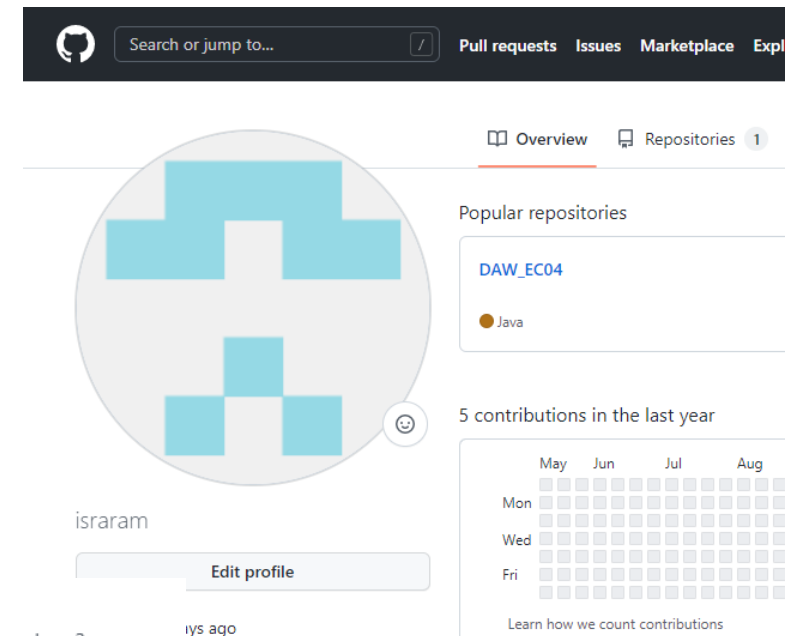
Owner \*      Repository name \*

israram / DAW\_EC0 ✓

Great repository names DAW\_EC0 is available. Need inspiration? How about [reimagined-lamp?](#)

Description (optional)

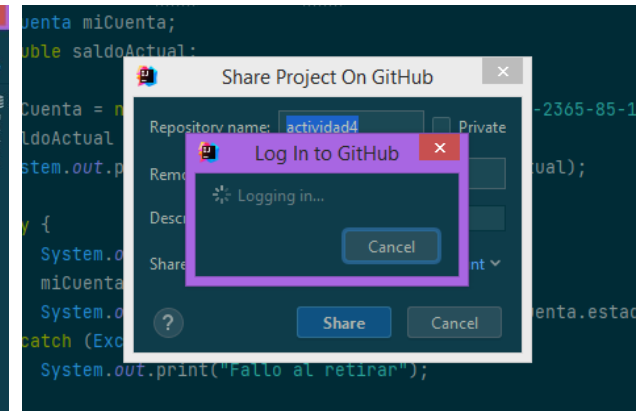
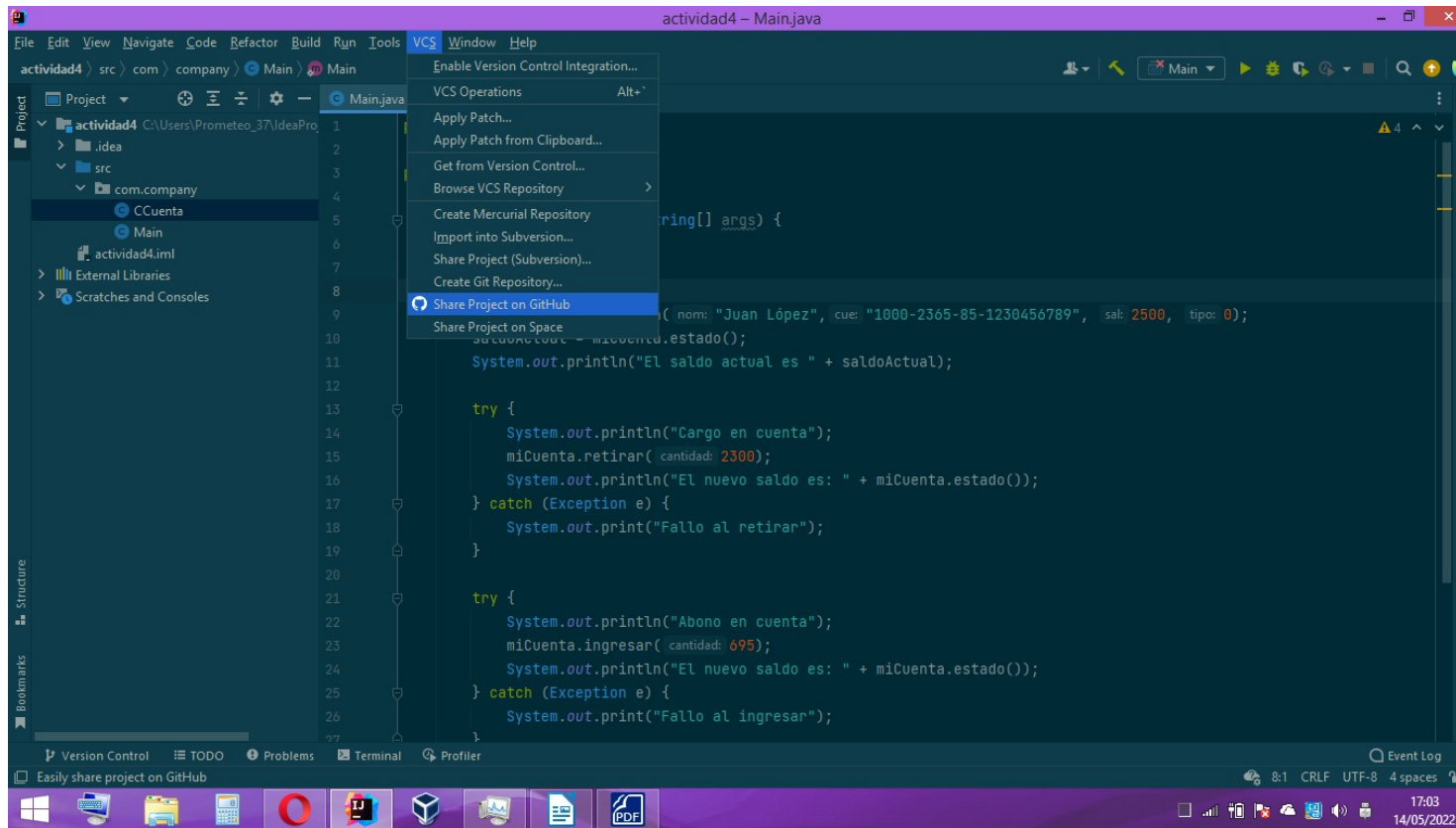
☒ Public  
Anyone on the internet can see this repository. You choose who can commit.



Hemos creado un perfil @IsraRam y a través del icono + creamos un nuevo repositorio dándole nombre. Lo mantenemos público.

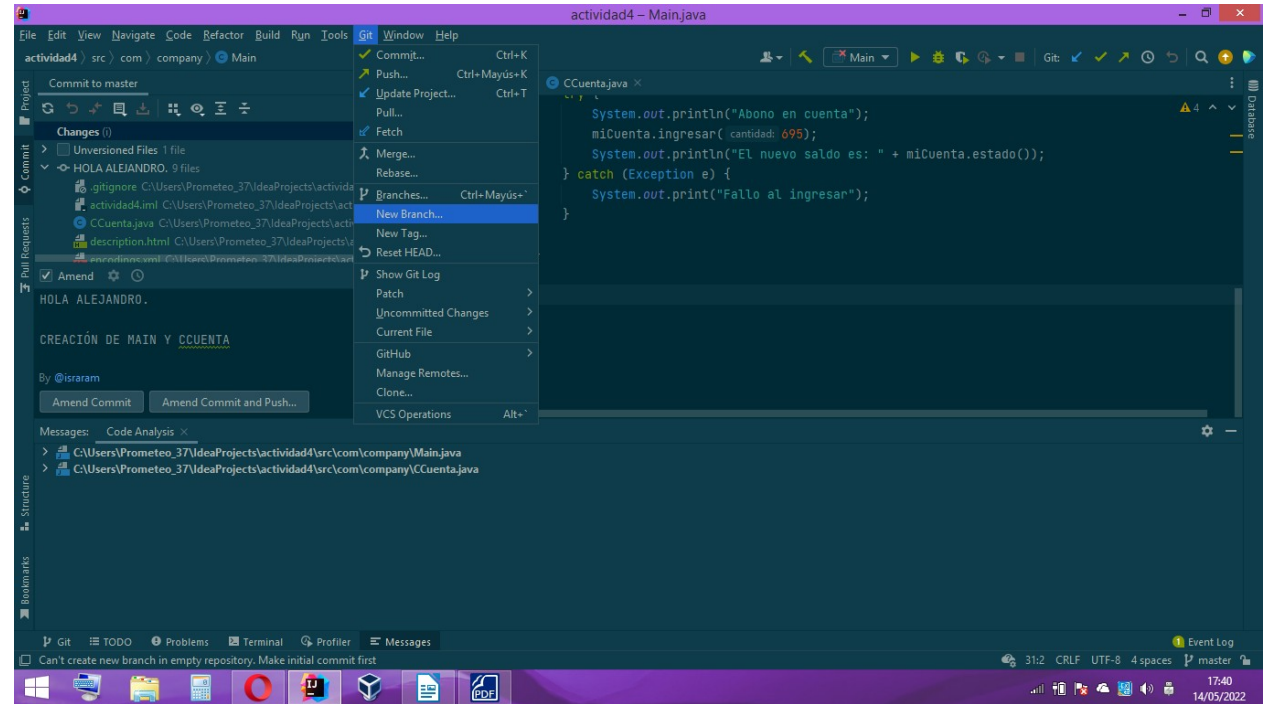
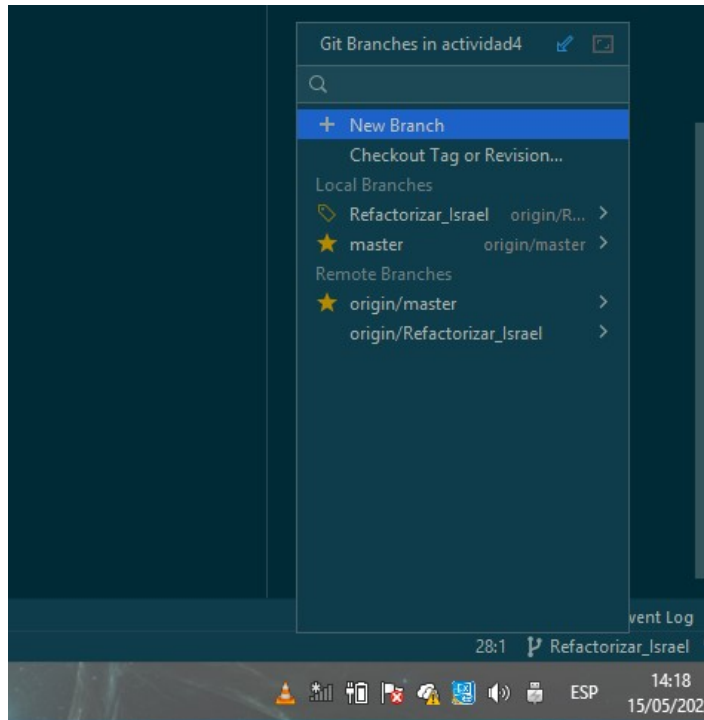


**PUNTO 2:** Sube el código a una rama que tengas como principal.



En nuestro IDE, sincronizamos nuestra cuenta de GitHub. Subimos el código a la rama principal desplegando el menú y clicando en Share Project.

**PUNTO 3:** Crea una rama a parte para realizar los cambios indicados en el apartado anterior (Refactorización). Realiza, al menos, una operación commit. Comentando el resultado de la ejecución y/o cambios



Podemos crear ramas en el menú inferior derecho o desde la pestaña GIT, seleccionando “New Branch”. Añadimos Commit.

Comparing master...Refacto... traduce amend - Buscar co... | +

github.com/israram/DAW\_EC04/compare/master...Refactorizar\_Israel

Ejercicios de Tipos... Manolo Hidalgo ~...

Load diff

Some generated files are not rendered by default. [Learn more.](#)

11 actividad4.iml

@@ -0,0 +1,11 @@

1 + <?xml version="1.0" encoding="UTF-8"?>

2 + <module type="JAVA\_MODULE" version="4">

3 + <component name="NewModuleRootManager" inherit-compiler-output="true">

4 + <exclude-output />

5 + <content url="file://\$MODULE\_DIR\$">

6 + <sourceFolder url="file://\$MODULE\_DIR\$/src" isTestSource="false" />

7 + </content>

8 + <orderEntry type="inheritedJdk" />

9 + <orderEntry type="sourceFolder" forTests="false" />

10 + </component>

11 + </module>

66 src/cuentas/CCuenta.java

@@ -0,0 +1,66 @@

1 + package cuentas;

2 +

3 + public class CCuenta {

4 +

5 + private String nombre;

Windows Taskbar

18:36 14/05/2022

## APARTADO 3: JAVADOC.

### Punto 1: Inserta comentarios JavaDoc en la clase *cuenta*.

```
/**
 *
 * @param cantidad se mete la cantidad que se quiere ingresar al parámetro saldo
 * @throws Exception para no ingresar cantidades negativas
 */

public void ingresar(double cantidad) throws Exception {
    if (cantidad < 0) {
        throw new Exception("No se puede ingresar una cantidad negativa");
    }
    saldo = saldo + cantidad;
}

/**
 *
 * @param cantidad se mete la cantidad que se quiere retirar al parámetro saldo
 * @throws Exception para no retirar cantidades negativas o mayores que el saldo actual
 */
public void retirar(double cantidad) throws Exception {
    if (cantidad < 0) {
        throw new Exception("No se puede retirar una cantidad negativa");
    }
    if (saldo < cantidad) {
        throw new Exception("No se puede retirar más del saldo actual");
    }
    saldo = saldo - cantidad;
}
```

Everything is up-to-date

```
}

/**
 * @param nom le da un valor al atributo nombre
 * @param cue le da un valor al atributo cuenta
 * @param sal le da un valor al atributo saldo
 * @param tipo le da un valor a tipoInteres en caso de que se llegue a implementar en el constructor
 */

public CCuenta(String nom, String cue, double sal, double tipo) {
    nombre = nom;
    cuenta = cue;
    saldo = sal;
}

/**
 *
 * @param cantidad se mete la cantidad que se quiere ingresar al parámetro saldo
 * @throws Exception para no ingresar cantidades negativas
 */
public void ingresar(double cantidad) throws Exception {
    if (cantidad < 0) {
        throw new Exception("No se puede ingresar una cantidad negativa");
    }
    saldo = saldo + cantidad;
}
```

Punto 2: generamos el documento JAVADOC haciendo clic sobre TOOLS/Generate JavaDoc...

