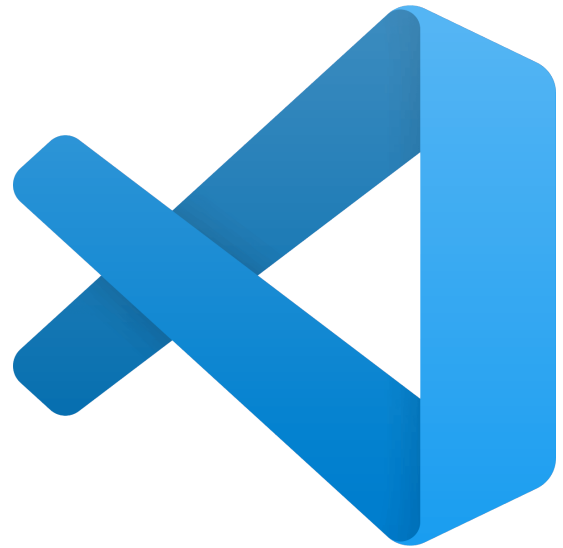


# *PRÀCTICA*



Mario Callejón Criado  
Joan Vicens Muntaner  
Entorns de desenvolupament  
S1W

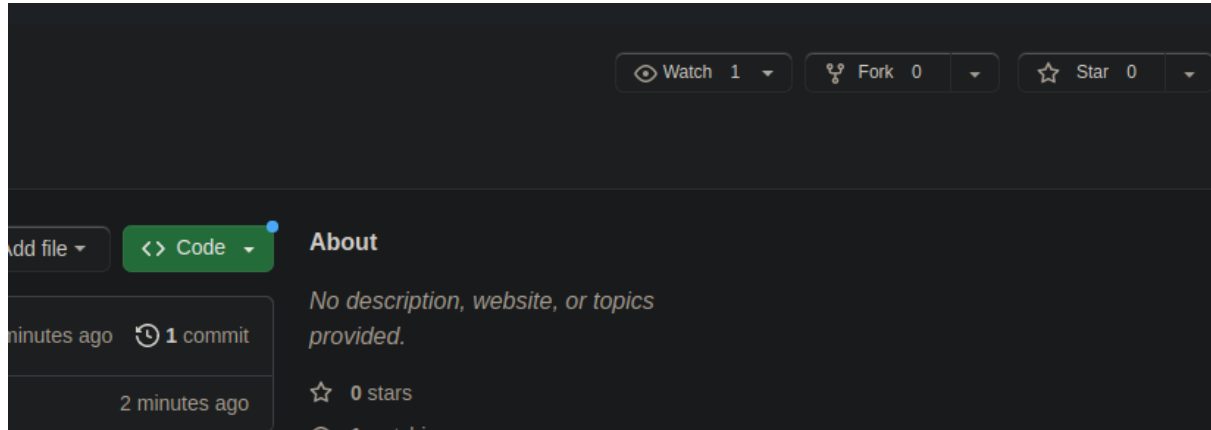
# Índice

<b>Índice</b>	<b>2</b>
<b>GIT y GITHUB</b>	<b>3</b>
Haz un fork del repositorio remoto de tu compañero en Github.	3
Elige con tu compañero/a quien implementará cada método.	5
Crea dos ramas para cada clase.	5
No olvides hacer pull del repositorio en la rama máster antes de empezar con la siguiente rama si tu compañero ha realizado cambios en el proyecto.	6
Crea otras dos ramas para comenzar a trabajar en la clase registroUsuario y sigue el mismo procedimiento que con la primera.	6
<b>REFACTORIZACIÓN</b>	<b>7</b>
Las clases deberán formar parte del paquete “registro”.	7
Introduce el método init en la clase registroUsuario, que englobe las sentencias de la clase Main que operan con el objeto validaUsuario, pasando por parámetro el objeto en cuestión.	9
Añade un nuevo parámetro al método compruebaNombre, de nombre “usuarios” y de tipo array de Strings y cambia el orden de los parámetros de los métodos si tienen más de uno.	10
Convierte las variables locales en atributos.	11
Crea varios constructores para la clase validarCampos.	12
Encapsula los atributos de la clase validarCampos (Getters i Setters).	12
<b>Comentarios JavaDoc</b>	<b>14</b>

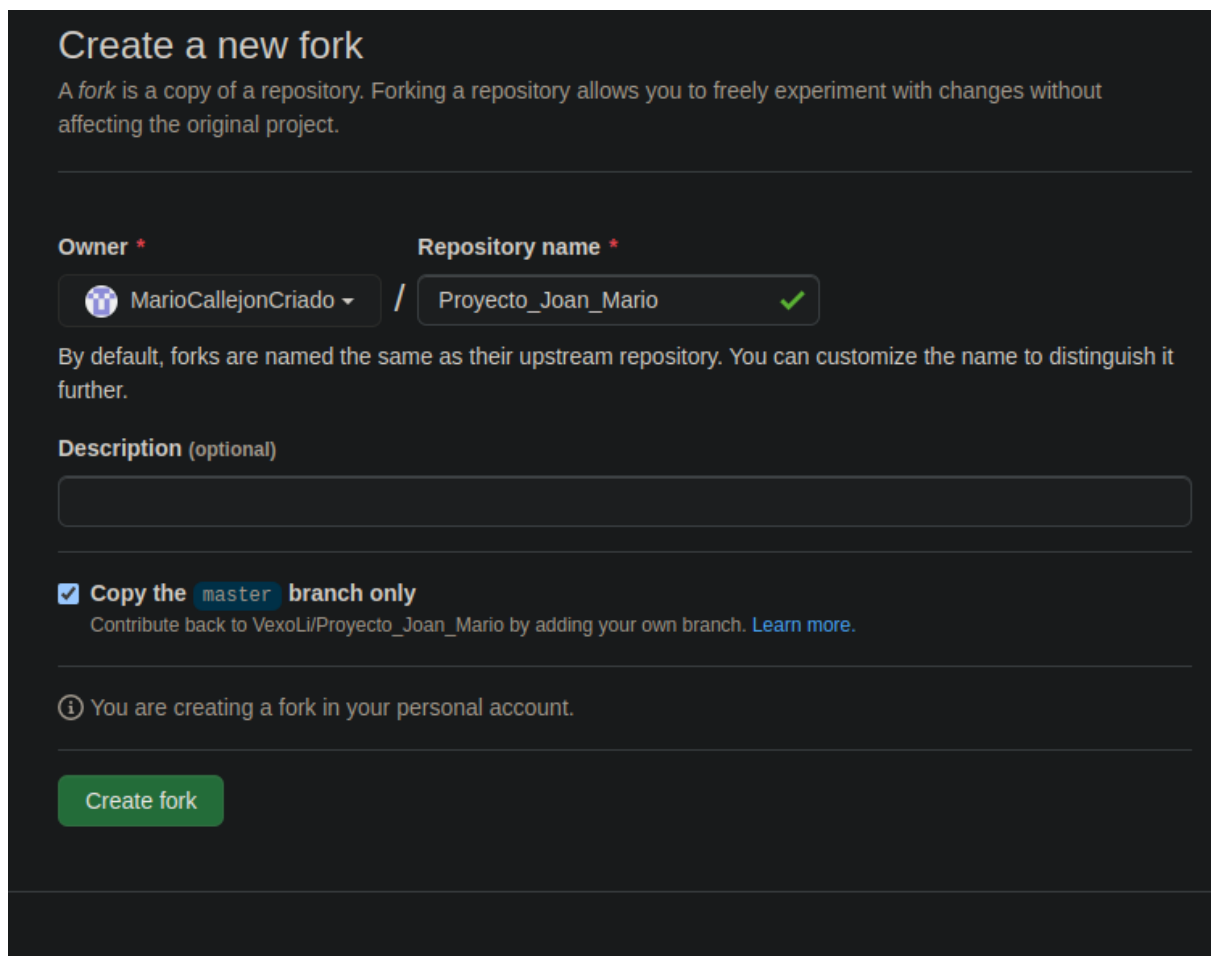
# ***GIT y GITHUB***

Haz un fork del repositorio remoto de tu compañero en Github.

Para comenzar a hacer este ejercicio, lo que tenemos que hacer es esperar a que el compañero haya subido el código al GitHub, en mi caso, él lo ha subido con las dos primeras refactorizaciones que he hecho.

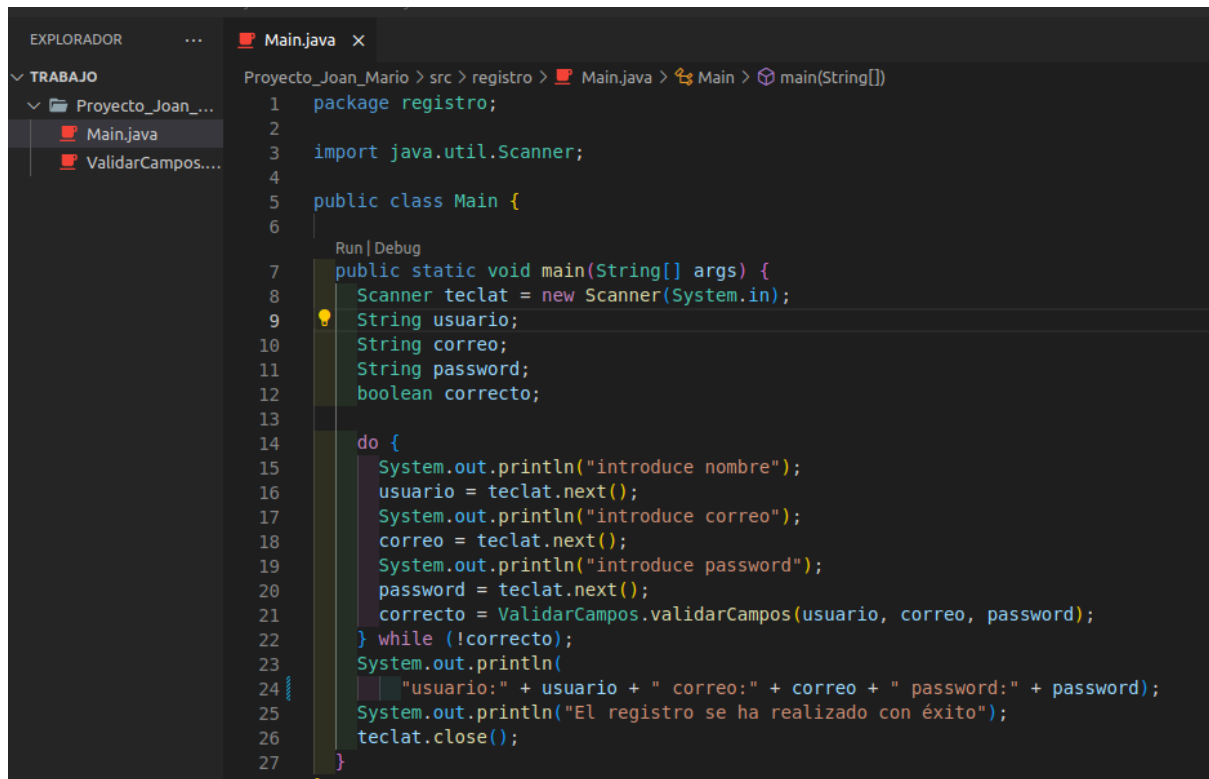


Después tenemos que ir al GitHub y darle al fork para bajarnos lo que él ha hecho



Aquí le tenemos que dar a create fork

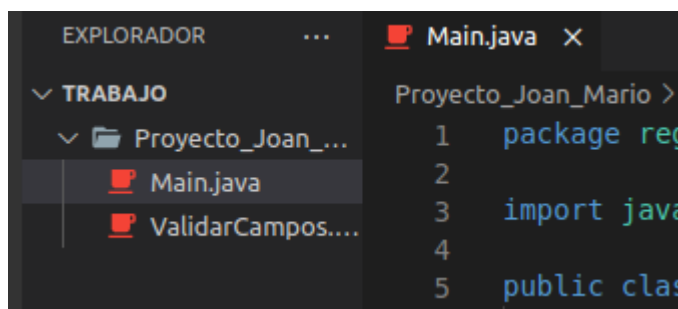
Una vez tenemos el fork hecho, tenemos que copiar el código que nos ha dado y meternos al visual y hacer un git clone y el url que nos ha dado.



```
1 package registro;
2
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Scanner teclado = new Scanner(System.in);
9         String usuario;
10        String correo;
11        String password;
12        boolean correcto;
13
14        do {
15            System.out.println("introduce nombre");
16            usuario = teclado.next();
17            System.out.println("introduce correo");
18            correo = teclado.next();
19            System.out.println("introduce password");
20            password = teclado.next();
21            correcto = ValidarCampos.validarCampos(usuario, correo, password);
22        } while (!correcto);
23        System.out.println(
24            "usuario:" + usuario + " correo:" + correo + " password:" + password);
25        System.out.println("El registro se ha realizado con éxito");
26        teclado.close();
27    }
28 }
```

Y ya tendríamos esto bajado.

Crea una carpeta en tu ordenador y descarga el contenido remoto a tu terminal. He creado la carpeta “Trabajo” que almacena el contenido que me he bajado de mi compañero.



```
1 package registro;
2
3 import java.util.Scanner;
4
5 public class Main {
```

Elige con tu compañero/a quien implementará cada método.

Sobre hacer los métodos, hemos quedado en lo siguiente, Joan hace los métodos de la clase ValidarCampos y yo hago la clase del Main.

Crea dos ramas para cada clase.

Para crear dos ramas para cada clase, lo que hemos hecho es lo siguiente, una vez me he bajado yo su proyecto haciendo el fork, he creado una rama llamada ramaMario y otra que se llama ramaJoan.

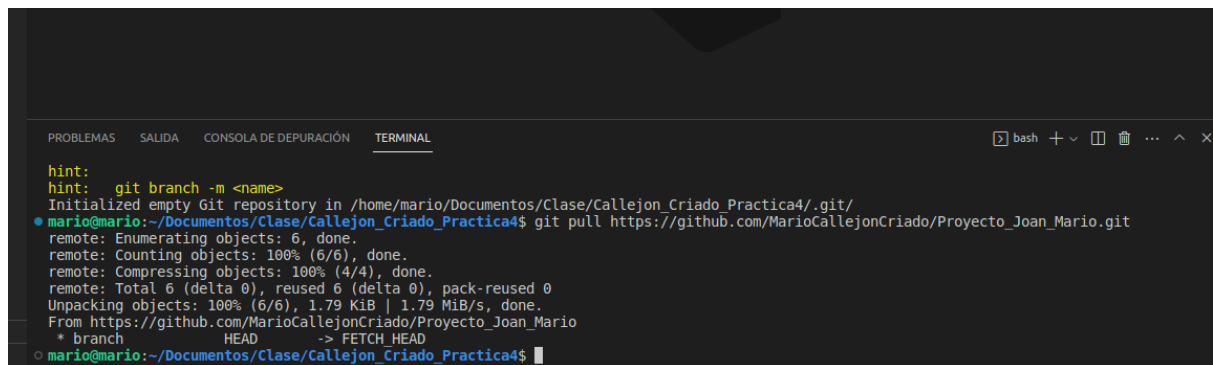
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

• mario@mario:~/Documentos/Entornos/Callejon_Criado_MarioPractica4/src/registro$ git branch ramaMario
• mario@mario:~/Documentos/Entornos/Callejon_Criado_MarioPractica4/src/registro$ git branch ramaJoan
• mario@mario:~/Documentos/Entornos/Callejon_Criado_MarioPractica4/src/registro$ git checkout
• mario@mario:~/Documentos/Entornos/Callejon_Criado_MarioPractica4/src/registro$ git branch

* master
  ramaJoan
  ramaMario
○ mario@mario:~/Documentos/Entornos/Callejon_Criado_MarioPractica4/src/registro$
```

Haz una pull request en GitHub a tu compañero para que revise los cambios.

No olvides hacer pull del repositorio en la rama máster antes de empezar con la siguiente rama si tu compañero ha realizado cambios en el proyecto.



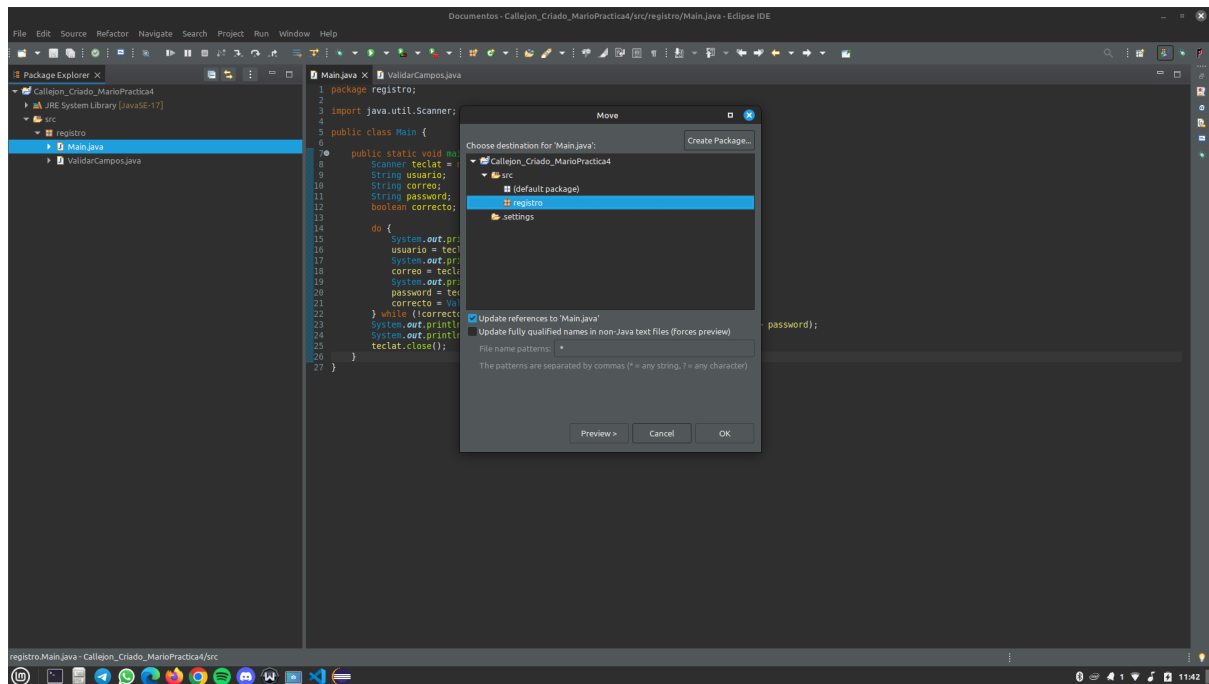
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/mario/Documentos/Clase/Callejon_Criado_Practica4/.git/
● mario@mario:~/Documentos/Clase/Callejon_Criado_Practica4$ git pull https://github.com/MarioCallejonCriado/Proyecto_Joan_Mario.git
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 1.79 KiB | 1.79 MiB/s, done.
From https://github.com/MarioCallejonCriado/Proyecto_Joan_Mario
 * branch            HEAD       -> FETCH_HEAD
● mario@mario:~/Documentos/Clase/Callejon_Criado_Practica4$
```

Crea otras dos ramas para comenzar a trabajar en la clase registroUsuario y sigue el mismo procedimiento que con la primera.

# REFACTORIZACIÓN

Las clases deberán formar parte del paquete “registro”.

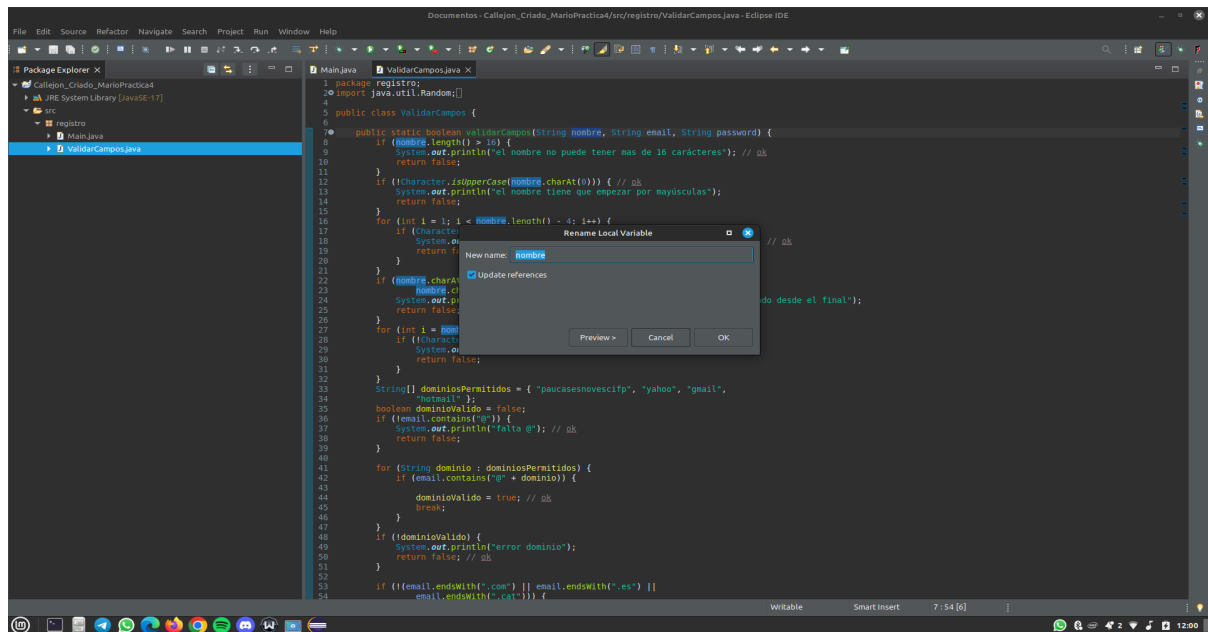
Para hacer este ejercicio, se tiene que hacer lo siguiente, en ECLIPSE, pulsamos click derecho sobre la clase que queremos meter en el paquete, por ejemplo, primero lo hacemos con la del main, le damos click derecho, le damos a “refactor” y después a “move” y nos saldrá esto:



Una vez estamos aquí, le damos a “create package” y le damos un nombre, en nuestro caso le ponemos “registro”, una vez creado, nos ponemos encima de “registro” y le damos a “ok”.

Cambia el nombre de la variable "nombre", o como la hayas denominado, por "nombreUsuario".

Para hacer este ejercicio, tenemos que hacer lo siguiente, iremos a donde hemos creado la variable “nombre” que en nuestro caso es en “ValidarCampos”, una vez hemos encontrado la variable, la seleccionamos y le damos a click derecho, “refactor” y después le damos a “rename”, después, lo que hemos hecho es darle a opciones y darle a “open rename log” para que me salga esta ventana, pero es opcional, una vez estamos aquí, ponemos el nombre que necesitamos y después le damos a “ok” y ya está

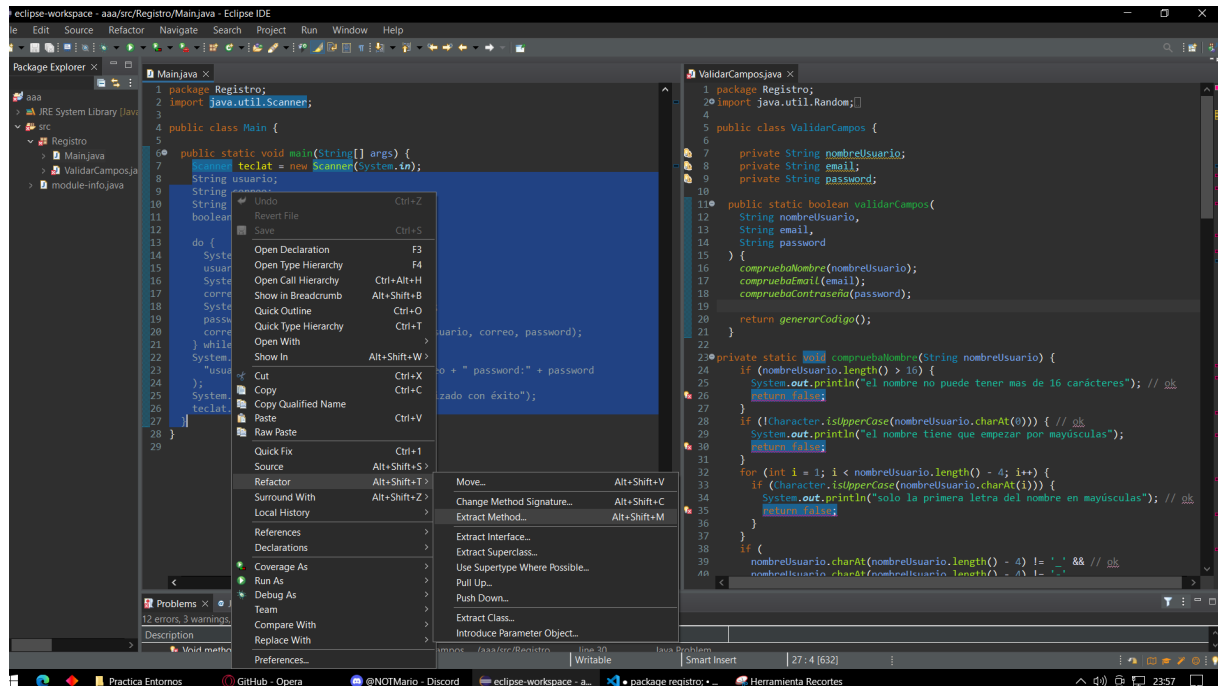


```
public static boolean validarCampos(String nombreUsuario, String email, String password) {
    if (nombreUsuario.length() > 16) {
        System.out.println("el nombre no puede tener mas de 16 caracteres"); // ok
        return false;
    }
    if (!Character.isUpperCase(nombreUsuario.charAt(0))) { // ok
        System.out.println("el nombre tiene que empezar por mayúsculas");
        return false;
    }
    for (int i = 1; i < nombreUsuario.length() - 4; i++) {
        if (Character.isUpperCase(nombreUsuario.charAt(i))) {
            System.out.println("solo la primera letra del nombre en mayúsculas"); // ok
            return false;
        }
    }
    if (nombreUsuario.charAt(nombreUsuario.length() - 4) != '_' && nombreUsuario.charAt(nombreUsuario.length() - 4) != '-') {
        System.out.println("Falta el guión o no esta en la cuarta posicion contando desde el final");
        return false;
    }
    for (int i = nombreUsuario.length() - 3; i < nombreUsuario.length(); i++) { // ok
        if (!Character.isDigit(nombreUsuario.charAt(i))) {
            System.out.println("El nombre tiene que terminar con tres dígitos");
            return false;
        }
    }
    String[] dominiosPermitidos = { "paucaesnovescifp", "yahoo", "gmail", "hotmail" };
    boolean dominioValido = false;
    if (!email.contains("@")) {
        System.out.println("falta @"); // ok
        return false;
    }
    for (String dominio : dominiosPermitidos) {
        if (email.contains("@" + dominio)) {
```



Introduce el método init en la clase registroUsuario, que englobe las sentencias de la clase Main que operan con el objeto validaUsuario, pasando por parámetro el objeto en cuestión.

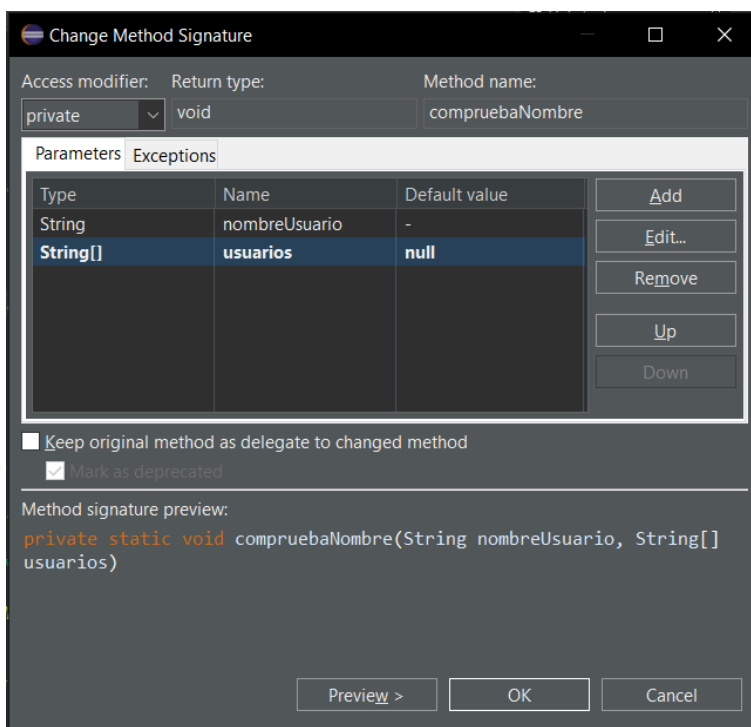
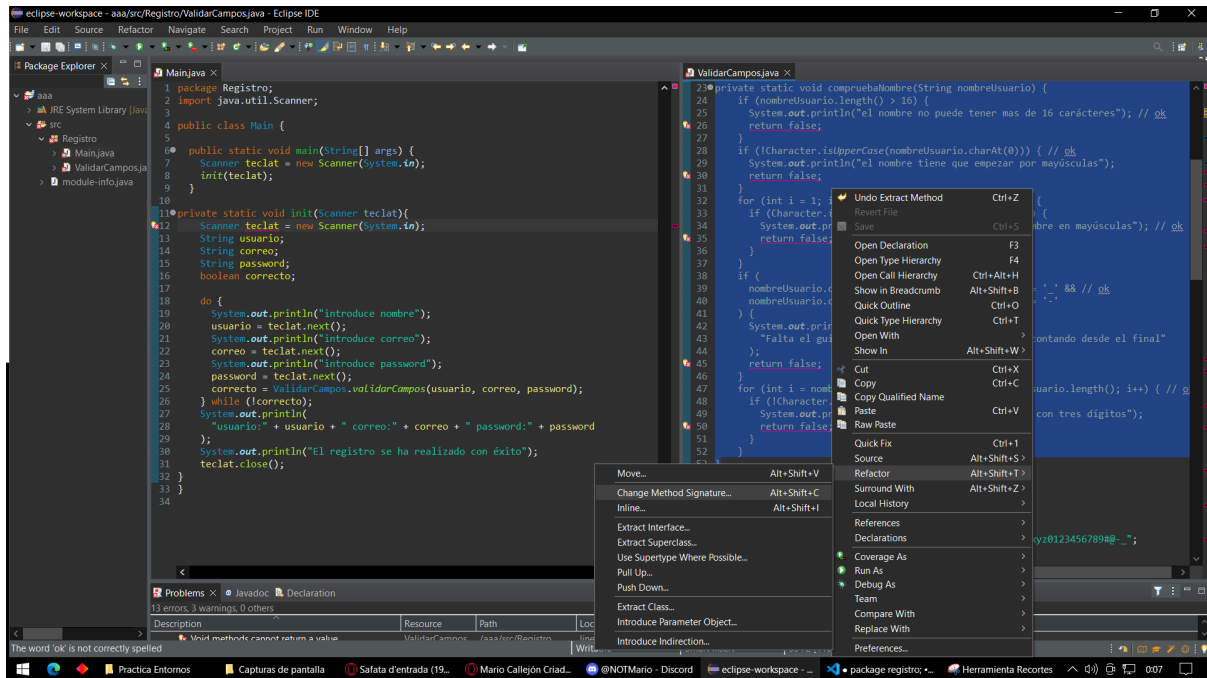
Para poder realizar esta parte del ejercicio, lo que tenemos que hacer es coger todo lo que tenemos dentro del main y darle a refactor como podemos ver en la captura de pantalla y darle a extract method y le daremos el nombre de init +



```
1 package Registro;
2 import java.util.Scanner;
3
4 public class Main {
5
6     public static void main(String[] args) {
7         Scanner teclado = new Scanner(System.in);
8         init(teclado);
9     }
10
11     private static void init(Scanner teclado){
12         Scanner teclado = new Scanner(System.in);
13         String usuario;
14         String correo;
15         String password;
16         boolean correcto;
17
18         do {
19             System.out.println("introduce nombre");
20             usuario = teclado.next();
21             System.out.println("introduce correo");
22             correo = teclado.next();
23             System.out.println("introduce password");
24             password = teclado.next();
25             correcto = ValidarCampos.validarCampos(usuario, correo, password);
26         } while (!correcto);
27         System.out.println(
28             "usuario:" + usuario + " correo:" + correo + " password:" + password
29         );
30         System.out.println("El registro se ha realizado con éxito");
31         teclado.close();
32     }
33 }
```

Añade un nuevo parámetro al método compruebaNombre, de nombre “usuarios” y de tipo array de Strings y cambia el orden de los parámetros de los métodos si tienen más de uno.

Para poder realizar este ejercicio, lo que tenemos que hacer es ir al método compruebaNombre que lo hemos refactorizado antes y cogerlo todo, darle click derecho , refactor y change method signature, dentro de ahí le damos a add, y ponemos String[] y en el name le ponemos usuarios y le damos a ok y quedará esto



```

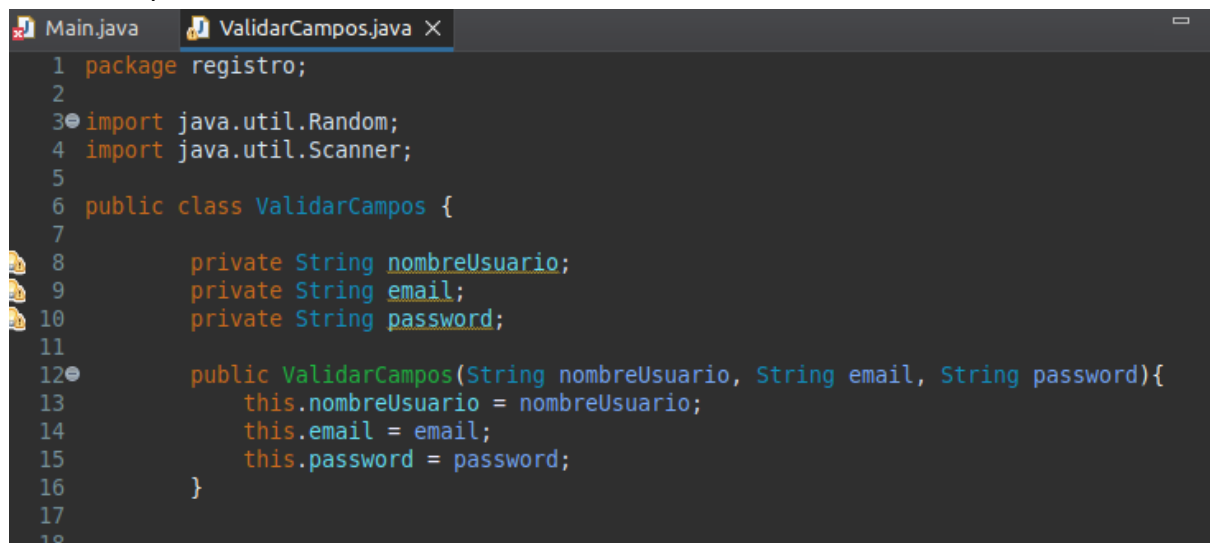
private static void compruebaNombre(String nombreUsuario, String[] usuarios) {
    if (nombreUsuario.length() > 16) {
        System.out.println("el nombre no puede tener mas de 16 caracteres"); // ok
        return false;
    }
    if (!Character.isUpperCase(nombreUsuario.charAt(0))) { // ok
        System.out.println("el nombre tiene que empezar por mayúsculas");
        return false;
    }
    for (int i = 1; i < nombreUsuario.length() - 4; i++) {
        if (Character.isUpperCase(nombreUsuario.charAt(i))) {
            System.out.println("solo la primera letra del nombre en mayúsculas"); // ok
            return false;
        }
    }
    if (
        nombreUsuario.charAt(nombreUsuario.length() - 4) != '_' && // ok
        nombreUsuario.charAt(nombreUsuario.length() - 4) != '-'
    ) {
        System.out.println(
            "Falta el guión o no esta en la cuarta posicion contando desde el final"
        );
        return false;
    }
    for (int i = nombreUsuario.length() - 3; i < nombreUsuario.length(); i++) { // ok
        if (!Character.isDigit(nombreUsuario.charAt(i))) {
            System.out.println("El nombre tiene que terminar con tres dígitos");
            return false;
        }
    }
}

```

Convierte las variables locales en atributos.

Crea varios constructores para la clase validarCampos.

Este apartado lo he hecho manualmente debido a que no me salía la opción en eclipse para hacer esta parte.

The image shows a screenshot of the Eclipse IDE with two tabs: 'Main.java' and 'ValidarCampos.java'. The 'ValidarCampos.java' tab is active, displaying the following Java code:

```
1 package registro;
2
3 import java.util.Random;
4 import java.util.Scanner;
5
6 public class ValidarCampos {
7
8     private String nombreUsuario;
9     private String email;
10    private String password;
11
12    public ValidarCampos(String nombreUsuario, String email, String password){
13        this.nombreUsuario = nombreUsuario;
14        this.email = email;
15        this.password = password;
16    }
17
18 }
```

Encapsula los atributos de la clase validarCampos (Getters i Setters).

Para realizar este ejercicio, lo he hecho con el visual que es el único IDE que me ha dejado hacer los getters y setters de una manera no manual, he dado click derecho y me ha salido la opción de generar getters y setters,

```
6 public class ValidarCampos {  
7  
8     private String nombreUsuario;  
9     private String email;  
10    private String password;  
11  
12    public ValidarCampos(String nombreUsuario, String email, String password) {  
13        this.nombreUsuario = nombreUsuario;  
14        this.email = email;  
15        this.password = password;  
16    }  
17  
18    public String getNombreUsuario() {  
19        return this.nombreUsuario;  
20    }  
21  
22    public void setNombreUsuario(String nombreUsuario) {  
23        this.nombreUsuario = nombreUsuario;  
24    }  
25  
26    public String getEmail() {  
27        return this.email;  
28    }  
29  
30    public void setEmail(String email) {  
31        this.email = email;  
32    }  
33  
34    public String getPassword() {  
35        return this.password;  
36    }  
37  
38    public void setPassword(String password) {  
39        this.password = password;  
40    }  
41  
42 }
```

# Comentarios JavaDoc

Main:

```
/**
```

Esta clase es la clase principal del proyecto, se encarga de realizar la validación de los campos introducidos por el usuario mediante la utilización de la clase ValidarCampos.

@author Mario

@version 1.0

@since [Inserta la versión del proyecto aquí]

```
*/
```

```
public class Main {
```

```
/**
```

Este método se encarga de inicializar el proceso de validación de los campos del usuario.

@param validar el objeto ValidarCampos utilizado para la validación.

```
*/
```

```
private static void init(ValidarCampos validar) {  
    boolean correcto;  
    do {  
        Scanner teclat = new Scanner(System.in);  
        System.out.println("Escribe un nombre");  
        validar.setNombreUsuario(teclat.nextLine());  
        correcto = validar.compruebaNombre(validar.getNombreUsuario(), null);  
        System.out.println("Escribe un email");  
        validar.setEmail(teclat.nextLine());  
        correcto = validar.compruebaEmail(validar.getEmail());  
        System.out.println("Escribe una contraseña");  
        validar.setPassword(teclat.nextLine());  
        correcto = validar.compruebaContraseña(validar.getPassword());  
        correcto = validar.generarCodigo();  
        System.out.println("Has entrado con éxito");  
        teclat.close();  
    } while (!correcto);  
}  
/**
```

Este método es el punto de entrada del programa. Se encarga de crear un objeto ValidarCampos y de llamar al método init para realizar la validación de los campos del usuario.

@param args los argumentos de la línea de comandos (no se utilizan en este programa).

\*/

```
public static void main(String[] args) {  
    ValidarCampos validar = new ValidarCampos(null, null, null);  
    init(validar);  
}  
}
```

/\*\*

\* Constructor que inicializa los campos de nombre de usuario, email y contraseña.

\* @param Mario.

\*/

```
public ValidarCampos(String nombreUsuario, String email, String password) {  
    this.nombreUsuario = nombreUsuario;  
    this.email = email;  
    this.password = password;  
}
```

/\*\*

\* Método que devuelve el nombre de usuario.

\* @return El nombre de usuario.

\*/

```
public String getNombreUsuario() {  
    return this.nombreUsuario;  
}
```

/\*\*

\* Método que establece el nombre de usuario.

\* @param nombreUsuario El nuevo nombre de usuario.

\*/

```
public void setNombreUsuario(String nombreUsuario) {  
    this.nombreUsuario = nombreUsuario;  
}
```

/\*\*

\* Método que devuelve el email.

\* @return El email.

\*/

```
public String getEmail() {  
    return this.email;  
}
```

/\*\*

\* Método que establece el email.

\* @param email El nuevo email.

\*/

```
public void setEmail(String email) {
```

```

        this.email = email;
    }

    /**
     * Método que devuelve la contraseña.
     * @return La contraseña.
     */
    public String getPassword() {
        return this.password;
    }

    /**
     * Método que establece la contraseña.
     * @param password La nueva contraseña.
     */
    public void setPassword(String password) {
        this.password = password;
    }

    /**
     * Método que genera un código de seguridad aleatorio y lo muestra por pantalla,
     * y luego solicita al usuario que introduzca el código generado para comprobar si es
     * correcto.
     * @return true si el código introducido por el usuario es correcto, false si no lo es.
     */
    public boolean generarCodigo() {
        StringBuilder codigo = new StringBuilder();
        Random random = new Random();
        String caracteres =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789#@-_"
;
        for (int i = 0; i < 8; i++) {
            int index = random.nextInt(caracteres.length());
            codigo.append(caracteres.charAt(index));
        }

        // Mostrar código de seguridad por pantalla
        System.out.println("El código de seguridad es: " + codigo);

        // Pedir al usuario que escriba el código de seguridad
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce el código de seguridad: ");
        String codigoUsuario = scanner.nextLine();
        scanner.close();

        // Comprobar si el código de seguridad introducido es correcto
        if (codigo.toString().equals(codigoUsuario)) {
            return true;
        } else {

```



```

        System.out.println("error");
        return false;
    }
}

/**
 * Método que comprueba si la contraseña cumple con las condiciones establecidas.
 * @param password La contraseña a comprobar.
 * @return true si la contraseña cumple con las condiciones, false si no las cumple.
 */

/**
public boolean compruebaContraseña(String password) {
    if (password.length() != 8) {
        System.out.println("Introduce una contraseña de 8 caracteres.");
        return false;
    }
    if (!Character.isUpperCase(password.charAt(0))) {
        System.out.println("La contraseña debe empezar con mayúsculas.");
        return false;
    }
    if (!(password.contains("@") || password.contains("-") ||
        password.contains("#") || password.contains("'"))) {
        System.out.println("Introduce uno de los siguientes símbolos @,#,-,'");
        return false;
    }
    if (!Character.isDigit(password.charAt(password.length() - 1))
        || !Character.isDigit(password.charAt(password.length() - 2))) {
        System.out.println("La contraseña debe terminar en dos dígitos.");
        return false;
    }
    return true;
}
/**

```

Verifica si el email ingresado cumple con las condiciones establecidas.

@param email El email que se desea verificar.

```

@return true si el email es válido, false si no cumple con los requisitos.
*/
public boolean compruebaEmail(String email) {
    String[] dominiosPermitidos = { "paucasesnovescifp", "yahoo", "gmail",
    "hotmail" };
    boolean dominioValido = false;
    if (!email.contains("@")) {
        System.out.println("Falta el símbolo @ en el email.");
        return false;
    }
}

```

```

}

for (String dominio : dominiosPermitidos) {
    if (email.contains("@" + dominio)) {
        dominioValido = true;
        break;
    }
}

if (!dominioValido) {
    System.out.println("El dominio del email no es válido.");
    return false;
}

if (!(email.endsWith(".com") || email.endsWith(".es") ||
    email.endsWith(".cat"))) {
    System.out.println("La extensión del email no es válida.");
    return false;
}

return true;
}

```

/\*\*

Verifica si el nombre de usuario ingresado cumple con las condiciones establecidas.  
 @param nombreUsuario El nombre de usuario que se desea verificar.  
 @param usuarios Un arreglo con los nombres de usuario ya registrados en el sistema.  
 @return true si el nombre de usuario es válido, false si no cumple con los requisitos.  
 \*/

```

public boolean compruebaNombre(String nombreUsuario, String[] usuarios) {
    if (nombreUsuario.length() > 16) {
        System.out.println("El nombre de usuario no puede tener más de 16 caracteres.");
        return false;
    }
    if (!Character.isUpperCase(nombreUsuario.charAt(0))) {
        System.out.println("El nombre de usuario debe empezar con mayúsculas.");
        return false;
    }
    for (int i = 1; i < nombreUsuario.length() - 4; i++) {
        if (Character.isUpperCase(nombreUsuario.charAt(i))) {
            System.out.println("Solo la primera letra del nombre de usuario debe estar en mayúsculas.");
            return false;
        }
    }
    if (nombreUsuario.charAt(nombreUsuario.length

```