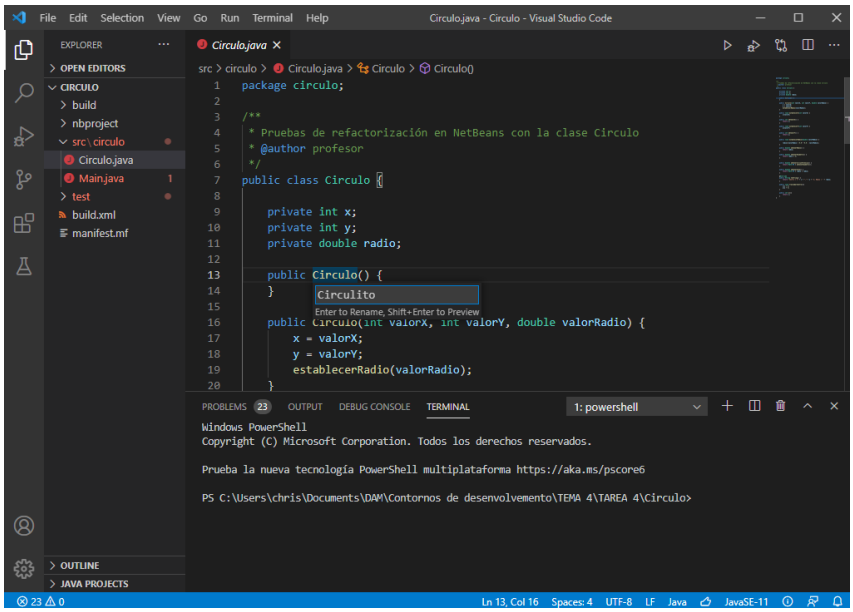
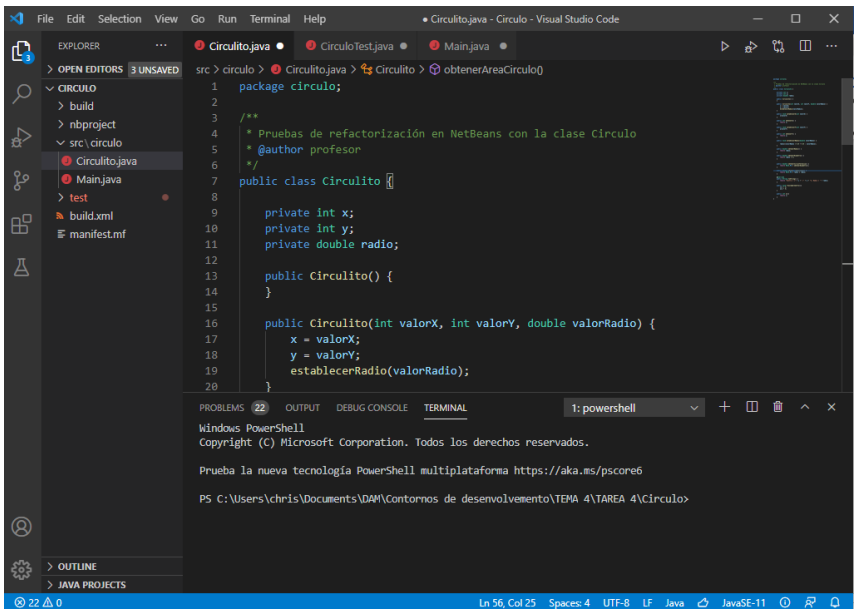
Ejercicio 1. Refactorización.

- Renombrar la clase Circulo por Circulito.

Nos situamos sobre la clase Circulo, pulsamos el *botón derecho* y seleccionamos la opción **Rename Symbol**. Introducimos el nuevo nombre (Circulito) y confirmamos los cambios con *Enter*.

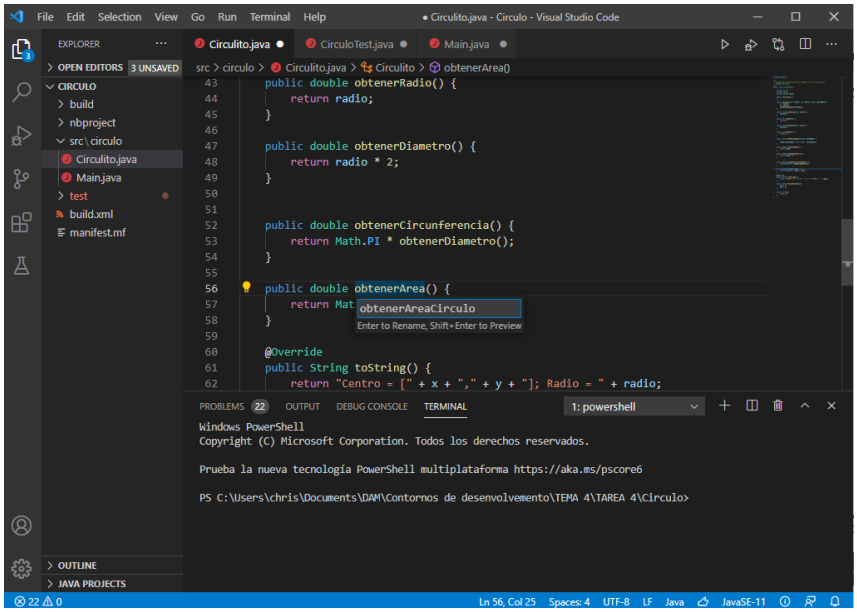


En la siguiente imagen se aprecia la modificación en el código.

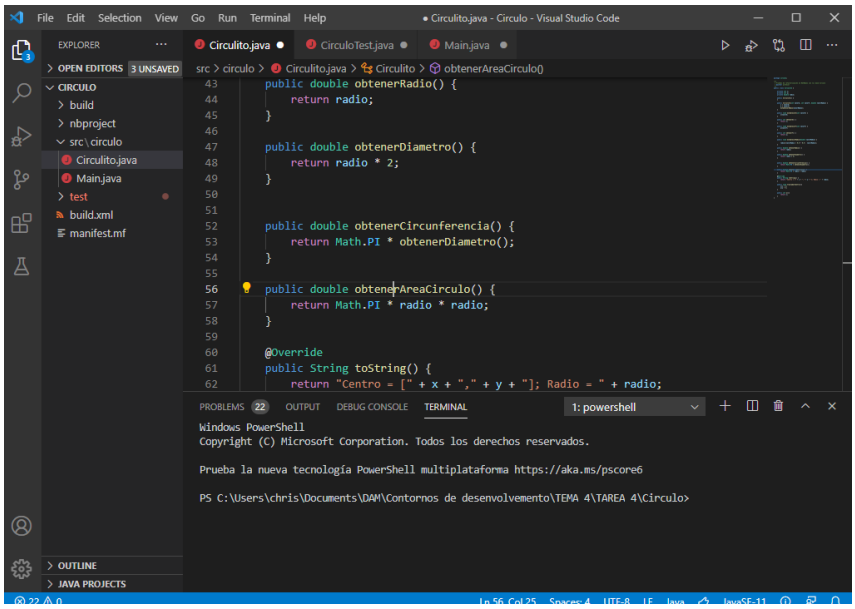


- Renombrar el método ObtenerArea por ObtenerAreaCirculo.

El procedimiento es idéntico al seguido en el paso anterior. Nos situamos sobre el método ObtenerArea, pulsamos el *botón derecho* y seleccionamos la opción **Rename Symbol**. Introducimos el nuevo nombre (ObtenerAreaCirculo) y confirmamos los cambios con *Enter*.

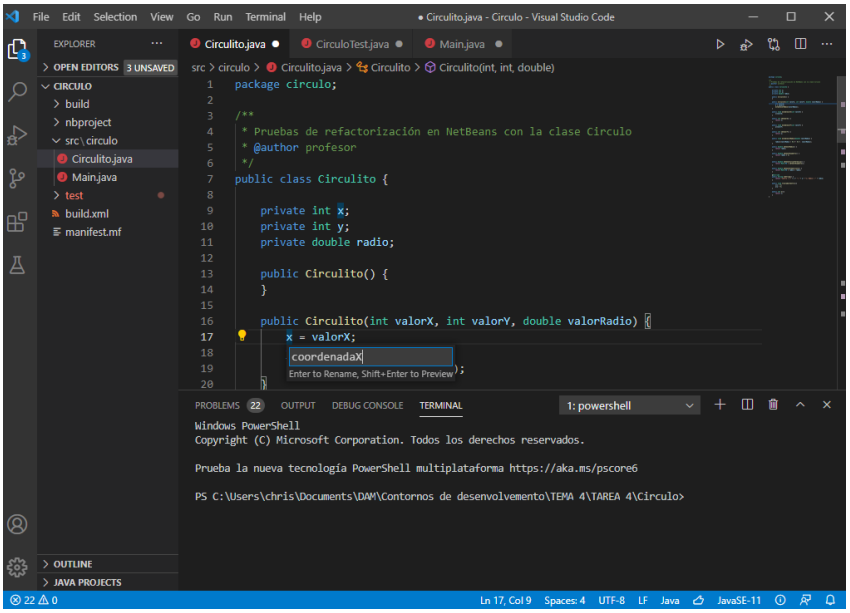


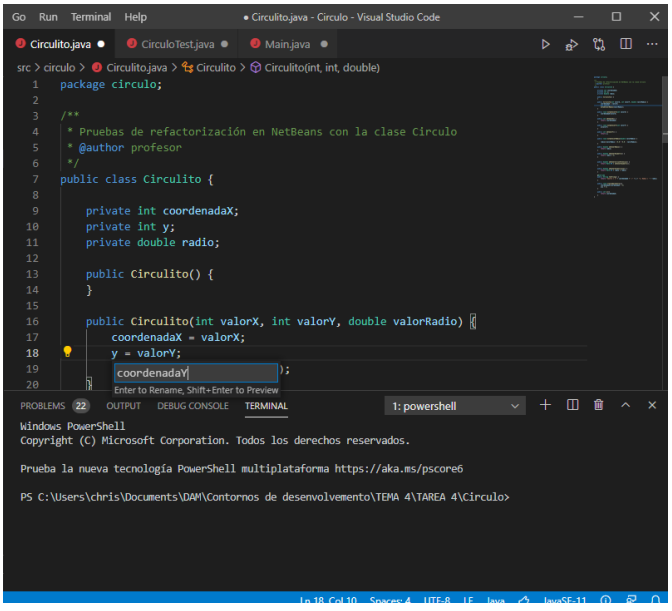
En la siguiente imagen se aprecia la modificación en el código.



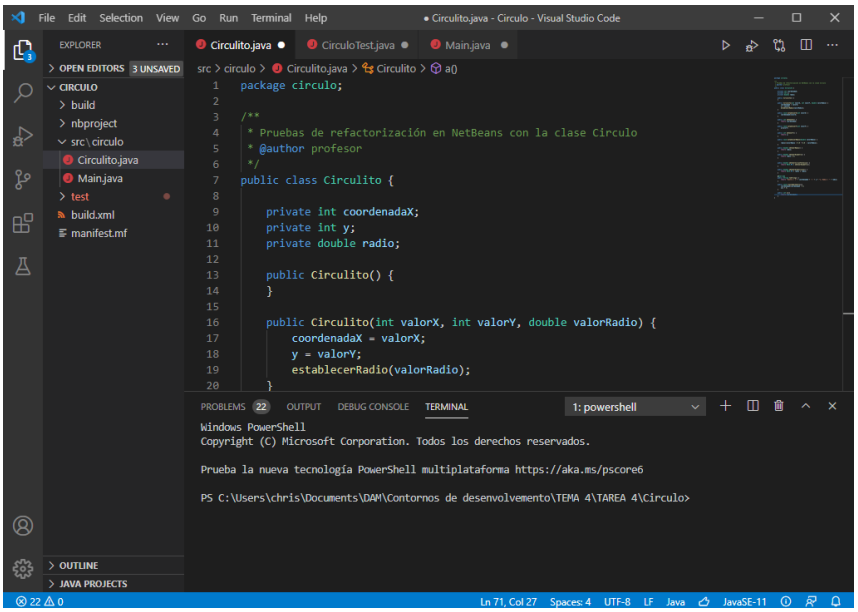
- Renombrar los campos x e y por coordenadaX y coordenadaY.

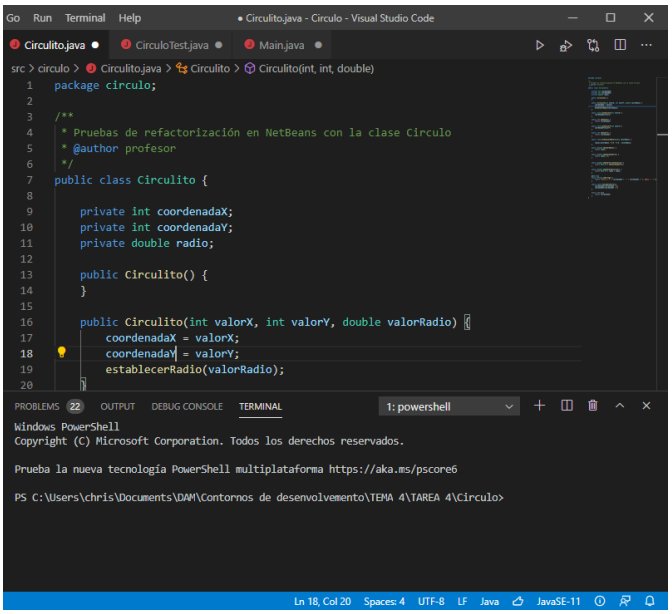
De la misma forma que en los casos anteriores, nos situamos sobre los campos x e y, pulsamos el *botón derecho* y seleccionamos la opción **Rename Symbol**. Introducimos el nuevo nombre (coordenadaX y coordenadaY, respectivamente) y confirmamos los cambios con *Enter*.





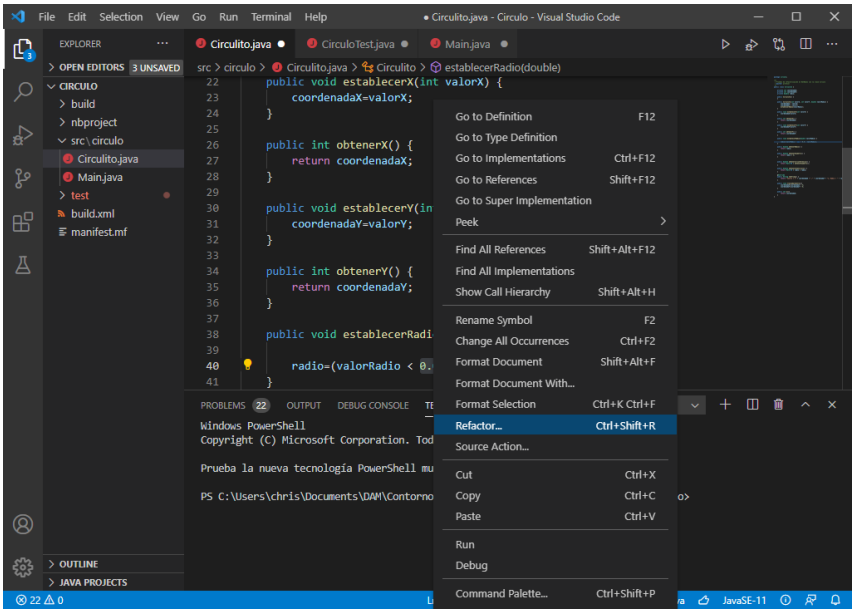
En las siguientes imágenes se aprecia la modificación en el código.

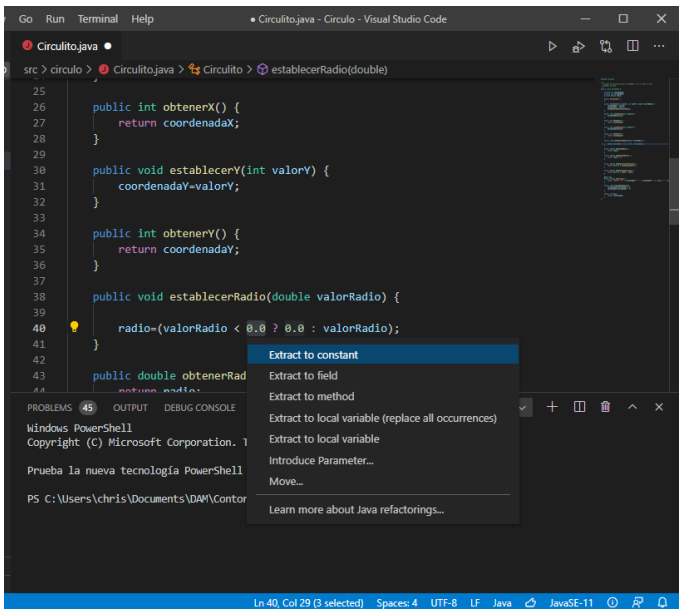


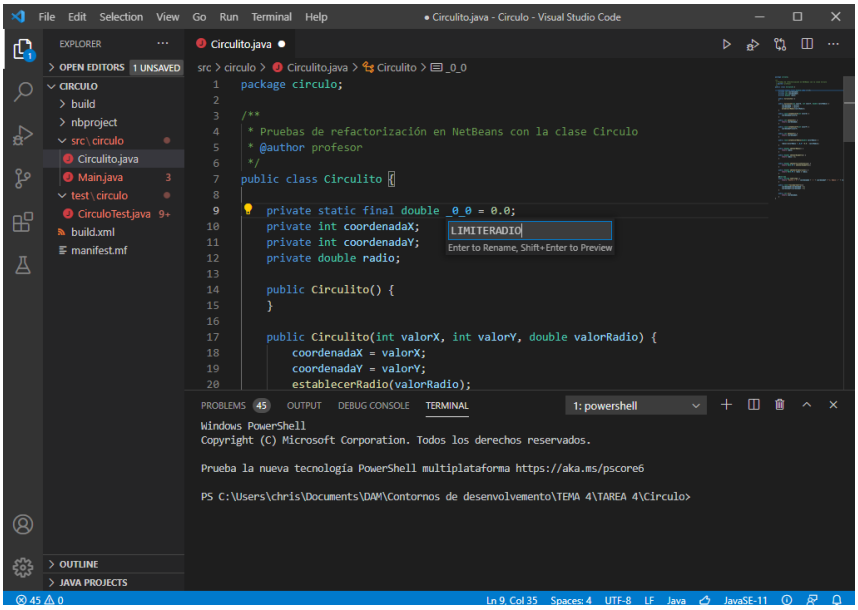


- Introducir la constante LIMITERADIO de tipo doble con valor 0.0

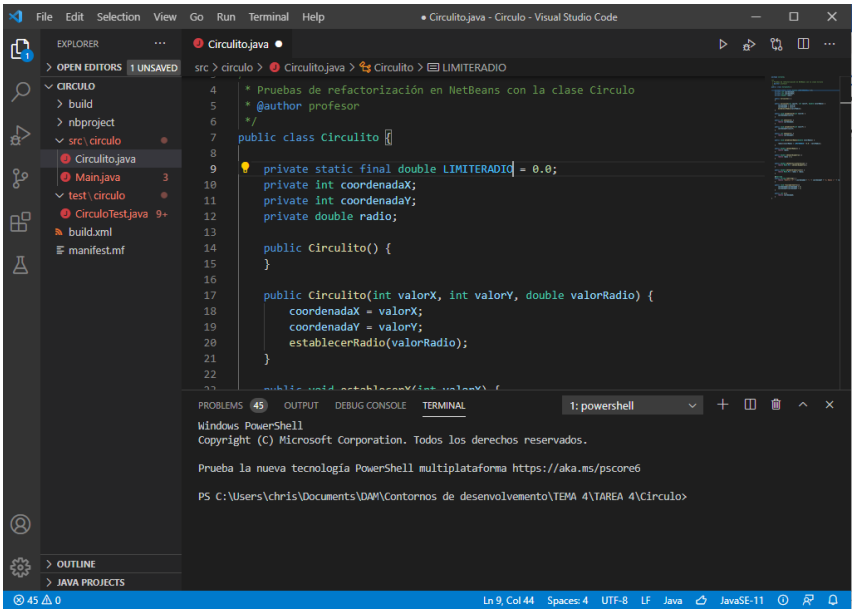
Nos situamos sobre el valo 0.0, pulsamos el *botón derecho* y seleccionamos la opción **Refactor**. Escogemos la opción **Extract to constant** e introducimos el nombre de ésta (LIMITERADIO). Confirmamos los cambios con *Enter*.

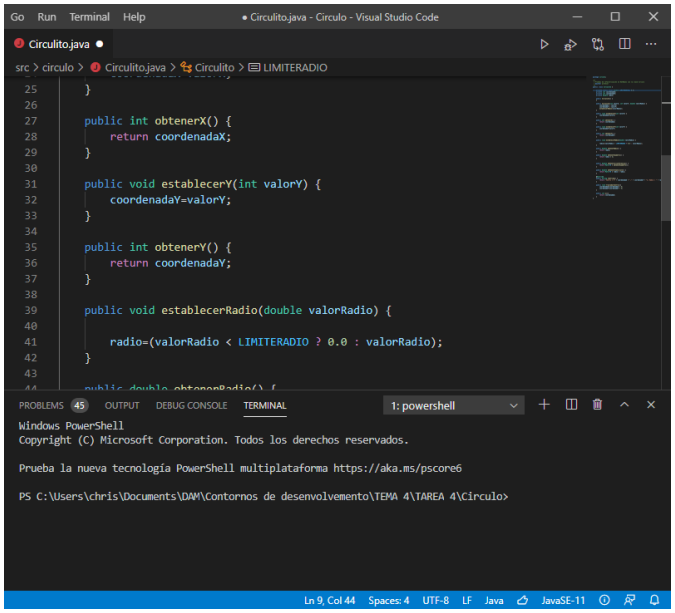






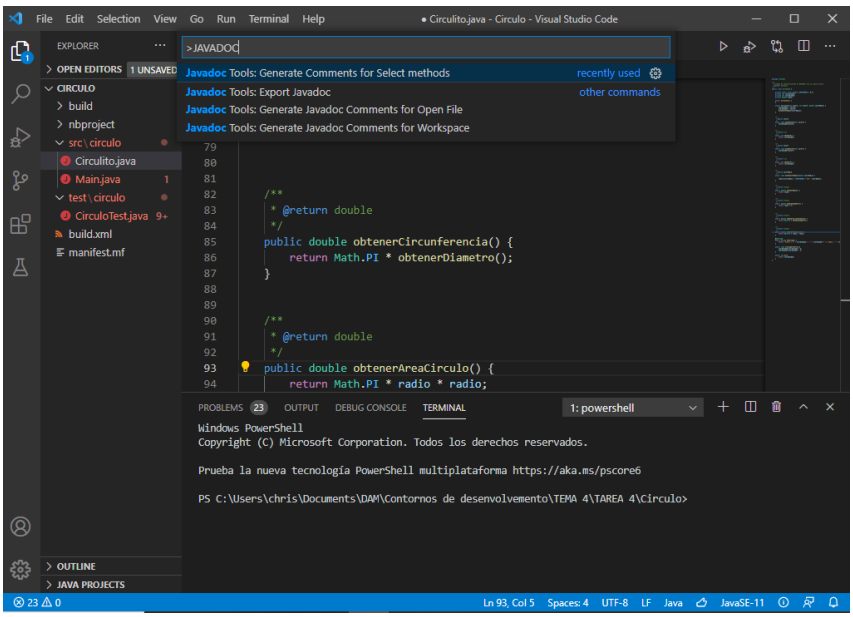
En las siguientes imágenes se aprecia la modificación en el código.

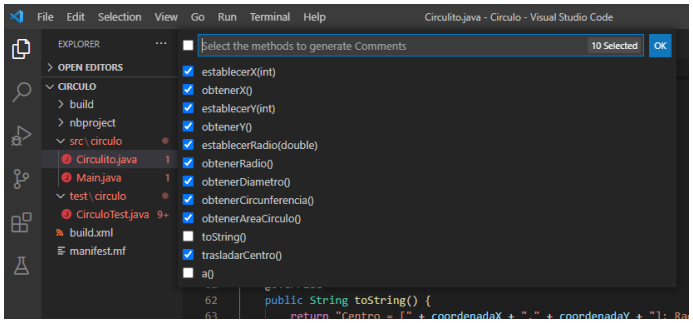




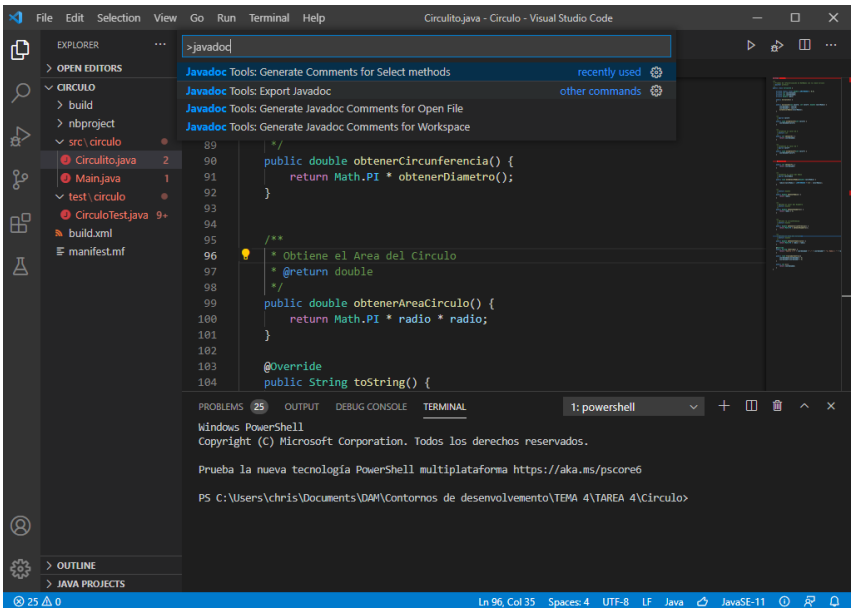
Ejercicio 2. JavaDoc.

Para documentar el proyecto, dejamos comentarios que describan las funciones de cada método, clase, etc. Para ello, desplegamos la paleta de comandos e introducimos Javadoc. Aparecerán varias opciones y seleccionaremos **Generate commentes for select methods**. Después escogemos los métodos que vamos a comentar.

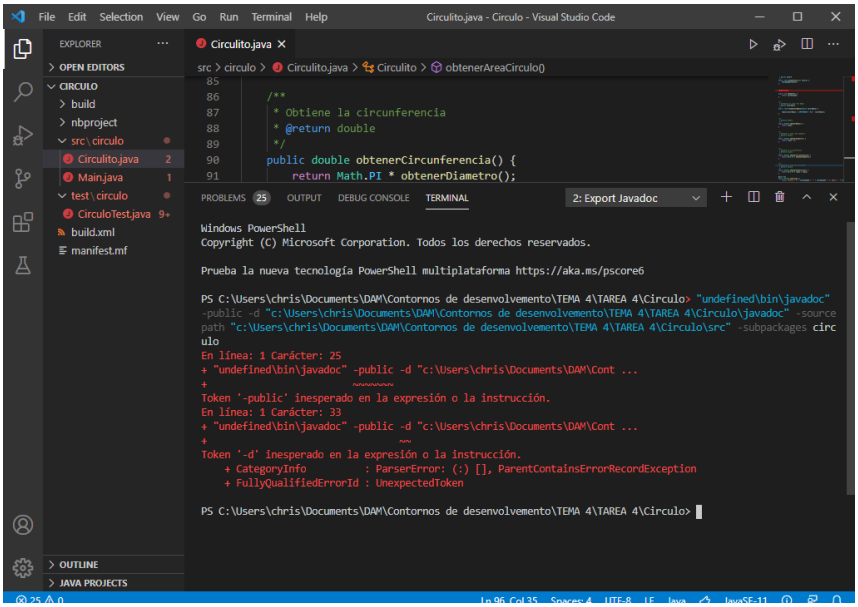




Para generar la página web asociada, volvemos a introducir Javadoc en la paleta de comandos. Esta vez seleccionaremos la opción **Export Javadoc** (la segunda de la siguiente imagen).



***NOTA:** en mi caso da un error que se adjunta en la siguiente imagen.



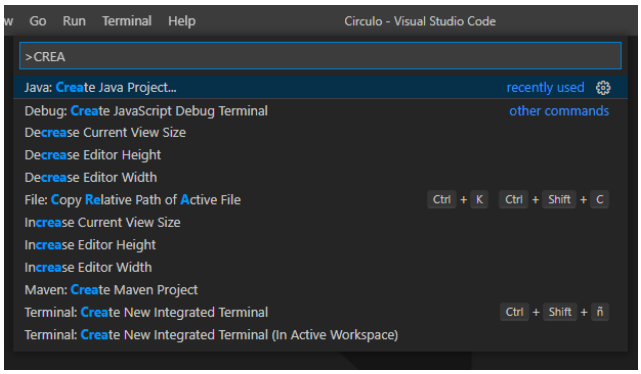
Ejercicio 3. SonarQube.

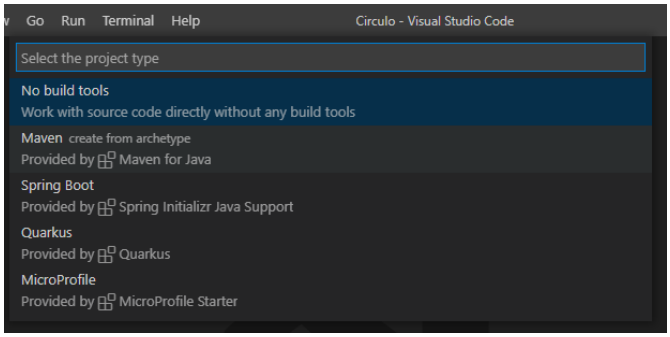
Abrimos la aplicación StartSonar.bat en la ruta sonarqube-8.6.1.40680\bin\windows-x86-64. Al hacerlo, se ejecutará en el procesador de comandos. Una vez que finalice, iniciamos sesión en http://localhost:9000/ con las credenciales de administrador de sistema:

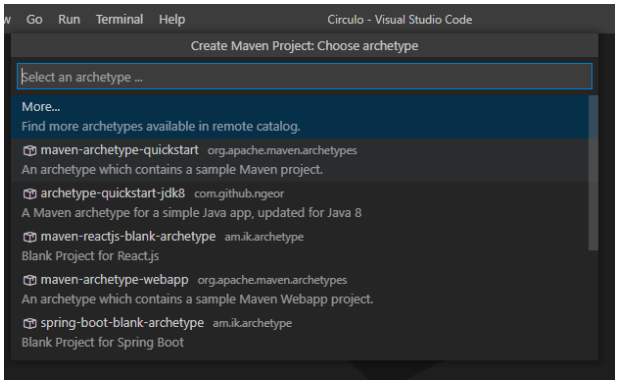
- iniciar sesión: admin
- contraseña: admin

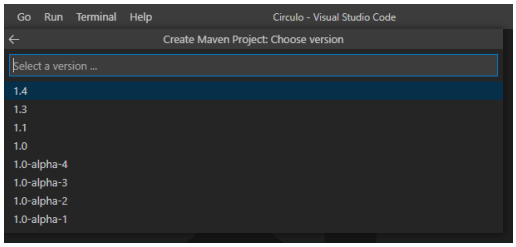
Nos pedirá cambiar la contraseña y después podremos crear un proyecto en la URL anterior.

Ahora, en Visual Studio Code, desplegamos la paleta de comandos e introducimos **Create Javadoc Project**. En el siguiente menú de opciones que aparecerá, seleccionamos **Maven** y más adelante **quickstart**.









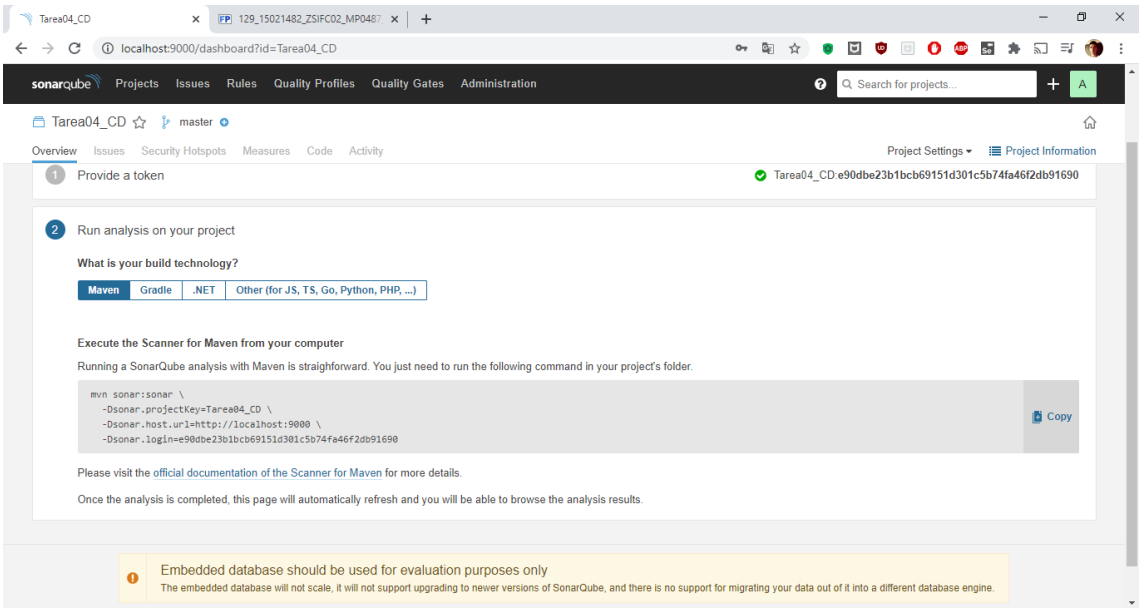
***NOTA:** llegados a este punto debería ejecutar una serie de comandos y solicitar el nombre del proyecto de SonarQube creado en la URL anterior, sin embargo me saltó el siguiente error:

Error: JAVA_HOME not found in your environment.

Please set the JAVA_HOME variable in your environment to match the

location of your Java installation.

De no saltar ese error, al finalizar la ejecución de comandos anterior, escribiríamos “mvn clean package”.



Se copiaría el código de la página del proyecto SonarQube y se ejecutaría. Al finalizar proporciona una URL donde se puede comprobar el estado de calidad del software.

Mostraría algo similar a la siguiente imagen (extraída del vídeo correspondiente a SonarQube de la lección), donde nos dice si la calidad está pasada. Seleccionando la acción Code Smells aparece un listado de aquello que es mejorable en el código