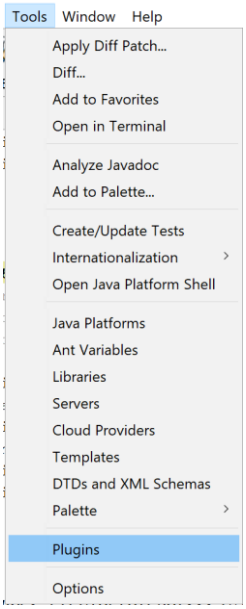
TAREA 04. ENTORNOS DE DESARROLLO 22-23(AYUDA)

Análisis de código

Sabemos que existen **analizadores de código** que se pueden añadir como complementos a los entornos de desarrollo. El analizador estático de código recibirá el código fuente de nuestro programa, lo procesará intentando averiguar la funcionalidad del mismo, y nos dará sugerencias, o nos mostrará posibles mejoras. En esta sección vamos a añadir el pluging ***SonarLint*** a nuestro entorno de desarrollo Netbeans para realizar los apartados que siguen.

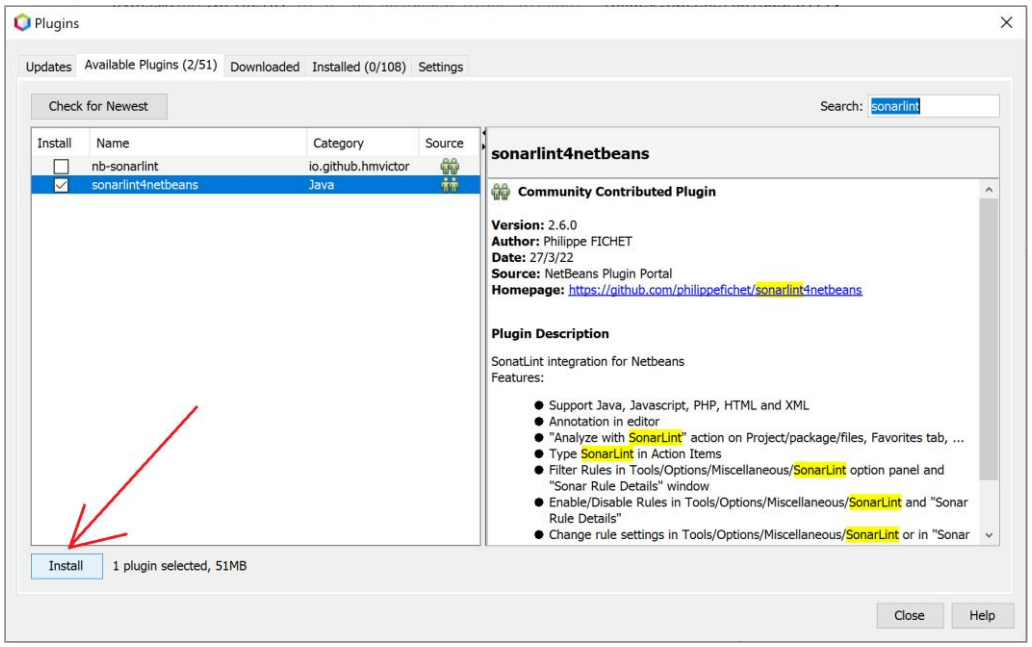


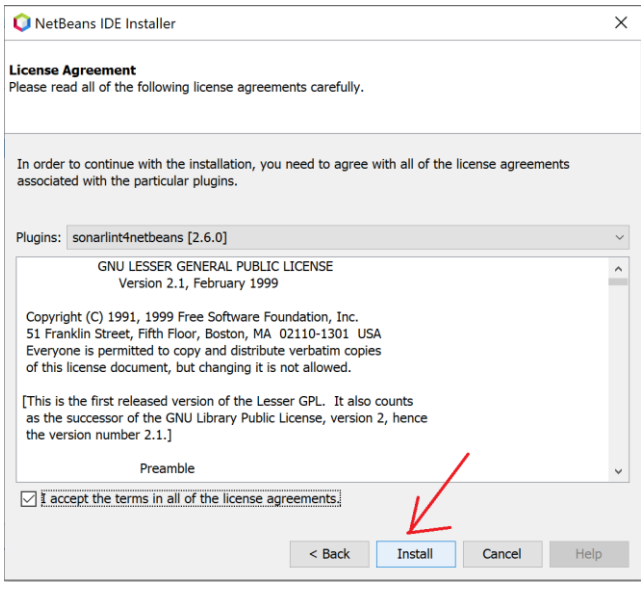
7.- Instala el plugin ***SonarLint*** en NetBeans (del autor FICHET Philippe).((En la versión 12.5 está ya preinstalado, por lo que bastaría activarlo)

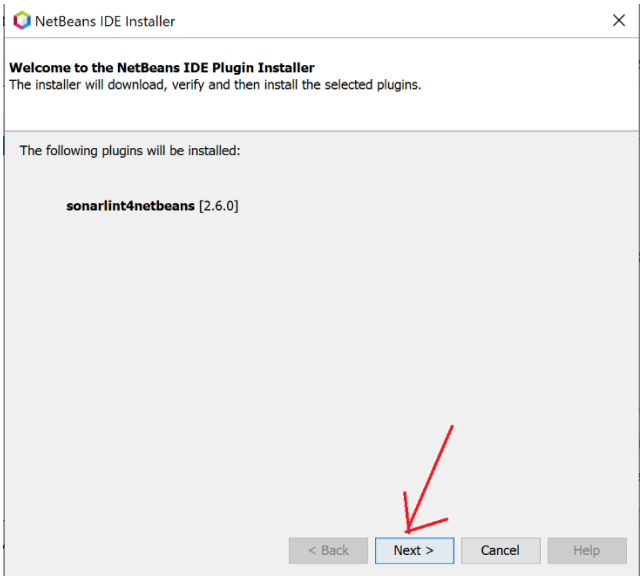
SOLUCIÓN:

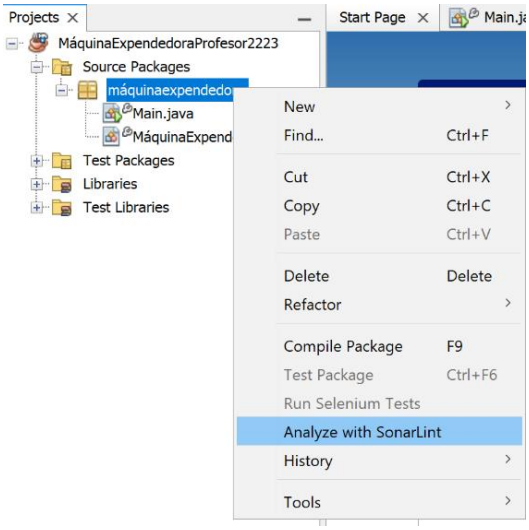
En plugins disponibles (***Available Plugins***) buscamos ***SonarLint*** y damos a activar.

Si no estuviera instalado procedemos del mismo modo, pero al llegar al punto anterior damos a instalar y seguimos los pasos (Al terminar nos pedirá reiniciar el IDE)









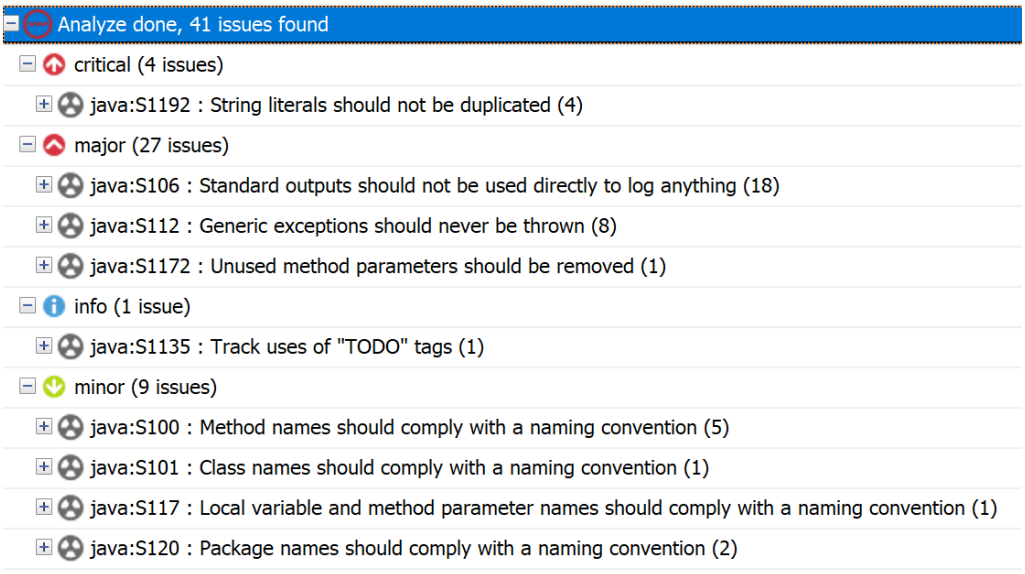
8.- Analiza el código del proyecto ***MáquinaExpendedora*** con el analizador de código Sonarlint. Muestra y comenta los resultados.

SOLUCIÓN:

Una vez instalado el plugins y reiniciado en entorno pasamos a analizar nuestro proyecto, haciendo click en el botón derecho sobre el mismo y eligiendo ***Analize with SonarLint***.

Cuando pasamos a analizar obtenemos un listado de las advertencias de calidad de nuestro software que deberemos a analizar.

Observamos como establece una gradación de mayor a menor “severidad”. Las desplegamos y explicamos el significado:



EJEMPLO DE CÓMO COMENTAR UN MENSAJE DEL ANALIZADOR (Ayudaos de Internet):

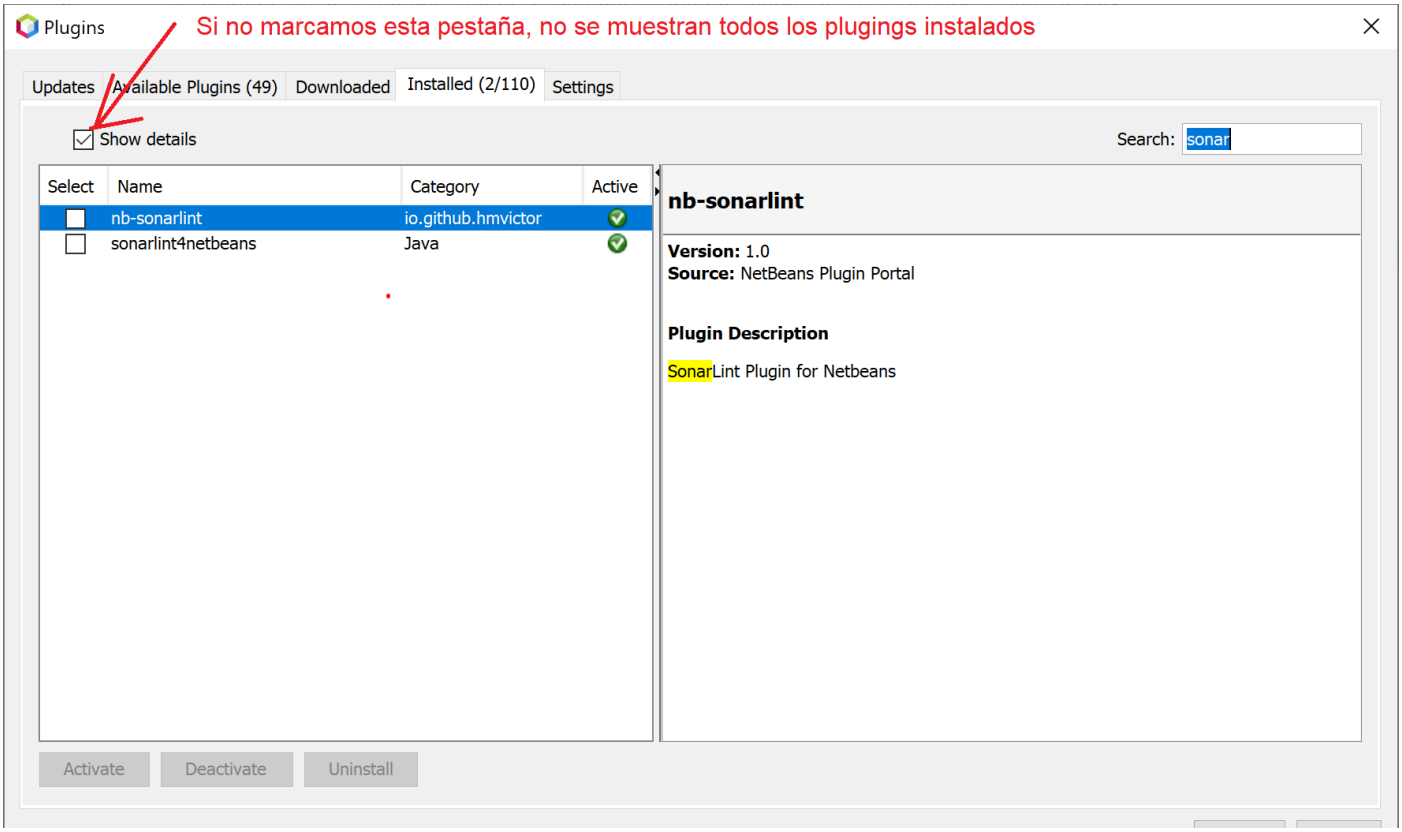
Java: S1172: Los parámetros de método no utilizados deben eliminarse. Por ejemplo, ***regaloProfesor2223***, creado para practicar el apartado de añadir parámetro mediante refactorización, pero que no se usa.

```
74      /*Método que permite obtener un snack si se tiene suficiente dinero y hay
75  suficientes unidades en la máquina*/
        public void sacarsnack(int unidades, double dinero, String regaloProfesor2223) throws Exception {
```

Java: S120: Los nombres de los paquetes de JAVA deben cumplir con una convención de nomenclatura. En el ejemplo, lleva tilde.

9.- Añade la regla ***Lines should not be too long*** para un valor mayor de 25. Analiza de nuevo y, muestra y comenta los resultados. (Deben aparecer los problemas detectados, que serán las líneas de código que tienen una longitud mayor de 25)

¡¡OJO!! IMPORTANTE (FALLO QUE DIO EN LA CONEXIÓN ONLINE)

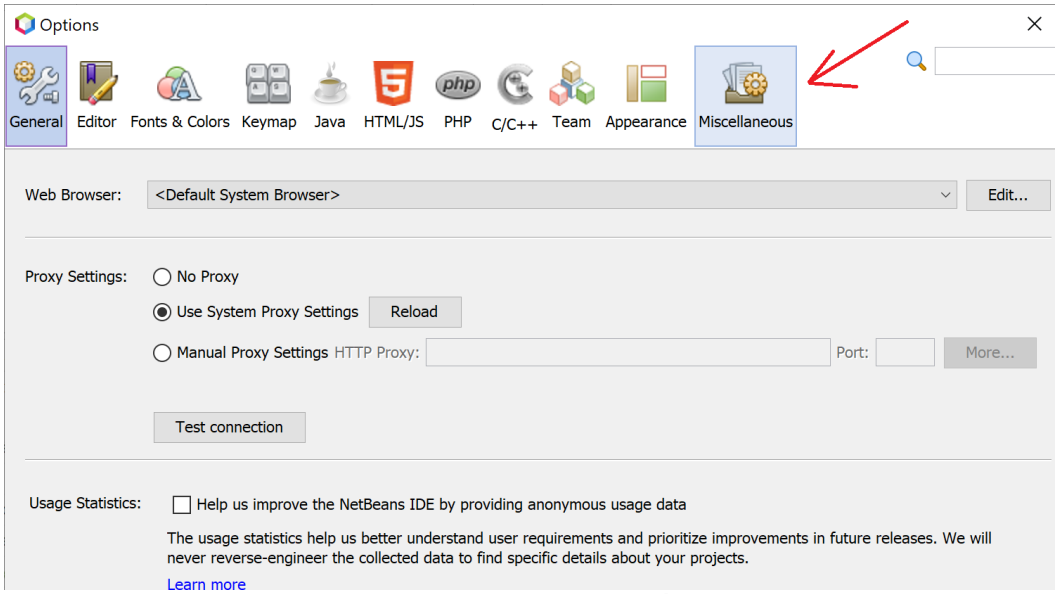


Para que funcione correctamente sólo tenemos que instalar sonartlint4netbeans (JAVA). Si instalamos el otro o los dos no nos sale la ventana que aparece a continuación.

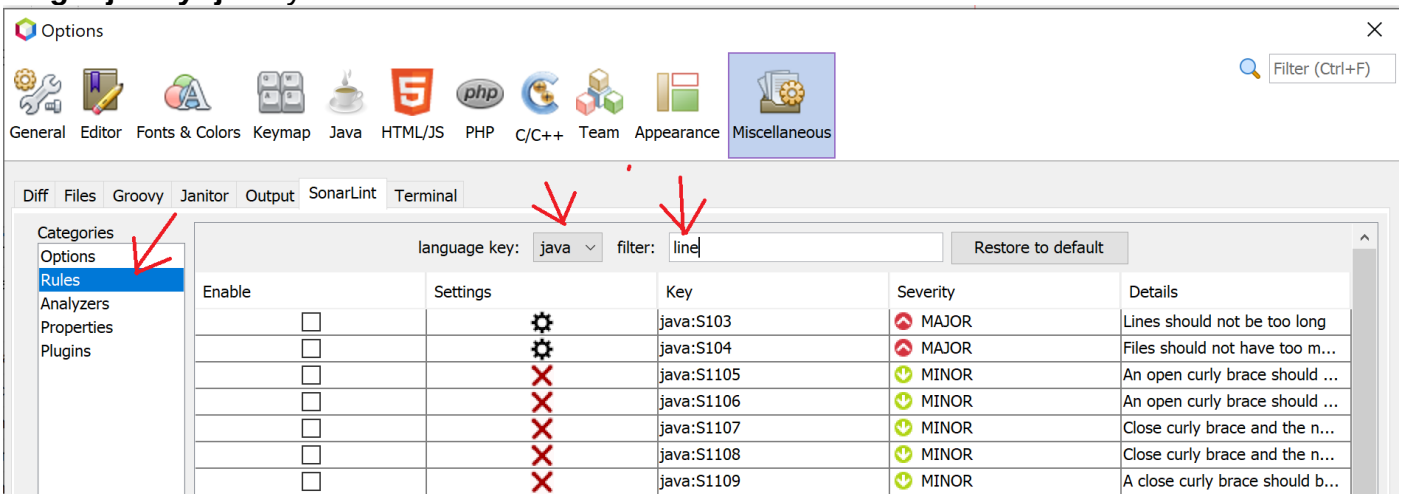
(***NOTA:*** Ten en cuenta que, aunque no lo indica en ningún momento cualquier cambio en la configuración del plugin va a precisar que reinicies el IDE para que puedas apreciar los cambios).

SOLUCIÓN:

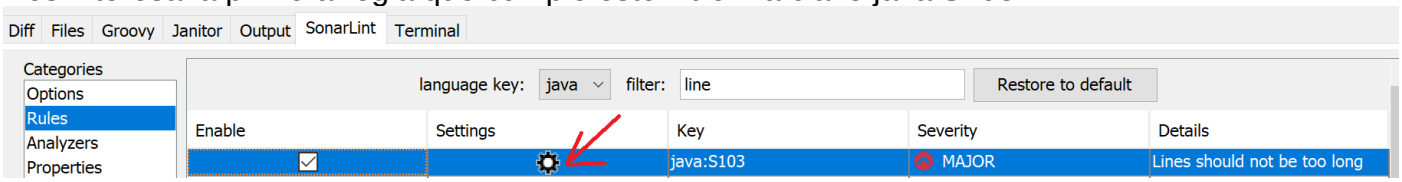
Para modificar una regla en SonarLint nos vamos a menu **Tools + Options** de Netbeans, y hacemos click en seccion **Miscelaneous**



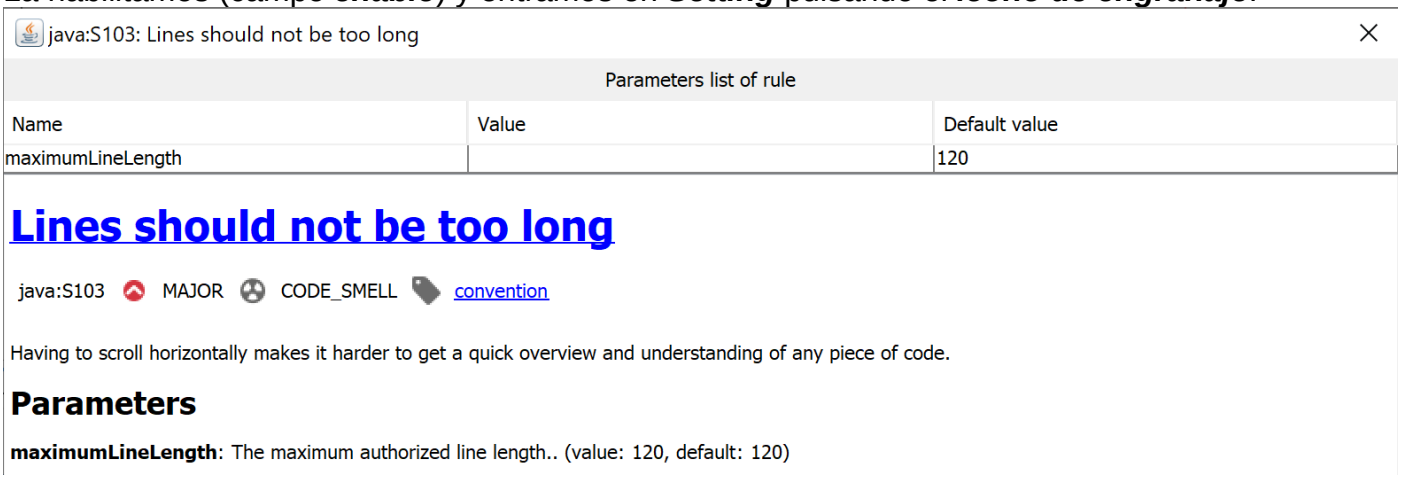
En esa sección, vamos a la pestaña **SonarLint** y marcamos la opción **"Rules".** Seleccionamos **languaje key: java** y buscamos con el **filtr: line**



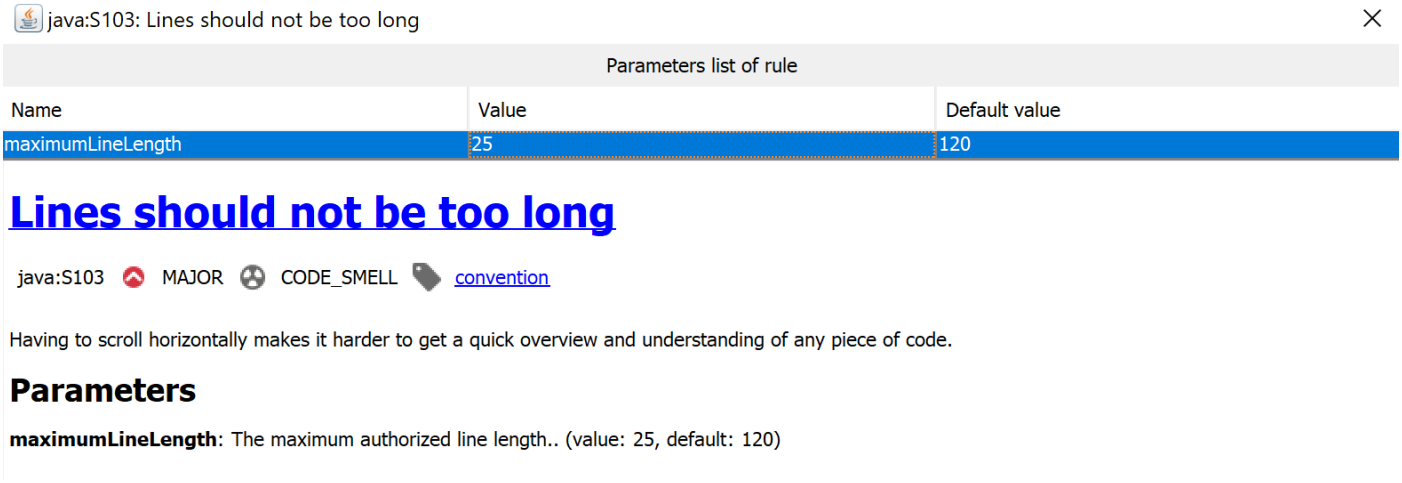
Nos interesa la primera regla que cumple este filtro. La clave java:S103



La habilitamos (campo ***enable***) y entramos en ***Setting*** pulsando el ***icono de engranaje***.

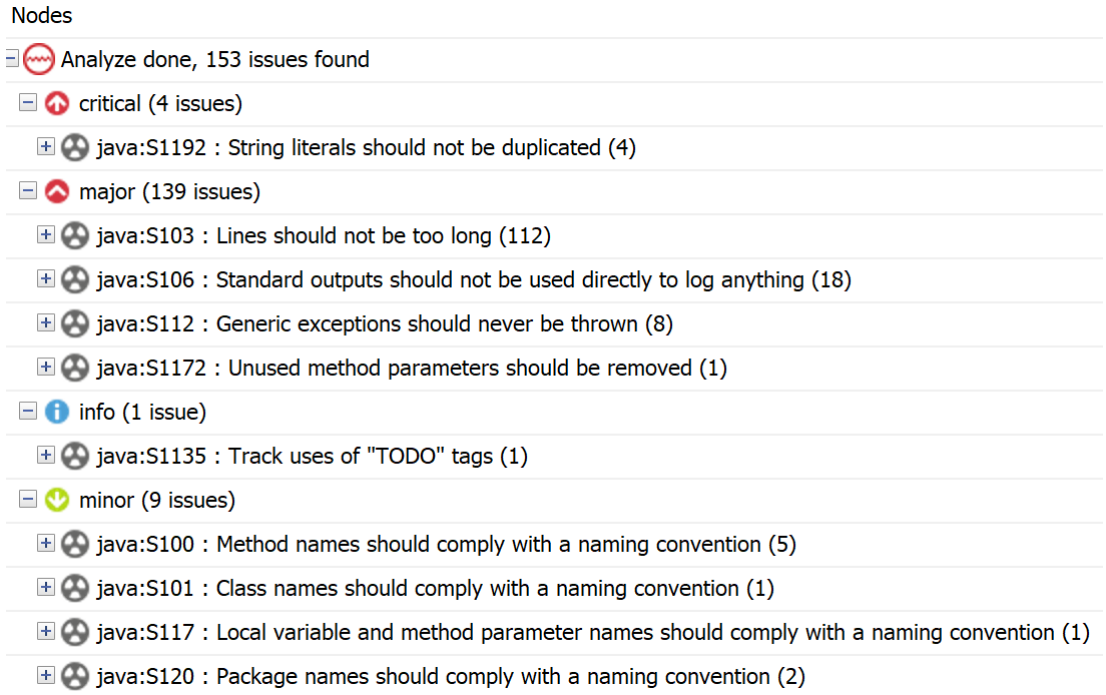


La ventana que se abre es la que tiene los valores por defecto de esa regla. ***Value*** está vacío y ***Default Value***, tiene 120. Vamos a establecer nosotros ***Value=25. Pulsamos INTRO y cerramos. Si volvemos a pulsar en el engranaje, vemos que ha cambiado.***



Luego pulsamos OK para guardar los cambios.

Si volvemos a ejecutar el análisis con sonarLint (proyecto, paquete máquinaexpendedora + click derecho + Analize with sonarLint) vemos que aparecen 153 problemas detectados, frente a los 41 encontrados antes. Entre ellos 112 de la categoría S103, que es la que hemos cambiado. Son las líneas de código que tiene una longitud mayor de 25 caracteres.



Control de versiones

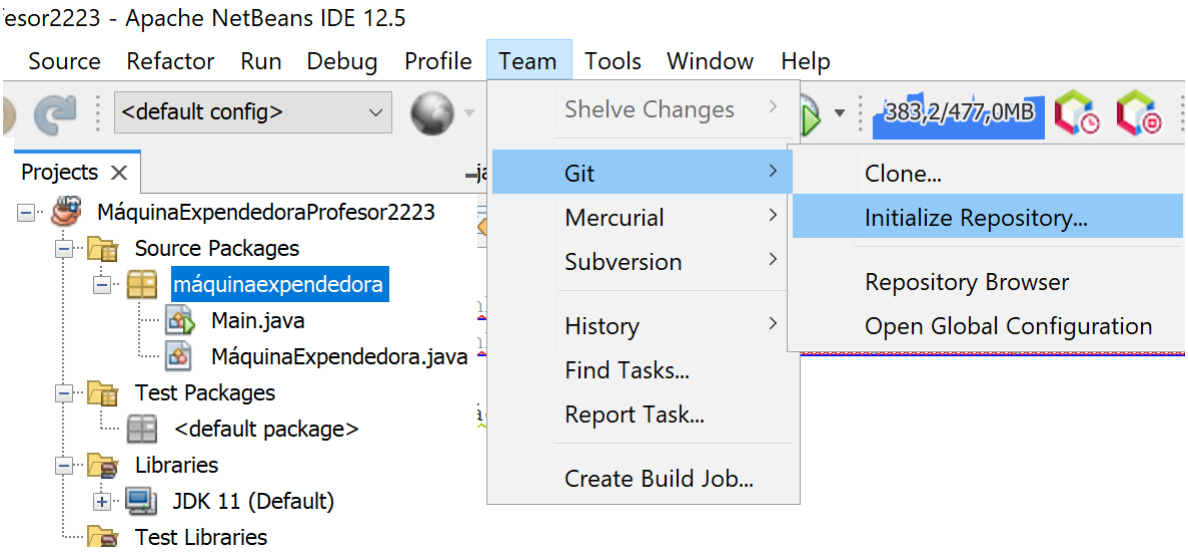
Durante el proceso de desarrollo de software, donde todo un equipo de programadores está colaborando en el desarrollo de un proyecto software, los cambios son continuos. Es por ello necesario que existan en todos los lenguajes de programación y en todos los entornos de programación, herramientas que gestionen el control de cambios.

10.- Inicializa el control de versiones con Git y haz un ***commit*** con el comentario "***Primer commit XXX2223***"

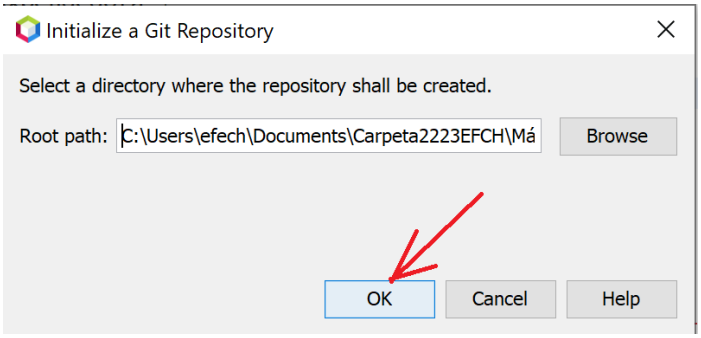
SOLUCIÓN:

Hacer ***commit*** hace referencia a confirmar un conjunto de cambios provisionales de forma permanente.

Para hacerlo, seleccionamos nuestro proyecto en la ventana de proyectos, nos vamos al menú **Team** de Netbeans y seleccionamos como herramienta de control de versiones **Git**. Para inicializar el control de versiones de nuestro proyecto hacemos click en **Initialize Repository**.

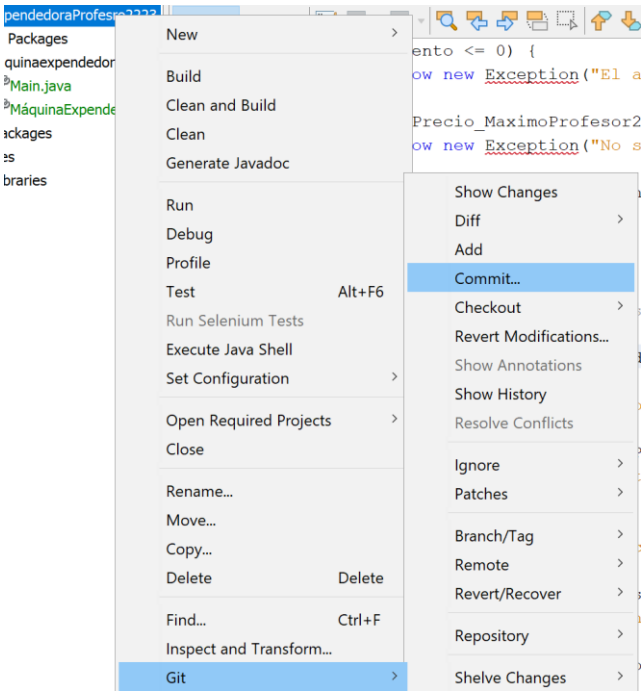


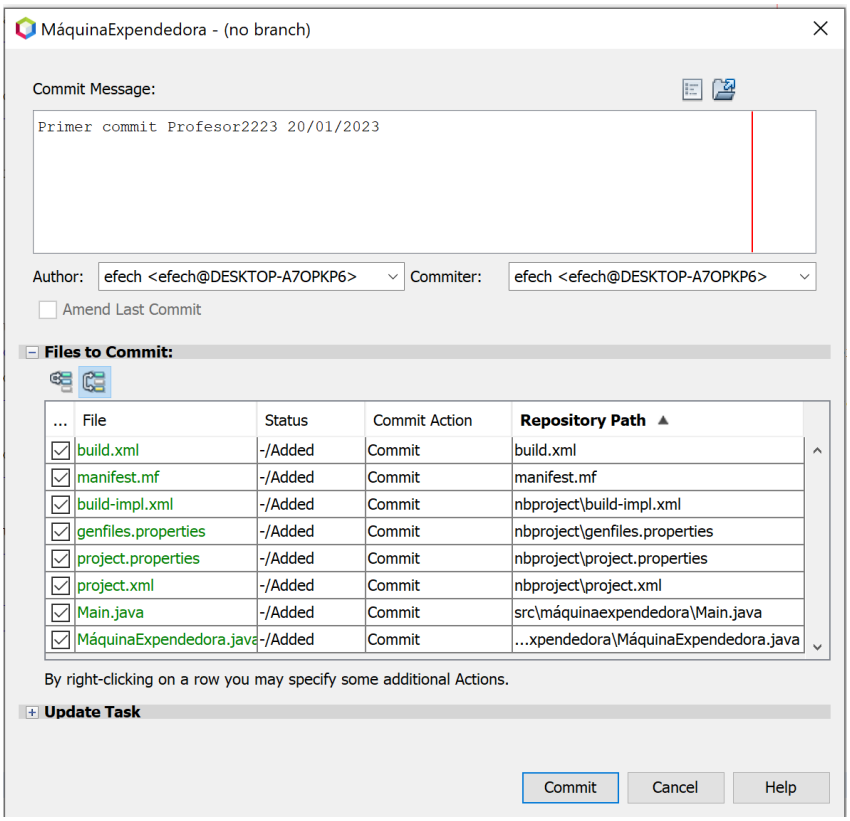
Nos pide confirmación de la carpeta que contiene el proyecto.

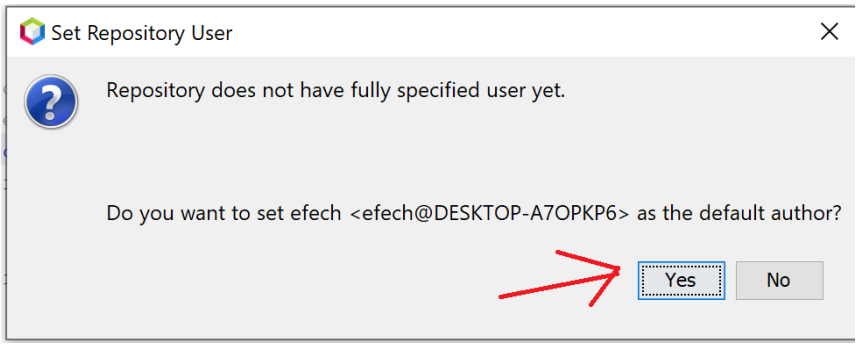


Tras pulsar OK aparecen coloreados los nombres de nuestras clases en verde, indicando que requieren commit.

Para hacerlo hacemos click en el botón derecho y se abre la ventana de ***commit*** para escribamos el comentario. Nos indica los ficheros afectados por el ***commit***.







Las clases vuelven a aparecer en color negro

Documenta el código con las facilidades que da el IDE Netbeans

El proceso de documentación de código es uno de los aspectos más importantes de la labor de un programador. Documentar el código nos sirve para explicar su funcionamiento, punto por punto, de forma que cualquier persona que lea el comentario, puede entender la finalidad del código.

11.- Comenta cada uno de los elementos principales de la clase ***MáquinaExpendedora*** y de la clase ***Main*** siguiendo las normas de JavaDoc.

SOLUCIÓN:

Los comentarios que se utilizan en Java para explicar qué hace un código, se denominan comentarios JavaDoc. Este tipo de comentarios tienen que seguir una estructura prefijada.

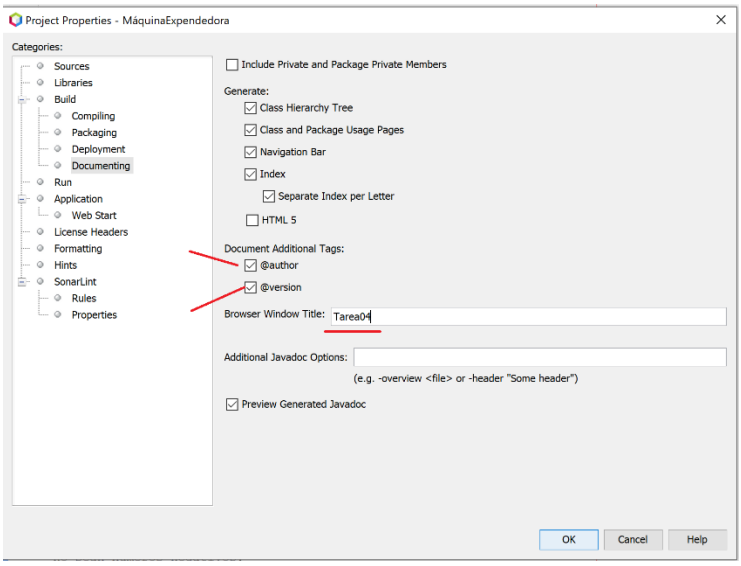
Los comentarios son obligatorios con JavaDoc, y se deben incorporar al principio de cada clase, al principio de cada método y al principio de cada variable de clase. No es obligatorio, pero en muchas situaciones es conveniente, poner los comentarios al principio de un fragmento de código que no

resulta lo suficientemente claro, a la largo de bucles, o si hay alguna línea de código que no resulta evidente y pueda llevarnos a confusión.

Hay que tener en cuenta, que, si el código es modificado, también se deberán modificar los comentarios.

Consideraciones que hay que tener en cuenta para realizar los comentarios:

Los comentarios de una clase deben comenzar con /** y terminar con */ .



- Entre la información que debe incluir un comentario de clase debe incluirse, al menos las etiquetas **@author** y **@version**, donde @author identifica el nombre del autor o autora de la clase y @version, la identificación de la versión y fecha. NOTA IMPORTANTE: Si queremos que aparezca en el Javadoc el autor y la versión de la clase tendremos que indicarlo en las propiedades del proyecto, categoría Documenting.

- Dentro de la la clase, también se documentan los constructores y los métodos. Al menos se indican las etiquetas:

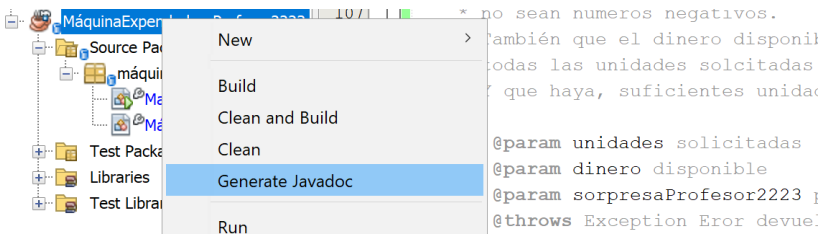
  - **@param**: seguido del nombre, se usa para indicar cada uno de los parámetros que tienen el constructor o método.
  - **@return**: si el método no es **`void`**, se indica lo que devuelve.
  - **@exception**: se indica el nombre de la excepción, especificando cuales pueden lanzarse.
  - **@throws**: se indica el nombre de la excepción, especificando las excepciones que pueden lanzarse.

12.- Genera documentación JavaDoc para todo el proyecto.

SOLUCIÓN:

Descripción. JAVADOC, **es una herramienta del SDK que permite documentar, de una manera rápida y sencilla, las clases y métodos que se proveen**, siendo de gran utilidad para la comprensión del desarrollo.

Para generar el ***Javadoc*** del proyecto, nos vamos a la ventana de proyectos, seleccionamos el nuestro y hacemos clic con el botón derecho marcando la opción ***Generate Javadoc***. Esto generaría un documento HTML.



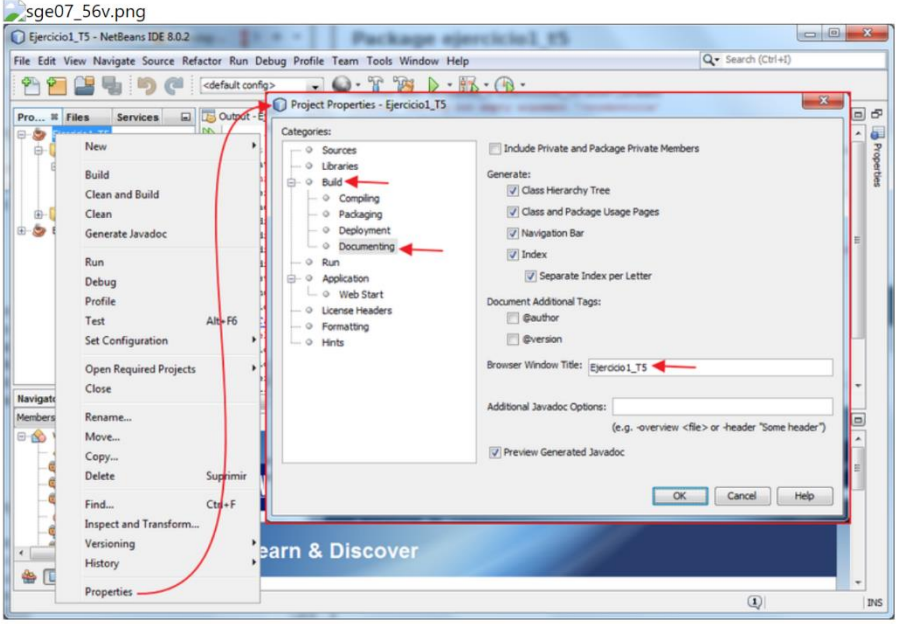
NOTAS:

Si da error como este:

Error 1: Warning: Leaving out empty argument '-windowtitle'

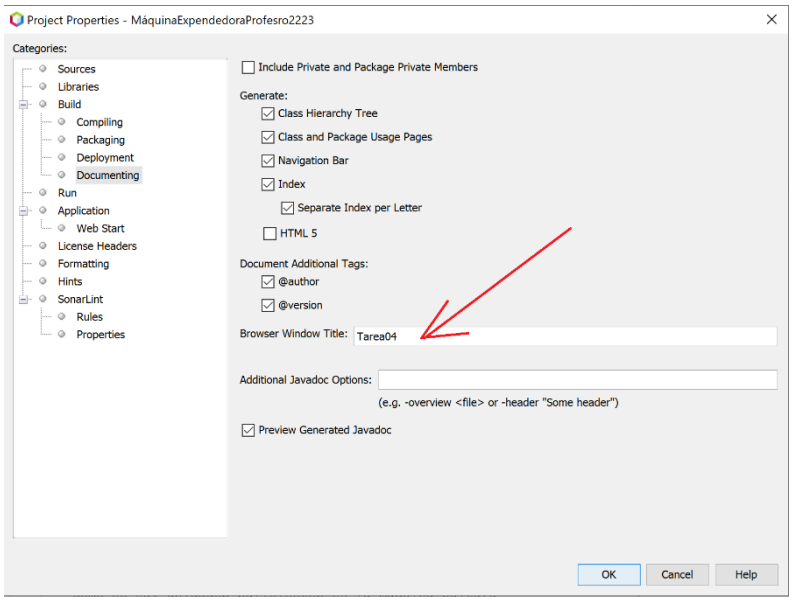
Indica que que falta una etiqueta de título que puede añadirse a mano, ver captura:

sge07_56v.png



Marcamos la aplicacion con boton derecho, seleccioamos propiedades, luego build y Documenting y en las opciones que se muestran añadimos en la etiqueta el nombre que queramos darle.

Tras esto ejecutamos de nuevo javadocs, el error estará corregido y el documento javadoc se instala también en el directorio dist de la aplicación.



Puede dar también el siguiente error:

javadoc: error - cannot read Input length = 1

Para solucionarlo comprobamos la ruta de creación del documento javadoc. Suele dar problema con nombres muy largos y tildes.

Control de versiones

13.- Realiza otro commit con el comentario "Proyecto XXX2223 comentado con JavaDoc".