

NetBeans

Introducción	1
Instalación	1
Documentación Java	3
Introducción a NetBeans	3
Atallos de NetBeans	6
Executar unha aplicación Java	7
Depurar o código	8
Distribución de aplicacións	11
Referencias	12

Introdución

NetBeans é un IDE (*Integrated Development Environment*) de código aberto creado principalmente para a linguaxe Java, aínda que tamén pode ser usado para outras linguaxes de programación. Permite crear aplicacións de escritorio, web e para dispositivos móbiles. Ten un número importante de módulos que permiten estendelo e ademais é un produto libre e gratuito sen restricións de uso.

Instalación

NetBeans está escrito en Java, polo que na máquina onde se vaia usar, ten que estar instalado JDK.

JDK (*Java Development Kit*) é un conxunto de ferramentas de desenvolvemento para construír aplicacións usando a linguaxe de programación Java. O JDK inclúe ferramentas útiles como o compilador (javac), debugger e outras ferramentas para desenvolver e testar programas escritos en Java.

Unha instalación do JDK tamén inclúe a instalación do JRE (*Java Runtime Environment* - Contorna de Execución en Java). O JRE contén a JVM (*Java Virtual Machine*) e outras ferramentas que permiten a execución das aplicacións Java. JRE, pola contra, non posee compiladores nin ferramentas para o desenvolvemento de aplicacións Java, só posee as ferramentas para executalas.

NOTA de [Java releases](#): Java 17 é a última versión LTS de Java. O 19 de setembro de 2023 está planeado o lanzamento de Java 21 que tamén é LTS.

OpenJDK é unha implementación de JDK gratuita, de código aberto e licenza GPL de GNU. Hai outras moitas versións con licenzas de pago doutros provedores (Oracle, Microsoft, AWS, etc).

NOTA: Oracle JDK 17 e as versións posteriores serán gratuítas, incluso para uso comercial e en produción, de acordo coa licenza de [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

OpenJDK pode descargarse da páxina <https://jdk.java.net/> e instalala. En Linux normalmente tamén está nos repositorios e pode instalarse a versión predeterminada que estes inclúen seguindo as [instrucións de instalación de Java en Ubuntu 22.04](#).

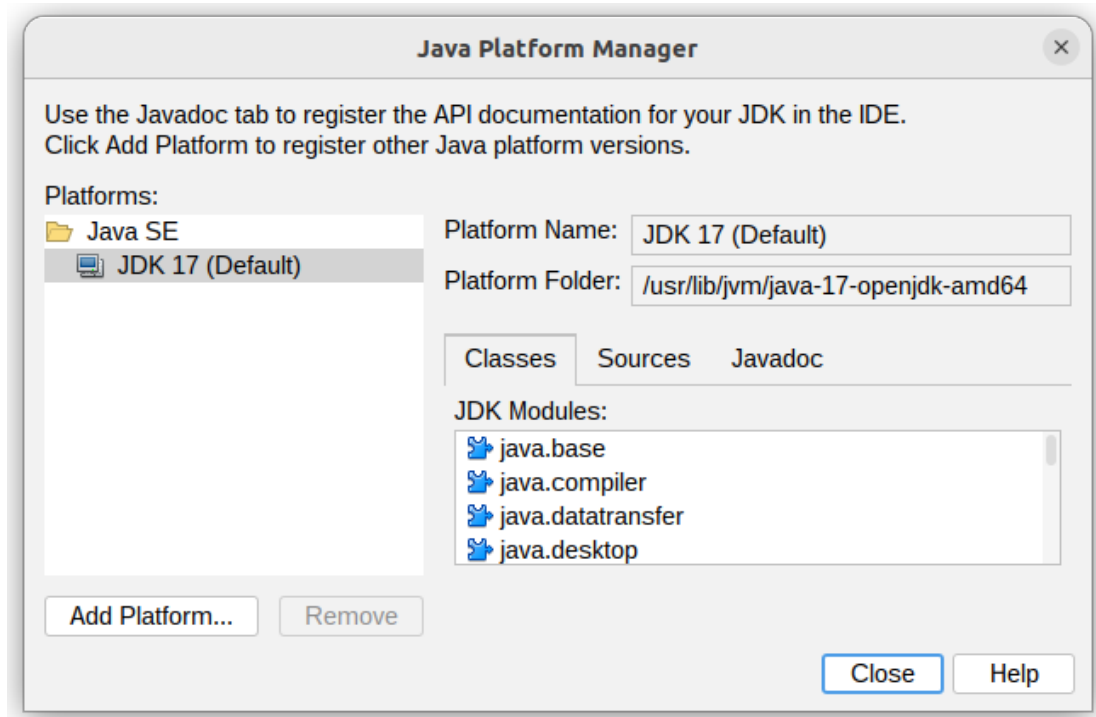
Unha vez configurado, comprobar cal é a versión instalada de java co comando

```
java -version
```

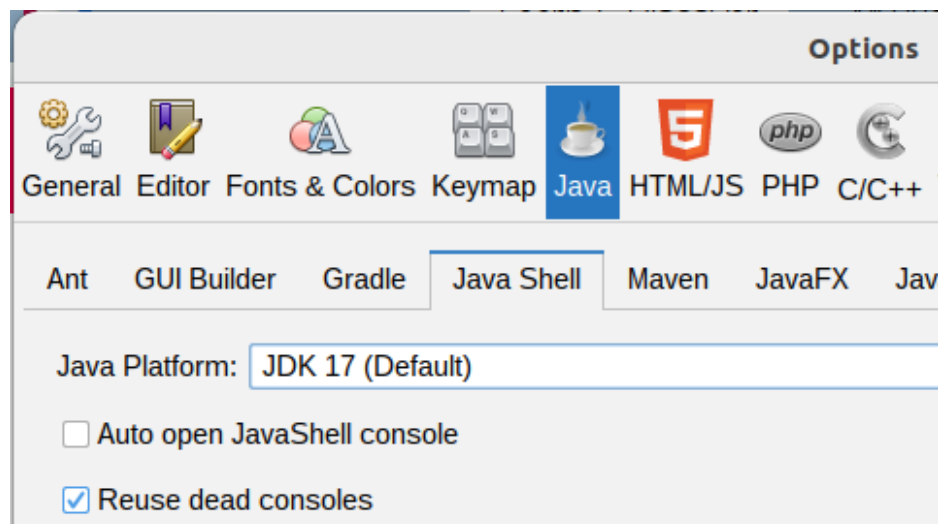
A versión actual de [NetBeans](#) é a 19. Para instalalo pode:

- Descargarse o ficheiro co instalador para o sistema operativo apropiado.
- En Linux tamén pode instalarse usando o paquete [snap](#) ou un [instalador](#).

Unha vez instalado NetBeans, hai que comprobar as plataformas de Java que detecta instaladas no sistema operativo e seleccionar a que se usará por defecto. Para iso hai que acceder ao menú **Tools -> Java Platforms**



Hai que comprobar tamén a versión de java que se usará. Para iso hai que acceder ao menú **Tools -> Options -> Java -> Java Shell**.



NOTA: para modificar o JDK por defecto que ten configurado NetBeans, hai que modificar a variable **netbeans_jdkhome** establecida no ficheiro de configuración **/opt/netbeans/netbeans/etc/netbeans.conf**. Se non se teñen permisos para modificar este ficheiro, é posible indicar que versión de java usar ao lanzar NetBeans usando o seguinte comando:

```
alias nb='netbeans --jdkhome "/usr/lib/jvm/java-17-openjdk-amd64"&'
```

Documentación Java

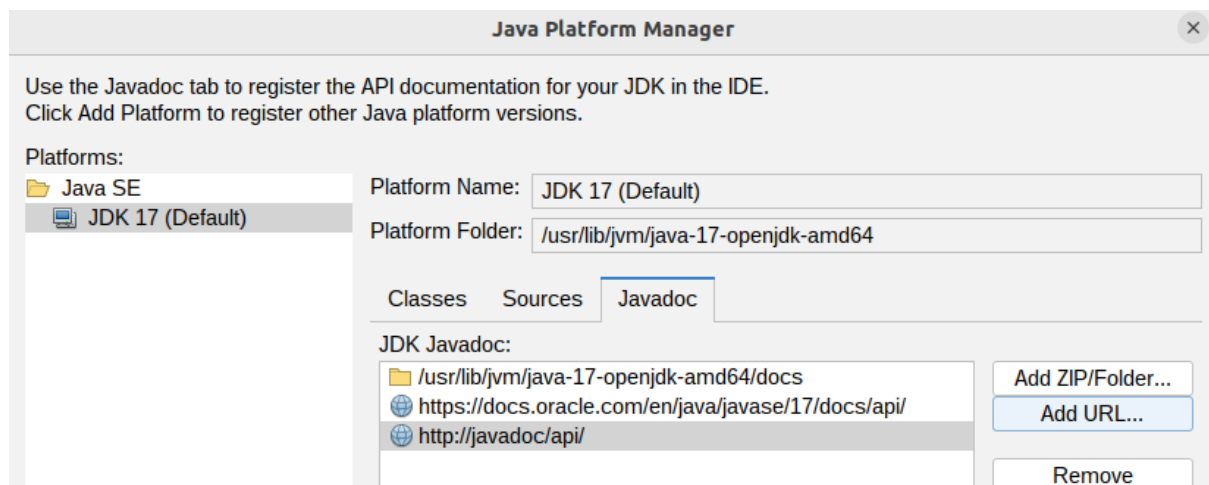
Para ter acceso á documentación de Java dende NetBeans mentres se codifica, primeiro haberá que instalala co comando:

sudo apt install openjdk-17-doc

Tamén é posible engadir a ruta a dita documentación dende o menú **Tools -> Java Platforms**

No centro educativo temos a documentación dispoñible dende <http://javadoc/api/>.

Tamén sería posible usar a ruta <https://docs.oracle.com/en/java/javase/17/docs/api/>

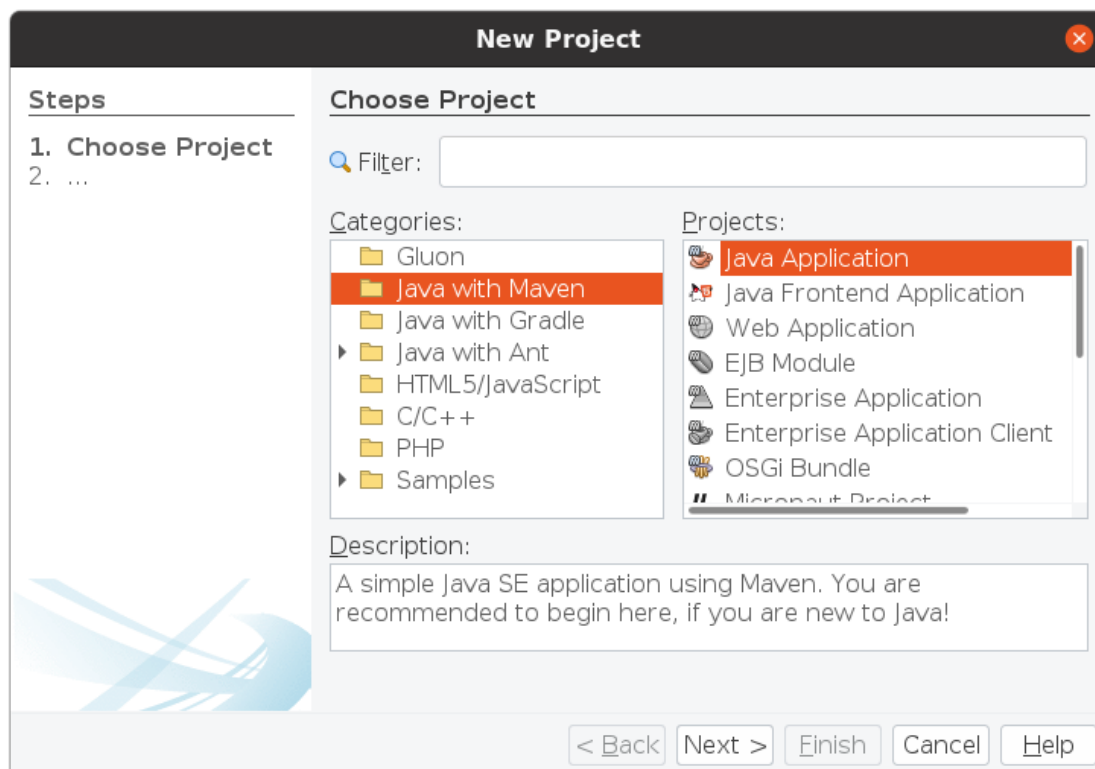


Introdución a NetBeans

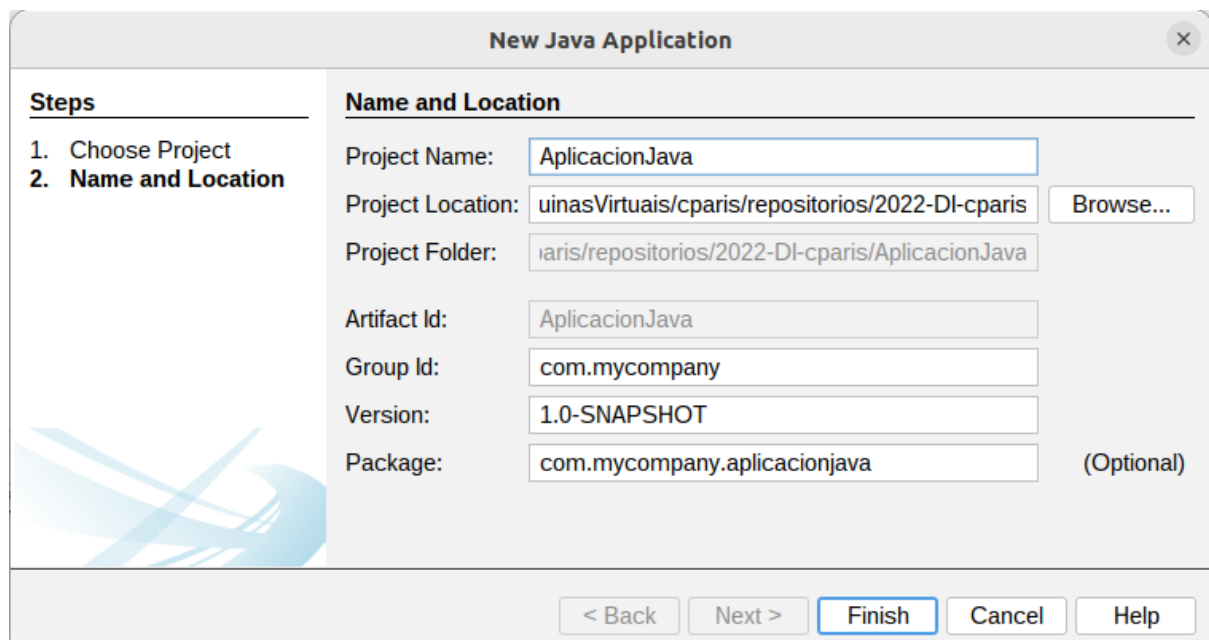
Hai documentación interesante de NetBeans en [Java SE Learning Trail](#).

Para crear un novo proxecto Java utilizando NetBeans, hai que seleccionar **Ficheiro -> Novo proxecto** (poden escollerse diferentes ferramentas de automatización da compilación: Maven, Gradle e Ant).

Neste exemplo, vaise crear unha aplicación Java utilizando Maven:



Cubrir o nome do proxecto e a carpeta onde se vai gardar. O resto dos parámetros imos deixalos cos valores por defecto. Recomendase **non utilizar caracteres especiais** (til, espazos, ñ, ...) nos nomes dos proxectos.

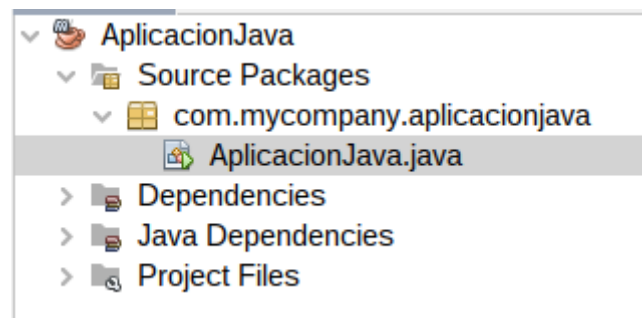


NOTA: a ruta por defecto dos proxectos de NetBeans está indicada pola chave **projectsFolder** do ficheiro **projectui.properties**. En Linux está almacenado na carpeta persoal do usuario nunha das seguintes rutas:

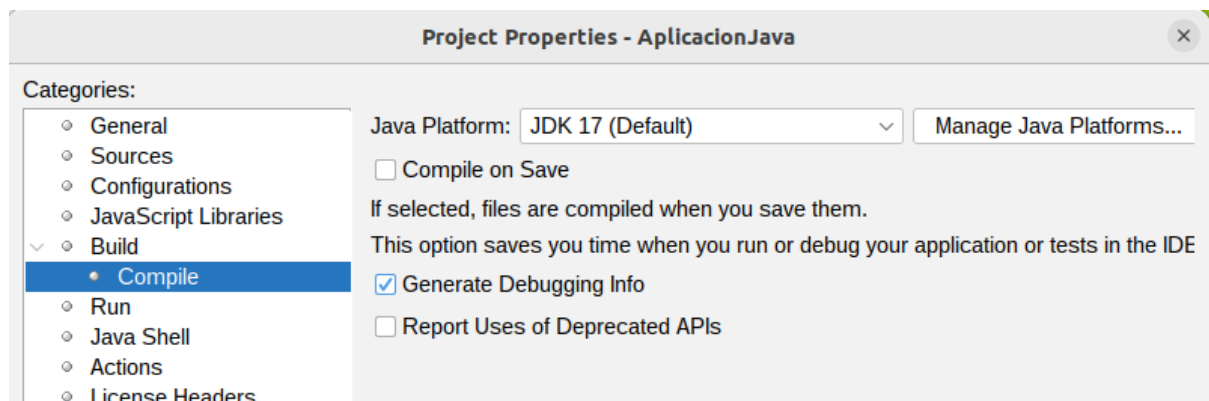
~/snap/netbeans/84/config/Preferences/org/netbeans/modules/projectui.properties

Unha vez creado o novo proxecto, este aparece na ventá do xestor de proxectos.

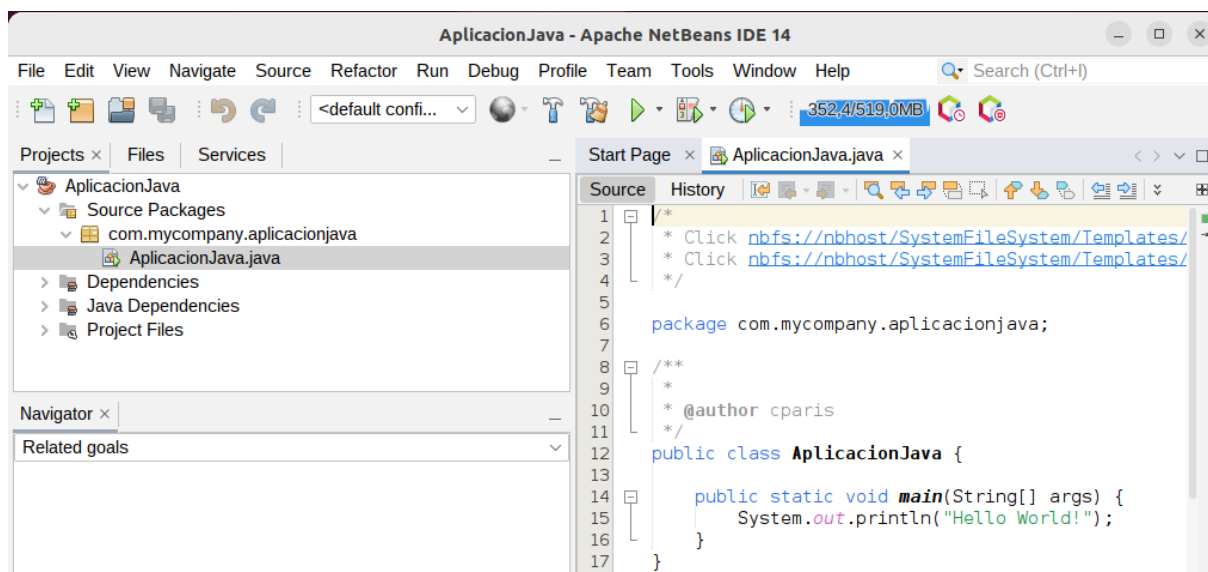
Pode navegarse polo contido da aplicación no xestor de proxectos e observar que, de momento, só ten unha clase.



Antes de continuar comprobar a versión de Java coa que se vai compilar o proxecto. Para iso acceder ás propiedades do proxecto (botón dereito sobre o nome do proxecto -> Properties) e seleccionar a opción “Compile”. Comprobar que a versión de Java é a 11.



Unha vez creado o proxecto, o IDE terá un aspecto similar ao da seguinte imaxe:



Na imaxe anterior obsérvase que a contorna de desenvolvemento está composta de varias ventás:

- **Xestor de proxectos:** contén os proxectos abertos con todas as súas clases, librerías, etc.
- **Xestor de arquivos:** contén a mesma información que o xestor de proxectos, pero utilizando unha vista que se parece máis ao xestor de arquivos do sistema operativo.
- **Servizos:** mostra os servizos instalados no sistema. Utilizarase, por exemplo, cando haxa que conectar cunha base de datos.
- **Navegador:** conterá os elementos gráficos dos formularios. Neste caso non se creou un formulario gráfico polo que non aparecen neste elemento ningún compoñente.
- **Parte dereita:** utilizarase para editar e deseñar a aplicación.

No caso de deseñar interfaces gráficas visualmente, aparecerán dous paneis a maiores á dereita da ventá:

- **Paleta:** contén os compoñentes gráficos que se poden engadir á aplicación. Para engadilos simplemente haberá que arrastralos ao formulario.
- **Propiedades do compoñente:** permite ver e modificar a lista de propiedades do compoñente seleccionado.

Se por algún motivo se cerra algunha das ventás anteriores, pode restaurarse pulsando no menú: **Ventá -> restablecer ventás**.

Atallos de NetBeans

O propósito de calquera entorno de desenvolvemento integrado (IDE) é maximizar a produtividade e apoiar o desenvolvemento de aplicacións sen problemas. NetBeans proporciona algúns trucos ou atallos que axudan á persoa desenvolvedora a codificar de forma máis rápida.

[Nesta ligazón](#) encóntanse funcións útiles de asistencia ao código e opcións de personalización de NetBeans. Algúns dos trucos máis útiles son:

- Formateo de código: **Menú Source -> Format** ou **Alt + Maiúsc + F**.

As opcións de formateo poden personalizarse no menú **Tools -> Options -> Editor -> Formatting**

- Suxestións: **Ctrl+espacio**. Permite suxerir código para codificar máis rápido.
- Xeración automática de código: **Alt+Insert**. Permite insertar código automaticamente: construtores, getters, setters, equals (compara dous obxectos), toString (permite escoller que campos queremos mostrar ao converter un obxecto a string),...
- Incluír un import: **Ctrl+Shift+I**

Tamén se pode acceder ao **menú Source -> Fix imports (Ctrl+Shift+I)** que permitirá eliminar os imports que non se usan e engadir os que faltan

Pode configurarse NetBeans para que se eliminen as sentencias “import” non necesarias ao gardar un arquivo: **Tools -> Options -> Editor -> On Save** seleccionar **Language: Java** e activar a opción **Remove Unused Imports**.

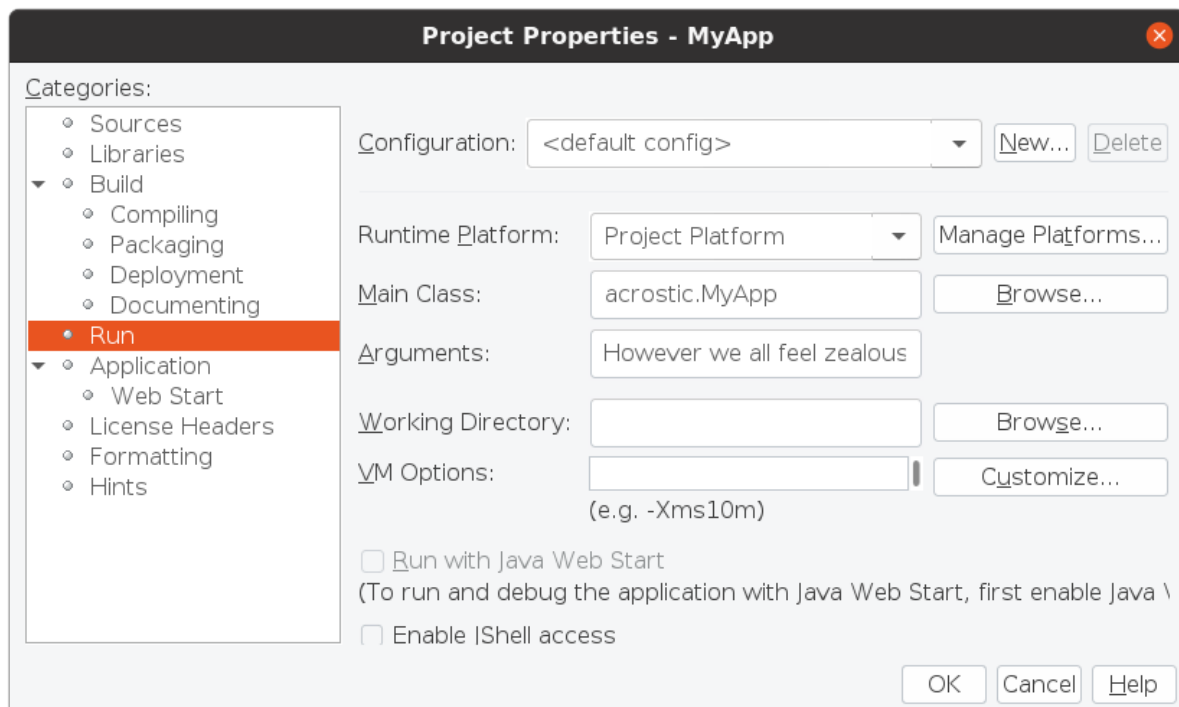
- Mostrar o Javadoc: **Ctrl + Shift. + espacio**
- Mostrar suxestión nun erro: **Alt+Enter**
- Borrado dunha liña: **Ctrl+X** ou tamén **Shift + Supr**
- Refactorizar o código: cambiar algo e que se cambie en todos os sitios onde se usa. **Ctrl+R**
- Navegación dentro do código: **Ctrl+”pulsar sobre un método/clase”** permitirá mostrar onde está definido.
- ¿Quen chamou a un método?: facer clic co botón dereito sobre o método -> Find usages (**Alt+F7**). Busca en todo o proxecto e mostra un listado de todas as chamadas ao método. Tamén se pode utilizar con variables e mostrará en todo o proxecto onde se usou esa variable.
- Buscar un texto en todos os proxectos abertos: **Ctrl+Shift+F**.
- Recuperar código borrado: o mellor sería usar unha ferramenta de control de versións como Git, pero NetBeans tamén proporciona un histórico dos ficheiros. Para acceder ao histórico dun ficheiro hai que pulsar sobre el co botón dereito do rato -> **History -> Show history**. Abrirá unha nova ventá coas versións do ficheiro e poderanse ver os cambios realizados.

Tamén pode resultar interesante configurar o IDE para que ao gardar un ficheiro se formatee automaticamente. Para iso acceder a **Tools -> Options -> Editor -> On Save** seleccionar **Language: Java** e activar a opción **Reformat: All Lines**.

É interesante mirar as opcións dos menús Source e Refactor e investigar as posibilidades que nos ofrecen para poder usar as máis útiles.

Executar unha aplicación Java

Para poder compilar e executar a aplicación é necesario establecer cal é a clase principal e os parámetros de execución. Esta configuración pode facerse pulsando co botón dereito sobre o nome do proxecto -> Propiedades -> Run. Na nova ventá que se abre establecerase a clase principal e os argumentos:



Pode encontrarse a información completa na documentación de NetBeans sobre como [desenvolver unha aplicación de propósito xeral Java](#).

NOTA: nun proxecto Maven, a clase principal configúrase no ficheiro **pom.xml**.

Depurar o código

Depurar o código é executar o código paso a paso e paralo onde creamos que pode haber algún erro para poder depuralo e corrixilo.

Como exemplo, vaise depurar o seguinte código:

```
public static void main(String[] args) {
    int n = 5;
    int factorial = 1;

    // n! = 1*2*3...*n
    for (int i = 1; i <= n; i++) {
        factorial *= i;
    }
    System.out.println("The Factorial of " + n + " is " + factorial);
}
```

Para facer a depuración deben establecerse puntos de ruptura (breakpoints). Hai que colocarse na liña onde se quere inserir o breakpoint e pulsar **Ctrl+F8**. Observar que na seguinte imaxe se inseriu un breakpoint na liña 12, simbolizado cun triángulo invertido á esquerda:

```

10 public class AplicacionJava {
11
12     public static void main(String[] args) {
13         int n = 5;
14         int factorial = 1;
15
16         // n! = 1*2*3...*n
17         for (int i = 1; i <= n; i++) {
18             factorial *= i;
19         }
20         System.out.println("The Factorial of " + n + " is " + factorial);
21     }
22 }
23

```

Unha vez establecidos os breakpoints, en vez de executar o código hai que depuralo (**Ctrl+F5**). Isto, o que fai, é executar o código ata que chega á liña do punto de ruptura, onde se parará. A liña na que se para estará de cor verde.

```

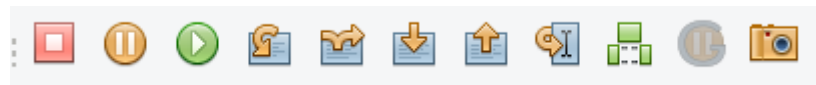
10 public class AplicacionJava {
11
12     public static void main(String[] args) {
13         int n = 5;
14         int factorial = 1;
15
16         // n! = 1*2*3...*n
17         for (int i = 1; i <= n; i++) {
18             factorial *= i;
19         }
20         System.out.println("The Factorial of " + n + " is " + factorial);
21     }
22 }
23

```

Observar que se abriron dúas novas pestanas: variables e breakpoints. A pestana de variables utilizarase para observar os valores das variables durante a depuración do programa:

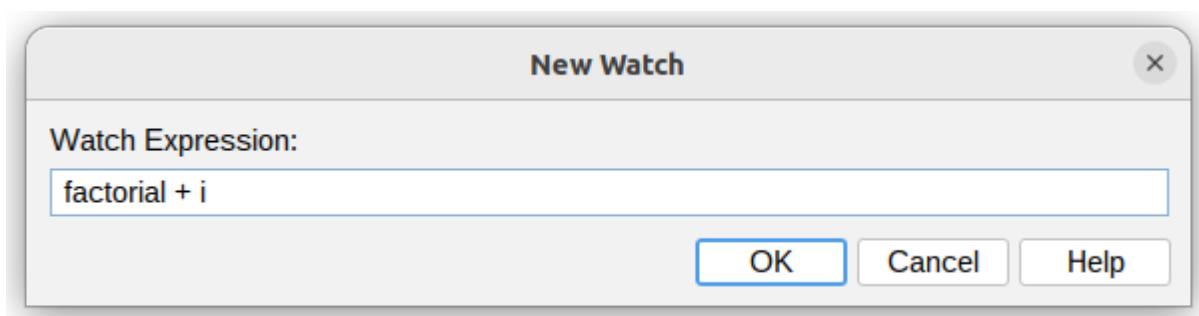
Notifications	Variables ×	Breakpoints	Output	
	Name	Type	Value	
	<Enter new watch>			
	> Static			
	> args	String[]	#59(length=0)	

Durante a depuración poden seleccionarse os seguintes botóns da barra de ferramentas para: avanzar unha liña (sen entrar nos métodos), entrar nun método para executar liña a liña, etc...:

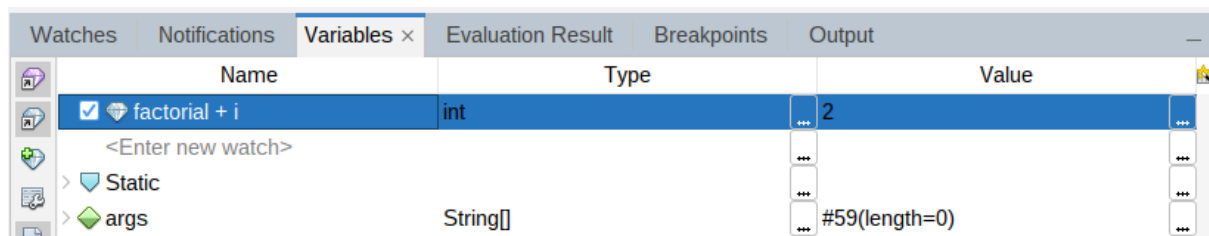


Tamén poden engadirse novos breakpoints durante a depuración do código.

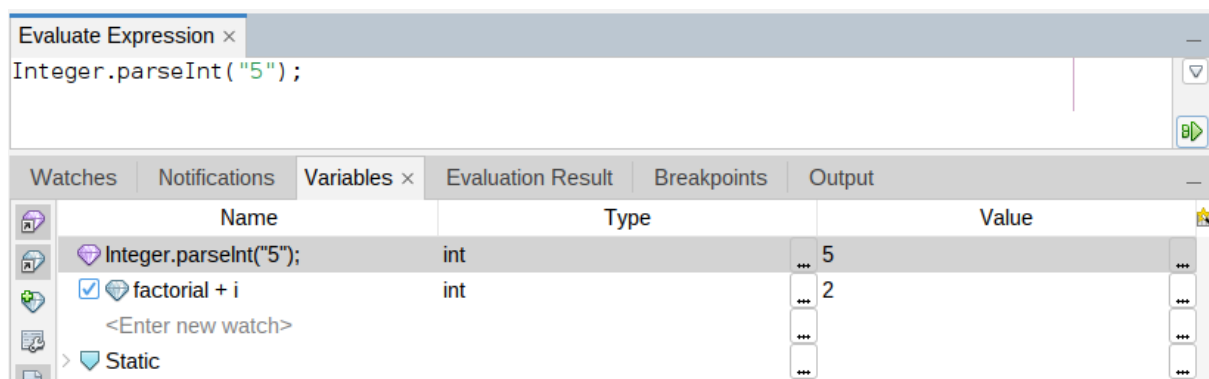
Podemos configurar un “novo observado” ou “new watch”. Isto utilízase cando queremos saber o valor dunha variable en todo momento. Para crear un “novo observado” hai que pulsar no menú **Debug -> New Watch** e escribimos a variable ou expresión que queremos observar.



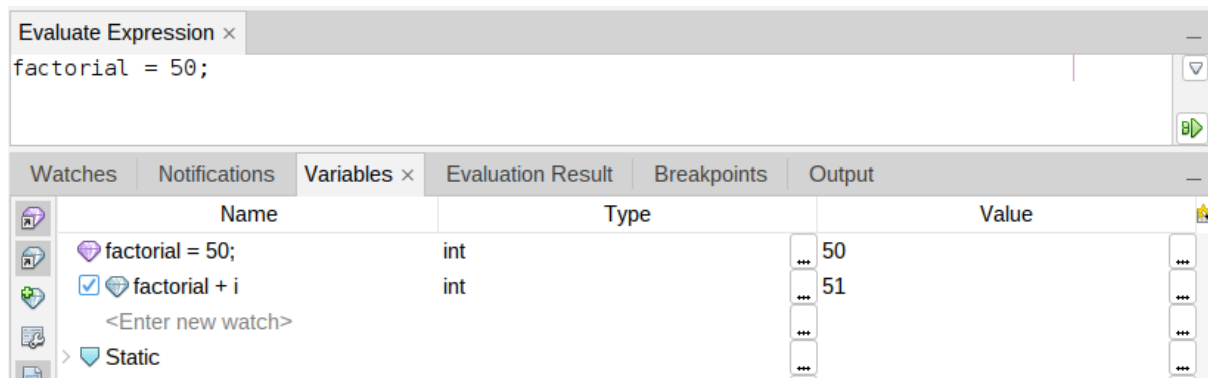
Ao crear un novo observado, aparecerá a expresión a observar e o seu valor na ventá coas variables:




Outra operación moi útil é a avaliación de expresións, á que se accede dende o menú **Debug -> Evaluate Expression**. Pode escribirse calquera expresión e esta será avaliada ao pulsar Ctrl+Enter.




Incluso podería cabiarse o valor de calquera variable para facer algunha proba:



Podemos seguir depurando a aplicación paso a paso ou pódese pulsar o botón  para que continúe ata o seguinte breakpoint.

Distribución de aplicacións

Unha vez se ten a aplicación rematada e funciona correctamente, pode prepararse para ser executada fóra do entorno de desenvolvemento.

O principal comando para construír a aplicación é **Clean and Build** . Este comando borra todas as clases previamente compiladas e reconstrúe todo o proxecto dende cero creando as carpetas **build** e **dist** na carpeta do proxecto nun proxecto **Ant** ou o directorio **target** nun proxecto Maven:

- Os ficheiros .class colócanse na carpeta **classes**.
- O ficheiro JAR coa distribución do proxecto colócase no directorio **dist** ou no directorio **target** dependendo da ferramenta de compilación.
- Cando o proxecto precisa librerías adicionais ao JDK, créase unha carpeta **lib** dentro da carpeta **dist**.

Para executar a aplicación fóra do ide utilizarase a terminal e executarase o seguinte comando colocándose no directorio do .jar:

java -jar MyApp.jar [parámetros]

NOTA: se dá un erro de “no main manifest attribute, in nomeFicheiro.jar” solucionarase engadindo a etiqueta <build> ao ficheiro pom.xml e configurando o nome da clase principal:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany</groupId>
  <artifactId>AplicacionJava</artifactId>
  <version>1.0-SNAPSHOT</version>
```

```

<packaging>jar</packaging>
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>17</maven.compiler.source>
  <maven.compiler.target>17</maven.compiler.target>
  <exec.mainClass>com.mycompany.aplicacionjava.AplicacionJava</exec.mainClass>
</properties>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>2.4</version>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <mainClass>com.mycompany.aplicacionjava.AplicacionJava</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>

```

Referencias

Para la elaboración de este material utilizáronse, entre otros, los recursos que se enumeran a continuación:

- [NetBeans Java SE Learning Trail](#)
- [Code Assistance in the NetBeans IDE Java Editor: A Reference Guide](#)
- [Developing General Java Applications](#)