2. INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

1. INTRODUCCIÓN

- IDE son las siglas de Integrated Development Environment (Entorno Integrado de Desarrollo)
- En la unidad anterior tratamos las fases en el proceso de desarrollo de software.
- Una de ellas era la fase de codificación, en la cual se hacía uso de algún lenguaje de programación para pasar todas las acciones que debía llevar a cabo la aplicación a algún lenguaje que la máquina fuera capaz de entender y ejecutar.
- Un IDE es una aplicación informática que está compuesta por un conjunto de herramientas de programación que van a facilitar esta tarea.

2. Componentes de un Entornos de Desarrollo

- Editor de Texto.
  - Es la parte que nos permite escribir el código fuente del programa, ofrece funciones propias de la edición como copiar, pegar, etc.
  - Además, es capaz de reconocer, resaltar y cambiar los colores de las variables, las cadenas de caracteres, las palabras reservadas, las instrucciones, el inicio y final de los corchetes, etc.
  - De esta manera el código fuente será mucho más visual, cómodo y se podrán reconocer los errores a simple vista.
- Compilador
  - Es el encargado de traducir el código fuente en lenguaje de alto nivel a lenguaje máquina.

- Intérprete o interpretador.
  - Los intérpretes se diferencian de los compiladores en que solo realizan la traducción a medida que se va ejecutando la instrucción. Normalmente no guardan el resultado de dicha traducción (por eso son un poco más lentos).
- Depurador (debugger)
  - Es el encargado de depurar y limpiar los errores en el código fuente de un programa informático. El depurador permite examinar paso a paso, instrucción a instrucción, la ejecución de un programa y examinar las distintas situaciones y cambios que se produzcan en las variables del programa o en los registros del procesador.
  - El depurador va a permitir detener el programa en cualquier punto de ruptura (breakpoint) para examinar la ejecución.

- Constructor de interfaz gráfica
  - Esta herramienta de programación simplifica la creación de interfaces gráficas de usuario permitiendo al diseñador colocar los controles (botones, listas, menús, y demás elementos) utilizando un editor WYSIWYG de arrastrar y soltar.
  - Algunos IDE incorporar estas herramientas con el plugin correspondiente, es el caso de Eclipse.
- Control de versiones
  - Estas herramientas controlan los cambios que se realizan sobre las aplicaciones, de esta manera se obtendrán revisiones y versiones de las aplicaciones en un momento dado de su desarrollo.

- 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 o Microsoft Visual Studio) mediante la instalación de plugins adicionales.

- Los IDE son altamente configurables, ya que las necesidades de cada programador o grupo de trabajo pueden ser diferentes. El objetivo es ofrecer al usuario una aplicación amigable con la que trabajar, por lo que poder personalizar y configurar la herramienta es un aspecto verdaderamente importante.
- Esta configuración permite añadir y modificar la barra de herramientas, pudiendo crear comandos y atajos de teclado, establecer el posicionado de las ventanas o barras.
- Las configuraciones de depuración y compilación son una práctica recurrente en el desarrollo de aplicaciones, por lo que resulta extremadamente efectivo poder configurarlas al gusto.

2.1 CRITERIOS DE ELECCIÓN DE UN IDE

- Para poder elegir correctamente el IDE en el que vamos a trabajar, necesitamos saber las características que buscamos en él, si satisface nuestras necesidades y si nosotros mismos cumplimos los requisitos del IDE.
- 1. Sistema Operativo
  - Sin lugar a dudas, uno de los criterios más restrictivos a la hora de seleccionar nuestro IDE es saber en qué sistema operativo vamos a trabajar y, más importante aún, para qué sistema operativo vamos a desarrollar nuestro software.
  - Siempre y cuando nuestro software no vaya a ser ejecutado en una máquina virtual, estaremos desarrollando para un sistema operativo concreto, por lo que si estamos desarrollando aplicaciones para Linux, resultaría bastante inusual desarrollar la aplicación en Windows para ella.

- 2. Lenguaje de Programación y Framework.
  - Un IDE puede soportar uno o varios lenguajes de programación, por lo que saber en qué lenguaje de programación vamos a codificar nuestro software y qué lenguajes nos ofrecen los distintos IDE es una información valiosa que hay que tener en cuenta.
- 3. Herramientas y disponibilidad
  - Las distintas herramientas de las que disponen los IDE son el último criterio de selección, seguramente nos encontraremos con varios IDE que cumplen los requisitos de lenguaje y sistema operativo, pero no todos tienen las mismas funciones, por lo que saber cuáles son esas herramientas es un dato sumamente importante en nuestra decisión.

- En ocasiones, pueden ser restrictivos ya no solo por tus propias preferencias, sino por trabajar de manera colaborativa y el modo de utilizar e interpretar diferentes códigos entre diferentes IDE (por ejemplo el sistema de control de versiones TFS, Team Server Foundation, solo funciona con Visual Studio).
- Podemos buscar un IDE que permita crear archivos de ayuda o con facilidad o que tuviese una extensión o plugar para aportar esa funcionalidad deseada.

- Los IDE pueden tener un precio de aplicación
- A simple vista, parecen demasiados factores a tener en cuenta y una operación larga y tediosa descubrir cual es nuestro IDE, pero no debemos agobiarnos por esta cuestión, ya que, al igual que con los lenguajes de programación, no podemos pretender empezar sabiendo todas las posibilidades que nos ofrecen y tenemos que ir aprendiéndolas de modo incremental en función de nuestra experiencia.

2.2 IDE LIBRES Y PROPIETARIOS

- 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.

IDE	Lenguajes que soporta	Sistema Operativo
NetBeans.	C/C++, Java, JavaScript, PHP, Python.	Windows, Linux, Mac OS X.
Eclipse.	Ada, C/C++, Java, JavaScript, PHP.	Windows, Linux, Mac OS X.
Gambas.	Basic.	Linux.
Anjuta.	C/C++, Python, Javascript.	Linux.
Geany.	C/C++, Java.	Windows, Linux, Mac OS X.
GNAT Studio.	Fortran.	Windows, Linux, Mac OS X.

- 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.

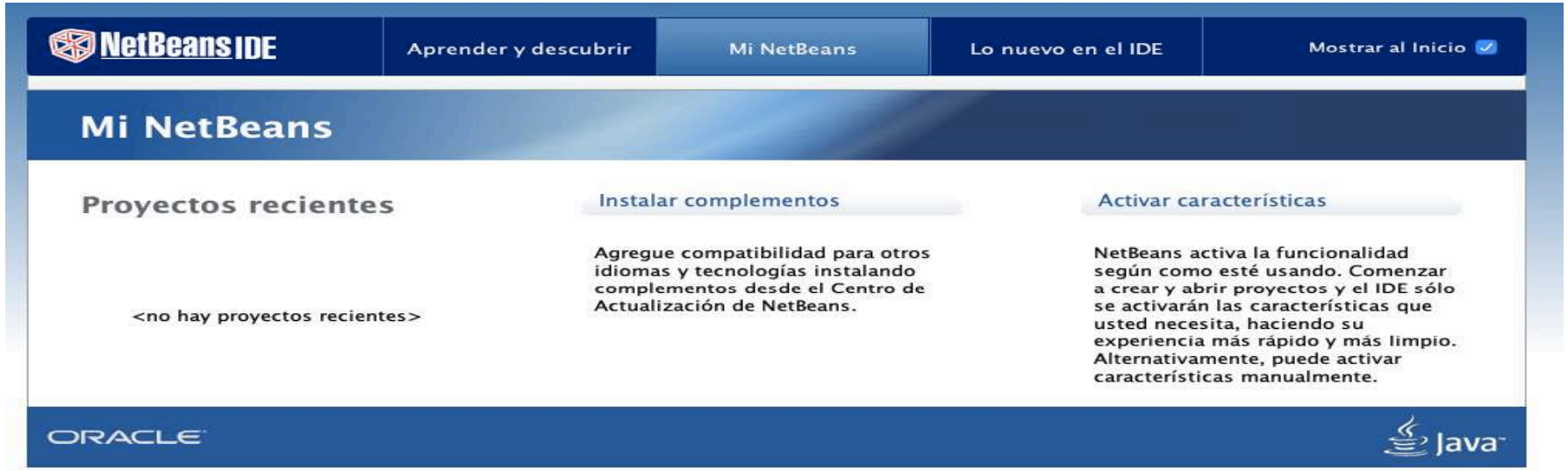
IDE	Lenguajes que soporta	Sistema Operativo
Microsoft Visual Studio.	Basic, C/C++, C#.	Windows.
FlashBuilder.	ActionScript.	Windows, Mac OS X.
C++ Builder.	C/C++.	Windows.
Turbo C++ profesional.	C/C++.	Windows.
JBuilder.	Java.	Windows, Linux, Mac OS X.
JCreator.	Java.	Windows.
Xcode.	C/C++, Java.	Mac OS X.

3. INSTALACIÓN DE IDE

- Siguiendo los criterios que hemos ido viendo a lo largo, hemos elegido utilizar NetBeans como entorno de desarrollo.
- NetBeans es uno de los entornos de desarrollo más pulidos, profesionales y completos que podemos encontrar en el mercado, además es ideal para trabajar con el lenguaje de programación JAVA siendo además un entorno integrado libre por lo que su instalación no llevará ningún coste y podrá ser instalado en cualquier plataforma.
  - Instalación en Linux: https://www.youtube.com/watch?v=aneC2LVizzY
  - Instalación en Mac: https://www.youtube.com/watch?v=-z3QPCHppNU
  - Instalación en Windows: https://www.youtube.com/watch?v=feiEbOPTM4g
- Nota: Antes de instalar NetBeans, instalaremos el Java (versión 8)

4. RECORRIDO POR NETBEANS

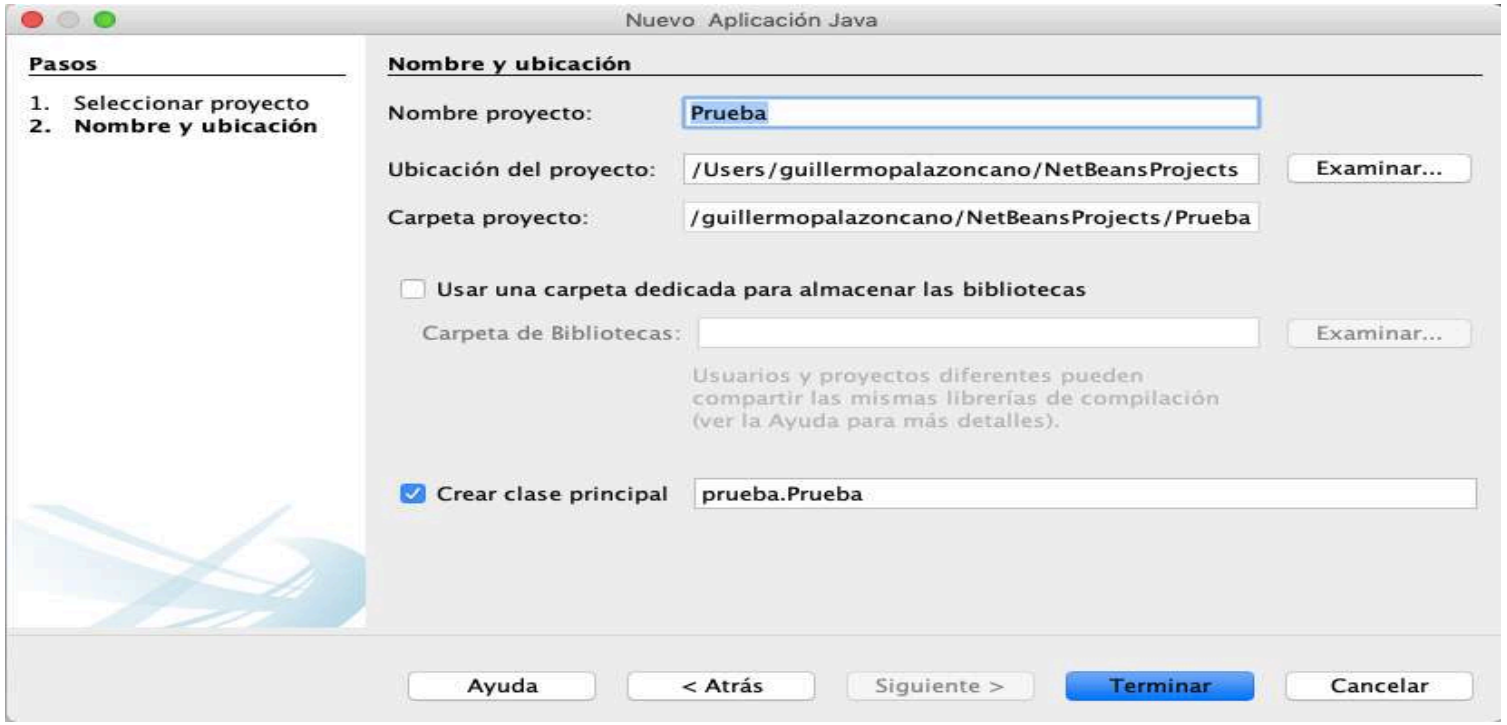
- AL arrancar el IDE, dispondremos siempre de una ventana de inicio con varias opciones. En la pestaña de "Mi NetBeans" podemos Instalar nuevos complementos o activar características



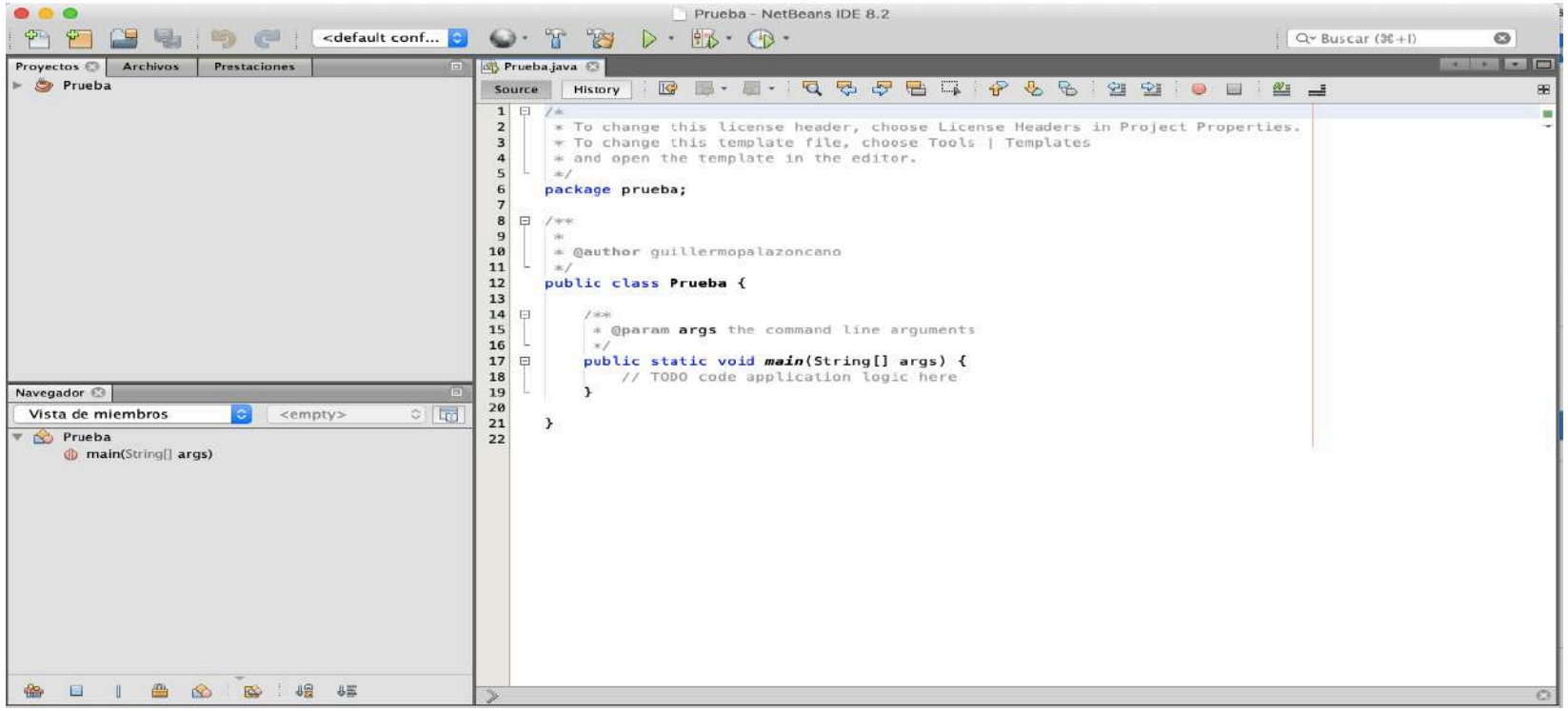
4.1 Crear proyectos

- Podemos crear nuevos proyectos desde Archivo -> Proyecto Nuevo

- Comenzaremos creando una aplicación Java, por lo elegiremos esa opción y le daremos un nombre a nuestro Proyecto (Prueba).
- Dentro de las opciones, podemos elegir crear la clase principal. La clase principal es la que posee el método main, por lo tanto será la que se lance cuando se pida compilar y ejecutar el proyecto.
- En un proyecto solo debe haber una clase con el método main, si la elegimos al crear el proyecto se crea directamente (es lo recomendable).
- También por defecto, Netbeans crea un paquete con el mismo nombre que el del proyecto. Aunque no es mala política, podemos desear estructurar de otra manera.

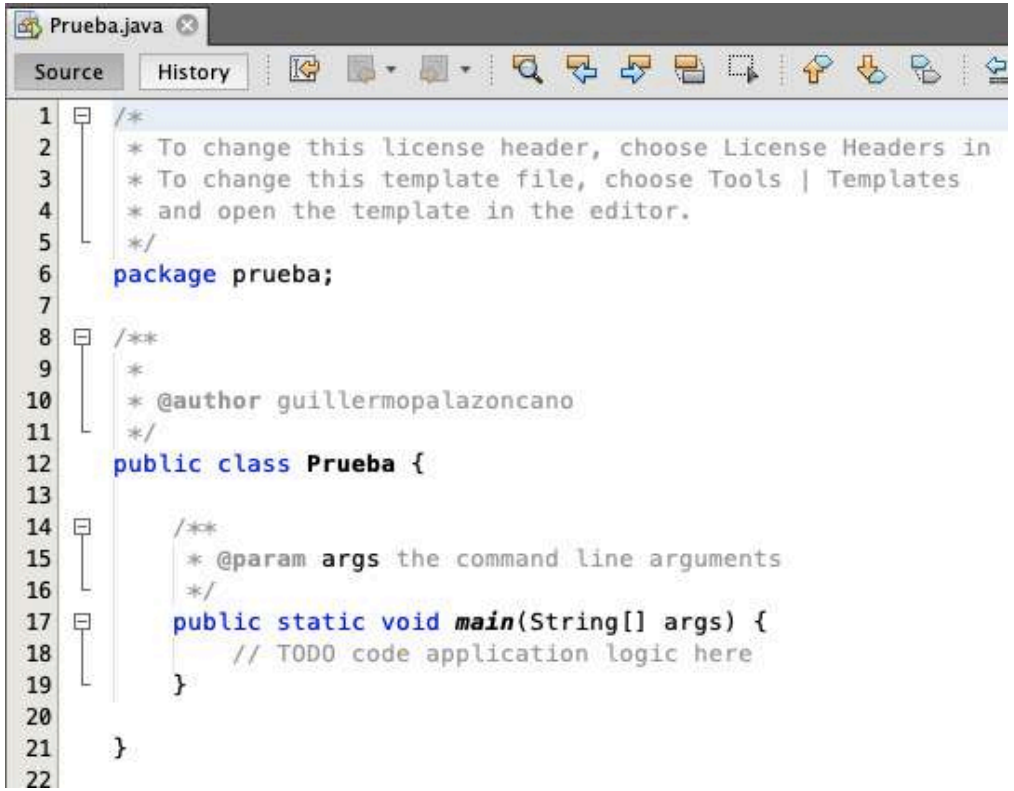

Nuevo Aplicación Java
Pasos
1. Seleccionar proyecto
2. Nombre y ubicación
Nombre y ubicación
Nombre proyecto:
Prueba
Ubicación del proyecto:
/Users/guillermopalazoncano/NetBeansProjects
Examinar...
Carpeta proyecto:
/guillermopalazoncano/NetBeansProjects/Prueba
Usar una carpeta dedicada para almacenar las bibliotecas
Carpeta de Bibliotecas:
Examinar...
Usuarios y proyectos diferentes pueden
compartir las mismas librerías de compilación
(ver la Ayuda para más detalles).
Crear clase principal
prueba.Prueba
Ayuda
< Atrás
Siguiente >
Terminar
Cancelar

- El aspecto tras crear el nuevo proyecto es este:



- En el lado izquierdo se seleccionan las carpeta fundamentales de los proyectos NetBeans.
- Los paquetes de fuentes contienen el código de las clases del proyecto. Ahí se crea el código del proyecto
- Las bibliotecas son clases directamente invocables desde nuestro proyecto.

- A la derecha tenemos el editor de texto.
- En ella se muestra el texto de aquellos ficheros que hayamos abierto.
- En la imagen podemos ver el fichero que se genera de manera automáticamente al crear el proyecto indicado (clase Prueba.java)



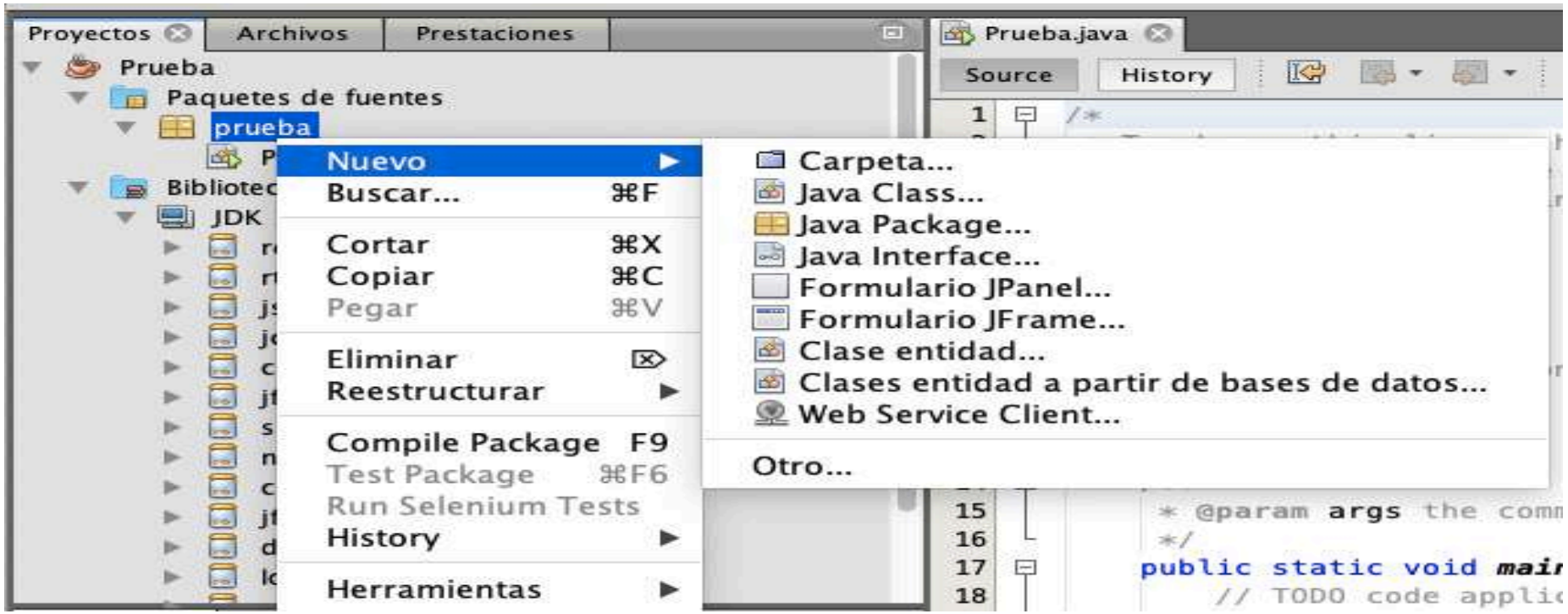
4.2 CREAR PAQUETES

- Las clases se organizan en paquetes.
- Es recomendable que cada proyecto tenga al menos un paquete, de otro modo las clases se crearían en el paquete por defecto y eso no es aconsejable.
- Pulsando el botón derecho en la zona del proyecto donde queremos crea el paquete (normalmente dentro de paquetes de fuentes) podremos elegir Nuevo-Paquete. Tras rellenar el cuadro siguiente (donde simplemente se elige el nombre para el paquete, siempre en minúsculas) tendremos un nuevo paquete.
- **NORMA DE CODIFICACIÓN: Los nombres de los paquetes en minúscula.**


Nuevo Java Package
Pasos
1. Escoja el tipo de archivo
2. Name and Location
Name and Location
Package Name: newpackage
Project: Prueba
Location: Paquetes de fuentes
Created Folder: /Users/guillermopalazoncano/NetBeansProjects/Prueba/src/
Ayuda
< Atrás
Siguiente >
Terminar
Cancelar

4.3 CREAR NUEVAS CLASES

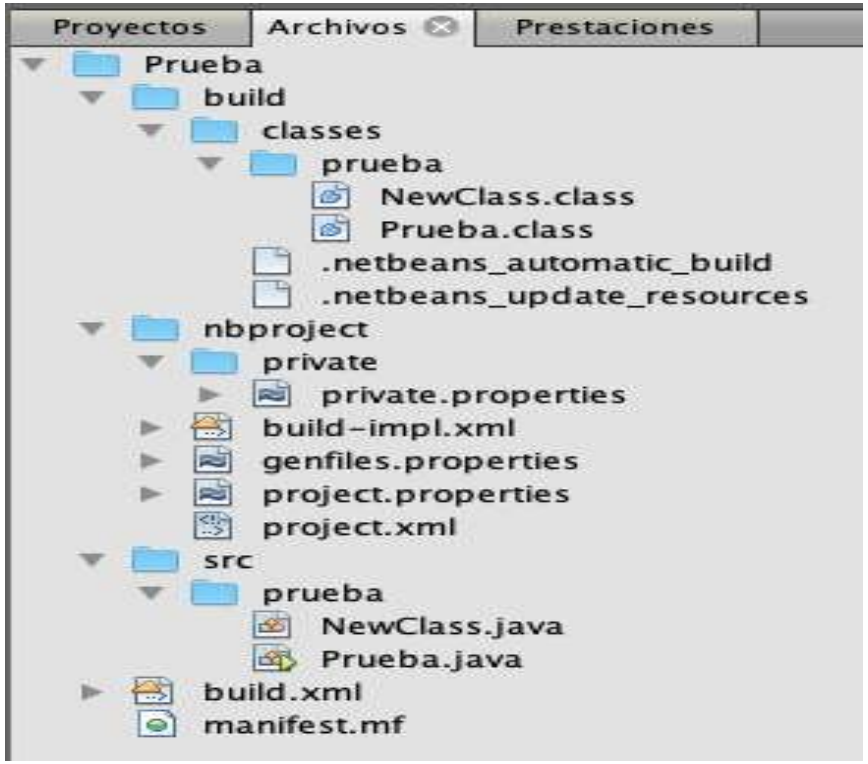
- Simplemente se crean pulsando el botón derecho en el paquete en el que deseamos almacenar la clase y eligiendo Nuevo-Clase.
- Al principio lo normal es que nuestros proyectos sólo contengan la clase principal. A medida que profundicemos en el aprendizaje de Java, necesitaremos más clases (y más paquetes) en cada proyecto.
- Las clases en Java son básicamente una plantilla que sirve para crear un objeto.
- Si imaginásemos las clases en el mundo en el que vivimos, podríamos decir que la clase "**persona**" es una plantilla sobre cómo debe ser un ser humano.


Proyectos
Archivos
Prestaciones
Prueba
Paquetes de fuentes
prueba
Nuevo
Buscar... ⌘F
Cortar ⌘X
Copiar ⌘C
Pegar ⌘V
Eliminar
Reestructurar
Compile Package F9
Test Package ⌘F6
Run Selenium Tests
History
Herramientas
Carpeta...
Java Class...
Java Package...
Java Interface...
Formulario JPanel...
Formulario JFrame...
Clase entidad...
Clases entidad a partir de bases de datos...
Web Service Client...
Otro...
Prueba.java
Source
History
1 /*
15 * @param args the comm
16 */
17 public static void mai
18 // TODO code applic

4.4 VISTAS DEL PROYECTO

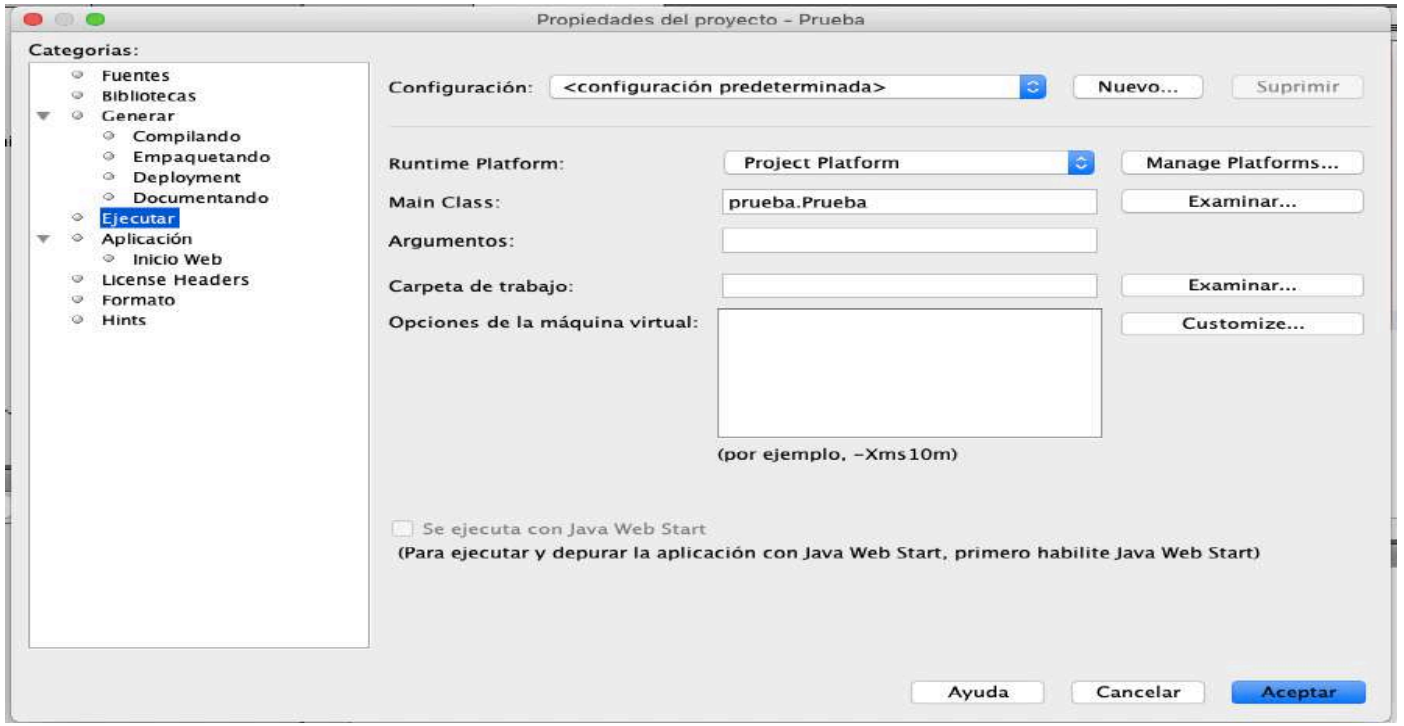
- En Netbeans sobre todo hay dos formas de examinar el proyecto. La primera es a través de la pestaña Proyectos. En ella aparece la organización lógica del proyecto.

- Sin embargo la vista Archivos nos enseña la realidad de los archivos del proyecto. Nos enseña la organización de los archivos del proyecto desde el punto de vista del sistema
- La carpeta build contiene los archivos compilados (los class).
- La carpeta src el código fuente.
- El resto son archivos por Netbeans para comprobar la configuración del proyecto o los archivos necesarios para la correcta interpretación del código en otros sistemas (en cualquier caso no hay que borrarlos).



4.5 PROPIEDADES DEL PROYECTO

- Pulsando el botón derecho sobre un proyecto, podemos elegir Propiedades. El panel de propiedades nos permite modificar la configuración del proyecto. Por ejemplo podremos indicar una nueva clase principal en el proyecto



4.6 ACCIONES SOBRE UN PROYECTO

- 1. Eliminar un Proyecto
  - Si deseamos quitar un proyecto podemos: pulsar el botón secundario del ratón sobre el icono de proyectos (en la ventana de Proyectos) y elegir Suprimir; o podemos seleccionar el proyecto y pulsar la tecla Suprimir.
  - En cualquier caso Netbeans nos pregunta si deseamos realmente borrar el proyecto. La casilla Eliminar también el código fuente que está en la carpeta… en el caso de marcarse, eliminaría no solo el proyecto de Netbeans, sino también todos los archivos del proyecto quedaría eliminados del disco.

- 2. Importar un proyecto de Eclipse
  - Para poder ver y trabajar un proyecto de Eclipse en Netbeans basta con:
  - Elegir Archivo-Importar Proyecto-Proyecto Eclipse
  - Seleccionar un directorio del que cuelguen proyectos de Eclipse que nos interese utilizar desde Netbeans
  - En el paso siguiente hay que elegir los proyectos concretos que vamos a importar.
  - Si modificamos proyectos desde Netbeans y luego queremos utilizarlos en Eclipse, podríamos tener problemas. Por ello es mejor hacer copia primero de los archivo que utilizaremos.

4.7 COMPILAR Y EJECUTAR PROGRAMAS

- Nuestro primer programa en NetBeans. Hola Mundo
  - Dentro del procedimiento main, añadiremos la línea: *System.out.println("Hola Mundo");*
- Para probar el proyecto, debemos ejecutarle. El proyecto principal (será aquel sobre el que marquemos la casilla al crearle) se puede compilar y ejecutar de golpe botón derecho sobre la imagen del proyecto y eligiendo Ejecutar, o simplemente pulsando la tecla F6. También podemos hacer lo mismo desde el menú Ejecutar.

- Un proyecto puede tener varias clases ejecutables (varias clases con main) en ese caso para compilar un archivo concreto, basta con seleccionarle en la ventana de proyectos, pulsar el botón derecho y elegir ejecutar archivo
- También podemos hacer lo mismo si estamos en el código que deseamos ejecutar y pulsamos Mayúsculas+F6.
- El panel de Salida mostrará el resultado de la compilación.

- Vamos a cambiar el texto de nuestro procedimiento y va a tener cuatro variables de tipo int numero1, numero2, numero3 y numero4.
- A numero1 vamos a asignarle el valor 5, numero2 va a tener el valor 7 y numero3 va a realizar la suma y la va a mostrar por consola.
- Escribe el código correspondiente para que nuestro Main llame a esta función y ejecuta el mismo.
  - ¿Te subraya algo NetBeans y por qué crees es?
- Cambia ahora el valor de numero2 por 7.3
  - ¿Qué nos está indicando ahora NetBeans?

- Vamos ahora a cambiar el código y en lugar de tener en el procedimiento main de nuestra clase principal vamos a crear una nueva clase que se va a llamar Matematicas.
- Dentro de esa clase vamos a tener una función que se va a llamar suma_fija que va a realizar lo indicado anteriormente.
- Ahora desde nuestra clase principal vamos a llamar a dicha función.
- La cosa se complica un poco porque debemos tener en cuenta que estamos trabajando con Java y es un lenguaje orientado a Objetos.
- ¿Cómo se realizaría?

```java
package prueba;

/**
 *
 * @author guillermopalazoncano
 */
public class Matematicas {
    public void suma_fija(){
        int numero1, numero2, numero3, numero4;
        numero1 = 5;
        numero2 = 7;
        numero3 = numero1+numero2;
        System.out.println("El resultado es: " + numero3);
    }
}
```

```java
public class Prueba {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Matematicas matematicas = new Matematicas();
        matematicas.suma_fija();

    }

}
```

- Por último, hemos decidido que en nuestra clase Matemáticas vamos a tener una nueva función que sea suma_variable.
- En esta función el valor de cada número a sumar lo va a pedir nuestro procedimiento por pantalla.
- Me pasan el código pero tiene errores.
- Vamos a tratar de solucionarlos y vamos a ver el motivo

```
public void suma_variable(){
    int numero1, numero2, resultado, resultado_final;

    double resultado;

    Scanner reader = new Scanner(System.in);
    System.out.println("Introduce el primer numero:");
    numero1= reader.nextDouble();

    System.out.println("Introduce el segundo numero:");
    numero2=reader.nextDouble();

    // Resultados
    resultado = numero1+numero2;

    System.out.println("Resultado: " + resultado);

}
```

```java
public void suma_variable(){
    int numero1, numero2, resultado, resultado_final;

    Scanner reader = new Scanner(System.in);
    System.out.println("Introduce el primer numero:");
    numero1= (int) reader.nextDouble();
    System.out.println("Introduce el segundo numero:");
    numero2=(int) reader.nextDouble();

    // Resultados
    resultado = numero1+numero2;
    System.out.println("Resultado: " + resultado);

}
```

5. CONFIGURACIÓN Y PERSONALIZACIÓN DE ENTORNOS DE DESARROLLO

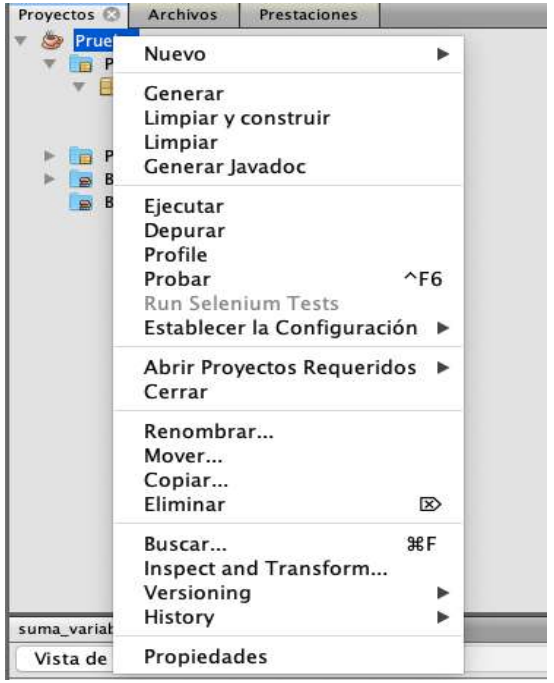
- Hemos indica en varias ocasiones ya que unas de las grandes ventajas de un IDE es la posibilidad de poder personalizarlo a nuestro gusto.
- Una vez tenemos instalado nuestro IDE podemos acceder a su configuración.
- Al abrir un proyecto existente, o bien crear un nuevo proyecto, podemos configurar sus propiedades. Esto nos permitirá personalizar distintas opciones del proyecto.
- Podemos también personalizar la configuración del entorno sólo para el proyecto actual, o bien para todos los proyectos, presentes y futuros con los que vayamos a trabajar.

- Algunos ejemplos de parámetros configurables del entorno:
  - Carpeta/s 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.

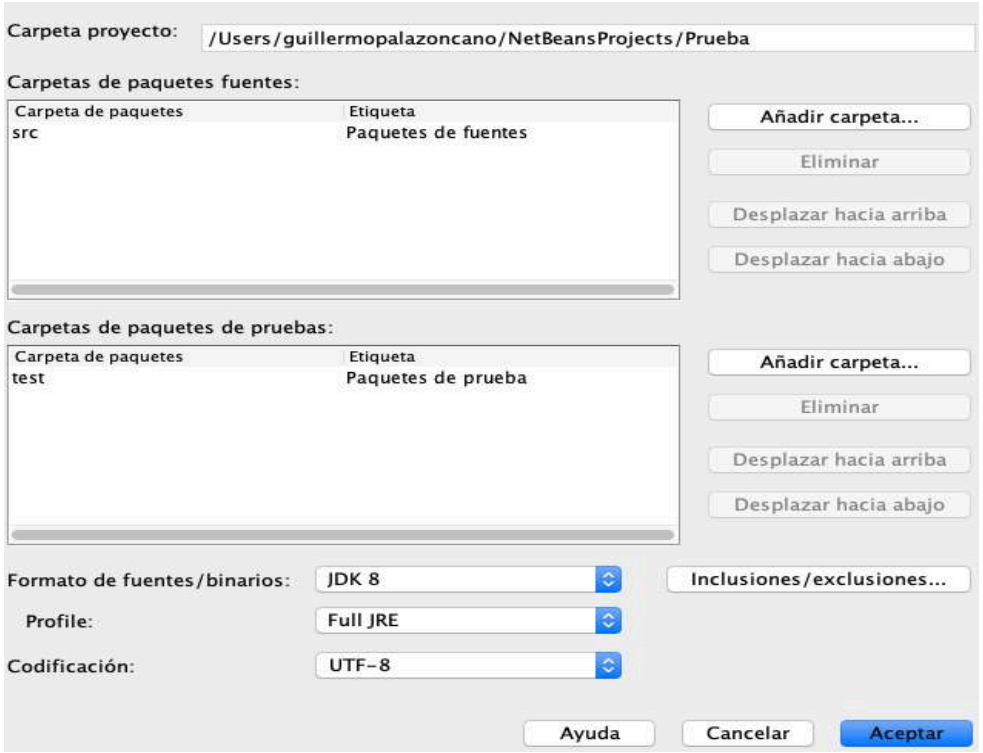
- Opciones de generación de documentación asociada al proyecto.
- Descripción de los proyectos, para una mejor localización de los mismos.
- Opciones globales de formato del editor: número de espaciados en las sangrías, color de errores de sintaxis, color de etiquetas, opción de autocompletado de código, propuestas de insertar automáticamente código...
- Opciones de combinación de teclas en teclado.
- Etc.

5.1 CONFIGURACIÓN PROYECTO

- En el menú izquierdo seleccionamos un proyecto y tras pulsar con el botón derecho, pulsamos en la opción de Propiedades para poder configurar nuevas cuestiones sobre el mismo.
- Podemos ver todo lo que se puede personalizar de la aplicación
  - Fuentes
  - Bibliotecas
  - Generación de código
  - Ejecución de código
  - Opciones de la aplicación
  - Formato del código



- **Fuentes**
  - Carpeta que contendrá el proyecto.
  - Carpeta que almacenará los paquetes fuentes.
  - Carpeta que almacenará los paquetes de prueba.
  - También se puede cambiar la codificación o el formato de fuentes/binarios.
  - También podemos incluir clases a nuestro proyecto o excluirlas



- **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 el proyecto.
  - También en esta ventana se puede configurar el paquete de pruebas que se realizará al proyecto.
  - En un apartado posterior veremos como se puede añadir una nueva biblioteca al proyecto.

- **Generación de código.** Dentro de generación de código tenemos a su vez diferentes opciones: Compilando, Empaquetando, Deployment (despliegue) y Documentando.
- Compilando.- Las opciones que nos permite modificar en cuánto a la compilación del programa entre otros son:
  - Compilar al grabar– Al guardar un archivo se compilará automáticamente
  - Generar información de depuración: para obtener la documentación asociada.
  - Enable annotation processing: permitir anotaciones durante el proceso.
  - Report uses of Deprecated APIs: Informe del uso de herramientas obsoletas

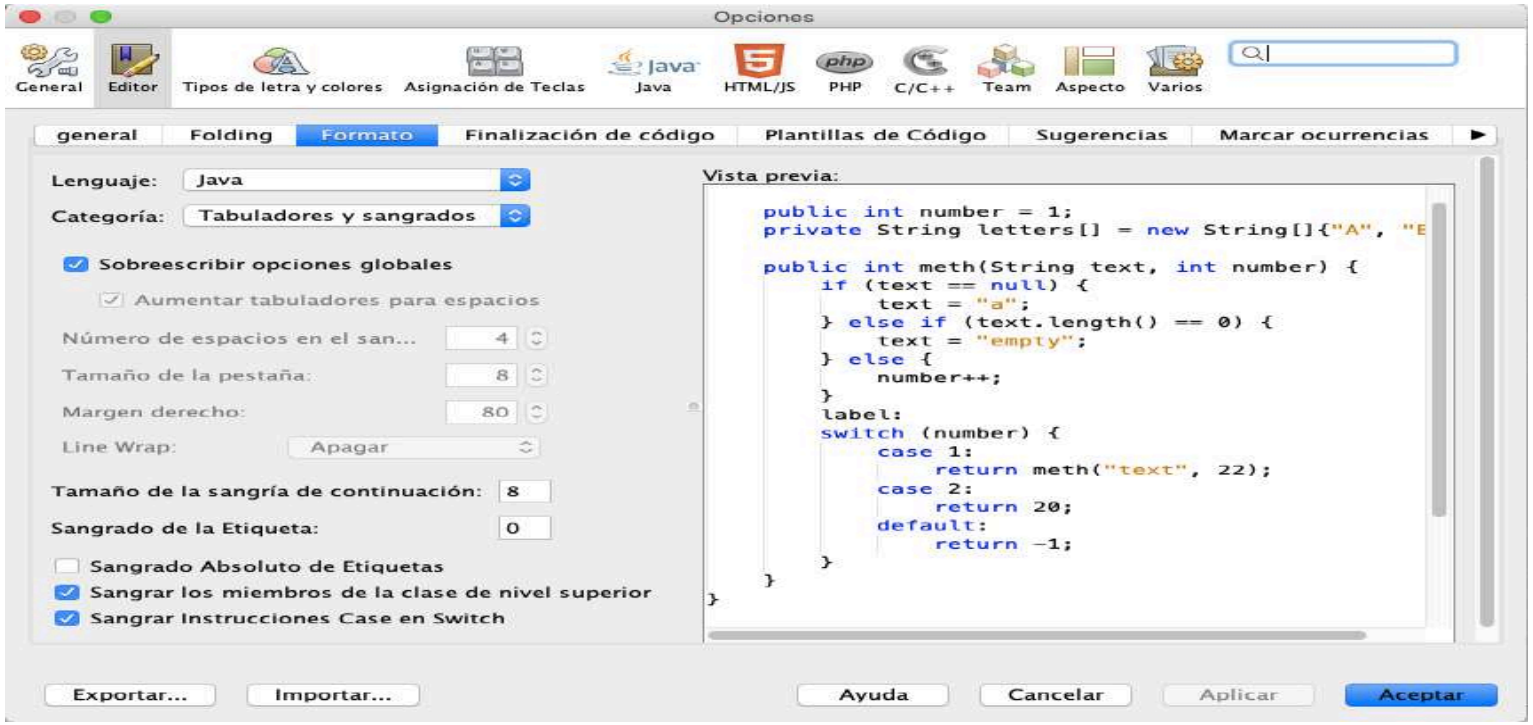
- Empaquetando
  - Las aplicaciones resultado de la compilación del código deben ser empaquetadas antes de su distribución, con objeto de tener un único archivo, generalmente comprimido, que contenga en su interior todos los archivos de instalación y configuración necesarios para que la aplicación pueda ser instalada y desarrollada con éxito por el usuario cliente.
  - En esta opciones podemos modificar el lugar donde se generará el archivo resultante del empaquetado, así como si deseamos comprimirlo.
  - También podemos elegir que el archivo empaquete se construya tras la compilación, que es lo habitual y por eso viene marcado y que se copien las librerías dependientes.
  - En un apartado posterior se explicará como empaquetar o preparar la distribución.

- Deployment
  - Permite activar el empaquetado nativo.
  - Toma la aplicación tal como está junto a la máquina Java y produce un instalador que es común para el sistema operativo que está utilizando.
  - El tamaño de dichos instaladores es bastante grande, porque incluso un "Hola mundo" La aplicación llevará consigo una gran parte de los artefactos de tiempo de ejecución de Java.
  - Con algunas aplicaciones extra podríamos crear un instalador EXE (Inno Setup en Windows)

- Documentando
  - Como ya hemos indicado la documentación de aplicaciones es un aspecto clave que no debemos descuidar nunca.
  - NetBeans 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 que indica
    - Árbol de jerarquía de clases, páginas de uso de clases y paquetes, barra de navegación e índices.
  - A lo largo de la asignatura aprenderemos a generar documentación

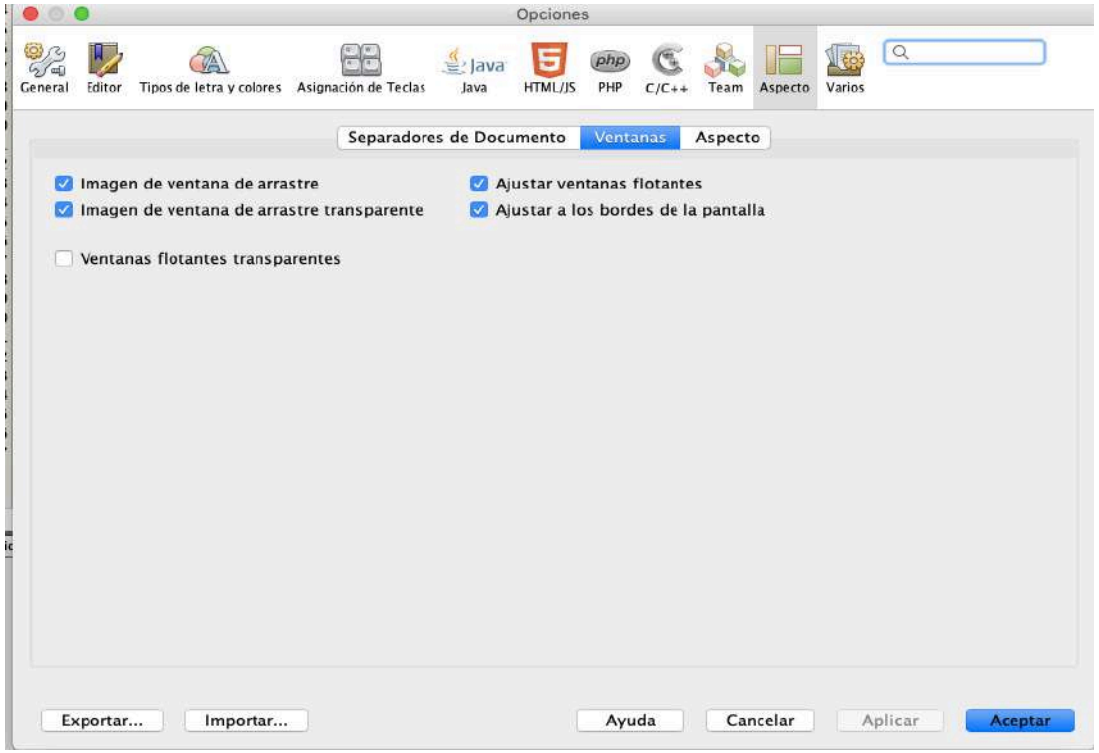
- **Ejecutar código**
  - Esta opción nos permite definir una nueva configuración de ejecución de código, elegir la clase principal, las carpetas de trabajo del proyecto y opciones de la máquina virtual.
  - Podríamos establecer diferentes configuraciones, se guardarían con un nombre y luego podríamos seleccionar la deseada a través del mismo.
- **Aplicación**
  - Podemos dar una descripción del proyecto, cambiarle el nombre, etc.
  - Es conveniente hacerlo, ya que el nombre de los nuevos proyectos se generan automáticamente por NetBeans al inicio de la sesión
  - También se pueden indicar unas propiedades específicas al inicio Web.

- **License Headers**
  - Temas de licencias por si en lugar de establecer una licencia global queremos establecer alguna licencia determinada.
- **Formato**
  - Aquí podemos personalizar aspectos globales del formato del código fuente en la aplicación.
  - Podemos personalizar las opciones solo para el proyecto actual o bien para todos los proyecto que estén basados en NetBeans a partir de ahora (utilizar opciones globales). Para ello pulsamos el botón de editar opciones globales.
- **Hints**
  - Son sugerencias que se van a mostrar a la hora de estar escribiendo el código. Es decir, indicaremos qué queremos que se nos muestre como sugerencia.

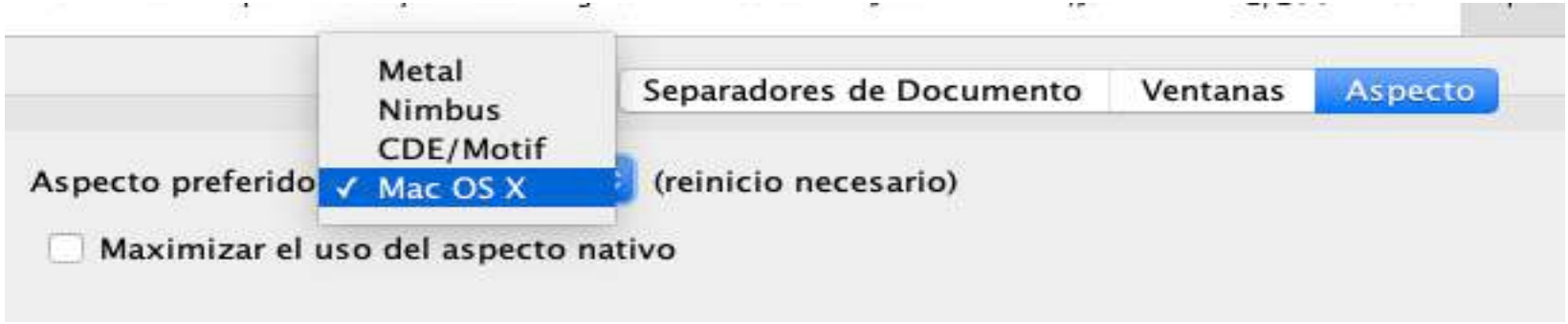

Opciones
General
Editor
Tipos de letra y colores
Asignación de Teclas
Java
HTML/JS
PHP
C/C++
Team
Aspecto
Varios
general
Folding
Formato
Finalización de código
Plantillas de Código
Sugerencias
Marcar ocurrencias
Lenguaje: Java
Categoría: Tabuladores y sangrados
Sobreescribir opciones globales
Aumentar tabuladores para espacios
Número de espacios en el san... 4
Tamaño de la pestaña: 8
Margen derecho: 80
Line Wrap: Apagar
Tamaño de la sangría de continuación: 8
Sangrado de la Etiqueta: 0
Sangrado Absoluto de Etiquetas
Sangrar los miembros de la clase de nivel superior
Sangrar Instrucciones Case en Switch
Vista previa:
public int number = 1;
private String letters[] = new String[]{"A", "E
public int meth(String text, int number) {
if (text == null) {
text = "a";
} else if (text.length() == 0) {
text = "empty";
} else {
number++;
}
label:
switch (number) {
case 1:
return meth("text", 22);
case 2:
return 20;
default:
return -1;
}
}
}
Exportar...
Importar...
Ayuda
Cancelar
Aplicar
Aceptar

5.2 CONFIGURACIÓN APARIENCIA

- Para poder cambiar el aspecto o configuración de la apariencia buscaremos en el menú de la barra de herramientas (en NetBeans las cosas no se pueden realizar de una única forma) Opciones Generales y tras pulsar en ella se elegirá la opción de Aspecto (en algunas versiones se llama Apariencia)



- Pulsando en la opción Aspecto inferior nos llevará a una ventana donde vamos a poder elegir otro aspecto preferido (dependiendo del sistema operativo pueden varias las opciones que te muestra)
- Una vez elegido podemos reiniciar nuestro NetBeans para que dicho efecto se lleve a cabo.



6. INSTALAR PLUGINS

- 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.

6.1 INSTALAR PLUGIN DISPONIBLE

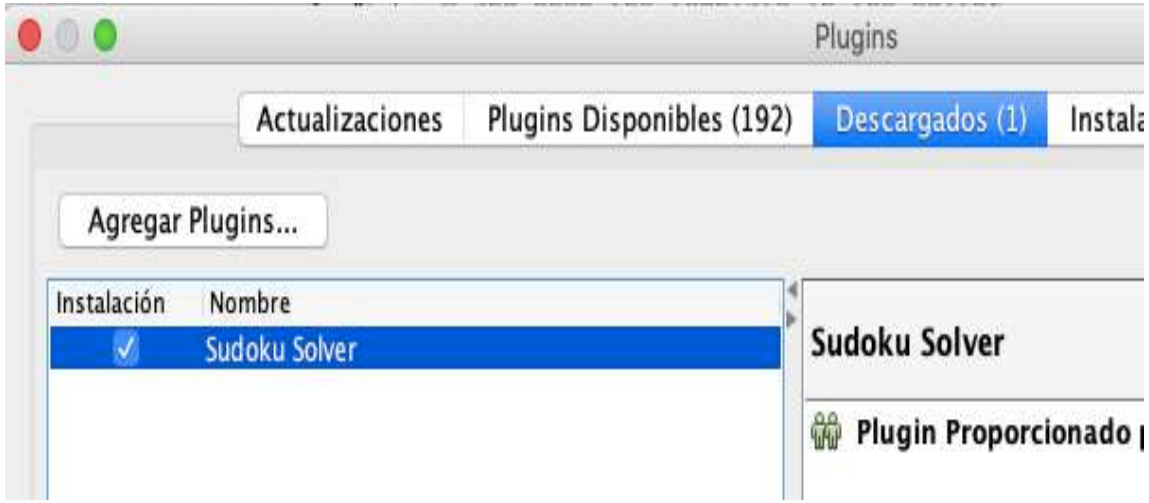
- Es posible completar nuestro IDE con nuevos plugin que pueden añadir nuevas funcionalidades o algún interés para nosotros.
- Por ejemplo, cuando vimos el apartado de los aspectos, es posible que ninguno de los mismos nos haya resultado lo suficientemente atractivo y queramos probar con algún otro que no viene instalado por defecto.
- Para instalar un nuevo plugin vamos a ir a la opción Herramientas -> Plugins. Dentro de los Plugins disponibles elegiremos la opción Dark Look And Feel Themes e instalar.

- El NetBeans se configurará automáticamente con uno de estos aspectos más oscuros que son ideales para trabajar por la noche.
- Además, si vamos al apartado de Aspecto podemos comprobar que dichos temas se han añadido por defecto a nuestro NetBeans.

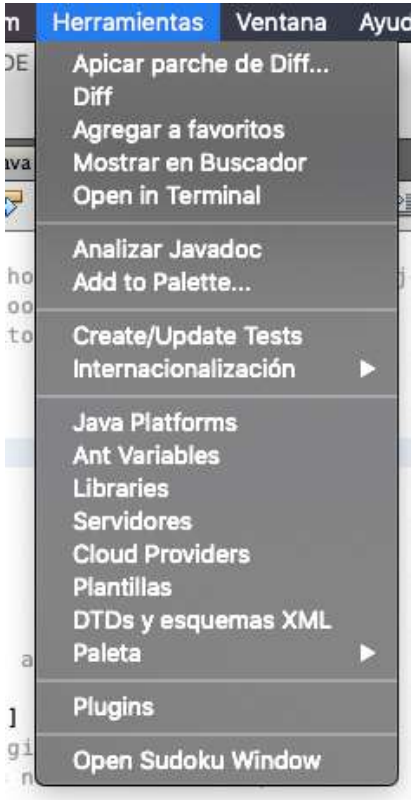
6.2 INSTALAR PLUGIN ON-LINE

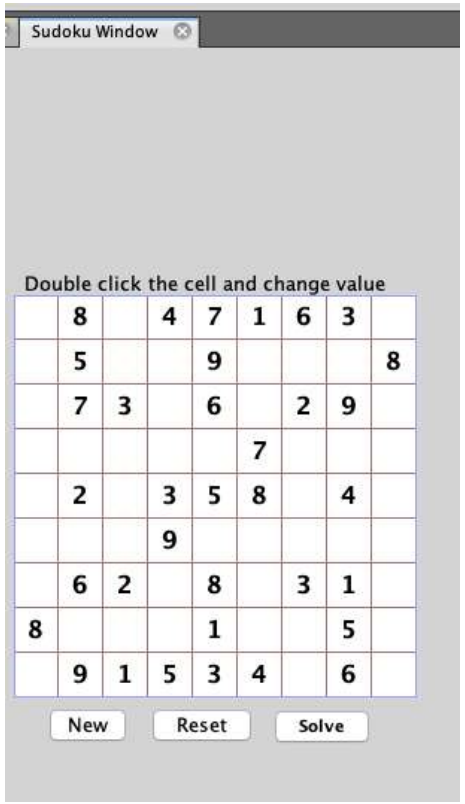
- No obstante, no siempre los plugins los vamos a instalar directamente desde NetBeans sino que es posible que en ocasiones dichos plugins deban descargarse.
- Existe un repositorio de plugins de NetBeans: http://plugins.netbeans.org/
- Vamos a buscar un plugin para jugar al sudoku desde nuestro IDE. 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. Elegimos la carpeta donde queramos que se guarde.

- Nos vamos ahora a la instalación de plugins pero a la opción de descargado y le damos a la opción de Agregar plugins.
- Buscamos el archivo anteriormente descargado y pulsaríamos instalar.



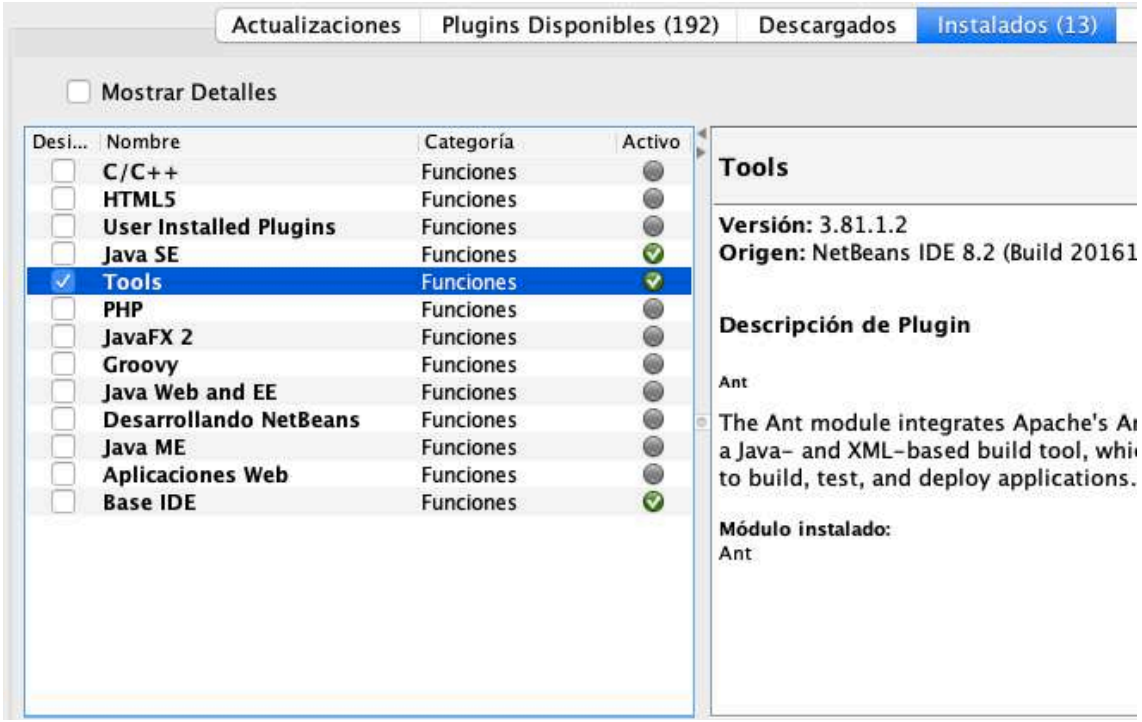
- Ya disponemos de una nueva opción en Herramientas para poder jugar un rato a nuestro Sudoku





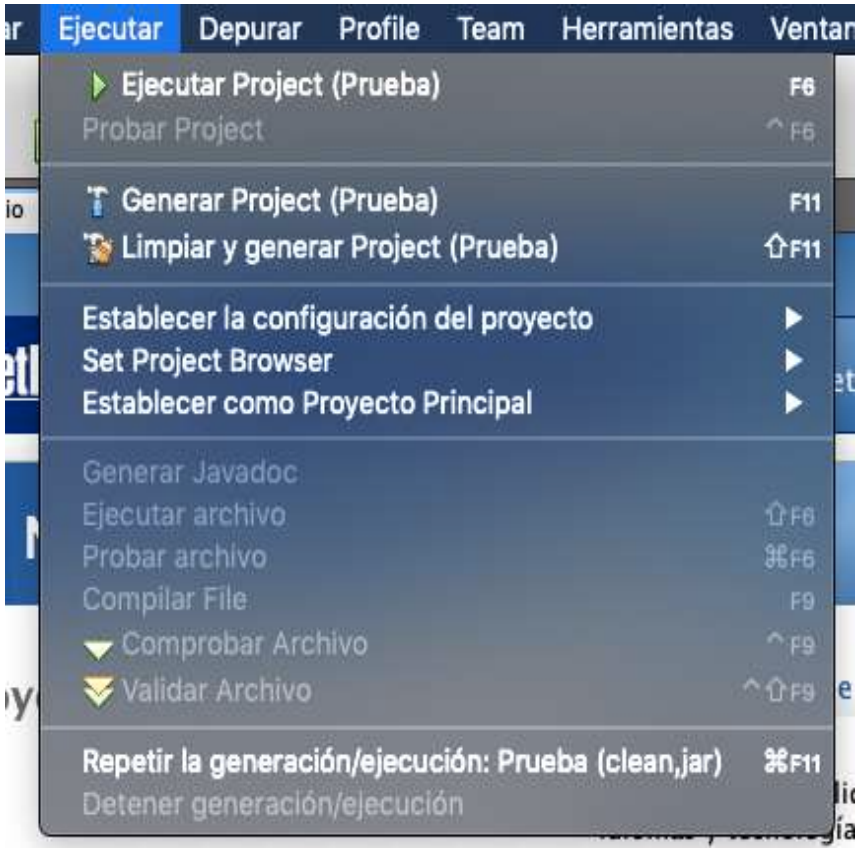
6.3 ELIMINAR PLUGIN

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

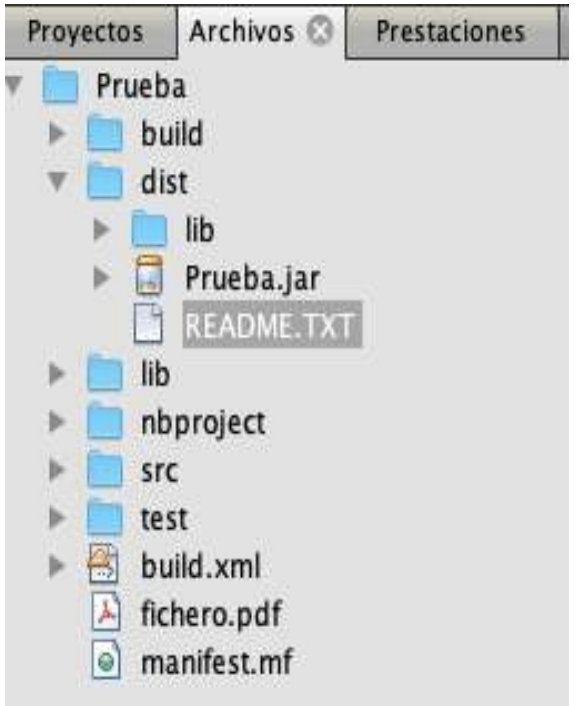

Actualizaciones
Plugins Disponibles (192)
Descargados
Instalados (13)
Mostrar Detalles
Desi...
Nombre
Categoría
Activo
C/C++
Funciones
HTML5
Funciones
User Installed Plugins
Funciones
Java SE
Funciones
Tools
Funciones
PHP
Funciones
JavaFX 2
Funciones
Groovy
Funciones
Java Web and EE
Funciones
Desarrollando NetBeans
Funciones
Java ME
Funciones
Aplicaciones Web
Funciones
Base IDE
Funciones
Tools
Versión: 3.81.1.2
Origen: NetBeans IDE 8.2 (Build 20161
Descripción de Plugin
Ant
The Ant module integrates Apache's A
a Java- and XML-based build tool, whi
to build, test, and deploy applications.
Módulo instalado:
Ant

7. PREPARAR LA DISTRIBUCIÓN

- Desde NetBeans podemos preparar el programa para su distribución e implantación en otras computadoras. Es una de las opciones más interesantes del entorno ya que nos facilita enormemente este cometido.
- Para hacerlo basta seleccionar el proyecto que deseamos preparar y después elegir Ejecutar -> Limpiar y generar Main proyect



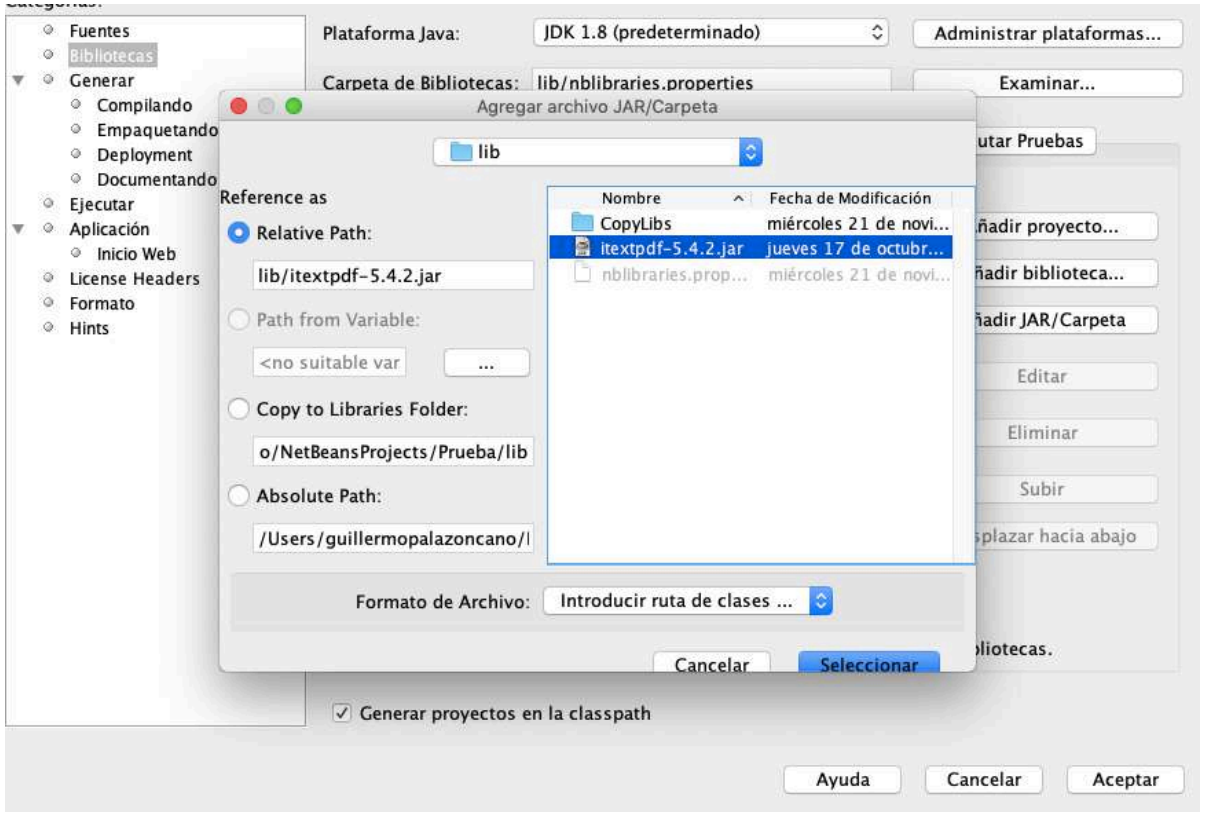
- Con esta instrucción NetBeans realiza la siguientes tareas
  - Crear una carpeta llamada dist en el directorio en el que se almacena el proyecto (este carpeta se puede visualizar en el panel Archivos)
  - Comprime todos los archivos compilados en un archivo de tipo jar dentro de la propia carpeta dist. Los archivo jar son archivos de tipo zip que contienen clases java y otros archivos necesarios para la correcta ejecución de una aplicación Java.
  - Crea un archivo llamado readme.txt con información sobre como ejecutar el proyecto (normalmente será java –jar nombreArchivoJar).



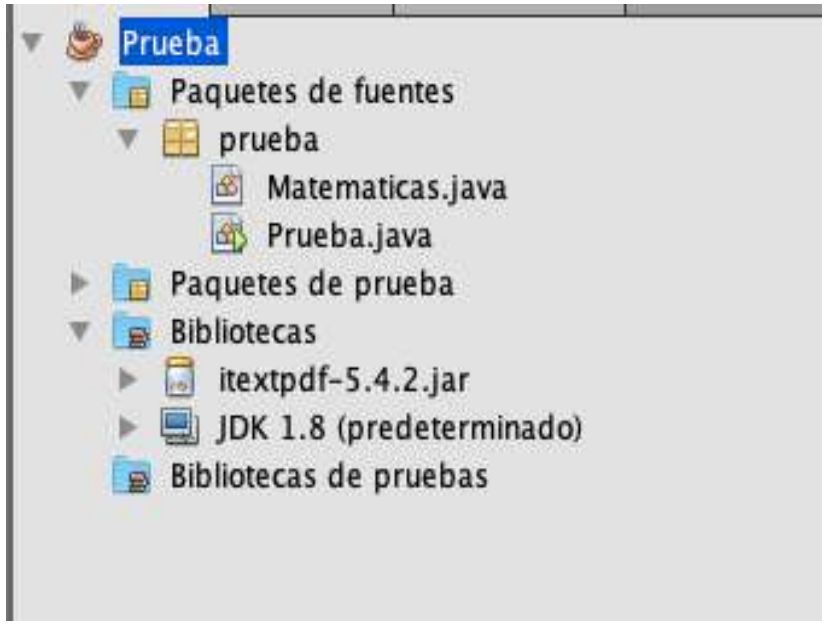
8. AÑADIR BIBLIOTECA PROYECTO

- Imagínate que nuestro famoso Hola Mundo queremos que se genere en un fichero PDF. Podemos aprovecharnos de alguna librería que ya está desarrollada y que de una manera muy sencilla nos permite generar archivos PDF.
- Para nuestro ejemplo vamos a aprovecharnos de la librería itextpdf en su versión 5.4.2 (tiene versiones posteriores pero para nuestro ejemplo nos sobra con esta)
- Podemos descargar esta librería desde el siguiente enlace: http://www.java2s.com/Code/Jar/i/Downloaditextpdf542jar.htm
- Luego habrá que descomprimir y coger el fichero .jar. ¿Dónde guardar el mismo? Se recomienda guardar el fichero dentro de nuestro proyecto, por lo que crearemos una carpeta lib dentro de nuestro proyecto y allí guardaremos el fichero .jar

- Para añadir esta biblioteca (existen otras opciones) nos vamos a las propiedades del proyecto, categoría Bibliotecas y le doy a Añadir JAR/Carpeta.
- Seleccionaremos la ruta y pulsaremos en el botón añadir.



- Podemos ver en nuestro menú lateral izquierda en la pestaña de Bibliotecas como aparece nuestra nueva Biblioteca que hemos incorporado.
- Una vez incorporada ya podríamos utilizarla dentro de nuestro código, para ello vamos a crear una nueva clase que se va a llamar PruebaPDF (en mayúsculas que es una clase)
- Aunque hayamos incorporado una biblioteca a nuestro proyecto, deberemos importarla



```java
package prueba;

import com.itextpdf.text.Document;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;
import java.io.FileOutputStream;

/**
 *
 * @author guillermopalazoncano
 */
public class PruebaPDF {
    public void generaHolaMundo(){
        try {
            String parrafo = "Hola Mundo";
            // Se crea el documento
            Document document = new Document();
            // Se crea el OutputStream para el fichero donde queremos dejar el pdf.
            FileOutputStream ficheroPdf = new FileOutputStream("fichero.pdf");
            PdfWriter.getInstance(document, ficheroPdf);

            document.open();
            document.add(new Paragraph(parrafo));
            document.close();
        }
        catch ( Exception e )
        {
            e.printStackTrace();
        }
    }
}
```

- El código del método sería el siguiente.

```
public void generaHolaMundo(){
    try {
        String parrafo = "Hola Mundo";
        // Se crea el documento
        Document document = new Document();
        // Se crea el OutputStream para el fichero donde queremos dejar el pdf.
        FileOutputStream ficheroPdf = new FileOutputStream("fichero.pdf");
        PdfWriter.getInstance(document, ficheroPdf);

        document.open();
        document.add(new Paragraph(parrafo));
        document.close();
    }
    catch ( Exception e )
    {
        e.printStackTrace();
    }
}
```

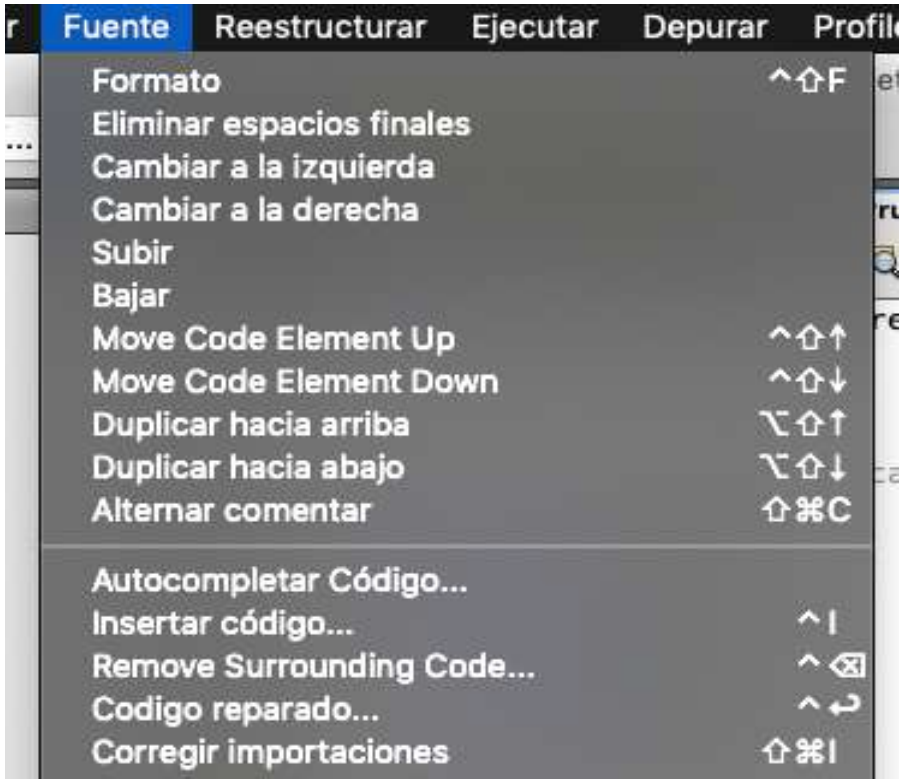
- Llamamos desde nuestro procedimiento main de la clase principal y lo ejecutamos.
- El fichero se generará (ya que en la clase PruebaPDF no hemos indicado ruta en la ruta del Proyecto)

```
public class Prueba {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        //Matematicas matematicas = new Matematicas();
        //matematicas.suma_fija();
        //matematicas.suma_variable();
        PruebaPDF pdf = new PruebaPDF();
        pdf.generaHolaMundo();
    }

}
```

9. AYUDAS A ESCRIBIR CÓDIGO

- Como la mayoría de IDEs actuales, NetBeans posee numerosas mejoras para escribir el código. Entre ellas:
  - Autocompletado del código. Funciona igual que en Eclipse, pulsando **Ctrl + Barra espaciadora**, se nos mostrará ayuda en un cuadro para ayudarnos a completar la expresión Java que estamos escribiendo. Es la opción de ayuda de código más interesante.
  - Plantillas de código. Abreviaturas que nos permiten escribir de golpe código habitual, ganando mucha rapidez. Para ello se escribe la abreviatura y después se pulsa el tabulador. Es posible ver estas plantillas dentro de Opciones, en la parte de Editor y plantillas de Código.
  - Marcado de errores y autocorrección. Se subraya de color rojo el código con errores en cuanto escribimos e incluso se corrigen automáticamente algunos errores. La tecla **Alt + Intro** permite examinar las circunstancias.

- Insercción de código. Pulsando el botón secundario sobre el código actual y eligiendo **Inserción del código** se nos facilitara la construcción de algunos elementos del código como constructores, métodos get y set,… Cuanto más conozcamos de Java, más interesante será esta opción.
- Dar formato al texto**.** Permite dar de nuevo formato al código para que tenga la apariencia más legible. Esta opción la encontramos en el menú Fuente -> Formato



- Buscar dentro de un fichero, buscar dentro del proyecto, etc. Muy interesante dentro de estas opciones es la opción “sustituir en proyectos” cuando has decido cambiar el uso de una función por otra en todo el proyecto.
- Muchas otras opciones interesantes que te permiten ir directamente a una línea del código así como otros accesos de teclado.
- Generar automáticamente el javadoc
- Cerrado automático de llaves, paréntesis, comillas,…
- Macros
- Administración de cláusulas import. Mediante las teclas Ctrl+Mayus+I Netbeans genera el código de tipo import necesario para que no haya errores en nuestro programa.