



Práctica Debugg en NetBeans

Entornos de desarrollo

Álvaro Cañizares Ramallo
DNI: 70427930K

Nov/2019



Índice:

1.Instalación pg2

2.Depuración NetBeans pg5

3.Depuración de codigo pg7

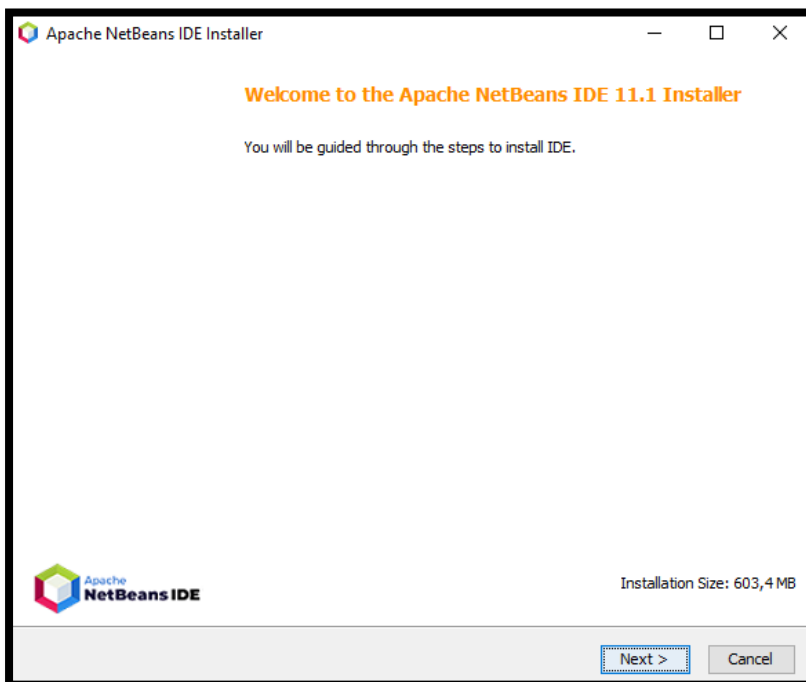
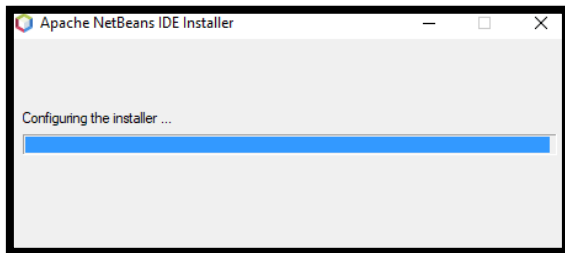
4.NetBeans vs Eclipse pg9

5.Conclusiones pg10

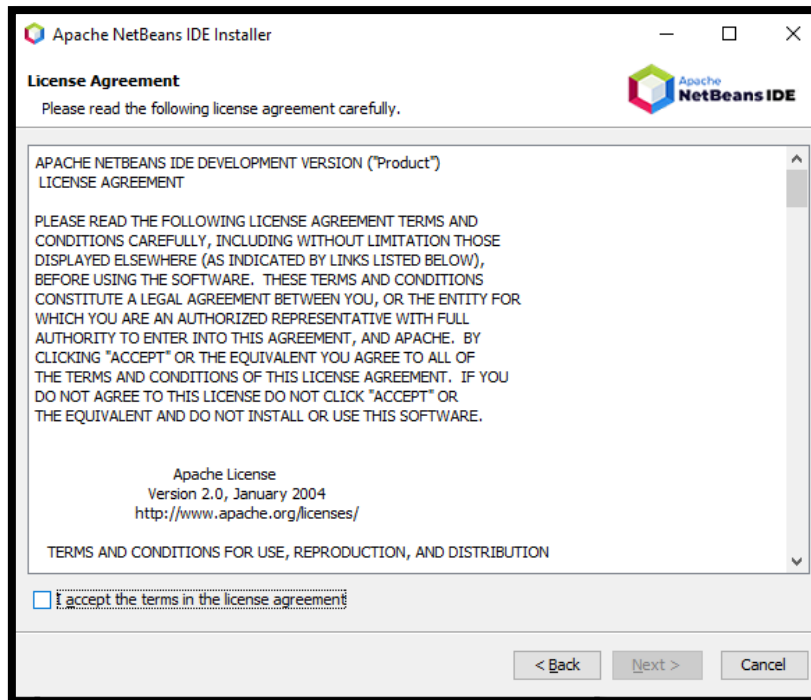
6.Bibliografía pg10

1.Instalación

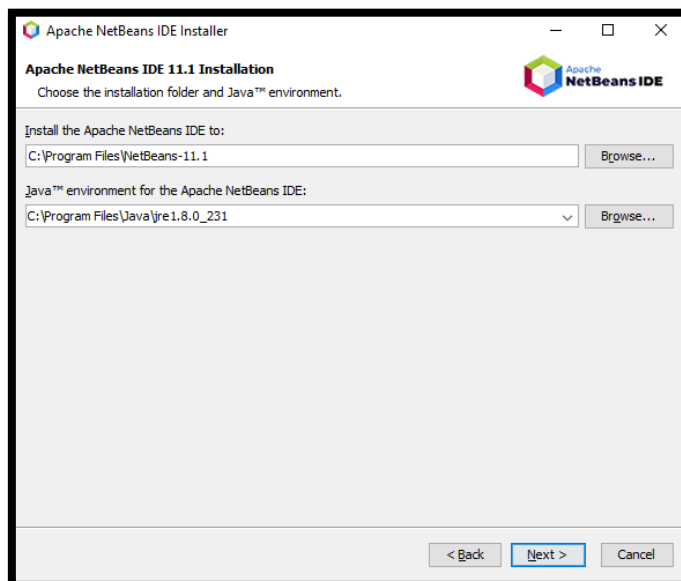
A la hora de instalar NetBeans, se elige la versión compatible con el SO, en este caso x64. Al clicar en el ejecutable, emerge estas imagenes.



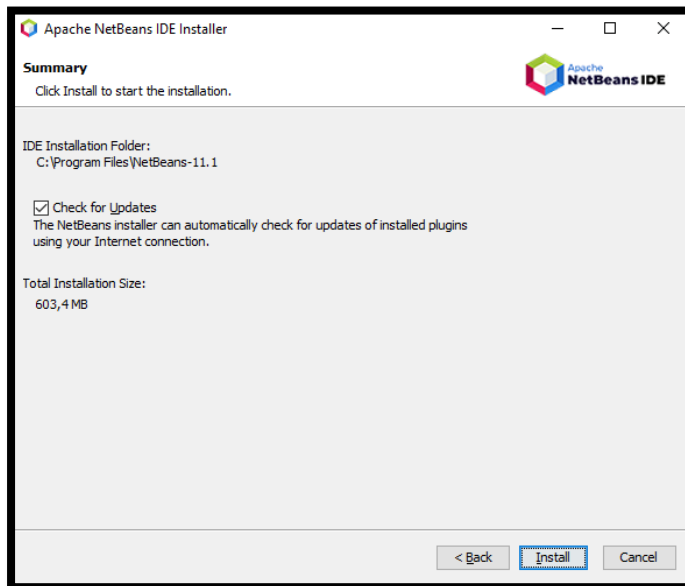
Esta pantalla nos muestra el inicio del instalador, la versión de NetBeans y el espacio total del programa.



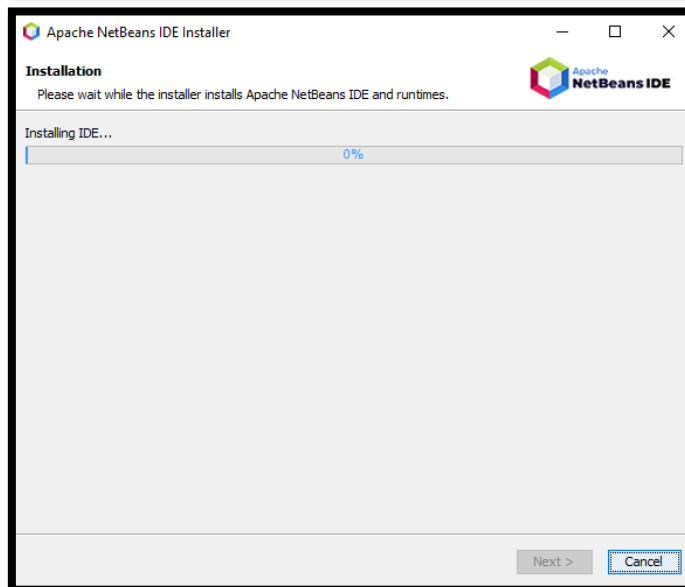
Posteriormente nos encontramos con la licencia de acuerdo, así accedemos a las condiciones de uso del sistema, se marca la opción y se activará la casilla "Siguiente".



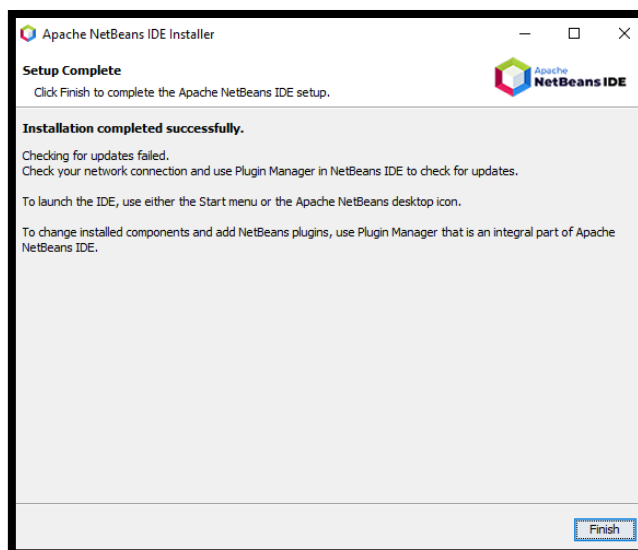
Después nos preguntará donde queremos instalar el programa y donde se encuentra el entorno java instalado.



En esta ventana se permite seleccionar si queremos o no que busque actualizaciones de manera automática.



Después comenzará a instalar el programa. Dependiendo de si hemos clicado o no la opción de actualizar, tendremos después otra ventana con este proceso.



Al terminar nos mostrará un mensaje que nos informa de como ejecutar el programa.

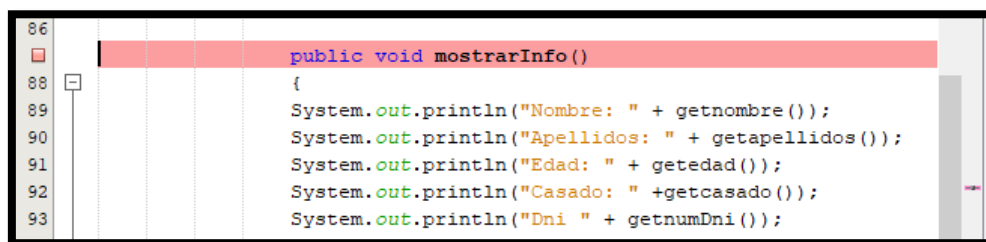
2. Depuración en NetBeans

A la hora de depurar, se han de tener una serie de factores en cuenta. A pesar de que existe compatibilidad entre NetBeans y Eclipse, lo cual permite importar y exportar los proyectos, paquetes y clases, pueden surgir una serie de problemas que impidan la ejecución correcta de NetBeans. Para instalar NetBeans hace falta descargar una versión compatible de JDK con la versión del programa que se quiera usar, en este caso se ha usado NetBeans 11.2 y JDK 13.0.1, si no se descarga el JDK, el paquete JRE de java que usa Eclipse no será suficiente para ejecutar NetBeans, por lo que, aunque se abra el programa, no tendrá funcionalidad en java.

Existen tres opciones a la hora de usar NetBeans, crear un proyecto nuevo y escribir el código de nuevo, copiarlo o importarlo (esta opción nos deja además guardar los cambios en el workspace de Eclipse, lo que permite modificar el archivo desde distintos IDEs teniendo siempre la última versión o, por el contrario, guardarlo en el workspace propio de NetBeans, en este caso se ha optado por importar el proyecto.

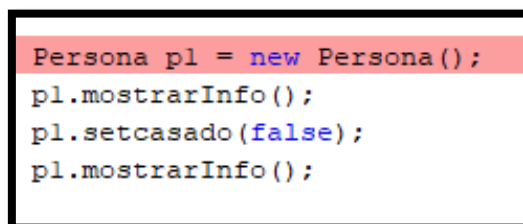
Para preparar el debugging se han establecido los mismos breakpoint clave que en la práctica con Eclipse:

-Tipo línea

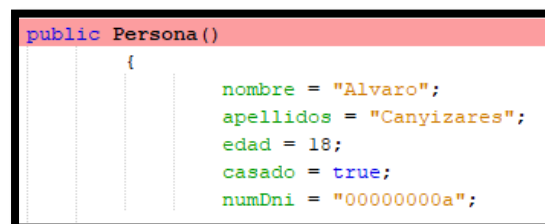


```
86  
87 public void mostrarInfo()  
88 {  
89     System.out.println("Nombre: " + getnombre());  
90     System.out.println("Apellidos: " + getapellidos());  
91     System.out.println("Edad: " + getedad());  
92     System.out.println("Casado: " + getcasado());  
93     System.out.println("Dni " + getnumDni());
```

-Tipo clase

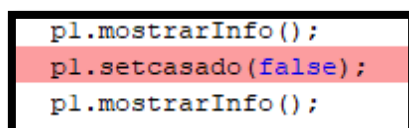


```
Persona pl = new Persona();  
pl.mostrarInfo();  
pl.setcasado(false);  
pl.mostrarInfo();
```



```
public Persona()  
{  
    nombre = "Alvaro";  
    apellidos = "Canyizares";  
    edad = 18;  
    casado = true;  
    numDni = "00000000a";
```

-Tipo campo



```
pl.mostrarInfo();  
pl.setcasado(false);  
pl.mostrarInfo();
```

-Tipo método

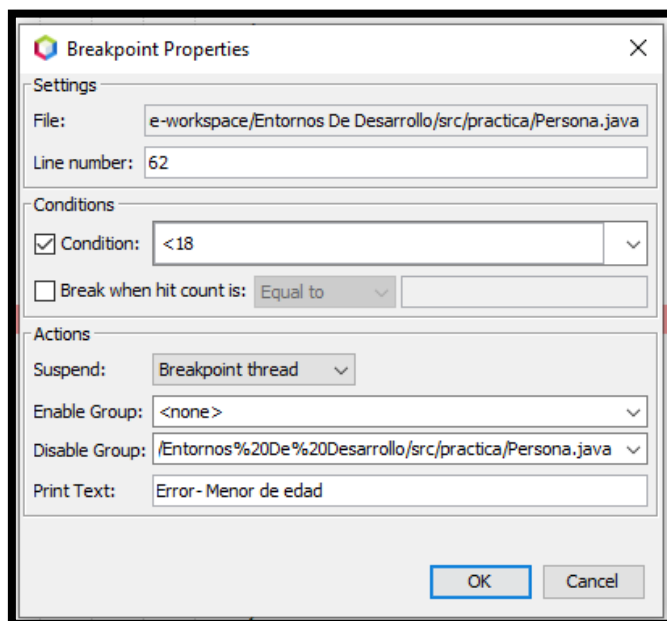
```
pl.mostrarInfo();  
pl.setcasado(false);  
pl.mostrarInfo();
```

```
public void mostrarInfo()  
{  
    System.out.println("Nombre: " + getnombre());  
    System.out.println("Apellidos: " + getapellidos());  
    System.out.println("Edad: " + getedad());  
    System.out.println("Casado: " + getcasado());  
    System.out.println("Dni " + getnumDni());  
}
```

En al menos alguno, en las propiedades de los breakpoints definiremos:

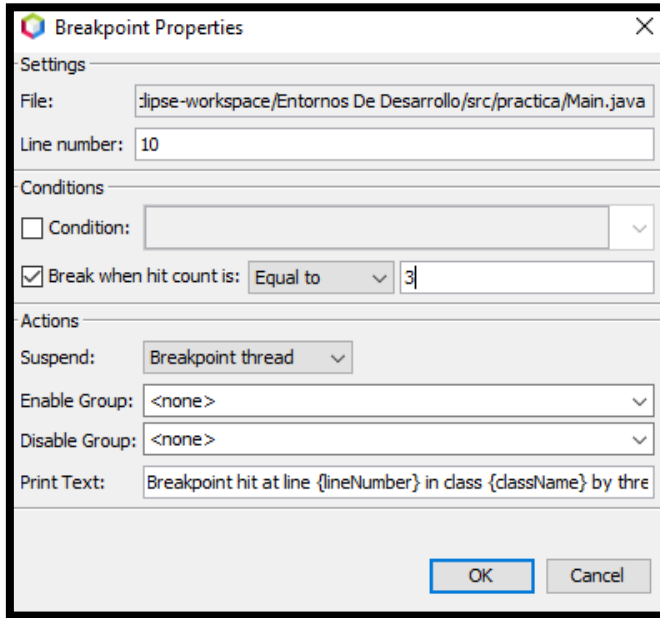
- Condiciones para que el programa se detenga al alcanzar las variables ciertos valores

```
public void setedad(int _edad)  
{  
    edad = _edad;  
}
```



- Se alcance un número de veces (hits), determinado.

```
pl.mostrarInfo();
pl.setcasado(false);
pl.mostrarInfo();
```



Parte 3: Depuración de código

Se ejecutará y se depurará el código adjunto a esta tarea (*comprensión de que hace, no de como lo hace*), se deberá:

1) Leer y analizar el código facilitado. Primeras conclusiones.

Ejercicio 2

Al iniciar el código, se definen las variables y se pide que se introduzca el nº de personas, siempre y cuando sea positivo, nos dejará continuar. Luego se genera un array de la variable alto que almacenará la lectura de las personas de manera individual. Finalmente, tras dos cálculos mostrará en pantalla la estatura media y la cantidad de gente por encima y debajo de esta.

2) Ejecutar de forma normal dicho código. Obtenemos y analizamos los resultados obtenidos.

Se ha probado el código con N = 5 personas, posteriormente, al recoger la quinta altura surge un error en la creación de arrays, donde estos se encuentran desbordados.

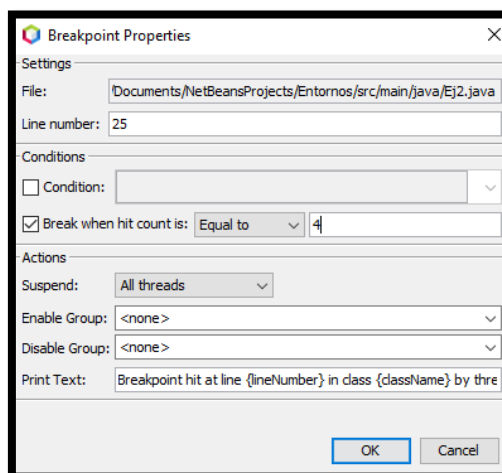

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
    at Ej2.main(Ej2.java:31)
Command execution failed.
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)
```

- 2) Depuramos, aplicando todo lo aprendido sobre los puntos rupturas (tipos y propiedades) este código. Tendremos que obtener las capturas necesarias, junto a los resultados y conclusiones obtenidas.

Dado que el error inicial ya comenta que el se debe a que el código pide más arrays de los que ha generado anteriormente a la hora de verificar en el bucle if.

```
if (alto[i] > media)
{
    contMas++;
} else if (alto[i] > media)
{
    contMenos++;
}
```

He puesto un breakpoint para que el proceso de debuggin sea manual desde la cuarta repetición de alto [i], para ver el fallo manualmente.



Dado que simplemente es un error de inicialización, el error del código se soluciona fácilmente modificando la cantidad de personas, de manera que se ajuste al conteo del array (desde 0 y no desde 1), quedando así:

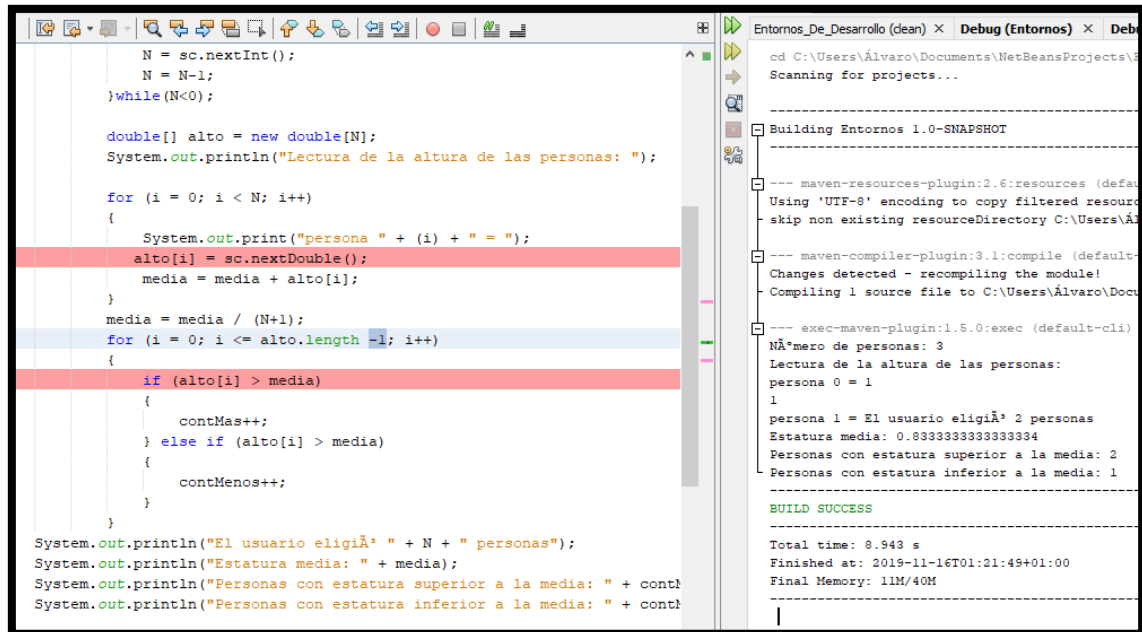
```
media = media / (N+1);
for (i = 0; i <= alto.length; i++)
{
    if (alto[i] > media)
```

```

media = media / (N+1);
for (i = 0; i <= alto.length -1; i++)
{
    if (alto[i] > media)
    {
        contMas++;
    } else if (alto[i] > media)

```

Finalmente, queda resuelto el fallo.



Parte 4: NetBeans vs Eclipse

Se pide profundizar (**que no copiar**) en las características de NetBeans, sus puntos fuertes y débiles con respecto a otros IDEs como Eclipse. Para ello, como *mínimo*, deberemos aportar:

1) Historia y evolución.

Netbeans surge del proyecto Xelfi, donde se pretendía generar el equivalente al IDE del lenguaje Delphi en java, dando nacimiento en 1997 y viendo la luz comercial dos años después como software de código abierto a través de Sun Microsystems.

2) Lenguajes soportados.

Netbeans soporta Java, PHP, Groovu, C/C++, HTML5.

3) Plugings, tipos e instalación.

Una de las características que definen a Netbeans es su código abierto, y por tanto, la posibilidad de implementar plugings de manera sencilla y pública, de estos pueden destacar, entre muchos otros easyUML, que permite la generación de unos esquemas donde se ven cómo se unen las clases, cómo heredan, agregan ,etc, Darcula LAF, que permite modificar la apariencia del navegador para ser más agradable a la vista e incluso proporcionar un tema oscuro o GIT Toolbar, que permite crear una barra de herramientas más accesibles con los Git más comunes.

Para instalarlo únicamente hay que tener descargada una versión de JDK compatible con la versión de Netbeans a usar

4) Principales características, ventajas e inconvenientes.

De las principales características que tiene NetBeans, a parte de la gestión de interfaz de usuario, almacenamiento y ventana, aporta un marco asistente o debugger, una librería visual y herramientas de desarrollo integrado.

5) Depuramos, aplicando todo lo aprendido sobre los puntos rupturas (tipos y propiedades) este código. Tendremos que obtener las capturas necesarias, junto a los resultados y conclusiones obtenidas.

Cabe destacar, que, a la hora de usarlo tras Eclipse, lo encuentro mucho menos intuitivo, con una interfaz menos amigable y especialmente más lento, aunque los informes de errores eran mucho más útiles y específicos que en Eclipse, así como los consejos (como añadir el import de Scanner, si no estaba escrito), que lo hacen más transparente que Eclipse. Cabe destacar la importancia de la herramienta de medición de recursos, que puede resultar muy útil en caso de generar programas a un nivel superior.

*NOTA: Las capturas se pueden ver a lo largo del trabajo.

Parte 5: Conclusiones

A la hora de realizar la práctica, me he sentido bastante perdido, habría deseado una interfaz más intuitiva, tal y como ofrece Eclipse. No he tenido la oportunidad de poder usarlo en profundidad, así que mi valoración es puramente subjetiva, pero parece una herramienta más especializada para el desarrollador experto, dada la abrumadora cantidad de opciones y configuraciones que hay que realizar incluso para comenzar un proyecto nuevo.

Sin embargo, le encuentro un punto a favor que le hace competir fuertemente contra Eclipse y se basa en su código abierto y comunidad, dado que es posible que en algún momento Java se vuelva un lenguaje obsoleto, esta capacidad de Netbeans hace que se vuelva mucho más fácil de adaptar a los futuros tiempos y usuarios.

Parte 6: Bibliografía

<https://es.wikipedia.org/wiki/NetBeans>
<http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/888>
<http://andriujavanetbeans.blogspot.com/2017/02/caracteristicas-de-java-netbeans.html>
<https://blog.megacursos.com/2018/05/neatbeans-y-sus-caracteristicas/>
<https://www.12caracteristicas.com/netbeans/>
<https://blog.idrsolutions.com/2017/04/top-5-netbeans-ide-plugins/>
<https://medium.com/issuehunt/10-best-netbeans-plugins-for-developers-496bc9fe8a90>
<http://plugins.netbeans.org/>
<https://www.genbeta.com/desarrollo/netbeans-1>