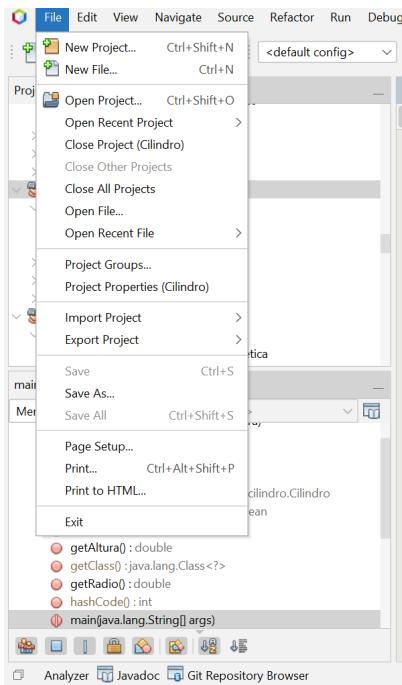
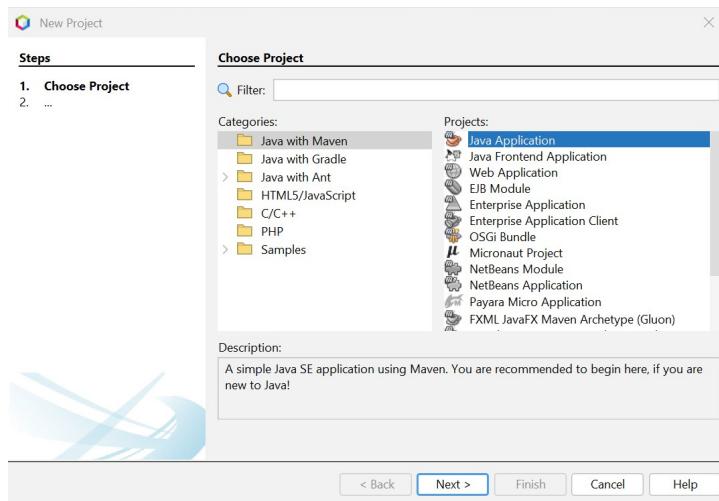


1.1.Crear un nuevo proyecto con el nombre “Lacteos”.

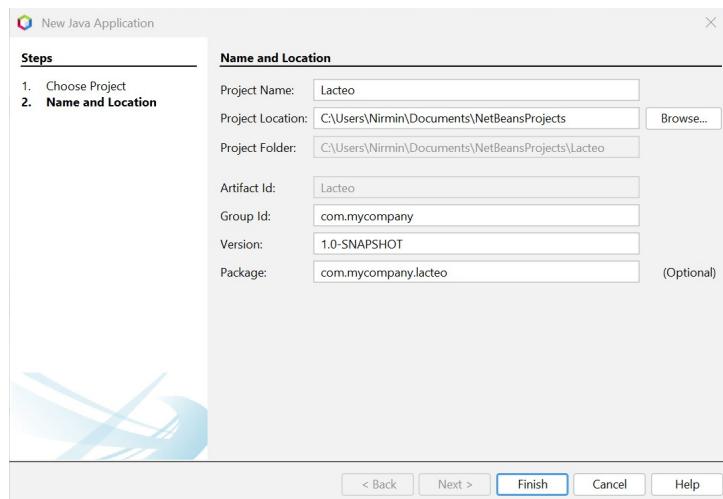
Creamos un nuevo proyecto de netbeans



Seleccionamos Java



Llamamos “Lacteo” al proyecto y escogemos la carpeta donde queremos guardarlo. En mi caso en el escritorio en una carpeta específica para el examen y pulsamos “Finish”



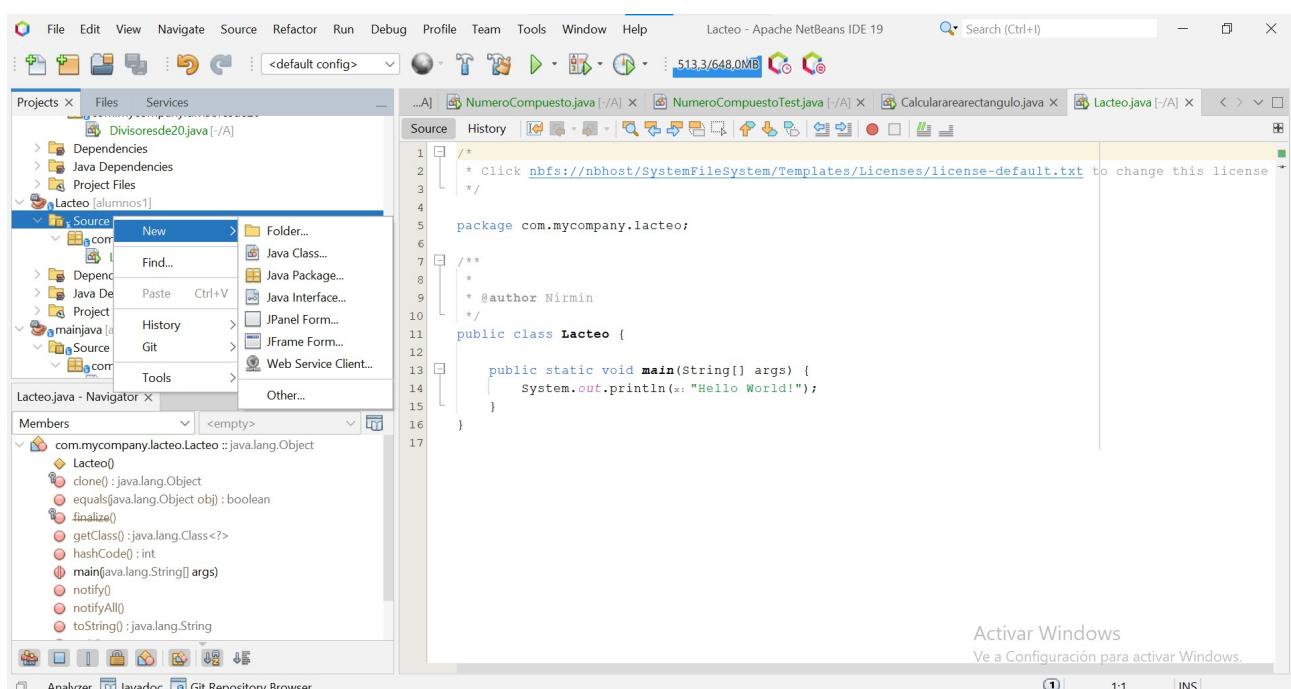
1.2.Crear una nueva clase llamada “Queso” que contendrá la información referente a la fabricación de un queso. La clase contendrá las siguientes variables:

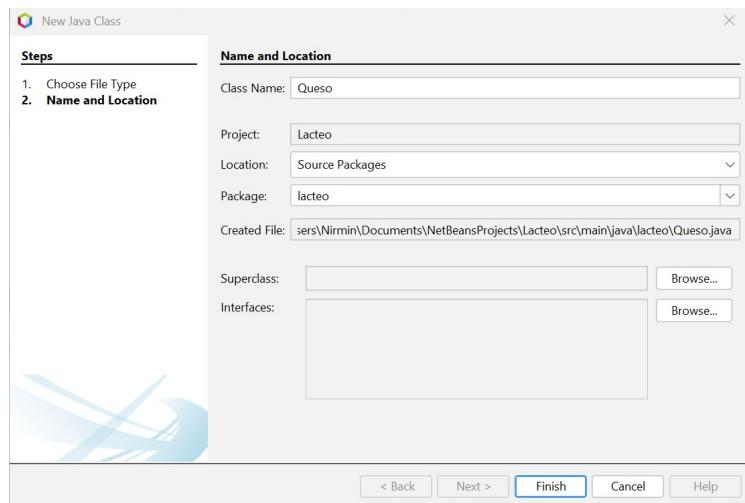
- tipo_leche: indicará el tipo de leche con que se ha fabricado el queso.
- cantidad_leche: indicará la cantidad, en centrilitros, utilizada para su fabricación.
- peso: peso en gramos del queso

El código para la clase será el siguiente:

```
public class Queso {
    String tipo_leche;
    int cantidad_leche;
    double peso;
}
```

Creamos la nueva clase Queso, pulsando dentro de la carpeta donde queremos que se cree el botón derecho y Java Class





Copiamos el código dentro de la clase de Queso.

```

1 package lacteo;
2
3 /**
4 * @author Nirmin
5 */
6 public class Queso {
7     String tipo_leche;
8     int cantidad_leche;
9     double peso;
10}
11
12
13
14

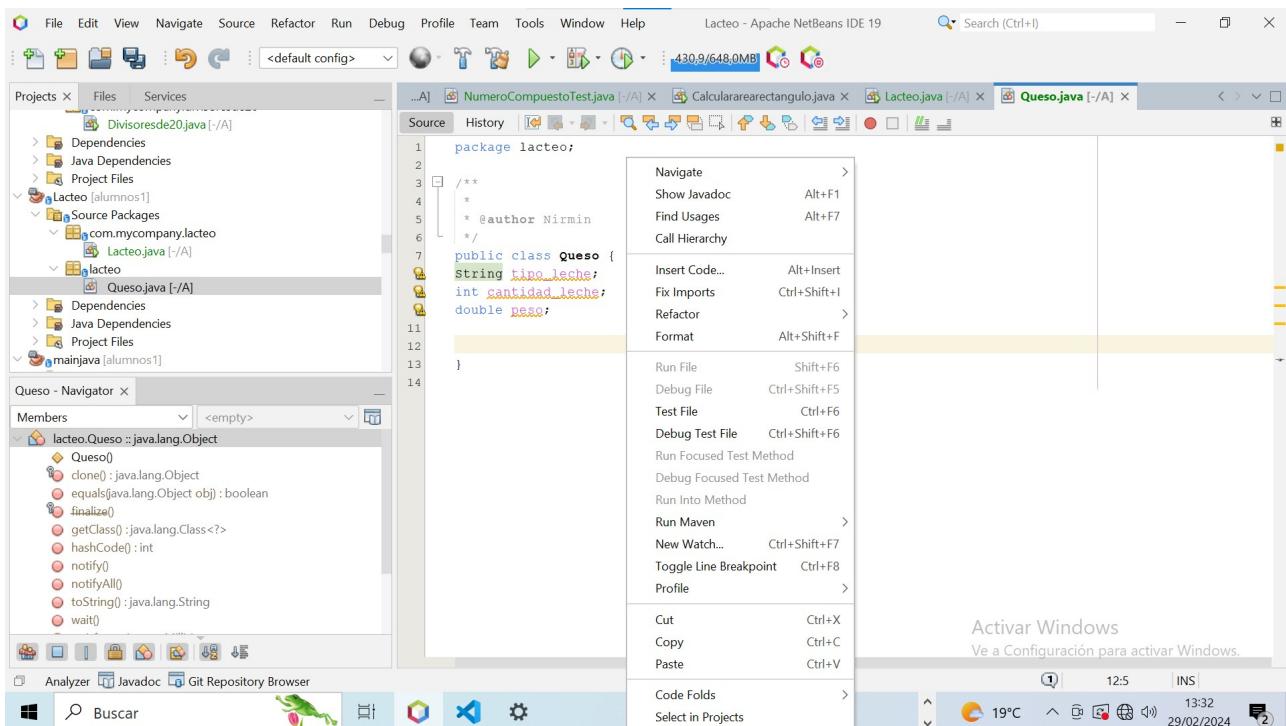
```

The Navigator panel on the left lists the members of the Queso class:

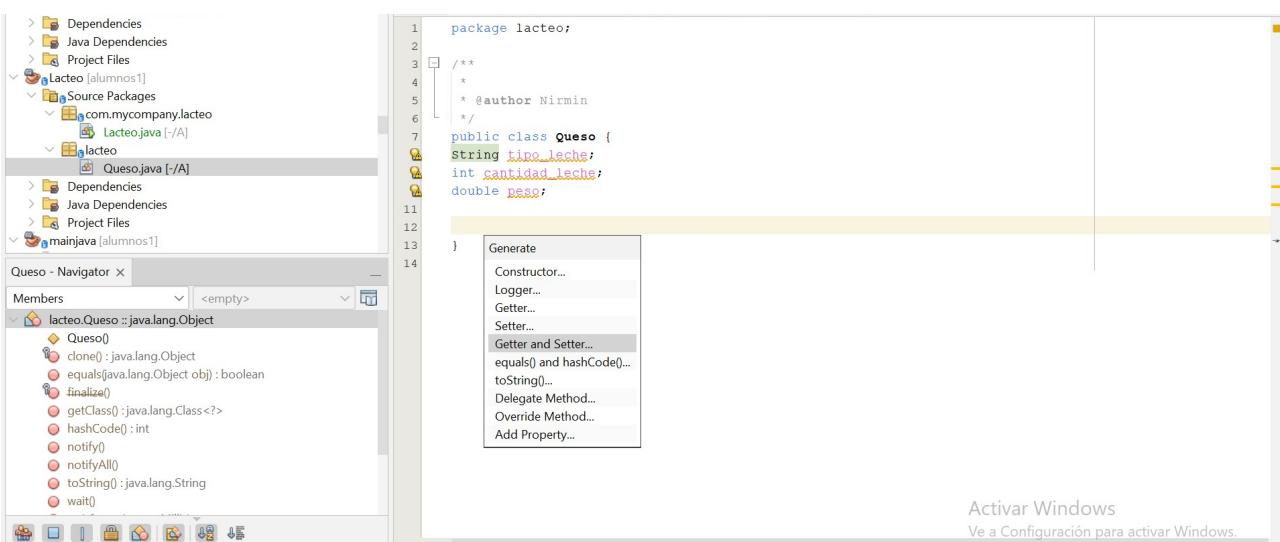
- getClass():java.lang.Class<?>
- hashCode():int
- notify()
- notifyAll()
- toString():java.lang.String
- wait()
- wait(long timeoutMillis)
- wait(long timeoutMillis, int nanos)
- cantidad_leche : int
- peso : double
- tipo_leche :java.lang.String

1.3.Crear dos métodos constructores (uno sin parámetros y otro con todos los parámetros), además de los métodos getter & setter para todas las variables de clase. Para esta tarea se debe usar la generación de código automática.

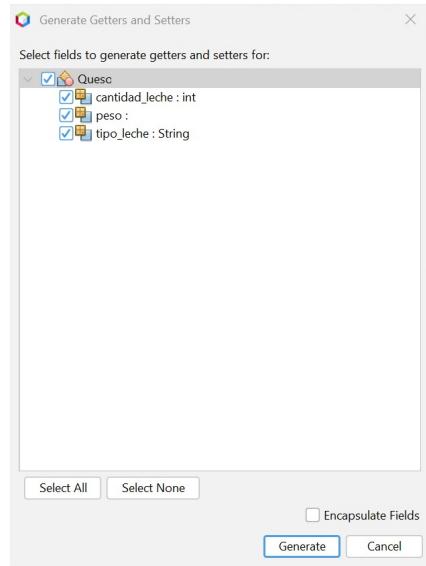
Pulsamos botón derecho dentro del proyecto y damos “insert code...”



Para los getter y setter seleccionamos getter y setter



Seleccionamos todos los atributos que queremos que nos genere getter y setter



Pulsamos generate y nos creará todos los getter y setter

```

1 package lacteo;
2 public class Queso {
3     String tipo_leche;
4     int cantidad_leche;
5     double peso;
6
7     public String getTipo_leche() {
8         return tipo_leche;
9     }
10
11    public void setTipo_leche(String tipo_leche) {
12        this.tipo_leche = tipo_leche;
13    }
14
15    public int getCantidad_leche() {
16        return cantidad_leche;
17    }
18
19    public void setCantidad_leche(int cantidad_leche) {
20        this.cantidad_leche = cantidad_leche;
21    }
22
23    public double getPeso() {
24        return peso;
25    }
26
27    public void setPeso(double peso) {
28        this.peso = peso;
29    }
}

```

Repetimos el mismo proceso con el constructor

The screenshot shows an IDE interface with a code editor and a project navigation pane. The code editor displays a Java class named `Queso` with the following code:

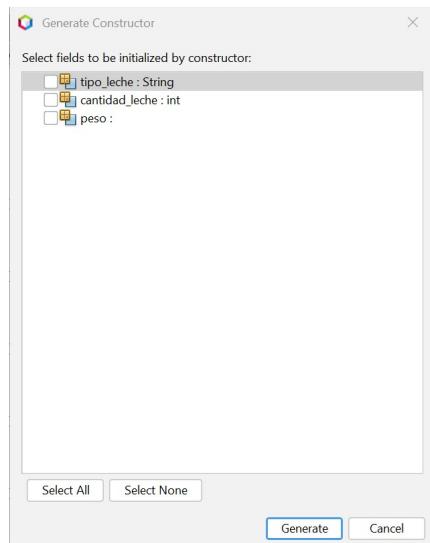
```

1 package lacteo;
2 public class Queso {
3     String tipo_leche;
4     int cantidad_leche;
5     double peso;
6
7     // Constructor...
8     public Queso(String tipo_leche) {
9         this.tipo_leche = tipo_leche;
10    }
11
12    // Getters and setters...
13
14    public String getTipo_leche() {
15        return tipo_leche;
16    }
17
18    public void setCantidad_leche(int cantidad_leche) {
19        this.cantidad_leche = cantidad_leche;
20    }
21
22    public double getPeso() {
23        return peso;
24    }
25
26    public void setPeso(double peso) {
27        this.peso = peso;
28    }
29

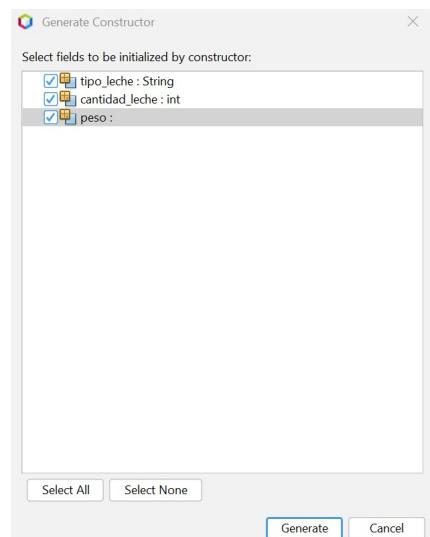
```

A context menu is open over the constructor line, with "Constructor..." highlighted. The menu also includes options like "Generate", "Logger...", "equals() and hashCode()", "toString()", "Delegate Method...", "Override Method...", and "Add Property...".

Damos select none para generar el constructor sin parámetros



Repetimos el proceso. Insert code... > constructor > select all...



Y ya tendremos todos los métodos necesarios

The screenshot shows the Java code for the `Queso` class in the `lacteo` package. The code includes constructor, getters, and setters for `tipo_leche`, `cantidad_leche`, and `peso`. The `imprimir` method is currently highlighted in the code editor. The Navigator pane on the left lists the members of the `Queso` class, including the constructor and the three methods. A status bar at the bottom right indicates "Activar Windows" and "Ve a Configuración para activar Windows".

```
1 package lacteo;
2 public class Queso {
3     String tipo_leche;
4     int cantidad_leche;
5     double peso;
6
7     public Queso(String tipo_leche, int cantidad_leche, double peso) {
8         this.tipo_leche = tipo_leche;
9         this.cantidad_leche = cantidad_leche;
10        this.peso = peso;
11    }
12    public Queso() {
13    }
14
15    public String getTipo_leche() {
16        return tipo_leche;
17    }
18
19    public void setTipo_leche(String tipo_leche) {
20        this.tipo_leche = tipo_leche;
21    }
22
23    public int getCantidad_leche() {
24        return cantidad_leche;
25    }
26
27    public void setCantidad_leche(int cantidad_leche) {
28        this.cantidad_leche = cantidad_leche;
29    }
30
31
32
33
34
35
36
37
38
39    public void imprimir () {
40        System.out.println("QUESERÍA ARTESANA TALAVERA DE LA REINA");
41        System.out.println("Registro Sanitario Nº 48/38751");
42        System.out.println("Para consultar el lote del producto revise la etiqueta");
43        System.out.println("Peso: " + this.peso);
44        System.out.println("Tipo de leche: " + this.tipo_leche);
45        System.out.println("Cantidad de leche: " + this.cantidad_leche);
46    }
47}
```

1.4. Crear un método llamado imprimir que muestre los datos de un objeto de esta clase. El código fuente para este método será el siguiente:

```
public void imprimir () { System.out.println("QUESERÍA ARTESANA TALAVERA DE LA REINA"); System.out.println("Registro Sanitario Nº 48/38751"); System.out.println("Para consultar el lote del producto revise la etiqueta"); System.out.println("Peso: " + this.peso); System.out.println("Tipo de leche: " + this.tipo_leche); System.out.println("Cantidad de leche: " + this.cantidad_leche); }
```

Creamos el método imprimir

The screenshot shows the Java code for the `Queso` class in the `lacteo` package. The `imprimir` method has been added to the end of the class definition. The `Navigator` pane on the left now includes the `imprimir` method in the list of members. A status bar at the bottom right indicates "Activar Windows" and "Ve a Configuración para activar Windows".

```
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39    public void imprimir () {
40        System.out.println("QUESERÍA ARTESANA TALAVERA DE LA REINA");
41        System.out.println("Registro Sanitario Nº 48/38751");
42        System.out.println("Para consultar el lote del producto revise la etiqueta");
43        System.out.println("Peso: " + this.peso);
44        System.out.println("Tipo de leche: " + this.tipo_leche);
45        System.out.println("Cantidad de leche: " + this.cantidad_leche);
46    }
47
```

1.5. Generar los comentarios Javadoc para nuestro proyecto, incluyendo comentarios para la clase y todos sus métodos. Comprobar mediante la herramienta “Analyze Javadoc” que no tenemos ningún método sin documentar y ningún error en la documentación.

Para crear el JavaDoc hay que poner `/**` y pulsar tabulación para que se cree automáticamente y añadimos el texto necesario en el Javadoc

Screenshot of an IDE showing the class structure and code for `Queso`.

Project Explorer:

- Dependencies
- Java Dependencies
- Project Files
- Lacteo [alumnos1]
 - Source Packages
 - com.mycompany.lacteo
 - Lacteo.java [-/A]
 - lacteo
 - Queso.java [-/A]
 - Dependencies
 - Java Dependencies
 - Project Files
- mainjava [alumnos1]

Navigator:

Members <empty>

lacteo.Queso :: java.lang.Object

- Queso()
- Queso(java.lang.String tipo_leche, int cantidad_leche, double peso)
- clone() : java.lang.Object
- equals(java.lang.Object obj) : boolean
- finalize()
- getCantidad_leche() : int
- getClass() : java.lang.Class<?>
- getPeso() : double
- getTipo_leche() : java.lang.String
- hashCode() : int

Code Editor:

```

2  /**
3   * Clase Queso para el software de gestión de productos lácteos
4   * @author Nirmín
5   */
6
7  public class Queso {
8      String tipo_leche;
9      int cantidad_leche;
10     double peso;
11
12     public Queso(String tipo_leche, int cantidad_leche, double peso) {
13         this.tipo_leche = tipo_leche;
14         this.cantidad_leche = cantidad_leche;
15         this.peso = peso;
16     }
17     public Queso() {
18     }
19     public String getTipo_leche() {
20         return tipo_leche;
21     }
22     public void setTipo_leche(String tipo_leche) {
23         this.tipo_leche = tipo_leche;
24     }
25     public int getCantidad_leche() {
26         return cantidad_leche;
27     }
28
29 }
```

Activar Windows
Ve a Configuración para activar Windows.

Screenshot of an IDE showing the class structure and code for `Queso`.

Project Explorer:

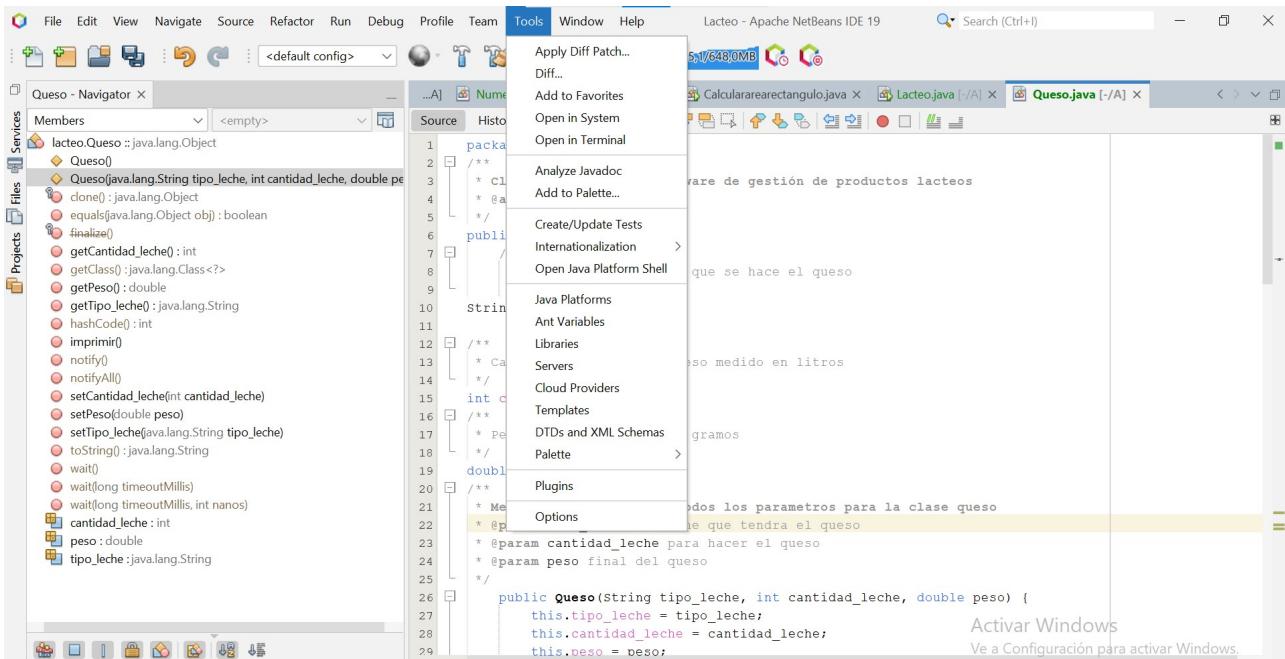
- Projects
- Files
- Serv
- lacteo.Queso :: java.lang.Object
 - Queso()
 - Queso(java.lang.String tipo_leche, int cantidad_leche, double peso)
 - clone() : java.lang.Object
 - equals(java.lang.Object obj) : boolean
 - finalize()
 - getCantidad_leche() : int
 - getClass() : java.lang.Class<?>
 - getPeso() : double
 - getTipo_leche() : java.lang.String
 - hashCode() : int
 - imprimir()
 - notify()
 - notifyAll()
 - setCantidad_leche(int cantidad_leche)
 - setPeso(double peso)
 - setTipo_leche(java.lang.String tipo_leche)
 - toString() : java.lang.String
 - wait()
 - wait(long timeoutMillis)
 - wait(long timeoutMillis, int nanos)
 - cantidad_leche : int
 - peso : double
 - tipo_leche : java.lang.String

Code Editor:

```

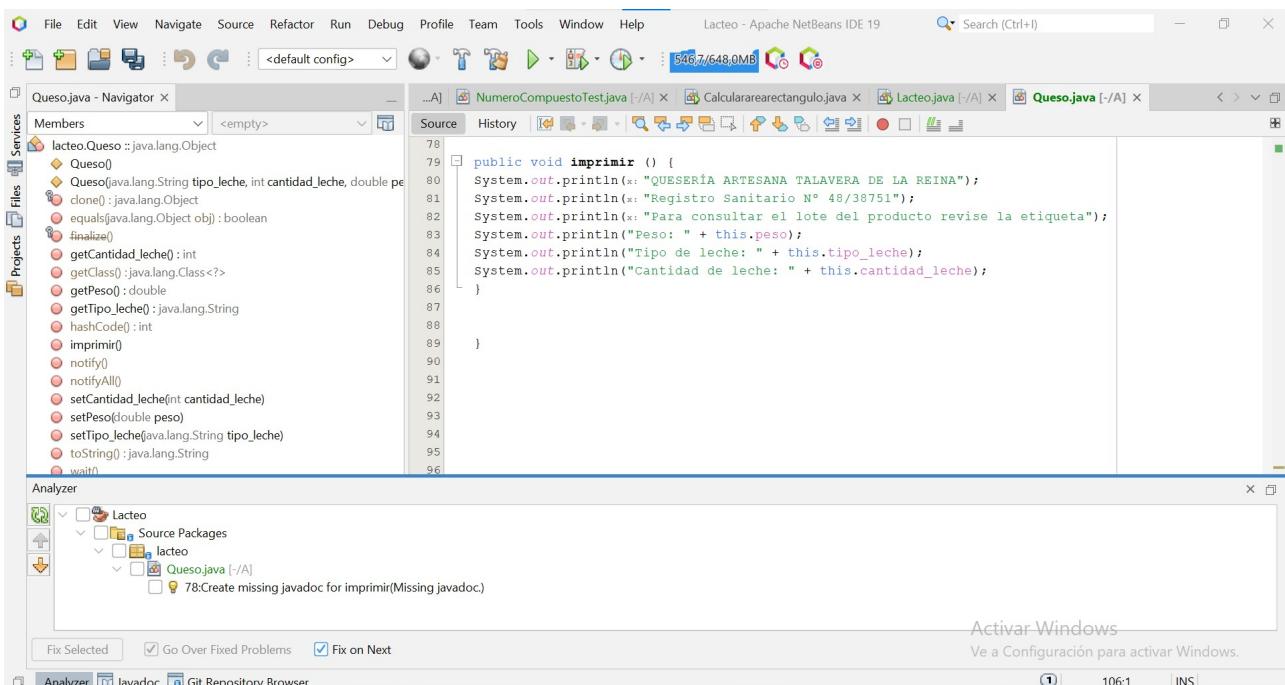
1 package lacteo;
2 /**
3  * Clase Queso para el software de gestión de productos lácteos
4  * @author Nirmín
5  */
6
7  public class Queso {
8
9      /**
10      * Tipo de leche con la que se hace el queso
11      */
12     String tipo_leche;
13
14     /**
15      * Cantidad de leche del queso medida en litros
16      */
17     int cantidad_leche;
18
19     /**
20      * Peso del queso medida en gramos
21      */
22     double peso;
23
24     /**
25      * Método constructor con todos los parámetros para la clase queso
26      * @param tipo_leche de leche que tendrá el queso
27      * @param cantidad_leche para hacer el queso
28      * @param peso final del queso
29     */
30     public Queso(String tipo_leche, int cantidad_leche, double peso) {
31         this.tipo_leche = tipo_leche;
32         this.cantidad_leche = cantidad_leche;
33         this.peso = peso;
34     }
35
36 }
```

Activar Windows
Ve a Configuración para activar Windows.



Para comprobar que el JavaDoc esta correcto usamos Analyze Javadoc pulsando botón derecho sobre la clase Queso > Tools > Analyze Javadoc

En el caso de que nos faltase alguno se nos indicaría de la siguiente forma.



Repetimos el análisis añadiendo el javadoc que falta en el método imprimir()

```

    /*
     * @param peso Nuevo peso la leche
     */
    public void setPeso(double peso) {
        this.peso = peso;
    }

    /**
     * Metodo que nos permite imprimir la etiqueta con los datos de fabricacion
     * del queso
     */
    public void imprimir () {
        System.out.println("QUESERIA ARTESANA TALAVERA DE LA REINA");
        System.out.println("Registro Sanitario N° 48/38751");
        System.out.println("Para consultar el lote del producto revise la etiqueta");
        System.out.println("Peso: " + this.peso);
        System.out.println("Tipo de leche: " + this.tipo_leche);
        System.out.println("Cantidad de leche: " + this.cantidad_leche);
    }
}

```

Analyzer

Fix Selected Go Over Fixed Problems Fix on Next

Activar Windows
Ve a Configuración para activar Windows.

Analyzer Javadoc Git Repository Browser 73:5 INS

1.6. Una vez comprobado que nuestra documentación está preparada (sin errores en la salida del Analyzer), generar la documentación Javadoc para el proyecto.

Para generar el javadoc buscamos javadoc y pulsamos sobre la opción Generate Javadoc

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Lacteo - Apache NetBeans IDE 19 Search (Ctrl+I)

imprimir - Navigator X Members <empty>

lacteo.Queso : java.lang.Object

Queso()

Queso(java.lang.String tipo_leche, int cantidad_leche, double peso)

clone() : java.lang.Object

equals(java.lang.Object obj) : boolean

finalize()

getCantidad_leche() : int

getClass() : java.lang.Class<?>

getPeso() : double

getTipo_leche() : java.lang.String

hashCode() : int

imprimir()

notify()

notifyAll()

setCantidad_leche(int cantidad_leche)

setPeso(double peso)

setTipo_leche(java.lang.String tipo_leche)

toString() : java.lang.String

wait()

Analyzer

Run Project (Lacteo) F6 Test Project (Lacteo) Alt+F6

Build Project (Lacteo) F11 Clean and Build Project (Lacteo) Shift+F11

Set Project Configuration >

Set Project Browser >

Set Main Project >

Open Java Shell for Project (Lacteo)

Generate Javadoc (Lacteo)

Run File Shift+F6

Test File Ctrl+F6

Compile File F9

Check File Alt+F9

Validate File Alt+Shift+F9

Repeat Build/Run: Build (Lacteo) Ctrl+F11

Stop Build/Run

Activar Windows
Ve a Configuración para activar Windows.

Analyzer Javadoc Git Repository Browser 81:17 INS

Se iniciará la creación de Javadoc como podemos ver en la barra de carga de abajo

The screenshot shows the Apache NetBeans IDE interface. The top menu includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The status bar at the bottom right says "Activar Windows" and "Ve a Configuración para activar Windows." The Navigator tab on the left lists the members of the 'lacteo.Queso' class, including its methods like 'imprimir()'. The Source tab shows the Java code for the 'imprimir()' method, which prints the product's details. The status bar at the bottom right shows "Activar Windows" and "Ve a Configuración para activar Windows."

Automáticamente nos genera el JavaDoc donde nos explica qué hace cada método

The screenshot shows the JavaDoc index page titled "Index". It lists methods for the "lacteo.Queso" class:

- getCantidad_leche()** - Method in class lacteo.Queso
Obtiene la cantidad de leche
- getPeso()** - Method in class lacteo.Queso
Obtiene el peso de la leche
- getTipo_leche()** - Method in class lacteo.Queso
Obtiene el tipo de leche
- imprimir()** - Method in class lacteo.Queso
Metodo que nos permite imprimir la etiqueta con los datos de fabricacion del queso

The status bar at the bottom right shows "Activar Windows" and "Ve a Configuración para activar Windows."

All Classes and Interfaces

Class	Description
Lacteo	
Queso	Clase Queso para el software de gestión de productos lácteos

Copyright © 2024. All rights reserved.

Activar Windows
Ve a Configuración para activar Windows.

1.7. Inicializar el repositorio Git para nuestro proyecto y generar una primera versión con el comentario “v1 – Creación clase queso”. Aseguraos que se incluyen todos los ficheros del proyecto (hay que posicionarse correctamente para llamar a la inicialización del repositorio).

Para inicializar el repositorio de git pulsamos Temas > Git > Connect

imprimir - Navigator X

Members

- lacteo.Queso :: java.lang.Object
- ↳ Queso()
- ↳ Queso(java.lang.String tipo_leche, int cantidad_leche, double peso)
- ↳ clone() : java.lang.Object
- ↳ equals(java.lang.Object obj) : boolean
- ↳ finalize()
- ↳ getCantidad_leche() : int
- ↳ getClass() : java.lang.Class<?>
- ↳ getPeso() : double
- ↳ getTipo_leche() : java.lang.String
- ↳ hashCode() : int
- ↳ imprimir() : void
- ↳ notify()
- ↳ notifyAll()
- ↳ setCantidad_leche(int cantidad_leche)
- ↳ setPeso(double peso)
- ↳ setTipo_leche(java.lang.String tipo_leche)
- ↳ toString() : java.lang.String
- ↳ wait()

Source

```

73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

```

Activar Windows
Ve a Configuración para activar Windows.

Nos aseguramos de que se crea en el directorio raíz del proyecto. A continuación las clases que se vayan a subir a git aparecerán en verde, en nuestro caso, la clase Queso

```

    /**
     * @param peso Nuevo peso la leche
     */
    public void setPeso(double peso) {
        this.peso = peso;
    }

    /**
     * Metodo que nos permite imprimir la etiqueta con los datos de fabricacion
     * del queso
     */
    public void imprimir() {
        System.out.println("QUESERIA ARTESANA TALAVERA DE LA REINA");
        System.out.println("Registro Sanitario N° 48/38751");
        System.out.println("Para consultar el lote del producto revise la etiqueta");
        System.out.println("Peso: " + this.peso);
        System.out.println("Tipo de leche: " + this.tipo_leche);
        System.out.println("Cantidad de leche: " + this.cantidad_leche);
    }

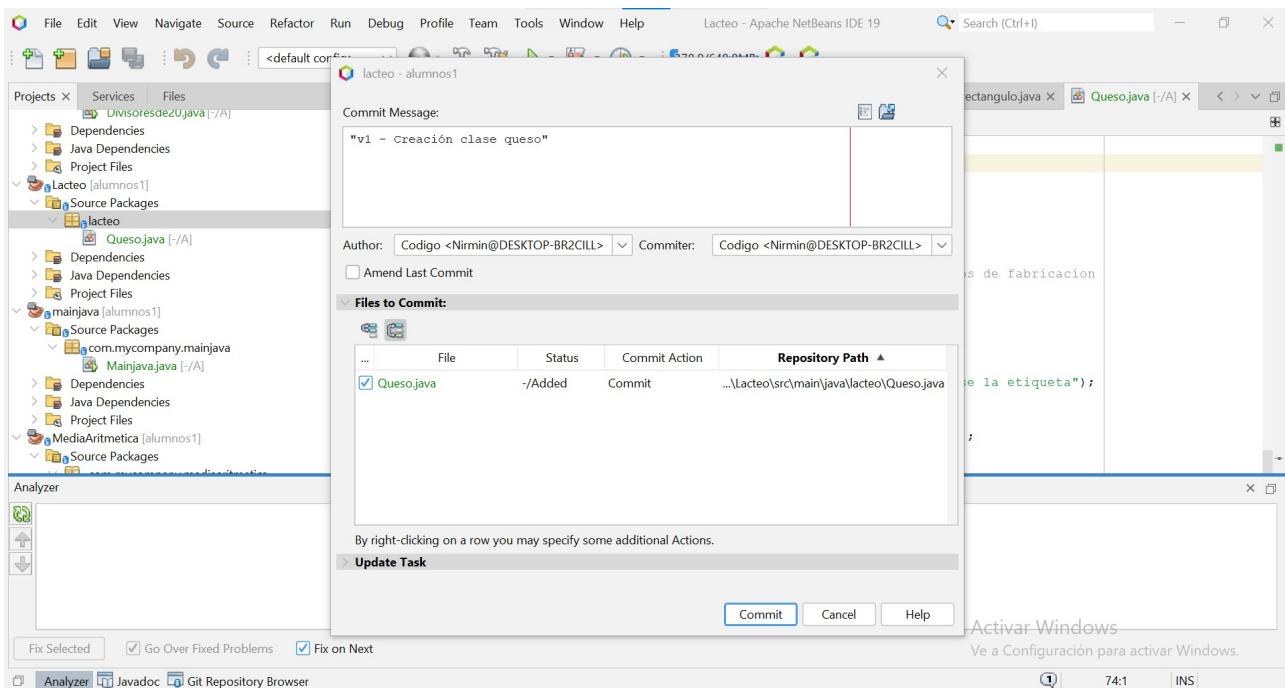
```

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The title bar says "Lacteo - Apache NetBeans IDE 19". The left sidebar displays the project structure under "Projects X Services Files". The main area shows the code editor for "Queso.java". The code defines a class "Queso" with two methods: "setPeso" and "imprimir". The "imprimir" method prints various details about the cheese, including its name, registration number, type, weight, and quantity. Javadoc-style comments are present in the code to describe the methods and their parameters.

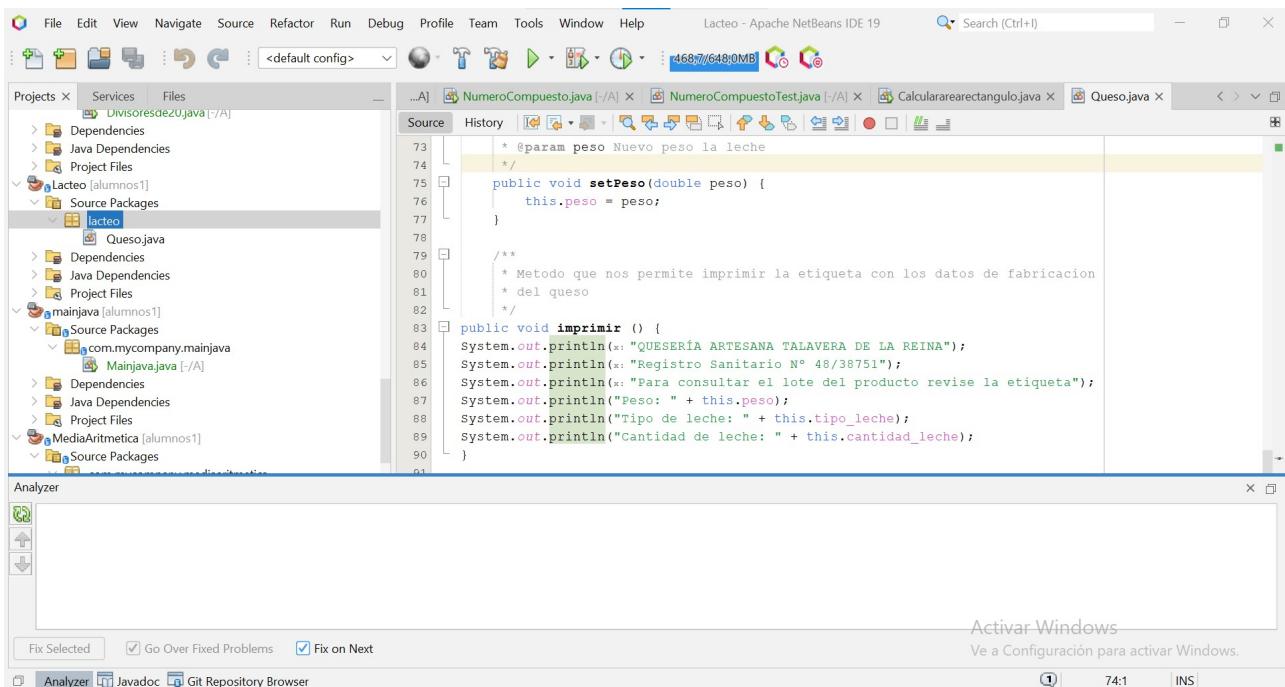
Creamos el primer comentario pulsando Team > Commit

The screenshot shows the Apache NetBeans IDE interface with the "Team" menu open. The "Commit..." option is highlighted. A context menu is displayed over the "Queso.java" file in the code editor. The menu options include Show Changes, Diff, Add, Commit..., Checkout, Revert Modifications..., Show Annotations, Show History, Resolve Conflicts, Ignore, Patches, Branch/Tag, Remote, Revert/Recover, Repository, Shelve Changes, Disconnect..., Other VCS, History, Find Tasks..., Report Task..., and Create Build Job... . The code editor shows the same Java code as the previous screenshot, with the cursor positioned at the start of the "imprimir" method's body.

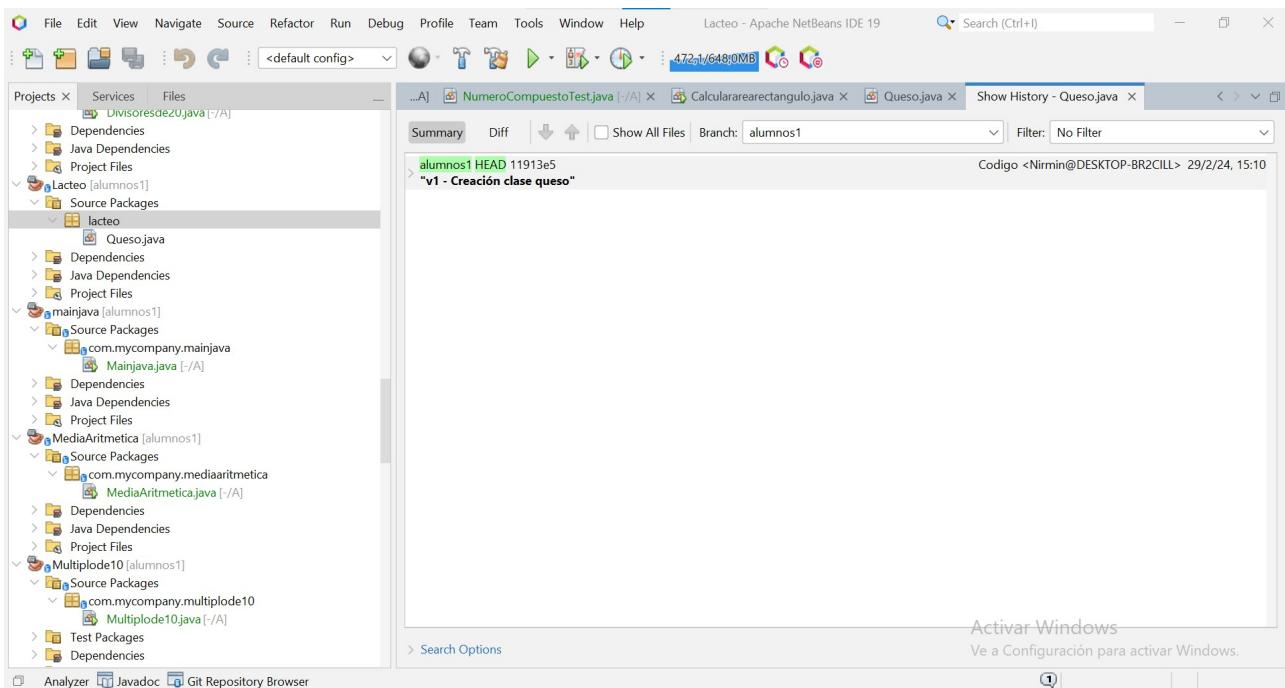
Añadimos el comentario v1- Creación clase queso y pulsamos commit



Cuando cerremos la clase aparecerá color blanco, lo que nos indica que el commit se ha realizado correctamente

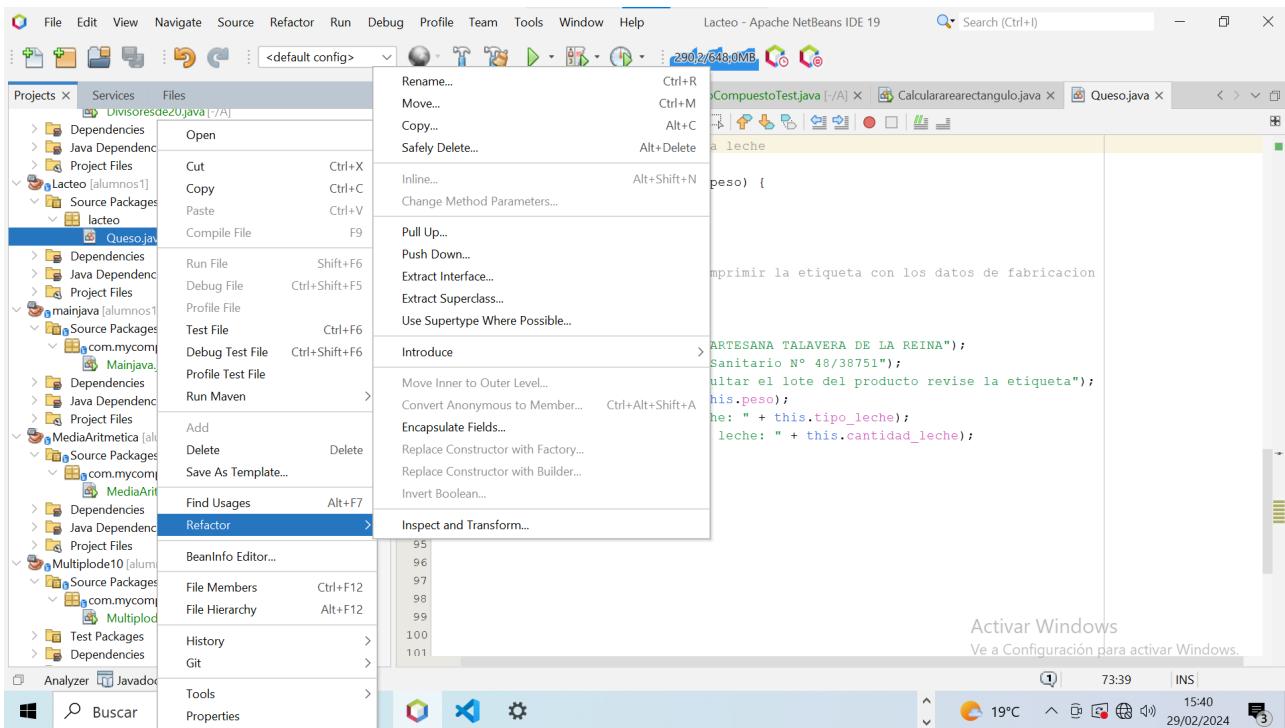


Si pulsamos en team en git history podemos ver que se ha creado el commit

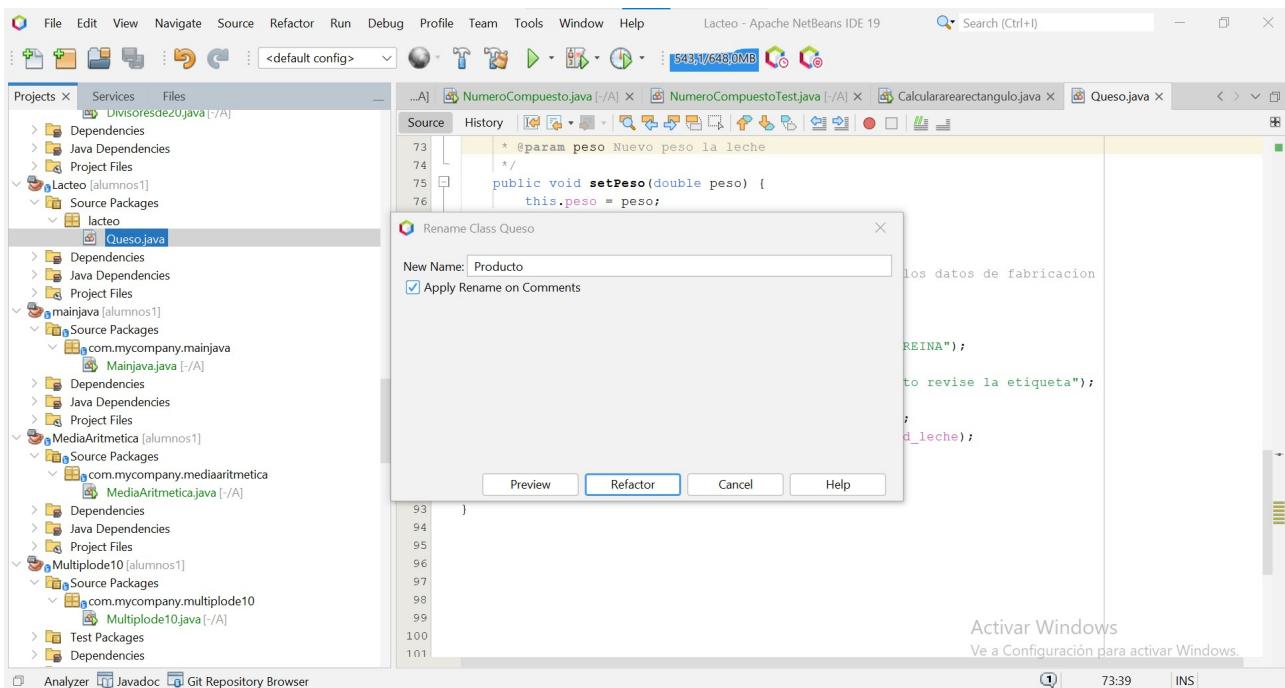


1.8. Cambiar el nombre a la clase Queso, asignando el nuevo nombre Producto. El cambio de nombre debe aplicarse también a los comentarios de la clase, si los hubiese.

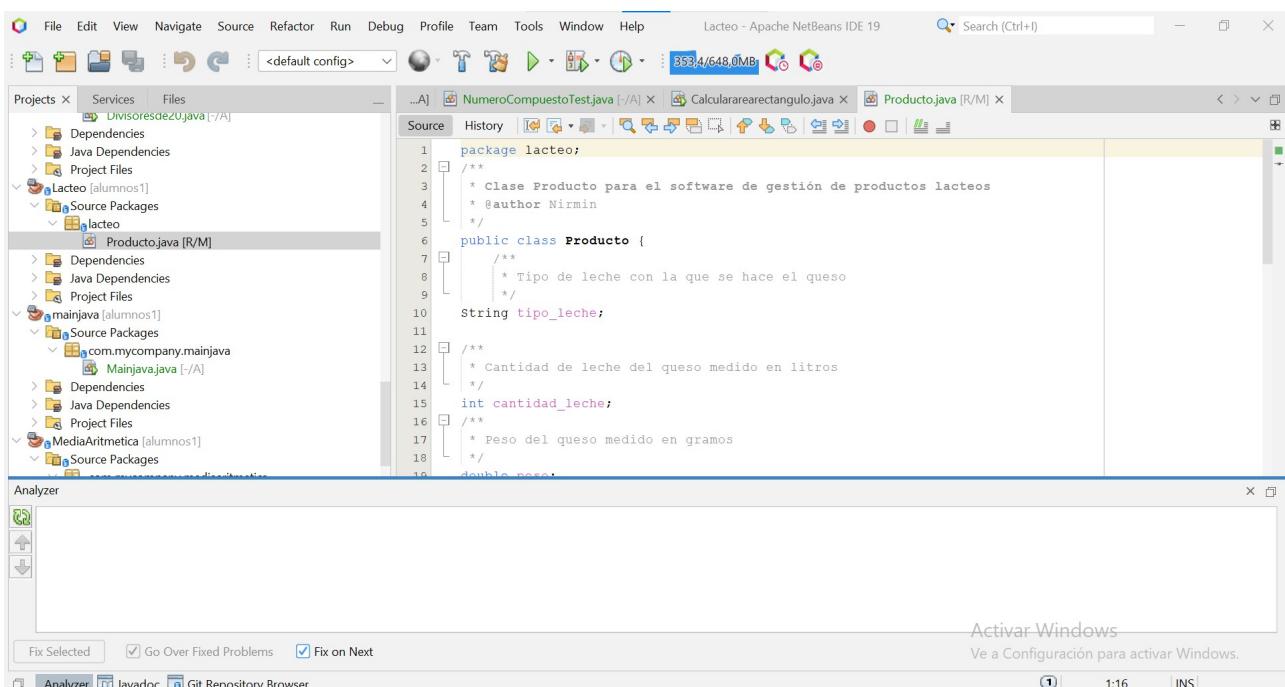
Seleccionamos la clase > botón derecho > Refactor > Rename



Cuando nos salga la ventana tenemos que asegurarnos de que esté marcada la opción Apply rename in comments para que se modifiquen los comentarios también



Cuando damos a Refactor vemos que se ha cambiado el nombre a Producto



1.9.Extraer un método con las líneas que imprimen los datos del objeto Queso (es decir, las tres últimas líneas del método). El nuevo método se llamará imprimir_detalle.

Seleccionamos las líneas de las que queremos extraer el método

```

    /**
     * Metodo que nos permite imprimir la etiqueta con los datos de fabricacion
     * del queso
     */
    public void imprimir () {
        System.out.println("QUESERIA ARTESANA TALAVERA DE LA REINA");
        System.out.println("Registro Sanitario N° 48/38751");
        System.out.println("Para consultar el lote del producto revise la etiqueta");
        System.out.println("Peso: " + this.peso);
        System.out.println("Tipo de leche: " + this.tipo_leche);
        System.out.println("Cantidad de leche: " + this.cantidad_leche);
    }
}

```

Activar Windows
Ve a Configuración para activar Windows.

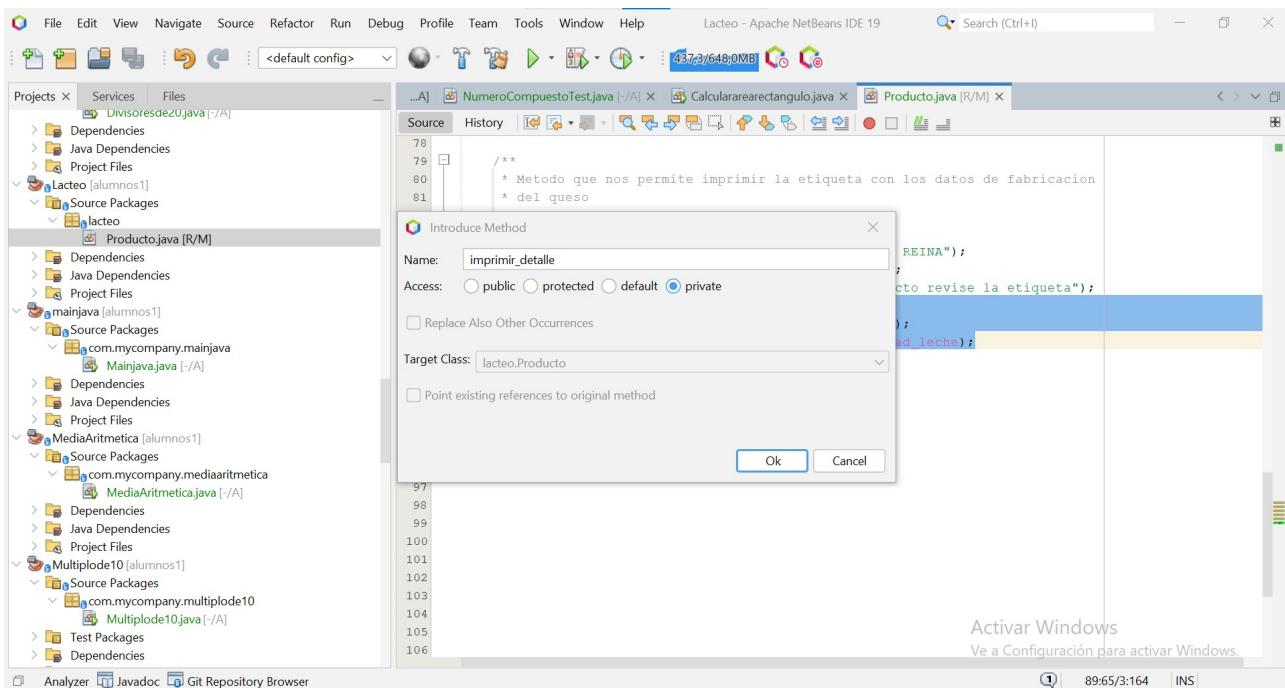
Pulsamos botón derecho > Refactor > Introduce > Method

Right-click context menu for the 'imprimir' method:

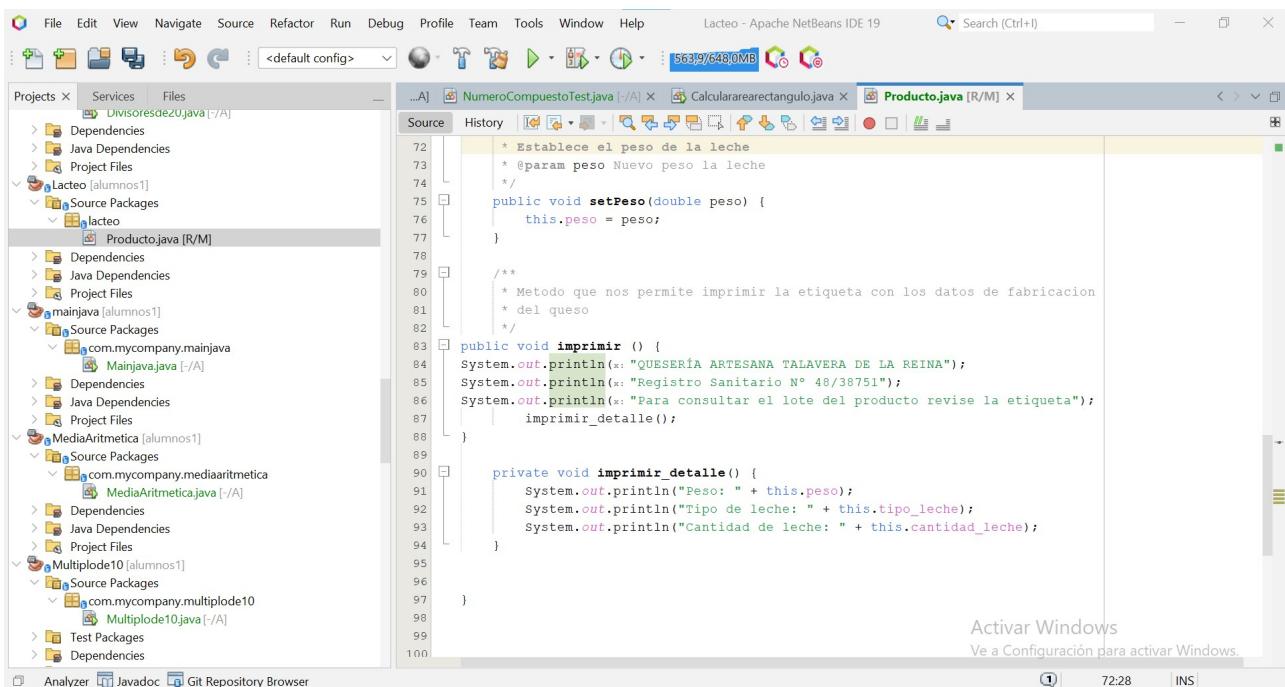
- Navigate
- Show Javadoc
- Find Usages
- Call Hierarchy
- Refactor**
- Format
- Run File
- Debug File
- Test File
- Debug Test File
- Run Focused Test Method
- Debug Focused Test Method
- Run Into Method
- Run Maven
- Variable...
- Constant...
- Field...
- Parameter...
- Method...
- Local Extension...
- Move Inner to Outer Level...
- Convert Anonymous to Member... Ctrl+Alt+Shift+A
- Encapsulate Field...
- Replace Constructor with Factory...
- Replace Constructor with Builder...
- Invert Boolean...
- Inspect and Transform...

Activar Windows
Ve a Configuración para activar Windows.

Ponemos el nombre del método que será “imprimir_detalle”

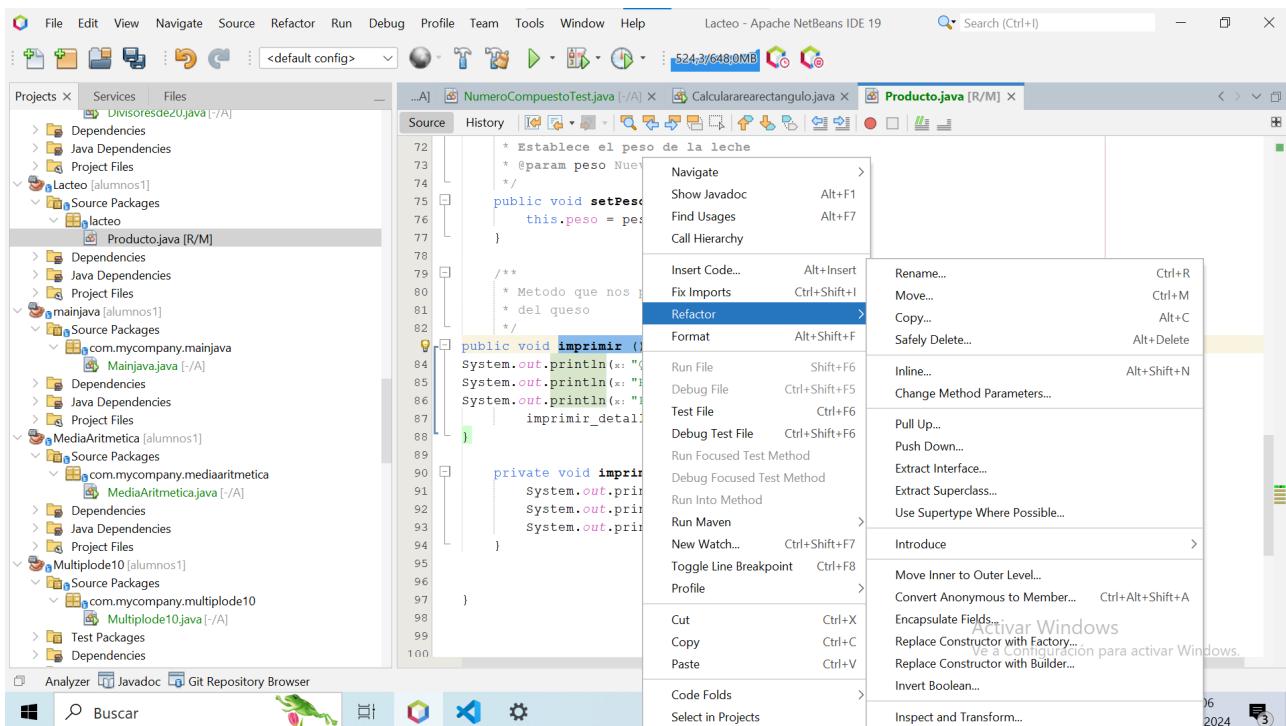


Pulsamos OK y nos generará un nuevo método

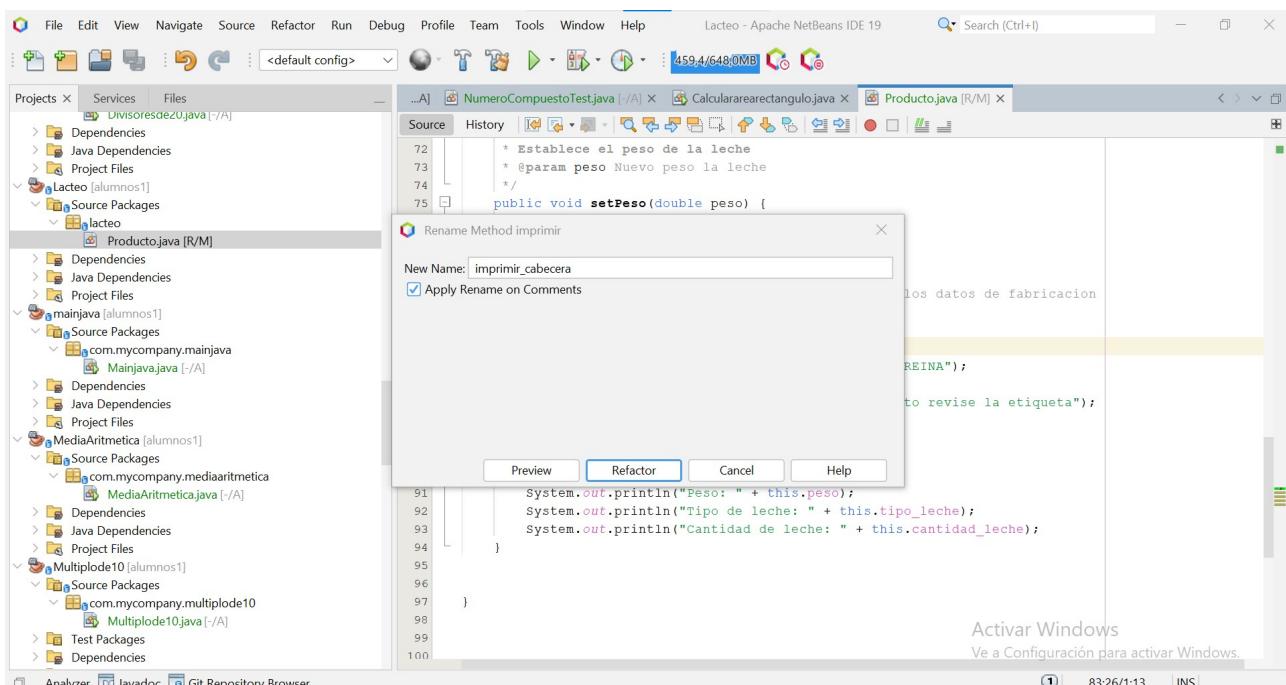


1.10. Cambiar el nombre al método imprimir de la clase Queso y asignarle el nuevo nombre imprimir_cabecera.

Seleccionamos el nombre `imprimir()` del método y seleccionamos botón derecho > Refactor > Rename



Le ponemos como nombre imprimir_cabecera



```

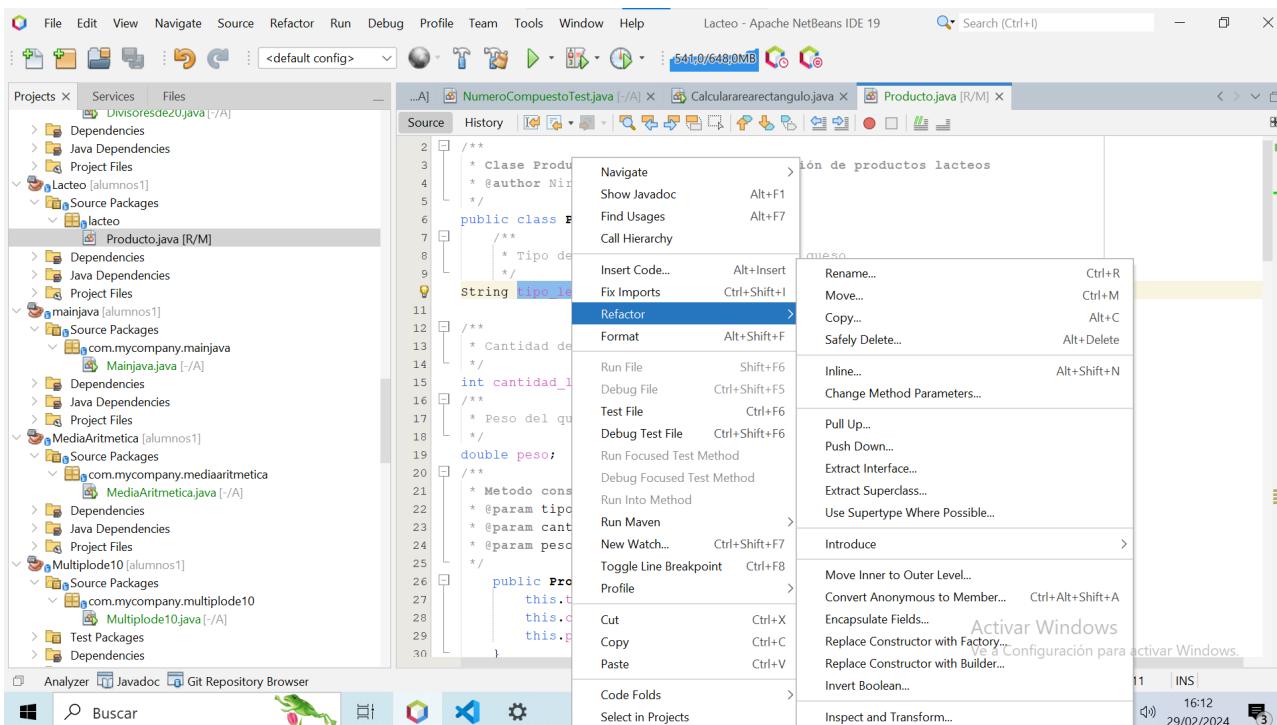
72     * Establece el peso de la leche
73     * @param peso Nuevo peso la leche
74     */
75     public void setPeso(double peso) {
76         this.peso = peso;
77     }
78
79     /**
80      * Metodo que nos permite imprimir_cabecera la etiqueta con los datos de fabricacion
81      * del queso
82     */
83     public void imprimir_cabecera () {
84         System.out.println("QUESERIA ARTESANA TALAVERA DE LA REINA");
85         System.out.println("Registro Sanitario N° 48/38751");
86         System.out.println("Para consultar el lote del producto revise la etiqueta");
87         imprimir_detalle();
88     }
89
90     private void imprimir_detalle() {
91         System.out.println("Peso: " + this.peso);
92         System.out.println("Tipo de leche: " + this.tipo_leche);
93         System.out.println("Cantidad de leche: " + this.cantidad_leche);
94     }
95
96 }
97
98
99
100

```

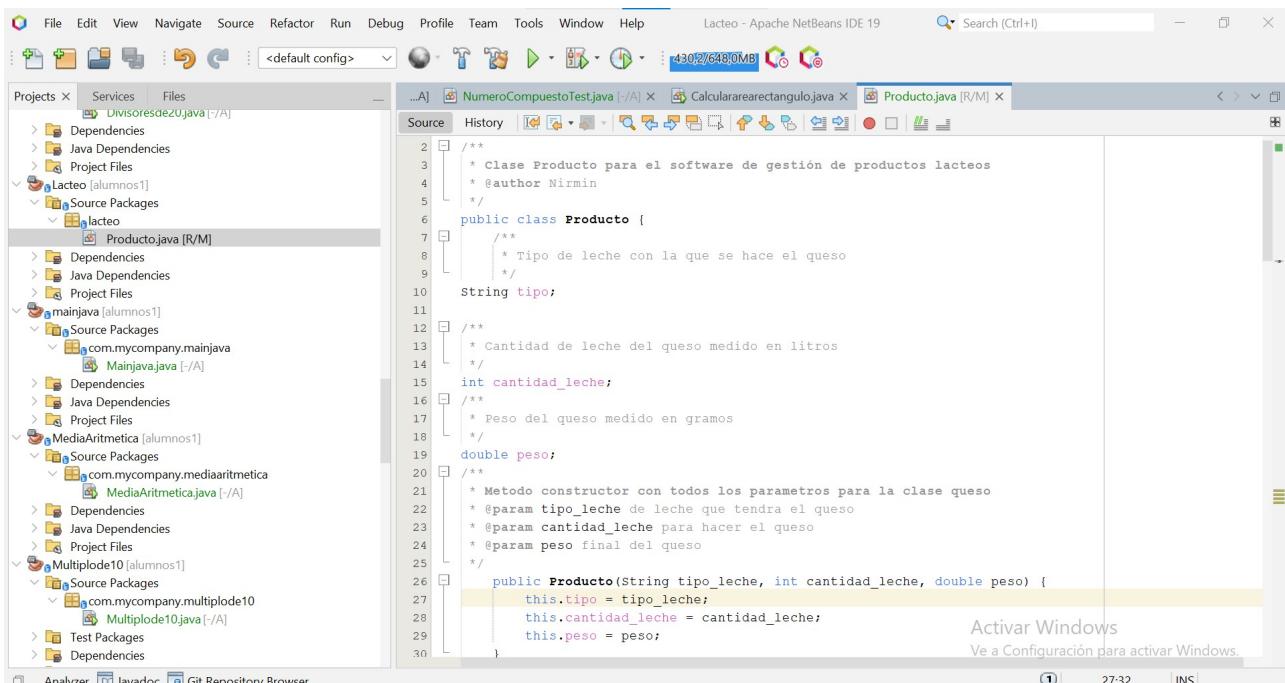
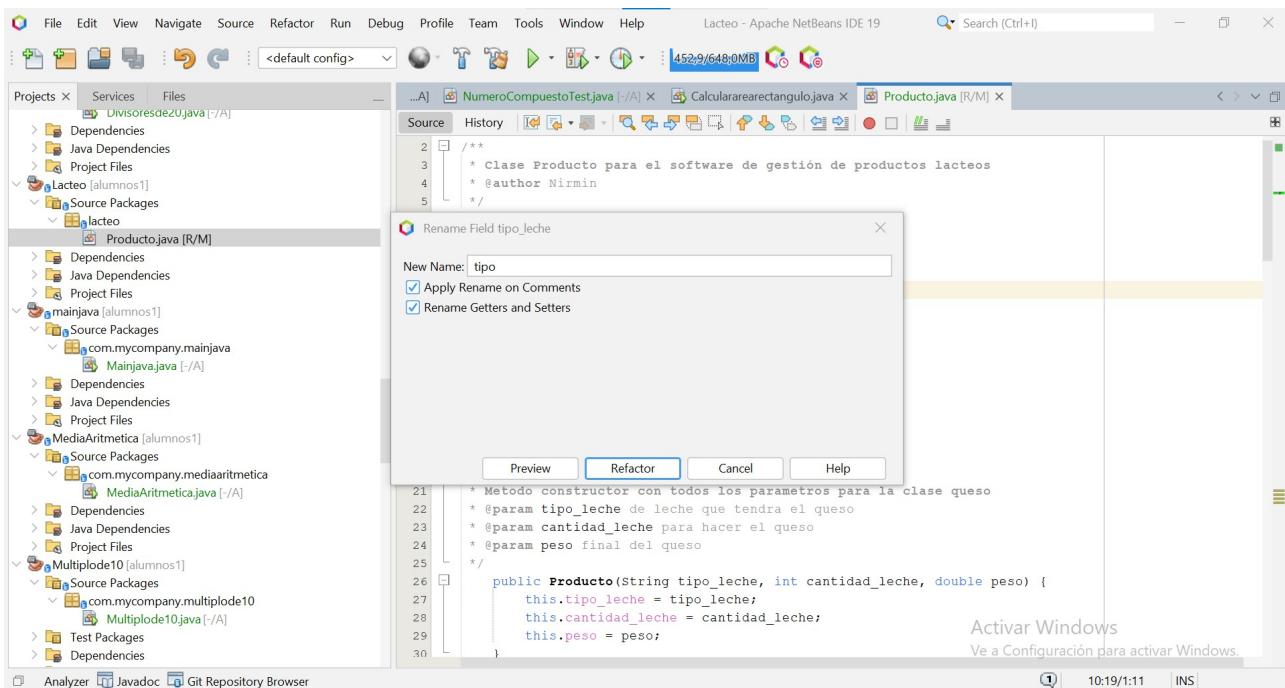
Activar Windows
Ve a Configuración para activar Windows.

1.11. Cambiar el nombre a la variable tipo_leche para que pase a llamarse tipo, de forma que el cambio aplique también a los métodos creados.

Seleccionamos el atributo tipo_leche y pulsamos >Refactor >Rename



Ponemos el nombre tipo y seleccionamos que se modifique también en los comentarios y en los getter y setter



1.12. Encapsular todas las variables de clase de la clase Queso.

Seleccionamos los atributos de la clase Producto que queremos encapsular

```

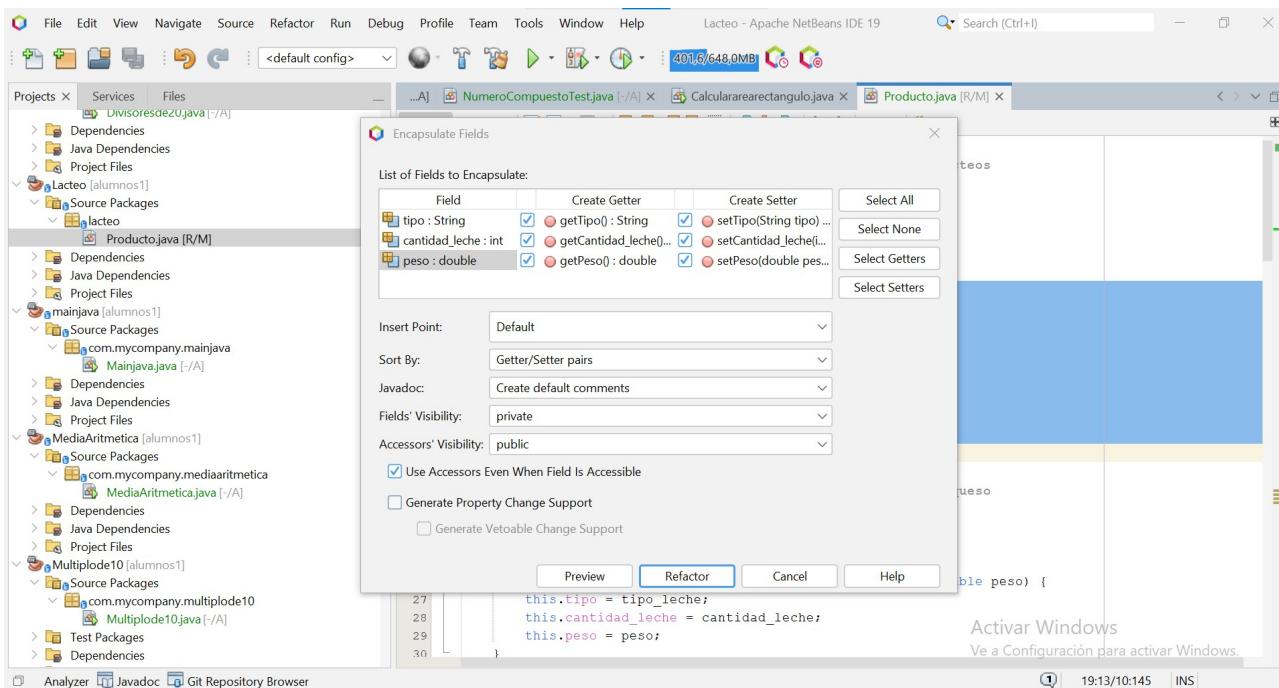
2 /**
3 * Clase Producto para el software de gestión de productos lácteos
4 * @author Nirmi
5 */
6 public class Producto {
7     /**
8      * Tipo de leche con la que se hace el queso
9      */
10    String tipo;
11    /**
12     * Cantidad de leche del queso medida en litros
13     */
14    int cantidad_leche;
15    /**
16     * Peso del queso medida en gramos
17     */
18    double peso;
19
20    /**
21     * Método constructor con todos los parámetros para la clase queso
22     * @param tipo_leche de leche que tendrá el queso
23     * @param cantidad_leche para hacer el queso
24     * @param peso final del queso
25     */
26    public Producto(String tipo_leche, int cantidad_leche, double peso) {
27        this.tipo = tipo_leche;
28        this.cantidad_leche = cantidad_leche;
29        this.peso = peso;
30    }

```

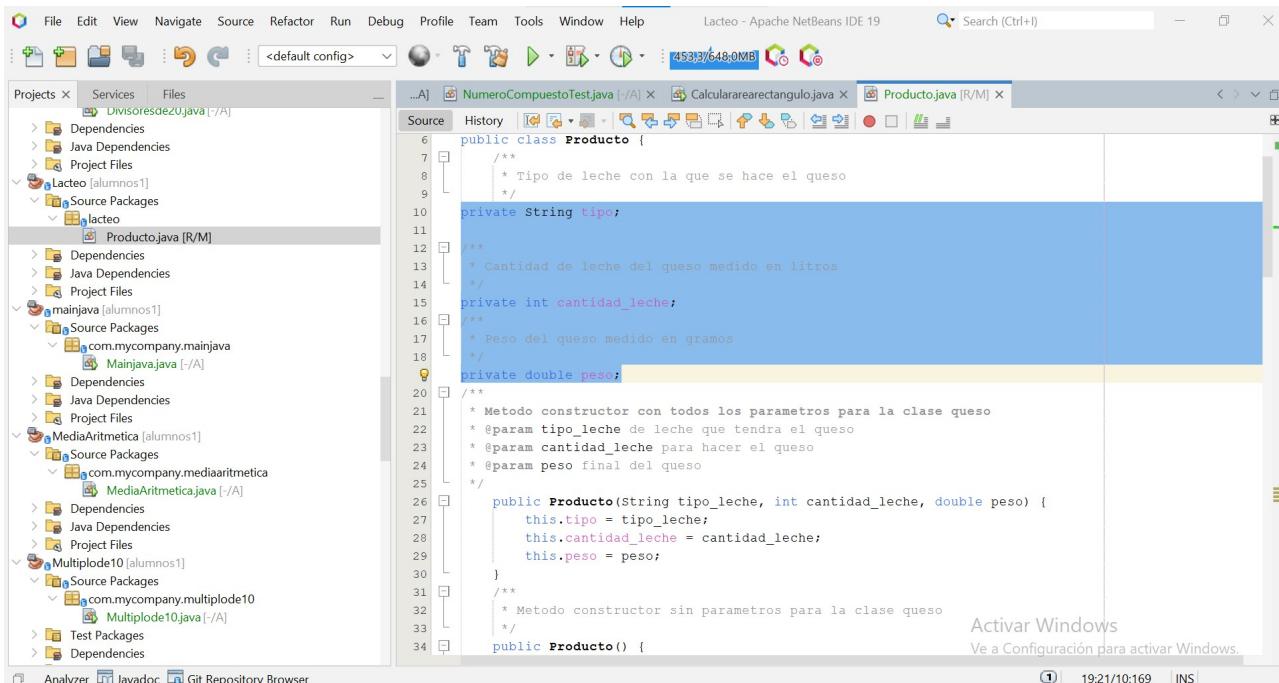
Pulsamos botón derecho > Refactor > Encapsulate Fields

The context menu for the 'peso' field in the code editor shows the 'Refactor' option selected, with a submenu containing various refactoring commands such as Format, Rename, Move, Copy, and Safely Delete.

Se nos abrirá una ventana y comprobamos que los campos “field visibility” estén privados y “accessors visibility” en public y pulsamos refactor

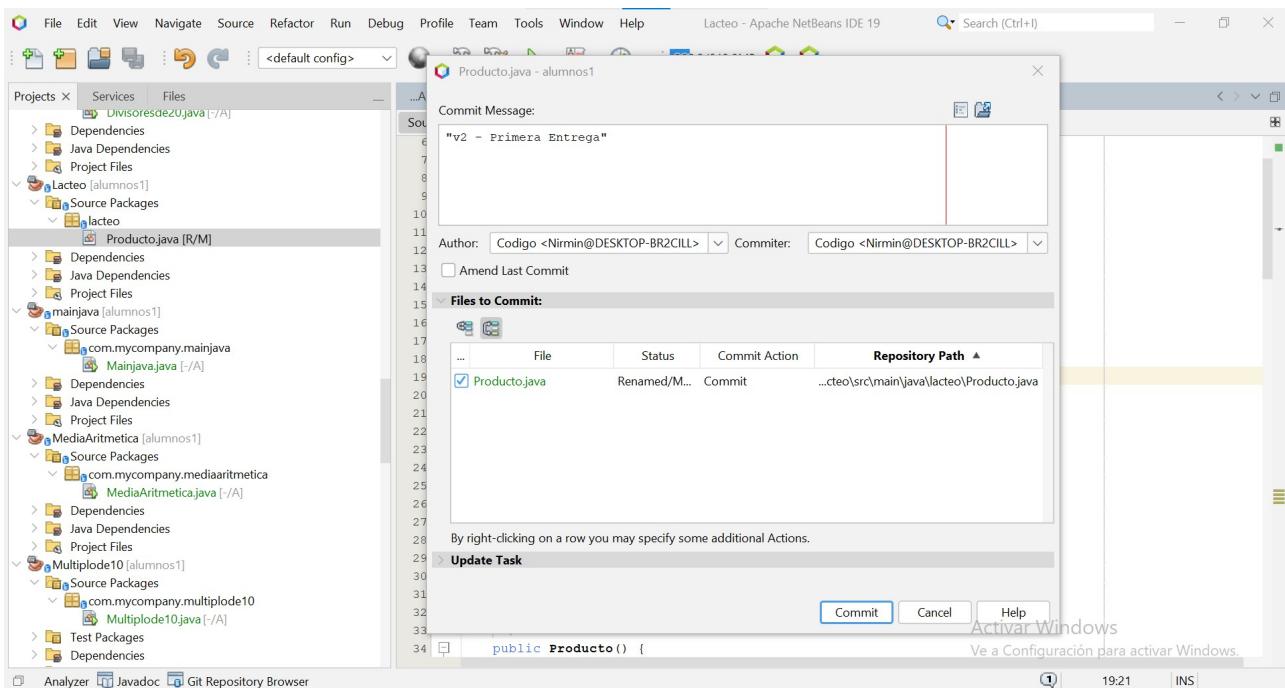


Automáticamente nos añadirá private delante de los atributos

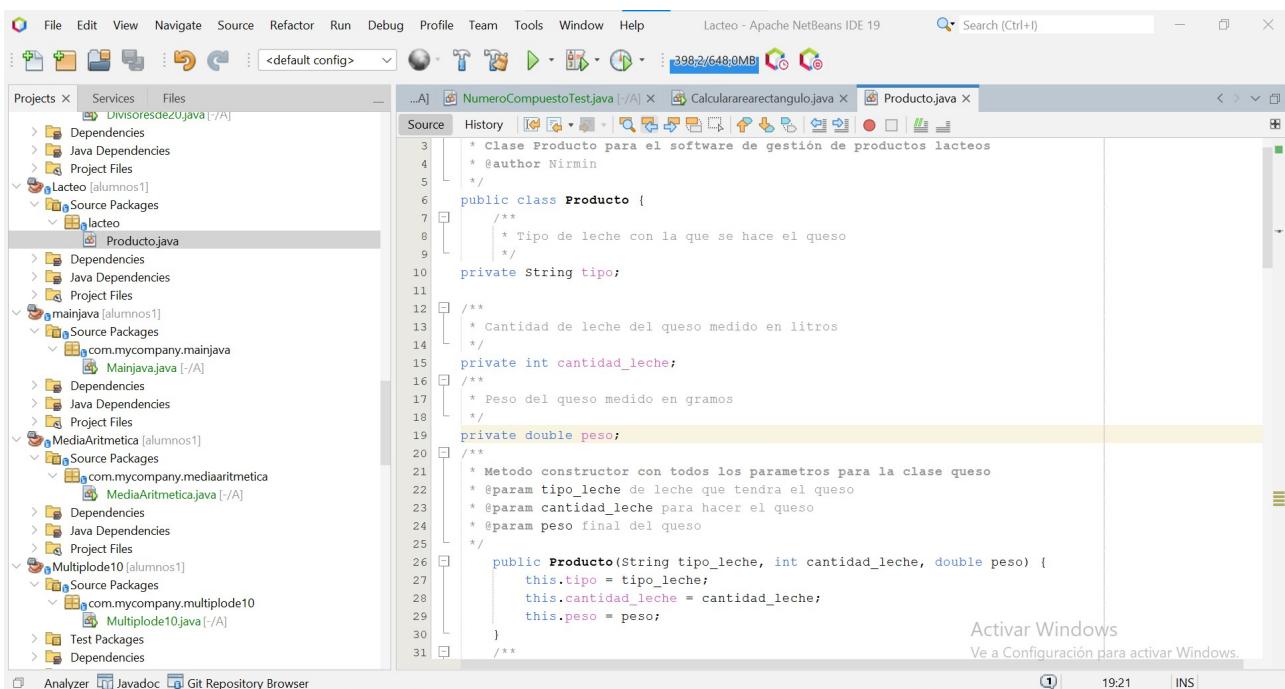


1.13. Generar una segunda versión con el comentario “v2 – Primera Entrega”.

Como existen cambios en la clase que no están registrados en git aparece en verde por lo que repetimos el proceso anterior y comentamos los cambios. Pulsando la pestaña Team > Commit



Los cambios se han guardado correctamente porque ya no aparece la clase Producto en verde



1.14. Mostrar las diferencias de los ficheros entre las versiones v1 y v2

Pulsas sobre Team > Diff y seleccionamos el commit anterior para comparar la versión 1 y 2

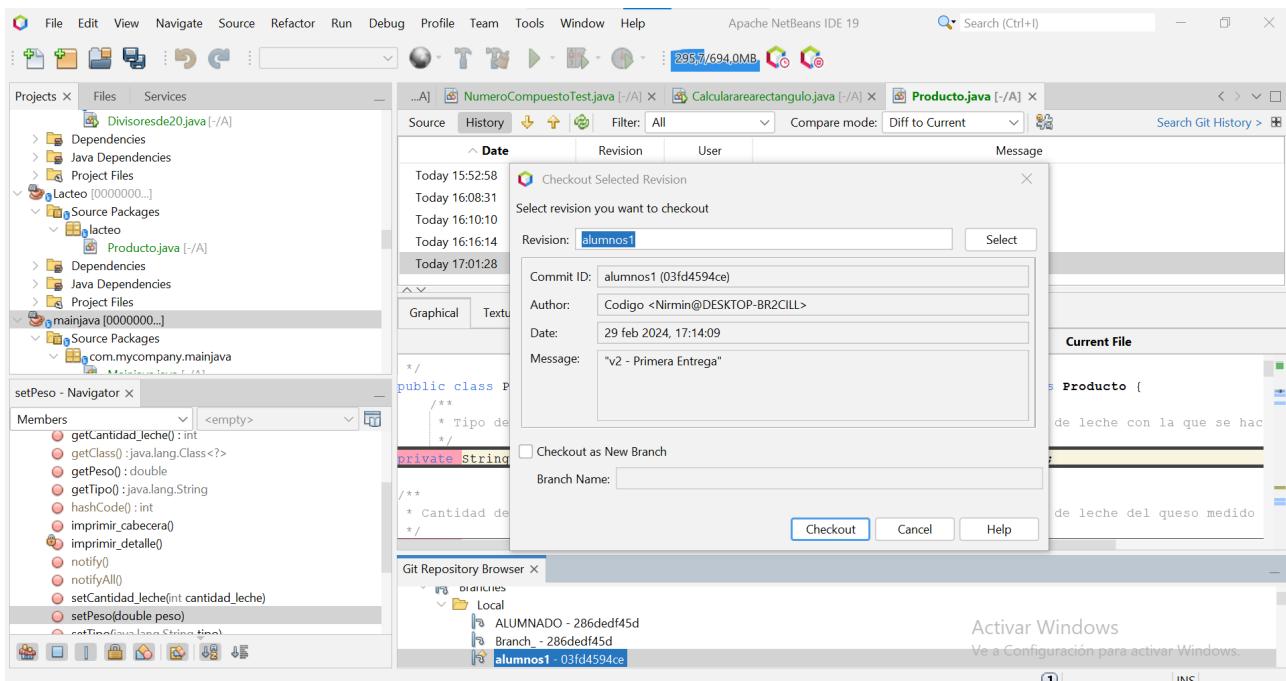
A la izquierda aparece los cambios de la versión 1 donde la clase se llamaba queso y los atributos no estaban encapsulados y si bajamos podemos ver todos los cambios realizados en el proyecto

1.15. Comprobar en el browser de repositorios de Git que la nueva versión ha sido creada.

Cuando pulsamos Team > Show history abajo te aparece el browser de repositorios.

Si abrimos el proyecto Lacteos vemos que el commit “primera entrega” se ha ejecutado correctamente

Abajo cuando comparamos ambos commits vemos el repository browser y si seleccionamos el commit local podemos ver la v2 lo que significa que el commit realizado en la clase Producto se ha realizado correctamente



2. Generar un diagrama de clases UML en Visual Paradigm según la siguiente descripción:

A un taller mecánico los clientes llevan sus vehículos para ser reparados. La primera vez que un cliente acude al taller se registrarán sus datos (DNI, nombre, apellidos y teléfono) y se le asignará un número de cliente del taller.

El taller realiza reparaciones de dos tipos de vehículos: comerciales (coches, motos,

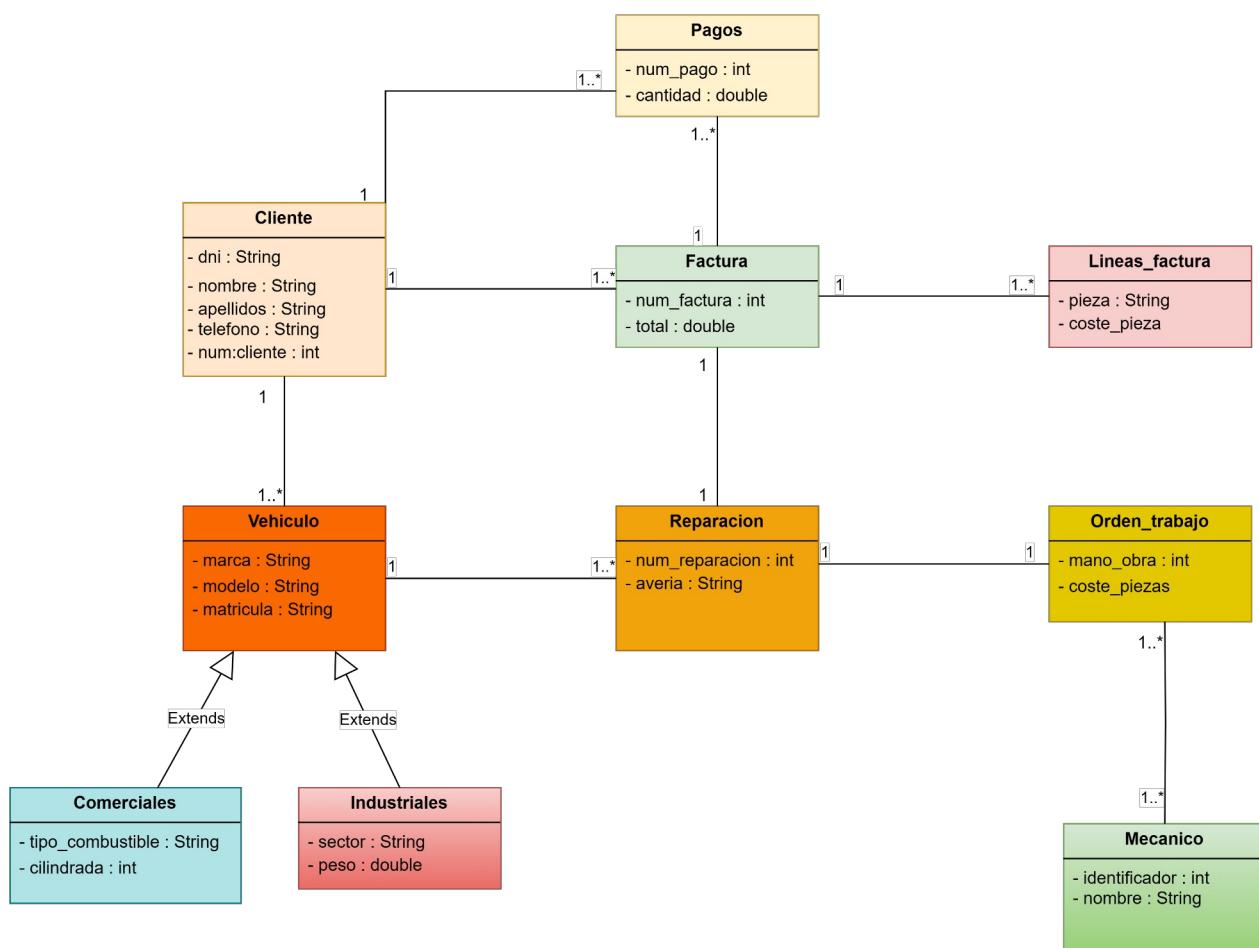
furgonetas, etc.) e industriales (camiones, tractores, etc.).

Cuando un vehículo acude por primera vez al taller se registrará su marca, el modelo y la matrícula. Para los vehículos comerciales se registrará ademas el tipo de combustible (diésel, gasolina, híbrido, eléctrico, etc) y la cilindrada. Y para el caso de los vehículos industriales se registrá el sector al que se dedican y el peso del vehículo.

Para cada reparación se registrará el número de reparación y la descripción de la avería detectada. Además, para cada reparación se generará una orden de trabajo donde se anotarán las horas de mano de obra, el coste de las piezas reparadas o sustituidas y se asignará a un mecánico (o mecánicos en caso de que participe más de uno) que realiza la reparación.

De cada reparación se obtendrá una factura. Cada factura tendrá un número de factura y contendrá una serie de líneas con el detalle de la reparación realizada (con el coste individual de cada uno) y un total que indicará el coste total de la reparación.

A su vez, cada factura tendrá asociados un número de pagos donde se indicará el número de pago y la cantidad a pagar en cada uno de ellos.



Explicación

Un cliente puede tener varios vehículos reparándose a su nombre pero un vehículo solo puede pertenecer a un cliente.

Los vehículos pueden ser de dos clases: comerciales e industriales y como tienen en común las características marca, modelo y matrícula crearemos una clase padre vehículo de la que ambos tipos (industrial y comercial) heredaran sus atributos.

Cuando hay un vehículo en el taller se genera una reparación y a su vez un vehículo puede tener varias reparaciones por lo que se identifica con un número de reparación. Una reparación automáticamente genera una orden de trabajo donde se almacenará la información acerca de dicha reparación concreta, es decir, el tiempo de mano de obra empleado en la reparación, y el coste de las piezas que se han cambiado

Los mecánicos pueden reparar varios vehículos en una jornada de trabajo por lo que pueden tener varias órdenes de trabajo

De cada reparación nace una factura que resume todos los costes de la reparación por ese motivo está formada de varias líneas que identifican las piezas reparadas y el coste de dicha pieza. Esta factura será entregada al cliente para saber a cuánto asciende la reparación y, a su vez esta puede pagarse en varios pagos por lo que se debe especificar cuántos pagos tiene que realizar el cliente y la cuantía de estos