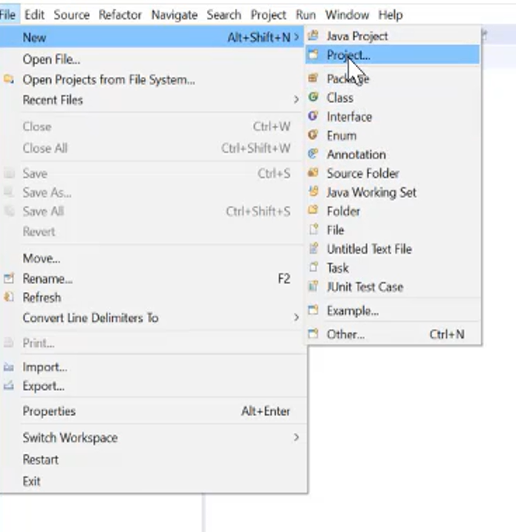
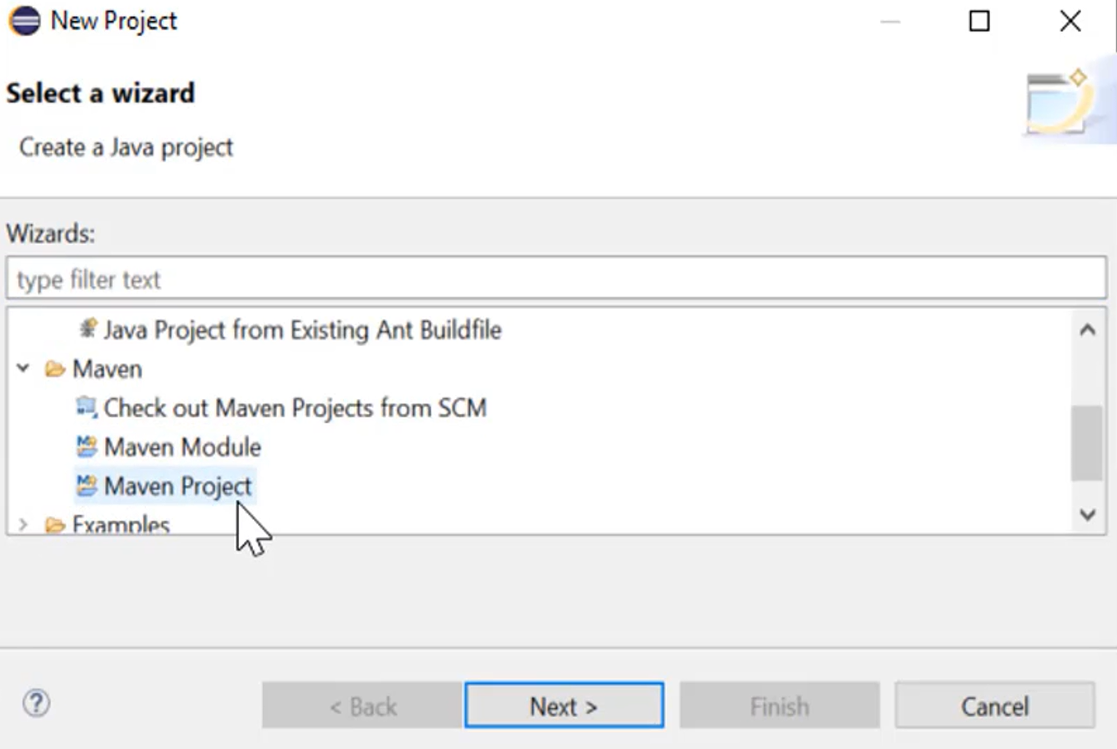
# Creación de proyecto con java Maven en Eclipse

**Creación de Proyecto Maven**

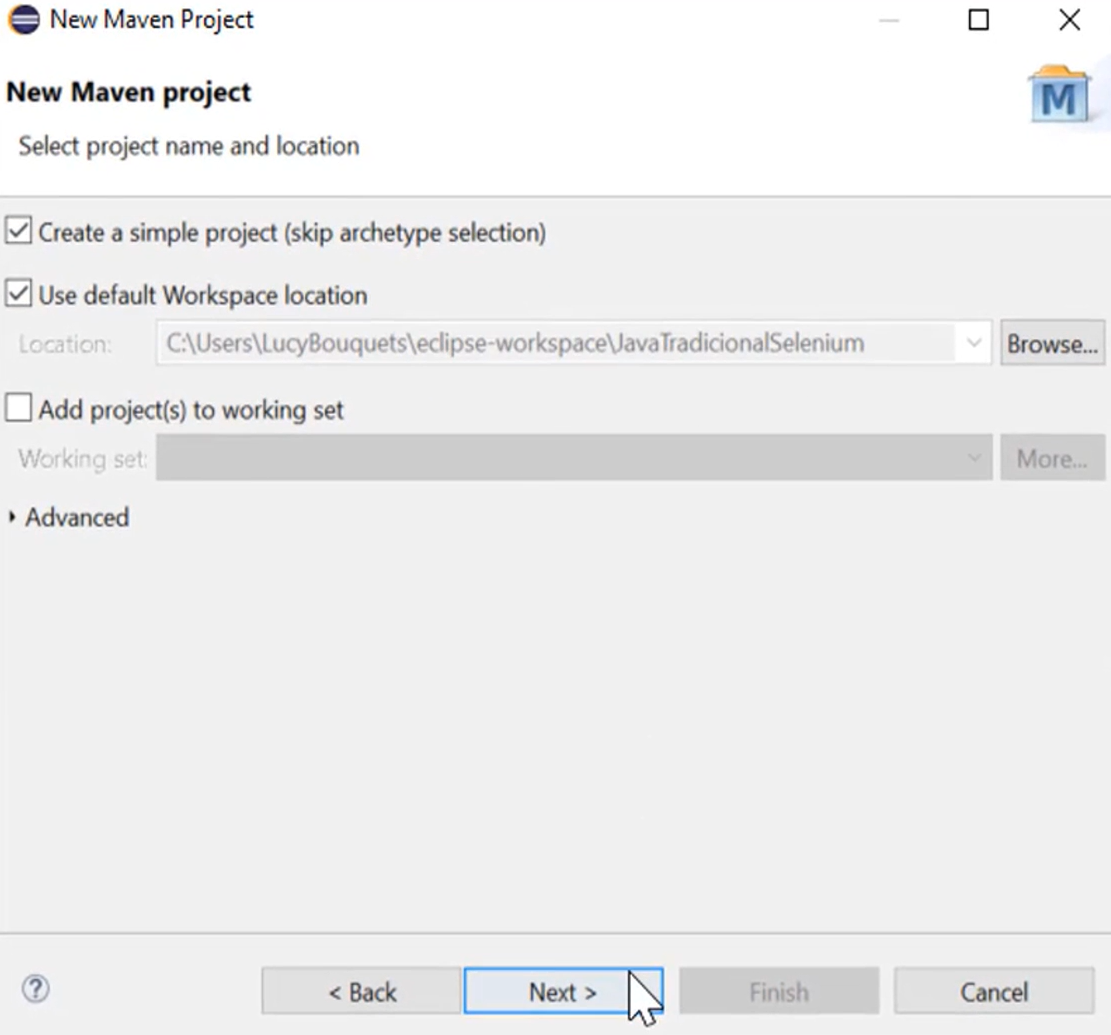
File/new/project

****

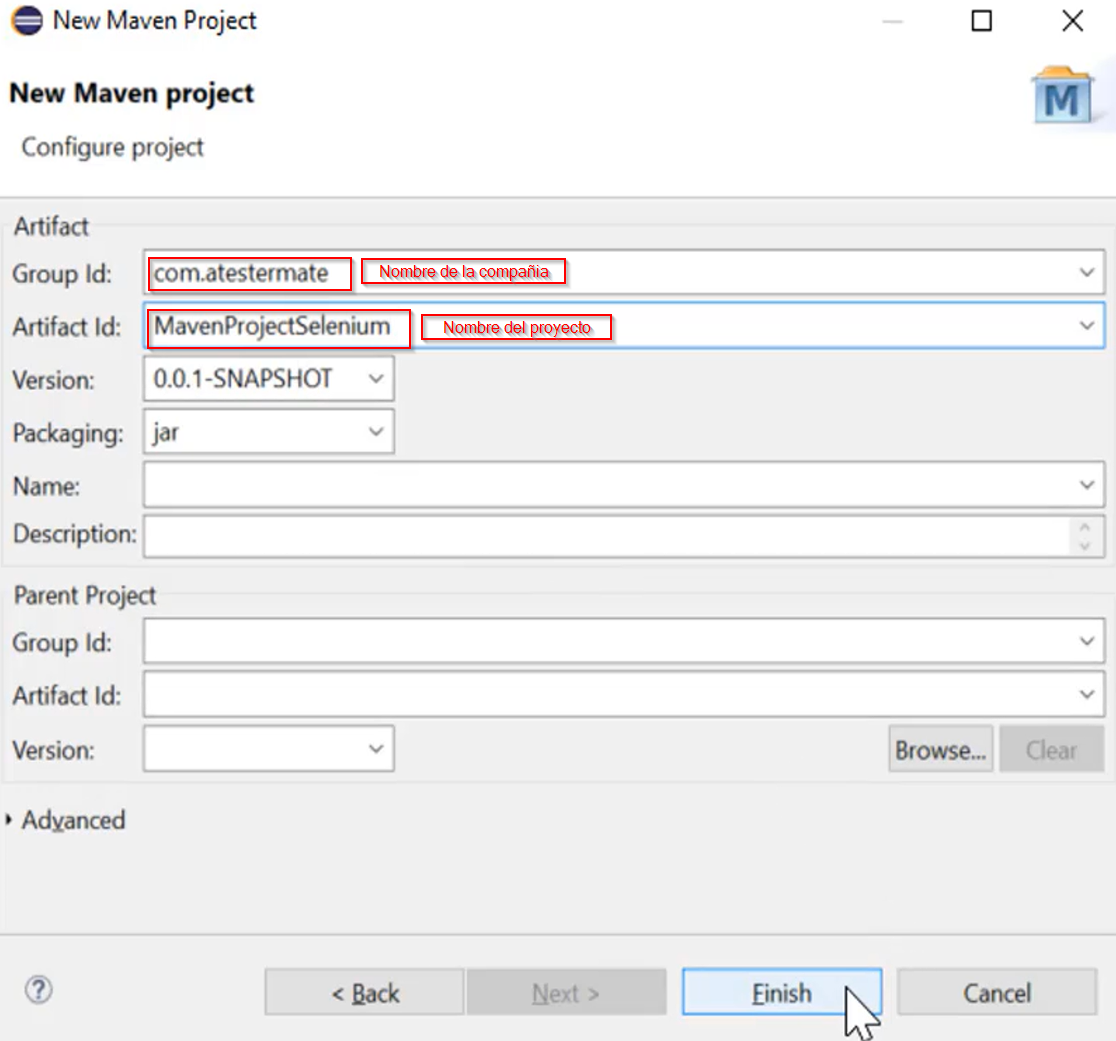
Seleccionamos Maven project

****

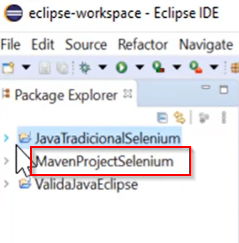
Marcamos los dos primero check y pulsamos Next



Introducimos el nombre de la compañía y del proyecto

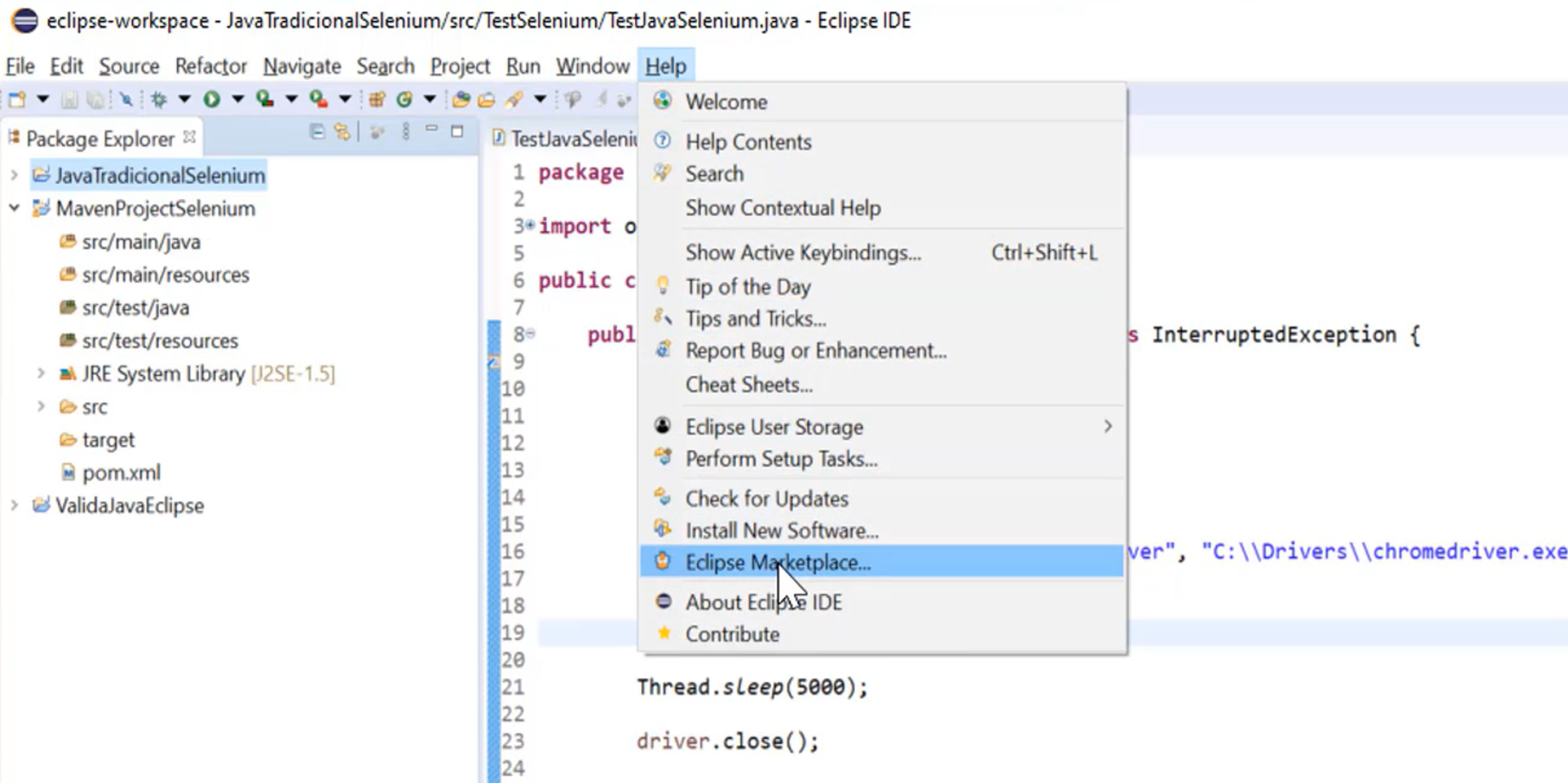


Se crea el proyecto

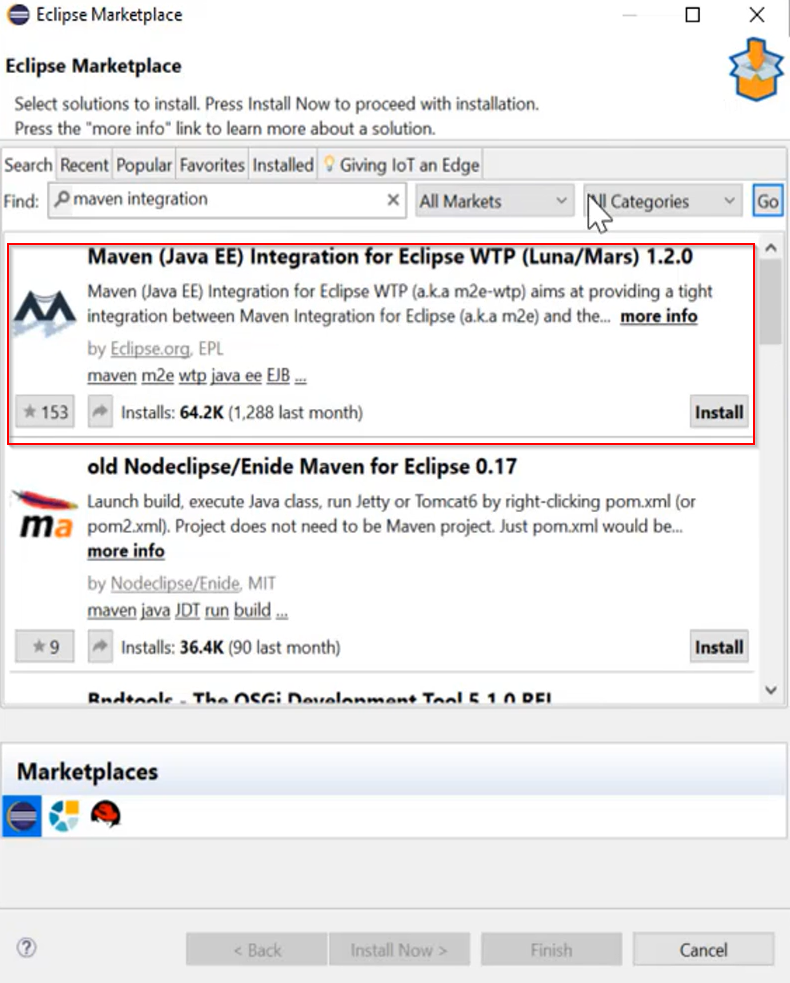


**Instalación de plugin**

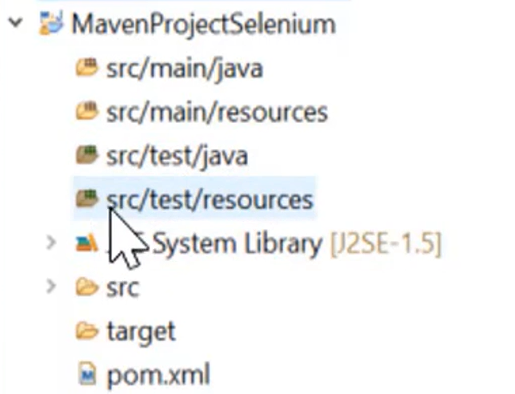
Tenemos que crear un plugin para que Eclipse pueda usar el Maven, para eso vamos a help/Eclipse Marketplace



Buscamos y instalamos Maven integration



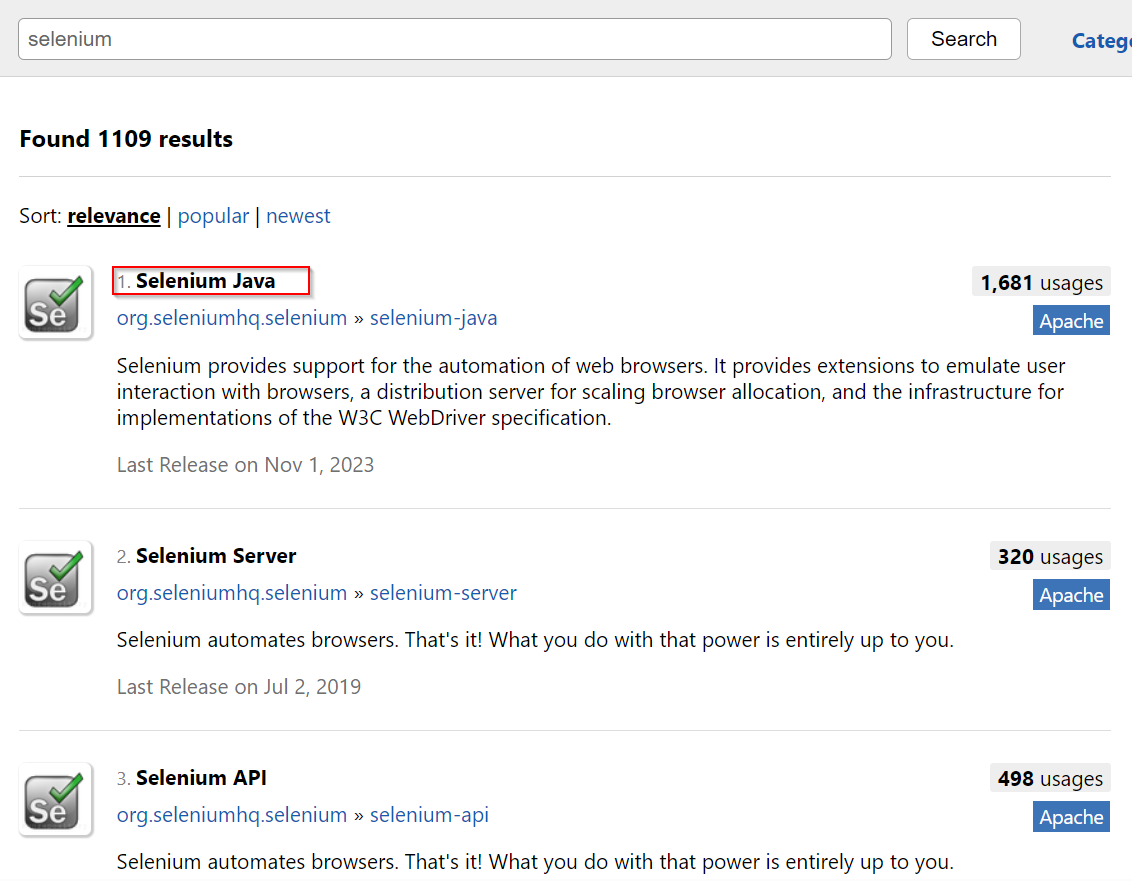
**Instalación de repositorio en pom**

En el proyecto se observa el src/test/java: aquí se guardara todos los script de java

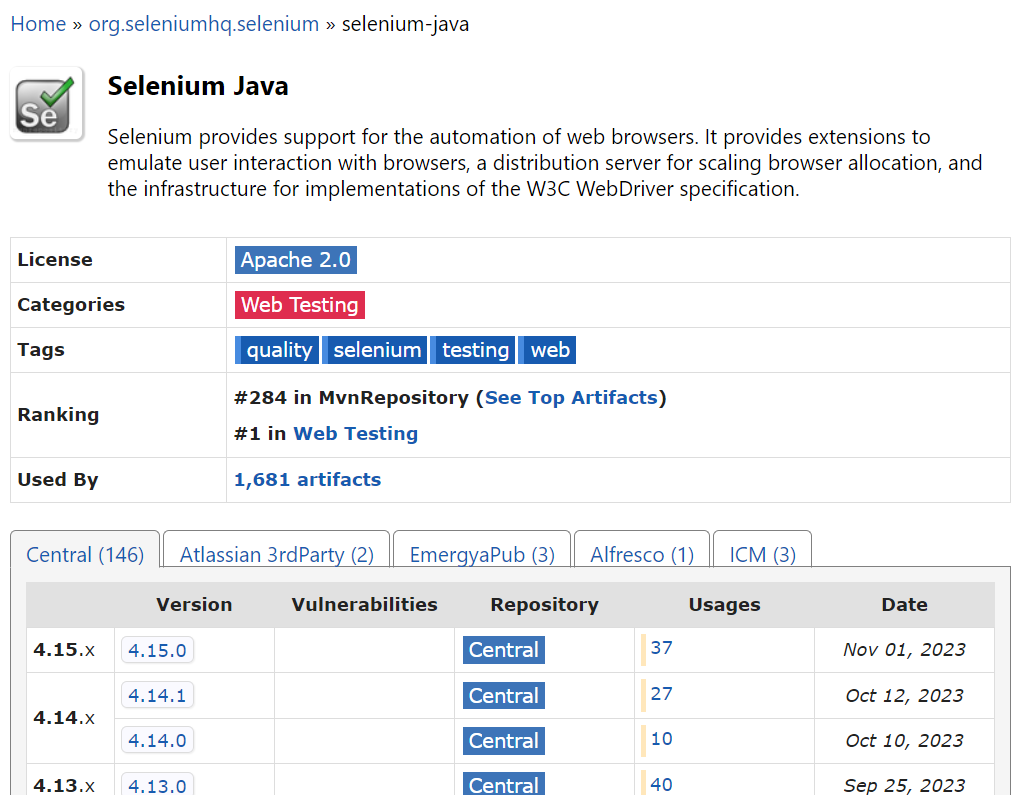
En src/test/resource: se guarda todos los recursos que no son de java

Pom.xml: aquí se guarda las dependencias Maven

Nos vamos a <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>



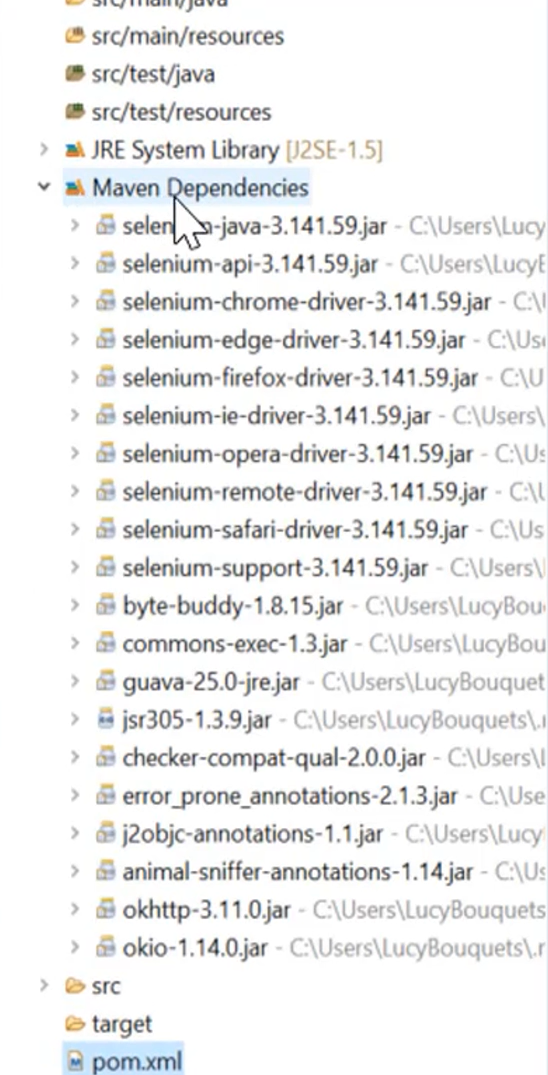
Cogemos el más reciente



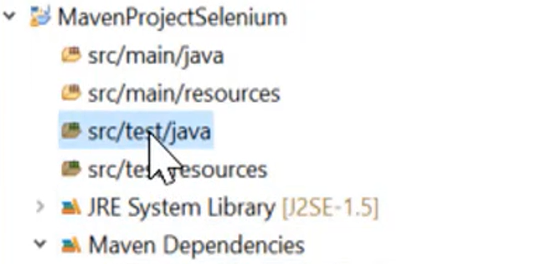
Se introduce la dependencia de selenium y se da a guardar



Comprobamos las dependencias:

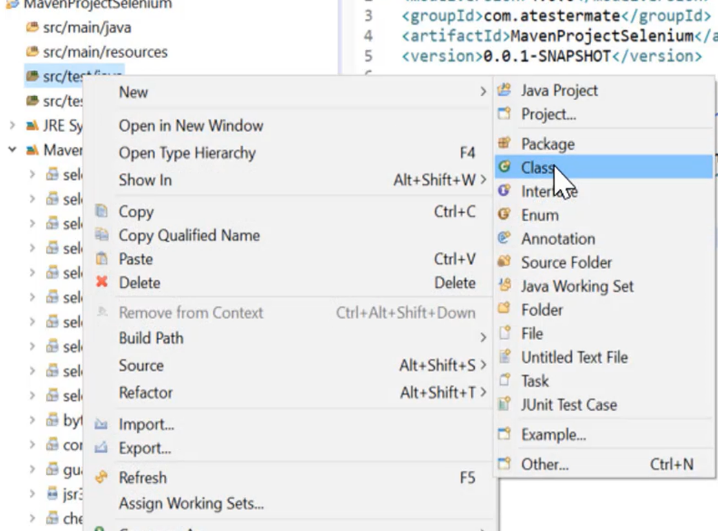


**Creación de clase**

Creamos una clase y en src/test/java pulsamos clic derecho

Existen dos maneras de ejecutar el código puede ser desde el propio ide o desde Maven se hará de las dos maneras:

**Lanzar desde IDE:**

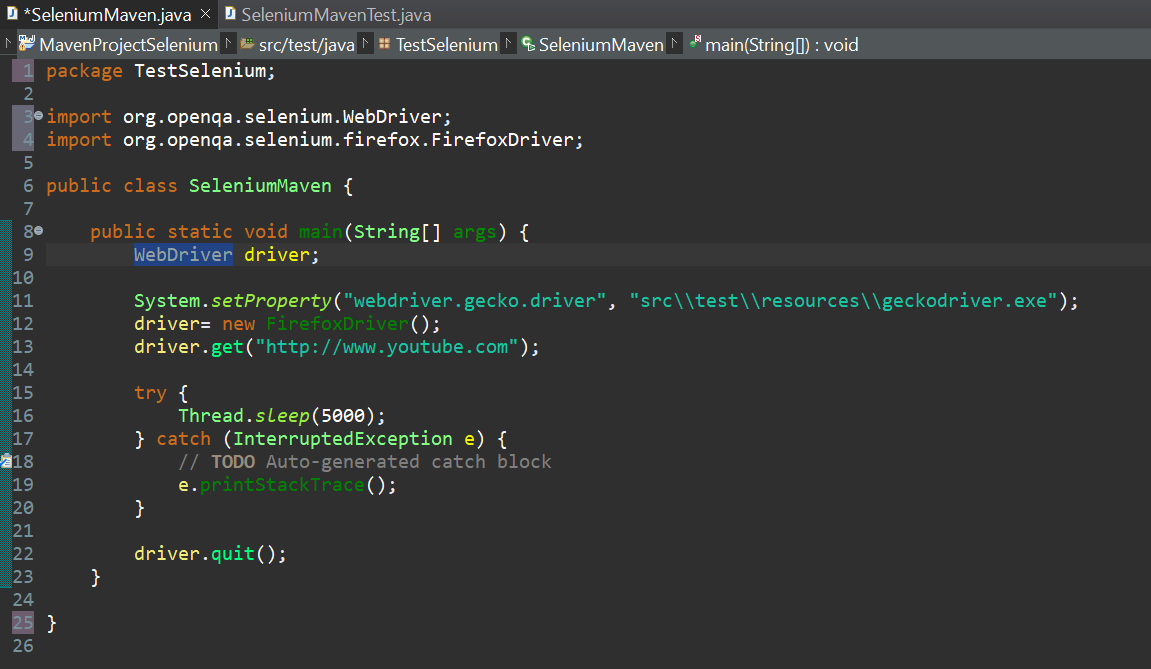


Primero creamos una clase

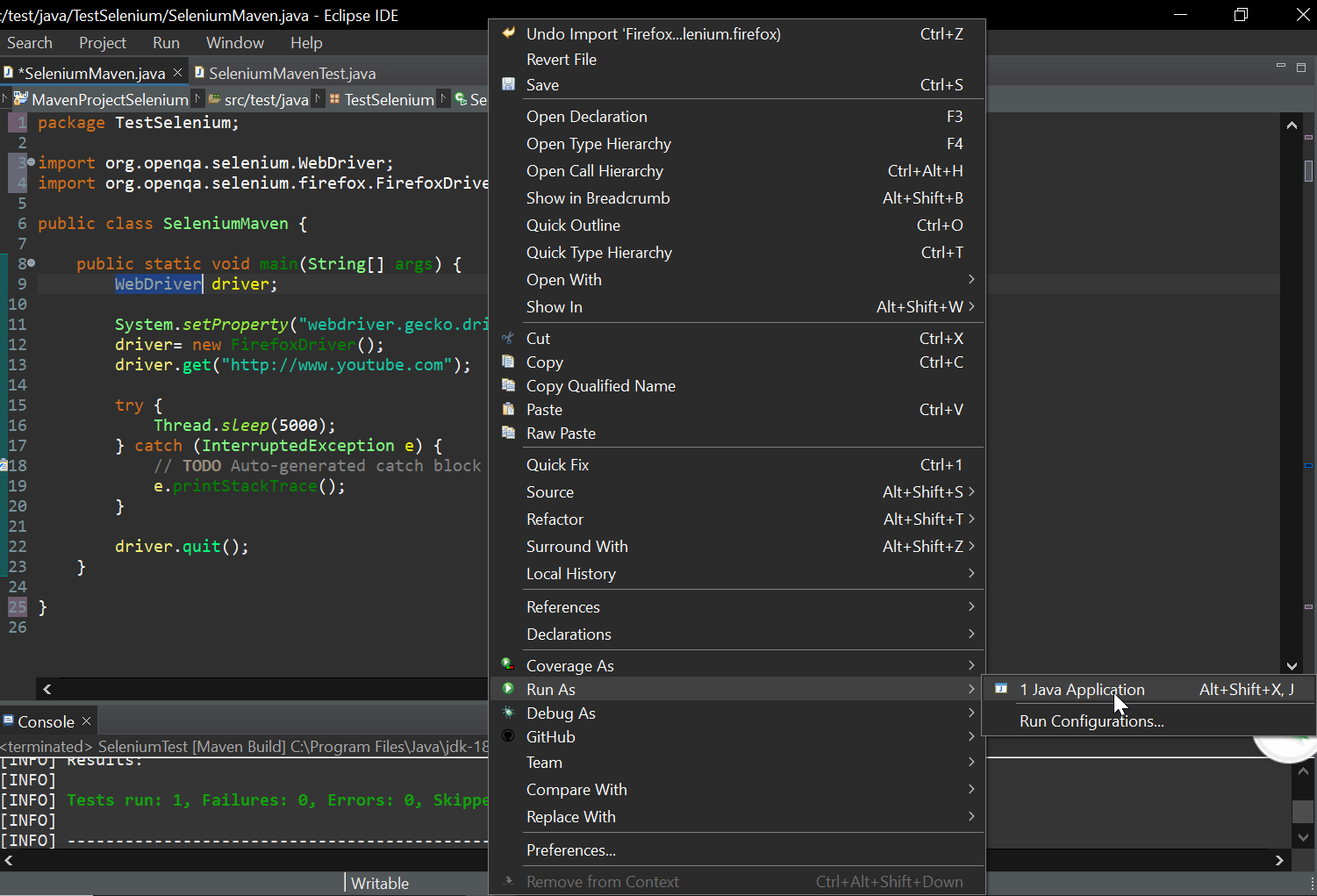


Le damos un nombre al paquete y a la clase.

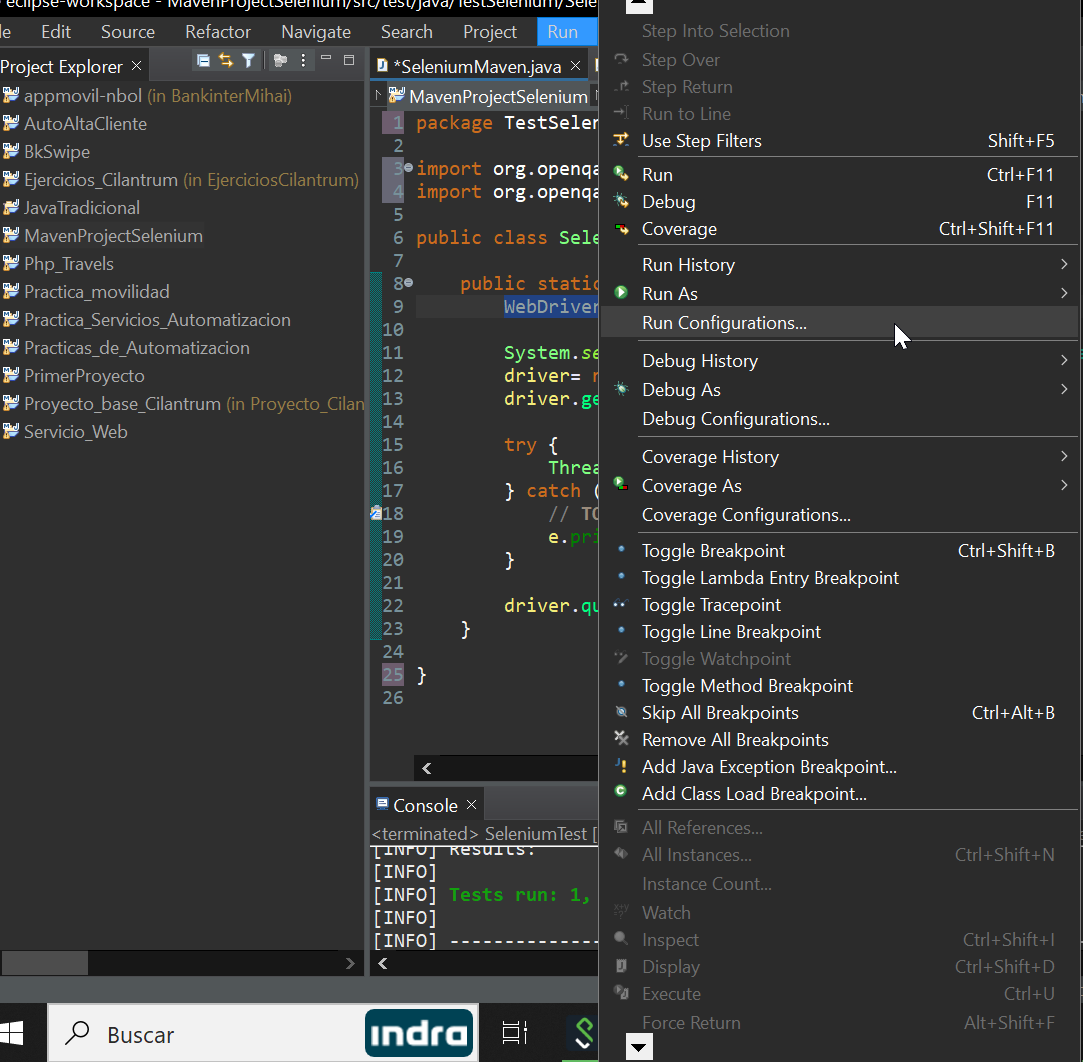
Marcamos la opción public static void main, para crear el main

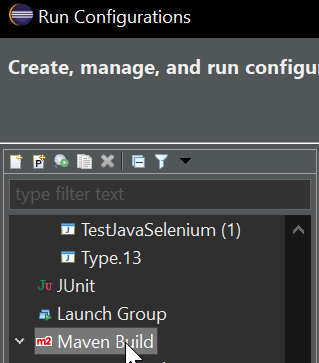


Dentro del main insertamos el código, no olvidar el driver del navegador en este caso el driver geckodriver.exe de mozilla

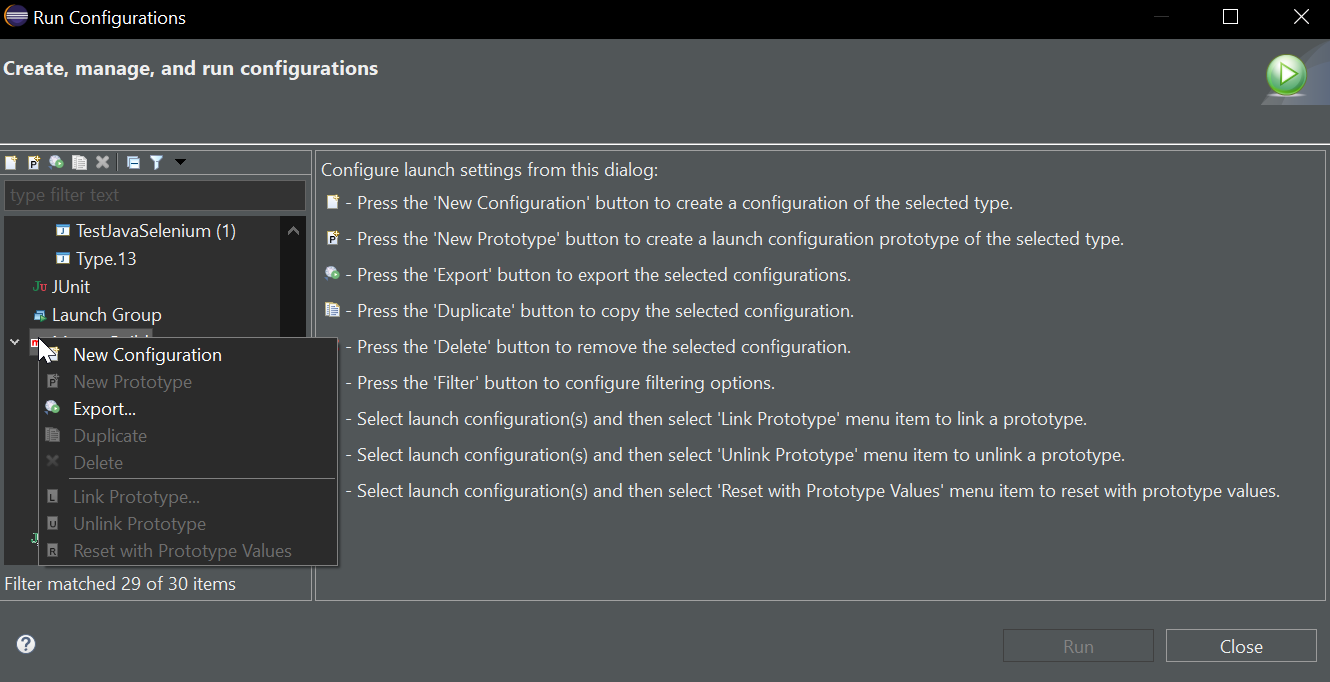
Corremos con Java Application

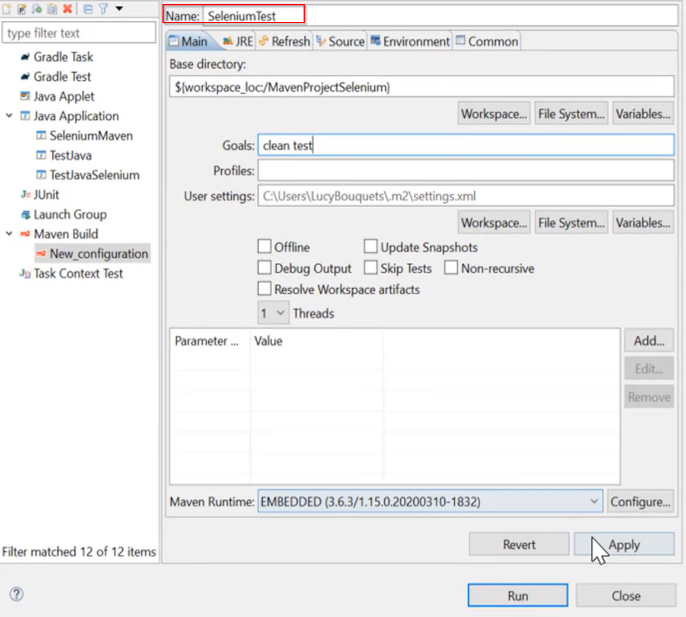
**Ejecutar proyecto como Maven:**

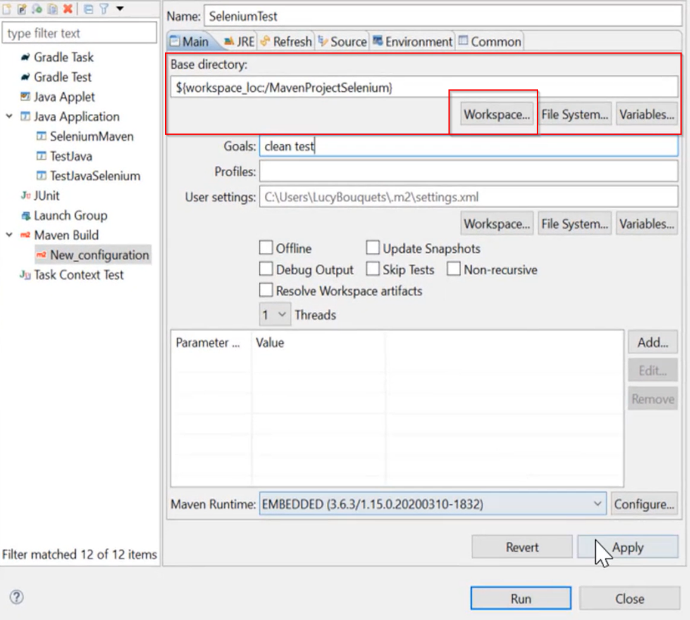
Nos vamos a Run Configurations

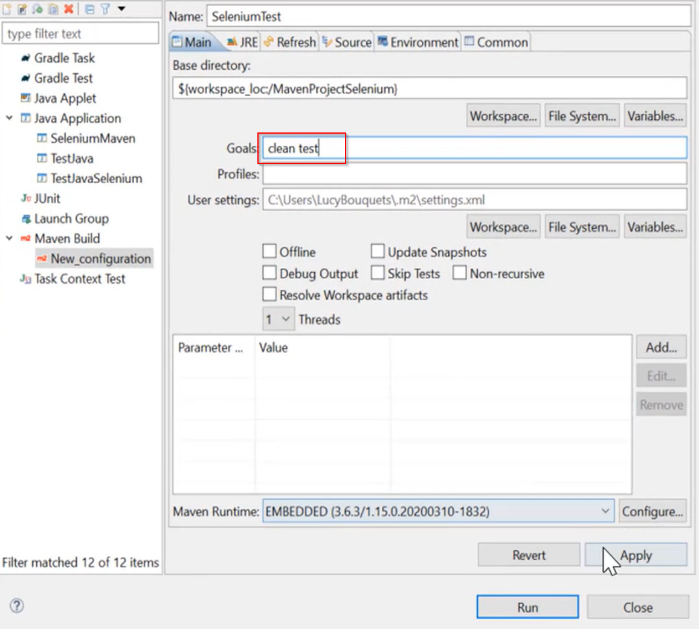
Seleccionamos Maven Build

Clic derecho y pulsamos new Configuration



Insertamos el nombre

Seleccionamos el proyecto

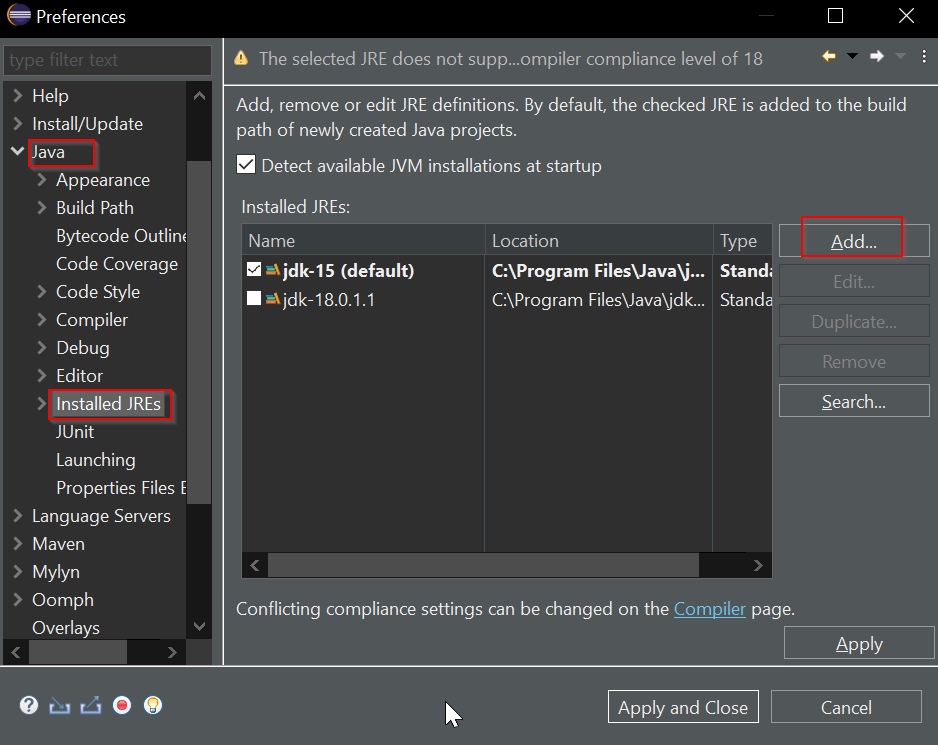


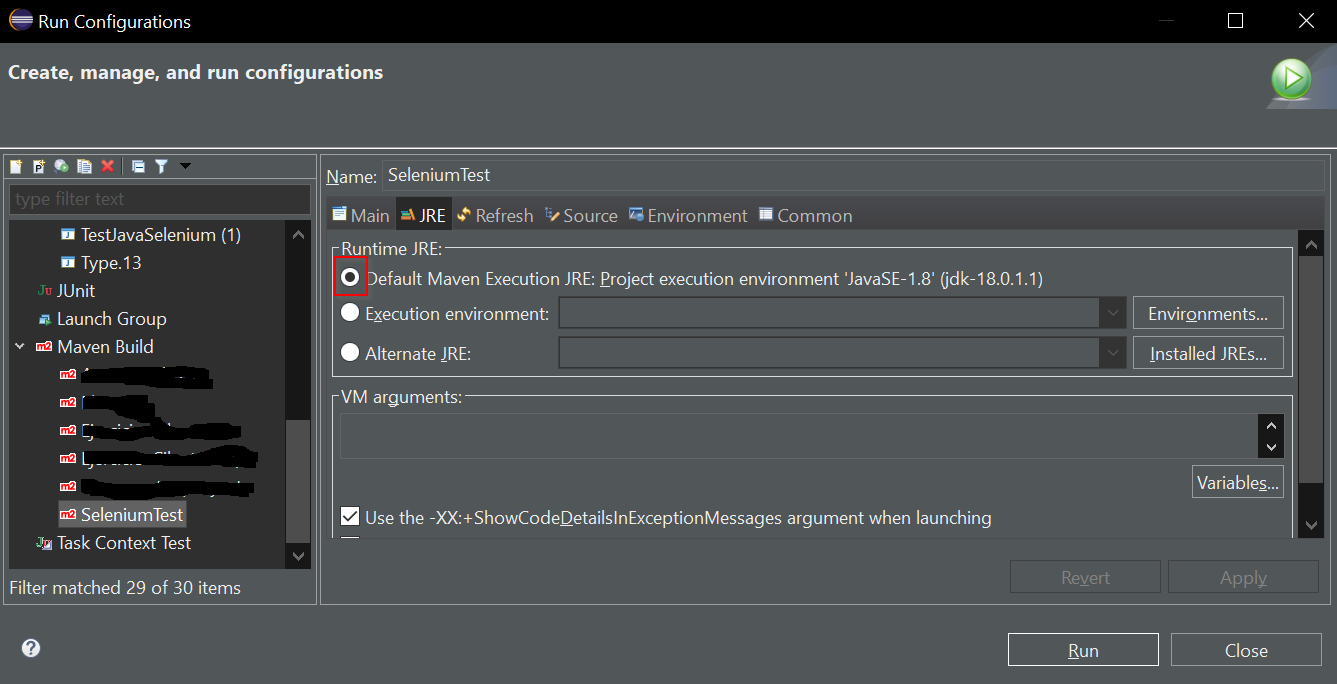
Le decimos que primero haga un clean y luego un test

Pulsamos Apply

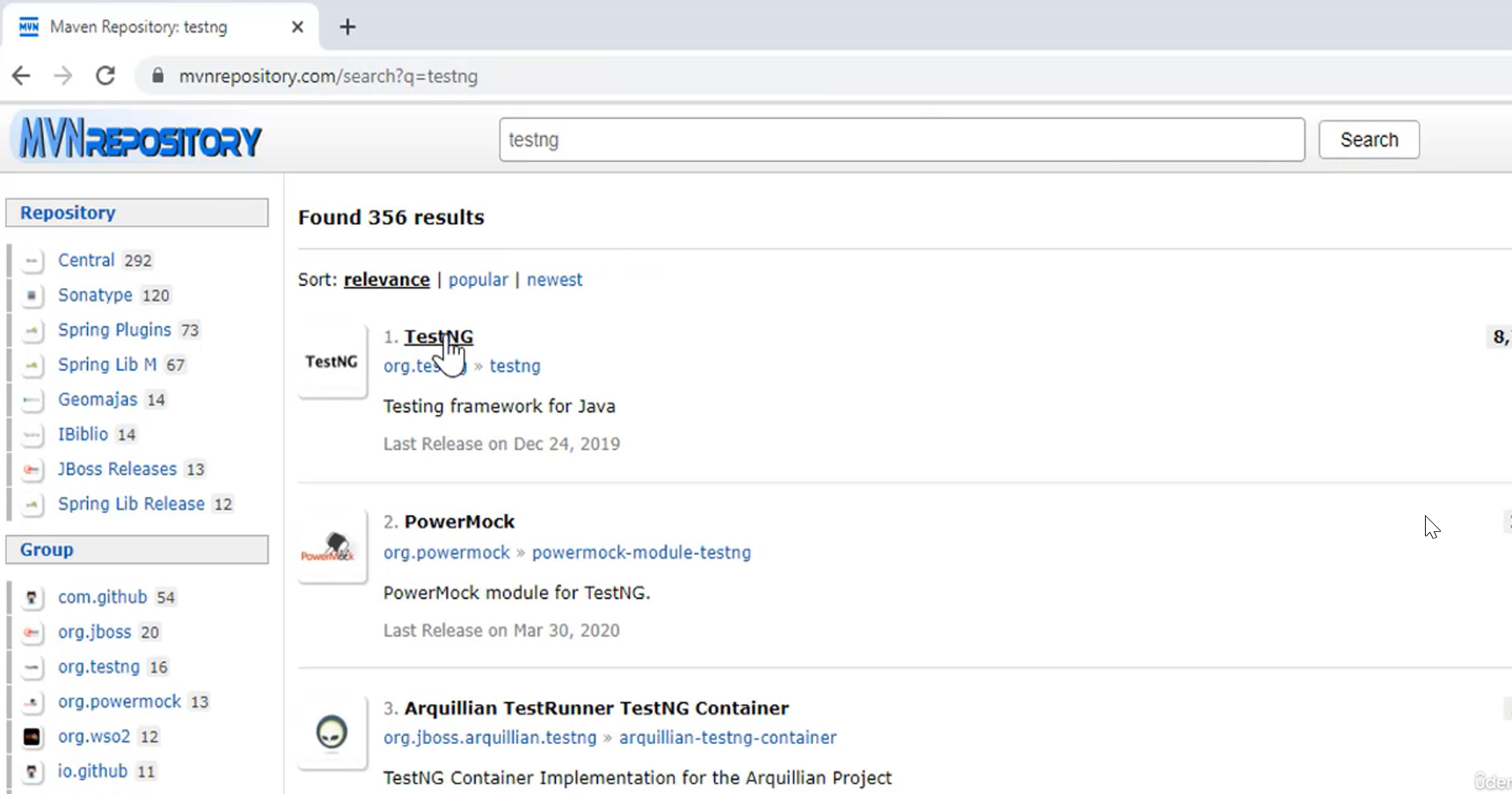
Luego se necesita Instalar el JDK

Vamos a Windows/preferences

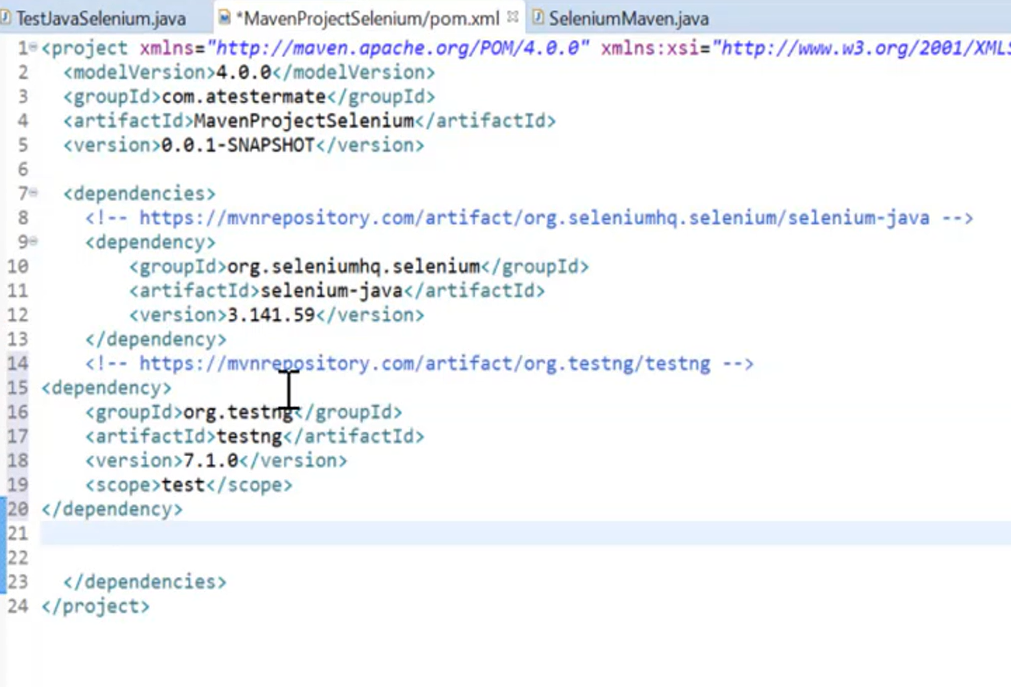
java/installed JREs/Add y agregamos la carpeta donde se encuentra el JDK

Volvemos Abrir el mavenBuild de SeleniumTest y en la pestaña de JRE seleccionamos el default Maven JRE

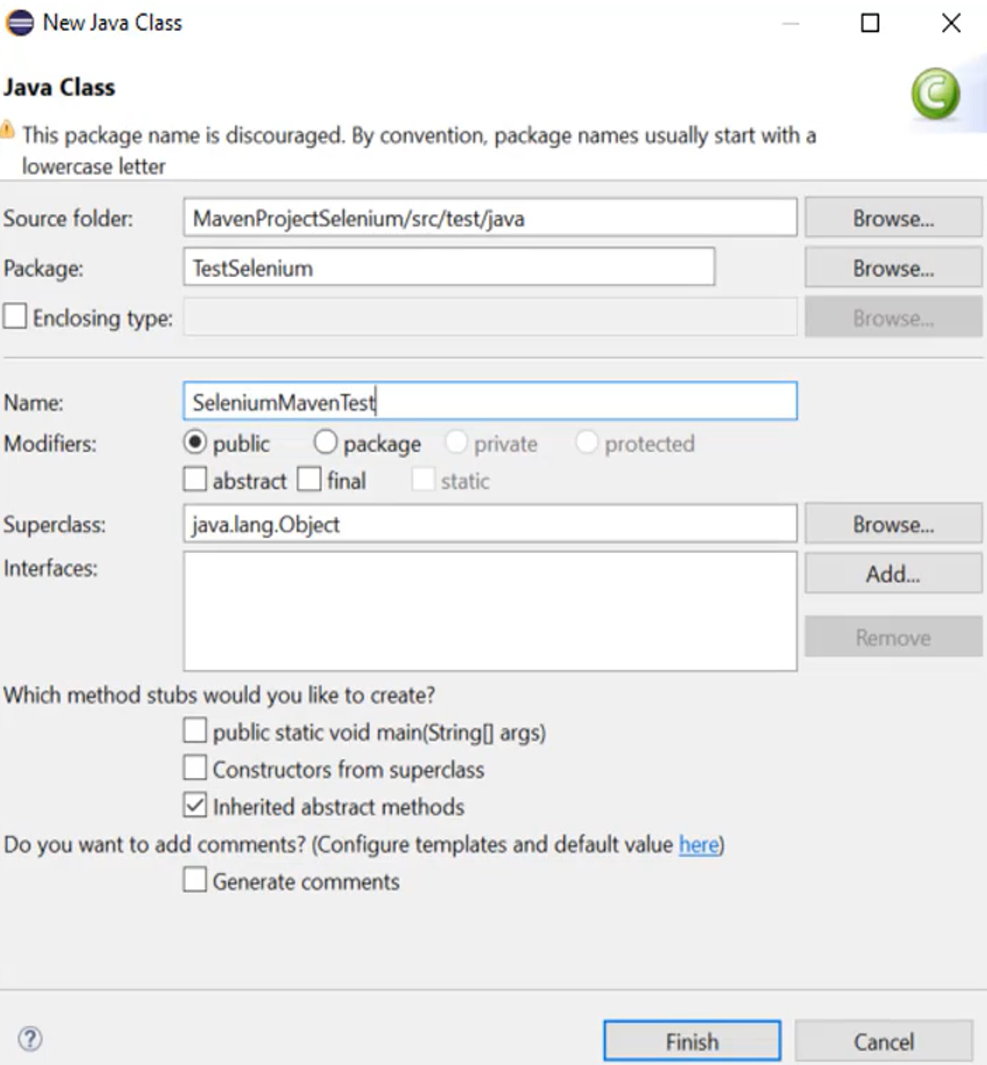
Luego necesitamos descargar la dependencia de TestNG



Pegamos la dependencia y le damos a guardar

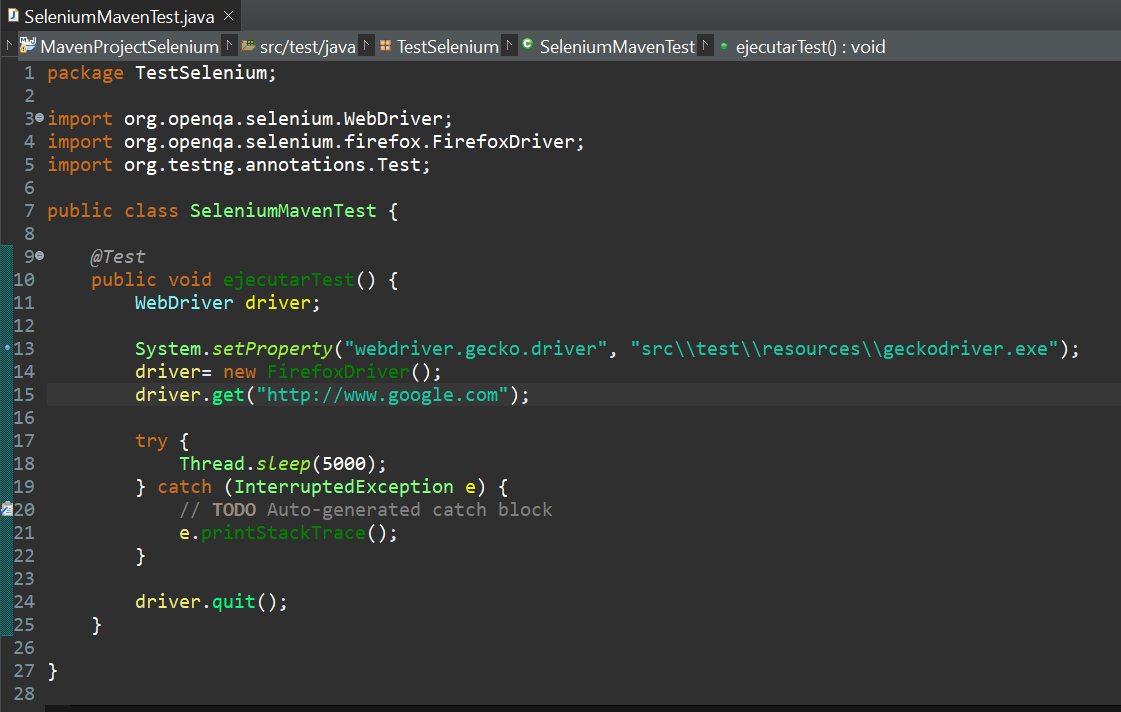


Después creamos una nueva clase sin main



Dentro de la clase crear un método public con una etiqueta @Test y dentro del método el codigo

Ejemplo:



Por último ejecutamos con el Maven que hemos creado anterior mente



Hay que recordar el driver del navegador en este caso al ser ejecutado en Firefox se llama geckodriver.exe

**Explicación del código:**

****

1. **Declaración de la variable driver:**

javaCopy code

WebDriver driver;

Se declara una variable llamada **driver** del tipo **WebDriver**, que es una interfaz en Selenium utilizada para la interacción con los navegadores web.

1. **Configuración del controlador de Gecko (para Firefox):**

javaCopy code

System.setProperty("webdriver.gecko.driver", "src\\test\\resources\\geckodriver.exe");

Se establece la propiedad del sistema "webdriver.gecko.driver" con la ubicación del controlador de Gecko (geckodriver.exe). Este controlador es necesario para que Selenium pueda interactuar con el navegador Firefox.

1. **Inicialización del objeto driver como un nuevo FirefoxDriver:**

javaCopy code

driver = new FirefoxDriver();

Se instancia un objeto de la clase **FirefoxDriver** y se asigna a la variable **driver**. Esto crea una nueva instancia del navegador Firefox.

1. **Abrir la página web de YouTube:**

javaCopy code

driver.get("http://www.youtube.com");

Se utiliza el método **get** de **driver** para abrir la URL especificada en el navegador. En este caso, la URL es "[http://www.youtube.com](http://www.youtube.com/)".

1. **Esperar 5000 milisegundos (5 segundos):**

javaCopy code

try { Thread.sleep(5000); } catch (InterruptedException e) { e.printStackTrace(); }

Se utiliza un bloque **try-catch** para manejar la excepción **InterruptedException**. Durante la ejecución, el hilo actual (en este caso, el programa) se duerme (o pausa) durante 5000 milisegundos (5 segundos). Esto se hace para esperar que la página cargue completamente antes de continuar con el código.

1. **Cerrar el navegador al finalizar:**

javaCopy code

driver.quit();

Se utiliza el método **quit** de **driver** para cerrar el navegador y liberar los recursos asociados.

En resumen, este código automatiza la apertura del navegador Firefox, navega a la página de YouTube, espera 5 segundos y luego cierra el navegador. Este es un ejemplo simple de automatización de pruebas o de interacción automatizada con un sitio web.