

Lucas Varela Negro Dam1

Tema 4: Optimización y Documentación

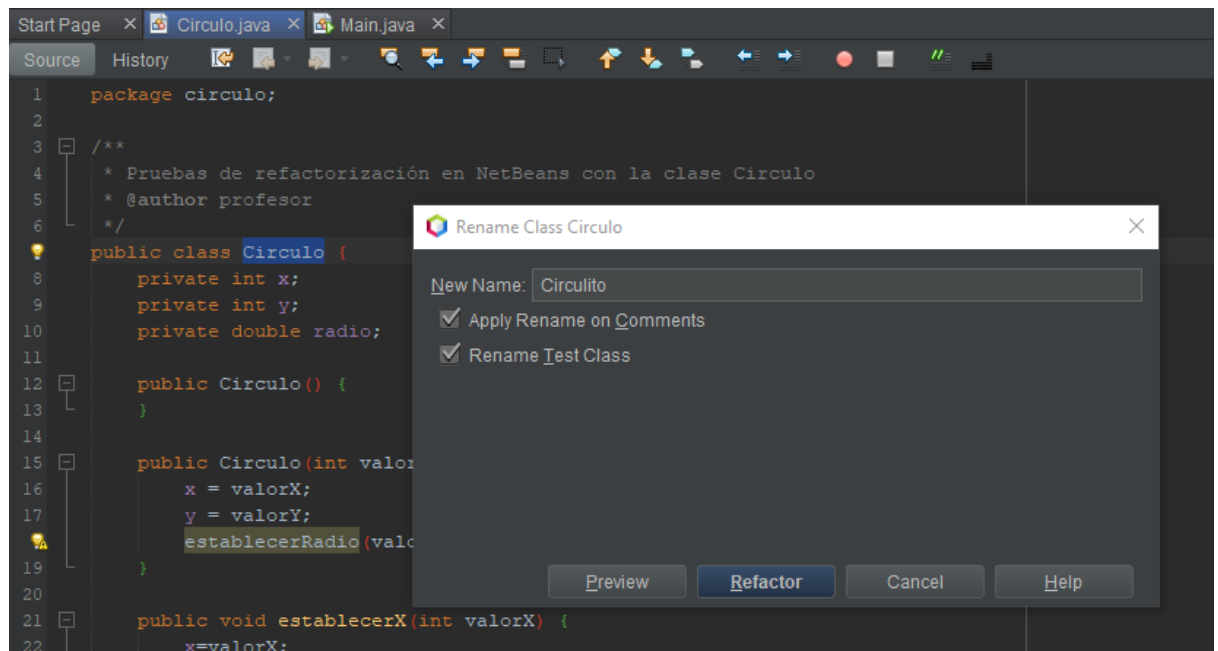
Tema 4: Optimización y Documentación	1
Práctica 4.1-Practicar reestructuración de código de Java en Netbeans:	3
Práctica 4.2:Busca analizadores estáticos de código en el mercado y compara sus características:	12
Práctica 4.3: Busca analizadores dinámicos de código en el mercado y compara sus características:	13
Práctica 4.4: Análisis de código Java en Netbeans:	13
Práctica 4.5: Operaciones básicas con Git:	13
Práctica 4.6: Gestión de Ramas en Git:	18
Práctica 4.7: Conflictos entre ramas:	23
Práctica 4.8: Más conflictos entre ramas:	25
Práctica 4.9: Operaciones con GitHub:	28
Práctica 4.10: Git con Netbeans:	30

Práctica 4.1-Practicar reestructuración de código de Java en Netbeans:

Proyecto Circulo:

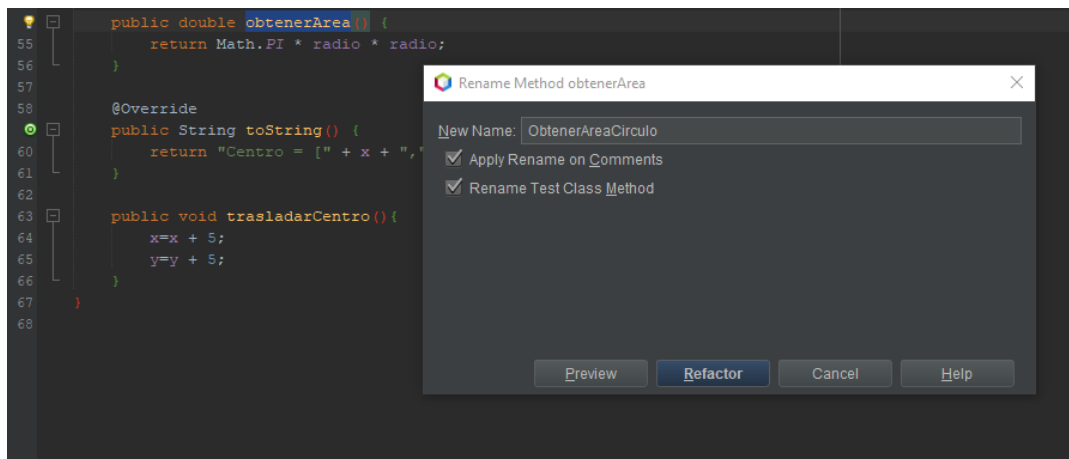
- **Renombrar la clase Circulo por Circulito.**

Seleccionamos la palabra `Círculo` de la clase, hacemos click derecho y en la opción *refactor* seleccionamos *rename*.

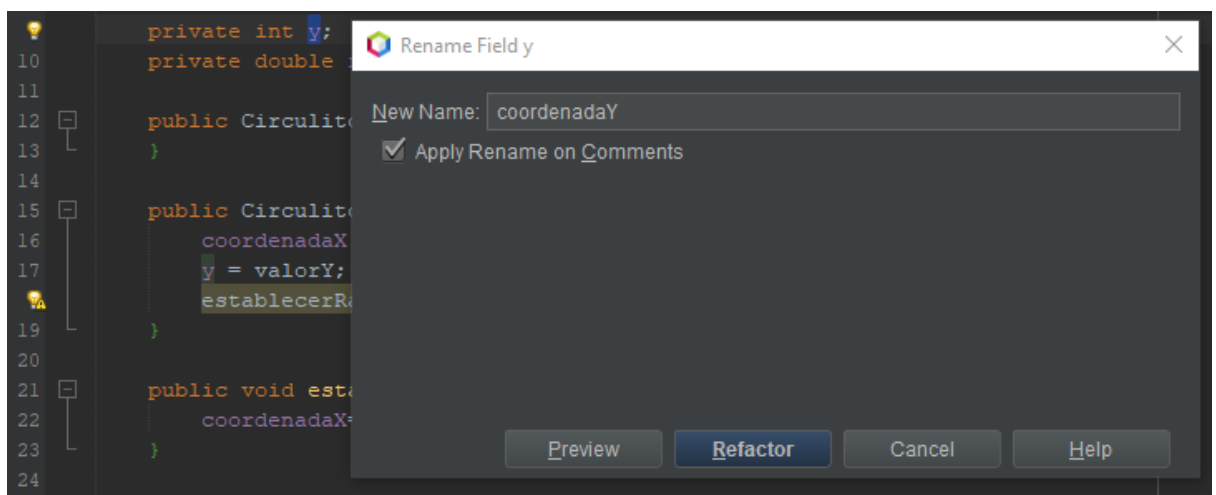
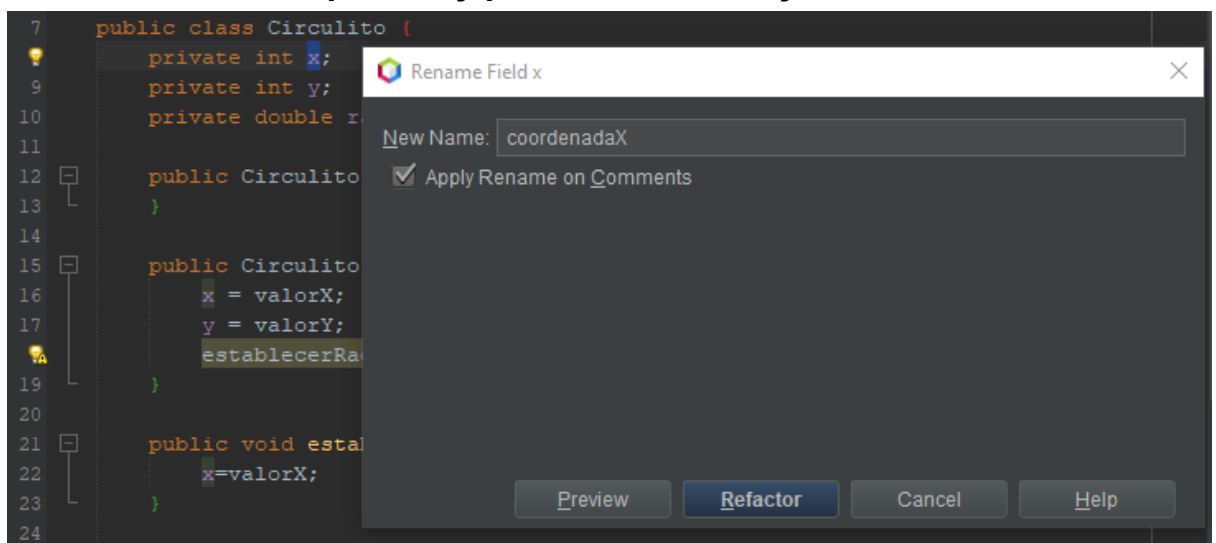


En la pestaña donde renombramos podemos seleccionar que también renombre la clase en los comentarios y en las clases de prueba, así como una opción de *Preview* que nos permite ver cómo se aplicarían los cambios.

- **Renombrar el método `ObtenerArea` por `ObtenerAreaCirculo`.**



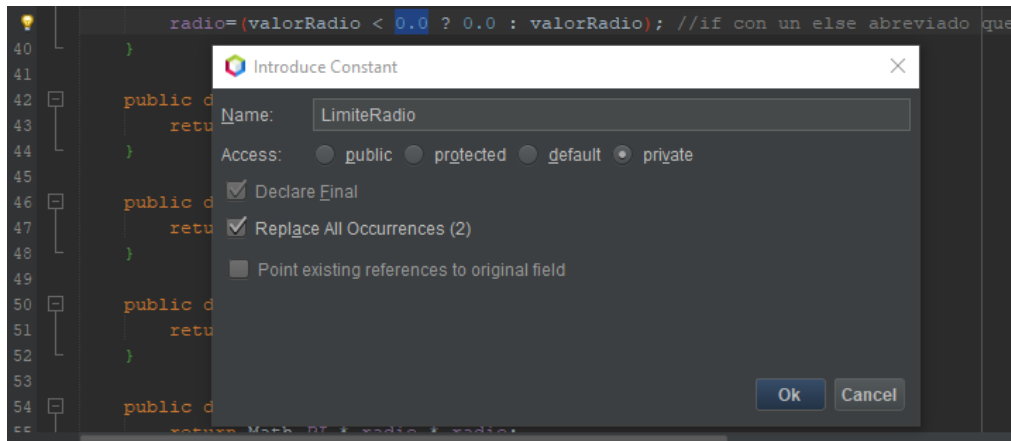
- Renombrar los campos x e y por coordenadaX y coordenadaY.



- Introducir constante Limiteradio de tipo double con el valor 0.0.

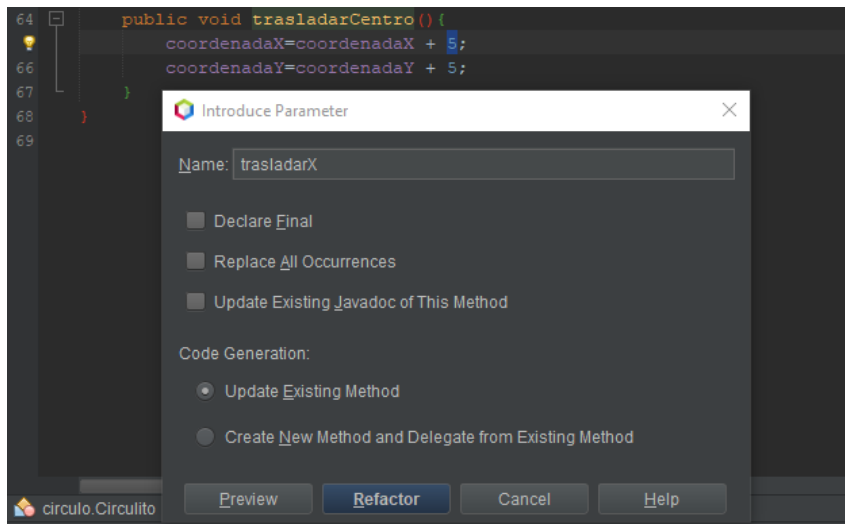
Nos posicionamos en la línea 39, donde usamos el valor 0.0 y seleccionamos el valor.

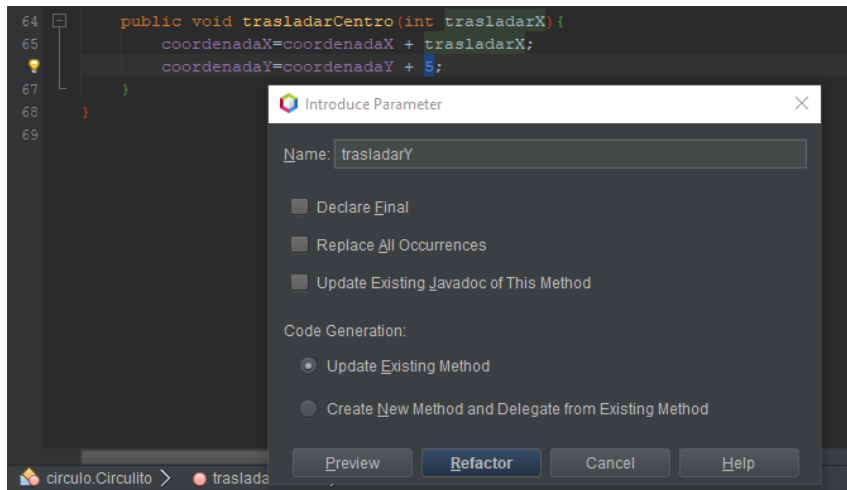
Hacemos click derecho, seleccionamos *refactor* y exactamente *introduce*, y *constant*. Establecemos el nombre y en este caso marcamos *Replace All Occurrences*.



- **Cambiar parámetros del método trasladarCentro para que tenga dos parámetros trasladarX y trasladarY de tipo int. Hacer los cambios necesarios para que el código del método permita añadir la coordenada x al valor de trasladarX y al añadir la coordenada y al valor de trasladarY.**

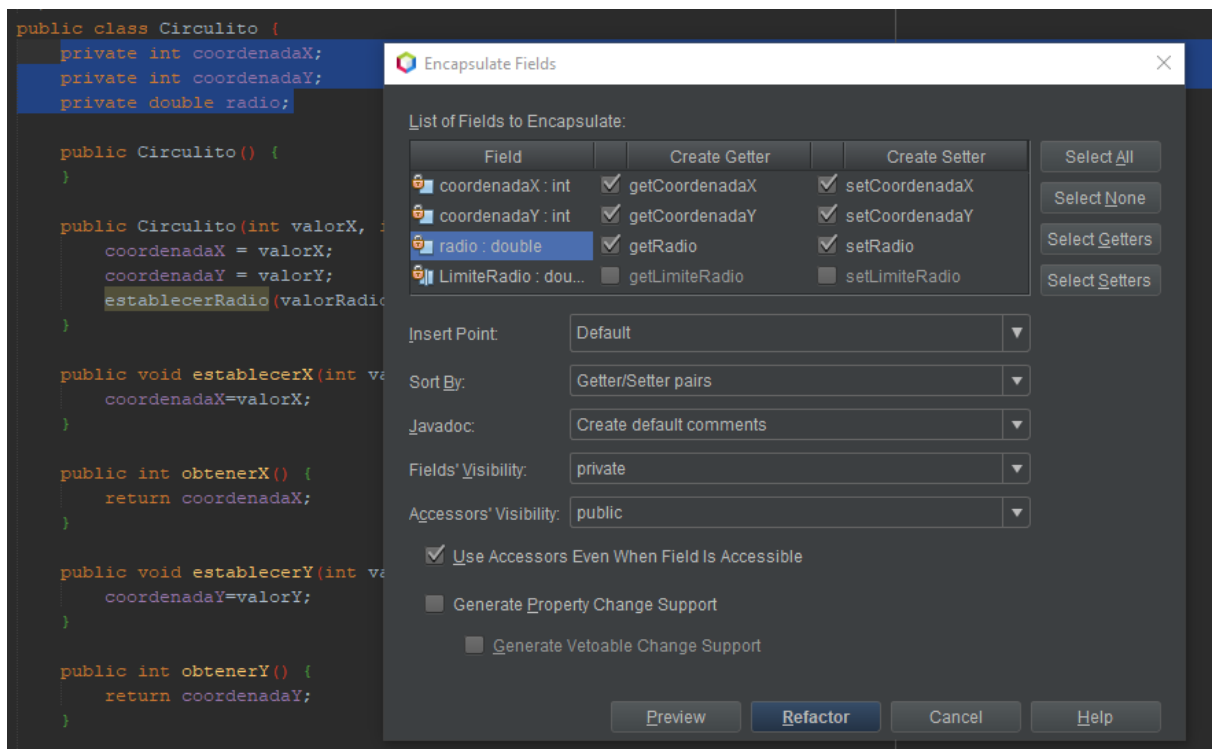
Nos posicionamos en la línea 64, seleccionamos el valor 5 de la coordenadaX, hacemos click derecho, seleccionamos *Refactor, Introduce, Parameters* y le ponemos el nombre trasladarX de tipo int, lo mismo para coordenadaY pero con el nombre trasladarY.





- **Encapsular los tres atributos de la clase: coordenadaX, coordenadaY, radio.**

Seleccionamos las tres variables, hacemos click derecho, y en *Refactor* seleccionamos *encapsulate fields* y sin cambiar nada le damos a *Refactor*.



- **Eliminar de forma segura los método obtenerX, obtenerY, obtenerRadio, establecerX, establecerY y establecerRadio que ahora no son necesarios haciendo los cambios necesarios en el código de la clase, de la clase Main y de las pruebas para que sean sustituidos por los métodos correspondientes tipo get y set creados.**

Seleccionamos los métodos indicados (Uno a uno para evitar errores), hacemos click derecho y en *Refactor* seleccionamos *Safely Delete* marcando la opción *search in comments*.

Ahora para que el código funcione correctamente debemos realizar los siguientes pasos:

- En la línea 38 de la clase Circulito Cambiar EstablecerRadio() por setRadio().
- En las líneas del 13 al 18 de la clase main debemos cambiar:
 - obtenerX() por getCoordenadaX().
 - obtenerY() por getCoordenadaY().
 - obtenerRadio() por getRadio().
 - establecerX() por setCoordenadaX().
 - establecerY() por setCoordenadaY().
 - establecerRadio() por setRadio().
- En las pruebas de JUnit debemos cambiar todos los obtenerX(), obtenerY(), establecerX(), establecerY(), obtenerRadio() y establecerRadio() por sus correspondientes getters y setters como en el clase main.
- **Al terminar comprueba que las pruebas siguen funcionando.**
Tras realizar los pasos anteriores podemos comprobar en las siguientes capturas que tanto los tests como el main siguen funcionando correctamente.

```
La coordenada X es 37
La coordenada Y es 43
El radio es 2.5
El diámetro es 8,40
La circunferencia es 26,39
El área es 55,42
-----
BUILD SUCCESS
-----
```

```

-----
T E S T S
-----
Running circulo.CirculitoTest
obtenerRadio
obtenerY
establecerY
obtenerX
establecerX
establecerRadio
obtenerCircunferencia
obtenerArea
trasladarCentro
obtenerDiametro
Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 sec

Results :

Tests run: 10, Failures: 0, Errors: 0, Skipped: 0

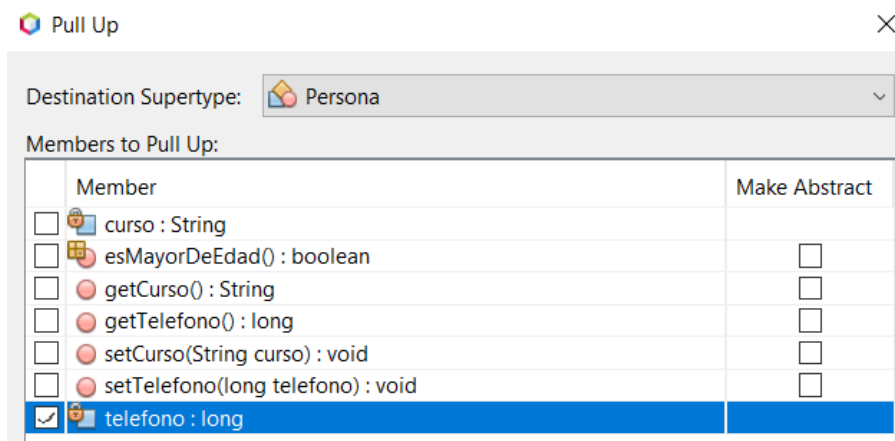
-----
BUILD SUCCESS

```

Proyecto Alumno:

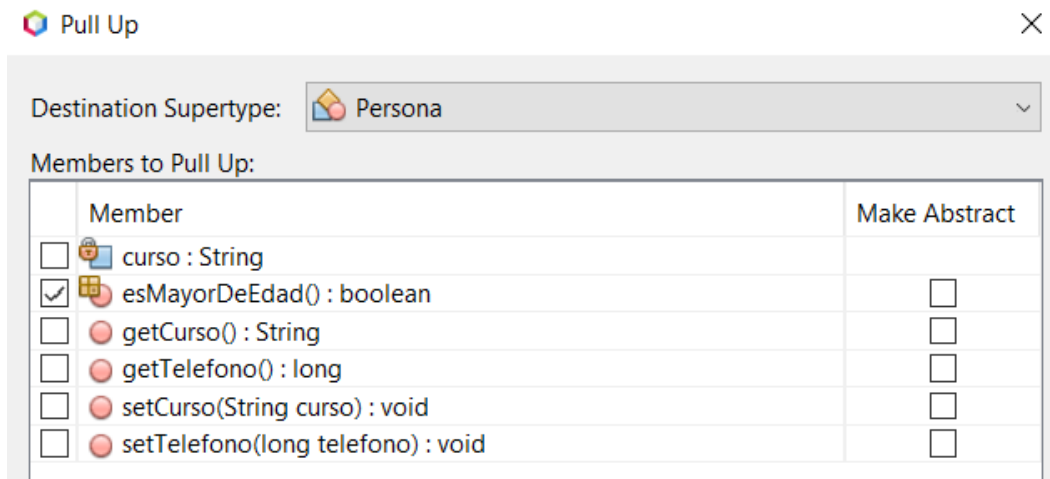
- **Mueve el atributo 'teléfono' de la clase Alumno a su superclase Persona.**

Seleccionamos el atributo teléfono, y en Refactor/Pull Up seleccionamos la clase persona marcando solo teléfono:



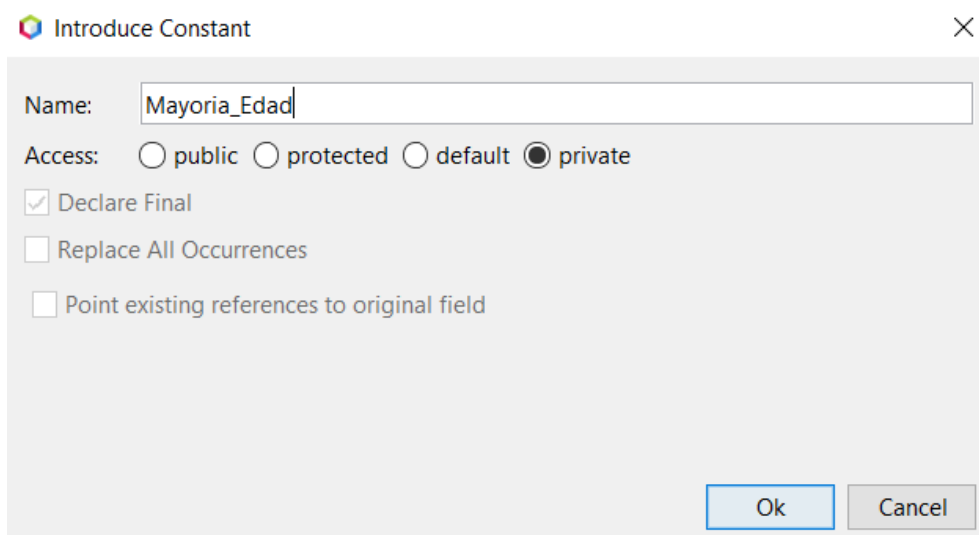
- **Mueve el método 'esMayorDeEdad' de la clase Alumno a su superclase Persona.**

Seleccionamos el método 'esMayordeEdad', le damos a Refactor/Pull Up y seleccionamos la superclase persona:



- En la clase 'Persona' substituir el número 18 por una constante estática.

Seleccionamos el 18 en la clase 'EsMayorEdad' recién subida a 'Persona', y Refactor/Introducir../Constante y le damos un nombre:



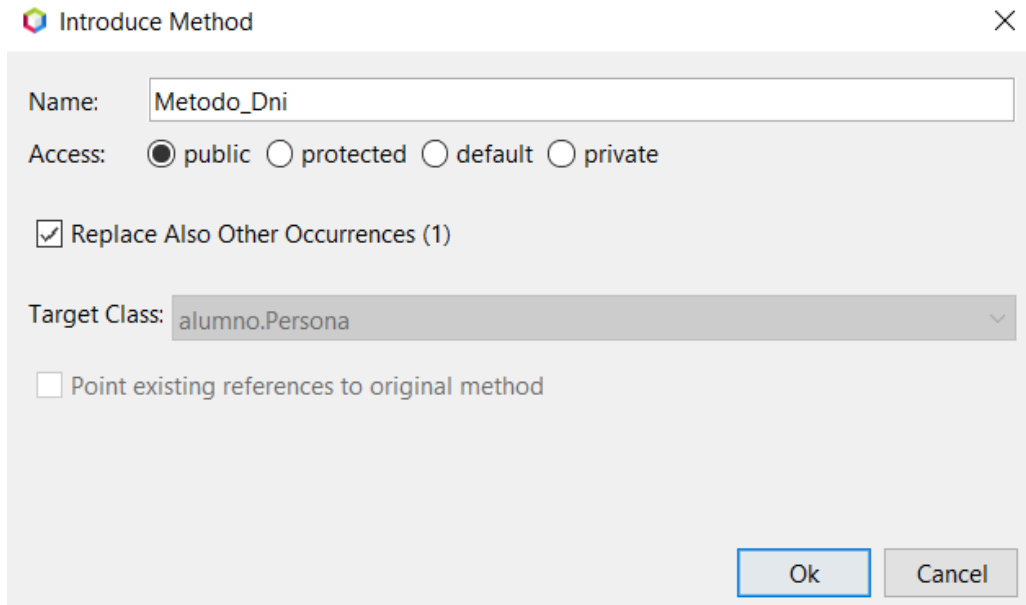
- En los métodos comprobarDNI y asignarLetraDNI se repite el código:

```
char letraCorrecta = ' ';
try {
    num = Integer.parseInt(numero);
} catch (NumberFormatException numberFormatException)
{ numeroValido = false;}

String letras = "TRWAGMYFPDXBNJZSQVHLCKE";
if (numeroValido)
    letraCorrecta= letras.charAt(num%23);
```

Crea un método privado con ese código que sea invocado desde los otros dos métodos.

Seleccionamos el código y vamos a Refactor/Introducir../Método, le damos nombre y hacemos los cambios necesarios en el código.



Introduce Method

Name:

Access: ☒ public ☐ protected ☐ default ☐ private

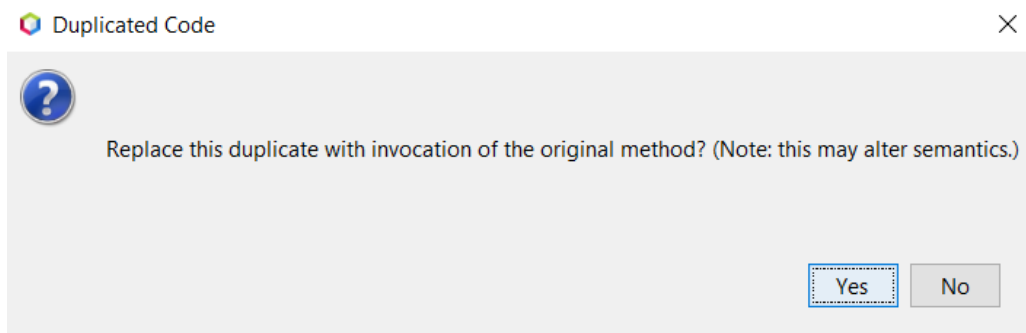
☒ Replace Also Other Occurrences (1)

Target Class:

☐ Point existing references to original method

Ok Cancel

Nos saldrá este mensaje indicando si queremos eliminar la duplicidad de código y le daremos a Sí/Yes:



Duplicated Code

Replace this duplicate with invocation of the original method? (Note: this may alter semantics.)

Yes No

- **Al terminar, comprueba que el proyecto y los tests siguen funcionando correctamente.**

<pre> Lista inicial: [pepe, dni=12345678Z, fecNac=2003-06-22 , ana, dni=87654321X, fecNac=1999-01-11 , luis, dni=00000000T, fecNac=2002-02-28] Lista ordenada por edad: [ana, dni=87654321X, fecNac=1999-01-11 , luis, dni=00000000T, fecNac=2002-02-28 , pepe, dni=12345678Z, fecNac=2003-06-22] </pre>	<pre> ----- T E S T S ----- Running alumno.PersonaTest asignarLetraDNI comprobarDNI Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.021 sec Results : Tests run: 2, Failures: 0, Errors: 0, Skipped: 0 ----- BUILD SUCCESS </pre>
--	--

TESTS

Running alumno.AlumnoTest

esMayorDeEdad

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 sec

Results :

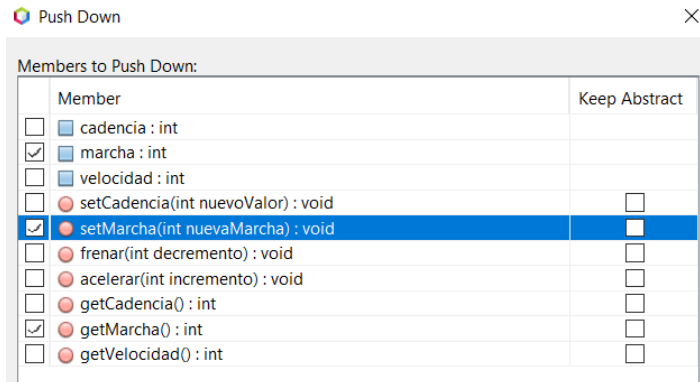
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

BUILD SUCCESS

Proyecto MTB:

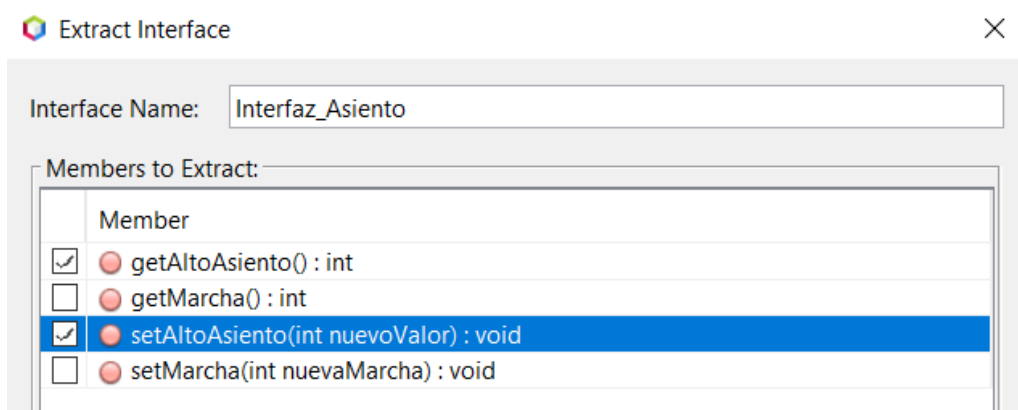
- Mueve el atributo marcha, el método getMarcha y el método setMarcha desde la superclase Bicicleta a la subclase MTB con la opción de menú Reestructurar/Descender probablemente quede algún problema que solucionar y no se haga todo automáticamente, soluciona y explica dicho problema.

Vamos a Refactor/Pull Down y seleccionamos los indicados para bajar a la subclase MTB y solucionamos problemas de código:

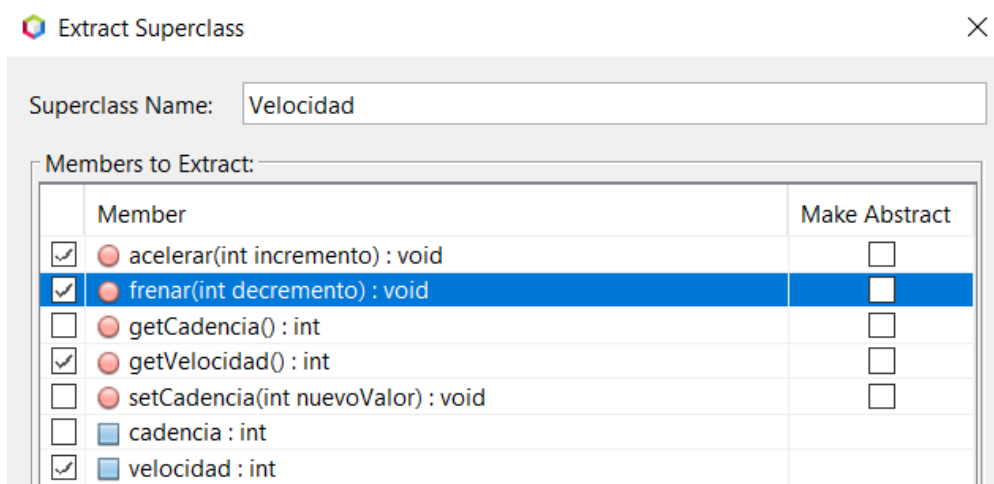


- Extrae una interfaz para los métodos getAltoAsiento y setAltoAsiento, usando la opción de menú Reestructurar/Extraer Interface.

Vamos a Refactor/Extract interface y seleccionamos los indicados:



- **Extrae una superclase con el campo velocidad y los métodos getVelocidad, acelerar y frenar usando la opción de menú Reestructurar/Extraer Superclase.**
Seleccionamos Refactor/Extract Superclass y seleccionamos las indicadas:



Práctica 4.2: Busca analizadores estáticos de código en el mercado y compara sus características:

Práctica 4.3: Busca analizadores dinámicos de código en el mercado y compara sus características:

Práctica 4.4: Análisis de código Java en Netbeans:

- Importa en Netbeans el proyecto RendimientoColecciones:
- Ejecuta el *profile* del mismo, configurando la sesión seleccionando "Métodos". Haz una captura de pantalla y comenta la diferencia de uso de CPU de los distintos métodos empleados.

- Toma una *instantánea* de análisis y guárdala.
- Ejecuta el *profile* opción *Telemetría* y haz una captura de pantalla.
- Sustituye el ArrayList por LinkedList y repite todos los pasos anteriores.
- Haz una comparativa del rendimiento de LinkedList vs. ArrayList.
- Haz una captura de la ventana SnapSHots en la que se vea la lista de snapshots hechos.

Práctica 4.5: Operaciones básicas con Git:

a) Crea una carpeta (y repositorio git) llamado *myweb*:

Nos situamos en la carpeta recién creada *myweb* y utilizamos el comando *git init*:

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/myweb
$ git init
Initialized empty Git repository in C:/Users/lucas.varelanegro/myweb/.git/

lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/myweb (master)
$
```

b) Incorpora a esa carpeta un archivo *index.html*:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Hola mundo</h1>
  </body>
</html>
```

y un archivo *style.css*:

```
body { background-color: pink;}
h1 {color: blue;}
```

Comprueba el estado de git.

Añadimos los archivos *index.html* y *style.css* con su respectivo contenido y para comprobar el estado de git utilizamos el comando *git.status*:

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/myweb (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
```

- c) Pasa los archivos a *staged*, comprueba el estado, pasalos al commit y vuelve a comprobar el estado y también el log. Supón que aquí terminaste tu web, y ya está enviada al cliente.

>>git add .

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git add .
```

>>git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css
```

>>git commit -m "Primer commit"

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git commit -m "Primer commit"
[master (root-commit) 60dc220] Primer commit
 2 files changed, 10 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
```

>>git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
nothing to commit, working tree clean
```

>>git log --oneline

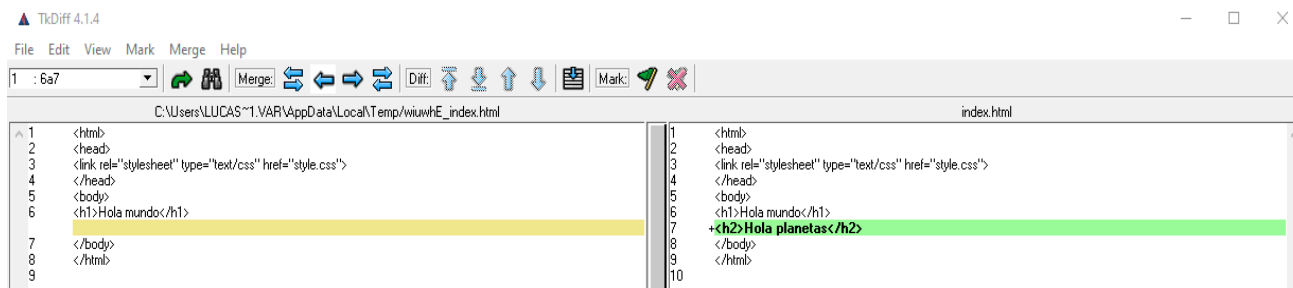
```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --oneline
60dc220 (HEAD -> master) Primer commit
```

- d) Ahora te pide unos cambios: modifica ligeramente el archivo *index.html* en la versión y ejecuta `tkdiff` para ver los cambios. Ahora te arrepientes de los cambios y aun no hiciste ningun add ni commit de esos cambios. Cómo recuperas el *index.html* que estaba en tu repositorio (en la zona de commit)?

>> `git difftool index.html`

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git difftool index.html

Viewing (1/1): 'index.html'
Launch 'tkdiff' [Y/n]? Y
```



```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git checkout index.html
Updated 1 path from the index
```

- e) Haz de nuevo algunos cambios en el *index.html*, y esta vez, si lleva esos cambios al repositorio y consulta el log.

Lo editamos y confirmamos que está modificado:

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

>> `git add .`

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git add .
```

>> `git commit -m "Segundo commit"`

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git commit -m "Segundo commit"
[master e69708a] Segundo commit
1 file changed, 1 insertion(+)
```

>> `git log --oneline`

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --oneline
e69708a (HEAD -> master) Segundo commit
60dc220 Primer commit
```

- f) A tu cliente no le gustan los cambios así que te pide que vuelvas la web a la situación anterior (toda la web, no lo hagas viendo que archivos modificaste). **X**
- g) Borra el archivo style.css (borrando por fuera de git, no con: *git rm*) En qué estado queda el repositorio? Vuelve a la situación anterior.

Lo borramos desde la carpeta y lo comprobamos con >>git status:

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    style.css

no changes added to commit (use "git add" and/or "git commit -a")
```

Para recuperarlo >>git restore style.css

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git restore style.css
```

>>git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- h) En qué estado queda el repositorio? Vuelve a la situación anterior.

>> rm style.css

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ rm style.css
```

>> git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    style.css

no changes added to commit (use "git add" and/or "git commit -a")
```

>> git restore --staged style.css

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git restore --staged style.css
```

>> git restore style.css

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git restore style.css
```


- i) **Modifica el archivo `style.css`. En qué estado queda el repositorio?**
Vuelve a la situación anterior.

Modificamos el archivo y comprobamos que esté modificado:

>> git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   style.css

no changes added to commit (use "git add" and/or "git commit -a")
```

>> git restore style.css

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git restore style.css
```

- j) **Ahora queremos probar la opción `-a` de `git commit`, para hacer commit sin el `add` previo (recuerda que solo funciona para los archivos modificados, no nuevos, no funcionaria en el punto 'c'))**

Primero debemos modificar un archivo para poder realizar el commit, en este caso modificaremos `index.html`:

>> git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

>> git commit -am "Tercer commit"

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git commit -am "Tercer commit"
[master 8b42b67] Tercer commit
1 file changed, 1 insertion(+)
```

- k) **Muestra el log con todos los cambios hechos en el ejercicio.**

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --oneline
8b42b67 (HEAD -> master) Tercer commit
e69708a Segundo commit
60dc220 Primer commit
```

Práctica 4.6: Gestión de Ramas en Git:

- a) Crea un repositorio similar al de la práctica 4.5 y haz commit de los dos archivos.

```
>> git init
>> git add .
>> git commit -m "Primer commit"
```

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git init
Initialized empty Git repository in C:/Users/lucas.varelanegro/Desktop/myweb/.git/

lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git add .

lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git commit -m "Primer commit"
[master (root-commit) 1d97685] Primer commit
2 files changed, 12 insertions(+)
create mode 100644 index.html
create mode 100644 style.css
```

- b) Crea una rama llamada “nuevaWeb” y cámbiate a ella.

```
>> git branch nuevaWeb
>> git checkout nuevaWeb
>> git branch (Para comprobar)
```

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git branch nuevaWeb

lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git checkout nuevaWeb
Switched to branch 'nuevaWeb'

lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git branch
  master
* nuevaWeb
```

- c) Borra en esa nueva rama style.css, crea style2.css y modifica index.html con nuevos textos utilizando el nuevo archivo css y muestra el estado:

html:

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="style2.css">
</head>
<body>
  <h1>Nova Web</h1>
</body>
</html>
```

Style2.css:

```
body { background-color: black;}
h1 {color: white;}
```

```
>> git rm style.css
```

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git rm style.css
rm 'style.css'
```

Modificamos el index.html y añadimos el style2.css:

>> git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git status
On branch nuevaWeb
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    style.css

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        style2.css
```

d) Asegúrate de que estás en la rama nuevaWeb y haz commit de los cambios.

>> git branch

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git branch
  master
* nuevaWeb
```

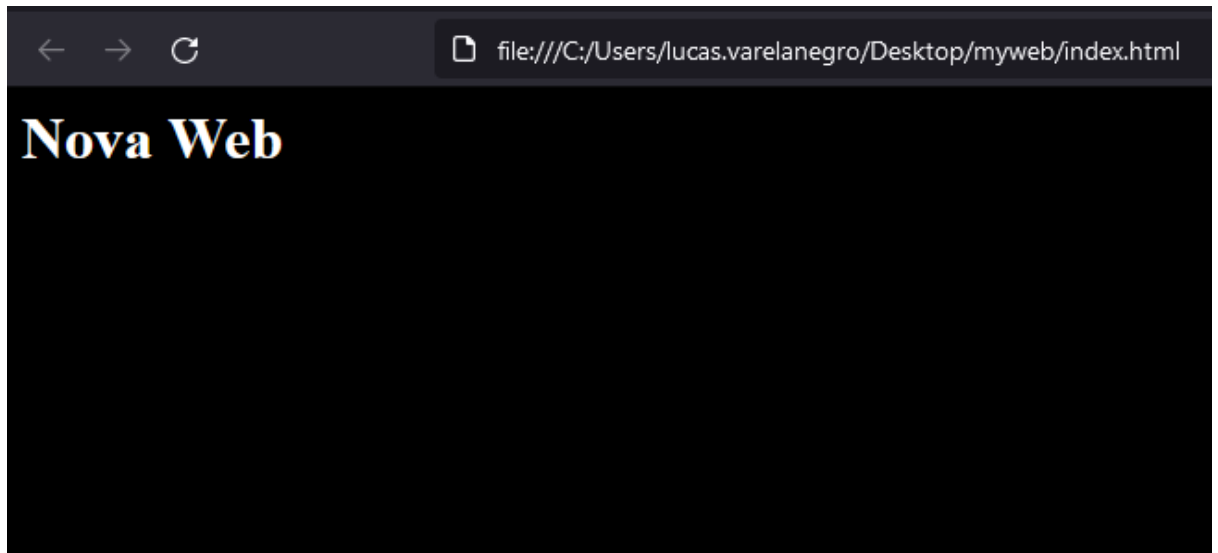
>>git add .

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git add .
```

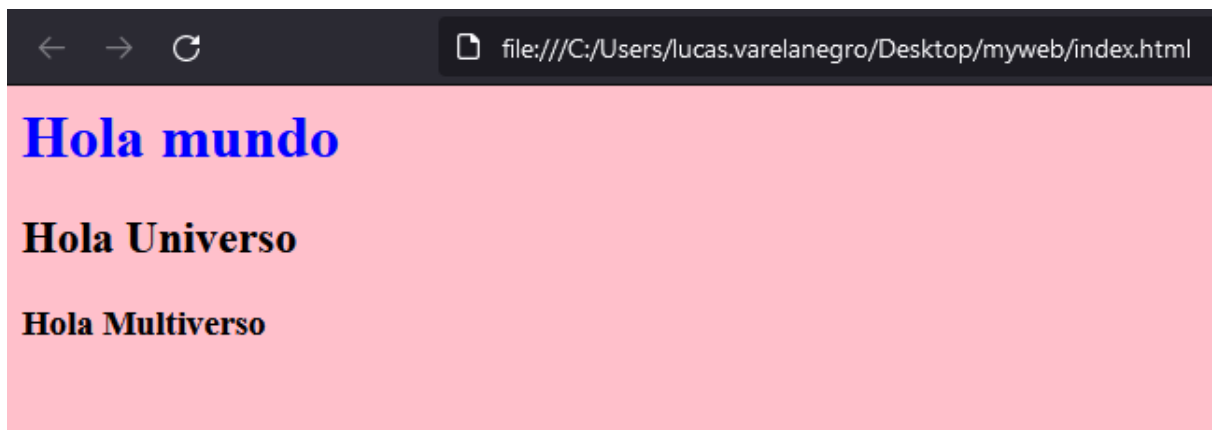
>> git commit -m "Segundo commit"

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git commit -m "Segundo commit"
[nuevaWeb 94b32bc] Segundo commit
3 files changed, 9 insertions(+), 11 deletions(-)
delete mode 100644 style.css
create mode 100644 style2.css
```

e) Muestra el contenido de la web de esta nueva rama en el navegador. Cambia a master y muestra el contenido de la web en el navegador.



>> git checkout master



f) Muestra el log con las opciones: oneline, all y graph (Muestra el último commit de cada rama)

En la rama master:

>> git log --oneline

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --oneline
1d97685 (HEAD -> master) Primer commit
```

>> git log --graph

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --graph
* commit 1d976859c816a74fdb0a3b31cca7a0ca2a4b76c (HEAD -> master)
  Author: Lucas <lucasvarelanegro@gmail.com>
  Date: Tue Mar 15 19:15:37 2022 +0100

    Primer commit
```

En la rama nuevaWeb:

>> git checkout nuevaWeb

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git checkout nuevaWeb
Switched to branch 'nuevaWeb'
```

>> git log --oneline

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git log --oneline
94b32bc (HEAD -> nuevaWeb) Segundo commit
1d97685 (master) Primer commit
```

>> git log --graph

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git log --graph
* commit 94b32bc3c268979185517efa5471e8a77fba5bc9 (HEAD -> nuevaWeb)
  Author: Lucas <lucasvarelanegro@gmail.com>
  Date: Tue Mar 15 19:30:52 2022 +0100

    Segundo commit

* commit 1d976859c816a74fdb0a3b31cca7a0ca2a4b76c (master)
  Author: Lucas <lucasvarelanegro@gmail.com>
  Date: Tue Mar 15 19:15:37 2022 +0100

    Primer commit
```

En las dos simultáneamente:

>> git log --all

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --all
commit 94b32bc3c268979185517efa5471e8a77fba5bc9 (nuevaWeb)
Author: Lucas <lucasvarelanegro@gmail.com>
Date: Tue Mar 15 19:30:52 2022 +0100

    Segundo commit

commit 1d976859c816a74fdb0a3b31cca7a0ca2a4b76c (HEAD -> master)
Author: Lucas <lucasvarelanegro@gmail.com>
Date: Tue Mar 15 19:15:37 2022 +0100

    Primer commit
```

g) Fusiona la rama creada con la rama principal y muestra en el navegador la rama principal.gi.

Primero debemos de estar en la rama master >> git checkout master

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (nuevaWeb)
$ git checkout master
Switched to branch 'master'
```

>> git merge nuevaWeb

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git merge nuevaWeb
Updating 1d97685..94b32bc
Fast-forward
 index.html | 16 ++++++-----
  style.css | 2 --
  style2.css | 2 ++
 3 files changed, 9 insertions(+), 11 deletions(-)
 delete mode 100644 style.css
 create mode 100644 style2.css
```

h) Muestra el estado y el log.

>> git status

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git status
On branch master
nothing to commit, working tree clean
```

>> git log --oneline

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git log --oneline
94b32bc (HEAD -> master, nuevaWeb) Segundo commit
1d97685 Primer commit
```

i) Elimina la rama creada.

>> git branch -D nuevaWeb

```
lucas.varelanegro@WIRTZWI-9QMLEGE MINGW64 ~/Desktop/myweb (master)
$ git branch -D nuevaWeb
Deleted branch nuevaWeb (was 94b32bc).
```

Práctica 4.7: Conflictos entre ramas:

a) Crea un repositorio igual al de la práctica 4.5 con los archivos index.html y style.css.

```
MINGW64;C:/Users/lucas/Desktop/myweb
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb
$ git init
Initialized empty Git repository in C:/Users/lucas/Desktop/myweb/.git/

lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)
$ git add .

lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)
$ git commit -m 'Primer commit'
[master (root-commit) 5e25cb4] Primer commit
2 files changed, 10 insertions(+)
create mode 100644 index.html
create mode 100644 style.css

lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)
$
```

- b) Crea una rama llamada cambios01a partir de la rama principal. En esa nueva rama, crea el archivo style2.css similar al de la práctica 4.6, elimina el style.css y haz que index.html use la nueva hoja de estilos.

```
>> git branch cambios01a  
>> git checkout cambios01a
```

Y realizamos los cambios:

```
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)  
$ git branch cambios01a  
  
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)  
$ git checkout cambios01a  
Switched to branch 'cambios01a'
```

- c) Crea una rama llamada cambios02 a partir de la rama principal y cambia el contenido de la etiqueta <h1>.

```
>> git branch cambios02
```

Y realizamos los cambios:

```
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)  
$ git branch cambios02
```

- d) Muestra el log con las opciones all y oneline.

```
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)  
$ git log --all  
commit 5e25cb425bd8f9f9089d67f409a712a1b1cf206d (HEAD -> master, cambios02, cambios01a)  
Author: Lucas <lucasvarelanegro@gmail.com>  
Date: Sun Mar 20 18:11:05 2022 +0100  
  
Primer commit  
  
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)  
$ git log --oneline  
5e25cb4 (HEAD -> master, cambios02, cambios01a) Primer commit  
  
lucas@DESKTOP-082FM4B MINGW64 ~/Desktop/myweb (master)  
$
```

- e) Muestra en el navegador el contenido de la web en los tres casos, es decir, cambiándo de Master a cambios01 a cambios02.



Hola mundo

Buenas Noches

Hola mundo

f) Fusiona Master y cambios01.

>> Git merge master cambios01a

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git merge master cambios01a
```

g) Fusiona Master y cambios02. ¿Algún conflicto? Explica por que. En caso afirmativo comprueba las líneas en conflicto y arregla la situación.

>> git merge master cambios02:

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git merge master cambios02a
Updating 56d12ae..cc03ef3
```

h) Borra las ramas que ya no son necesarias.

>> Git branch -D

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git branch -D cambios01a
Deleted branch cambios01a (was 56d12ae).

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git branch -D cambios02a
Deleted branch cambios02a (was cc03ef3).

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ .....
```


Práctica 4.8: Más conflictos entre ramas:

- a) Crea un repositorio igual al de la práctica 4.5.

>> git init

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb
$ git init
Initialized empty Git repository in C:/Users/Usuario/Desktop/myweb/.git/
```

- b) Crea una rama llamada cambios01 a partir de la rama principal. En esa nueva rama, crea el archivo style2.css similar al de la práctica 4.6, elimina el style.css y haz que index.html use la nueva hoja de estilos. Modifica el contenido de la etiqueta <h1>. Haz add/commit de los cambios.

>> Git branch cambios01

>> Git checkout cambios01

>> Git add .

>> Git commit -m "Commit cambios01"

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git branch cambios01

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git checkout cambios01
Switched to branch 'cambios01'

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (cambios01)
$ git add .

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (cambios01)
$ git commit -m "Commit cambios01"
[cambios01 ea7c4c5] Commit cambios01
3 files changed, 9 insertions(+), 8 deletions(-)
delete mode 100644 style.css
create mode 100644 style2.css
```

- c) Crea una rama llamada cambios02 a partir de la rama principal y cambia el contenido de la etiqueta <h1>.

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (cambios01)
$ git checkout master
Switched to branch 'master'

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git branch cambios02

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git checkout cambios02
Switched to branch 'cambios02'
```

d) Muestra el log con las opciones all y oneline.

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git log --all
commit 22f47e6926657fdb4f99ef0fbdce21f99b92f697 (cambios02)
Author: Lucas <lucasvarelanegro@gmail.com>
Date:   Sun Mar 20 23:24:32 2022 +0100

    Commit cambios02

commit ea7c4c58c03cc06340c10e8fb99443b22bfc2580 (HEAD -> master, cambios01)
Author: Lucas <lucasvarelanegro@gmail.com>
Date:   Sun Mar 20 23:15:54 2022 +0100

    Commit cambios01

commit 6cdf6d4dbee6928b0cee1883cb709bcc533a2170
Author: Lucas <lucasvarelanegro@gmail.com>
Date:   Sun Mar 20 23:13:11 2022 +0100

    Commit master
```

```
Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git log --oneline
ea7c4c5 (HEAD -> master, cambios01) Commit cambios01
6cdf6d4 Commit master
```

e) Muestra en el navegador el contenido de la web en los tres casos.

Hola mundo

Nova Web

Hola mundoqwugfvqwq

f) Fusiona master y cambios01.

```
Usuario@DESKTOP-8DC78S6 MINGW64 ~/Desktop/myweb (master)
$ git merge master cambios01
Updating 6cdf6d4..ea7c4c5
Fast-forward
 index.html | 13 ++++++-----
 style.css  |  2 --
 style2.css |  2 ++
 3 files changed, 9 insertions(+), 8 deletions(-)
 delete mode 100644 style.css
 create mode 100644 style2.css
```

g) Fusiona master y cambios02. ¿Algún conflicto? Explica por que. En caso afirmativo comprueba las líneas en conflicto y arregla la situación.

Hay conflicto por lo que debemos solucionar las líneas en conflicto y repetir el comando.

```
Usuario@DESKTOP-8DC78S6 MINGW64 ~/Desktop/myweb (master)
$ git merge master cambios02
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

h) Borra las ramas que no sean necesarias.

```
Usuario@DESKTOP-8DC78S6 MINGW64 ~/Desktop/myweb (master|MERGING)
$ git branch -D cambios01
Deleted branch cambios01 (was ea7c4c5).

Usuario@DESKTOP-8DC78S6 MINGW64 ~/Desktop/myweb (master|MERGING)
$ git branch -D cambios02
Deleted branch cambios02 (was 22f47e6).
```

Práctica 4.9: Operaciones con GitHub:

Cree una cuenta de GitHub y luego:

a) Crea un repositorio localmente como el de la práctica 4.5.

```

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb
$ git init
Initialized empty Git repository in C:/Users/Usuario/Desktop/myweb/.git/

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git add .

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git commit -m "Practica 5"
[master (root-commit) f94b6c8] Practica 5
 2 files changed, 10 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$

```

b) Crea un repositorio vacío en GitHub.

Vamos a nuestros repositorios en GitHub, le damos a new y le ponemos en este caso Practica9.

c) "Subir" el repositorio a GitHub.

Ponemos git remote add origin "enlace de nuestro repositorio":

```

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb (master)
$ git remote add origin https://github.com/LucasDam1/Practica9.git

```

```

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb(cambios) (master)
$ git add .

Usuario@DESKTOP-8DC7856 MINGW64 ~/Desktop/myweb(cambios) (master)
$ git commit -m "Commit cambios"
On branch master
nothing to commit, working tree clean

```

Para subirlo ponemos git remote push origin "enlace".

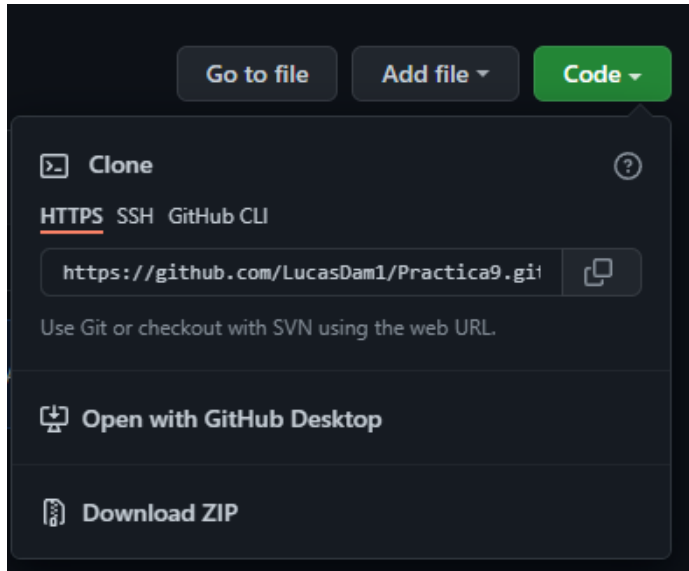
Nos reclamaran un token que debemos generar desde las opciones de nuestro perfil:



Una vez puesto el token y dándole a Sign in, se habrá subido a nuestro GitHub.

d) Eliminar el repositorio local y descargarlo con clon.

Eliminamos, pinchamos en Code y descargamos el clon:



e) Crear una sucursal localmente con cambios como en la tarea 4.6.

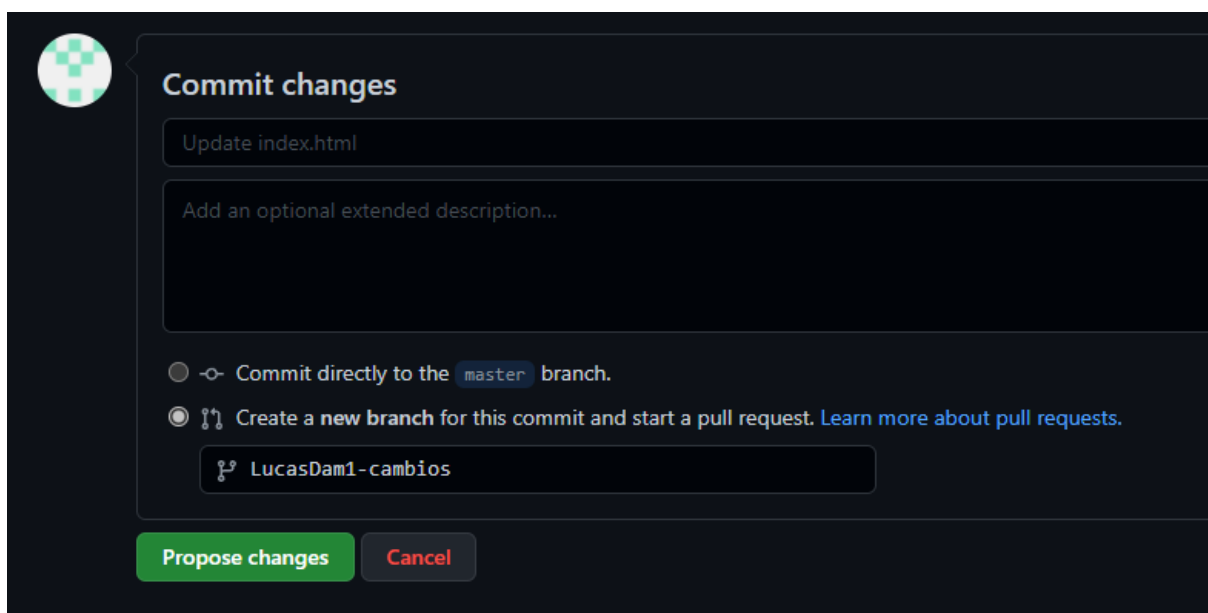
f) “Subir” la nueva rama.

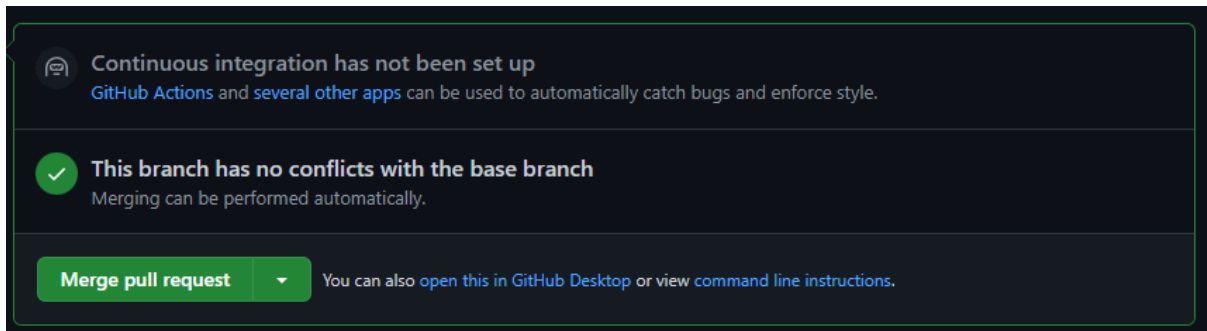
f) “Subir” la nueva rama.

g) En Github, no local: compara la nueva rama y el master y haz un merge.

h) Desde la consola git local, descarga la rama Master (pull) a local.

Podemos editar directamente desde Github los cambios pero seleccionando una nueva rama en vez de commit:





Luego para hacer merge vamos a pull request y pulsamos en el boton de merge.

Repetimos lo mismo para style.css.

i) Realice una solicitud de extracción en el repositorio de GitHub de un socio (agregando algún párrafo <p> al archivo index.html.

Para realizar la solicitud de extracción de un usuario de GitHub debemos ir a su repositorio y hacer click en Fork para pedir permiso o directamente descargar el archivo.

j) Verifique y acepte las "solicitudes de extracción" realizadas por sus compañeros.

Para verificar y aceptar las solicitudes de extracción lo haremos como si fuese una pull request.

Práctica 4.10: Git con Netbeans:

a) Crea un proyecto con dos archivos: una clase como la siguiente:

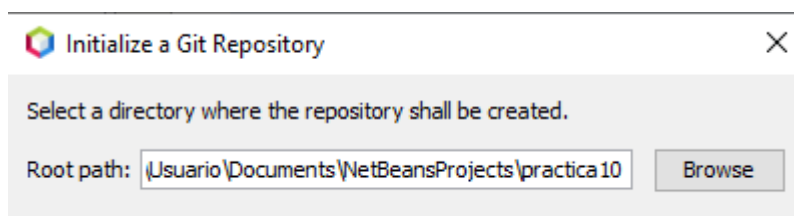
```
public class Sumador {
    public static float sumar (float a, float b ) {
        return a + b;
    }
}
```

y un programa que invoca alguno de sus métodos:

```
public class NewMain {
    public static void main(String[] args) {
        System.out.println(Sumador.sumar (15f,4f));
    }
}
```

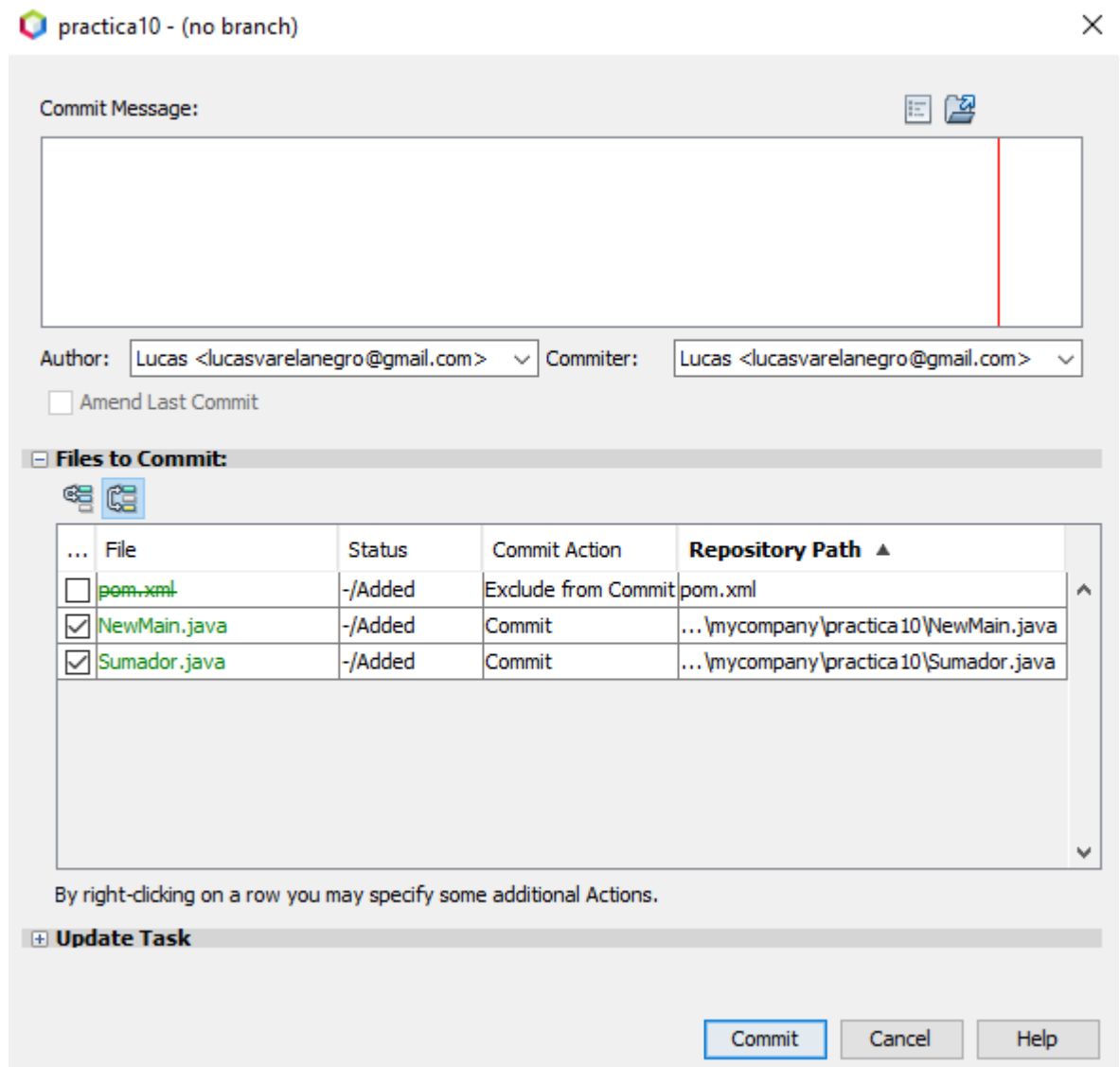
Haz que el proyecto tenga control de versiones con git. ¿Qué color tienen los nombres de los archivos?

Hacemos click derecho en el proyecto, exactamente Versioning > Initialize a Git repository, los archivos tienen color verde:



b) Haz commit del proyecto. ¿Qué color tienen ahora los nombres de los archivos?

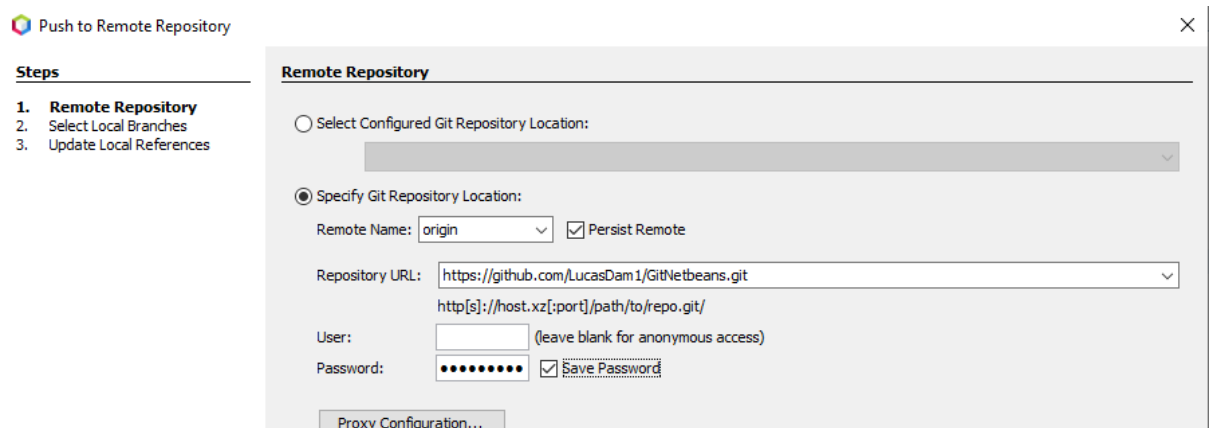
Hacemos click derecho, Git > Commit aceptamos el commit:



Tras esto los archivos no tienen ningún color especial.

c) Haz push del proyecto a un nuevo repositorio en tu cuenta de GitHub.

Click derecho, Git > Remote > Push y ponemos la Url de nuestro repositorio de GitHub:



- d) Crea en local una rama llamada “dev” en la que, elimina la clase Sumador del punto anterior y crea una nueva llamada Restador:

```
public class Restador {  
    public static float restar (float a, float b ){  
        return a - b;  
    }  
}
```

En la clase NewMain que contiene el main() llama al nuevo método en vez de al anterior:

```
System.out.println(Restador.restar (15f,4f));
```

Haz commit de la nueva rama:

Click derecho en el proyecto, Git > Branch/Tag > Create Branch y le ponemos el nombre dev:

Create Branch

Branch Name: dev

Revision: master Select

Commit ID: master (b6625f88fb)

Author: Lucas <lucasvarelanegro@gmail.com>

Date: 21 mar. 2022 2:23:47

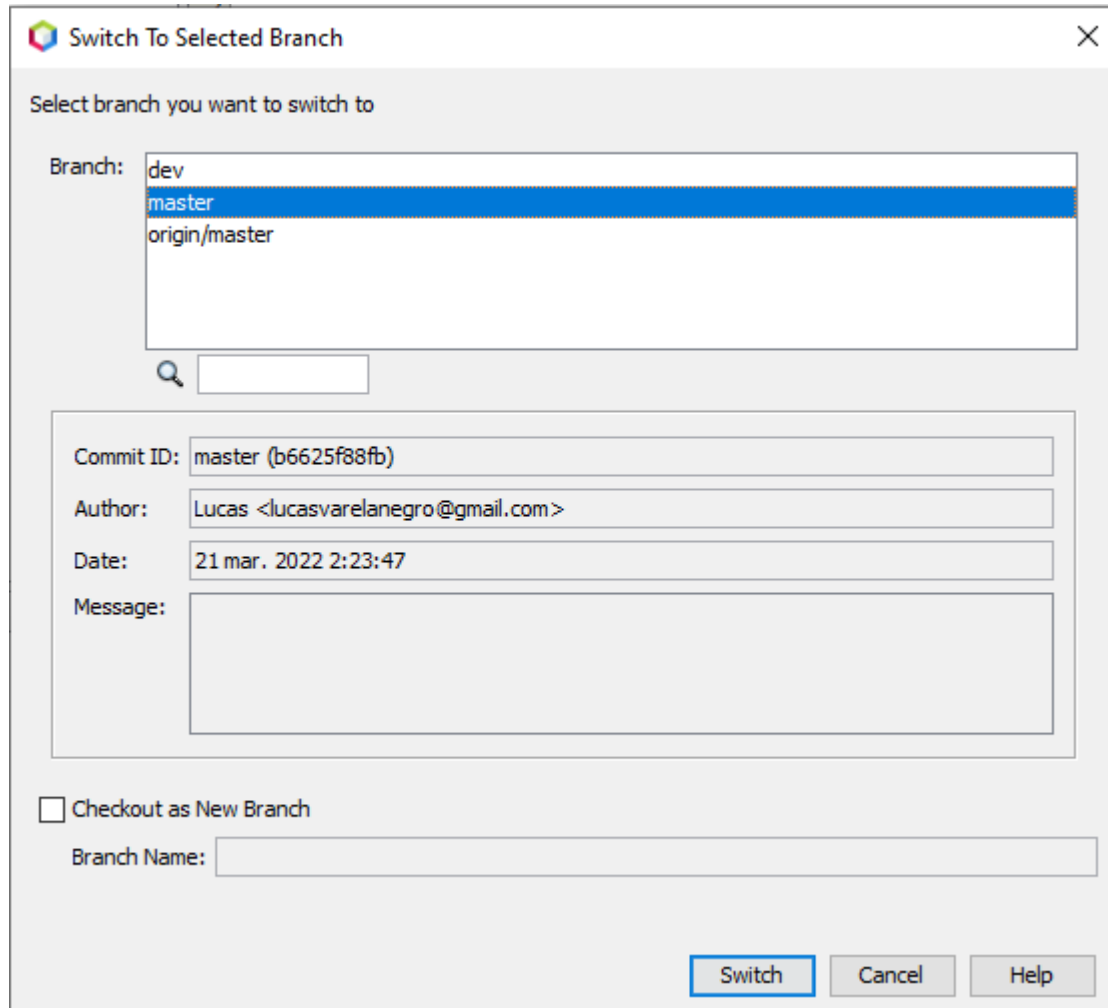
Message:

☒ Checkout Created Branch

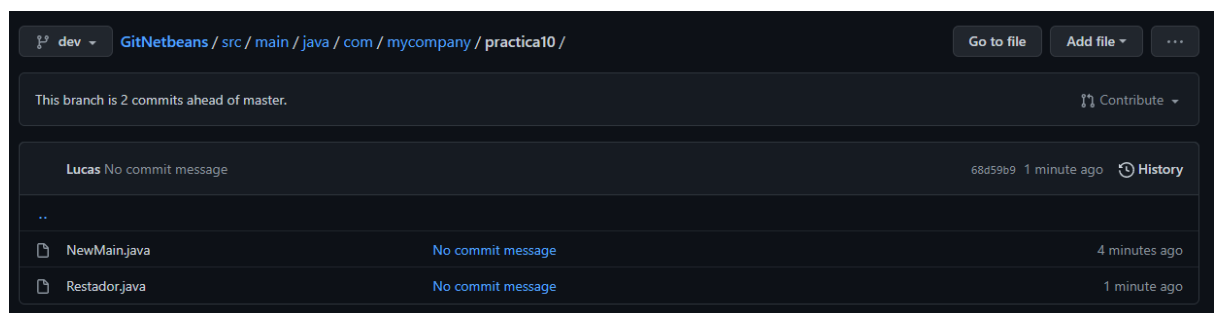
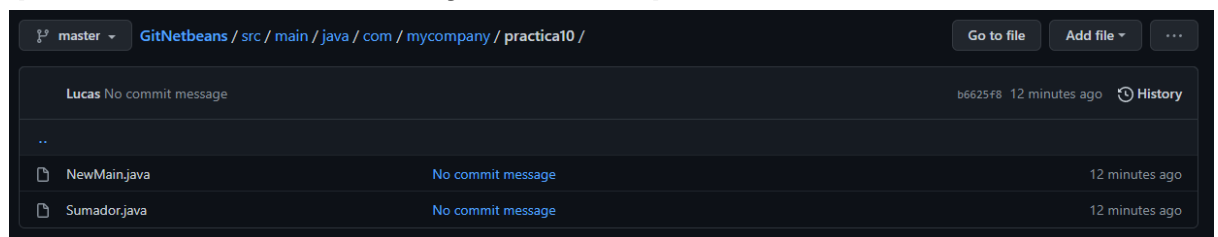
Create Cancel Help

Hacemos los cambios indicados y hacemos el commit en Git > Commit.

- e) Prueba en Netbeans a hacer git > Branch > Switch to... para cambiar de una rama a otra y verás como en la ventana de proyectos ves el archivo Sumador.java o Restador.java segun la rama en la que estés.

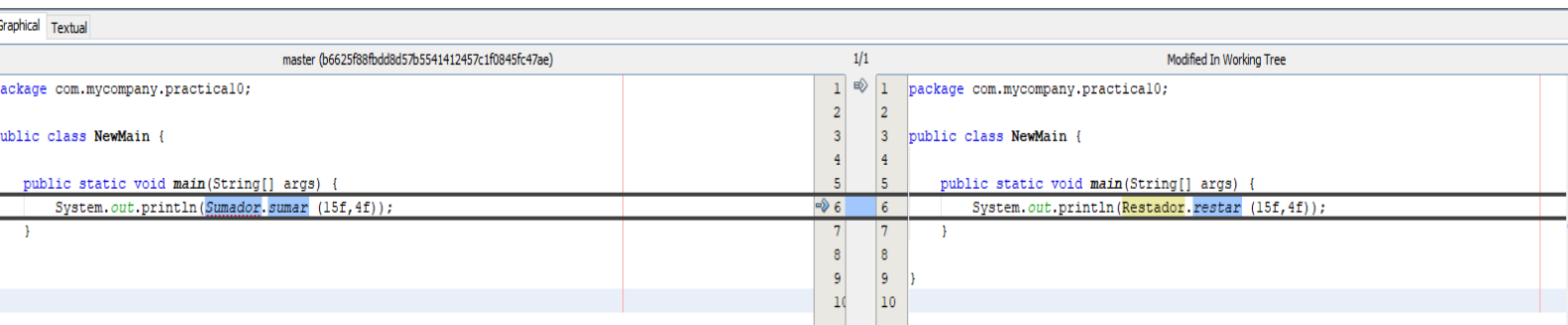
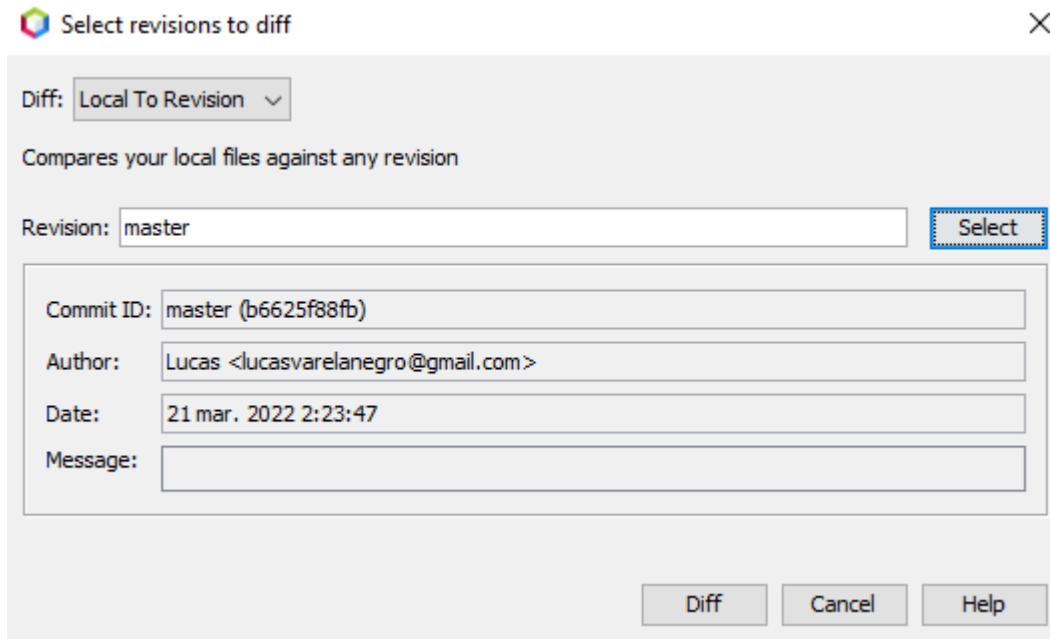


- f) Ve a tu cuenta de GitHub y haz dos capturas de pantalla, una en la que se vea la rama master y otra en la que se vea la rama dev.



- g) En Netbeans, comprueba las diferencias entre la clase NewMain de la versión master con las de la versión dev.

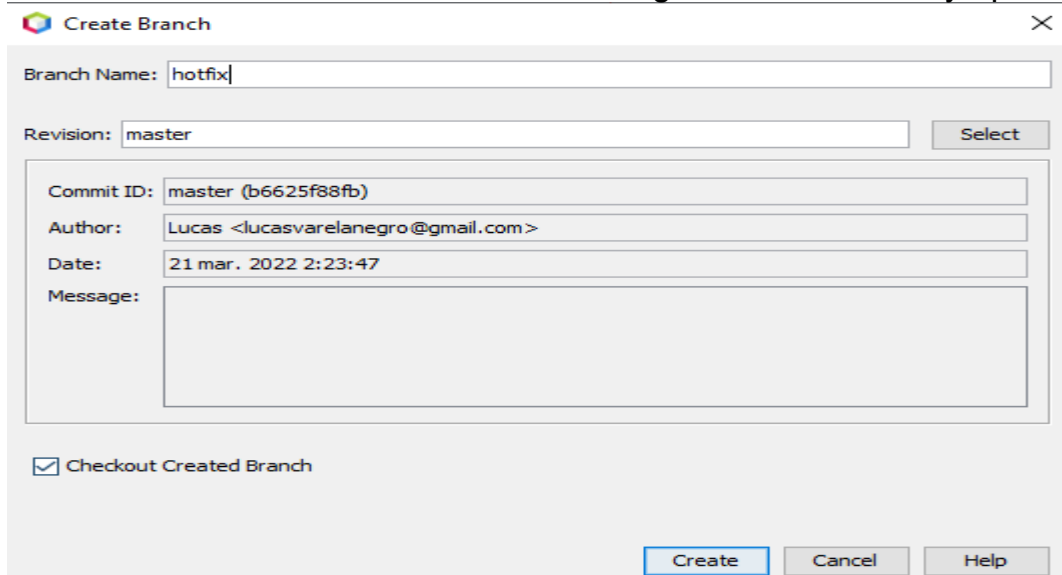
Accedemos a Git > Diff > Diff to... y seleccionamos la otra rama:



- h) Crea una rama llamada “hotfix” a partir de master en la que cambias la línea por:

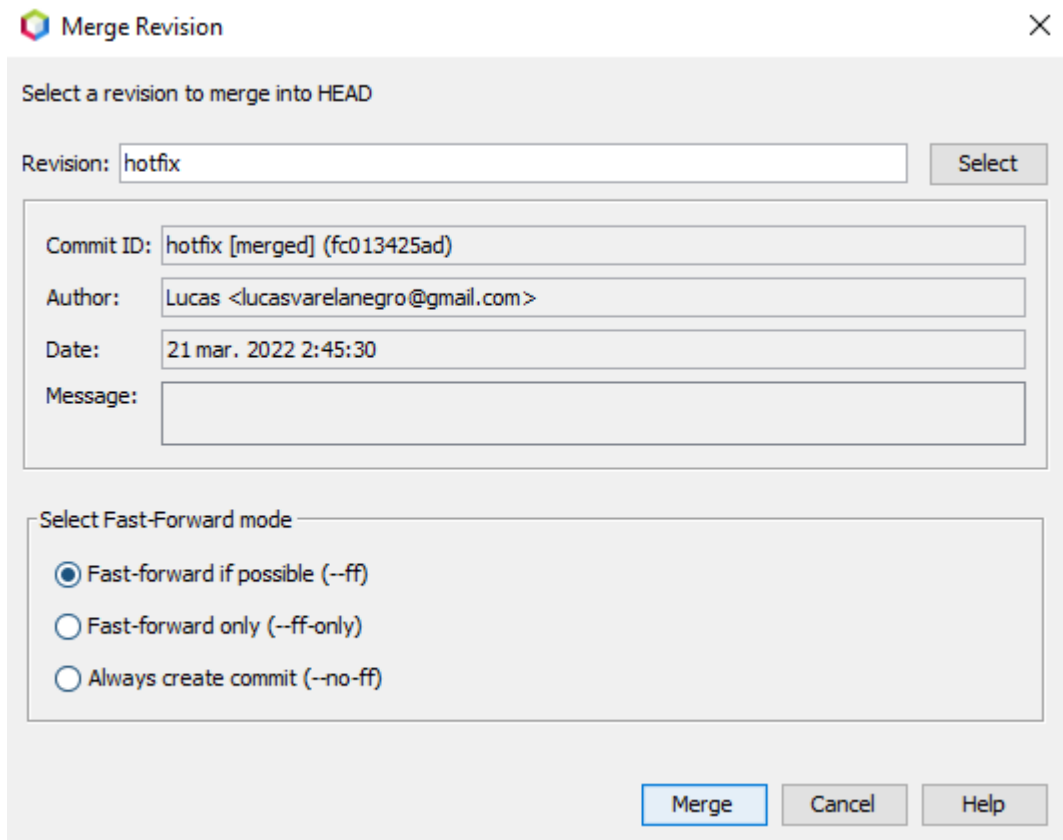
```
System.out.println(Sumador.sumar (15f,4f));
System.out.println(Sumador.sumar (16f, 4f));
```

Creamos la rama desde Git > Branch/Tag > Create Branch y aplicamos:



- i) **Haz commit de la rama hotfix y aplica los cambios sobre la rama master. ¿Es posible fast forward?**

Vamos a Git > Branch/Tag > Merge revision:



Merge Revision

Select a revision to merge into HEAD

Revision:

Commit ID:

Author:

Date:

Message:

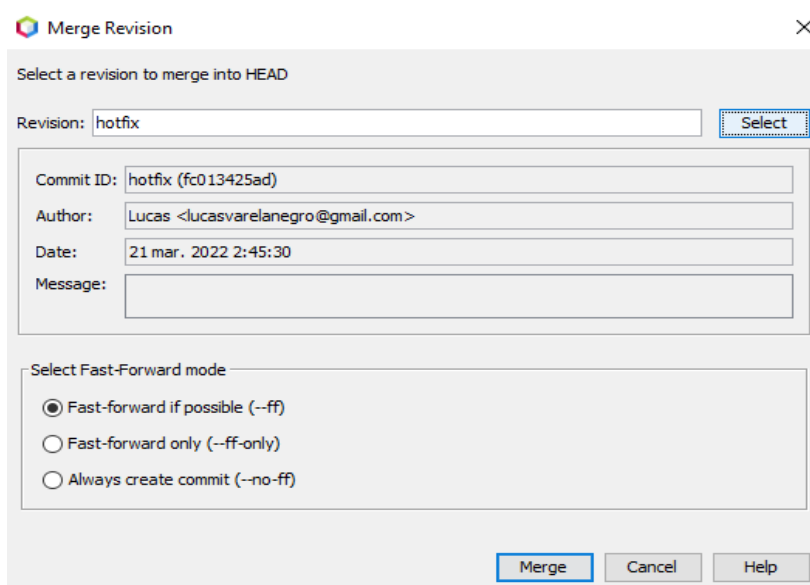
Select Fast-Forward mode

☒ Fast-forward if possible (--ff)

☐ Fast-forward only (--ff-only)

☐ Always create commit (--no-ff)

- j) **Aplica los cambios de la rama dev sobre master. Si hay conflictos resuélvelos:**



Merge Revision

Select a revision to merge into HEAD

Revision:

Commit ID:

Author:

Date:

Message:

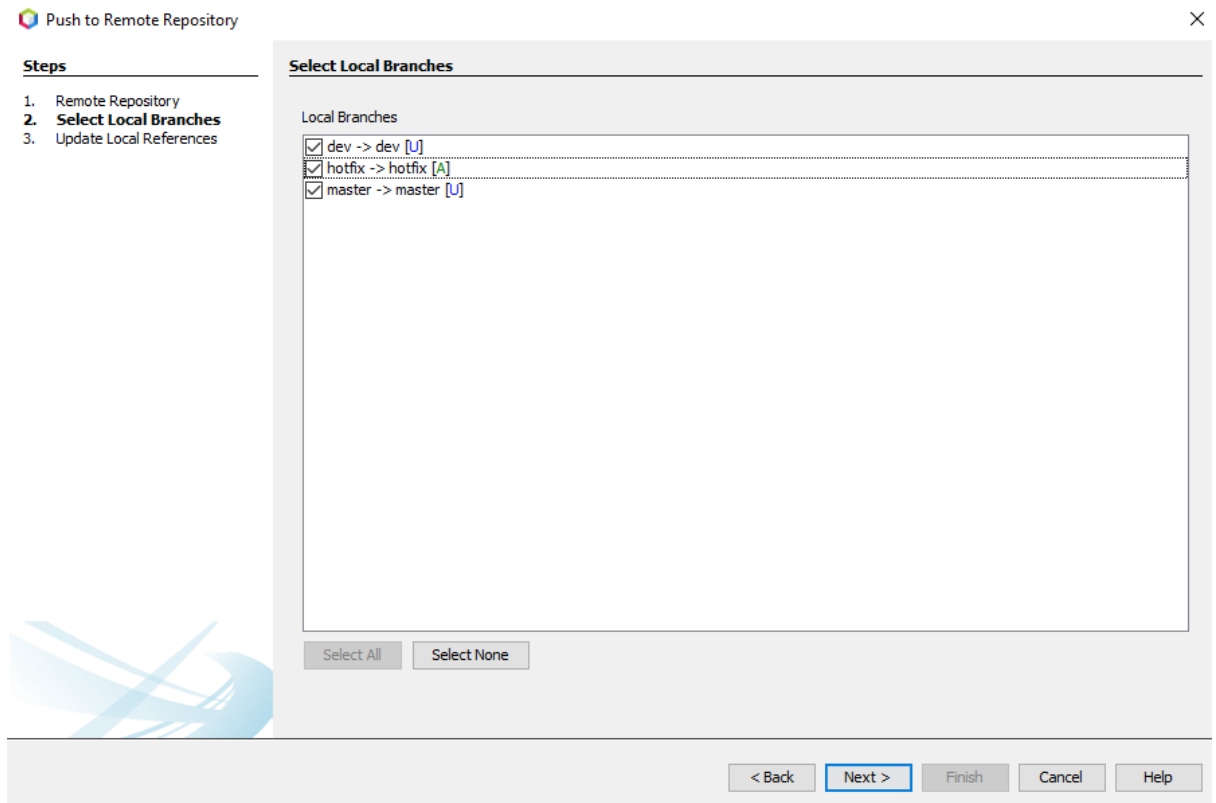
Select Fast-Forward mode

☒ Fast-forward if possible (--ff)

☐ Fast-forward only (--ff-only)

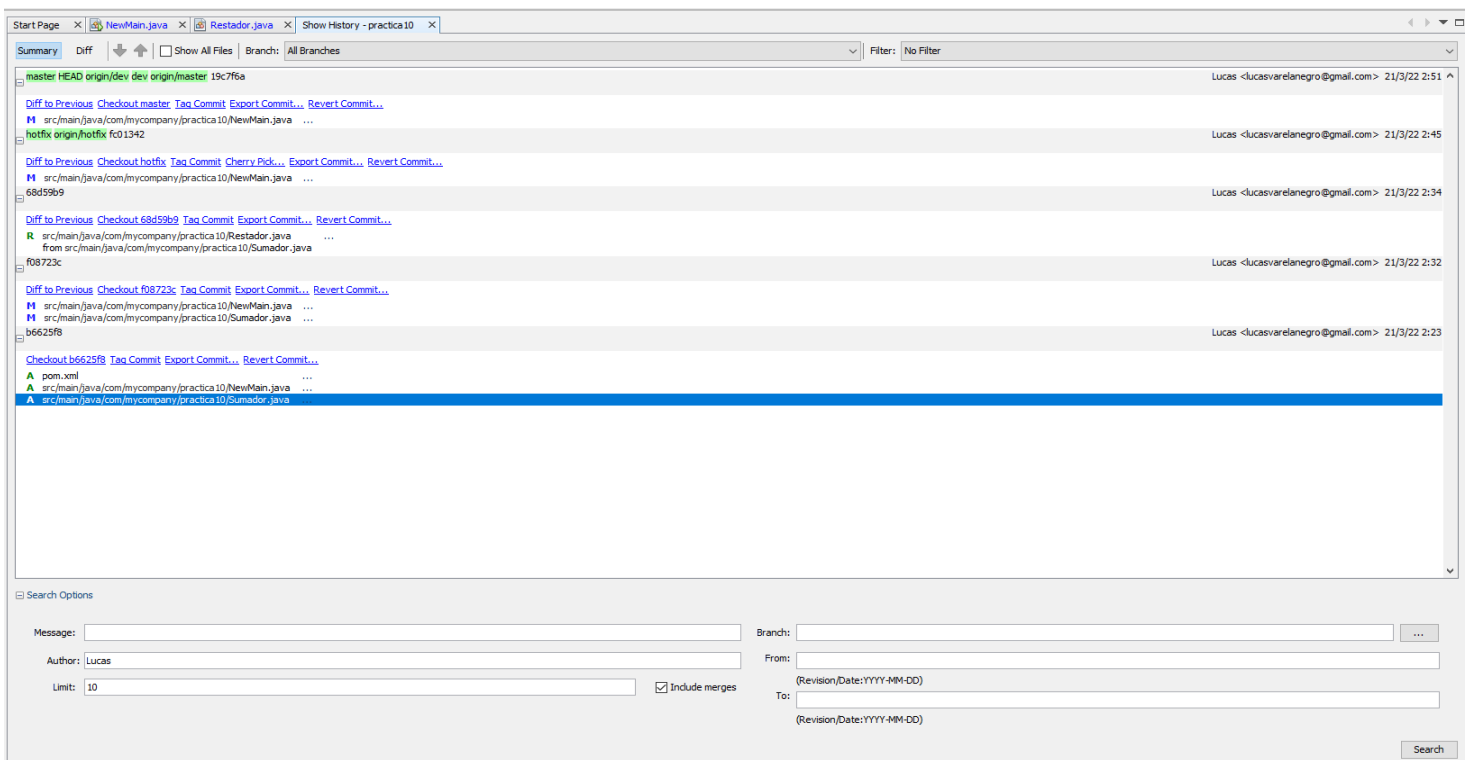
☐ Always create commit (--no-ff)

k) Haz push de todas las ramas al repositorio remoto.



l) Muestra un histórico de los cambios hechos.

Git > Show History:



Token GITHUB: ghp_uyqfu5vc8aDqMKwZo9BCIYviBYQYcm3yCHMb