

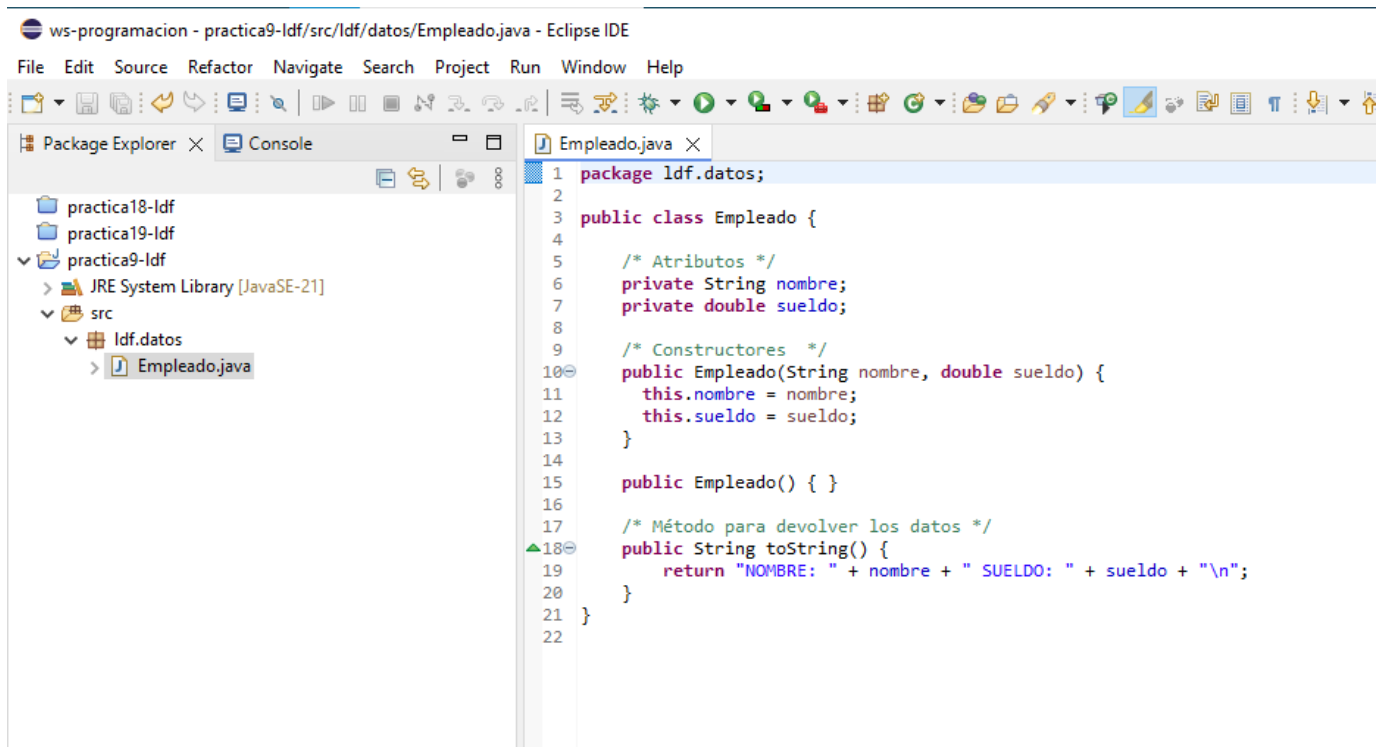
Práctica 9 – Control de versiones

En esta práctica utilizaremos el Git (con su plugin en Eclipse) para realizar un control de versiones de una aplicación en su desarrollo.

Práctica

Documenta todos los pasos realizados para realizar lo indicado en cada apartado. En varios pasos necesitarás archivos que se proporcionan con esta práctica.

1. Crear un proyecto llamado **practica9-xxx** (siendo xxx tus iniciales) e incorpora a este proyecto el archivo *Empleado.java* en el paquete xxx.datos (siendo xxx tus iniciales). Haz los cambios que sean necesarios para que la clase compile correctamente, en caso de que no lo haga.

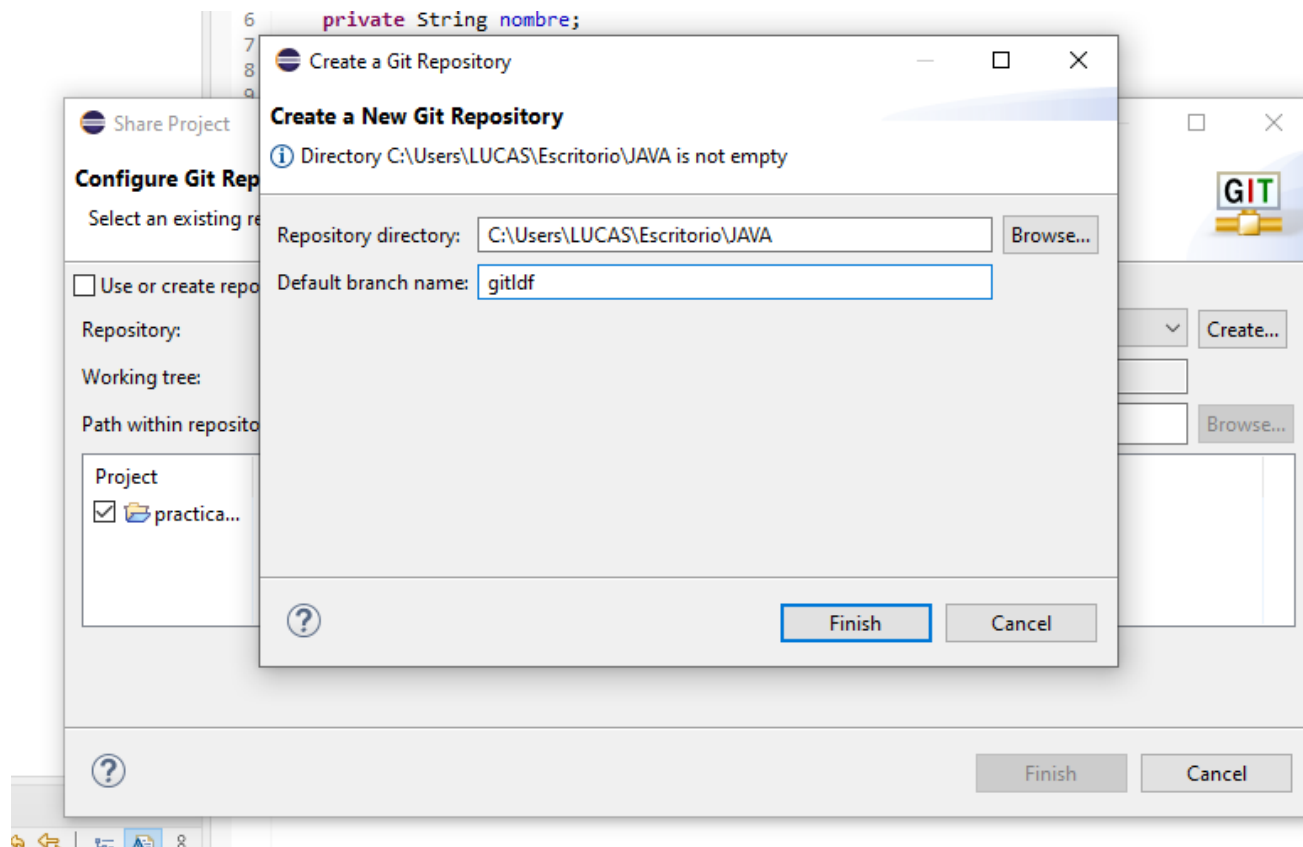


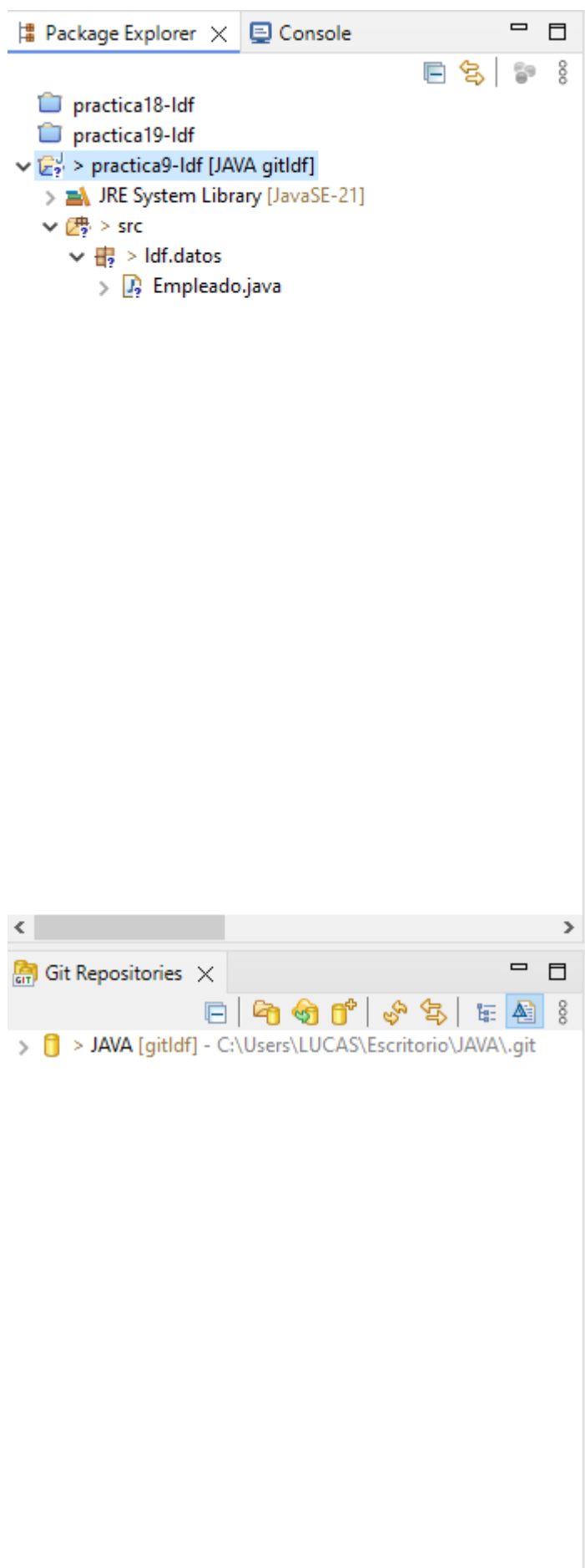
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with folders 'practica18-ldf', 'practica19-ldf', and 'practica9-ldf'. Under 'practica9-ldf', there is a 'src' folder containing an 'ldf.datos' package, which in turn contains the 'Empleado.java' file. The main editor window shows the code of 'Empleado.java' with the following content:

```
1 package ldf.datos;
2
3 public class Empleado {
4
5     /* Atributos */
6     private String nombre;
7     private double sueldo;
8
9     /* Constructores */
10    public Empleado(String nombre, double sueldo) {
11        this.nombre = nombre;
12        this.sueldo = sueldo;
13    }
14
15    public Empleado() { }
16
17    /* Método para devolver los datos */
18    public String toString() {
19        return "NOMBRE: " + nombre + " SUELDO: " + sueldo + "\n";
20    }
21 }
22
```

Creemos un nuevo proyecto seleccionando la pestaña de Java Project, una vez pulsado pondremos el nombre de practica9-xxx (sindo xxx las 3 iniciales de nuestro nombre y apellidos). Crearemos dentro del proyecto en la carpeta src un paquete de nombre xxx.datos (sindo xxx las 3 iniciales de nuestro nombre y apellidos) y añadiremos a este la clase Empleados.java que nos proporciona el profesor.

2. Crea un repositorio Git para llevar un control de este proyecto en un directorio de nombre gitxxx siendo xxx tus iniciales). Este directorio deberá estar al mismo nivel que el workspace de Eclipse (es decir, si la ruta del workspace es D:\workspace-ed la del repositorio será D:\gitxxx. Añade el proyecto al repositorio (incluye capturas de la vista Explorador de paquetes, la vista Git Repositoriesy la carpeta del repositorio en el Explorador de Windows con el resultado final).





JAVA				
Nombre	Fecha de modificación	Tipo	Tamaño	
eclipse	19/10/2024 20:14	Carpeta de archivos		
SQLiteDatabaseBrowserPortable	20/04/2025 21:16	Carpeta de archivos		
.git	07/05/2025 12:07	Carpeta de archivos		
practica9-ldf	07/05/2025 12:07	Carpeta de archivos		
ws-programacion	07/05/2025 12:07	Carpeta de archivos		
sqlite-jdbc-3.49.1.0.jar	03/04/2025 12:50	Executable Jar File	13.983 KB	

JAVA > practica9-ldf				
Nombre	Fecha de modificación	Tipo	Tamaño	
.settings	07/05/2025 11:48	Carpeta de archivos		
bin	07/05/2025 11:49	Carpeta de archivos		
src	07/05/2025 11:49	Carpeta de archivos		
.classpath	07/05/2025 11:48	Archivo CLASSPATH	1 KB	
.gitignore	07/05/2025 12:07	Archivo de origen ...	1 KB	
.project	07/05/2025 11:48	Archivo PROJECT	1 KB	

Para la creación del repositorio de Git, nos dirigiremos a la zona superior del Eclipse donde dice Window → Show View → Other → Git → Git Repositories. Una vez tengamos la ventana en el Eclipse, pulsaremos click derecho sobre nuestro proyecto y le daremos a Team → Share Projects. Después indicaremos la ubicación del repositorio que debe estar en la misma ubicación que el workspace y pondremos el nombre de gitxxx (sino xxx las 3 iniciales de nuestro nombre y apellidos).

3. Confirma en el repositorio los archivos de código fuente del proyecto con el mensaje "Commit inicial del proyecto".

> JAVA [gitldf]

Unstaged Changes (4402)

- .api_description - eclipse/plugins/org.eclipse.jdt.debug_3.21.500.v20240
- .classpath - ws-programacion/practica18-ldf
- .classpath - ws-programacion/practica19-ldf
- .contributions.42 - eclipse/configuration/org.eclipse.core.runtime
- .contributors.42 - eclipse/configuration/org.eclipse.core.runtime
- .eclipseproduct - eclipse
- .empty - eclipse/configuration/org.eclipse.osgi/337/0/.cp
- .extraData.42 - eclipse/configuration/org.eclipse.core.runtime
- .fileTable.48 - eclipse/configuration/org.eclipse.core.runtime/.manager

Staged Changes (7)

- .api_description - eclipse/configuration/org.eclipse.osgi/456/0/.cp
- .classpath - practica9-ldf
- .gitignore - practica9-ldf
- .project - practica9-ldf
- Empleado.java - practica9-ldf/src/ldf/datos
- org.eclipse.core.resources.prefs - practica9-ldf/.settings
- org.eclipse.jdt.core.prefs - practica9-ldf/.settings

Commit Message

i Unborn branch: this commit will create the branch 'gitldf'.

Commit inicial del proyecto

Author: LUCAS <LUCAS@DESKTOP-8FUS8TI>

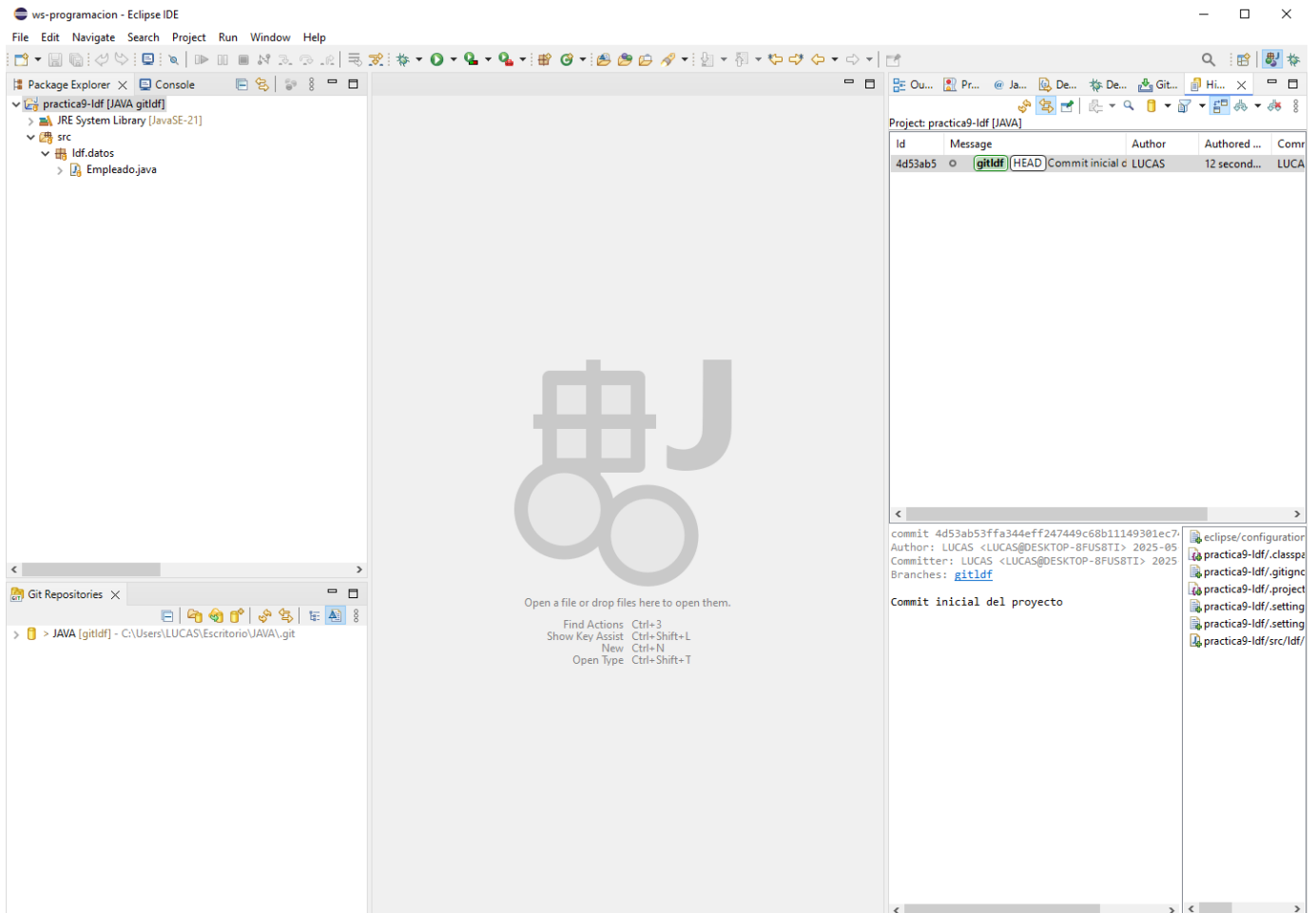
Committer: LUCAS <LUCAS@DESKTOP-8FUS8TI>



Commit and Push...

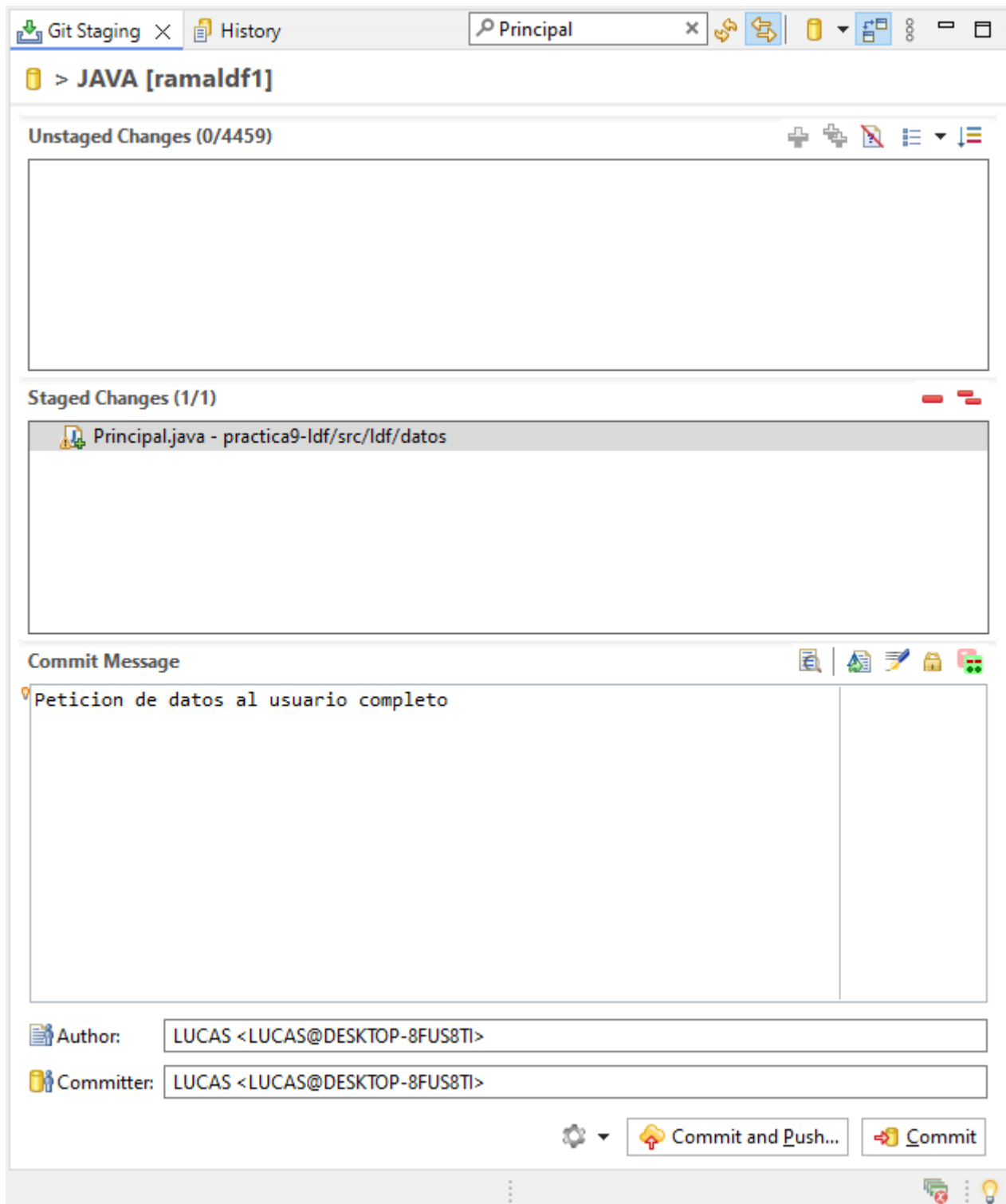


Commit



Abriremos la ventana Git Staging, para ellos nos dirigiremos a Window → Show View → Other → Git → Git Staging. Después de tener la ventana tendremos que agregar los archivos del proyecto a Staged changes seleccionándolos y dándole al icono del + en color verde. Después en Commit Message escribiremos el mensaje que nos pide el ejercicio. Posteriormente, iremos a al proyecto y haremos click derecho y seleccionaremos Team → Show in history y veremos los archivos del proyecto junto con el mensaje del commit.

4. Crea una rama de nombre *ramaxxx1* (siendo xxx tus iniciales) para desarrollar el apartado de pedir datos al usuario de la clase principal del proyecto. El código fuente de la clase (Principal.java) está ya desarrollado y solo tendrás que incluir dicho archivo en el proyecto en el paquete xxx (siendo xxx tus iniciales). Haz los cambios que sean necesarios para que la clase compile correctamente, en caso de que no lo haga. Confirma en el repositorio los cambios en el proyecto con el mensaje “Petición de datos al usuario completado” (incluye capturas de la vista Explorador de paquete y del historial correspondiente a esta rama del repositorio).



Git Staging
History X

Project: practica9-ldf [JAVA]

Id	Message	Author	Authored ...	Committer	Committe...
f077654	ramaldf1 (HEAD) Peticion de datos al	LUCAS	15 second...	LUCAS	15 second...
4d53ab5	gitldf Commit inicial del proyecto	LUCAS	20 minute...	LUCAS	20 minute...

commit f077654cc219e603840f6ff68aabaj9b3a671f58

Author: LUCAS <LUCAS@DESKTOP-8FUS8TI> 2025-05-07 12:39:35

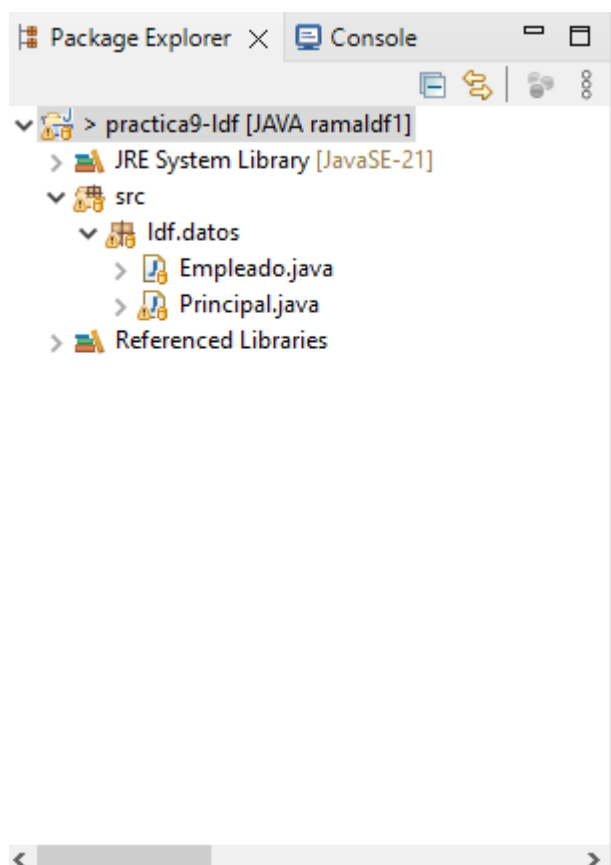
Committer: LUCAS <LUCAS@DESKTOP-8FUS8TI> 2025-05-07 12:39:35

Parent: [4d53ab53ffa344eff247449c68b11149301ec741](#) (Commit inici

Branches: [ramaldf1](#)

Peticion de datos al usuario completo

practica9-ldf/src/ldf/datos/Prii



Debermos de incluir Principal.java en el paquete y para que compile deberemos de añadir al classpath los .jar que nos proporciona el profesor. Para crear la rama pulsaremos click derecho sobre el gitxxx (siendo xxx tus iniciales), pulsaremos Switch to → New Branch, y pondremos el nombre de ramaxxx1 (siendo xxx tus iniciales). Posteriormente iremos a la pestaña Git Staging y añadiremos Principal.java dondole a el icono de + verde, y pondremos en la zona de Commit Message el mensaje que nos ha mandado el profesor. Y podremos revisarlo yendo a la rama dando click derecho Team → Show on history.

5. Vuelve a trabajar en la rama original (rama main) para simular que se están desarrollando otros apartados de la clase principal ejecutable del proyecto. El código fuente de la clase (Principal.java) está en la carpeta Segundo Principal que proporciono. Inclúyelo en el paquete xxx (siendo xxx tus iniciales), ¿qué problemas te encuentras para que compile correctamente? ¿por qué? Haz los cambios que sean necesarios para que la clase compile correctamente y comprueba que se ejecuta correctamente. Confirma en el repositorio todos los cambios realizados en el proyecto con el mensaje “Incluida funcionalidad de la aplicación ejecutable excepto Pedir datos”. (incluye captura con el historial de la rama main).

ws-programacion - practica9-ldf/src/ldf/datos/Principal.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Console

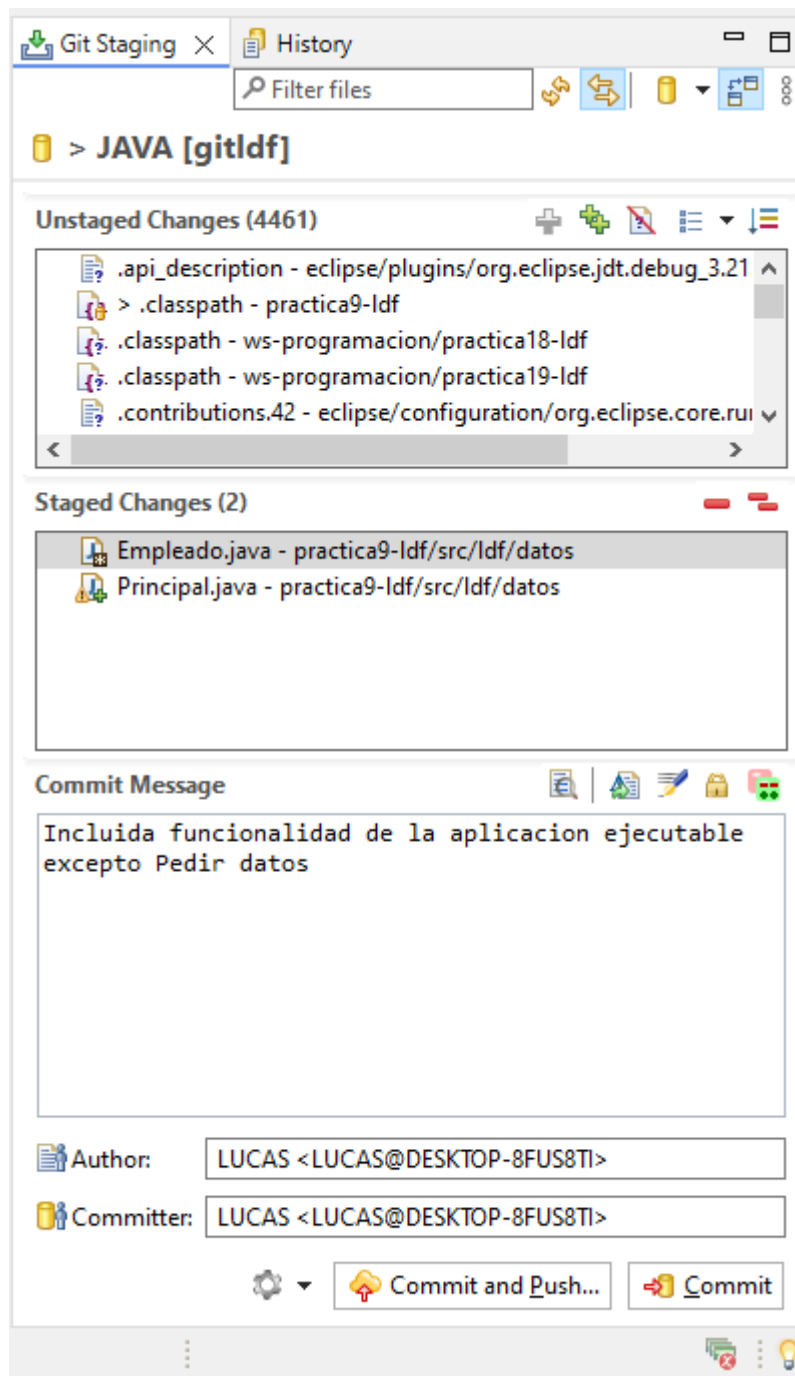
practica9-ldf [JAVA gitldf]

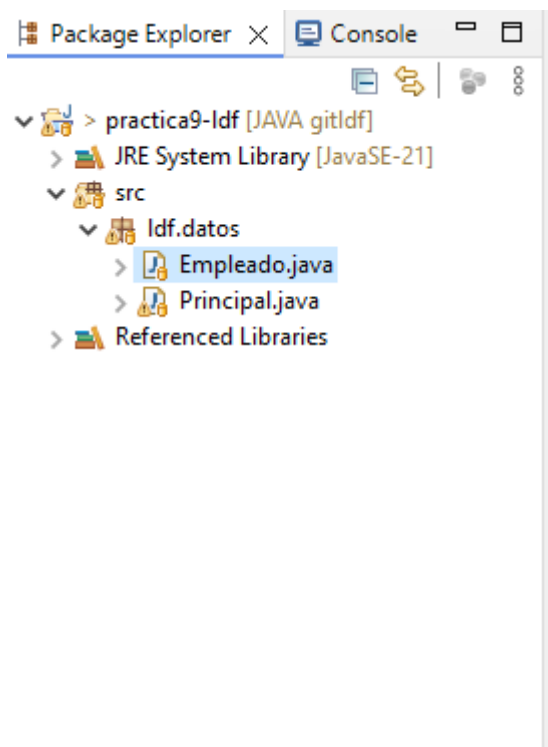
- JRE System Library [JavaSE-21]
- src
 - ldf.datos
 - Empleado.java
 - Principal.java
- Referenced Libraries

```
1 package ldf.datos;
2 import java.util.Iterator;
3
4 public class Principal {
5
6     // Atributo
7     private static LinkedList<Empleado> empresa;
8
9     // Constructor
10    Principal() {
11        empresa = new LinkedList<Empleado>();
12        int opcion = 0;
13        Empleado empleado = null;
14        do {
15            opcion = CrearMenus.crearMenu("1.- Añadir\n2.- Borrar por nombre\n3.- Borrar por índice\n4.- Borrar por índice\n5.- Modificar por índice\n6.- Insertar en una posición dada\n7.- Salir\n");
16            switch (opcion) {
17                case 1:
18                    // TO-DO
19                    break;
20                case 2:
21                    /* Listar por orden de entrada */
22                    mostrarOrdenEntrada();
23                    break;
24                case 3:
25                    /* Borrar por nombre */
26                    borrarPorNombre();
27                    break;
28                case 4:
29                    /* Borrar por índice */
30                    borrarPorIndice();
31                    break;
32                case 5:
33                    /* Modificar por índice */
34                    modificarPorIndice();
35                    break;
36                case 6:
37                    /* Insertar en una posición dada */
38                    insertarPosicionDada();
39                    break;
40                case 7:
41                    System.exit(0);
42            }
43        } while (opcion != 7);
44    }
45 }
```

Git Repositories

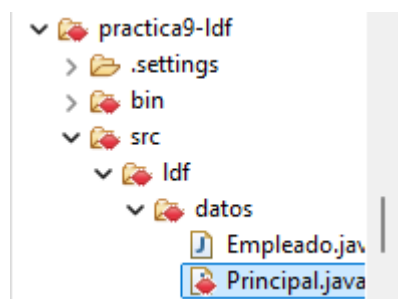
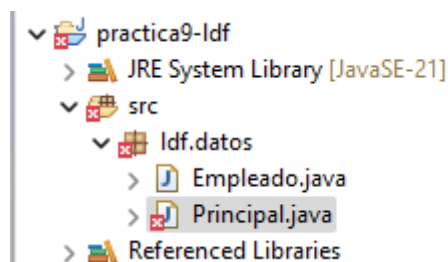
> JAVA [gitldf] - C:\Users\LUCAS\Escritori





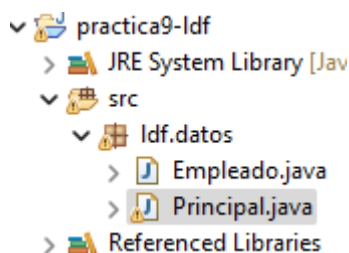
Cambiamos a la rama Main, para ellos iremos a la pestaña Git Repositories y daremos click derecho, Switch to → gitxxx (siendo xxx tus iniciales) .Añadimos la clase Principal de la carpeta que Segundo Principal, para que compile generamos getters and setters en la clase Empleado.java. Una vez tengamos la nueva clase con perfecta compilación iremos de nuevo a la pestaña Git Staging y cogeremos los archivos modificados y daremos al icono + verde, y pondremos en la zona de Commit Message el mensaje que nos ha mandado el profesor. Y podremos revisarlo yendo a la rama dando click derecho Team → Show on history.

6. Fusiona a la rama *Main* la rama *ramaxxx1*, e incluye una captura de pantalla del Explorador de paquetes con la situación en que ha quedado el proyecto.

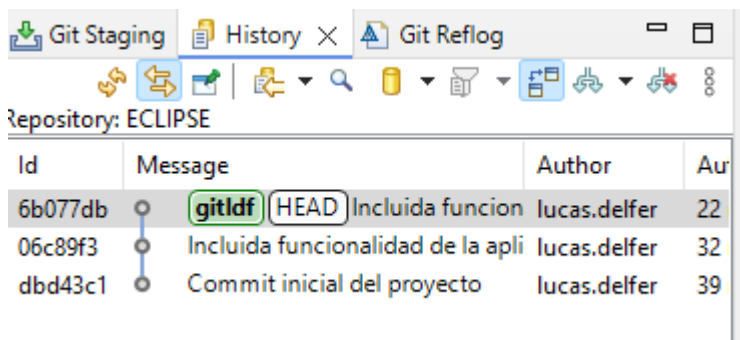


7. Verás que la fusión de las dos ramas (*main* y *ramaxxx1*) no ha sido limpia y hay conflictos en uno de los ficheros, con las herramientas de que dispone Eclipse resuelve los conflictos para que la aplicación tenga toda la funcionalidad de ambas ramas y confirma en el repositorio los cambios. ¿Con qué mensaje se ha guardado la confirmación?

Lo he cambiado manualmente usando el código proporcionado, ya que no tenía acceso para poder solucionarlo con la herramienta de Eclipse. Ahora compila correctamente y el código funciona.

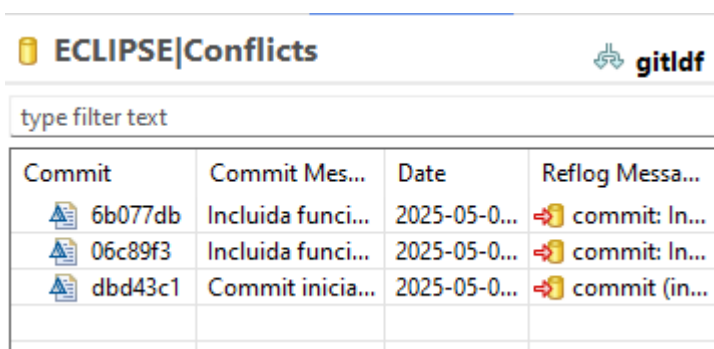
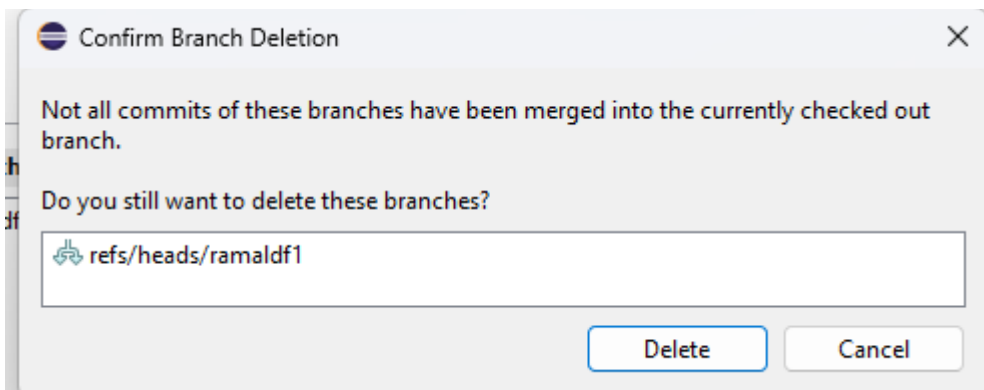


8. Incluye una captura de pantalla de la vista History de la rama main, ¿qué significa la imagen que aparece en la columna message y las etiquetas de las diferentes confirmaciones?



Significa que los commits se hicieron de manera lineal.

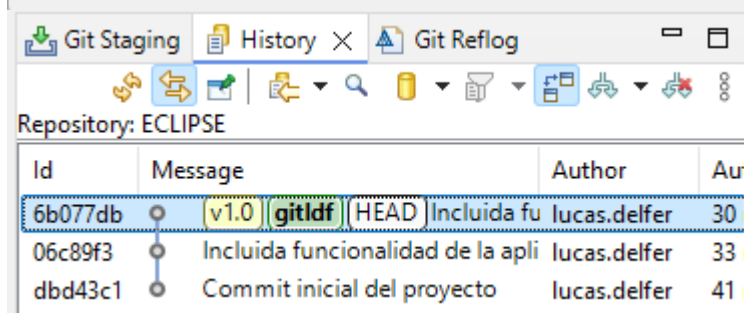
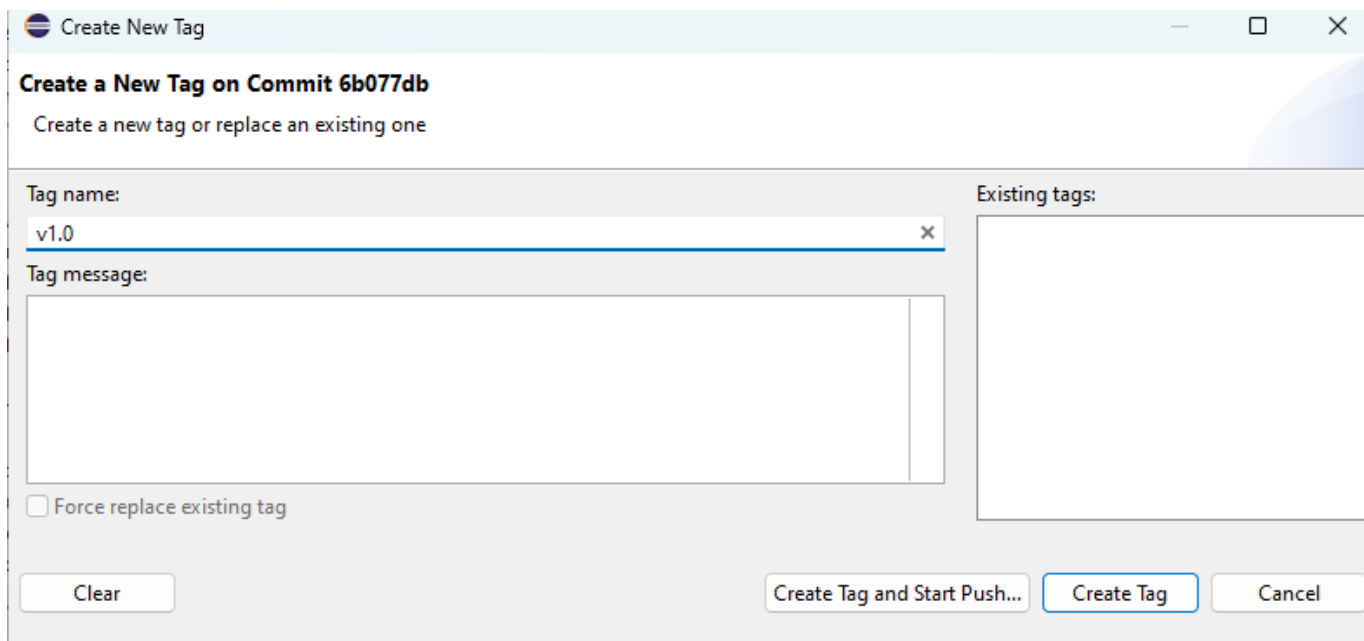
9. Elimina la rama *ramaxxx1*, ¿qué ha sucedido? ¿qué información o archivos se han perdido? Incluye de nuevo una captura de pantalla de la vista History de la rama main.



Para eliminarla tendremos que ir a la pestaña Git Repositories, desplegaremos el repositorio y haremos click en Branches --> Local --> ramaxxx1 (siendo xxx tus iniciales). Se han perdido los commit que habíamos realizado con la rama. Y por tanto ahora nuestro repositorio solo funciona con los del repositorio principal.

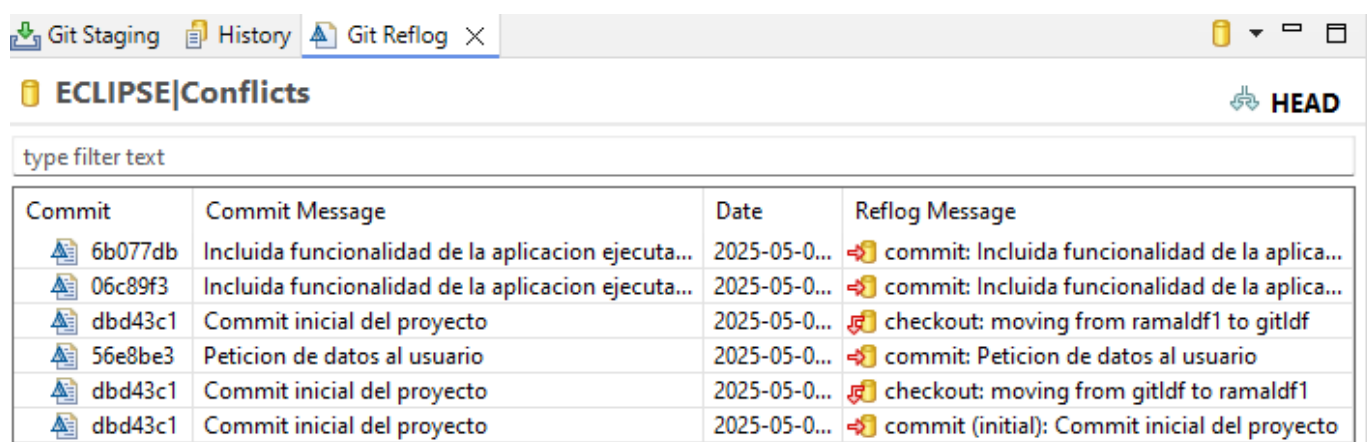
10. Crea una etiqueta de la versión final del proyecto con el nombre v1.0.





Para poner el tag iremos a History del main del repostirorio y daremos click derecho sobre el ultimo commit que hayamos realizado, clickearemos en Create tag y pondremos el nombre que nos ha mandado el profesor.

11. Incluye una captura de pantalla que muestre todos los pasos que has realizado en Git para realizar esta práctica.



1. Crear el proyecto, crear el paquete e introducir la clase .java.
2. Crear el repositorio Git en la ubicación a la misma altura donde tengamos el workspace.
3. Introducir en el repositorio los archivos de la practica9 y poner un mensaje de Commit para indicar que hace dicha acción.



4. Crear una rama (ramaxxx1) donde introduciremos la otra clase Principal.java y añadiremos los .jar para que compile. Despues añadiremos los archivos al repositorio de la rama creada poniendo un mensaje de Commit para indicar dicha acción.
5. Volveremos a trabajar con la rama main y añadiremos la clase Principal.java (2.0) y haremos los cambios necesarios para que compile, en este caso generación de getters y setters en la clase Empleado. Posteriormente añadiremos los archivos al repositorio poniendo un mensaje de Commit para indicar dicha acción.
6. Fusionaremos la rama main con la rama creada anteriormente (ramaxxx1) y ver que ocurre al fusionarlo. Se ve que hay un conflicto y hay que arreglarlo para que el cdigo compile. (En mi caso de manera manual, pero también hay otros métodos que ofrece Eclipse, yo no he conseguido poder usarles).
7. Ver el History de la rama main del repositorio e indicar que significa la imagen.
8. Eliminar la rama creada (ramaxxx1).
9. Crear una etiqueta.