

UD02: Installing and using Development Environments



1. History

2. Functions and structure

- 2. 1. Graphical user interfaces
- 2. 2. Text editor
- 2. 3. Compiler
- 2. 4. Interpreter
- 2. 5. Debugger
- 2. 6. Database access and file management
- 2. 7. Version control
- 2. 8. Refactoring
- 2. 9. Documentation and help
- 2. 10. Development environment examples

3. Types of IDE's and how to choose

- 3. 1. According to their license
- 3. 2. Depending on the platform
- 3. 3. Programming language and framework
- 3. 4. Tools and availability

4. Java

- 4. 1. Download and install the JDK
- 4. 2. Configure environment variables "JAVA_HOME" and "PATH"

5. Eclipse

- 5. 1. Installation
- 5. 2. Setting
- 5. 3. Modules
- 5. 4. Basic usage ("Hello world!")
- 5. 5. Update and maintenance

6. Netbeans

- 6. 1. Installation
 - 6. 1. 1. Install from binaries
 - 6. 1. 2. Install from script
 - 6. 1. 3. Install via snap
 - 6. 1. 4. First run
- 6. 2. Setting
- 6. 3. Modules
- 6. 4. Basic usage ("Hello world!")
- 6. 5. Update and maintenance

7. Sources of Information

1. History

Previously, at unit 1, we talked about the phases in the software development process.

One of them was the coding phase, in which a programming language was used to pass all the actions that the application had to carry out into a language that the machine was capable of understanding and executing.

Reference was also made to tools to support the programming process.

In this unit we are going to analyze, install and run these tools to understand their action and effect.

Many people learn to program using a simple text editor, compiler, and debugger. But most eventually end up making use of some integrated development environment (IDE) to build applications.

The first integrated development environments were born in the early 70s, and became popular in the 90s. They aim to gain reliability and time in software projects.

They provide the programmer with a series of components with the same graphical interface, with the consequent convenience, increased efficiency and reduced coding time.

Typically, an IDE is dedicated to a certain programming language. However, the latest versions of IDEs tend to be compatible with several languages (eg Eclipse, NetBeans, Microsoft Visual Studio ...) by installing additional plugins.

In the decades of using the punch card as a storage system the concept of Integrated Development Environment simply did not make sense.

The programs were written with flow charts and entered the system through the punch cards. Later, they were compiled.

The first programming language to use an IDE was BASIC (which was the first to abandon punched cards or paper tapes as well).

This first IDE was based exclusively on the command console (normal on the other hand, if we take into account that operating systems with a graphical interface did not enter the market until the 1990s). However, its use of file management, compilation, debugging ... is perfectly compatible with current IDEs.

At a popular level, the first IDE can be considered to be the IDE called Maestro. It was born in the early 70s and was installed by about 22,000 programmers around the world. He led the field during the '70s and '80s.

The use of integrated development environments was ratified and strengthened in the 90s and today we have countless IDEs, both free and proprietary licenses.

2. Functions and structure

An integrated development environment, or IDE, is a program made up of a series of tools that programmers use to develop code.

This program can be intended for use with a single programming language, or it can accommodate several.

The tools that typically make up an integrated development environment are a graphical user interface (GUI) building aid system, a text editor, a compiler/interpreter, and a debugger.

The functions of the IDEs are:

- Code editor: syntax coloring.
- Auto-completion of code, attributes and methods of classes.
- Automatic code identification.
- Visual conception tools to create and manipulate visual components.
- Code generation and management wizards and utilities.
- Source files in some folders and compiled to others.
- Compilation of complex projects in one step.
- Version control: have a single file store shared by all collaborators of a project. In the event of an error, self-recovery mechanism to a previous stable state.
- Supports changes of several users simultaneously.
- Integrated documentation generator.
- Detection of syntax errors in real time.

Other important functions are:

- Offers code refactoring: minor changes to the code that make it easier to read without altering its functionality (for example, changing the name of a variable).
- Automatically enter tabs and spacing to increase readability.
- Debugging: tracking variables, breakpoints, and interpreter error messages.
- Increase of functionalities through the management of its modules and plugins.
- Administration of user interfaces (menus and toolbars).
- Administration of user configurations.

2.1. Graphical user interfaces

The graphical user interfaces are a set of functionalities that allow to incorporate, edit and eliminate graphical components in a simple way in the application that is being developed. These components will facilitate user interaction with the computer.

2.2. Text editor

A text editor is a program that allows you to create and modify digital files composed solely of plain text, commonly known as plain text or text files.

The text editor is the most used tool because it offers the possibility of creating and modifying the developed contents, the programming code that will make the application work properly.

As long as the IDE recognizes the programming language (or has the appropriate component) the editor will offer:

- Syntax highlighting: the keywords will be recognized with colors, which will greatly facilitate the programmer's work.
- Code completion: the code that is being written will be recognized and, for example in an object or class, it will offer its attributes, properties or methods so that the programmer can select which one they want to reference.

- Bug fixer, usually from a syntax point of view.
- Foldable regions: hiding certain parts of the code, in order to facilitate the visualization of the most relevant fragments.

2.3. Compiler

Depending on the programming language used, the IDE may offer the functionality of compiling it. A compiler translates programming code into a machine language capable of being interpreted by processors and of being executed. When compiling a programming code, integrated development environments will have different phases of code analysis, such as the semantic phase and the lexicographic phase. The compilation will show the errors found or generate executable code, if it finds none.

2.4. Interpreter

The interpreter translates high-level code to byte code, but unlike the compiler, it does so at run time.

2.5. Debugger

The debugger is a program that allows you to test and debug the source code of a program, making it easier to detect errors.

Some of the typical features of debuggers are:

- Allow line by line execution of the code, validating the values that the variables acquire.
- Pause the program in a certain line of code, making use of one or more breakpoints.
- Some debuggers offer the possibility of being able to modify the content of a variable while it is being executed.

2.6. Database access and file management

Fourth generation languages offer the possibility of integrating database access code (or structured statement code). To facilitate this functionality, it is highly recommended to have the possibility of directly accessing the database from the same development environment and not having to use another. The same will happen with file management.

2.7. Version control

As a programmer develops new lines of code, it is important to have a history of the work carried out or of the versions that have been accepted or modified. Version control allows you to go back to a previous version that is stable or that meets certain characteristics that the changes made do not meet.

2.8. Refactoring

Refactoring is a software engineering technique aimed at restructuring a source code, altering its internal structure without changing its external behavior.

2.9. Documentation and help

The documentation and help provides access to documentation, manuals and contextual help on the instructions and syntax of the supported languages.

2.10. Development environment examples

Today's development environments provide a framework for most of the programming languages on the market (for example C, C ++, C #, Java or Visual Basic, among others). In addition, it is possible that the same development environment allows the use of several programming languages, such as Eclipse (to which support for additional languages can be added through -plugins- connectors) or Visual Studio (which is designed to work with the languages VB.Net, C #, C ++ ...).

Examples of cross-platform IDE can be found, among many others:

- Eclipse, open source multiplatform project (Windows, GNU/Linux, Mac), founded by IBM in November 2001, developed in Java.
- Netbeans, open source multiplatform project (Windows, GNU/Linux, Mac, Solaris), founded by Sun Microsystems in June 2000, developed in Java.
- Anjuta DevStudio, for GNU/Linux, created by Naba Kumar in 1999.
- JBuilder, a multiplatform tool (Windows, GNU/Linux, Mac), owned by the Borland company, released in 1995. The 2008 version incorporates three editions (Enterprise - paid-, Professional -paid- and Turbo -free-).
- JDeveloper, a free multiplatform tool (Windows, GNU/Linux, Mac), owned by the Oracle company, appeared in 1998, initially based on JBuilder but developed since 2001 in Java.
- BlueJ is an integrated development environment for the Java programming language, developed primarily for educational purposes, but is also suitable for small-scale software development.

BlueJ was developed to support the teaching and learning of object-oriented programming, and as a result, its design differs from other development environments. The main screen graphically shows the class structure of an application under development (in a very UML-like diagram), and objects can be created and tested interactively. Combined with a simple user interface, this ease of interaction allows for easy experimentation with objects in development. The concepts of object orientation (classes, objects, communication through method calls) are represented visually in the interaction design in the interface.

Apart from the ones we have mentioned, there are many other well-known IDEs for certain platforms, such as:

- Visual Studio
- Dev-Pascal
- Dev-C++
- MonoDevelop

3. Types of IDE's and how to choose

In this link you can consult the different IDE's available according to the programming language, their type of license, in which language they have been programmed, compatible platforms, etc.: https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments

In order to correctly choose the IDE to work with, we need to know the characteristics we are looking for from it. We will see various criteria that are usually the most used when choosing the IDE, they are: license, platform, programming language and tools-availability.

3.1. According to their license

We can opt for **Free** IDE's, in which their license offers us the freedoms of free software. Or **Privative**, in which we do not have access to all the freedoms of the software. It must be remembered that this is independent of the economic cost, since we can find free proprietary IDE's.

3.2. Depending on the platform

It is one of the most restrictive criteria for selecting the IDE and, more importantly, it is for which Operating System we are going to develop our software.

Today there are emulators or virtualizers of Operating Systems, but it is not advisable to use these methods if our software is not going to be executed through a virtual machine. We will have to develop applications in the same OS in which they will work when they are in the exploitation phase. This restriction is inherent to the IDE compiler since it is the one in charge of translating our source code into object code, which is the one that will be executed on the system.

3.3. Programming language and framework

An IDE can support one or more programming languages, so it is important to know in which language we are going to code our project and what languages the different IDEs available offer us.

This criterion goes hand in hand with the operating system that we use, since if we wanted to develop in Visual Basic under a GNU/Linux operating system, Visual Studio would not be the IDE option but Gambas.

The same happens with work platforms or frameworks, it not only depends on the framework that we are going to use, we also need to know under what operating system we are going to run it. For example, if we were to develop with Visual Basic on the .NET platform under GNU/Linux, we would have to use Mono Develop instead of Visual Studio.

3.4. Tools and availability

The tools are the criterion that completes the choice of the IDE because if we find several IDEs that meet the language and operating system criteria, we will opt for the one with the best tools.

Sometimes IDEs can be restrictive because of the programmer's preferences, because they work collaboratively or because of the way different codes are used between different IDEs. For example, if our co-workers use the Team Server Foundation (TSF) as a version control system, we must use it too, and the only way to do it is with Visual Studio; but if the colleagues are working in Java with a Subversion version control system (SVN), we could interchangeably use Netbeans, Eclipse or IntelliJ IDEA, among others, since SVN is available for all those IDEs.

Among our needs and preferences could be to have a functionality to create help files and documentation. In that case we should look for an IDE with that functionality or look for an extension (plugin) for that IDE that supports the desired functionality.

A significant amount of time must be invested to thoroughly research and document the potential IDEs that we can choose so as not to choose incorrectly or not know what functionalities the IDE we have chosen has.

After checking the above criteria, the most restrictive aspect of availability is the price of the application; depending on our budget we can access different IDEs.

If we do not have a budget, there are certain free solutions to have an IDE to work with that will meet most of the criteria we need.

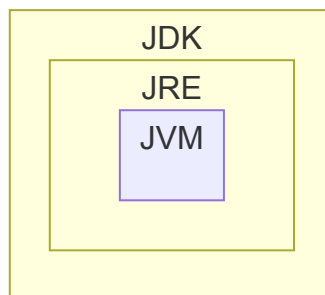
4. Java

Each software and each development environment has specific characteristics and functionalities. This will also be reflected in the installation and configuration of the software. Depending on the platform, environment or operating system in which the software is to be installed, one installation package or another will be used, and some options or others must be taken into account in its configuration.

The following shows how to install an integrated software development tool, such as Eclipse. But you will also be able to observe the procedures to install other necessary or recommended tools to work with the JAVA programming language, such as Tomcat or the Java Virtual Machine.

You must bear in mind the following concepts:

- JVM (Java Virtual Machine, Java virtual machine) is in charge of interpreting the byte code and generating the machine code of the computer (or device) on which the application is executed. This means that we need a different JVM for each environment.
- JRE (Java Runtime Environment) is a set of Java utilities that includes the JVM, libraries and the set of software necessary to run Java client applications, as well as the connector for Internet browsers to run applets.
- JDK (Java Development Kit) is the set of tools for developers; It contains, among other things, the JRE and the set of tools necessary to compile the code, package it, generate documentation ...



The installation process consists of the following steps:

1. Download, install and configure the JDK.
2. Download and install a web or application server.
3. Download, install and configure the IDE (Netbeans or Eclipse).
4. Configure JDK with IDE.
5. Configure the web or application server with the IDE installed.
6. If necessary, installation of connectors.
7. If necessary, install new software.

4.1. Download and install the JDK

We can differentiate between:

- Java SE (Java Standard Edition): this is the standard version of the platform, this platform being the basis for all Java development environments in terms of client, desktop or web applications.
- Java EE (Java Enterprise Edition): This is the largest version of Java and is generally used for creating large client/server applications and for WebServices development.

In this course, the functionalities of Java SE will be used.

The file is different depending on the operating system where it has to be installed. So:

- For Windows and Mac OS operating systems there is an installable file.
- For GNU/Linux operating systems that support .rpm or .deb packages, packages of this type are also available.
- For the rest of the GNU/Linux operating systems there is a compressed file (ending in .tar.gz).

In the first two cases, you just have to follow the usual installation procedure for the operating system you are working with.

In the latter case, however, you have to unzip the file and copy it to the folder where you want to install. Normally, all users will have read and execute permission to this folder.

Starting with JDK version 11, Oracle distributes the software with a significantly more restrictive license than previous versions. In particular, it can only be used to "develop, test, prototype and demonstrate your applications." Any use "for internal commercial, production or business purposes" other than that mentioned above is explicitly excluded. If you need it for any of these uses not allowed in the new license, in addition to the previous versions of the JDK, there are reference versions of these versions licensed "GNU General Public License version 2, with the Classpath Exception", which allows most of the usual uses. These versions are linked to the same download page and also to the address jdk.java.net.

An alternative is to use <https://adoptium.net/> formerly known as adoptOpenJDK, which has now been integrated into the Eclipse foundation. From there we can download the binaries of the openJDK version for our platform without restrictions. [Full story] (<https://es.wikipedia.org/wiki/OpenJDK>).

In GNU/Linux we can use the commands:

- `sudo apt install default-jdk` to install the default jdk.
- `java --version` to see the versions available on our system.
- `sudo update-alternatives --config java` to choose which of the installed versions we want to use by default or even see the path of the different versions that we have installed.

4.2. Configure environment variables "JAVA_HOME" and "PATH"

Once the JDK is downloaded and installed, you need to set some environment variables:

- The `JAVA_HOME` variable: indicates the folder where the JDK has been installed. It is not mandatory to define it, but it is very convenient to do so, since many programs look for the location of the JDK in it. Also, it makes it easy to define the following two variables.
- The `PATH` variable. It should point to the directory that contains the executable of the virtual machine. It is usually the `bin` subfolder of the directory where we have installed the JDK.

CLASSPATH variable

Another variable that the JDK takes into account is the `CLASSPATH` variable. It points to the folders where the libraries of the application that you want to run are located with the java order. It is preferable, however, to indicate the location of these folders with the `-cp` option of the same java command, since each application can have different libraries and environment variables affect the entire system.

Setting the `PATH` variable is essential for the operating system to find the JDK commands and be able to execute them.

5. Eclipse

Eclipse is an open source application currently developed by the Eclipse Foundation, an independent, non-profit organization, which fosters an open source community and the use of a set of products, services, capabilities and plug-ins for the disclosure of the use open source in the development of computer applications. Eclipse was originally developed by IBM as the successor to VisualAge.

As Eclipse is developed in Java, it is necessary, for its execution, to have a JRE (Java Runtime Environment) previously installed on the system. To find out if you have this JRE installed, you can do the test on the official Java website, in the Do I have Java? Section.

If we will develop with Java, as is our case, we must have the JDK installed (remember that it is a superset of the JRE).

5.1. Installation

Current versions of the Eclipse environment are installed with an installer. This, basically, is in charge of decompressing, solving some dependencies and creating the shortcuts.

This installer can be obtained by downloading it directly from the official website of the Eclipse Project www.eclipse.org. You can find the versions for the different operating systems and instructions for using it. They are not complex at all.

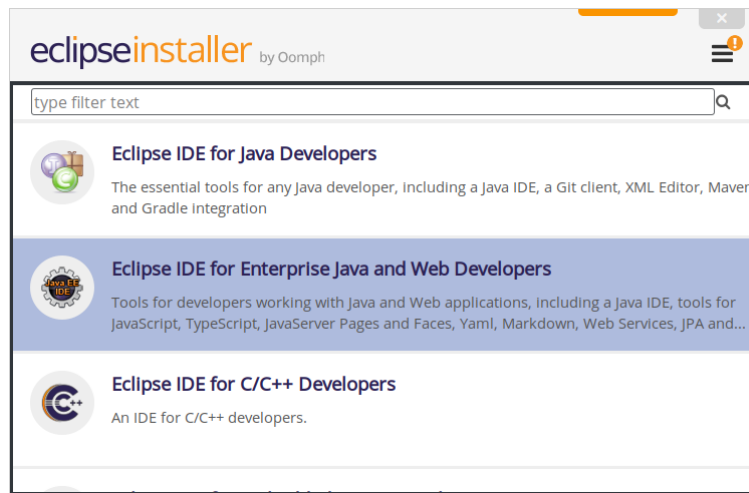
In the case of GNU/Linux and MAC OS, the file is a compressed file. So you have to unzip it and then run the installer. This is the *eclipse-inst* file, inside the eclipse folder, which is a subfolder of the result of unzipping the previous file.

If only the current user will use the IDE, the installation can be carried out without using administrator or root privileges and selecting, for the installation, a folder belonging to this user. If you want to share the installation between different users, you should indicate to the installer a folder on which all these users have read and execute permission.

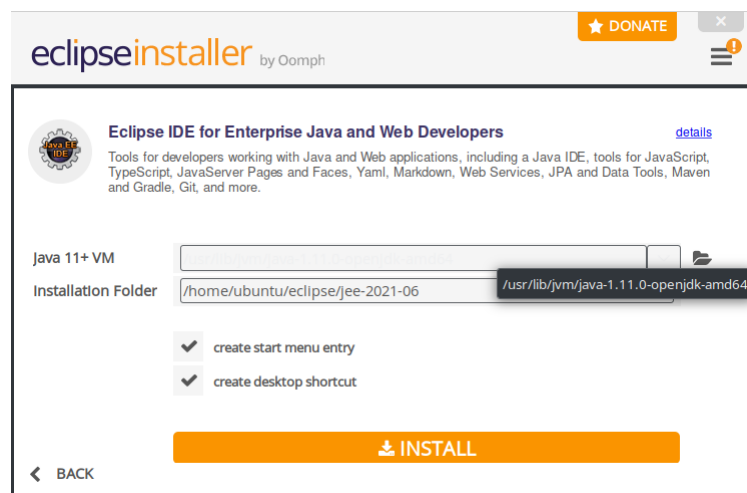
When we start the installer we will see a screen similar to this:



The installer will ask which version we want to install. The version we will use is "Eclipse IDE for Java EE Developers".

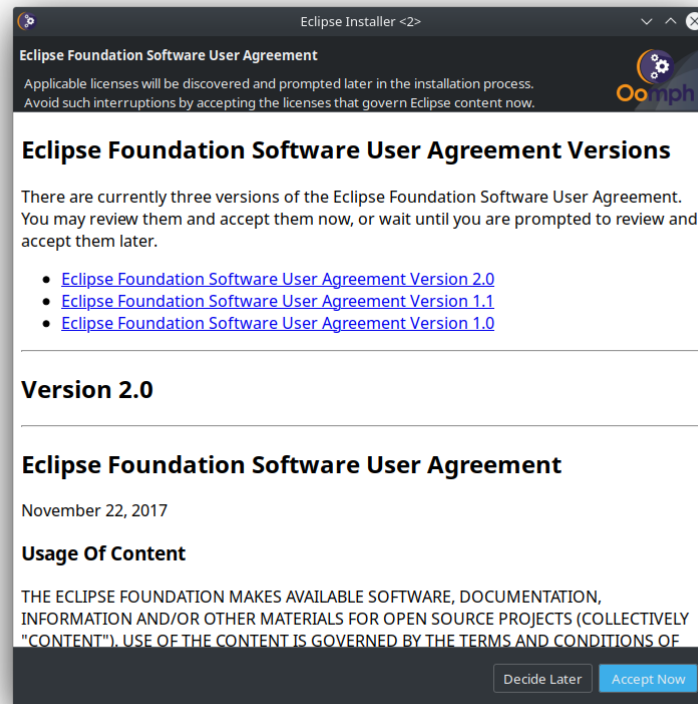


Then it will ask for the version of JDK/JRE that we are going to use (in the screenshot it appears with white letters). It also asks us for the folder where we will install. And two check boxes to indicate if we want it to create for us the shortcut to the application menu already on the desktop.

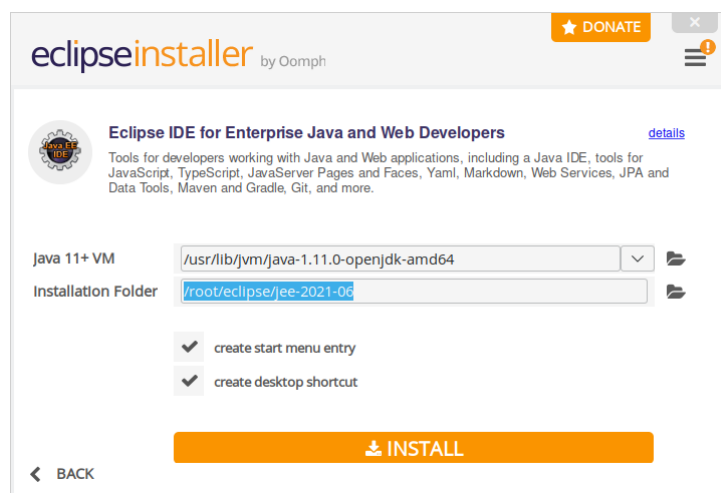


To select the correct folder, you must take into account which users will use the environment. All of them must have read and execute permission on the folder in question. Once the folder is entered, we can press the INSTALL button to start the installation.

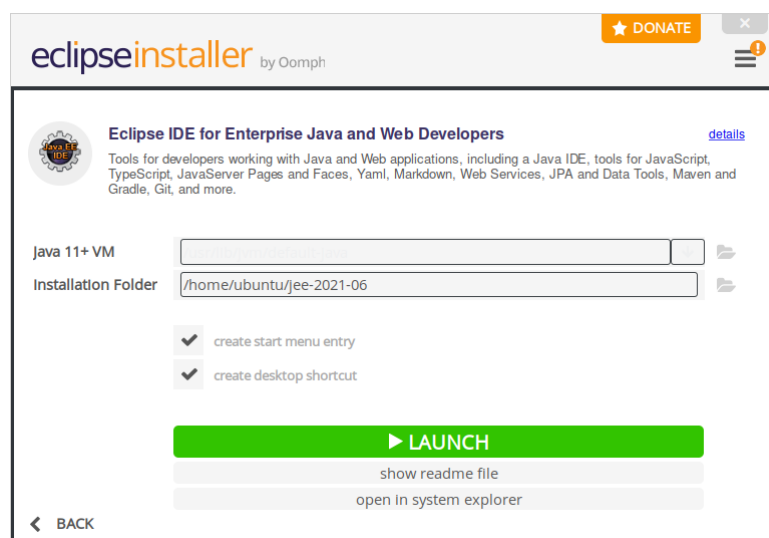
We will also be asked to accept the licenses for the software to be installed, as the screenshot shows:



During the installation we will see a progress screen like the one shown below:



Once the installation is finished, we are shown a screen that invites us to run the environment directly.



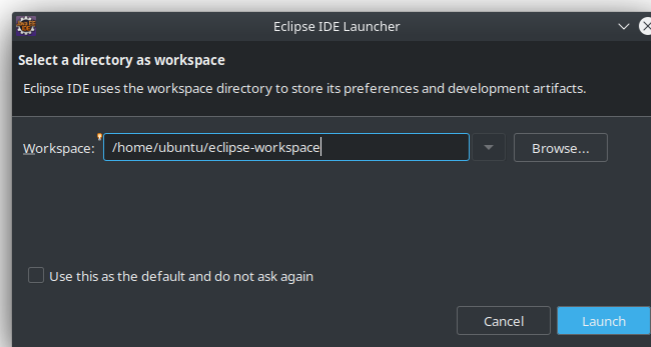
This first time we can run the Eclipse environment by clicking the LAUNCH button. The rest of the time, it will be necessary to invoke it from the shortcuts or launchers, if they have been created or, otherwise, directly invoking the executable. This is called eclipse and you will find it in a subfolder of the installation folder also called eclipse. The exact path may vary from version to version.

If in the future it is necessary to uninstall it, only the folder where it has been installed should be deleted since the Eclipse installation does not appear in the GNU/Linux repository or in the control panel in Windows.

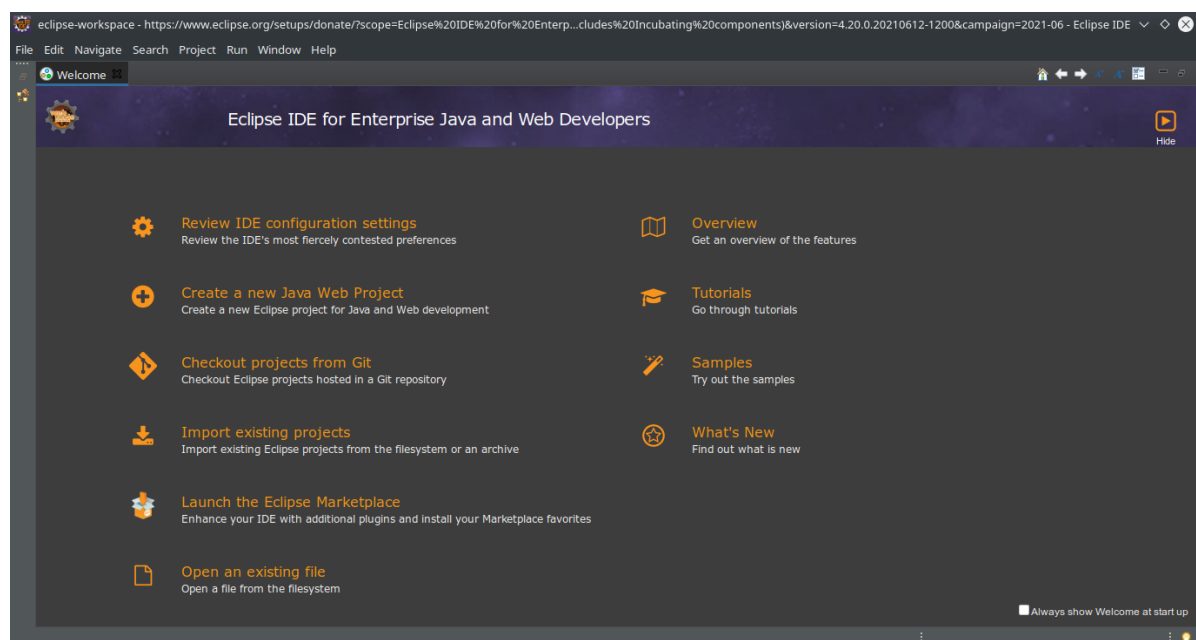
When we run the environment, a screen like the following will appear:



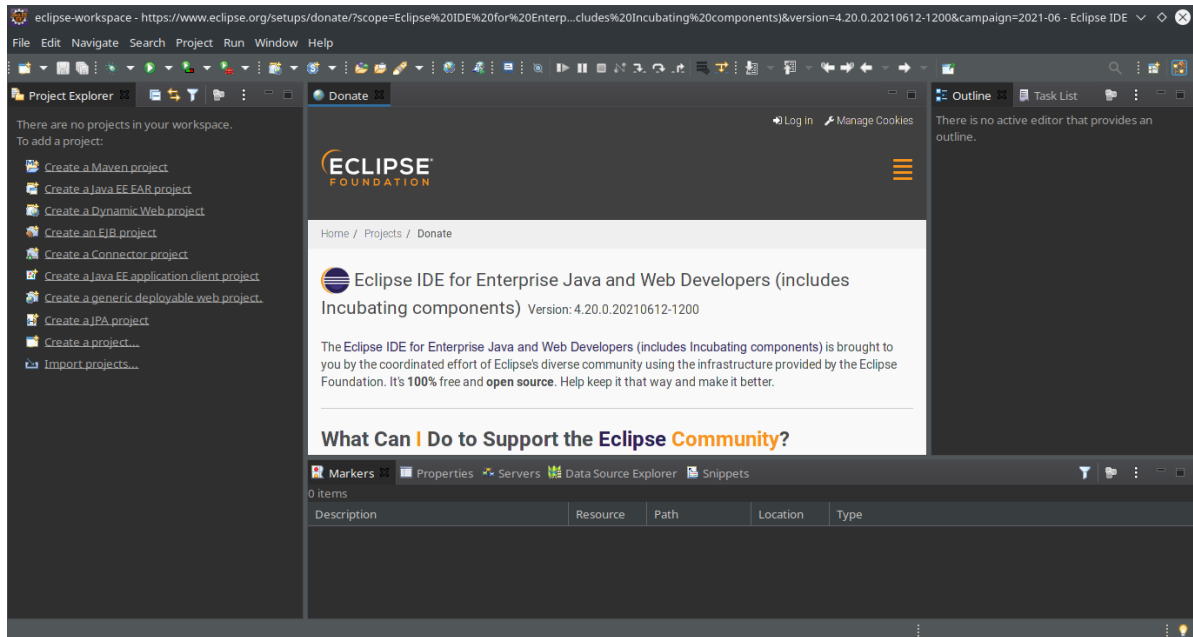
Immediately, we will be asked in which folder the workspace will be located. We can ask it to remember it for the rest of the executions by activating the option "* Use this as the default and do not ask again *".



The first time we run it, the welcome tab will be displayed. We can ask you not to show it to us any more by deactivating the option "Always show Welcome at start up".



Once this tab is closed, the work environment will be similar to this:



By default Eclipse offers us the download of the lightest installer that will download from the Internet the necessary packages to complete the installation according to our choices. If this installation gives us problems, we can download the "package" version in which we must previously choose the installation package we want, it will take up a lot more, but it will download all the necessary packages. Then we just have to unzip the downloaded file in a folder of our choice and we will have eclipse installed. We will have to create our own start menu and desktop icons (you can follow this [guide] (<https://www.donovanbrown.com/post/Adding-Eclipse-to-Launcher-on-Ubuntu-1604>) changing the path where you have unzipped your version of eclipse).

5.2. Setting

Java version

By default, Eclipse tries to use the new features of JDK 16, but in our case, for example, we have version 11. We can customize these options in the `Window/Preferences/Java/Compiler` section and choose the version for the `Compiler compliance level` field. correct, in our case 11.

Also, if we need it, we can configure the JDKs that are available, add or remove them from the `Window/Preferences/Java/Installed JREs` option.

Outlook

Eclipse calls the distribution of the panels in the Perspective window, there are a few predefined ones, and we can configure ours, to our liking in the `Window/Perspective` section.

Appearance

Eclipse allows us to customize any aspect of the appearance of our environment, change both the IDE theme and the font size and colors for the source code coloring. All these options are available in `Window/Appearance`.

5.3. Modules

Eclipse's options and functionality can be expanded by adding modules from its plugin "store". In `Help/Eclipse Marketplace ...` we can for example search by text, or search in the popular tab. That will show us all the add-ons that contain the searched word, or the most downloaded add-ons from the marketplace. We can install, for example, `SonarLint 6.0` that helps us to keep our code clean of common errors, for this we simply have to press the `INSTALL` button that appears next to it in the list, accept the use license and it will automatically ask us to restart the `IDE`.

5.4. Basic usage ("Hello world!")

Eclipse provides information about its use in the `Help` section, and we can learn to create our first project in Java (the typical "Hello World! "). To do this we must open the `Welcome` window, which is the one that appears when we open eclipse by first time, or we can open it from `Help/Welcome`, from this window we can choose the `Tutorials` section, and within the Java Development section, choose the first item "Create a Hello World application", and Eclipse itself will guide us step by step to create and run our first Java project in Eclipse.

5.5. Update and maintenance

In the same section `Help` Eclipse provides us with the options to update Eclipse itself or the add-ons that we have installed `Help/Check for Updates`.

We can customize the behavior regarding updates in the `Window/Preferences/Install/Update/Automatic Updates` section.

6. Netbeans

NetBeans is a very powerful Integrated Development Environment (IDE) tool used primarily for Java and C/C++ development. It allows you to easily develop web, desktop and mobile applications from its modular framework. You can add support for other programming languages like PHP, HTML, JavaScript, C, C++, Ajax, JSP, Ruby on Rails, etc. using extensions.

NetBeans IDE 12 has been released with support for Java JDK 11. It also comes with the following features:

- Support for PHP 7.0 to 7.3, PHPStan and Twig.
- Include modules in the "webcommon" cluster. That is, all JavaScript functions in Apache NetBeans GitHub are part of Apache NetBeans 10.
- The "groovy" cluster modules are included in Apache NetBeans 10.
- OpenJDK can automatically detect JTRG from the OpenJDK configuration and register the expanded JDK as a Java platform.
- Support for JUnit 5.3.1

6.1. Installation

We can install NetBeans in three ways:

6.1.1. Install from binaries

Step 1: Download the NetBeans file

Download the NetBeans 12 binary file `netbeans-12.4-bin.zip`.

Step 2: Extract the file

Wait for the download to finish and then extract it.

```
1 $ unzip netbeans-12.4-bin.zip
```

Confirm the content of the created directory file:

```
1 $ ls netbeans
2 apisupport    enterprise    groovy        javafx        netbeans.css  profiler
3 bin           ergonomics    harness       LICENSE       NOTICE       README.html
4 cpp1ite       etc           ide           licenses      php           webcommon
5 DEPENDENCIES extide        java          nb            platform      websvccommon
```

Step 3: Move the `netbeans` folder to `/opt`

Now let's move the `netbeans/` folder to `/opt`

```
1 $ sudo mv netbeans/ /opt/
```

Step 4: Configuration path

The Netbeans executable binary is located in `/opt/netbeans/bin/netbeans`. We need to add its parent directory to our `$PATH` so that we can start the program without specifying the absolute path to the binary file.

Open your `~/.bashrc` or `~/.zshrc` file.

```
1 | $ nano ~/.bashrc
```

Add the following line to the end of

```
1 | export PATH = "$PATH:/opt/netbeans/bin/"
```

Get the file to start Netbeans without restarting the shell.

```
1 | $ source ~/.bashrc
```

Step 5: Create the NetBeans IDE Desktop Launcher (optional)

Create a new file in `/usr/share/applications/netbeans.desktop`.

```
1 | $ sudo nano /usr/share/applications/netbeans.desktop
```

Add the following data.

```
1 | [Desktop Entry]
2 | Name=Netbeans IDE
3 | Comment=Netbeans IDE
4 | Type=Application
5 | Encoding=UTF-8
6 | Exec=/opt/netbeans/bin/netbeans
7 | Icon=/opt/netbeans/nb/netbeans.png
8 | Categories=GNOME;Application;Development;
9 | Terminal=false
10 | StartupNotify=true
```

To uninstall NetBeans we must delete the `netbeans/` folder that is inside the `/opt/` folder, we can use the command:

```
1 | $ sudo rm /opt/netbeans -rf
```

Step 6: Correctly configure the JDK (optional)

In the file `/opt/netbeans/etc/netbeans.conf` we must correctly specify the path of our JDK in the variable `netbeans_jdkhome`. In GNU/Linux we can know the available JDKs with the command `sudo update-alternatives --config java` which will show a result similar to this:

```

1 Hi ha 3 possibilitats per a l'alternativa java (que proveeix /usr/bin/java).
2
3 Selecció      Camí                                     Prioritat  Estat
4 -----
5 * 0           /usr/lib/jvm/java-14-openjdk-amd64/bin/java    1411      mode
6 automàtic
7 1           /usr/lib/jvm/java-11-openjdk-amd64/bin/java    1111      mode
8 manual
9 2           /usr/lib/jvm/java-14-openjdk-amd64/bin/java    1411      mode
10 manual
11 3           /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  1081      mode
12 manual
13
14 Premu retorn per a mantenir l'opció per defecte[*], o introduïu un número de
15 selecció:

```

In the netbeans configuration you do not have to specify the end of the path `bin/java`

```
1 netbeans_jdkhome="/usr/lib/jvm/java-11-openjdk-amd64/"
```

6.1.2. Install from script

Step 1: Download the NetBeans file

You can also install Netbeans 12.4 on GNU/Linux from a script provided to download `Apache-NetBeans-12.4-bin-linux-x64.sh`.

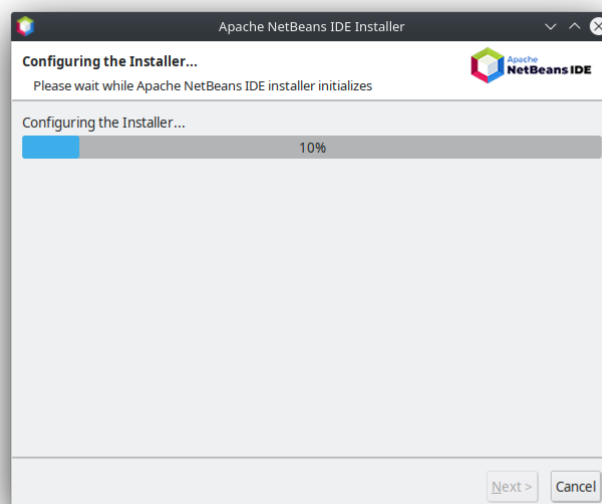
Step 2: Run the script

You must run the installation script

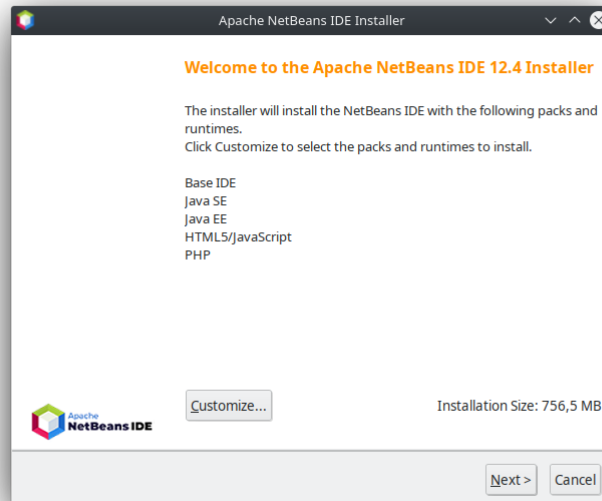
```
1 $ sudo sh ./Apache-NetBeans-12.4-bin-linux-x64.sh
```

If you run the script as `root` (`sudo`) Netbeans will be available to all users. By contrast, if you run the user without `sudo` it will only be available to your user.

A progress bar will appear like this:

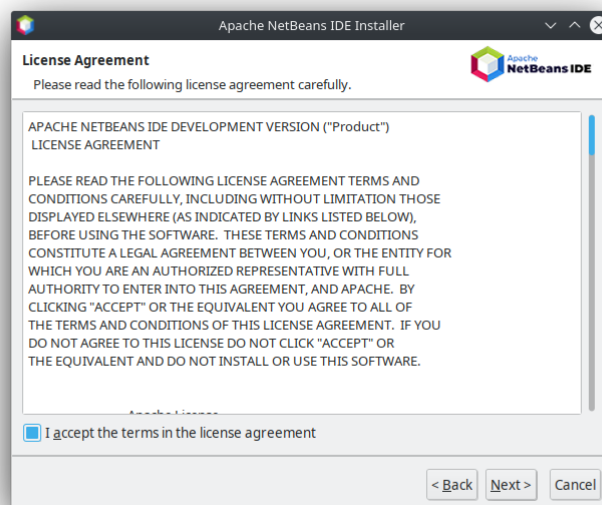


Now we can choose the components we want to install with the Netbeans IDE, we will leave it by default and click on the next button.



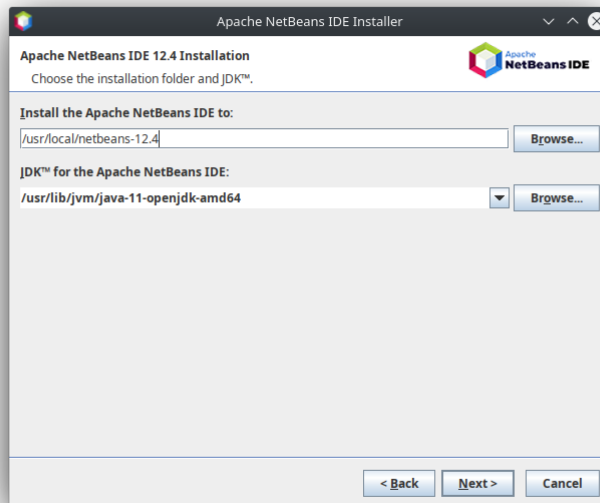
Step 3: Accept the license

Then we must accept the use license agreement by checking the box and pressing the next button.



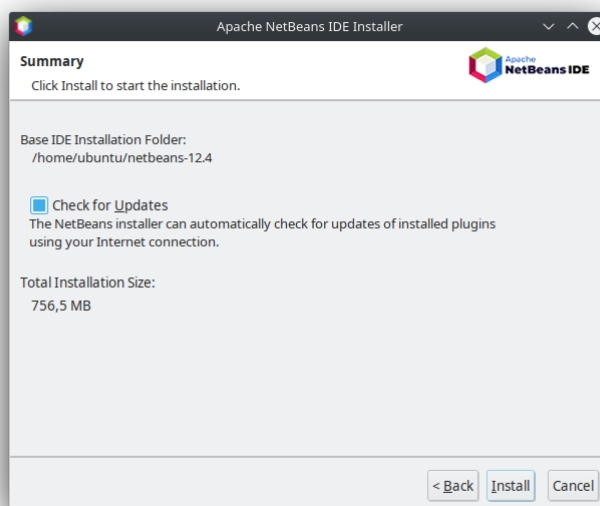
Step 4: Choose the installation path and JDK

Now we must choose the path where Netbeans 12.4 will be installed. And we must choose the path where the JDK is located (by default it indicates `/usr`, but we must specify the location as for example `/usr/lib/jvm/java-11-openjdk-amd64`).



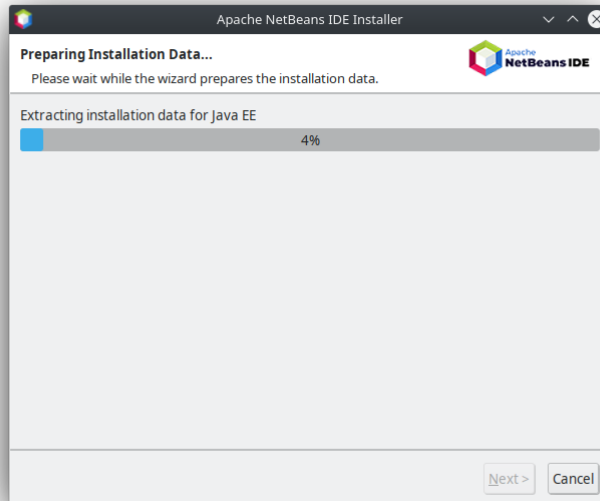
Paso 5: Actualizaciones automáticas

At this point a summary of the installation is shown, and we can choose if we want NetBeans to search for and install updates from the Internet, and click the install button.



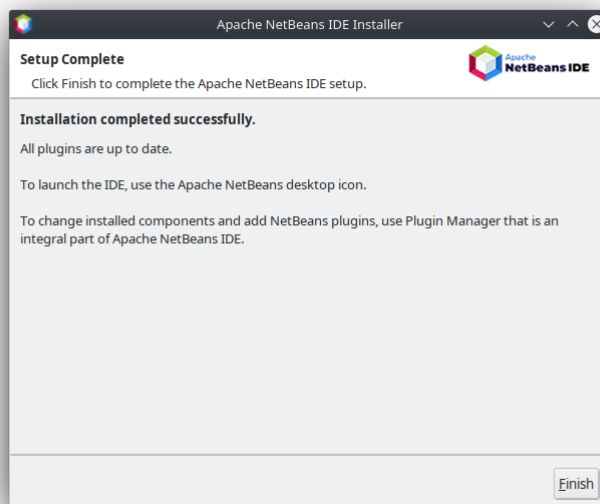
Step 6: Installation

A progress bar will appear.



Step 7: Final step

When finished, a screen will appear with the actions carried out by the installer, and we will have the launchers created in the applications menu.



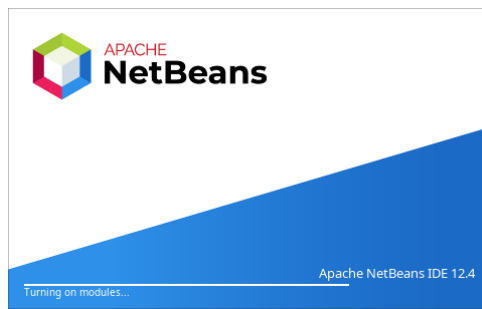
6.1.3. Install via snap

Perhaps an easier way to install the latest version of Netbeans on our GNU/Linux system is through `snap`:

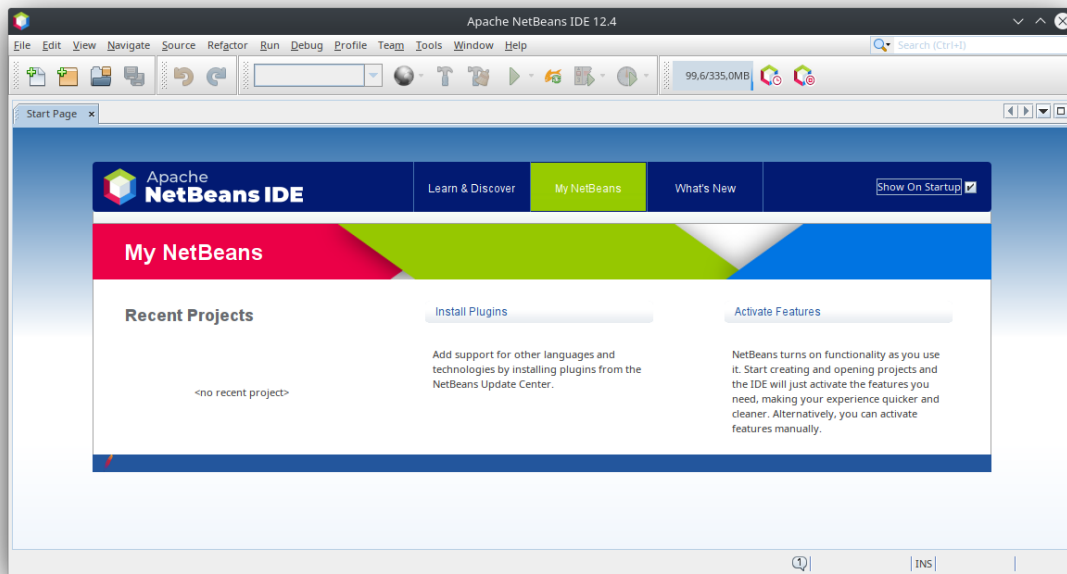
```
1 | $ sudo snap install netbeans --classic
```

6.1.4. First run

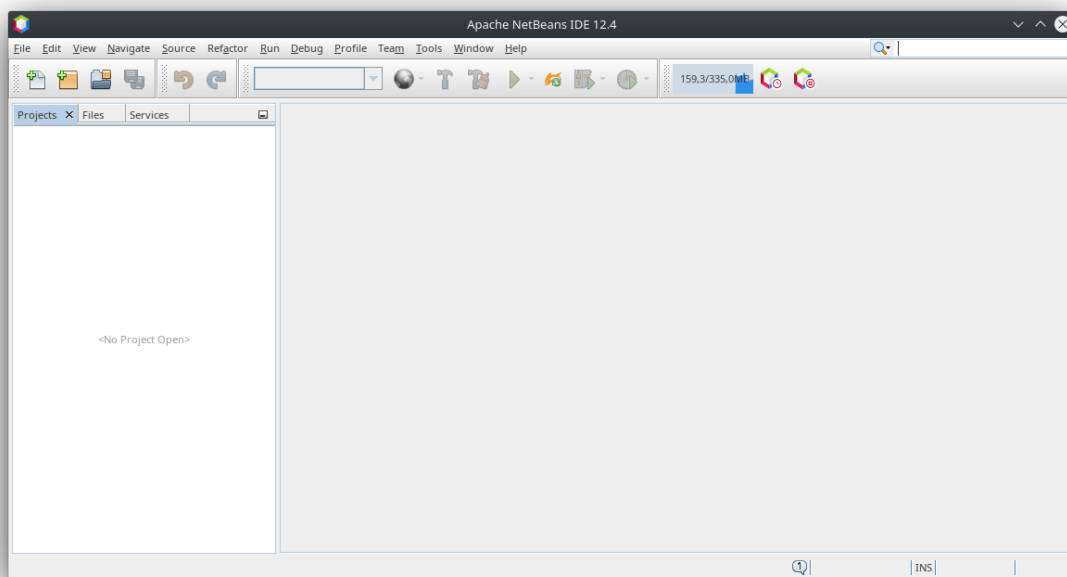
When we run the environment, a screen like the following will appear:



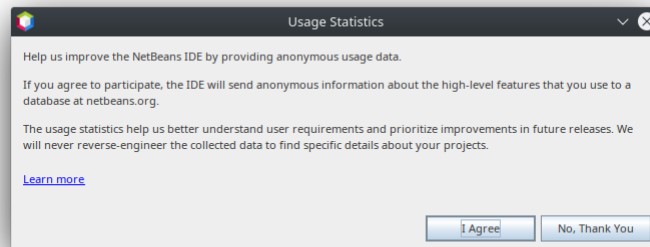
The first time we run it, the welcome tab will be displayed. We can ask that it no longer be shown to us by deactivating the "Show on Startup" option.



Once this tab is closed, the work environment will be similar to this:



NetBeans may ask us for permission to use our information on a statistical level, we choose the desired behavior and we agree.



To uninstall NetBeans in this case we must execute the `uninstall.sh` file found in the installation folder.

6.2. Setting

Activate Modules

By default Netbeans has the modules deactivated and it will be the first time we need them when they become active and available. For example, if we create a new project and choose `Java Application` within the `Java with Ant` category, we will see at the bottom that Netbeans warns us that the necessary module is not active and that we must press `Next` to make it available. . We do it, and then it will ask us to activate the `nb-javac Impl` module, we leave the check marked and press the `Activate` button, and the wizard for creating our first Java project will appear.

Java version

Within the `Tools/Java Platforms` menu we can change or view the location of our JDK installation.

Outlook

In Netbeans perspectives are not necessary, the Netbeans environment, although it is customizable, automatically adapts to the tasks you are doing at any given time.

Appearance

Netbeans allows us to customize any aspect of the appearance of our environment, change the IDE theme as well as the font size and colors for the source code coloring. All these options are available in `Tools/Options`, and within this window choose the third tab `Font & Colors` and the penultimate tab `Appearance`.

Export/Import configuration

A very interesting option of Netbeans is that it allows us to export or import configurations and share them with other colleagues or even between our computers or different installations. The option is available in `Tools/Options`, at the bottom left we find the `Export ...` and `Import ...` buttons.

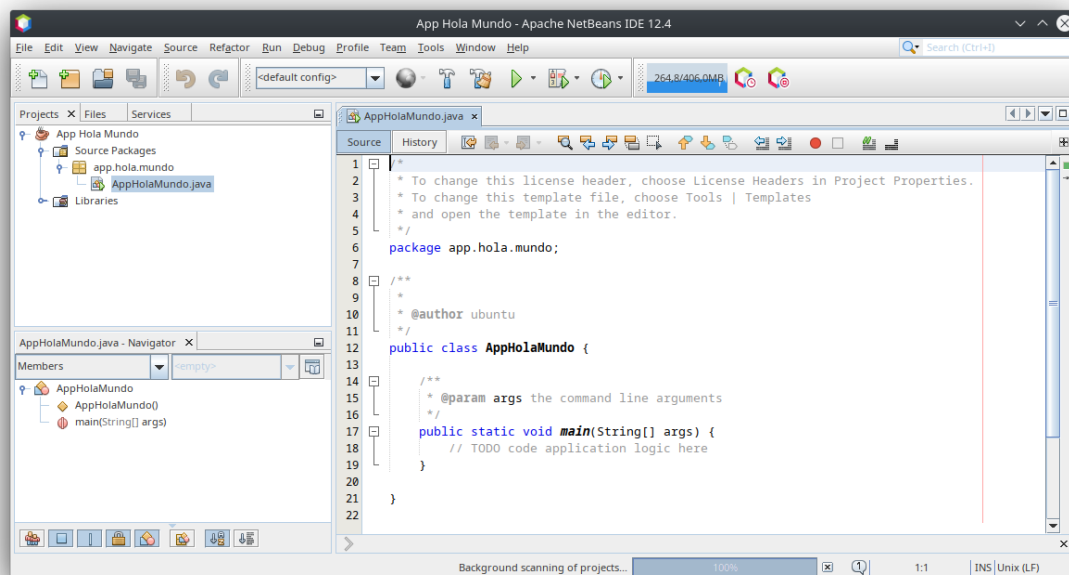
6.3. Modules

The options and functionalities of Netbeans can be expanded by adding modules from its plugins section. In `Tools/Plugins` we can for example search by text, or search in the available plugins tab. That will show us all the plugins that contain the searched word, or the plugins available. We can install for example `sonarlint4netbeans` that helps us to keep our code clean of common errors, for this we simply have to check the box in front of the name of the plugin, and

press the **INSTALL** button that appears below, click next, accept the license of use and install. When the installation is finished it will ask us to restart the **IDE**.

6.4. Basic usage ("Hello world!")

To create our first application in Netbeans, we must create a Java application, from the **File/New Project ...** menu we must choose **Java Application** within the **Java with Ant** category. Next we must specify the name of the project, for example "App Hello World", and we make sure to leave the option **Create main class app.hello.world.AppHelloWorld** checked and we should see something like this:



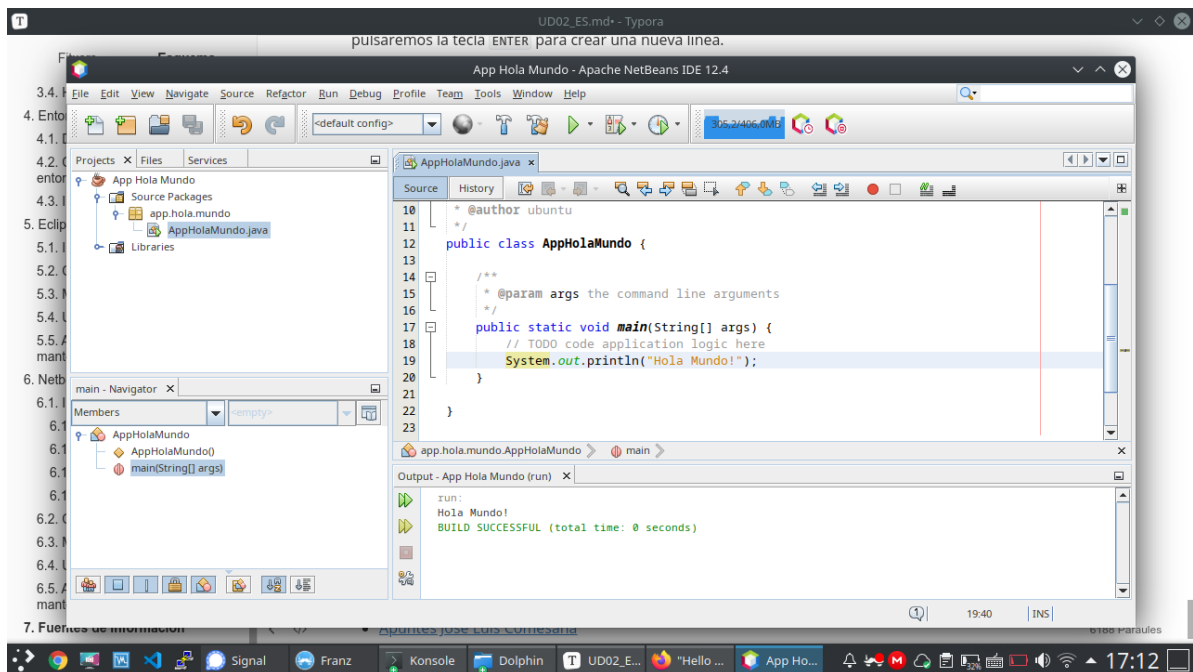
At this point, we can only include the line of code necessary to print the text message on the screen. To do this, we will go to the end of the line `// TODO code application logic here` and press the **ENTER** key to create a new line.

Once located in the right place we will use one of the most interesting features of Netbeans, which is the code templates. We type the word "sout" and then we press the **TAB** key and Netbeans will replace it with the correct code: `System.out.println ("");`.

Now we must write between the two double quotes the text message that should appear on the screen, and it should look like this:

```
1 System.out.println("Hola Mundo!");
```

Then we can press the upper button with a green triangle (**Run project**) or press the **F6** key on the keyboard:



A new section will appear in the window (at the bottom) called `Output` in which we can view the result of the execution of our first program.

6.5. Update and maintenance

In the `Help` section, Netbeans provides us with the options to update Netbeans itself with the `Help/Check for Updates` option.

7. Sources of Information

- [Wikipedia](#)
- [Code&Coke \(Fernando Valdeón\)](#)
- Apuntes IES El Grao (M^a Isabel Barquilla?)
- [Apuntes IOC \(Marcel García\)](#)
- [Apuntes José Luis Comesaña](#)
- [Apuntes IES Luis Vélez de Guevara 17-18 \(José Antonio Muñoz Jiménez\)](#)