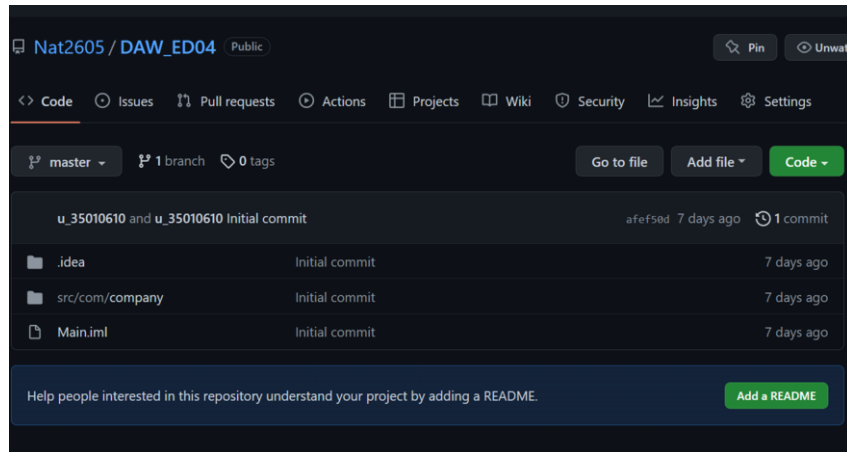


GIT

1. (0.5 ptos) Configurar GIT para el proyecto. Crear un repositorio público en GitHub para este proyecto llamado DAW_ED04.

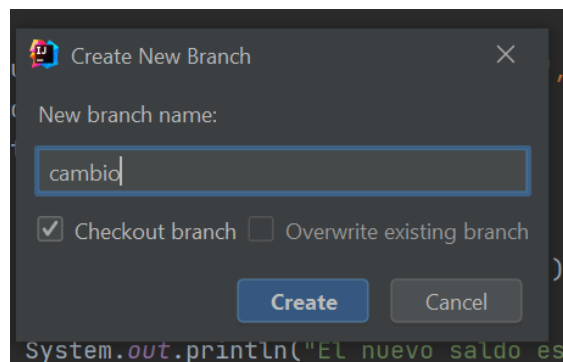


-Primero creamos el repositorio de DAW_ED04, lo configuramos en el entorno que mi caso fue el intelliJ y luego lo miramos en la propia en página de github.

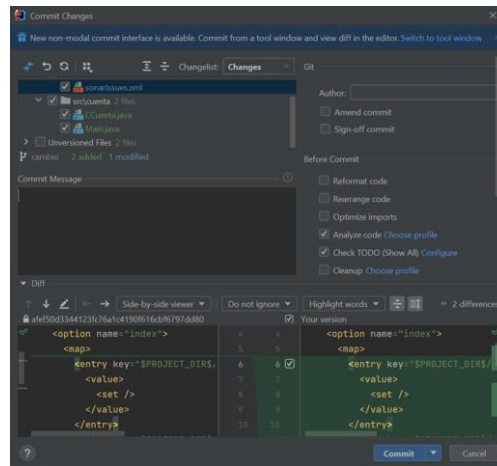
2. (0.5 ptos) Sube el código a una rama que tengas como principal

Al generar el repositorio desde el intelliJ compartimos el código y así luego lo tenemos como sale en la imagen de arriba

3. (1 pto) 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



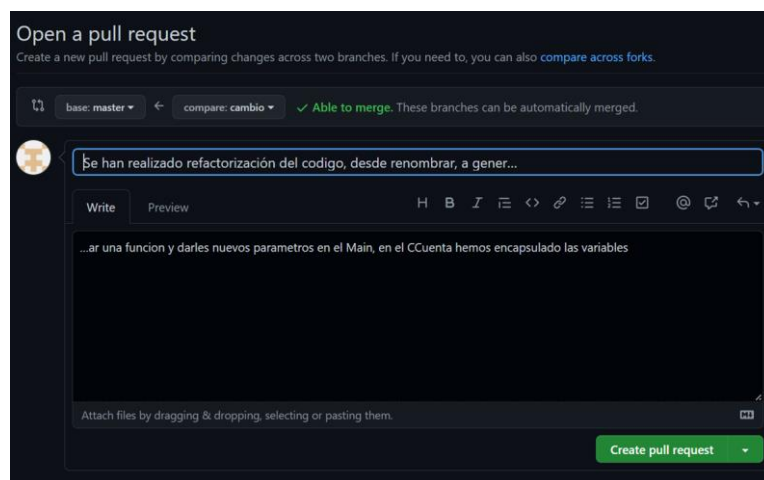
Generamos la rama con la herramienta en intelliJ que hay debajo a la derecha y ponemos el nombre de nuestra rama



-Arriba a la derecha en IntelliJ, tenemos el icono de un check verde, donde al pulsar tendremos la ventana del commit, donde podremos comentar los cambios y realizar el propio commit.

4. (1 pto) Luego, integra los cambios desde la rama secundaria a la rama principal realizando las operaciones pertinentes.

Tras lo anterior hacemos un push que nos sirve para actualizar nuestra rama en remoto, que de nuevo esta en los iconos de la derecha. Y luego para subir estos cambios a nuestra rama principal haremos desde github un Pull request



Creamos el pull request, y nos saldrá para confirmar el merge como en la imagen de abajo, que nos servirá para confirmar nuestros cambios a la rama principal.



5. (1 pto) Realiza los pasos 3 y 4 para el siguiente apartado (JavaDoc)

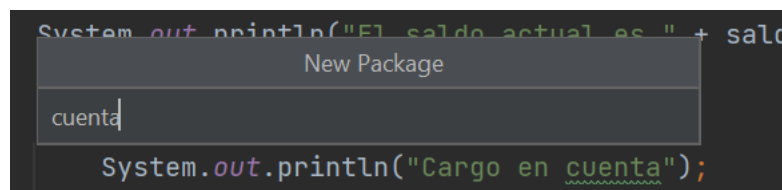
-Ahora haremos los mismos pasos anteriores, crearemos una rama nueva que en mi caso se llamará javadoc, donde pasaremos a documentar el código

```
67  /**
68   * Metodos de la clase CCuenta
69   * @param cantidad para ingresar de la cuenta
70   * @throws Exception si se cumple la condición muestra un mensaje
71   */
72  public void ingresar(double cantidad) throws Exception {
73      if (cantidad < 0) {
74          throw new Exception("No se puede ingresar una cantidad negativa");
75      }
76      setSaldo(getSaldo() + cantidad);
77  }
78
79  /**
80   * Metodos de la clase CCuenta
81   * @param cantidad para retirar de la cuenta
82   * @throws Exception si se cumple la condición muestra un mensaje
83   */
```

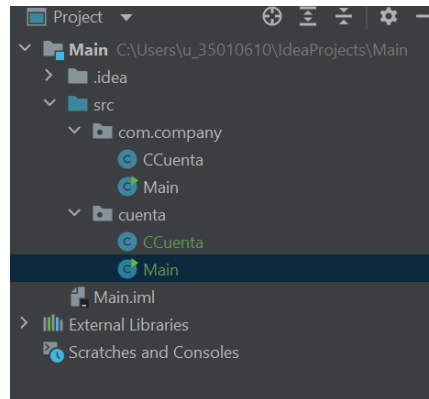
Creamos la documentación de JavaDoc en la clase CCuenta, donde hablamos de los parámetros y métodos, luego hacemos el proceso de los apartados anteriores, es decir, un commit, push y pull request.

Refactorización

1. Las clases deberán formar parte del paquete cuentas



-Nos ponemos encima del src de nuestro proyecto, pinchamos con el click derecho y le damos a package, luego ponemos el nombre que queramos como sale en la imagen en este caso será cuenta.



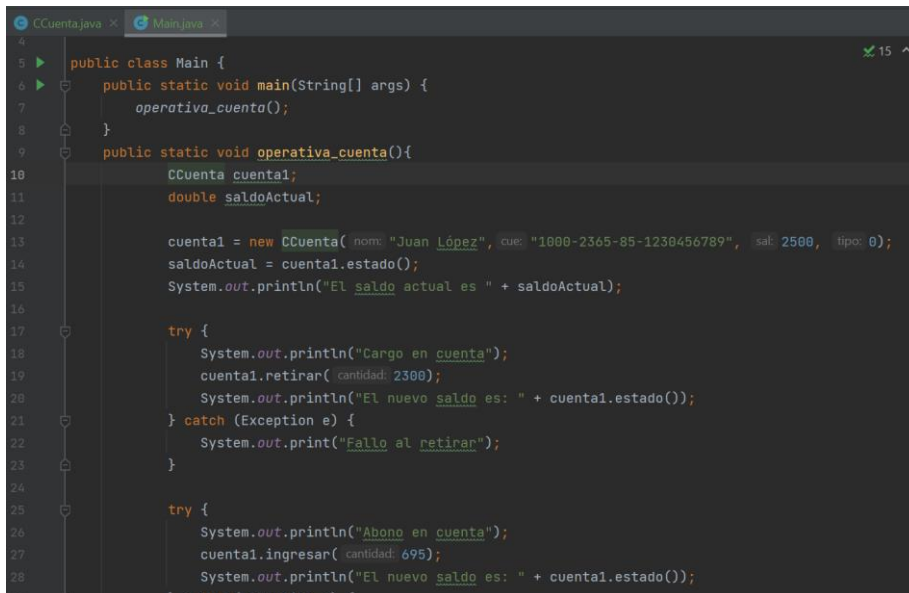
-Luego generamos en el nuevo package las clases Main y Ccuenta, para ello nos pondríamos encima del package cuenta y le damos al click derecho y generamos las nuevas clases

2. (1 pto) Cambiar el nombre de la variable "miCuenta" por "cuenta1"

```
public class Main {  
    public static void main(String[] args) {  
        com.company.CCuenta miCuenta;  
        double saldoActual;  
  
        miCuenta = new CCuenta( nombre: "Juan López", cue: "1000-2365-85-1230456789", sal:  
        saldoActual = miCuenta.estado();  
        System.out.println("El saldo actual es " + saldoActual);  
  
        try {  
            System.out.println("Cargo en cuenta");  
            miCuenta.retirar( cantidad: 2300);  
            System.out.println("El nuevo saldo es: " + miCuenta.estado());  
        } catch (Exception e) {  
            System.out.print("Fallo al retirar");  
        }  
  
        try {  
            System.out.println("Abono en cuenta");  
            miCuenta.ingresar( cantidad: 695);  
            System.out.println("El nuevo saldo es: " + miCuenta.estado());  
        } catch (Exception e) {  
            System.out.print("Fallo al ingresar");  
        }  
    }  
}
```

-Nos ponemos encima de la variable miCuenta y le damos con el click derecho y en el menú contextual seleccionamos refactorizar y luego rename, tras ello podremos cambiar el nombre de miCuenta en todo el código y pasa a ser cuenta1.

3. (1 pto) Introducir el método operativa cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.



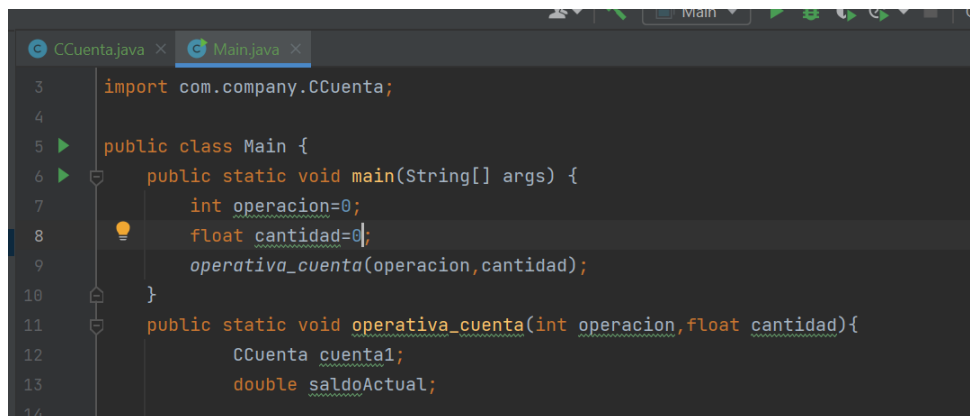
```

4
5 public class Main {
6     public static void main(String[] args) {
7         operativa_cuenta();
8     }
9     public static void operativa_cuenta(){
10         CCuenta cuenta1;
11         double saldoActual;
12
13         cuenta1 = new CCuenta( nom: "Juan López", cue: "1000-2365-85-1230456789", sal: 2500, tipo: 0);
14         saldoActual = cuenta1.estado();
15         System.out.println("El saldo actual es " + saldoActual);
16
17         try {
18             System.out.println("Cargo en cuenta");
19             cuenta1.retirar( cantidad: 2300);
20             System.out.println("El nuevo saldo es: " + cuenta1.estado());
21         } catch (Exception e) {
22             System.out.print("Fallo al retirar");
23         }
24
25         try {
26             System.out.println("Abono en cuenta");
27             cuenta1.ingresar( cantidad: 695);
28             System.out.println("El nuevo saldo es: " + cuenta1.estado());
29         } catch (Exception e) {
30             System.out.print("Fallo al ingresar");
31         }
32     }
33 }

```

-Creamos el método de operativa_cuenta donde metemos todas las sentencias de la clase Main

4. (1 pto) Añadir dos nuevos parámetros al método operativa_cuenta, el primero tendrá el nombre operacion y de tipo entero (nos indicará el tipo de operación a realizar) y el segundo tendrá el nombre cantidad y de tipo float.



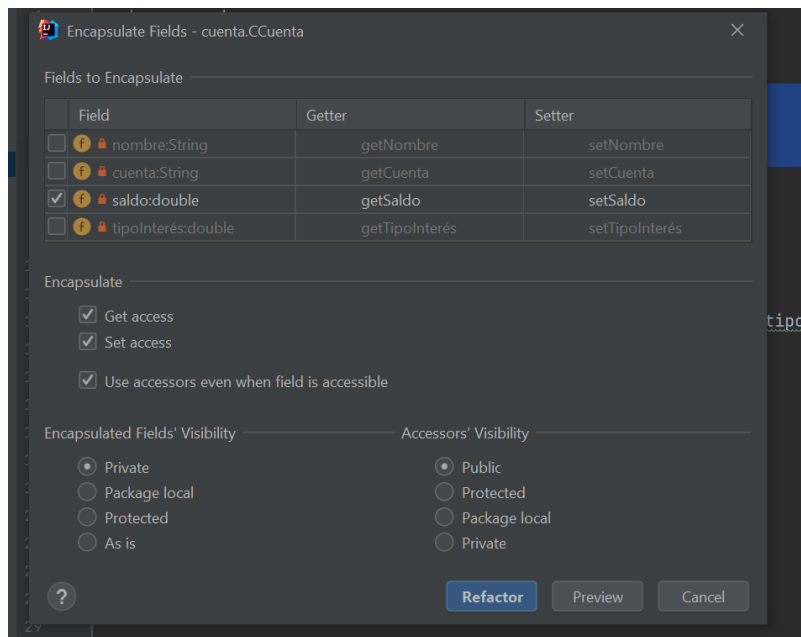
```

3 import com.company.CCuenta;
4
5 public class Main {
6     public static void main(String[] args) {
7         int operacion=0;
8         float cantidad=0;
9         operativa_cuenta(operacion,cantidad);
10    }
11    public static void operativa_cuenta(int operacion,float cantidad){
12        CCuenta cuenta1;
13        double saldoActual;
14    }
15 }

```

-Metemos los nuevos parámetros, creamos las variables en el main para así poder llamarla dentro de la función que tenemos en el main.

5. (1 pto) Encapsular los atributos de la clase Ccuenta.



-Seleccionamos todos los atributos de la clase Cuenta, volvemos a dar a refactorizar y saldrá una opción de encapsular, al darle nos saldrá el menú de la imagen de arriba donde seleccionaremos todos los atributos y luego daremos a refactorizar.