Instalación y uso de entornos de desarrollo.

Caso práctico

Tras el éxito del anterior proyecto, en BK están recibiendo más peticiones de creación de software que nunca. Ana y Antonio, que ya hace unas semanas que están estudiando el Ciclo de Diseño de Aplicaciones Multiplataforma, piensan que este es un buen momento para participar activamente en los proyectos, pues a sus compañeros no les vendría nada mal un poco de ayuda.

Ada confía en ellos, pero aún es pronto. Por lo menos, ya conocen las fases por las que tiene que pasar todo el desarrollo de aplicaciones, pero eso no será suficiente.

María, sin embargo, no piensa lo mismo y decide darles una oportunidad trabajando en la fase de codificación de un nuevo proyecto de la empresa.

Ana se muestra muy ilusionada y no piensa desperdiciar esta gran oportunidad. Sabe que tiene a su disposición los llamados entornos de desarrollo que le facilitarán su futura tarea.

¿Cómo influirá el conocimiento de esta herramienta en el futuro de Ana y Antonio? A través de esta unidad, veremos si nuestros amigos van logrando ganarse un puesto en la empresa, y de paso, la confianza de Ada.

La fase de codificación es compleja, pero Ana y Antonio están aprendiendo a dominar los llamados entornos integrados de desarrollo de software.

1.- Concepto de entorno de desarrollo. Evolución histórica.

Caso práctico

Todos en la empresa están sorprendidos del entusiasmo de Ana ante los nuevos proyectos que BK programación tiene por delante. Juan, que acabó el Ciclo Superior de Desarrollo de Aplicaciones Informáticas (DAI) hace algunos años, se muestra inquieto porque es consciente de que en sólo unos cuatro años han salido muchas herramientas nuevas en el mercado y necesita reciclarse. Escucha a Ana decir que está estudiando los entornos de desarrollo.

—Yo también debería ponerme al día —piensa Juan.

En la unidad 1 se trataron las fases a seguir en un proceso de desarrollo de software.

La fase de codificación se puede llevar a cabo casi exclusivamente con un editor de texto y un compilador. Pero prácticamente la totalidad de programadores, terminan haciendo uso de algún entorno de desarrollo integrado para crear aplicaciones.

Un entorno integrado de desarrollo (**IDE**), es un tipo de software compuesto por un conjunto de herramientas de programación.

En concreto, el **IDE** entre otras aplicaciones se compone de:

- Editor de código de programación.
- Accesos al compilador desde botones u opciones de menu.
- Acceso a la ejecución del programa desde botones u opciones de menu.
- Depurador.
- Constructor de interfaz gráfico.

Los primeros entornos de desarrollo integrados nacieron a principios de los años 70, y se popularizaron en la década de los 90.

Tienen el objetivo de ganar fiabilidad y tiempo en los proyectos de software. Proporcionan al programador una serie de componentes con la misma interfaz gráfica, con la consiguiente comodidad, aumento de eficiencia y reducción de tiempo de codificación.

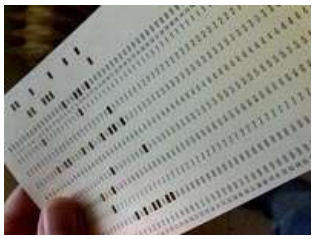
Normalmente, un **IDE** está dedicado a un determinado lenguaje de programación. No obstante, las últimas versiones de los **IDE** tienden a ser compatibles con varios lenguajes (por ejemplo, **Eclipse**, **NetBeans**, **Microsoft Visual Studio**) mediante la instalación de plugins adicionales.

En este tema, nuestro interés se centra en conocer los entornos de desarrollo, los tipos (en función de su licencia y del lenguaje de programación hacia el cual están enfocados). Veremos cómo se configuran y cómo se generan ejecutables, haciendo uso de sus componentes y herramientas.

Reflexiona

Según datos, casi todas las personas que empiezan a programar utilizan un editor simple de textos y un compilador-depurador instalado en su equipo. Sin embargo, prácticamente todas acaban utilizando un entorno de desarrollo.

1.1.- Evolución Histórica.



En las décadas de utilización de la tarjeta perforada como sistema de almacenamiento el concepto de **Entorno de Desarrollo Integrado** sencillamente no tenía sentido.

Los programas estaban escritos con diagramas de flujo y entraban al sistema a través de las **tarjetas perforadas**. Posteriormente, eran compilados.

El primer lenguaje de programación que utilizó un IDE fue el BASIC (que fue el primero en abandonar también las tarjetas perforadas o las cintas de papel).

Éste primer IDE estaba basado en consola de comandos exclusivamente (normal por otro lado, si tenemos en cuenta que hasta la década de los 90 no entran en el mercado los sistemas operativos con interfaz gráfica). Sin embargo, el uso que hace de la gestión de archivos, compilación y depuración; es perfectamente compatible con los IDE actuales.

A nivel popular, el primer IDE puede considerarse que fue el IDE llamado **Maestro**. Nació a principios de los 70 y fue instalado por unos 22.000 programadores en todo el mundo. Lideró este campo durante los años 70 y 80.

El uso de los entornos integrados de desarrollo se ratifica y afianza en los 90 y hoy en día contamos con infinidad de IDE, tanto de licencia libre como no.

Tipos de entornos de desarrollo más relevantes en la actualidad.

Entorno de desarrollo	Lenguajes que soporta	Tipo de licencia
NetBeans.	C/C++, Java, JavaScript, PHP, Python.	De uso público.
Eclipse.	Ada, C/C++, Java, JavaScript, PHP.	De uso público.
Microsoft Visual Studio.	Basic, C/C++, C#.	Propietario.
C++ Builder.	C/C++.	Propietario.
JBuilder.	Java.	Propietario.

No hay unos entornos de desarrollo más importantes que otros. La elección del IDE más adecuado dependerá del lenguaje de programación que vayamos a utilizar para la codificación de las aplicaciones y el tipo de licencia con la que queramos trabajar.

2.- Funciones de un entorno de desarrollo.

Caso práctico

Juan, que asume por fin su desconocimiento, habla con Ana para que le pase sus apuntes de entornos de desarrollo. Ésta se muestra encantada, y le anima a matricularse al ciclo de Desarrollo de Apliaciones Multiplataforma (DAM) a distancia. Juan se muestra reacio (ya he estudiado el ciclo y durante cuatro años he cumplido con éxito en la empresa). Pero piensa que quizás debería reciclarse si no quiere quedarse atrás en los proyectos Juan aprendió a programar usando un editor simple de textos, ¿qué ventajas tendrá programando con un IDE?

Como sabemos, los entornos de desarrollo están compuestos por una serie de herramientas software de programación, necesarias para la consecución de sus objetivos. Estas herramientas son:

Un editor de código fuente.
Un compilador y/o un intérprete.
Automatización de generación de herramientas.
Un depurador.

Las funciones de los IDE son:

Editor de código: coloración de la sintaxis.
Auto-completado de código, atributos y métodos de clases.
Identificación automática de código.
Herramientas de concepción visual para crear y manipular componentes visuales.
Asistentes y utilidades de gestión y generación de código.
Organización de los archivos fuente en unas carpetas y compilados a otras.
Compilación de proyectos complejos en un solo paso.
Control de versiones: tener un único almacén de archivos compartido por todos los colaboradores de un proyecto. Ante un error, mecanismo de auto-recuperación a un estado anterior estable.
Soporta cambios de varios usuarios de manera simultánea.
Generador de documentación integrado.
Detección de errores de sintaxis en tiempo real.

Otras funciones importantes son:

Ofrece refactorización de código: cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo cambiar el nombre a una variable).
Permite introducir automáticamente tabulaciones y espaciados para aumentar la legibilidad.
Depuración: seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
Aumento de funcionalidades a través de la gestión de sus módulos y plugins.
Administración de las interfaces de usuario (menús y barras de herramientas).
Administración de las configuraciones del usuario.
Empaquetar software para su posterior despliegue o instalación en el entorno de ejecución.

Autoevaluación

Un entorno integrado de desarrollo está compuesto por:

- ○ Editor de código y traductor.
- ○ Editor de código, compilador e interfaz de comandos.
- ○ Editor de código, compilador, intérprete, depurador e interfaz gráfica.
- ○ Interfaz gráfica, editor de código y depurador.

Incorrecta, se compone de más herramientas.

No es correcta porque la interfaz es gráfica.

Muy bien. Esa es la idea.

No es del todo correcta: faltaría el traductor de código (compilador e intérprete).

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

3.- Entornos integrados libres y propietarios.

Caso práctico

Juan ha buscado por Internet distintos entornos de desarrollo para aplicarlos en la fase de codificación.

—Cuidado —le dice Ada—. Ya sabes que es de vital importancia el tema de la Licencia de Software. Hay Entornos de desarrollo de licencia libre y otros no, y este aspecto es fundamental ni no queremos tener problemas.

Entornos Integrados Libres

Entornos Integrados libres son aquellos con licencia de uso público. No hay que pagar por ellos, y aunque los más conocidos y utilizados son **Eclipse** y **NetBeans**, hay bastantes más.

Por otra parte, los entornos integrados de desarrollo propietarios necesitan licencia. No son free software, hay que pagar por ellos. El más conocido y utilizado es **Microsoft Visual Studio**, desarrollado por **Microsoft** (sólo disponible en plataformas **Windows**).

La mayor parte de ellos, aunque no todos, están disponibles en diferentes plataformas.

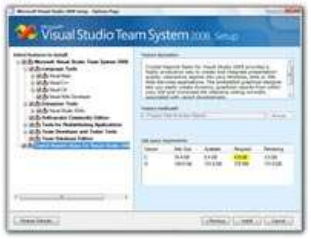
Entornos de desarrollo libres más relevantes en la actualidad		
IDE	**Algunos lenguajes que soporta**	**URL**
Eclipse	Ada, C/C++, Java, JavaScript, PHP	https://www.eclipse.org/
NetBeans	C/C++, Java, JavaScript, PHP, HTML5 ...	*https://netbeans.org/*
CodeLite	C/C++, PHP, Node.js	https://codelite.org
JDeveloper	Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, UML ...	http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html
IntelliJ	Java, Groovy, Perl, Scala,XML/XSL,Python ...	https://www.jetbrains.com/idea/
Microsoft Visual Studio	C#, Visual Basic, F#, C++, HTML, JavaScript, TypeScript, Python, ...	https://visualstudio.microsoft.com/es/vs/community/ https://code.visualstudio.com/

Para saber más

En el siguiente enlace encontrarás un documento muy interesante, en inglés, donde se detallan todos los entornos de desarrollo existentes en la actualidad con todas sus características: licencias, sistemas operativos donde pueden ser instalados y configurados, lenguajes que soporta, desarrolladores y última versión estable.

Entornos de desarrollo actuales.

Entornos Integrados Propietarios



Son aquellos entornos integrados de desarrollo que necesitan licencia. No son free software, hay que pagar por ellos.

El más conocido y utilizado es Microsoft Visual Studio, que usa el framework .NET y es desarrollado por Microsoft.

Entornos de desarrollo propietarios más relevantes en la actualidad		
IDE	**Algunos lenguajes que soporta**	**URL**
Microsoft Visual Studio	C++, C#, Visual Basic ...	https://visualstudio.microsoft.com/es/
C++ Builder	C++	https://www.embarcadero.com/es/
IntelliJ	Java, Groovy, Perl, Scala, ML/XSL,Python, Ruby, Sql ...	https://www.jetbrains.com/idea/
QtCreator	C++ con framework QT	https://www.qt.io/

Autoevaluación

Relaciona los siguientes entornos de desarrollo con sus características, escribiendo el número asociado a la característica en el hueco correspondiente.

Ejercicio de relacionar

Entorno de desarrollo.	Relación	Características.
Microsoft Visual Studio.	☐	1. Libre. Soporta C/C++, Java, PHP, Javascript, Python.
NetBeans.	☐	2. Propietario. Soporta Basic, C/C++, C#.
C++ Builder.	☐	3. Propietario. Soporta C/C++.

Enviar

En la elección del entorno de desarrollo más adecuado para desarrollar un proyecto de software influye el tipo de licencia del entorno y los lenguajes de programación que soporta.

4.- Estructura de entornos de desarrollo.

Caso práctico

Juan aprendió a programar utilizando un editor de textos, un compilador y un depurador. Todas estas herramientas se instalaban de forma independiente. A Ana le cuesta creer que los programadores tuvieran que buscar estas herramientas e instalarlas por separado. –En un entorno se integran todas estas cosas y muchas más, y sin salir del mismo puedes programar en varios lenguajes y puedes documentar y…. —Ya lo veo, —le replica Juan—. ¿Cuántos componentes tiene el entorno en total?

Los entornos de desarrollo, ya sean libres o propietarios, están formados por una serie de componentes software que determinan sus funciones.

Estos componentes son:

Componentes	Funciones
Editor de textos.	Resaltado y coloreado de la sintaxis del código. Funciones de completado automático de código. Inserción automáticamente paréntesis, corchetes, tabulaciones y espaciados. Ayuda y listado de parámetros de funciones y métodos de clase
Compilador/intérprete.	Detección de errores de sintaxis en tiempo real.
Depurador.	Ejecución del programa paso a paso, definición de puntos de ruptura y seguimiento de variables. Opción de depurar en servidores remotos.
Generador automático de herramientas.	Herramientas para la visualización, creación y manipulación de componentes visuales y todo un arsenal de asistentes y utilidades de gestión y generación código.
Interfaz gráfica.	Brinda la oportunidad de programar en varios lenguajes con un mismo IDE. Es una interfaz agradable que puede acceder a innumerables bibliotecas y plugins, aumentando las opciones de nuestros programas.

Para saber más

En el siguiente enlace accederás a una página web donde se detallan todos los componentes del entorno de desarrollo, junto con sus funciones.

Estructura de Entornos de Desarrollo

5.- Instalación de entornos integrados de desarrollo.

Caso práctico

Juan está decidido a aprender a usar un entorno de desarrollo. Después de documentarse, piensa que lo idóneo es trabajar con un IDE libre. Además, el tema del sistema operativo que soporta es importante. Juan quiere trabajar bajo Linux, y se decide por el entorno NetBeans. Ahora bien, ¿Qué hay que hacer para instalarlo?

Se va a describir cómo instalar **Netbeans. IDE** con mucha presencia en el mercado y de libre distribución. Dispone de distribuciones bajo diferentes plataformas.

Para compilar los programas en **Java** desde este IDE que se verán en las próximas secciones, habrá que haber instalado primero el paquete **JDK** de **Java**.

5.1.- Instalación de JDK.

La instalación del IDE NetBeans, ya sea en Linux, Windows o Mac OS X, requiere la instalación previa del JDK compatible con la versión de NetBeans que se quiera instalar.

JDK son las siglas de Java Development Kit: Kit de desarrollo de Java. Consiste en la plataforma del entorno, imprescindible para que éste pueda ser instalado y ejecutado.

En el caso del lenguaje **Java**, se indicaba en temas anteriores que tras la compilación del código fuente se obtiene otro llamado **bytecode**. Para que el **bytecode** pueda ser interpretado, el equipo deberá tener instalado el **JRE (Java Runtime Environment)**, definido en wikipedia como sigue:

> **JRE** es un conjunto de utilidades que permite la ejecución de programas Java.
>
> En su forma más simple, el entorno en tiempo de ejecución de Java está conformado por una **Máquina Virtual de Java o JVM**, un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.
>
> La **JVM** es el programa que ejecuta el código Java previamente compilado (bytecode) mientras que las librerías de clases estándar son las que implementan el API de Java. Ambas JVM y **API** deben ser consistentes entre sí, de ahí que sean distribuidas de modo conjunto.
>
> Un usuario sólo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JDK, que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java.

El **JRE** es desarrollado y distribuido de forma gratuita por **Oracle**.

Pero si nuestra voluntad es convertirnos en desarrolladores de código **Java**, no será suficiente el **JRE**, tendremos que instalar el **JDK (Java Development Kit)**, también distribuido por Oracle y que wikipedia define como:

> **Java Development Kit (JDK)** es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red.
>
> Los programas más importantes que se incluyen
>
> - **appletviewer.exe**: es un visor de applets para generar sus vistas previas, ya que un applet carece de método main y no se puede ejecutar con el programa java.
> - **javac.exe**: es el compilador de Java.
> - **java.exe**: es el masterescuela (intérprete) de Java.
> - **javadoc.exe**: genera la documentación de las clases Java de un programa.

Para instalar el JDK accede a la siguiente dirección:

https://www.oracle.com/technetwork/es/java/javase/overview/index.html

En el apartado de Updates verás las distintas versiones del JDK de Java SE (Standard Edition). Un número más alto indica que es más reciente. Elige una de ellas y descarga el fichero adecuado para tu plataforma. En la página de descarga también vienen las instrucciones de instalación por si tuvieses algún problema.
Apunta la ruta donde se instala el JDK pues lo necesitarás más adelante.

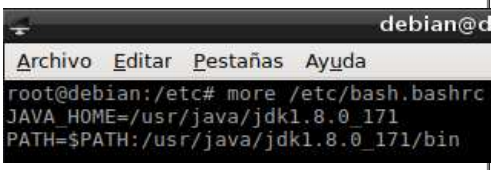
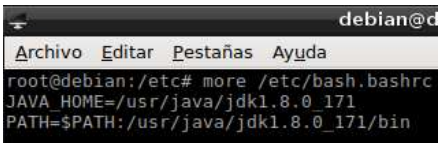
Configuración de las variables de entorno

El **JDK** podrá ser utilizado por diversas aplicaciones entre las que se encuentran los **IDEs.** Al ser la ubicación del **JDK** en el sistema de archivos configurable durante la instalación, las aplicaciones que hacen uso del mismo no saben donde localizarlo.

Para resolverlo, se definen variables de entorno, cuyo nombre será de conocimiento público y por tanto común para cualquier aplicación que quiera utilizarlas. En particular, las propuestas a continuación tienen como cometido informar de las rutas elegidas en la instalación para el **JDK** y facilitar el acceso a sus ejecutables.

Configuración de las variables de entorno en Linux (con bash)

Desde un terminal en **Linux** se dan de alta/modifican las variables de entorno **JAVA_HOME** y **PATH**.

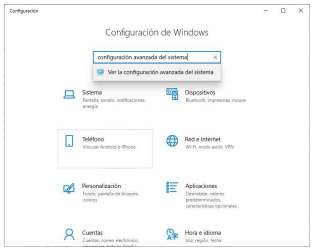
Acción	Orden en consola linux
Abrir un terminal **Linux.**	
Ganar privilegios de administrador.	sudo su + Contraseña
Con un editor de texto (por ejemplo nano), acceder al fichero /etc/bash.bashrc.	nano /etc/bash.bashrc
Introducir las variables de entorno: **JAVA_HOME=/usr/java/jdk1.8.0_171** **PATH=$PATH:/usr/java/jdk1.8.0_171/bin** **En este caso, la ruta del jdk en JAVA_HOME y también añadimos la ruta de los ejecutables en PATH**	 

Tras reiniciar, se puede comprobar que han quedado activadas desde terminal con los comandos:

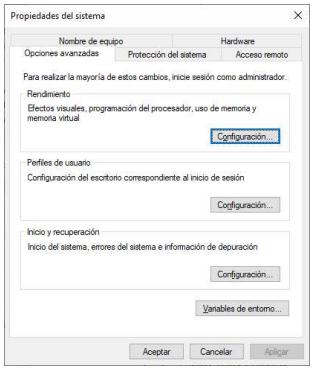
- **echo $PATH**
- **echo $JAVA_HOME**

Configuración de las variables de entorno en Windows (ejemplo en Windows 10)

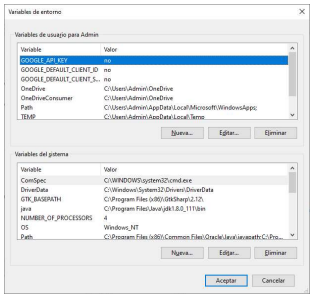
1. Ir a configuración de Windows.
2. Poner en el buscador: "configuración avanzada del sistema":



Se visualizará esta ventana:



A continuación, se pulsa en "variables de entorno"



Después elegimos Nueva (en variables de Sistema) e introducimos los datos de la variable de entorno, su nombre: JAVA_HOME y su valor (la ruta donde está instalado el jdk). En este ejemplo es un jdk 8. Después Aceptar.

Como la variable PATH ya existe, seleccionamos la variable PATH y elegimos Editar. Añadimos la ruta del directorio bin de nuestro jdk. Ponemos un punto y coma y luego %JAVA_HOME%\bin. Para finalizar, Aceptar.

Configuración de las variables de entorno en MAC OS

En este enlace puedes ver un ejemplo:

https://sintaxispragmatica.wordpress.com/2014/04/07/establecer-la-variable-java_home-en-mac-osx/

Habría que editar también la variable PATH:

https://professor-falken.com/mac/como-anadir-una-nueva-ruta-a-la-variable-path-en-tu-mac/

Autoevaluación

En tu opinión, ¿Por qué crees que la instalación del JDK sólo la puede realizar el root del sistema?

- ○ Porque se trata de un archivo binario de sistema.
- ○ Porque ningún archivo puede ser ejecutado por un usuario que no sea el root.
- ○ Porque estamos trabajando en la terminal del sistema.

Efectivamente. Es un archivo con extensión .bin y sólo puede ser manipulado por el root.

No. Los ejecutables normales pueden ser ejecutados por cualquiera que tenga permisos para ello.

Incorrecta. La terminal es una forma de trabajar cuando tenemos que modificar permisos y realizar acciones que no podemos desde la interfaz gráfica.

Solución

1. Opción correcta
2. Incorrecto
3. Incorrecto

5.2.- Instalación de entornos de desarrollo.

Una vez tenemos instalado el JDK en nuestro equipo, ya tenemos preparado el contexto en el que se instalará el entorno NetBeans.

Recuerda que para trabajar con un IDE que trabaja con lenguaje Java, necesitarás tener el JDK instalado.

5.2.1.- Instalación de NetBeans.

Caso práctico

Juan ya ha instalado el JDK.

—Uff, me ha costado un poco… —le comenta a Ana. —Hace tiempo que no trabajaba en la terminal de Linux y se me habían olvidado algunas órdenes básicas. Ana le comenta que ya tiene el equipo preparado para instalar NetBeans. Decide pasarle los apuntes del ciclo a distancia para que Juan no tenga que perder mucho tiempo buscando los comandos necesarios.

En el caso del IDE Netbeans, instalaremos la última versión disponible en la siguiente dirección:

https://netbeans.apache.org/download/index.html

Elige el adecuado para tu plataforma. Descarga el instalador y ejecútalo. No es necesario configurar nada. Al instalarse puedes elegir dónde instalarlo y decirle dónde está el jdk (lo detectará si lo tienes instalado).

En el "Anexo III.- Instalación JDK y NetBeans" se explica detalladamente como instalar el JDK y el NetBeans conjuntamente.

5.2.2.- Instalación de Eclipse.

Recuerda que necesitarás tener el JDK instalado.

Descarga de Eclipse:

https://www.eclipse.org/downloads/packages/

Elige el Eclipse IDE for Java developers para tu plataforma. Si tienes problemas con la última versión de Eclipse, en esa misma página puedes encontrar versiones anteriores que quizá te sirvan.

Te habrás descargado un zip. Descomprime el fichero y ejecuta eclipse con el fichero ejecutable llamado eclipse. Al arrancar, detecta automáticamente si el JDK está instalado. A continuación, deberás elegir el **Workspace.**

Selección del Workspace de trabajo.

Eclipse organiza los proyectos en espacios de trabajo. Se trata de poner en común diferentes bloques de código bajo algún criterio en particular (funcionalidades comunes, mismo cliente, clases desarrolladas por un mismo programador ...).

Eclipse asocia cada **Workspace** con un directorio en el sistema de archivos, a partir del cual irán "colgando" nuevos directorios, uno por proyecto creado.

Al iniciar **Eclipse** solicita la ubicación del **Workspace**. Elige la que consideres oportuna.

Una vez haya arrancado puedes configurar el idioma si lo deseas

Configuración del idioma.

Para poner **Eclipse** en español se va a utilizar la solución propuesta en el proyecto **babel** de **Eclipse**. Los pasos a seguir podrás encontrarlos en la url:

https://www.eclipse.org/babel/downloads.php

Bajo la etiqueta de "**Installing the language packs**", se dan las siguientes instrucciones.

- En **Eclipse** accede al asistente Help/Install New Softwaree
- Añade el repositorio **Babel** de **Eclipse**.

https://download.eclipse.org/technology/babel/update-site/R0.17.0/2019-06/

No te impacientes, la carga se toma un poco tiempo mostrando el mensaje Pending

- Una vez cargado, seleccionar los paquetes **babel para Eclipse in spanish** y pulsar next
- Reinicia **Eclipse**.

6.- Configuración y personalización de entornos de desarrollo.

Caso práctico

Juan está consternado. NetBeans parece albergar tanta información que no sabe por donde empezar. Le gustaría personalizar la configuración de su primer proyecto en el IDE (que va a ser un aplicación de Java). ¿Cómo lo hace? ¿Qué parámetros puede configurar?

Una vez tenemos instalado nuestro entorno de desarrollo podemos acceder a personalizar su configuración.

Al abrir un proyecto existente, o bien crear un nuevo proyecto, seleccionaremos un desplegable con el nombre de “configuración” desde el que podremos personalizar distintas opciones del proyecto.

Podemos personalizar la configuración del entorno sólo para el proyecto actual, o bien para todos los proyectos, presentes y futuros.

Parámetros configurables del entorno:

Carpeta o carpetas donde se alojarán todos los archivos de los proyectos (es importante la determinación de este parámetro, para tener una estructura de archivos ordenada).
Carpetas de almacenamiento de paquetes fuente y paquetes prueba.
Administración de la plataforma del entorno de desarrollo.
Opciones de la compilación de los programas: compilar al grabar, generar información de depuración.
Opciones de empaquetado de la aplicación: nombre del archivo empaquetado (con extensión .jar, que es la extensión característica de este tipo de archivos empaquetados) y momento del empaquetado.
Opciones de generación de documentación asociada al proyecto.
Descripción de los proyectos, para una mejor localización de los mismos.
Opciones globales de formato del editor: número de espaciados en las sangrías, color de errores de sintaxis, color de etiquetas, opción de autocompletado de código, propuestas de insertar automáticamente código.
Opciones de combinación de teclas en teclado.
Etc.

En el "Anexo IV- Configuración y personalización de NetBeans" se explica en detalle como hacer estas configuraciones.

Debes conocer

Busca una guía donde explique cómo acceder a los parámetros de configuración personalizada de los proyectos en NetBeans, y las opciones entre las que podemos elegir para decidir cómo queremos trabajar en un proyecto software:

7.- Gestión de módulos.

Caso práctico

Después de haber probado a configurar algunos aspectos del entorno, ahora Juan desea empezar a programar. Tiene un trabajo pendiente en JavaScript, pero observa que, tristemente, este lenguaje no es soportado por NetBeans.

—¿Cómo que no? —Le dice Ana. —Basta con encontrar el módulo de JavaScript (estructuras del lenguaje más bibliotecas asociadas) y añadirlo como complemento al entorno. Entonces sí que podrás programar (también) en ese lenguaje.

A Juan le parece fascinante.

Con la plataforma dada por un entorno de desarrollo como NetBeans podemos hacer uso de módulos y plugins para desarrollar aplicaciones.

En la página oficial de NetBeans encontramos una relación de módulos y plugins, divididos en categorías.

Seleccionando la categoría Lenguajes de Programación, encontraremos aquellos módulos y plugins que nos permitan añadir nuevos lenguajes soportados por nuestro IDE.

Un módulo es un componente software que contiene clases de Java que pueden interactuar con las API del entorno de desarrollo y el manifest file, que es un archivo especial que lo identifica como módulo.

Los módulos se pueden construir y desarrollar de forma independiente. Esto posibilita su reutilización y que las aplicaciones puedan ser construidas a través de la inserción de módulos con finalidades concretas. Por esta misma razón, una aplicación puede ser extendida mediante la adición de módulos nuevos que aumenten su funcionalidad.

Existen en la actualidad multitud de módulos y plugins disponibles para todas las versiones de los entornos de desarrollo más utilizados. En las secciones siguientes veremos dónde encontrar plugins y módulos para NetBeans 6.9.1 que sean de algún interés para nosotros y las distintas formas de instalarlos en nuestro entorno.

También aprenderemos a desinstalar o desactivar módulos y plugins cuando preveamos que no los vamos a utilizar más y cómo podemos estar totalmente actualizados sin salir del espacio de nuestro entorno.

Veremos las categorías de plugins disponibles, su funcionalidad, sus actualizaciones…

Autoevaluación

¿Cómo crees que influye el hecho de tener módulos y plugins disponibles en el éxito que tenga un IDE?

- ○ Contribuyen al éxito del entorno.
- ○ No influyen en el éxito del entorno.

Efectivamente. Poder añadir funcionalidades concretas según lo que necesitemos hacen que el IDE sea muy aceptado por los usuarios.

No. La aceptación de un entorno será mayor si permite que el usuario decida qué funcionalidades desea incluir.

Solución

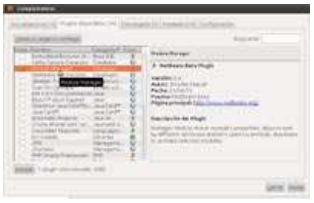
1. Opción correcta
2. Incorrecto

7.1.- Añadir.

Caso práctico

Ya sabemos que podemos añadir funcionalidades a nuestro entorno. Pero ni Juan ni Ana saben cómo hacerlo. Piden ayuda a María, que decide ayudarles.

—Añadir módulos y plugins es muy sencillo, prestad atención.



Añadir un módulo va a provocar dotar de mayor funcionalidad a nuestros proyectos desarrollados en NetBeans.

Para añadir un nuevo módulo tenemos varias opciones:

1. Añadir algún módulo de los que NetBeans instala por defecto.
2. Descargar un módulo desde algún sitio web permitido y añadirlo.
3. Instalarlo on-line en el entorno.

Por supuesto, una cuarta posibilidad es crear el módulo nosotros mismos (aunque eso no lo veremos aquí).

Sin embargo, lo más usual es añadir los módulos o plugins que realmente nos interesan desde la web oficial de NetBeans. El plugin se descarga en formato .nbm que es el propio de los módulos en NetBeans. Posteriormente, desde nuestro IDE, cargaremos e instalaremos esos plugins. A esta manera de añadir módulos se le conoce como adición off-line.

También es habitual instalarlos on-line, sin salir del IDE.

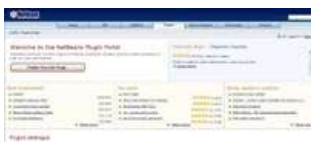
La adición on-line requiere tener instalado el plugin Portal Update Center en NetBeans 6.9.1 y consiste en instalar complementos desde nuestro mismo IDE, sin tener que descargarlos previamente.

Al final del tema, en "Anexo V (apartado a).- Adición de módulo en NetBeans" se explica en detalle como añadir un plugin en Netbeans.

Debes conocer

Navegar y familiarizarse por la plataforma web que NetBeans pone a disposición de los desarrolladores es fundamental para estar al día de las últimas funcionalidades que podemos añadir a nuestro entorno mediante la instalación de plugins

Búsqueda online de plugins para NetBeans



7.2.- Eliminar.

Cuando consideramos que algún módulo o plugin de los instalados no nos aporta ninguna utilidad, o bien que el objetivo para el cual se añadió ya ha finalizado, el módulo deja de tener sentido en nuestro entorno. Es entonces cuando nos planteamos eliminarlo.

Eliminar un módulo es una tarea trivial que requiere seguir los siguientes pasos:

1. Encontrar el módulo o plugin dentro de la lista de complementos instalados en el entorno.
2. A la hora de eliminarlo, tenemos dos opciones:
    1. Desactivarlo: El módulo o plugin sigue instalado, pero en estado inactivo (no aparece en el entorno).
    2. Desinstalarlo: El módulo o plugin se elimina físicamente del entorno de forma permanente.

Esta es la ventana, desde el gestor de complementos de NetBeans, que nos aparece cuando queremos eliminar un módulo del entorno.

Siempre nos pedirá elegir entre dos opciones: desactivar o desinstalar.

En este ejemplo, se opta por desactivar el complemento, como podemos ver en la imagen.

Al final del tema, en "Anexo VI.- Eliminar módulos en NetBeans" se detalla como quitar un plugin en NetBeans"

Autoevaluación

Para añadir un módulo desde la web oficial de NetBeans:

- ○ Hay que instalar el plugin Update Center.
- ○ Hay que conectar con la web desde Netbeans y instalar on-line.
- ○ Hay que encontrar el complemento, descargarlo y luego instalarlo en el IDE.
- ○ No se pueden descargar los complementos desde ahí.

Incorrecta, ese plugin sólo es necesario en adiciones on-line.

No es correcta porque la instalación tiene que ser off-line.

Muy bien. Esa es la idea.

Incorrecta. Sí se pueden descargar plugins y módulos para nuestro IDE.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

7.3.- Funcionalidades.

Caso práctico

—Para que sepas qué puedes encontrar en los complementos de NetBeans, te recomiendo que tengas claras las funcionalidades que ofrece, teniendo en cuenta que se van ampliando día a día, —le comenta Ana a Juan.

Los módulos y plugins disponibles para los entornos de desarrollo, en sus distintas versiones, tienen muchas y muy variadas funciones.

Podemos clasificar las distintas categorías de funcionalidades de módulos y plugins en los siguientes grupos:

1. **Construcción de código**: facilitan la labor de programación.
2. **Bases de datos**: ofrecen nuevas funcionalidades para el mantenimiento de las aplicaciones.
3. **Depuradores**: hacen más eficiente la depuración de programas.
4. **Aplicaciones**: añaden nuevas aplicaciones que nos pueden ser útiles.
5. **Edición**: hacen que los editores sean más precisos y más cómodos para el programador.
6. **Documentación de aplicaciones**: para generar documentación de los proyectos en la manera deseada.
7. **Interfaz gráfica de usuario**: para mejorar la forma de presentación de diversos aspectos del entorno al usuario.
8. **Lenguajes de programación y bibliotecas**: para poder programar bajo un Lenguaje de Programación que, en principio, no soporte la plataforma.
9. **Refactorización**: hacer pequeños cambios en el código para aumentar su legibilidad, sin alterar su función.
10. **Aplicaciones web**: para introducir aplicaciones web integradas en el entorno.
11. **Prueba**: para incorporar utilidades de pruebas al software.

Autoevaluación

¿Qué categoría de funcionalidad de NetBeans te parece más interesante? ¿Por qué?

- ○ Todas son igual de interesantes porque aumentan la funcionalidad.
- ○ Depende de la tarea a realizar y el nivel del usuario.

No es del todo correcto. Según el nivel del usuario y la tarea unas serán más importantes que otras.

Muy bien. Esa es la idea.

Solución

1. Incorrecto
2. Opción correcta

7.4.- Herramientas concretas.

Importador de Proyectos de NetBeans: permite trabajar en lenguajes como JBuilder.
Servidor de aplicaciones GlassFish: Proporciona una plataforma completa para aplicaciones de tipo empresarial.
Soporte para Java Enterprise Edition: Cumplimiento de estándares, facilidad de uso y la mejora de rendimiento hacen de NetBeans la mejor herramienta para crear aplicaciones de tipo empresarial de forma ágil y rápida.
Facilidad de uso a lo largo de todas las etapas del ciclo de vida del software.
NetBeans Swing GUI builder: simplifica mucho la creación de interfaces gráficos de usuarios en aplicaciones cliente y permite al usuario manejar diferentes aplicaciones sin salir del IDE.
NetBeans Profiler: Permite ver de forma inmediata ver cómo de eficiente trabajará un trozo de software para los usuarios finales.
El editor WSDL facilita a los programadores trabajar en servicios Web basados en XML.
El editor XML Schema Editor permite refinar aspectos de los documentos XML de la misma manera que el editor WSDL revisa los servicios Web.
Aseguramiento de la seguridad de los datos mediante el Sun Java System Acces Manager.
Soporte beta de UML que cubre actividades como las clases, el comportamiento, la interacción y las secuencias.
Soporte bidireccional, que permite sincronizar con rapidez los modelos de desarrollo con los cambios en el código conforme avanzamos por las etapas del ciclo de vida de la aplicación.
Etc.

Para saber más

Amplía las herramientas concretas que ofrece NetBeans para el desarrollo de aplicaciones multiplataforma.

Visita la web oficial:

Información herramientas concretas de NetBeans

Autoevaluación

¿En qué fases del desarrollo de software ayudan los entornos integrados de desarrollo?

- ○ En codificación, pruebas, documentación, explotación y mantenimiento.
- ○ En codificación y documentación.
- ○ En análisis y documentación.

Efectivamente. Ofrece funcionalidades concretas en todas esas fases.

No. También ayudan a la gestión de pruebas al software y mantenimiento.

Incorrecta. El análisis es una tarea donde hay que concretar todos los aspectos que queremos que la aplicación resuelva, y eso sólo puede hacerlo una persona (analista).

Solución

1. Opción correcta
2. Incorrecto
3. Incorrecto

8.- Uso básico de entornos de desarrollo.

Caso práctico

—En qué partes se divide el espacio principal del entorno? Vamos a echar un vistazo, —le comenta Juan a Antonio. (A Juan le gusta explicárselo a su compañero, ahora que va descubriendo las ventajas de los IDE).

En el sitio principal del entorno de desarrollo de NetBeans nos encontramos con la siguiente ventana, que aparece cuando seleccionamos archivo, nuevo proyecto, java:

Vemos que el espacio se divide en dos ventanas principales.

Ventana Izquierda: ventana de proyectos.
Aquí irá apareciendo la relación de proyectos, archivos, módulos o clases que vayamos abriendo durante la sesión.

Cada proyecto comprende una serie de archivos y bibliotecas que lo componen.

El principal archivo del proyecto Java es el llamado `Main.java`. Es el archivo por donde empieza la ejecución. No tiene porque llamarse Main pero es aconsejable para que cualquier persona que vea el proyecto sepa porque fichero (clase) empieza la ejecución.

Ventana derecha: espacio de escritura de los códigos de los proyectos.
Aquí aparece el esqueleto propio de un programa escrito en lenguaje Java.

Se ha añadido el código:

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

Y veremos su significado en las siguientes páginas. De momento, saber que para escribir cualquier código, hay que hacerlo en esta ventana.

BARRA DE HERRAMIENTAS: Desde aquí podremos acceder a todas las opciones del IDE.

8.1.- Edición de programas.

Caso práctico

—Vamos a hacer el primer ejemplo -—comenta Ana, entusiasmada—. Después de todo, no debemos perder de vista la finalidad de la herramienta, ESCRIBIR PROGRAMAS!

En este sencillo ejemplo se ve una modificación de las líneas de código en la ventana de codificación del archivo `Main.java` del proyecto **ejemplo** que acabamos de crear.

Las dos líneas que aparecen resaltadas se han escrito sobre la ventana y, tal y como significan en lenguaje Java, su ejecución implicará que sendos mensajes encerrados entre comillas y entre paréntesis saldrán impresos.

No hay que decir que la programación en Java no es objeto del presente módulo, pero puedes probar con algunos ejemplos en Java que tengas de otros módulos.

Mientras escribimos en el editor de textos nos percatamos de varias características de NetBeans que ya hemos señalado en páginas anteriores:

Autocompletado de código.
Coloración de comandos.
Subrayado en rojo cuando hay algún error y posibilidad de depuración y corrección de forma visual, mediante un pequeño icono que aparece a la izquierda de la línea defectuosa.

Al final del tema, en el "Anexo VII.- Ejemplo de edición de código" se explica, en detalle, como escribir código en NetBeans.

Debes conocer

El proceso de edición de un programa desde que arranca el entorno hasta que está libre de errores sintácticos.

En el siguiente documento tienes el código del ejemplo de arriba.

Pequeño ejemplo de edición de código

Este es otro ejemplo (pide dos números y te visualiza la suma de ambos números)

Ejemplo de la suma de dos números

8.2.- Generación de ejecutables.

Una vez tenemos el código plasmado en la ventana de comandos y libre de errores de sintaxis, los siguientes pasos son: compilación, depuración, ejecución.

Para ejecutar el programa sólo tienes que pulsar shift+F6. De esta forma se ejecutará el código que tengas delante.

Al ejecutar el ejemplo anterior, el resultado es:

Si a este ejemplo le añadimos la funcionalidad de JFrame, el resultado de la ejecución es:

Este ejemplo (hecho con JFrame) aparece en el siguiente documento, al que accederás siguiendo el siguiente enlace:

Pequeño ejemplo de ejecución de código

Al final del tema, en el "Anexo VIII.- Ejecución de un programa en NetBeans" se detalla como ejecutar un programa en NetBeans, de otra forma que no sea pulsando shift+F6

Autoevaluación

Los pasos que debemos dar para generar un ejecutable son:

- ○ Ejecución directa.
- ○ Ejecución, una vez que el editor esté libre de errores sintácticos.
- ○ Una vez que el editor esté libre de errores, compilar, depurar y ejecutar.

Incorrecto. La ejecución directa nunca es recomendable (como mucho, sólo en programas muy pequeños).

No es del todo correcto. Además de los errores sintácticos, existen otro tipo de errores que debemos detectar antes de proceder a la ejecución del programa.

Así es. Estos serían los pasos y la secuencia correcta de actuación para generar ejecutables.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta

9.- Actualización y mantenimiento de entornos de desarrollo.

Caso práctico

—Por último, es de vital importancia el mantener y actualizar el entorno de desarrollo —comenta Ana—. Deberíamos tener permanentemente actualizados todos los complementos y realizar un correcto mantenimiento a las bases de datos asociadas a nuestros proyectos.

El mantenimiento del entorno de desarrollo es una tarea fundamental que requiere tener todos sus componentes periódicamente actualizados.

También es de vital importancia realizar copias de seguridad sobre las bases de datos de nuestros proyectos por si ocurriera algún error o proceso defectuoso poder restaurarlos.

El mantenimiento y las actualizaciones se hacen de forma on-line. En NetBeans contamos con el complemento llamado Auto Update Services. Lo podemos encontrar en el siguiente enlace:

Complementos de Netbeans

Una vez instalado, nos permitirá realizar continuas revisiones del entorno y actualizaciones de todos los plugins.

Para añadir módulos y plugins on-line, hay que tener este complemento instalado en el entorno.

La gestión de las bases de datos asociadas a nuestros proyectos es muy importante. Habrá que realizarles copias de seguridad periódicamente, para asegurar su restauración en caso de fallos en el sistema, y mantenerlas actualizadas para su posible portabilidad futura a nuevas versiones del entorno que utilicemos.

Autoevaluación

¿Cuál es la razón, en tu opinión, de que salgan nuevas versiones de los entornos de desarrollo tan rápidamente?

- ○ Para adaptarse a la evolución del hardware.
- ○ Para incluir y modificar funcionalidades del entorno.

Incorrecto. Esto no es significativo en la evolución de los entornos de desarrollo.

Así es. Cada nueva versión tiene mejoras que permite aumentar la funcionalidad del entorno.

Solución

1. Incorrecto
2. Opción correcta

Anexo I.- Crear, compilar y ejecutar un programa (sin el uso de un entorno de desarrollo).

Vamos a **crear, compilar y ejecutar** nuestro primer programa en java.

Vamos a hacerlo usando un editor de textos como es el bloc de notas y el JDK. No vamos a usar ningún entorno de desarrollo.

Para ello será necesario haber configurado correctamente las variables de entorno JAVA_HOME y PATH como se indica en los contenidos de la unidad.

Paso 1: Utilizando un editor de textos crear el fichero **Hola.java** con el siguiente código:

```
public class Hola {
            public static void main(String[ ] args) {
                        System.out.println("Hola");
            }
}
```

Es muy importante que el nombre del fichero sea **Hola.java** (respeta mayúsculas y minúsculas)

Salva el archivo en alguna ruta de tu sistema de archivos.

Paso 2: Deberás abrir un terminal. Más adelante, se ofrece ayuda en este paso

Paso 3: Una vez te encuentres en el directorio/carpeta donde está el fichero Hola.java hay que ejecutar el comando:

```
javac Hola.java
```

De esta forma compila el fichero, generando el fichero "Hola.class."

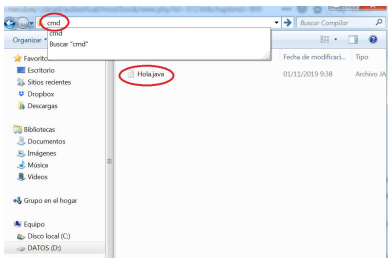
Paso 4: Una vez que tenemos el fichero "Hola.class" hay que interpretarlo con el comando "**`java`**". *Hay que poner esta instrucción:*

```
java Hola.
```

Resultado de la compilación y ejecución (ejemplo en Linux).

Ayuda para el paso 2:
En Windows:

Lo haremos desde la ubicación del fichero Hola.java para tener un acceso más sencillo a él desde el terminal. Abrimos el explorador y nos situamos en la ubicación del fichero. En la barra de direcciones escribimos cmd y pulsamos ENTER.



Se abre el terminal.

También se puede acceder al terminal desde Inicio, tecleando cmd y ENTER en el campo de texto. Pero de este modo tendrás que usar el comando cd para situarte en el directorio/carpeta donde se encuentra el fichero Hola.java

En Linux

https://www.comoinstalarlinux.com/como-abrir-una-terminal-en-ubuntu-linux-mint-centos-debian/

En MAC OS

https://www.soydemac.com/abrir-terminal-mac/

NOTA:

Para que el paso 3 resulte más sencillo puedes abrir el terminal en el directorio/carpeta donde se encuentre el fichero Hola.java. Si no es así, deberás usar el comando cd en el terminal para situarte en esa ruta. En las siguientes URLs puedes ver ejemplos de uso de ese comando para que puedas situarte en el directorio/carpeta correcto.

En Windows: https://www.abrirllave.com/cmd/comando-cd.php.

En Linux: https://www.solvetic.com/tutoriales/article/7190-como-usar-el-comando-cd-en-linux/ .

En MAC OS: https://www.faq-mac.com/2003/06/guia-del-terminal-unix-command-line-para-usuarios-mac-parte-iii/

Con esta práctica habrás visto que no hemos usado el IDE "NetBeans" en ningún momento. Solamente hemos usado un editor de textos (como puede ser el "bloc de notas") y los programas "javac" y "java".

JavaC es un compilador que genera un fichero con el mismo nombre del fichero original pero con extensión class. Traduce el fichero escrito en Java y lo pasa a bytecode.

java es un intérprete que traduce los bytecode a código máquina, para que el ordenador lo entienda y lo pueda ejecutar. Como intérprete que es, va traduciendo línea a línea y ejecutando. No genera ningún fichero nuevo (los interpretes no generan ficheros).

Anexo II.- Crear, compilar y ejecutar un programa con varias clases (sin el uso de un entorno de desarrollo).

Esta práctica nos servirá de apoyo para poder entender, a través de un ejemplo, algunos de los conceptos que hemos estado viendo hasta el momento, tales como:

- Lenguaje de alto nivel. Un acercamiento al lenguaje Java.
- Código fuente y código máquina.
- Traductor. En este caso se trata del compilador de Java.
- Código interpretable o bytecode. Intérprete.
- Conceptos asociados con la programación orientada a objetos.

No forma parte de este ejercicio, entender el código **Java** en su totalidad. Dichos contenidos serán tratados en otros módulos del ciclo.

Esta práctica, igual que hemos hecho en la anterior, se va hacer usando un editor de textos (como, por ejemplo, el bloc de notas) y el JDK. No se va a usar ningún entorno de desarrollo.

Ejercicio Propuesto

Un profesor de autoescuela pretende enseñar a un estudiante la combinación de colores entre los que puede ir cambiando un semáforo.

En el aula existe un ordenador que simula el comportamiento del semáforo, al que podemos consultar el color que tiene en cada momento.

En la secuencia de ejecución, el profesor preguntará al estudiante el color que tiene el semáforo. Para responder, el estudiante obtiene el resultado del ordenador.

Cuando damos solución a un problema propuesto mediante un programa, éste puede ser implementado de muy diversas formas. Cualquiera de ellas será igualmente válida siempre y cuando cumpla los requisitos que se le piden (funcionales, de rendimiento, disponibilidad,...).

A continuación se propone una de las posibles soluciones. Pulsa aquí para descargar el fichero **semaforo.zip** que incluye el código fuente.

Mostrar retroalimentación

Identificando actores / objetos involucrados

De acuerdo a lo solicitado en el enunciado, es posible plantear el problema haciendo uso de tres actores (profesor, estudiante, ordenador). Para cada uno de ellos se desarrollará una clase **Java**.

Se crearán cinco ficheros de código fuente, que una vez compilados nos proporcionarán otros tantos ficheros en código intermedio (**bytecode de Java**).

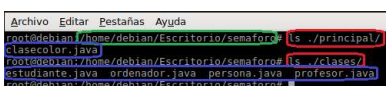
Para la ejecución del programa, el **intérprete de Java** hará uso de los ficheros **bytecode** creados, junto a otras funciones pertenecientes a las bibliotecas de **Java**.

Cuando queremos implementar una solución, se crea un "proyecto", cuyos contenidos quedan almacenados en el sistema de archivos del ordenador en un directorio (semaforo en nuestro caso).

A partir de aquí, se pueden ver subdirectorios, que organizan los diferentes archivos del proyecto en paquetes (paquetes clases y principal para este ejemplo).

Por último, cada paquete puede contener un conjunto de ficheros, donde se implementan las clases. Podrá ser una o varias por fichero (típicamente fichero .java implementa una clase).

Ejemplo de visualización de ficheros en Linux:



Descripción de las clases

Clase ClaseColor (ClaseColor.java)

```
1  package principal;
2  import clases.Profesor;
3  // Clase color, el profesor pregunta a un alumno por un color entre (rojo, amarillo y verde)
4  public class ClaseColor {
5                public static void main(String[] args) {
6                              Profesor teacher = new Profesor();
7                              String color = teacher.preguntacolor();
8                              System.out.println("La respuesta recibida es:" + color);
9                }
10 }
```

Consideraciones de la clase ClaseColor:

- La clase ClaseColor está contenida en el fichero ClaseColor.java y se incluye en un paquete llamado principal (1).
- Como parte del código de la clase, se va a utilizar la clase Profesor que forma parte del paquete clases (2).
- El inicio del programa se lleva a cabo en esta clase, puesto que incluye la función main (5).
- De la clase Profesor se crea un objeto llamado teacher (6).
- Desde la clase ClaseColor se envía un evento a la clase Profesor (a través del objeto teacher), para que esta última ejecute el método preguntacolor (7).
- También utilizamos funciones proporcionadas en las librerías de Java, como println .

Clase Ordenador (Ordenador.java)

```
1  package clases;
2  import java.util.Random;
3  public class Ordenador {
4    public Ordenador() {}
5    public String color(){
6                Random randomGenerator = new Random();
7                int randomInt = randomGenerator.nextInt(3);
8                if(randomInt == 0)
9                {  return "rojo";}
10               else if(randomInt == 1)
11               { return "amarillo";}
12               else
13               { return "verde";}
14   }
15 }
```

Consideraciones de la clase ordenador:

- La clase Ordenador utiliza el método Random para obtener un número aleatorio entre 0 y 2.
- El método color devuelve una de las siguientes cadenas de caracteres (“rojo”, “amarillo”, “verde”), instrucción return (9, 11, 13).

Clase Persona (Persona.java)

```
1  package clases;
2  // Clase utilizada para ser herencia de estudiante y profesor
3  public class Persona {
4                // Métodos de clase. Edad y nombre
5                int i_Edad;
6                String s_Nombre;
7   }
```

Consideraciones de la clase persona:

- La clase Persona no va a ser instanciada directamente en el programa (crear un objeto de la clase). Se trata de una clase padre, que va a ser utilizada por las clases estudiante y profesor para heredar sus características.
- En esta clase se definen las variables de clase int i_Edad y s_Nombre (5, 6).

Clase Estudiante (Estudiante.java)

```
1  package clases;
2  public class Estudiante extends Persona{
3                  // Incluye un método de clase que se une a los heredados
4                  int i_Curso;
5                  public Estudiante() {
6                                  i_Edad=25;
7                                  s_Nombre = "Luis";
8                                  i_Curso = 1;
9                  }
10                 public void presentarse(){
11                 System.out.println("Soy " + s_Nombre +  " Alumno de " + Integer.toString(i_Curso) + "
12                 }
13                 public String preguntacolor(){
14                                 presentarse();
15                                 Ordenador mipc = new Ordenador();
16                 return mipc.color();
17                 }
18 }
```

Consideraciones de la clase Estudiante:

- Esta clase hereda de la clase Persona, claúsula extends (2).
- En la clase estudiante se pueden utilizar las propiedades y métodos definidos en la clase padre (6, 7, 11).
- Además puede incluir otros métodos y propiedades propias (4).
- Si hubiéramos sobrescrito una de las propiedades o métodos del padre, en la propia clase, ésta utilizaría sus propios métodos y propiedades en lugar de los heredados.

Clase Profesor (Profesor.java)

```
1  package clases;
2
3  public class Profesor  extends Persona{
4                  Public Profesor() {}                                          // Constructor
5
6                  // Hace la pregunta al estudiante sobre el color
7                  Public String preguntacolor(){
8                          Estudiante alumno = new Estudiante();
9                          String colorRec = alumno.preguntacolor();;
10                          return colorRec;
11                 }
12 }
```

Uso del traductor/compilador de Java

Como se indicaba anteriormente, cuando se crea un nuevo proyecto **Java** todos sus recursos “cuelgan” de un determinado directorio en el sistema de archivos ("/home/debian/Escritorio/semaforo" en este ejemplo). Puedes elegir el que quieras en tu equipo.

A partir de aquí, podemos ir distribuyendo los ficheros en diferentes paquetes, que corresponderán a subdirectorios del directorio proyecto ("/home/debian/Escritorio/semaforo"). Así hemos incluido los ficheros Ordenador.java, Persona.java, Profesor.java y Estudiante.java en el paquete clases. Además de distribuir los paquetes en el directorio correspondiente, es necesario indicarlo en el código (ver **instrucción package**).

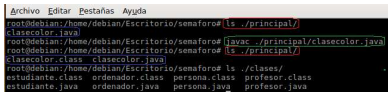
Para utilizar cada una de estas clases en otros ficheros .java deberemos importarlos (**instrucción import**). Si dos ficheros forman parte del mismo paquete, no es necesario que sean importados.

El fichero de inicio del programa ClaseColor.java (incluye la **función main**), y forma parte del paquete principal.

Para obtener el código **bytecode** del fichero clasecolor.java:

- Acceder al directorio de inicio del proyecto desde un terminal.
- Llamar al compilador (traductor) de Java "javac" indicando la ruta relativa del fichero que queremos compilar. En nuestro caso “javac principal/ClaseColor.java”.
- Como resultado obtenemos el fichero interpretable por Java ClaseColor.class.

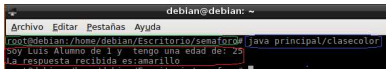
Puedes ver el proceso en la siguiente imagen (ejemplo en Linux).



Para que el compilador funcione tal y como se ha descrito en este punto, será necesario que se hayan metido las variables de entorno tal y como se ha indicado la práctica anterior.

Nota: la compilación del fichero principal "javac principal/ClaseColor.java" puede ya compilar todos los ficheros del proyecto al ser unos dependientes de otros. En caso contrario, compilaremos el resto uno a uno en el siguiente orden: Persona.java, Ordenador.java, Estudiante.java, Profesor.java y ClaseColor.java.

Una vez disponibles todos los ficheros **bytecode**, procedemos a la ejecución del programa desde un terminal, vamos al directorio de inicio del proyecto y ejecutamos el intérprete java: "java principal/ClaseColor".



Empaquetar .class en fichero .jar

Java nos permite empaquetar todos los ficheros (.class, junto a otros recursos utilizados) del proyecto en uno único con extensión **.jar (Java ARchives)**. Además el formato del fichero .jar es comprimido, por lo que ocupa menos espacio que la información original.

A continuación se indica el procedimiento para llevarlo a cabo.

Crear el fichero MANIFEST.MF

Crear un directorio **META-INF** (¡las mayúsculas son importantes!) y dentro un fichero **MANIFEST.MF**. Este fichero indica, entre otras cosas, cuál será la clase principal. A menudo el fichero **MANIFEST.MF** contiene una única línea:

Main-Class: principal.ClaseColor

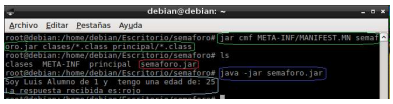
donde se indica que ClaseColor.class (del paquete principal) es la clase que contiene el método main.
Nota : el fichero **MANIFEST.MF** se puede crear con cualquier editor de texto. Hay que introducir un fin de línea (enter) tras la última línea con texto.

Crear el fichero semaforo.jar

El paquete **JDK** proporciona una serie de ejecutables. Ya se ha trabajado con algunos de ellos como el compilador de java (javac) y con el intérprete (java) en prácticas anteriores. Otro de los programas a nuestra disposición es **jar**, que permite empatequetar todos los recursos de un proyectos **Java** en un fichero con extensión .jar.

Desde consola del sistema, ejecutar el programa jar con los siguientes parámetros:

jar cmf META-INF/MANIFEST.MF semaforo.jar clases/.class principal/*.class*

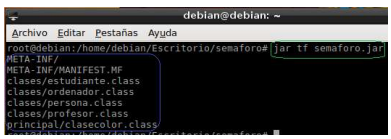


Observa en la imagen una ejecución del proyecto utilizando el fichero .jar obtenido:

java –jar semaforo.jar

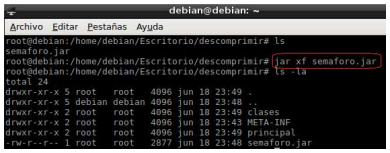
*El programa **jar** también nos permite ver el contenido de proyecto comprimidos .jar:*

jar tf semaforo.jar



Si deseamos descomprimir la información del fichero .jar, para obtener sus fichero originales:

jar xf semaforo.jar



Script Compilar.sh

Para facilitar la creación de nuestros ficheros interpretables, es útil crear un **script** que realice todo el proceso de una sola vez. En nuestro proyecto podría ser **compilar.sh**, al que se deberá dar permisos de ejecución.

En el módulo de sistemas informáticos se estudia como crear scripts. Se puede dejar este ejercicio para cuando se haya visto como crear scripts en dicho módulo.

Ejemplo en Linux:

Anexo III.- Instalación JDK y NetBeans en Windows

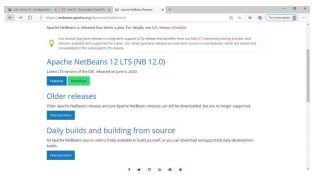
PASOS

Descargar NetBeans de la siguiente URL:

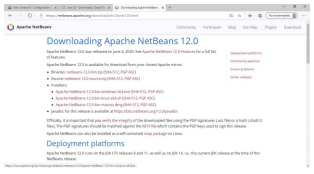
Descargar Netbeans.

Aparecerá la siguiente ventana:

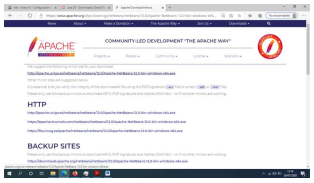
Ves a descarga y pulsa e “última versión”. Te saldrá esta ventana.



Pulsa en “DownLoad”



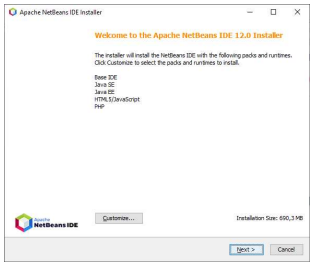
Elige la opción del instalador para Windows. Elige el primer enlace.



Se descargará un fichero con este nombre: Apache-NetBeans-12.0-bin-windows-x64.exe.

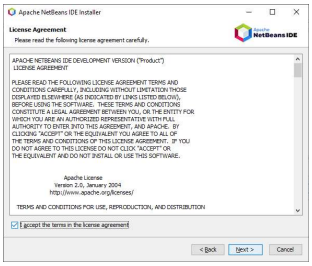
Instálelo.

Saldrá esta ventana.



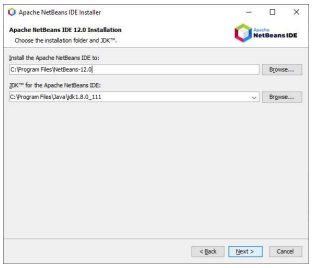
Pulsa “Next”.

Saldrá esta ventana.



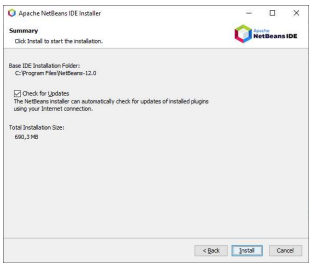
Acepta la licencia y pulsa “Next”.

Te informa que se va instalar NetBeans y el JDK.

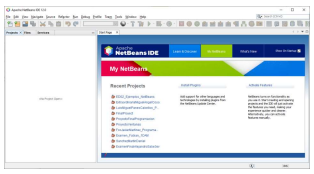


Pulsa “Next”.

Y, por último, pulsa “Install”.



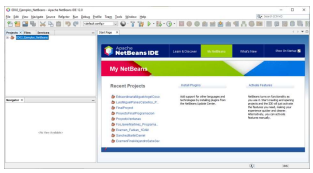
Una vez instalado , ejecútalo y te visualizará esta ventana:



Donde podrás empezar a desarrollar software.

Anexo IV.- Configuración y personalización de NetBeans.

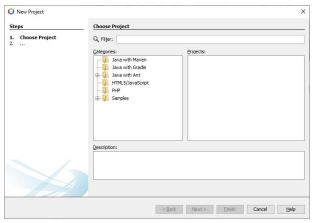
Accedemos a NetBeans y entramos en la página principal de la aplicación.



A la izquierda muestra los proyectos que tengas abiertos.

Para crear un nuevo proyecto, selecciona File/New Project.

Te visualizará esta ventana:



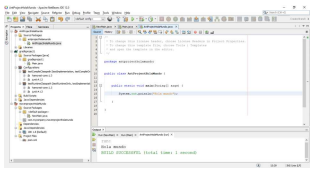
A la hora de crear un proyecto en Java existen tres opciones:

1. **Java with Maven**
2. **Java with Gradle**
3. **Java with Ant**

La diferencia entre ellos lo puedes ver en estos enlaces:

1. **Java with Maven:** https://es.wikipedia.org/wiki/Maven
2. **Java with Gradle:** https://es.wikipedia.org/wiki/Gradle
3. **Java with Ant:** https://es.wikipedia.org/wiki/Apache_Ant

Aquí se muestra el típico programa que se hace siempre cuando se empieza a manejar un lenguaje y es un simple programa que visualiza “Hola Mundo”. Se muestra a continuación un ejemplo de las tres formas para ver la diferencia entre ellos.

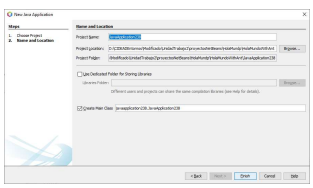


Principalmente mira todo el código que genera según la opción que elijas.

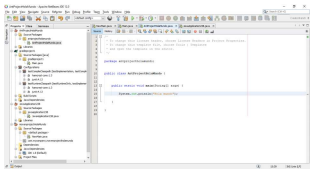
Nosotros, como no estamos iniciando en la programación en Java vamos a trabajar siempre con la opción: Java with Ant.

En esa opción cogéis “Java Application”.

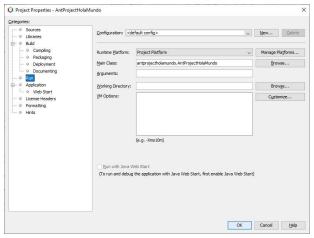
Os pide que indiquéis donde queréis guardar el proyecto y con qué nombre:



Seguidamente os visualizará esta ventana para que empecéis a programar:



Una vez que tienes el proyecto, sobre el nombre del proyecto pulsa el botón derecho del ratón y elige la opción “Set configuration” y ahí la opción “Customize”:



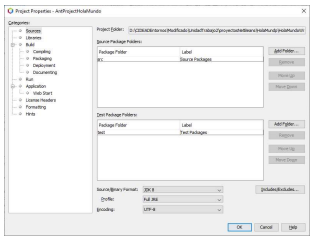
Aquí vemos todo lo que podemos personalizar de la aplicación:

- Fuentes.
- Bibliotecas.
- Generación de código.
- Ejecución de código.
- Opciones de la aplicación.
- Formato del código en el editor de textos.

FUENTES

Podemos modificar:

- La carpeta que contendrá el proyecto.
- La carpeta que almacenará los paquetes fuentes.
- La carpeta que contendrá los paquetes prueba.
- La versión del JDK con la que queremos trabajar.

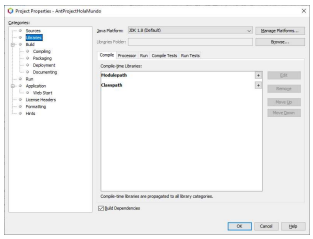


BIBLIOTECAS

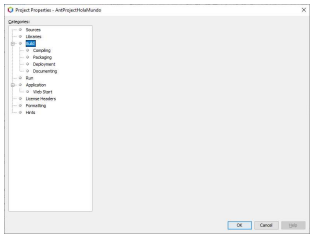
Desde esta ventana podemos elegir la plataforma de la aplicación.

Toma por defecto el JDK, pero se puede cambiar si se quiere, siempre y cuando sea compatible con la versión de NetBeans utilizada.

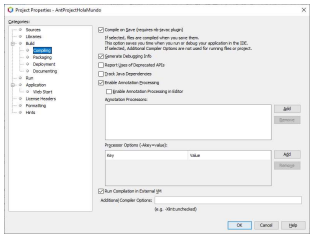
También en esta ventana se puede configurar el paquete de pruebas que se realizará al proyecto.



GENERACIÓN DE CÓDIGO



GENERACIÓN DE CÓDIGO - COMPILANDO

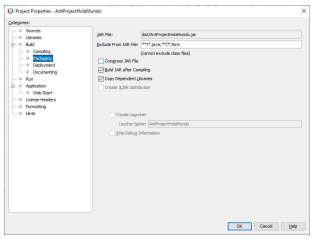


Las opciones que nos permite modificar en cuanto a la compilación del programa son:

- Compilar al guardar: al guardar un archivo se compilará automáticamente.
- Generar información de depuración: para obtener la documentación asociada.
- Enable annotation processsing: permitir anotaciones durante el proceso.

También podemos agregar anotaciones concretas para el proceso de compilación y añadir opciones de proceso que, según las características del proyecto, puedan ser de interés para nosotros.

GENERACIÓN DE CÓDIGO - EMPAQUETANDO

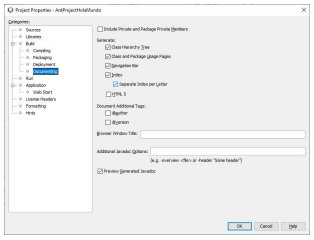


Las aplicaciones resultado de la compilación del código deben ser empaquetadas antes de su distribución, con objeto de tener un único archivo, generalmente comprimido, que contenga en su interior todos los archivos de instalación y configuración necesarios para que la aplicación pueda ser instalada y desarrollada con éxito por el usuario cliente.

Como vemos en la imagen, en esta opción podemos modificar el lugar donde se generará el archivo resultante del empaquetado, así como si deseamos comprimirlo.

También podemos elegir que el archivo empaquetado se construya tras la compilación, que es lo habitual (por eso esta opción aparece como predeterminada).

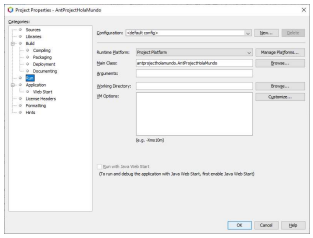
GENERACIÓN DE CÓDIGO – DOCUMENTANDO



Como ya vimos en la unidad anterior, la documentación de aplicaciones es un aspecto clave que no debemos descuidar nunca. NeatBeans nos ofrece una ventaja muy considerable al permitirnos obtener documentación de la fase de codificación de los programas de forma automática.

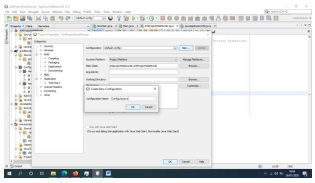
Dentro del documento que se va a generar podemos elegir que se incluyan todas las opciones anteriores. Esto es lo más recomendable, por eso aparecen todas marcadas de forma predeterminada y lo mejor es dejarlo como está.

EJECUTANDO CÓDIGO



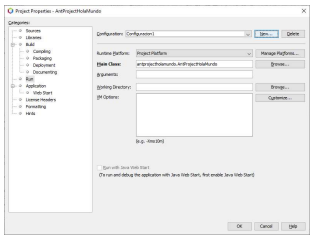
Esta opción nos permite definir una nueva configuración de ejecución de código, elegir la clase principal, las carpetas de trabajo del proyecto y opciones de la máquina virtual.

En la ventana de “Configurar el nombre” escribimos el nombre que tendrá nuestra configuración personalizada.



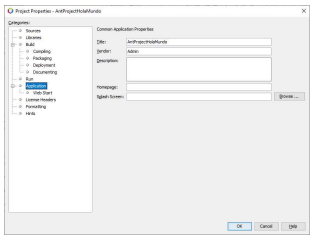
En este caso, escribimos “configuración1” y pulsamos “aceptar”.

A partir de este momento, todas las opciones de configuración que seleccionemos se guardarán en “configuración1”.



Ahora podemos elegir la aplicación sobre la cual queremos aplicar la configuración personalizada de “configuración 1”.

OPCIONES DE LA APLICACIÓN

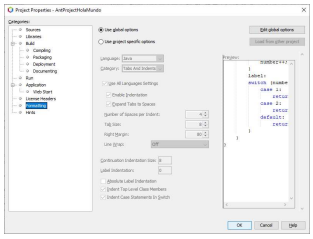


Como vemos, podemos dar una descripción al proyecto, cambiarle el nombre…

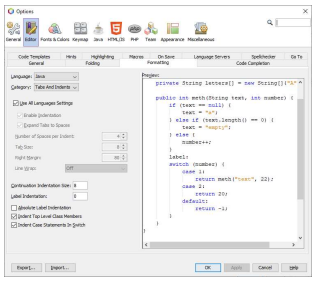
Es conveniente hacerlo, ya que el nombre de los nuevos proyectos se generar automáticamente por NetBeans al inicio de la sesión.

FORMATO

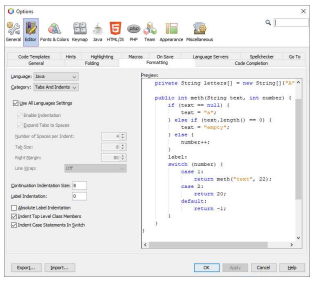
Aquí podemos personalizar aspectos globales del formato del código fuente en la aplicación.
Podemos personalizar las opciones sólo para el proyecto actual o bien para todos los proyectos que estén basados en NetBeans a partir de ahora (utilizar opciones globales).



Si seleccionamos “Edit global options” (en opciones globales) nos encontramos con la siguiente ventana, que tiene una barra superior de pestañas para configurar cada apartado del formato de forma independiente.

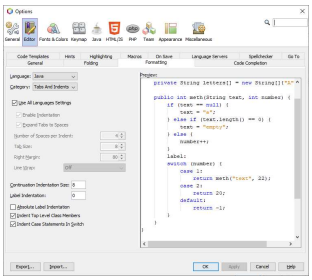


Opción “Editor”:

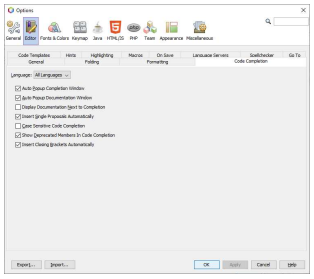


Pestaña Formatting:

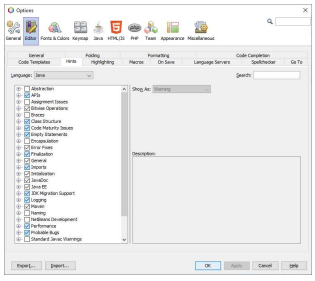
Se puede configurar los tamaños de los espaciados, pestañas…



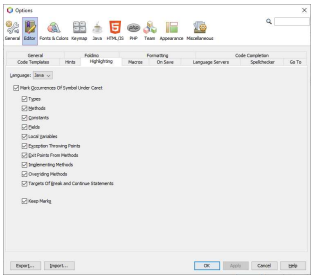
En la pestaña de “Code Completion” podremos cambiar:



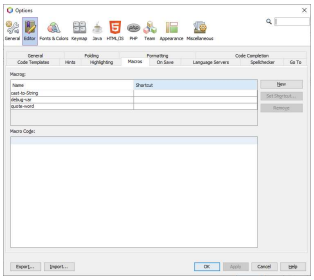
En la pestaña de “Hints”:



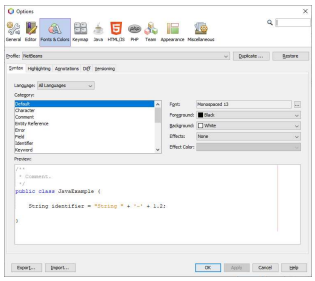
En la pestaña de "HighLighting" (Marcar ocurrencias):



En la pestaña de macros:

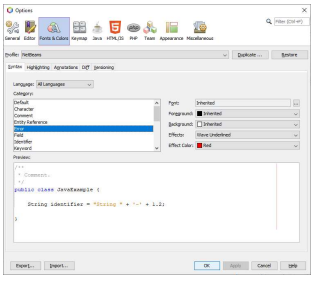


Opción "Fonts & Colors":



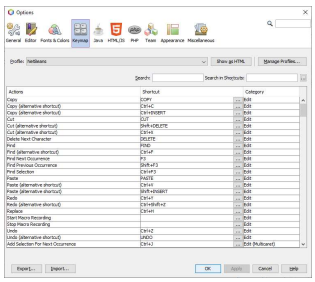
Consiste en elegir el tipo de letra y colores que prefiramos para el texto del código, así como efectos (si es que así lo deseamos).

También podemos configurar el tipo de letra y color de los errores del código (por defecto, de color rojo).



Y lo mismo con los números, espacios en blanco…

Opción "KeyMap":



En cuanto a los métodos abreviados de teclado (combinación de teclas equivalente a las acciones en NetBeans), podemos modificar aquellas acciones que hagamos con más frecuencia por aquella combinación de teclas que nos sea más fácil recordar.

Anexo V.- Adición de módulos.

Hay muchos programas que se le pueden añadir más funcionalidades instalándoles unos módulos conocidos como plugins.

En este anexo vamos a ver como añadir dichos plugins en NetBeans y en Eclipse.

a.- Adición de módulos en NetBeans.

Hay dos formas de añadir módulos y plugins en NetBeans:

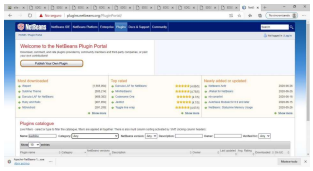
1.- Off-line: Buscar y descargar plugins desde la página web oficial de la plataforma

Descarga de plugins para NetBeans.

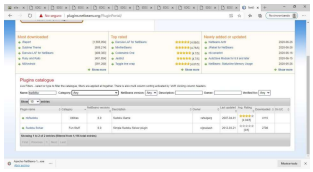
Ejemplo:

Vamos a buscar un plugin para jugar al sudoku desde nuestro IDE. No es muy educativo, pero sirva como ejemplo la manera en que se va a realizar el proceso (será igual en todos los casos):

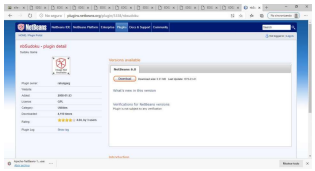
Entramos en la zona de descargas de plugins para NetBeans :



y en la zona del catálogo, escribiremos la palabra sudoku:

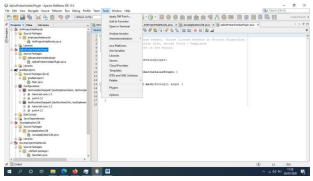


Se nos abre una ventana con las características del plugin y la opción de descargarlo.

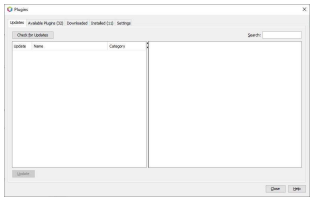


Damos a la opción de descargar

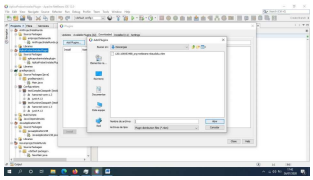
Entramos en NetBeans y pinchamos en Tools:



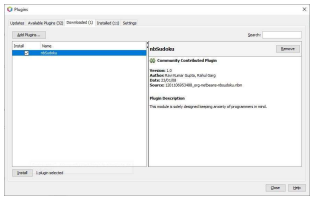
Elegimos la opción Plugins.



En la pestaña "downloaded" (descargado) seleccionamos "Add Plugins" (Agregar Plugins)



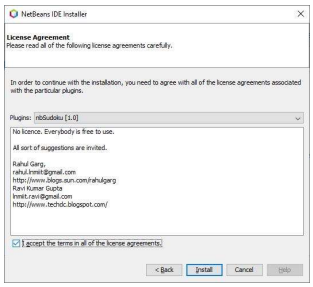
Seleccionamos la carpeta donde habíamos guardado el plugin del sudoku y le damos a "aceptar"



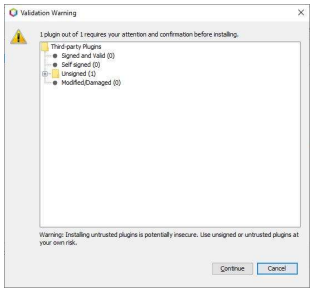
Estando el plugin seleccionado, pulsamos "Install".

Empieza la instalación:

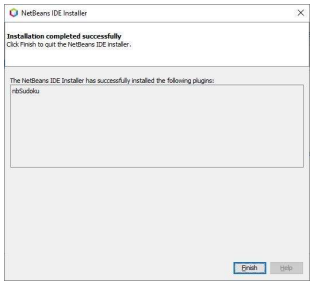
Pulsamos siguiente. Después, aceptamos la licencia:



Pulsamos "instalar"

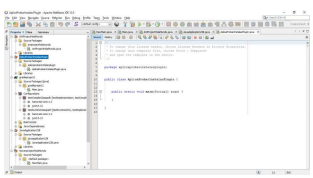


Nos pide una confirmación y se terminará por instalar el plugin.



Seleccionamos "Finish"

Observamos el icono que aparece en la barra de iconos superior del sitio:



Si lo pulsamos, ya podemos jugar un ratito al sudoku para despejarnos:

2.- On-Line: Instalarlos desde el propio entorno de desarrollo:

Se pueden instalar plugins desde NetBeans.

Para ello, vamos a Tools/Plugins. Y pinchamos sobre la pestaña: "Available plugins". Ahí podemos elegir alguno de los plugins que hay e instalarlo.

b.- Adición de módulos en Eclipse.

Introducción

Eclipse emplea módulos (**plug-in**) para añadir funcionalidades a las ya existentes, se trata de "un esqueleto" capaz de integrar diferentes paquetes que proporcionarán los servicios que en cada caso el proyecto a desarrollar demande. Por tanto, **Eclipse** se aleja de los diseños monolíticos donde todo queda instalado desde el inicio sea o no necesario.

Existen en la actualidad multitud de módulos disponibles , en este documento veremos dónde encontrar módulos para **Eclipse** que sean de algún interés para nosotros y las distintas formas de instalarlos/desinstalarlos.

El primer lugar a visitar es el "mercado de plugins" de **Eclipse:** https://marketplace.eclipse.org/

Instalación de plugins

Existen varias alternativas para instalar un **plugin** en **Eclipse**. Vamos a ver algunas formar de como hacerlo.

Para verlo vamos a instalar tres plugins: Enhanced class decompiler,JBC y UMLet.

Enhanced class decompiler

El **plugin Enhanced class decompiler** ofrece la posibilidad de descompilar ficheros .class (**bytecode**) para ver su código fuente.

Se encuentra disponible en el **Marketplace de Eclipse** y procederemos a su instalación desde la opción de menú "Ayuda/EclipseMarketPlace del IDE".

Los pasos a seguir son:

- Indicar el nombre del plugin a instalar y pulsar "Search".
- Una vez localizado, pulsar Install.
- La instalación es guiada y no presenta dificultades. Habrá que aceptar las licencias de instalación y reiniciar Eclipse.

Una vez reiniciado, habrá que seleccionar este **plugin** para mostrar el código descompilado de ficheros con extensión .class.

Para ello aplicaremos la siguiente configuración:

- Mostrar la ventana de preferencias desde la opción de menú "Ventana/Preferencias".
- Seleccionar "General/Editores/Asociaciones de Archivo/*.class without source".
- Marcar el módulo "Class Decompiler Viewer" y pulsar el boton "Por defecto". Finalizar el proceso aplicando los cambios.

Para comprobar su funcionamiento es necesario acceder a un fichero .class (se obtendrá cuando se compile el código) y abrirlo con el "Class Decompiler Viewer". Cuando compiles tu primer programa obtendrás un fichero .class y podrás

probarlo.

Para desinstalar el **plugin**, acudir nuevamente a la opción de menú "Ayuda/EclipseMarketPlace" del **IDE** y seleccionar la pestaña Installed donde aparecen los módulos instalados en **Eclipse**. Aquí tendremos la posibilidad de solicitar su cambio-actualización o desinstalación.

JBC

BC (Java ByteCode) permite ver la información compilada de un fichero .class desde el **IDE**.

La instalación se hará nuevamente desde el **MarketPlace de Eclipse**, a través de la opción de menú "Ayuda/EclipseMarketPlace".

Una vez localizado, pulsar el botón Install, aceptar los términos de la licencia, proceder con la instalación por defecto y finalmente reiniciar **Eclipse**.

Para comprobar su funcionamiento, pulsar con el botón derecho del ratón sobre un fichero con extensión .java en el "Explorador de paquetes" (vista disponible en la perspectiva Java). Seleccionar la opción **Open JBC** del menú contextual.

Para desinstalar el módulo, acudir nuevamente a la opción de menú "Ayuda/EclipseMarketPlace", pestaña Installed, botón Uninstall.

UMLet

Durante el curso conoceremos una serie de diagramas de modelado útiles en las fases de análisis y diseño en los desarrollos software. Existe muchas herramientas que nos permite dibujar estos diagramas, una de ellas es **UMLet**. UMLet está disponible tanto en formato **standalone**, como para ser integrada en otras aplicaciones. Existe un **plugin** de **UMLet** para **Eclipse**.

En esta ocasión utilizaremos una técnica distinta para añadir el plugin.

- Descargar de la url http://www.umlet.com/changes.htm el **plugin UMLet** para **Eclipse**.
- Descomprimir y copiar el fichero .jar en la ruta **plugins o dropings de Eclipse**.
- Para iniciar **UMLet**, crea un nuevo proyecto desde la opción de menú "Archivo/Nuevo/Otras/Otro/Umlet diagram".

Anexo VI.- Eliminar módulos en NetBeans.

Vamos a ver la secuencia de pasos a seguir para eliminar el plugin del juego del sudoku del entorno.

El proceso es muy sencillo: basta con conseguir la lista de complementos instalados (Tools - Plugins). Vamos a la pestaña ("Installed") y localizamos el complemento que queremos eliminar escribiendo su nombre en el lugar destinado para ello y seleccionamos una de entre las dos opciones posibles: desinstalarlo o desactivarlo

En la pestaña de complementos instalados, escribimos el nombre del plugin (sudoku) en la barra de búsqueda:

Cuando lo encuentra, en la ventana aparecen las dos posibilidades de eliminación (desactivarlo o desinstalarlo).

En este caso, hemos optado por desactivarlo.

Anexo VII.- Ejemplo de edición de código.

En este documento vamos a introducirnos en la edición de programas en NetBeans a través de un ejemplo sencillo de una aplicación de Java.

Lo primero es iniciar la plataforma:

Seleccionamos File- New Project.

Elegimos "Java With Ant" /"Java Application"

Pulsamos sobre siguiente. En la siguiente ventana te pide un nombre (para el proyecto) y su ubicación.

Lo vamos a llamar Ejemplo y ubícalo donde creas conveniente.

Una vez iniciado el proyecto, en la ventana de proyectos (izquierda) vemos cómo se ha cargado el proyecto "Ejemplo". Lo seleccionamos con el ratón y se despliega, mostrando todos sus archivos componentes. Seleccionamos Ejemplo.java (que es el archivo por donde va arrancar el proyecto). Dicho fichero le vamos a editar:

.

En la ventana de edición (a la derecha) nos aparece el esqueleto de la estructura básica de una aplicación en Java.

Lo que vamos a hacer a lo largo del ejemplo es añadir código.

La primera línea de código que vamos a agregar es una orden sencilla en Java, cuya ejecución posterior dará lugar a la aparición de un mensaje por pantalla.

Añadimos otra línea más con otro mensaje "Creando mi primer ejemplo"

Ahora vamos a modificar la parte de arriba del programa. Añadimos la siguiente línea:

Esta línea nos va a servir para adentrarnos en una de las utilidades más importantes de un entorno de desarrollos.

NetBeans entiende esta orden como un error (aparece subrayada en una línea roja ondulada y con un pequeño icono al lado izquierdo)

Si pulsamos sobre ese icono con el ratón, NetBeans nos aporta sugerencias para deshacer el error:

En este caso, elegimos importar JFrame a la librería.

Y seguimos añadiendo código en el editor:

Llegados a este punto, ya hemos comprobado que el editor no nos da ningún problema más. En el siguiente punto del tema, veremos cómo ejecutar esto.

Vemos también cómo se han importando con éxito las librerías que nos han hecho falta:

El código completo del ejemplo es el siguiente:

```
package ejemplo;

import javax.swing.JFrame;
import javax.swing.JLabel;

public class Ejemplo extends JFrame{

public Ejemplo()
{
JLabel lblSaludo = new JLabel( "Hola Mundo. Creando mi primer ejemplo");
add(lblSaludo);
this.setSize(600,200);
this.setTitle("JFrame");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);

}
public static void main(String[] args) {
new Ejemplo();
}

}
```

Anexo VIII.- Ejecución de un programa en NetBeans.

Continuando con el ejemplo anterior, recuerda que habíamos llegado a este punto:

Tenemos el programa escrito en el editor libre de errores sintácticos.

¿Cómo convertir ese programa en ejecutable?

Cabe destacar que, por la sencillez y pequeñez del programa, la ejecución del mismo podría ser directa sin ningún problema.

Sin embargo, debemos acostumbrarnos a seguir los pasos adecuados, que son: Editor libre de errores → Compilación → Depuración → Ejecución

Para ejecutar el programa se puede hacer de varias formas:

- Sobre el propio fichero, pulsar el botón derecho del ratón y elegir la opción "Run File".

- O pinchar en el menú "Run" y coger la opción "Run project". Si se hace de esta forma, tiene que estar bien configurado el proyecto para indicarle que empiece la ejecución del proyecto desde el fichero que quieres. Esto se hace en: Run/Set project configuration/Customize

Te visualiza esta ventana:

Donde pone "Main Class" hay que indicar el fichero por el que quieres que empiece la ejecución del programa. Dando a "Browse" podrás elegir el fichero. Sólo te dejaré elegir los ficheros que tengan un método con esta cabecera: "public static void main(String[] args)".

- Mediante el icono de acceso directo en la parte superior de la ventana de edición de código. Es igual que si dieras a Run/Run project. Con lo cual, tiene que estar configurado (como se indicó en su momento) para que en el ejecución del proyecto empiece por el fichero que tú quieres.

- Pulsando F6. Es igual que si dieras a Run/Run project. Con lo cual, tiene que estar configurado (como se indicó en su momento) para que en el ejecución del proyecto empiece por el fichero que tú quieres.

El resultado que obtenemos (si todo ha ido bien) es:

Anexo IX.- Ejemplos completo con Eclipse

En el módulo **Programación** realizarás gran cantidad de programas, probablemente en **Java** con **Eclipse** o con alguna otra combinación de lenguaje/**IDE**.

Aquí se presentan un par de sencillos programas para iniciarnos en el manejo de **Eclipse**.

a.- Ejemplo "Hola Mundo"

La secuencia para crear un programa Java mediante **Eclipse** es:

Crear un nuevo proyecto **Java**. Opción de menú "Archivo/Nuevo/Proyecto Java." Si no aparece directamente la opción de proyecto **Java** probar con "Archivo/Nuevo/Otras/Java/Proyecto Java".	
Introducir el nombre del proyecto. En este caso **HolaMundo** y pulsar Finalizar.	
En el explorador de paquetes ya aparecerá el nuevo proyecto.	
Crear la **clase HolaMundo** que incluya el **método main** de inicio del programa. Botón derecho sobre el proyecto y seleccionar "Nuevo/Clase". Indicar como nombre HolaMundo y seleccionar el check-box para que se cree automáticamente el método main.	
En el apartado anterior se habrá generado el fichero **HolaMundo.java** en "/src/paquete predeterminado". Pulsa doble click con el ratón para editarlo.	
El fichero ya incluye el **método main** (anteriormente lo habíamos indicado al crear la clase). Introducir el código que aparece en la línea 6 para mostrar por consola un mensaje de saludo.	
La ejecución del programa se puede llevar a cabo de varias formas, una de ellas es desde la opción de menú "Ejecutar/ejecutar". Obtendremos el siguiente resultado.	
Eclipse ha organizado el proyecto como un directorio a partir de la ruta del **WorkSpace** y una serie de directorios/ficheros colgando de éste.	

Para saber más

En el siguiente vídeo, se hace un repaso de la adición de nuevas funcionalidades a NetBeans:

Resumen textual alternativo

b.- Ejemplo "Calculadora"

Ejercicio Propuesto

Esta práctica si se hará usando el entorno de desarrollo "Eclipse".

Tiene que crear un proyecto que se llame "Calculadora"

Crea la clase "Calculadora" y en su método main copia el siguiente código:

```
import java.util.Scanner;
 public class Calculadora {
     public static void main(String[] args) {
          Scanner miScan = new Scanner(System.in);
          System.out.println("Calculadora que suma dos numeros enteros");
          System.out.println("Introduce el primer numero");
          String sPrimerNum = miScan.nextLine();
          int iPrimerNum = Integer.parseInt(sPrimerNum);
          System.out.println("Introduce el primer numero");
          String sSegNum = miScan.nextLine();
          int iSegNum = Integer.parseInt(sSegNum);
          int iResul = iPrimerNum + iSegNum;
          System.out.println("El resultado es: " + iResul);
     }
}
```

Compilar el programa y corrige los errores sintácticos. Incluye las librerías que sean necesarias. Ejecuta el programa.

Instalar el plugin ECalculator (calculadora integrada en el IDE), que se encuentra disponible en el Market place de Eclipse. Una vez instalado, aparece disponible como una vista que podrás añadir a tu perspectiva de trabajo.

Comprobar que el programa funciona correctamente haciendo uso del **plugin**. Realizar una suma ejecutando el programa y cotejar su resultado con el plugin.

Comprobar si existen actualizaciones disponibles para el **IDE**. Del listado de actualizaciones pendientes que te aparezca, seleccionar las que consideres oportunas e instalar sus nuevas versiones.

Desinstalar el plugin ECalculator.

Mostrar retroalimentación

Esta práctica si se hará usando el entorno de desarrollo "Eclipse".

Para ello hay que instalar JDK y Eclipse indicado en el punto 5.1 (instalación del JDK) Y 5.2.2 (instalacion de Eclipse)

Tiene que crear un proyecto que se llame "Calculadora"

Crea la clase "Calculadora" y en su método main copia el siguiente código:

```
import java.util.Scanner;
 public class Calculadora {
     public static void main(String[] args) {
          Scanner miScan = new Scanner(System.in);
          System.out.println("Calculadora que suma dos numeros enteros");
          System.out.println("Introduce el primer numero");
          String sPrimerNum = miScan.nextLine();
          int iPrimerNum = Integer.parseInt(sPrimerNum);
          System.out.println("Introduce el primer numero");
          String sSegNum = miScan.nextLine();
          int iSegNum = Integer.parseInt(sSegNum);
          int iResul = iPrimerNum + iSegNum;
```

```
            System.out.println("El resultado es: " + iResul);
        }
    }
```

Compilar el programa y corrige los errores sintácticos. Incluye las librerías que sean necesarias. Ejecuta el programa.

Instalar el plugin ECalculator (calculadora integrada en el IDE), que se encuentra disponible en el Market place de Eclipse. Una vez instalado, aparece disponible como una vista que podrás añadir a tu perspectiva de trabajo.

Disponible para instalación en: https://marketplace.eclipse.org/content/ecalculator.

Comprobar que el programa funciona correctamente haciendo uso del **plugin**. Realizar una suma ejecutando el programa y cotejar su resultado con el plugin.

Comprobar si existen actualizaciones disponibles para el **IDE**. Del listado de actualizaciones pendientes que te aparezca, seleccionar las que consideres oportunas e instalar sus nuevas versiones.

Desinstalar el plugin ECalculator.

c.- Ejemplo "Semáforo"

En el bloque de Actividades y Recursos de esta unidad en el apartado Recursos disponibles está disponible el fichero semaforo.zip que incluye el código fuente. que está organizado en dos paquetes: principal y clases. En este ejemplo crearemos el proyecto, los dos paquetes e incluiremos el código fuente arrastrando los ficheros java disponibles hasta el **IDE** para integrarlos en el proyecto.

Crear un nuevo proyecto Java Semaforo.	
Crear los paquetes principal y clases. Sobre el proyecto, botón derecho del ratón y seleccionar "Nueva/Paquete". Una vez por cada paquete.	
Desde el administrador de archivos, arrastrar los ficheros java a sus respectivos paquetes (principal y clases). Indicando que se haga una copia de los mismos al proyecto.	
Ejecutamos con la opción de menú "Ejecutar/ejecutar".	
Puede resultar interesante comprobar en el sistema de archivos la estructura de proyecto que se ha generado. Acude a la ruta del proyecto creada en tu **Workspace**.	

Anexo X.- Licencias de recursos.

Recurso (1)	Datos del recurso (1)	Recurso (2)	Datos del recurso (2)
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: http://netbeans.org		Autoría: jongalloway. Licencia: CC BY-NC-SA 2.0. Procedencia: http://www.flickr.com/photos/jongalloway/2053978954/
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: http://www.eclipse.org/downloads/packages/eclipse-classic-37/indigor Imagen ampliada		Autoría: Hadrián Fernández. Licencia: CC-by-nc-sa. Procedencia: http://www.flickr.com/photos/adrian-deejay/442309087/
	Autoría: Francisco Palacios. Licencia: CC by -NC-ND 2.0. Procedencia: http://www.flickr.com/photos/wizard_/3303810302/		Autoría: Ubuntu 10.10. Licencia: GNU. Procedencia: http://www.ubuntu.com.
	Autoría: Ubuntu10.10. Licencia: GNU Procedencia: http://www.ubuntu.com.		Autoría: Windows Licencia: GNU Procedencia: http://www.windows.com
	Autoría: Windows Licencia: GNU Procedencia: http://www.windows.com		Autoría: Windows Licencia: GNU Procedencia: http://www.windows.com
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.		Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.
	Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.		Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.
	Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.		Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.
	Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial.		Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial.

	Procedencia: Elaboración propia.		Procedencia: Elaboración propia.
	Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.		Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: Silveira Neto. Licencia: CC by-sa 2.0. Procedencia: http://www.flickr.com/photos/s ilveiraneto/2579658422/		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: http://netbeans.org.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbean s. ED02_CONT_R19_JDK-ubuntu 10.pdf Miniatura Comentarios Credenciales del recurso.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones.

	Procedencia: Captura de pantalla de Netbeans.		Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.

	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.
	Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.		Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.
	Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.		Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.
	Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.		Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.
	Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.		Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.

	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.
	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.

	Autoría: netbeans.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Netbeans.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.		Autoría: eclipse.org. Licencia: Copyright (cita), se autoriza el uso sin restricciones. Procedencia: Captura de pantalla de Eclipse.
	Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux		Autoría: Windows Licencia: GNU. Procedencia: Captura de pantalla de Windows.
	Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux		Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux
	Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux		Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux
	Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux		Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux
	Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux		Autoría: Linux Licencia: GNU. Procedencia: Captura de pantalla de Linux