ENTORNOS DE DESARROLLO INTEGRADOS

Integrated Development Environment

Descripción breve

Veremos que es un entorno de desarrollo, que es necesario para su instalación y de que herramientas están compuestos algunos de ellos como NetBeans, Eclipse, Visual Studio y IntelliJ IDEA.

Javier García García
Javialmaden.c@gmail.com

INDICE

INDICE 1
1. ¿QUE ES UN ENTORNO DE DESARROLLO? 3
2. PRINCIPALES HERRAMIENTAS INTEGRADAS EN UN ENTORNO DE DESARROLLO 3
- Editor de texto: 3
- Un compilador: 3
- Un intérprete: 3
- Un depurador: 3
- Constructor de interfaz gráfica: 3
- Control de versiones: 3
- Panel de configuración: 3
- Plugins: 3

3. ESTUDIO DE ALGUNOS ENTORNOS DE DESARROLLO 4
3.1 **NETBEANS** 4
3.1.1 INSTALACION 5
3.1.2 EDITOR DE TEXTO 11
3.1.3 COMPILADOR 13
3.1.4 DEPURADOR 14
3.1.4.1 ESTABLECER PUNTOS DE RUPTURA 15
3.1.4.2 VENTANAS DEL DEPURADOR 16
3.1.4.3 LA VENTANA DE VARIABLES LOCALES 16
3.1.5 PLUGINS 17
3.1.6 CONTROL DE VERSIONES 20
3.1.7 PANEL DE CONFIGURACION 20
3.1.7.1 GENERAL 21
3.1.7.2 EDITOR 21
3.1.7.3 TIPOS DE LETRA Y COLORES 21
3.1.7.4 ASIGNACION DE LETRAS 21
3.1.7.5 JAVA 21
3.1.7.6 HTML/JS 21
3.1.7.7 TEAM 21
3.1.7.8 ASPECTO 21
3.1.7.9 VARIOS 21
3.1.8 REFACTORIZACION 22
3.2 **VISUAL STUDIO** 23
3.2.1 INSTALACION 24

3.2.2 EDITOR DE TEXTO 29
3.2.3 COMPILADOR 31
3.2.4 DEPURADOR 32
3.2.4.1 ESTABLECER PUNTOS DE RUPTURA 34
3.2.4.2 VENTANAS DEL DEPURADOR 35
3.2.5 PLUGINS 36
3.2.6 PANEL DE CONFIGURACION 37
3.3 **ECLIPSE** 38
3.1.3 INSTALACION 39
3.3.2 EDITOR DE TEXTO 45
3.3.3 COMPILADOR 46
3.3.4 DEPURADOR 47
3.3.4.1 ESTABLECER PUNTOS DE RUPTURA 48
3.3.5 PLUGINS 48
3.3.6 CONTROL DE VERSIONES 50
3.3.7 PANEL DE CONFIGURACION 51
3.3.8 REFACTORIZACION 51
3.4 **INTELLIJ IDEA** 53
3.4.1 INSTALACION 54
3.4.2 EDITOR DE TEXTO 60
3.4.3 COMPILADOR 62
3.4.4 DEPURADOR 64
3.4.4.1 ESTABLECER PUNTOS DE RUPTURA 65
3.4.4.2 LA VENTANA DE VARIABLES LOCALES 65
3.4.5 PLUGINS 66
3.4.6 CONTROL DE VERSIONES 67
3.4.7 PANEL DE CONFIGURACION 68
3.4.8 REFACTORIZACION 69
4. CONCLUSION 70
5. CLASIFICACION DE LOS ENTORNOS 71

1. ¿QUE ES UN ENTORNO DE DESARROLLO?

Un entorno de desarrollo o IDE (Integrated Development Environment) es una aplicación informática que está compuesta por un conjunto de herramientas que facilitan la tarea al programador de forma que ayuda a desarrollar aplicaciones con más rapidez.

Los entornos de desarrollo son utilizados en la fase de codificación del ciclo de vida de software independientemente del modelo que se utilice.

Existen entornos de desarrollo para un solo lenguaje o para múltiples lenguajes.

2. PRINCIPALES HERRAMIENTAS INTEGRADAS EN UN ENTORNO DE DESARROLLO.

Un entorno de desarrollo está compuesto por distintas herramientas que nos ayudarán durante todo el proceso de creación del código fuente. Algunas de estas son:

- **Editor de texto:** Es la herramienta que nos permite escribir el código fuente del programa, ofrece funciones para ayudarnos según estamos desarrollando nuestro código, como son la de resaltar de forma visual palabras reservadas según el lenguaje de programación que estemos utilizando, incluye un analizador léxico que nos corrige en caso de haber escrito mal alguna palabra y un analizador sintáctico que nos informa de si la estructura está bien realizada.
- **Un compilador:** Es la herramienta encargada de traducir el código fuente que hemos escrito en lenguaje que sea legible para las máquinas (código máquina).
- **Un intérprete:** Un intérprete es algo similar al compilador, es decir, su misión es la de traducir en código fuente en código máquina con la diferencia de que el intérprete lo traduce línea a línea según se va ejecutando. Es más lento que el compilador.
- **Un depurador:** Es una herramienta que nos permite probar y eliminar posibles errores en un programa. Nos permite ejecutar el código línea a línea avanzando o retrasando la ejecución a nuestro gusto, detener el programa en los puntos de rupturas que el desarrollador desee, ver los datos de variables en el momento de ejecución, etc.
- **Constructor de interfaz gráfica:** Es la herramienta que permite crear y diseñar las interfaces gráficas de forma más intuitiva con las cuales habrá interacción entre la aplicación y el usuario.
- **Control de versiones:** Es una herramienta que permite al desarrollador controlar los distintos cambios que sufre el código de una aplicación a lo largo de su construcción.
- **Panel de configuración:** Todo entorno de desarrollo dispone de un panel en el que cambiar la configuración del mismo, podemos modificar el aspecto visual, conexiones de red, asignación de teclas, entre muchas otras funciones.
- **Plugins:** son complemento que se relacionan con otras herramientas para agregarle una nueva función y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal.

3. ESTUDIO DE ALGUNOS ENTORNOS DE DESARROLLO

3.1 NETBEANS

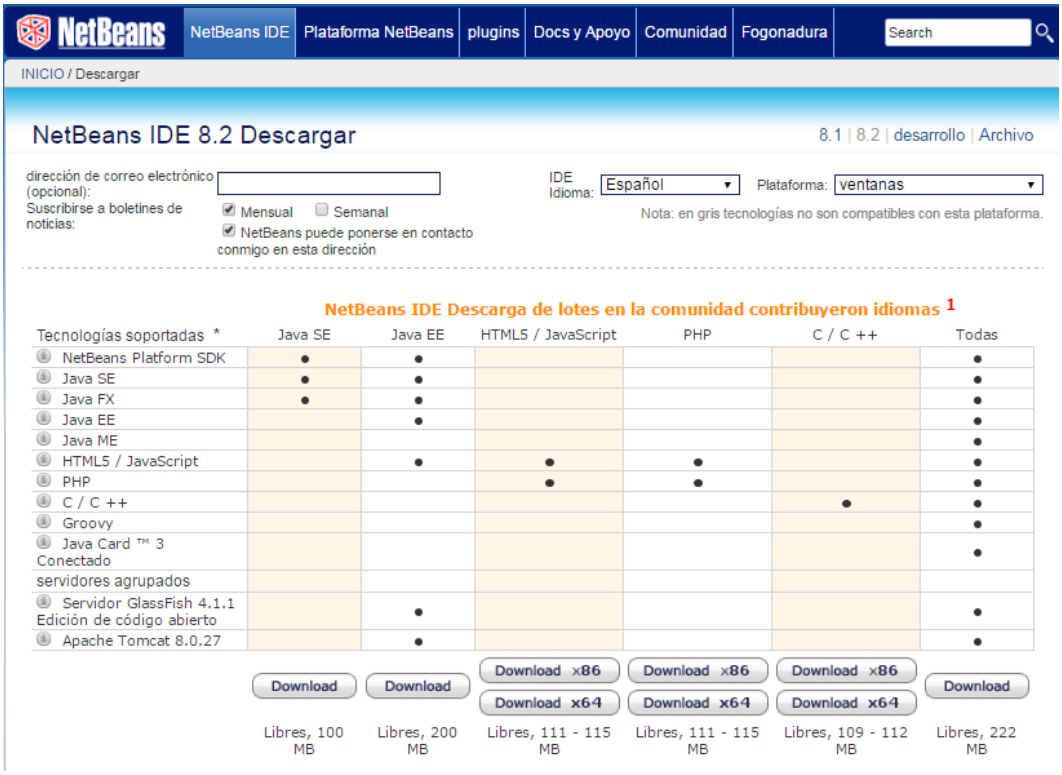
NetBeans es un entorno de desarrollo integrado hecho en java y principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation).

NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

Su última versión es la 8.2 que a su vez se divide en diferentes versiones según su cantidad de módulos, puedes encontrarla en su página oficial https://netbeans.org/downloads/index.html.

NetBeans está disponible para Windows, Linux y Mac OS, aunque según que plataforma incorpora más o menos funcionalidades.



Para la instalación de NetBeans es necesario instalar la última versión del JDK de java.

8u111 Java SE Development Kit

Debe aceptar el Acuerdo de licencia del código binario de Oracle para Java SE para descargar este software.

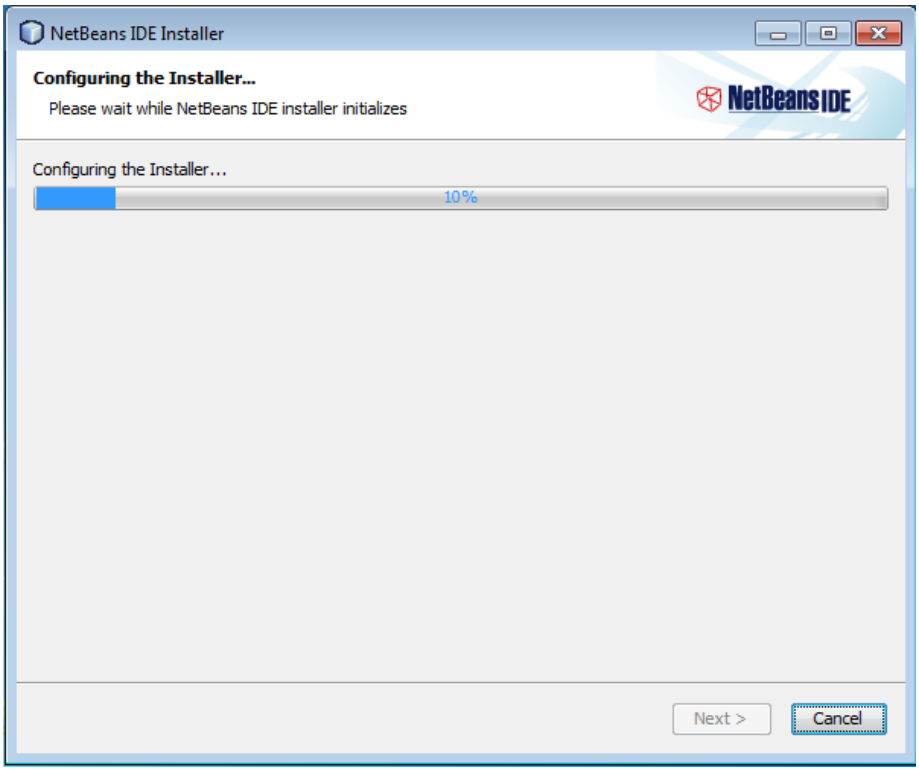
○ Aceptar el acuerdo de licencia ◉ Negar acuerdo de licencia

Producto / Descripción del archivo	Tamaño del archivo	Descargar
Linux ARM 32 Float duro ABI	77.78 MB	JDK-8u111-linux-ARM32-VFP-hflt.tar.gz
Linux ARM 64 Float duro ABI	74,73 MB	JDK-8u111-linux-arm64-VFP-hflt.tar.gz
Linux x86	160.35 MB	JDK-8u111-linux-i586.rpm
Linux x86	175.04 MB	JDK-8u111-linux-i586.tar.gz
Linux x64	158.35 MB	JDK-8u111-linux-x64.rpm
Linux x64	173.04 MB	JDK-8u111-linux-x64.tar.gz
Mac OS X	227.39 MB	JDK-8u111-MacOSX-x64.dmg
Solaris SPARC de 64 bits	131.92 MB	JDK-8u111-solaris-sparcv9.tar.Z
Solaris SPARC de 64 bits	93.02 MB	JDK-8u111-solaris-sparcv9.tar.gz
Solaris x64	140.38 MB	JDK-8u111-solaris-x64.tar.Z
Solaris x64	96.82 MB	JDK-8u111-solaris-x64.tar.gz
x86 de windows	189.22 MB	jdk-8u111-windows-i586.exe
64 bits de windows	194.64 MB	jdk-8u111-windows-x64.exe

3.1.1 INSTALACION

A continuación, veremos el proceso de instalación de NetBeans 8.2 de 64 bit en español paso a paso.

Una vez descargado, hacemos doble clic sobre el paquete y se nos abrirá la siguiente ventana, donde prepara todo lo necesario para su correcta instalación.

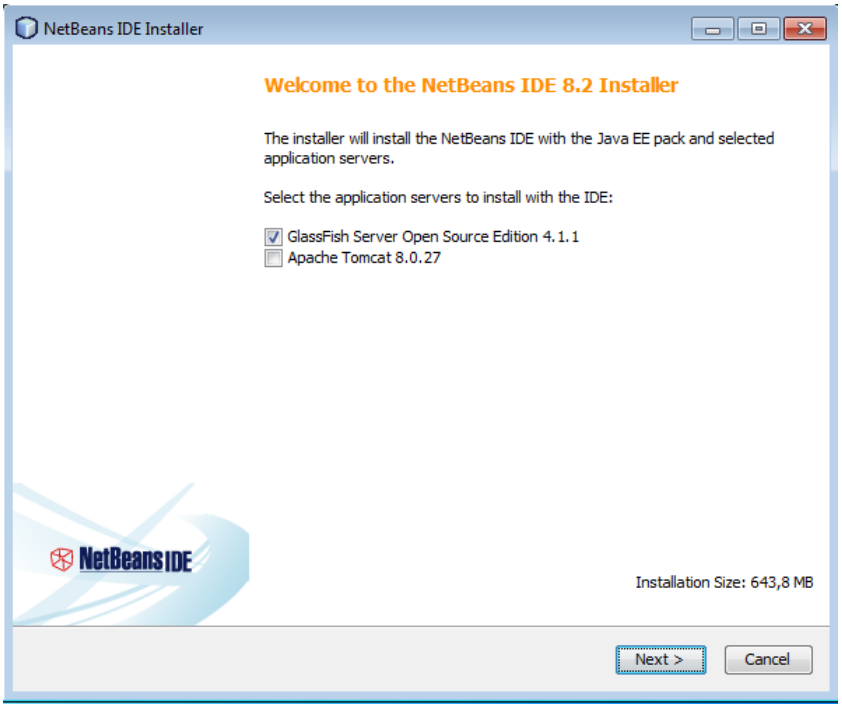


En la siguiente ventana nos pedirá si deseamos instalar GlassFish Server o apache Tomcat.

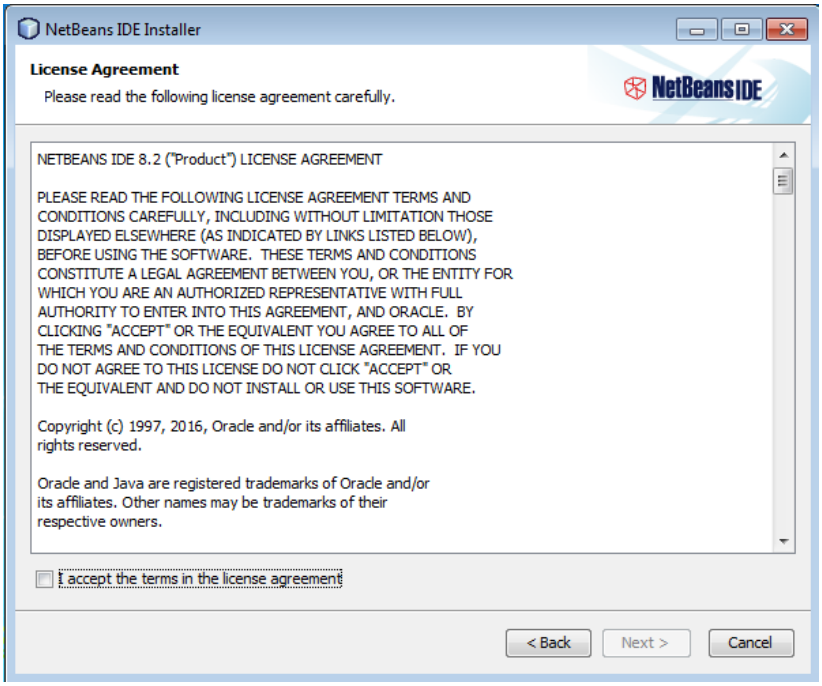
GlassFish es un servidor de aplicaciones que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación.

Apache Tomcat implementa las especificaciones de los servlets (clase en el lenguaje de programación Java) y de JavaServer Pages (JSP).

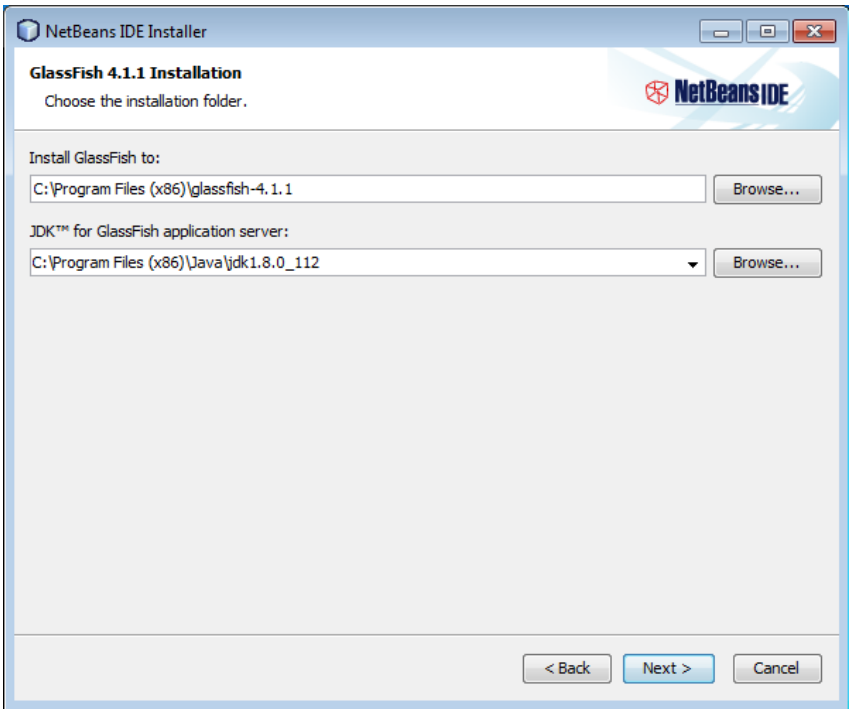
Nosotros dejaremos los valores que trae por defecto.



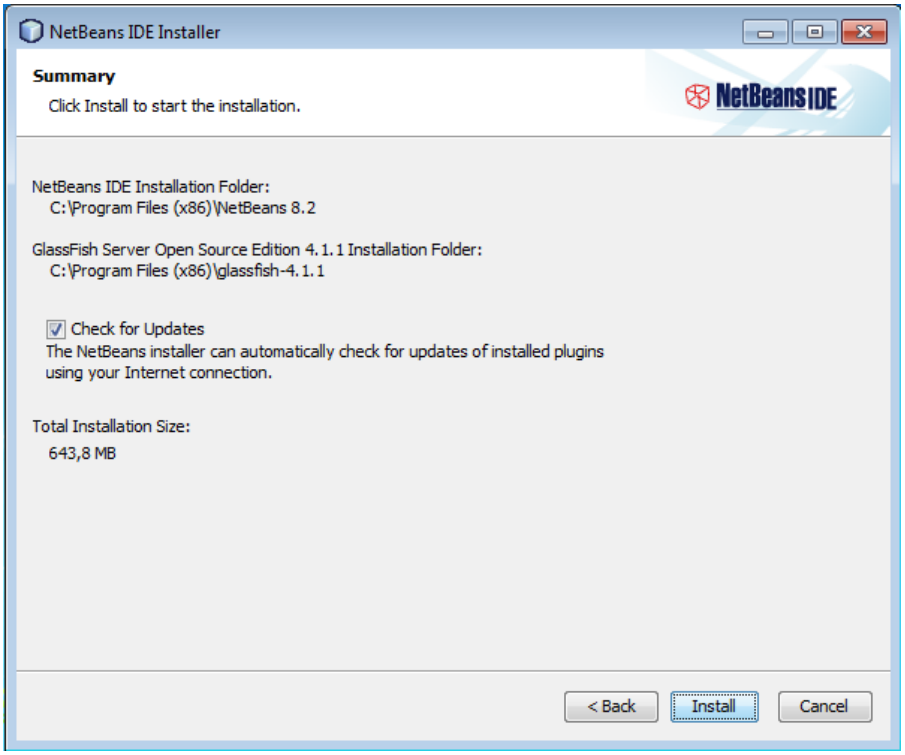
A continuación, debemos aceptar los términos de la licencia (si estamos conformes con ellos).

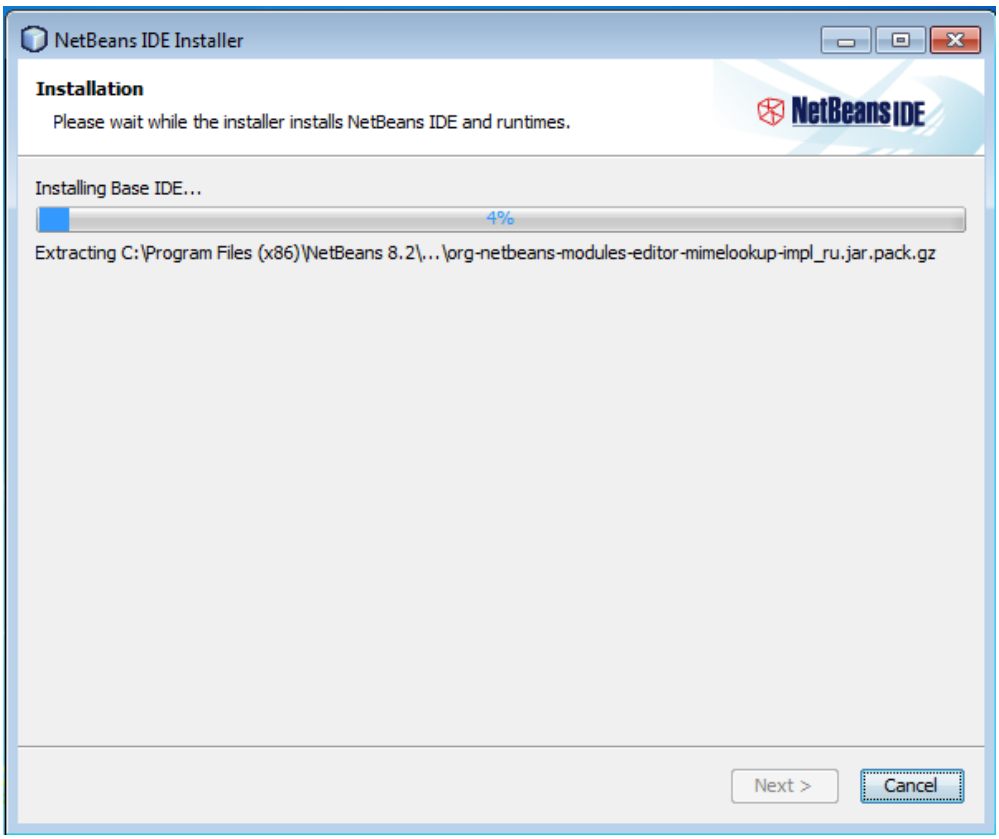


Posteriormente debemos indicar la ubicación de donde instalar el servidor GlassFish, nosotros lo dejamos en la ubicación por defecto.

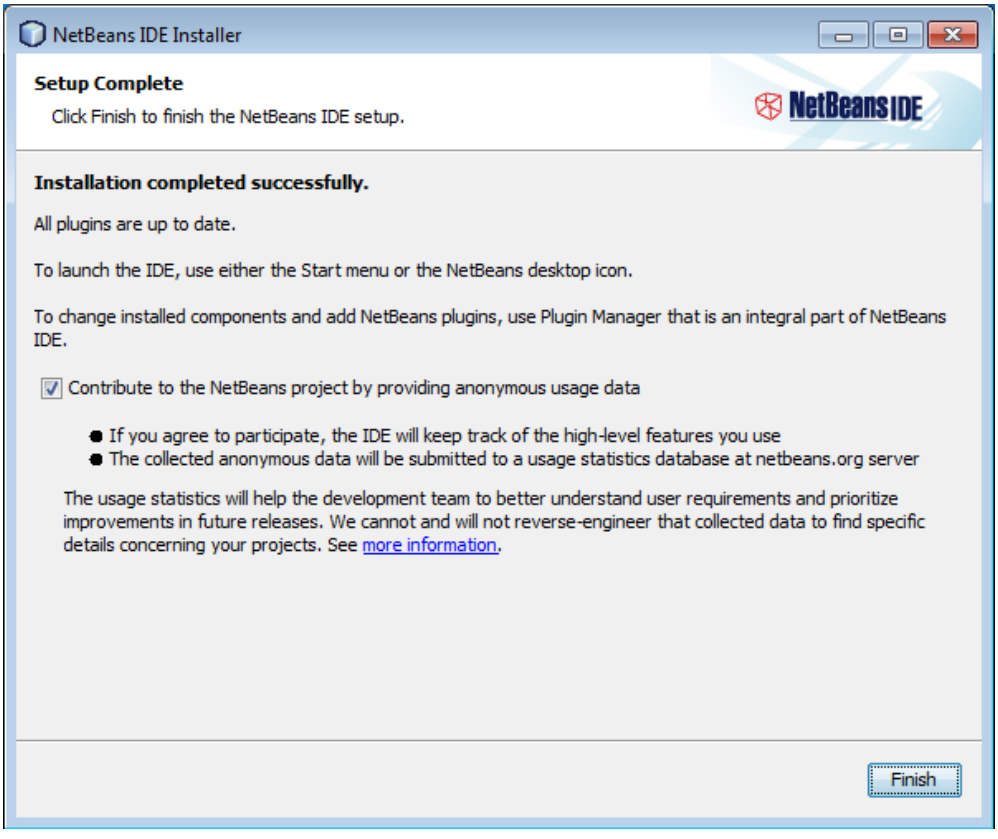


Por último, antes de instalar, nos muestra un resumen y nos pregunta si queremos comprobar si hay nuevas actualizaciones.

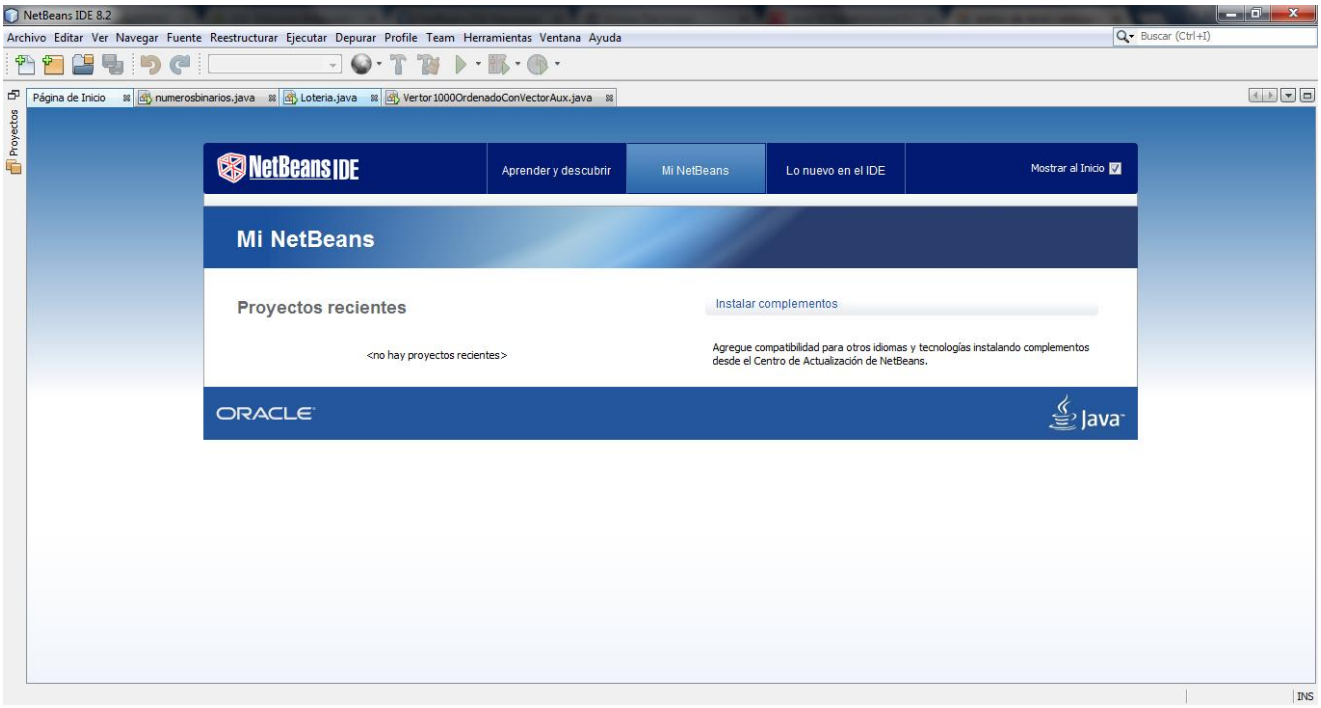




Una vez finalizada la instalación, tenemos la opción de contribuir con el proyecto NetBeans, marcando la casilla correspondiente.

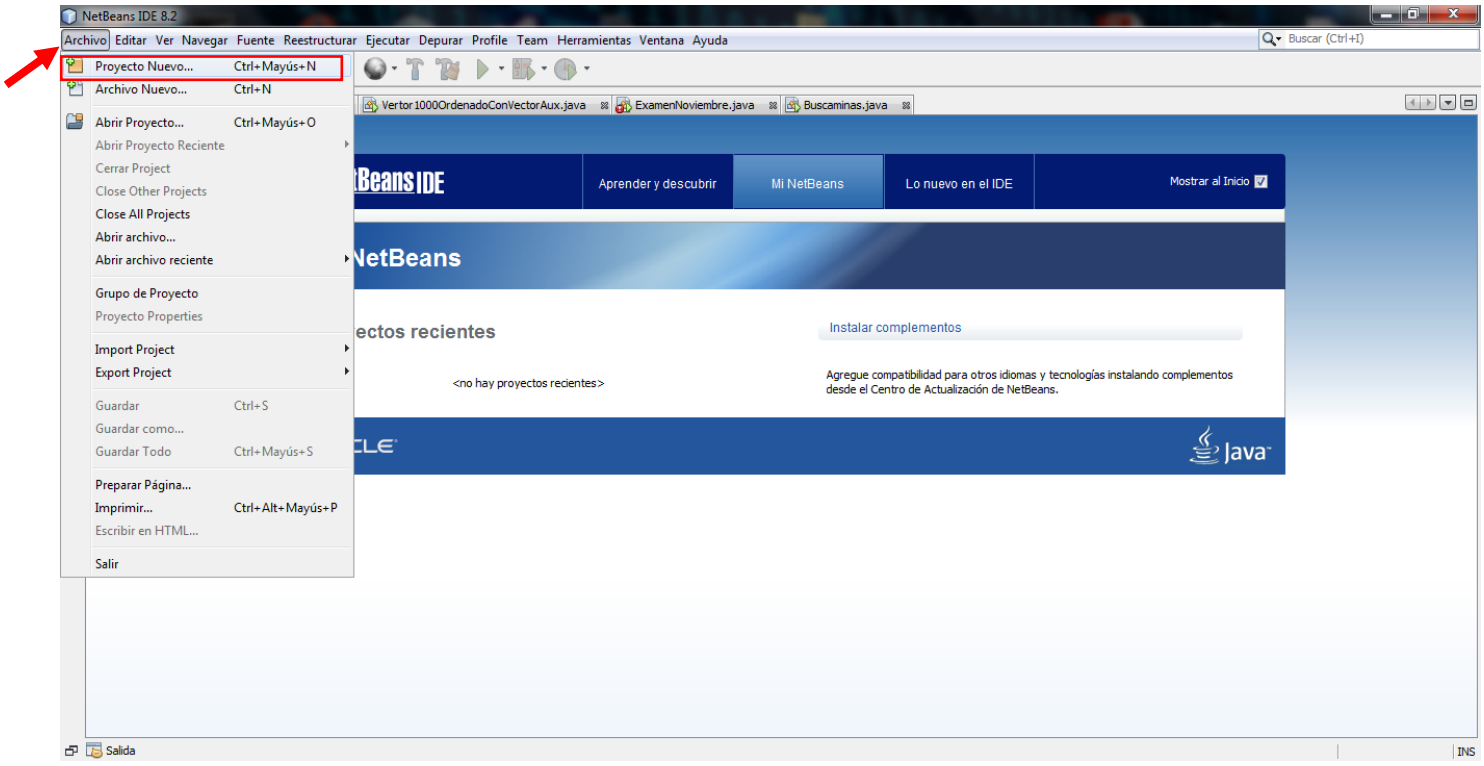


Y aquí lo tenemos, la pantalla principal de NetBeans.

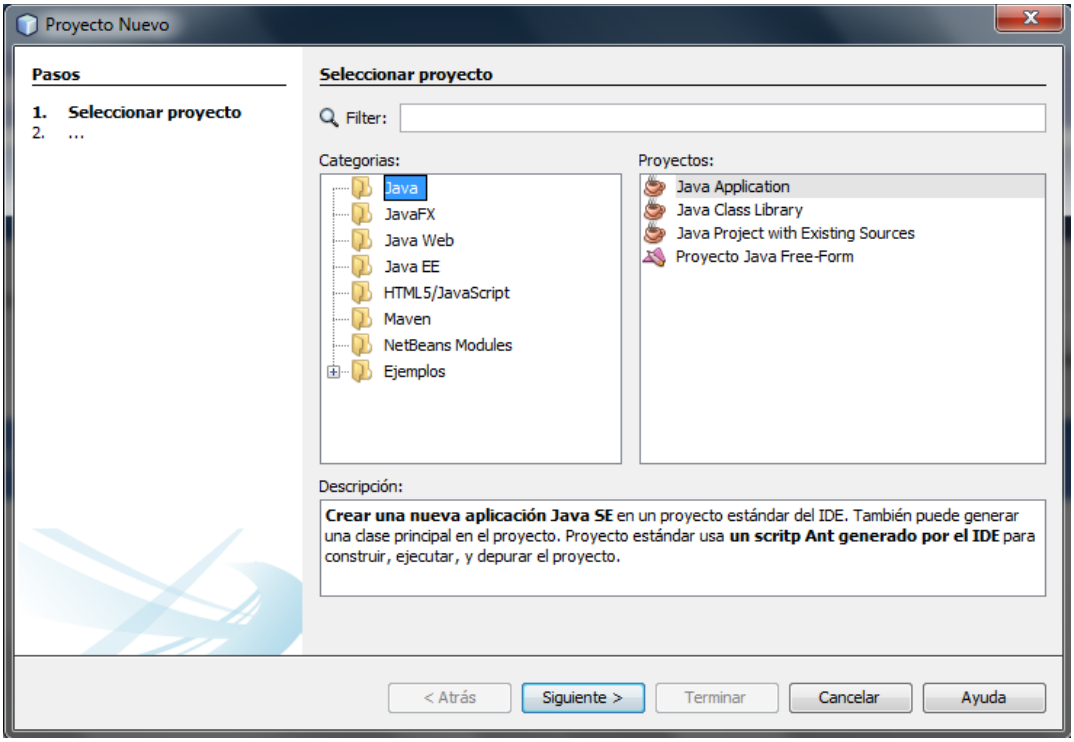


Ahora vamos a crear un nuevo proyecto para ver algunas de sus herramientas.

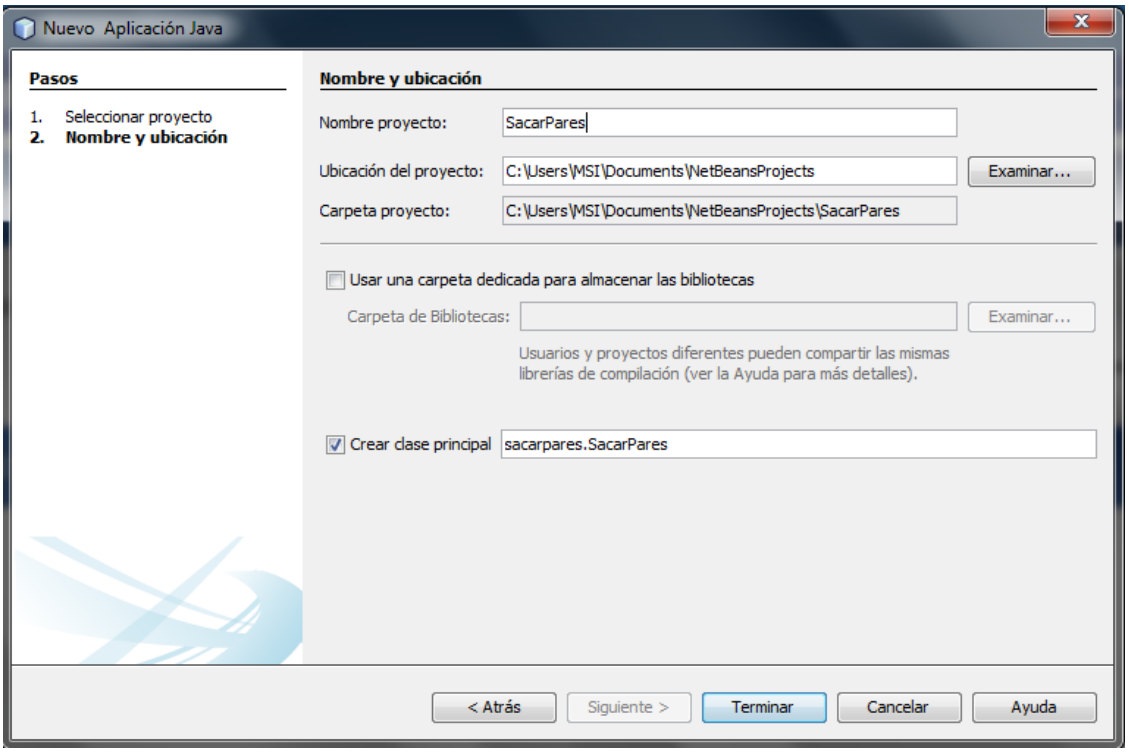
Para iniciar un nuevo proyecto iremos a la pestaña **Archivo → Proyecto Nuevo**



Se nos abrirá la siguiente ventana en la cual debemos de indicarle que se trata de una aplicación en lenguaje java.

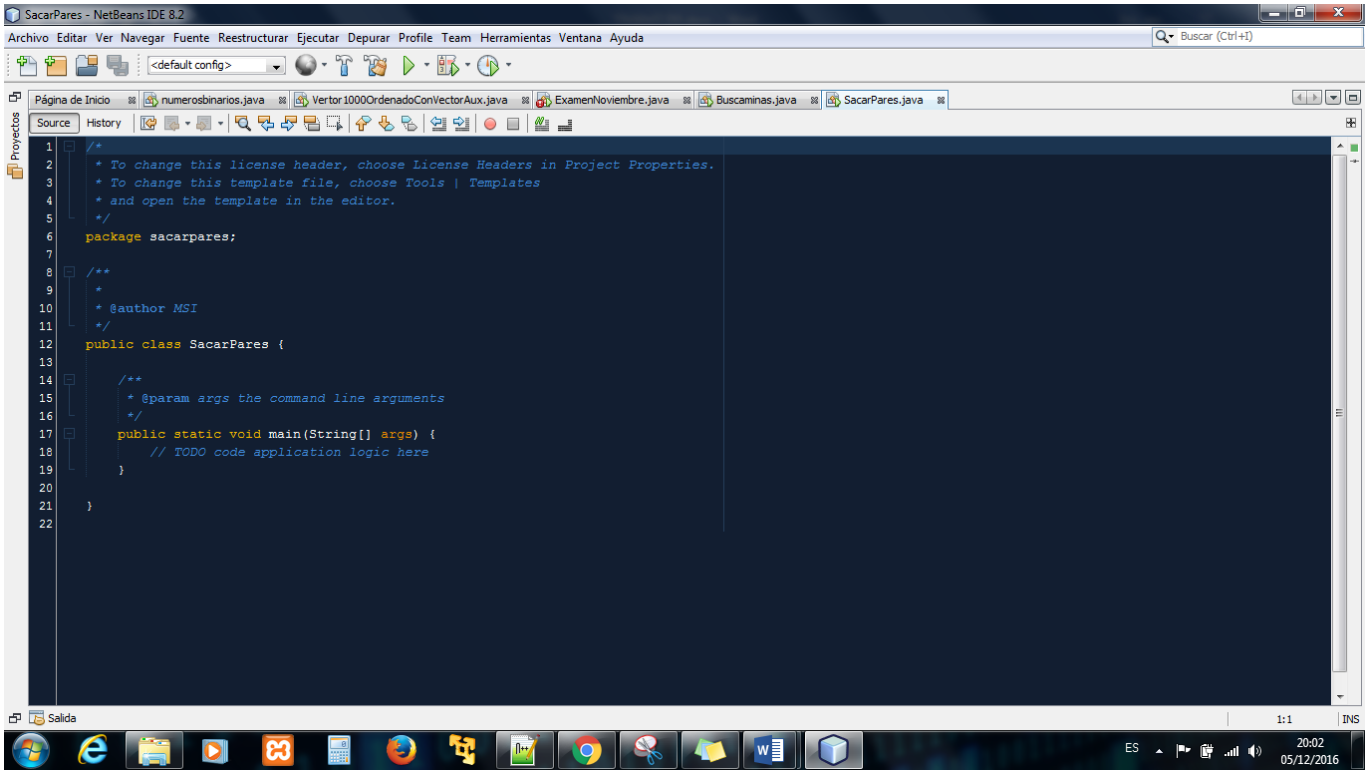


A continuación, en la siguiente ventana introduciremos el nombre del proyecto y la ubicación del mismo, teniendo en cuenta que está marcada la casilla de ***crear la clase principal*** de nuestro programa, de tal forma que nos ahorramos trabajo.

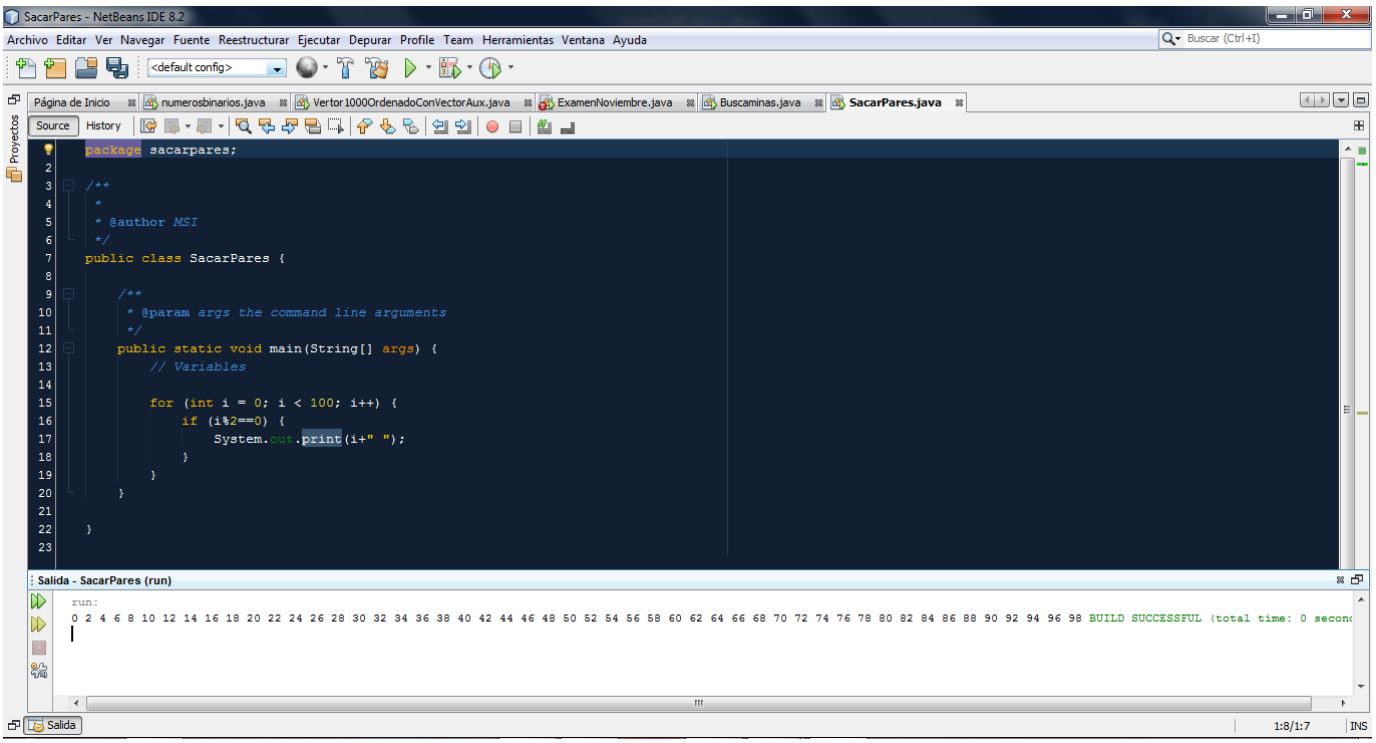


3.1.2 EDITOR DE TEXTO

Ya tenemos aquí nuestro editor de texto, que por defecto es una plantilla con la clase **main** ya construida.



A continuación, debemos escribir nuestro programa. El editor de texto nos ayuda según vamos escribiendo, irá marcando con un subrayado rojo los posibles errores de sintaxis y posibles errores léxicos.

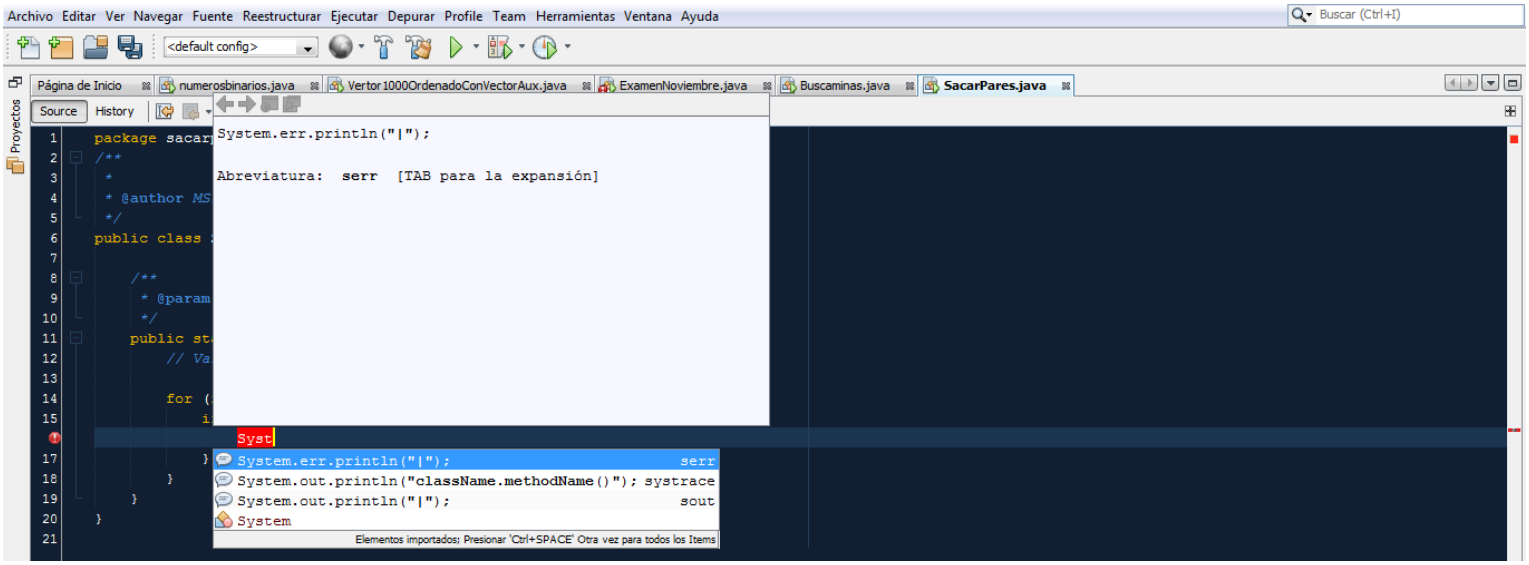


También podemos distinguir que nos muestra de forma resaltada las palabras reservadas del lenguaje.

```
15              for (int i = 0; i < 100; i++) {
16                  if (i%2==0) {
17                      System.out.print(i+" ");
18                  }
19              }
```

```
15              for (int i = 0; i < 100; i++) {
16                  if (i%2==0) {
                        Systemm.out.print(i+" ");
18                  }
19              }
```

El error descrito anteriormente es difícil de cometer puesto que como podemos ver en la imagen inferior, el propio editor nos muestra las posibles opciones disponibles según vamos escribiendo.



Aquí podemos observar cómo quedaría el código final de un algoritmo que saca los números pares entre 1 y 100.

```
1   package sacarpares;
2   /**
3    *
4    * @author MSI
5    */
6   public class SacarPares {
7
8       /**
9        * @param args the command line arguments
10       */
11      public static void main(String[] args) {
12          // Variables
13
14          for (int i=1;i<100; i++) {
15              if (i%2==0) {
16                  System.out.println(i+" ");
17              }
18          }
19      }
20  }
```

3.1.3 COMPILADOR

Para ejecutar el programa debemos compilar nuestro código, es decir, transformar nuestro código fuente en un código binario o código máquina, y para ello, tenemos varias formas de hacerlo, a través de los siguientes botones según cuál sea nuestro propósito.

El botón con el icono del martillo, también llamado constructor: es el encargado de compilar nuestro programa y crear el archivo .jar que podremos ejecutar en cualquier otra máquina que tenga instalada la máquina virtual de java

El botón con el icono del martillo y la escoba: es exactamente igual que el anterior, con la diferencia de que antes de compilar realiza una limpieza del buffer.

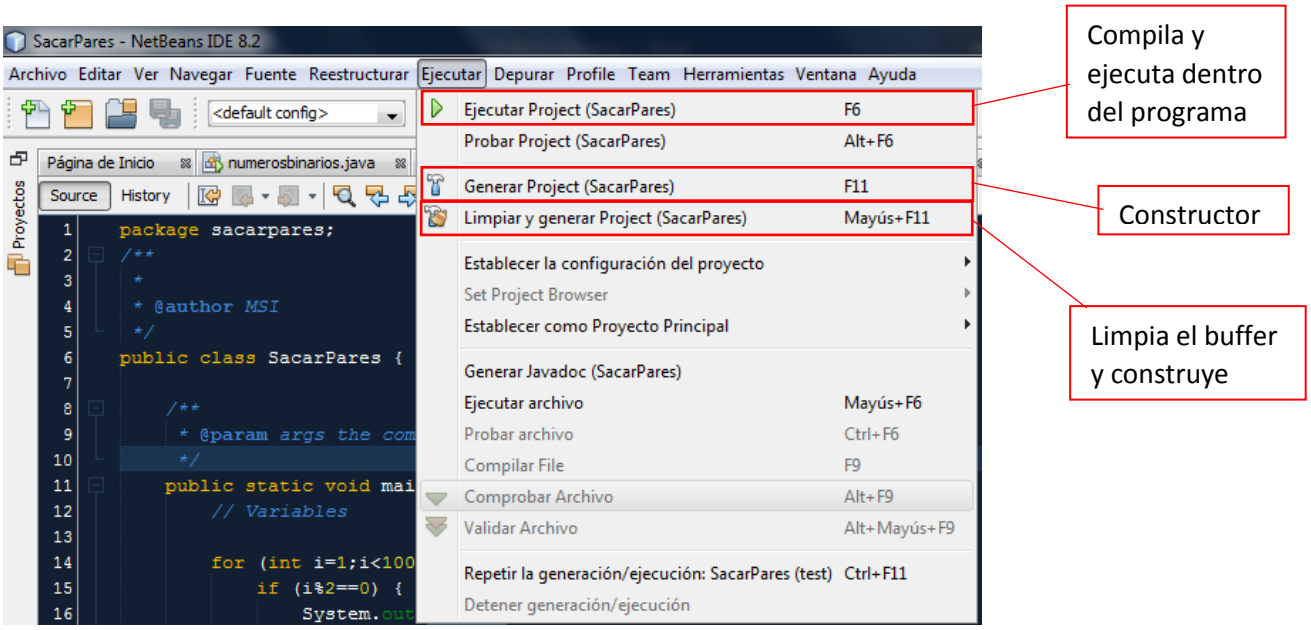
El botón con el icono del play: sirve para compilar y ejecutar el programa dentro de nuestro entorno y realizar la salida de datos por la salida estándar de forma que podamos ver el resultado.

El botón del reloj con el símbolo del play, llamado profile: es una herramienta de perfilado que permite a los desarrolladores que sean más productivos en la solución de problemas de rendimiento y de memoria.

También es posible el iniciar la compilación con una serie de atajos del teclado como son:

F6	Constructor
F11	Limpiar y compilar
Alt+F6	Probar proyecto
Ctrl+F2	Profile

Otra forma de ejecutar, compilar, depurar o ver la optimización de nuestro algoritmo, es a través de las pestañas de la barra de menús.



3.1.4 DEPURADOR

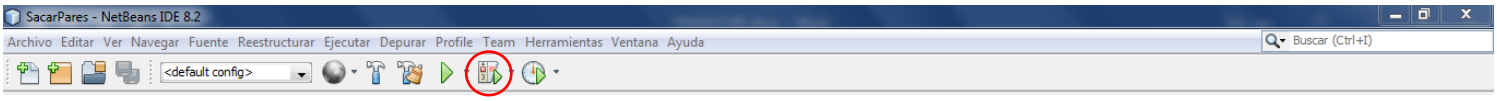
El depurador o debugger nos permite ejecutar el código fuente paso a paso para identificar posibles errores de forma más sencilla y rápida, facilitándonos su búsqueda y obteniendo una forma más eficiente de desarrollo.

Para ello, nos permite utilizar puntos de ruptura o breakpoint para detener la ejecución del programa y poder observar el estado de las variables, su valor, e incluso modificarlo sobre la marcha y continuar la ejecución.

Hay varias formas de iniciar la ejecución del programa en modo depuración:

Podemos hacerlo mediante la pestaña **Depurar → debug proyect**

Podemos hacerlo con un atajo de teclado (**Ctrl. + F5**) o pulsando sobre el botón:



El programa se ejecuta hasta llegar al primer punto de ruptura. Si no han establecido dichos puntos, el programa se ejecutará normalmente hasta el final.

Otra forma de depurar el código es **Depurar → Ejecutar** hasta el cursor o con el atajo **F4**

El depurador empieza e ejecutar el programa hasta la instrucción donde se encuentra el cursor.

También es posible la depuración paso a paso desde **Depurar → Paso a paso** o con el atajo F7

Comienza la depuración desde la primera línea del método main. El depurador se detiene esperando que decidamos el modo de depuración.

Una vez iniciada la depuración, el depurador se detiene sobre la siguiente línea de código que se va a ejecutar y esta aparece en verde, con una flecha verde a su izquierda:

```
11      public static void main(String[] args) {
12          // Variables
13
14          for (int i=1;i<100; i++) {
15              if (i%2==0) {
16                  System.out.println(i+" ");
17              }
18          }
19      }
20  }
```

Para el uso del depurador manejamos distintas opciones a través de botones:

	**Step Over (F8)** Ejecuta una línea de código. Si la instrucción es una llamada a un método, ejecuta el método sin entrar dentro del código del método.
	**Step Into (F7)** Ejecuta una línea de código. Si la instrucción es una llamada a un método, salta al método y continúa la ejecución por la primera línea del método.
	**Step Out (Ctrl + F7)** Ejecuta una línea de código. Si la línea de código actual se encuentra dentro de un método, se ejecutarán todas las instrucciones que queden del método y se vuelve a la instrucción desde la que se llamó al método.
	**Run to Cursor (F4)** Se ejecuta el programa hasta la instrucción donde se encuentra el cursor.
	**Continue (F5)** La ejecución del programa continúa hasta el siguiente punto de ruptura. Si no existe un punto de ruptura se ejecuta hasta el final.
	**Finish Debugger Session (Mayúsculas + F5).** Termina la depuración del programa.

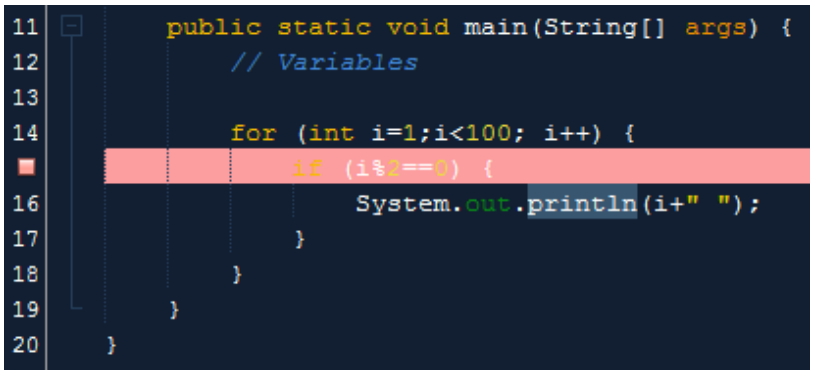
3.1.4.1 ESTABLECER PUNTOS DE RUPTURA

Un punto de ruptura es una marca que indica al depurador que debe detenerse cuando la ejecución del programa llegue a ella.

Cuando el programa se detiene podemos:

- Examinar los valores actuales de las variables.
- Continuar la depuración línea a línea del programa.

Para fijar un punto de ruptura simplemente pulsamos sobre el número de línea donde se desea colocar y ésta queda resaltada en color rojo con una marca del mismo color en el margen izquierdo.

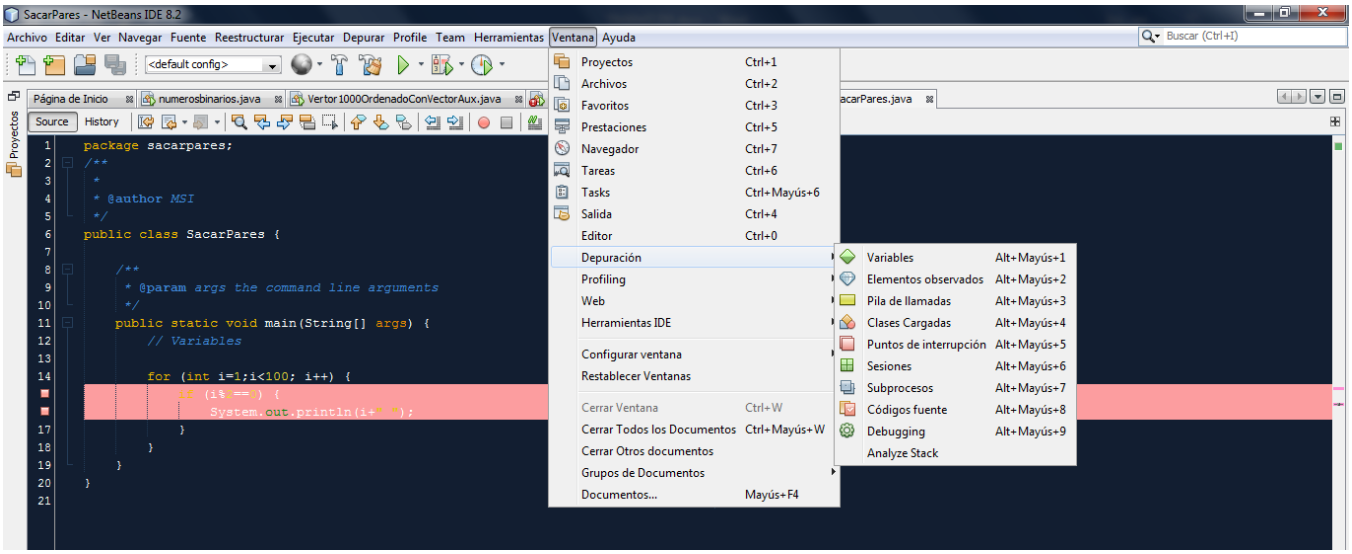


Para eliminar un punto de ruptura se pulsa sobre el cuadrado rojo.

3.1.4.2 VENTANAS DEL DEPURADOR

Para el proceso de depuración se utilizan distintas ventanas situadas bajo la ventana del editor de código. Algunas aparecen automáticamente.

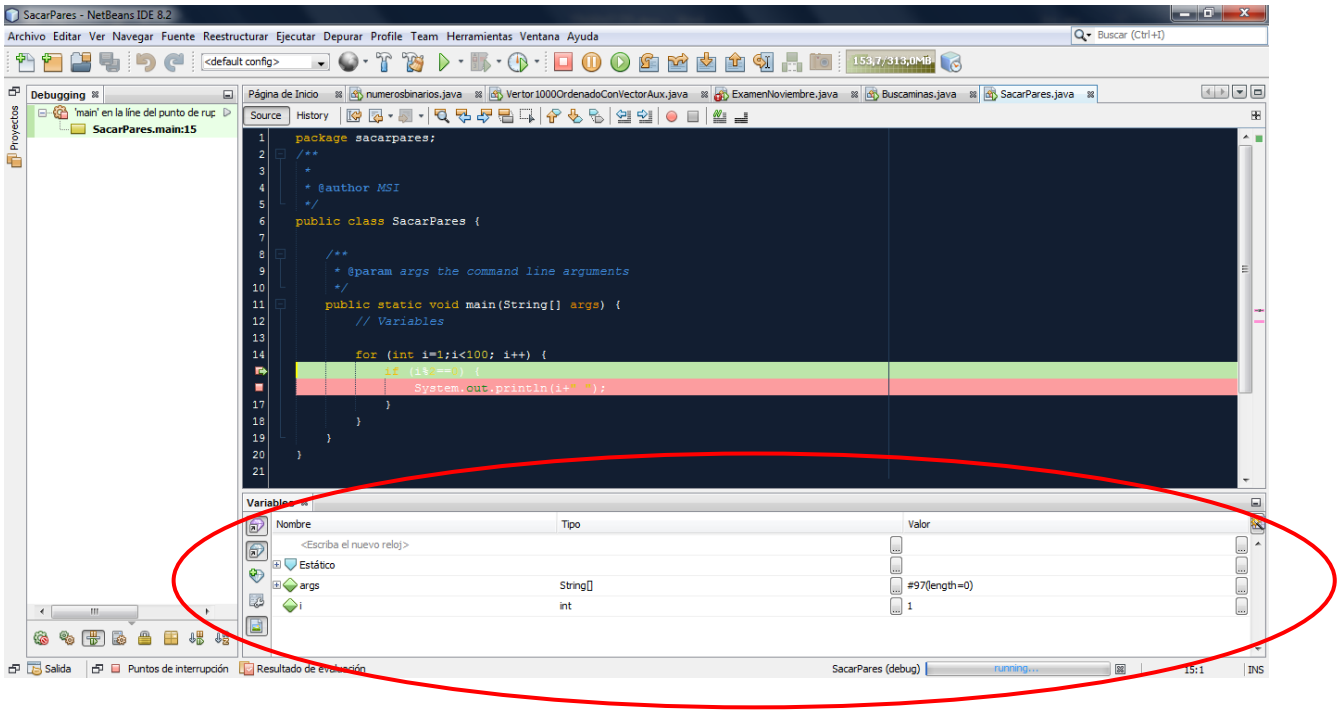
Para mostrarlas vamos a **Ventana-> Depuración** y seleccionar la que queremos.



Las ventanas más importantes y utilizadas son: puntos de interrupción, Variables y Pila de llamadas.

3.1.4.3 LA VENTANA DE VARIABLES LOCALES

En esta ventana se muestran las variables, su tipo y su valor actual.

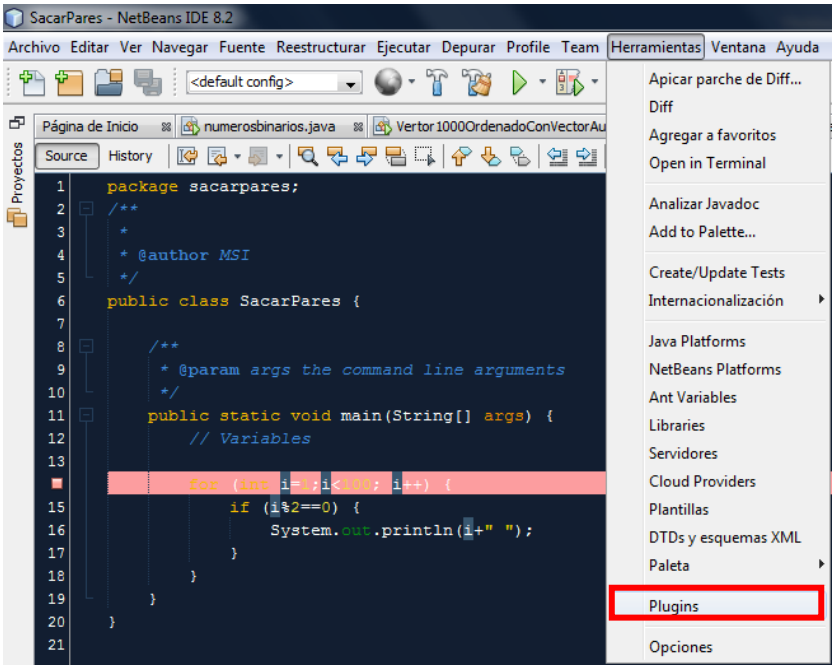


El depurador permite cambiar el valor de una variable local en esta ventana y continuar la ejecución del programa usando el nuevo valor de la variable.

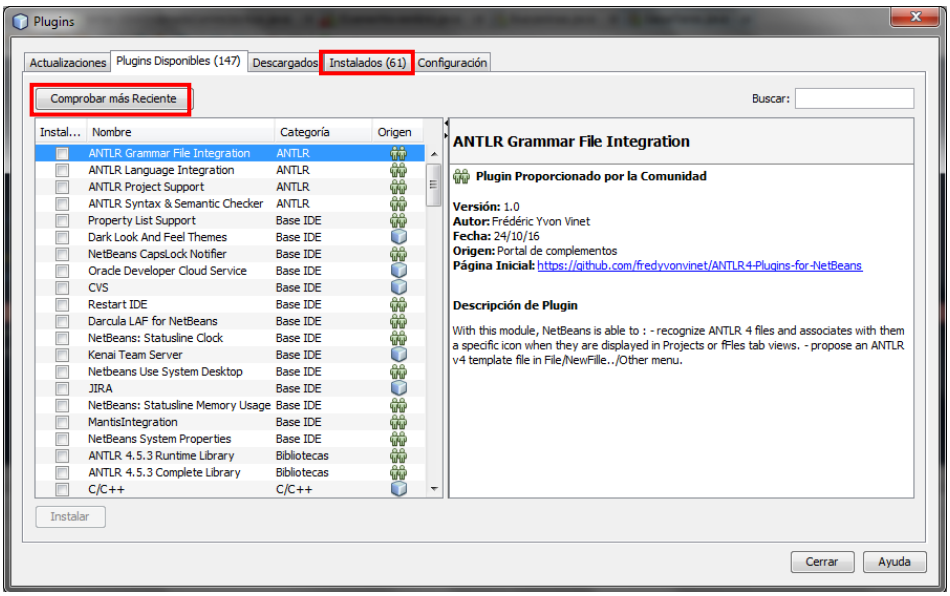
3.1.5 PLUGINS

NetBeans nos ofrece una serie de plugins con las que podremos trabajar y complementar nuestra herramienta de trabajo (la lista de plugins que podemos instalar la podemos encontrar en: http://plugins.netbeans.org/PluginPortal/).

Para descargar e instalar cualquier plugin en el entorno de NetBeans tenemos que ir a **Herramientas → Plugins**

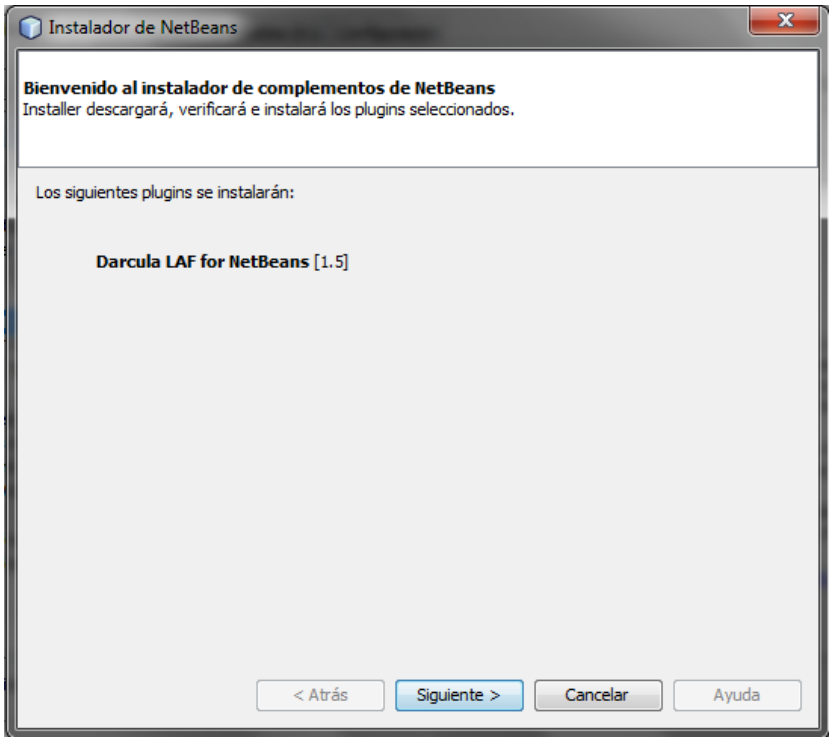


Nos aparecerá la siguiente ventana desde la que podemos actualizar la lista o podemos instalar el deseado si nos situamos sobre alguno de ellos; en la parte derecha de la ventana disponemos de una descripción de dicho plugin; podemos consultar los que tenemos ya instalados, etc.

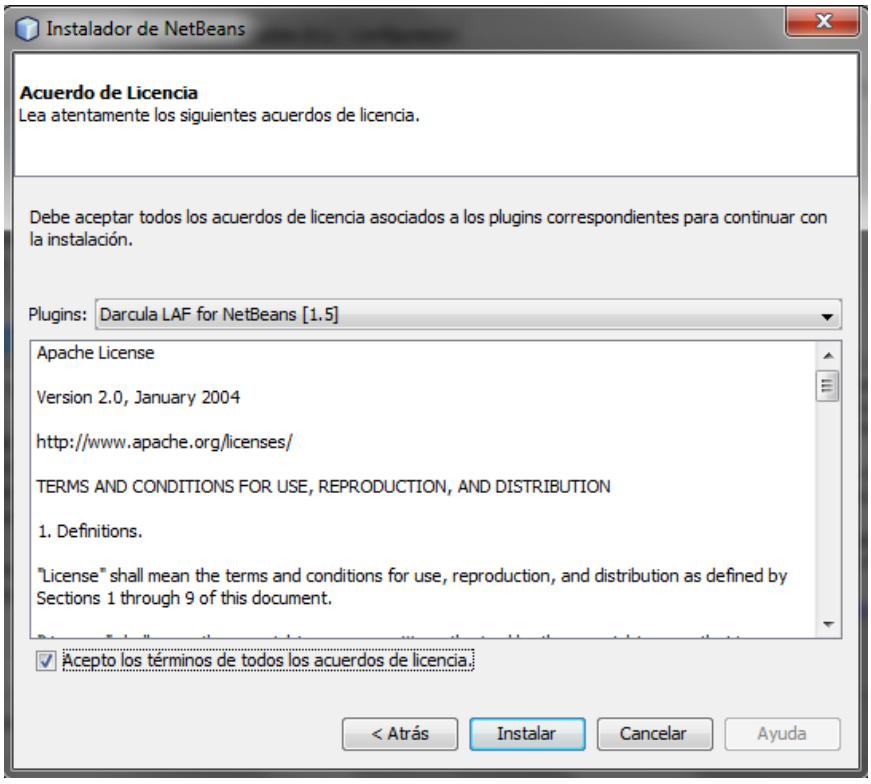


Para la instalación de alguno de los plugins simplemente debemos marcar el deseado y pulsar sobre instalar; a continuación, debemos seguir el asistente de instalación.

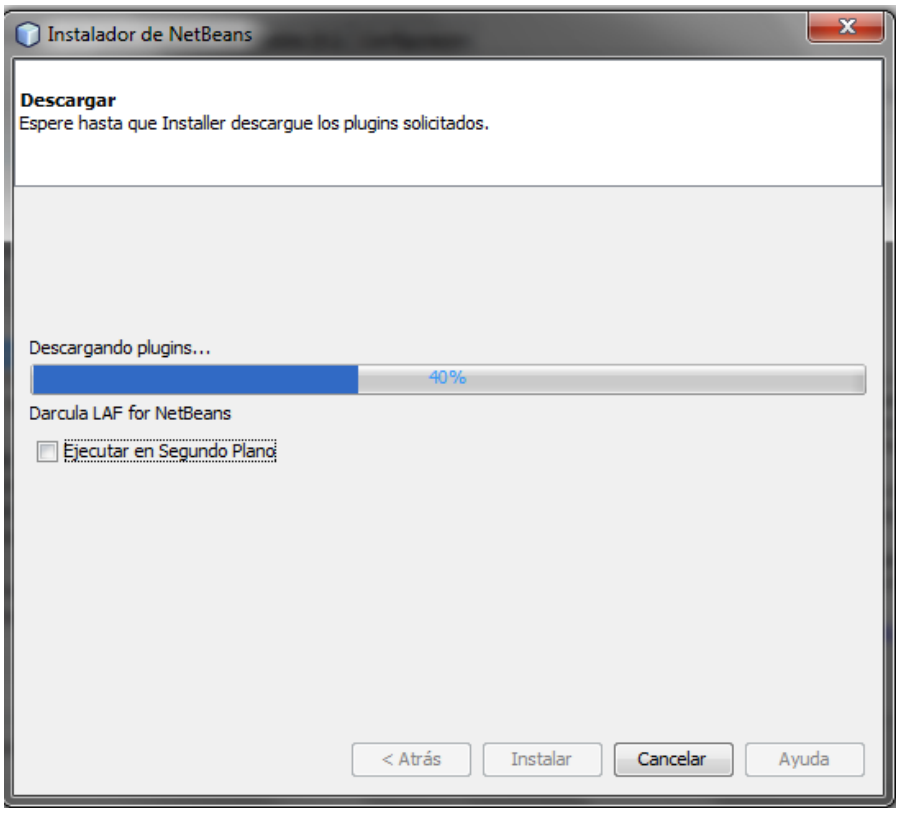
La primera ventana que nos aparece nos informa del plugin que vamos a instalar:



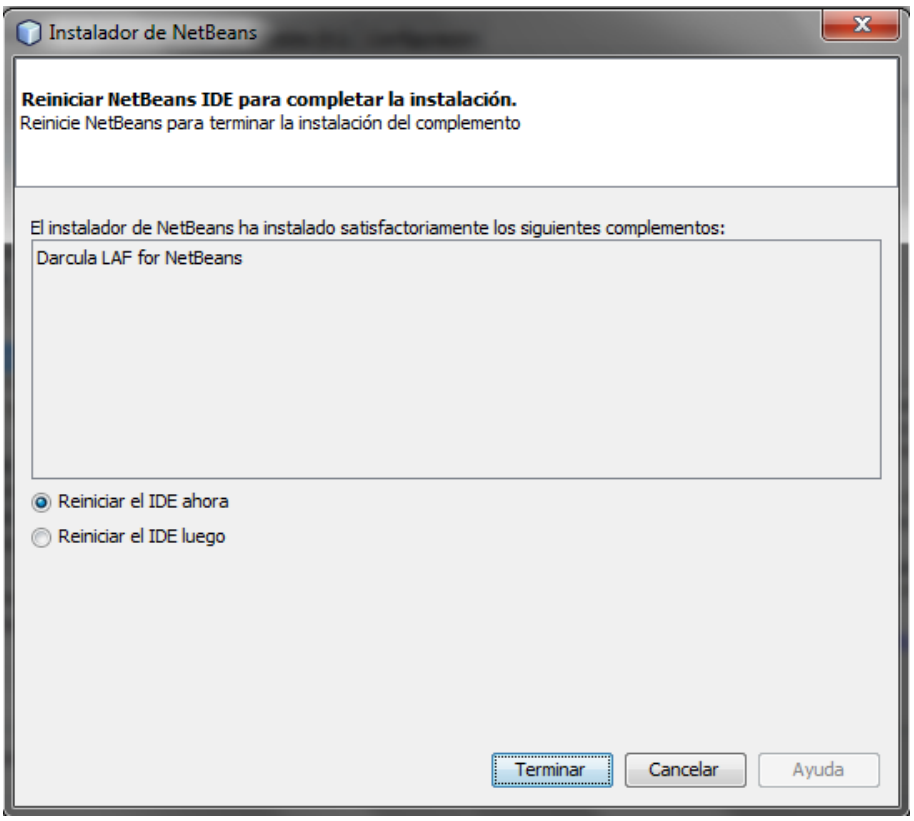
En la siguiente ventana debemos aceptar los términos de licencia si estamos de acuerdo:



Posteriormente comenzará a instalar el plugin:



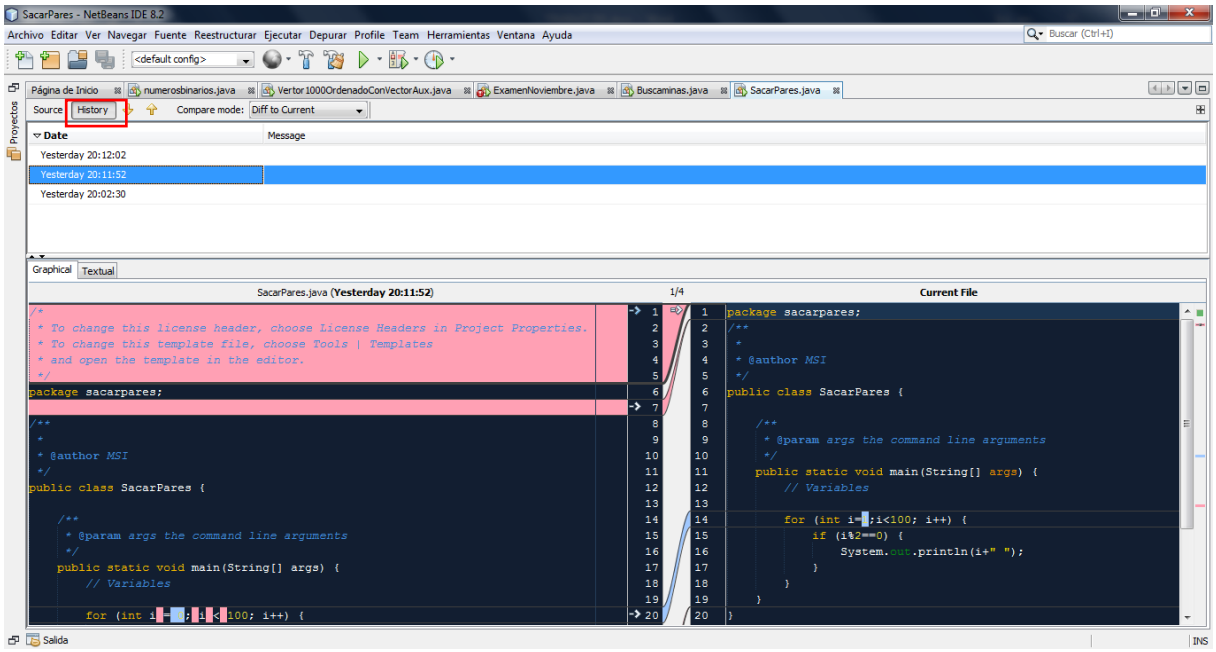
Por último solo debemos reiniciar el IDE para aplicar las nuevas actualizaciones:



3.1.6 CONTROL DE VERSIONES

NetBeans dispone de una herramienta muy útil para los desarrolladores de software, es el control de versiones. Esta herramienta nos muestra los cambios sufridos en nuestro código fuente a lo largo de su codificación.

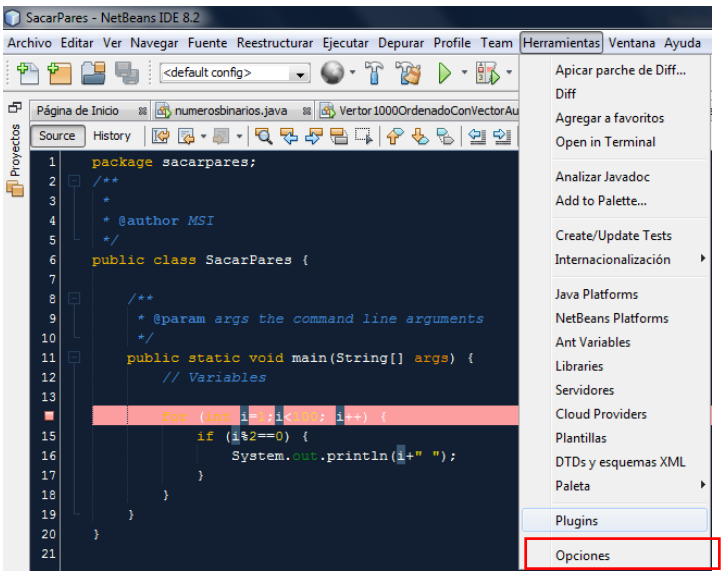
Para acceder a dicha herramienta simplemente pulsamos en la pestaña ***History*** y nos aparecerán las distintas versiones que han sido guardadas a lo largo del desarrollo, ahora solo tenemos que seleccionar la versión que queremos mostrar.



Como podemos observar en la imagen de arriba, al seleccionar una de las versiones nos destaca visualmente cuales han sido los cambios entre la versión seleccionada y la última disponible.

3.1.7 PANEL DE CONFIGURACION

Para acceder a la configuración de NetBeans debemos ir a la pestaña **Herramientas → opciones:**



Nos aparecerá la siguiente ventana con distintas opciones:

3.1.7.1 GENERAL

En esta pestaña podemos encontrar opciones como elegir el navegador, configuraciones de red y si deseamos enviar datos anónimos para la mejora de NetBeans.

3.1.7.2 EDITOR

Aquí aparecerán opciones relacionadas con el editor de texto, algunas de estas opciones son seleccionar el tamaño de la sangría, las sugerencias que nos hace el editor; podemos realizar plantillas, crear macro, activar autoguardado del código, etc.

3.1.7.3 TIPOS DE LETRA Y COLORES

Pues como indica la pestaña, nos muestra opciones para la apariencia de nuestro editor de texto, colores, fuentes, color de fondo, iconos de advertencia, etc.

3.1.7.4 ASIGNACION DE LETRAS

Aquí podemos configurar los atajos del teclado.

3.1.7.5 JAVA

En esta pestaña encontramos todo lo relacionado con el lenguaje de java, su apariencia, el filtrado de paquetes, opciones de depuración, etc.

3.1.7.6 HTML/JS

Aquí podemos encontrar opciones del lenguaje HTML y JavaScript.

3.1.7.7 TEAM

Encontraremos opciones relacionadas con tareas.

3.1.7.8 ASPECTO

En esta pestaña encontraremos todo lo relacionado con la apariencia de nuestro entorno en lo referido a ventanas.

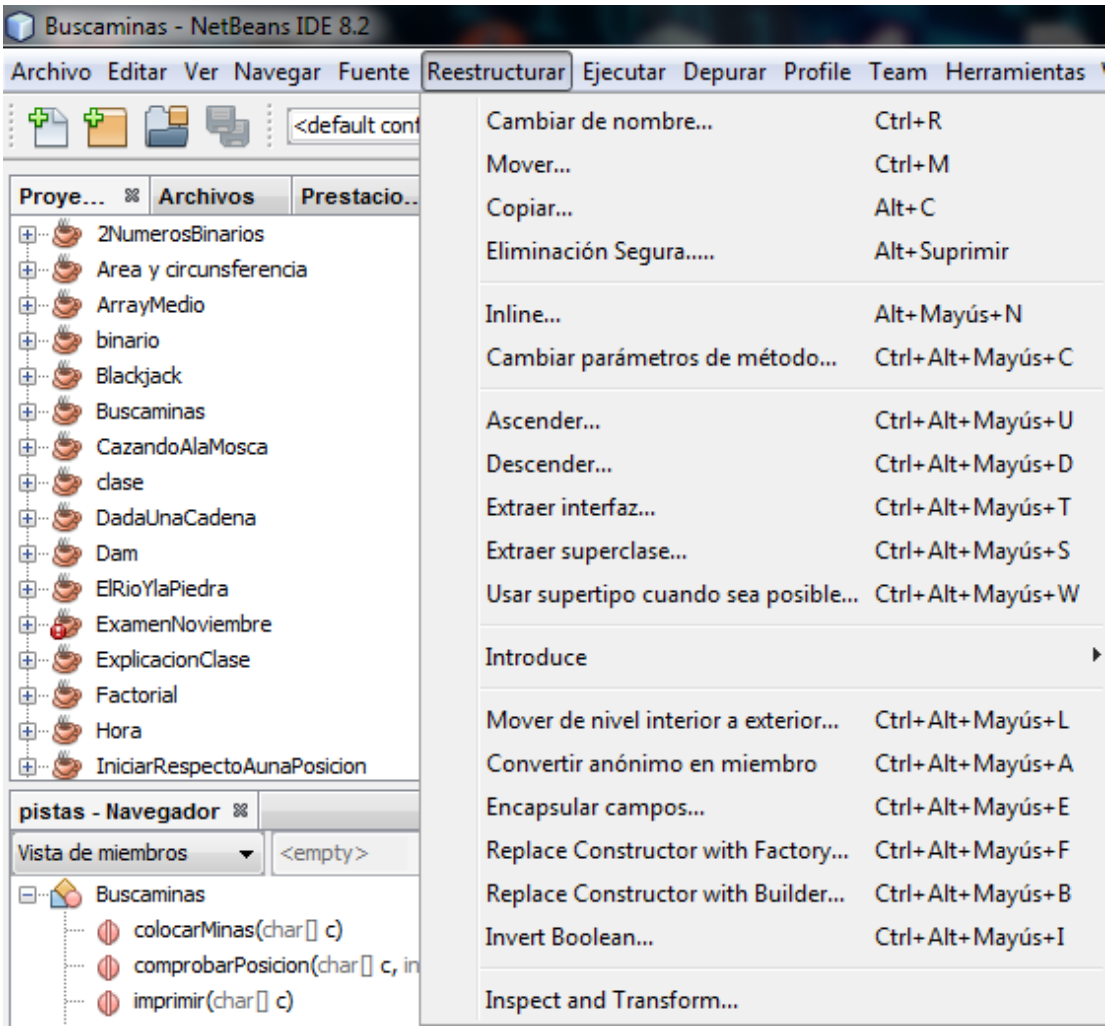
3.1.7.9 VARIOS

Son las opciones relacionadas con la apariencia de la consola en lo referido a fuente, colores, tamaño, etc.

3.1.8 REFACTORIZACION

La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. El objetivo, por el contrario, es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño, y eliminar código muerto para facilitar el mantenimiento en el futuro. Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede refactorizarlo primero para facilitar esta tarea y luego añadir el nuevo comportamiento.

En el entorno de desarrollo de NetBeans para acceder a las opciones de refactorización debemos ir a la pestaña de **Restructurar.**



En esta pestaña disponemos de opciones como cambiar el nombre del paquete principal, mover, copiar y eliminar código, seleccionando una parte de código lo podemos reestructurar creando métodos, etc.

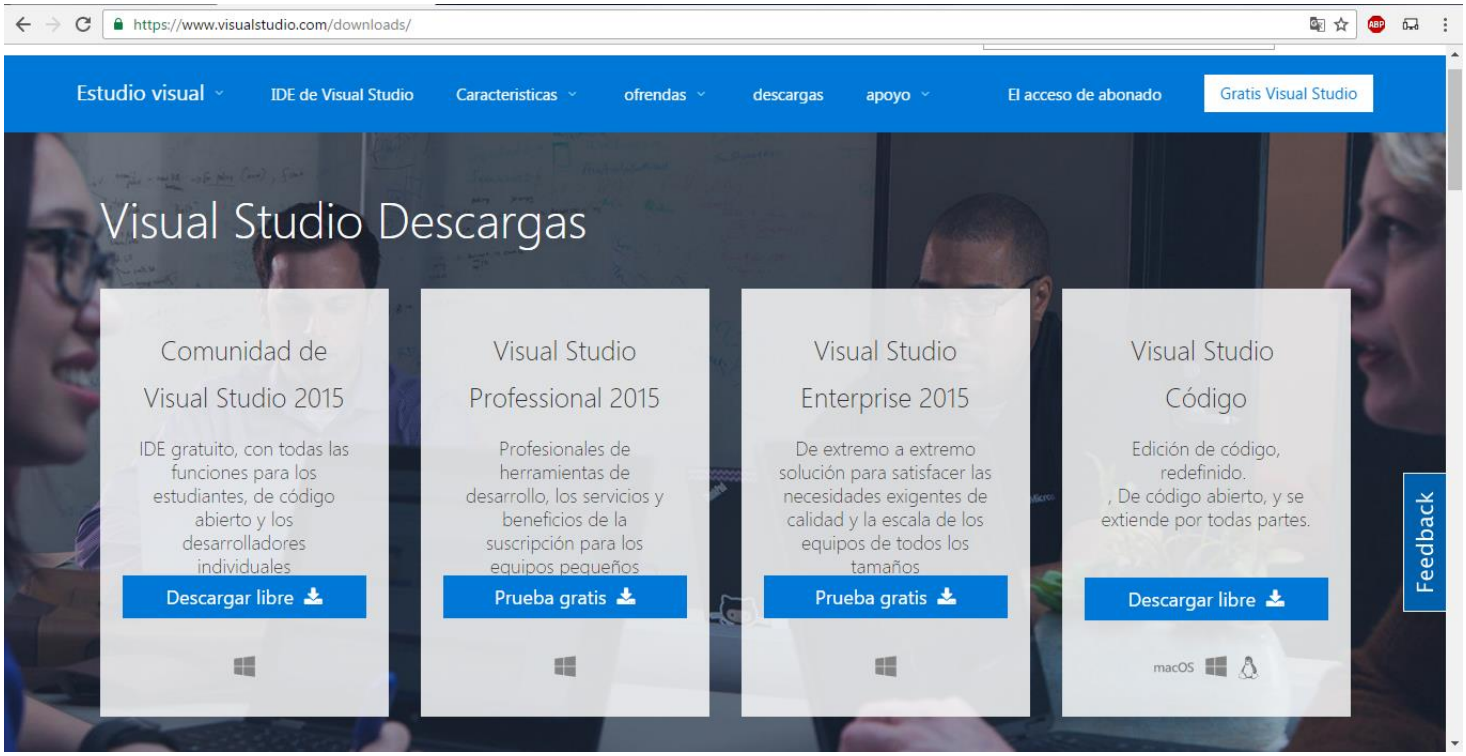
3.2 VISUAL STUDIO

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas entre otros.

Actualmente dispone de 2 versiones Visual Studio 2015 que es con la que nosotros trabajaremos ya que es gratuita y una versión de uso profesional. También hay disponible la versión del 2017 Enterprise, pero aun esta en pruebas.

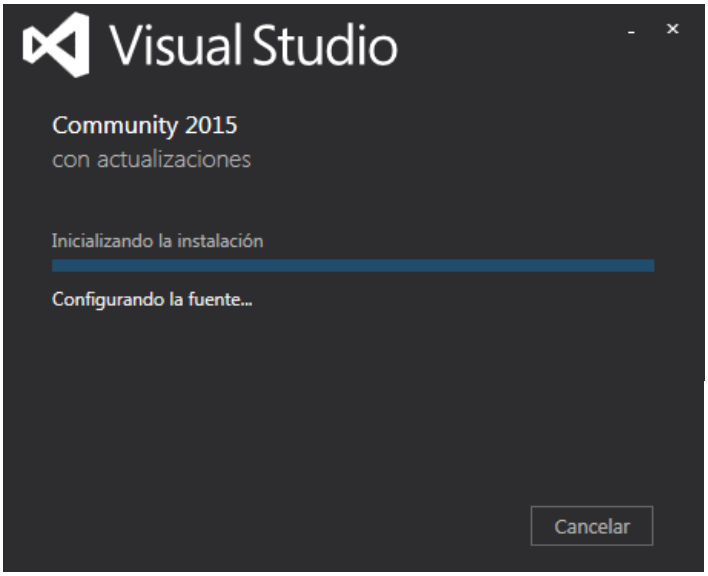
Podemos descargarnos cualquiera de sus versiones desde la página oficial
https://www.visualstudio.com/downloads/



A continuación, veremos el proceso de instalación de Visual Studio 2015 paso a paso.

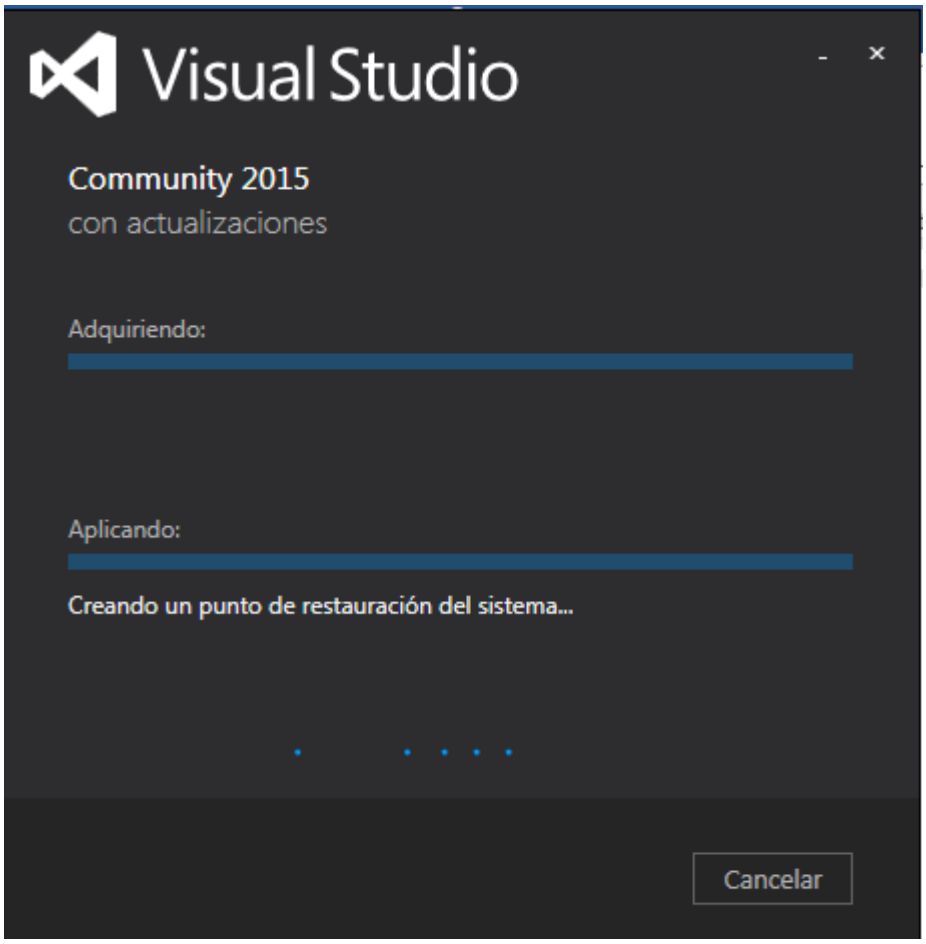
3.2.1 INSTALACION

Una vez descargado, hacemos *doble clic* sobre el paquete y se nos abrirá la siguiente ventana, donde prepara todo lo necesario para su correcta instalación.



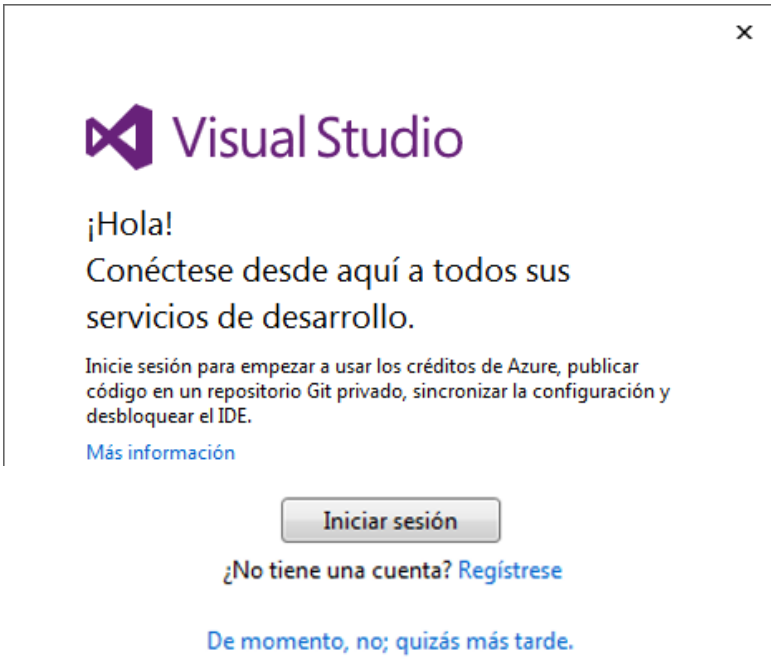
A continuacion se nos pedirá indicar la ubicación, el tipo de instalación y aceptar los términos de la licencia:

Posteriormente comenzará la instalación.

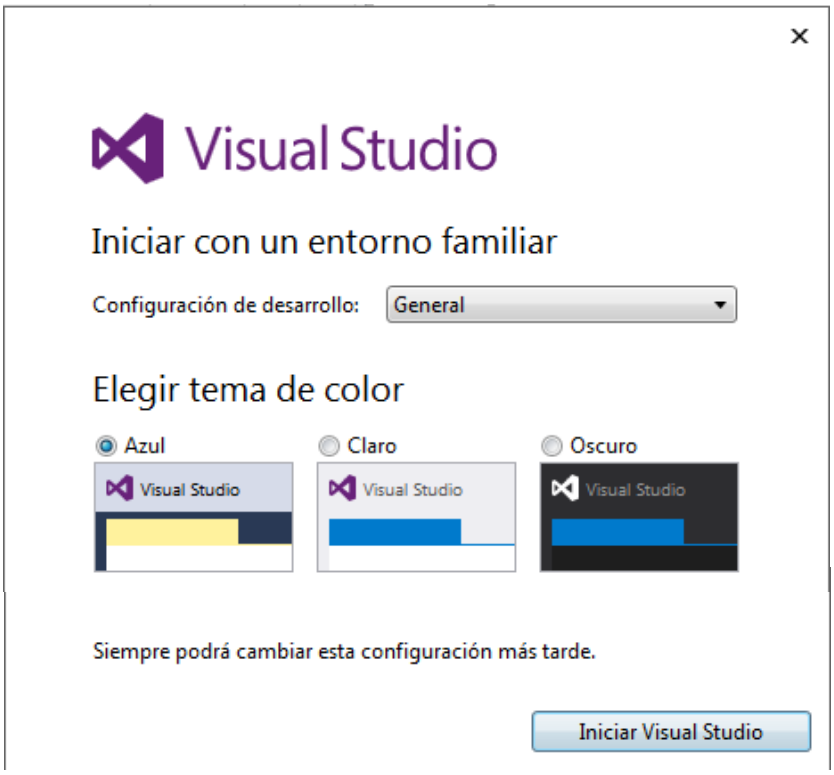


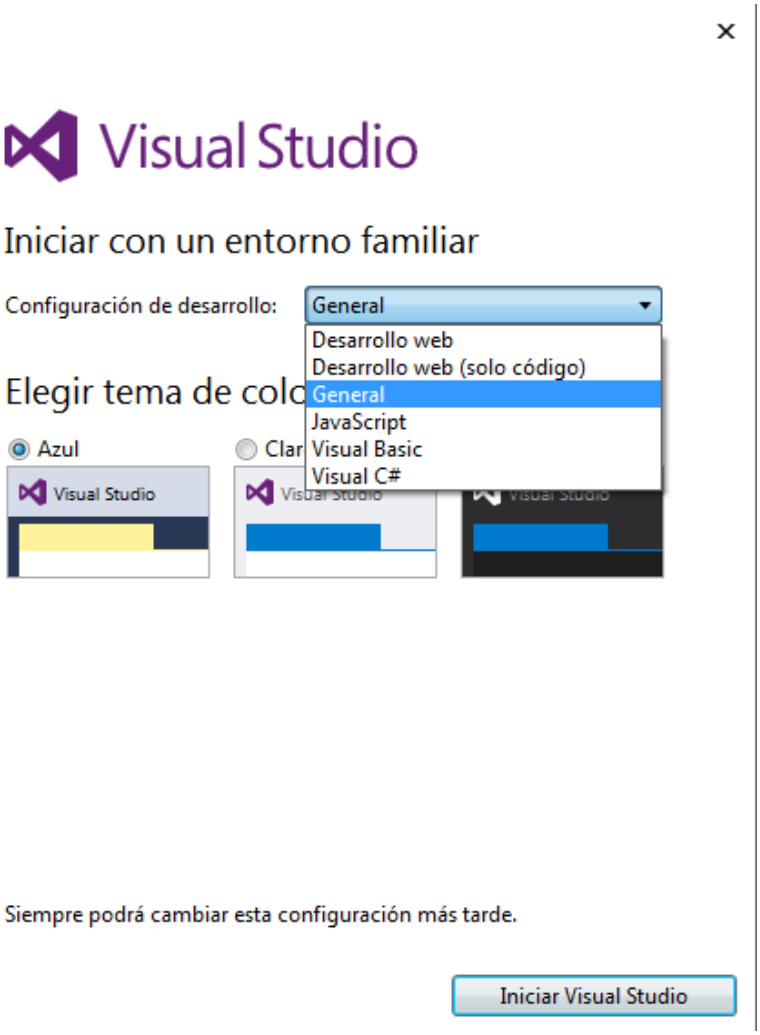
Hemos de decir, que la instalacion ha tardado demasiado tiempo, ya que el software necesita muchisimas librerías adicionales. Por último, reiniciamos el equipo.

Cuando el equipo vuelve a reanudarse nos aparece la siguiente ventana en la que nos pide si deseamos conectarnos con una cuenta de Microsoft, en nuestro caso no lo hemos hecho.

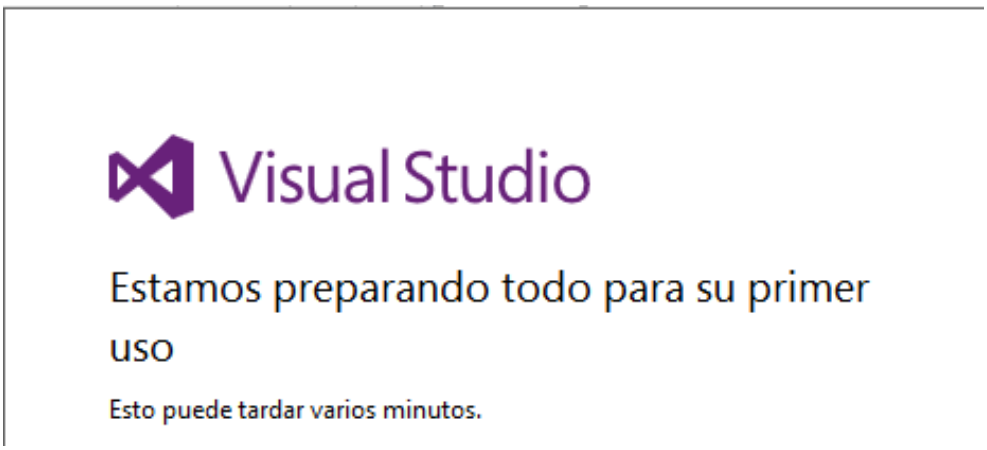


Posteriormente nos aparece la siguiente, ventana en la cual debemos seleccionar una de las siguiente opciones de apariencia.

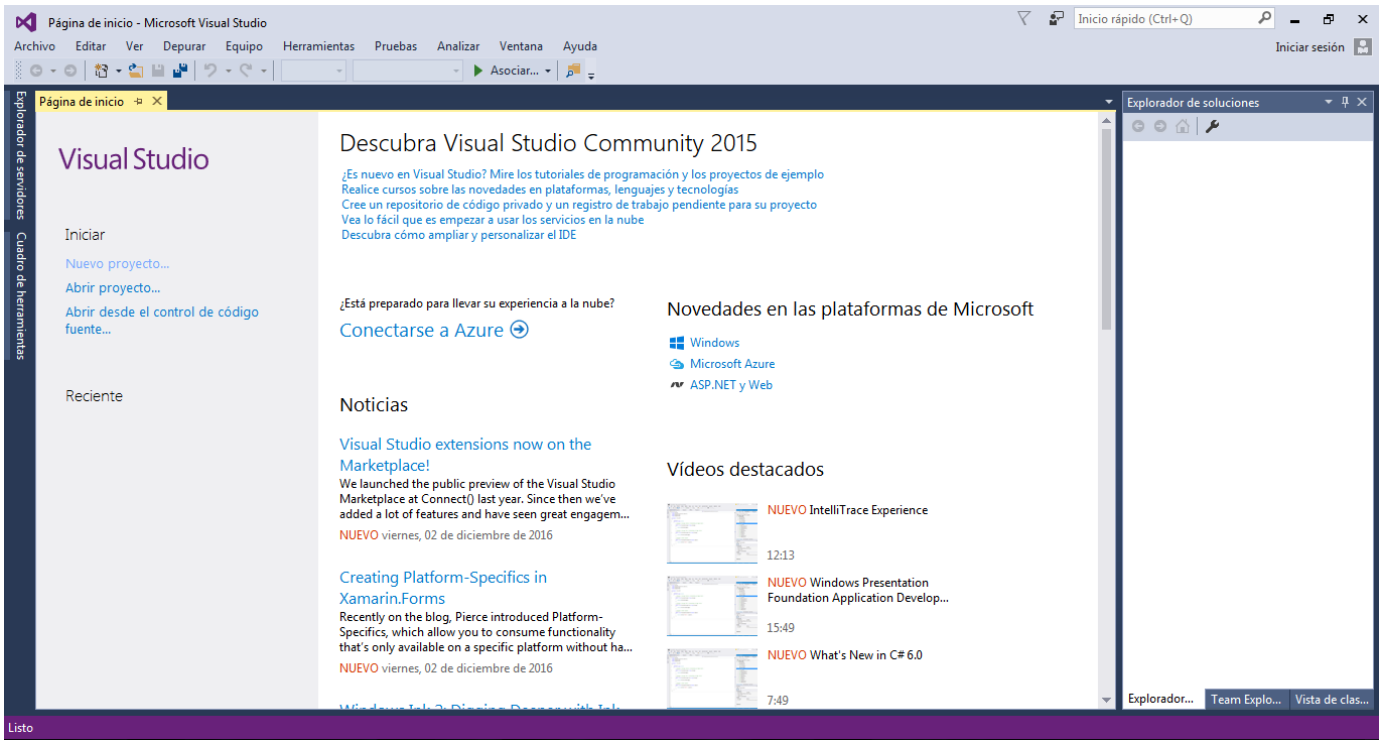




Una vez elegida la configuración el IDE actualizará los cambios.

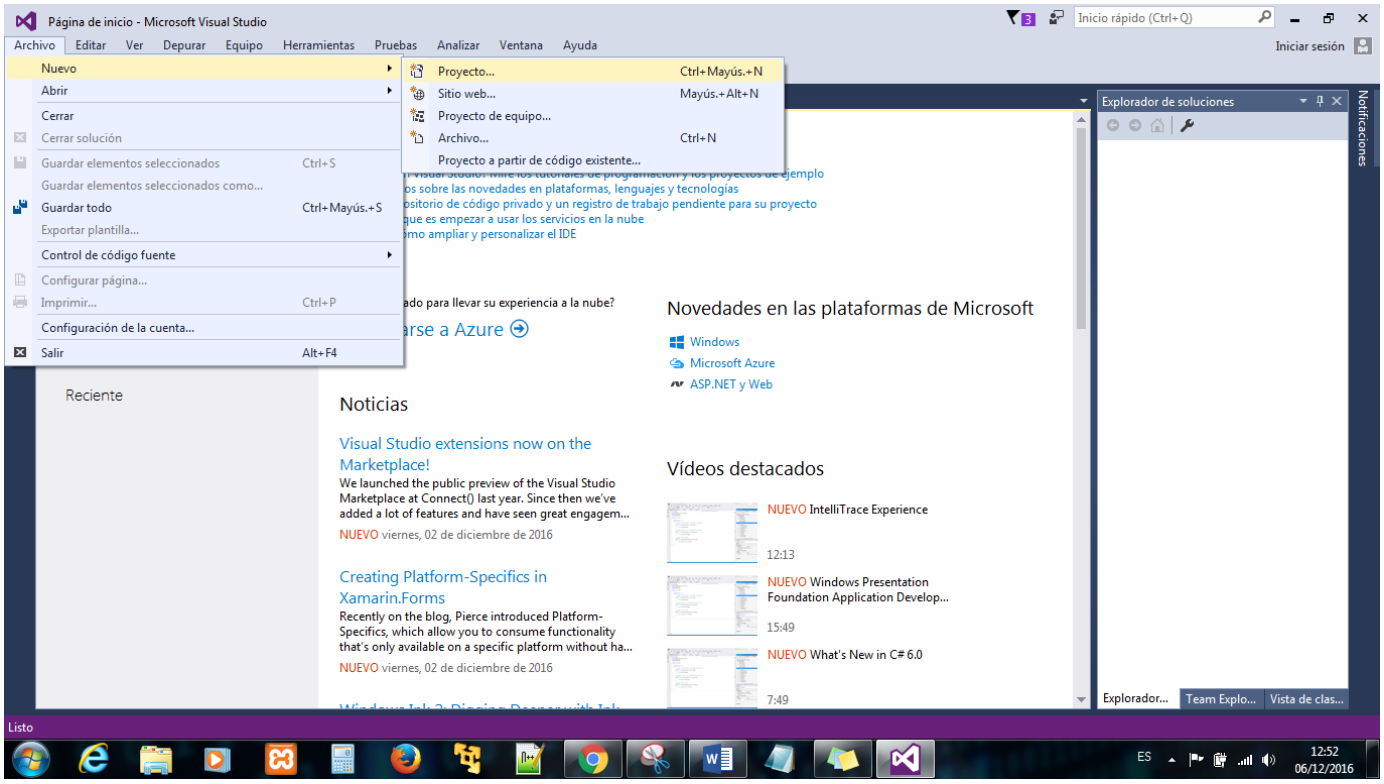


Y aquí tenemos la pantalla principal de nuestro Visual Studio 2017.

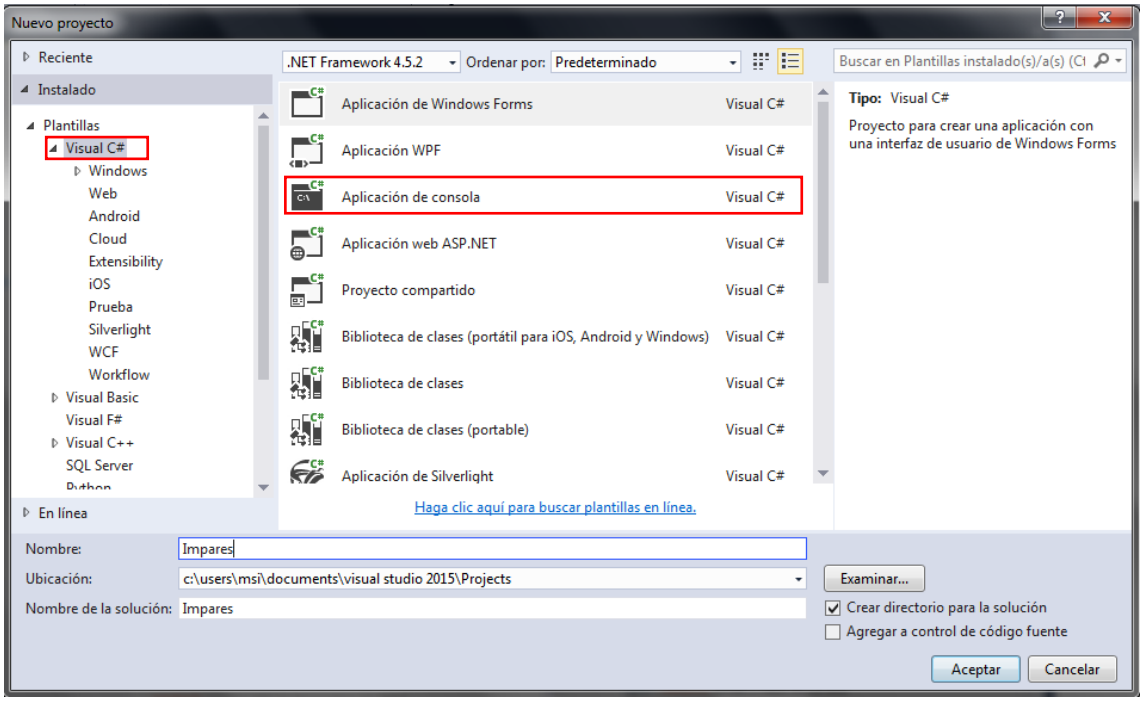


Ahora vamos a crear un nuevo proyecto para ver algunas de sus herramientas.

Para iniciar un nuevo proyecto iremos a la pestaña **Archivo → Nuevo → Proyecto.**

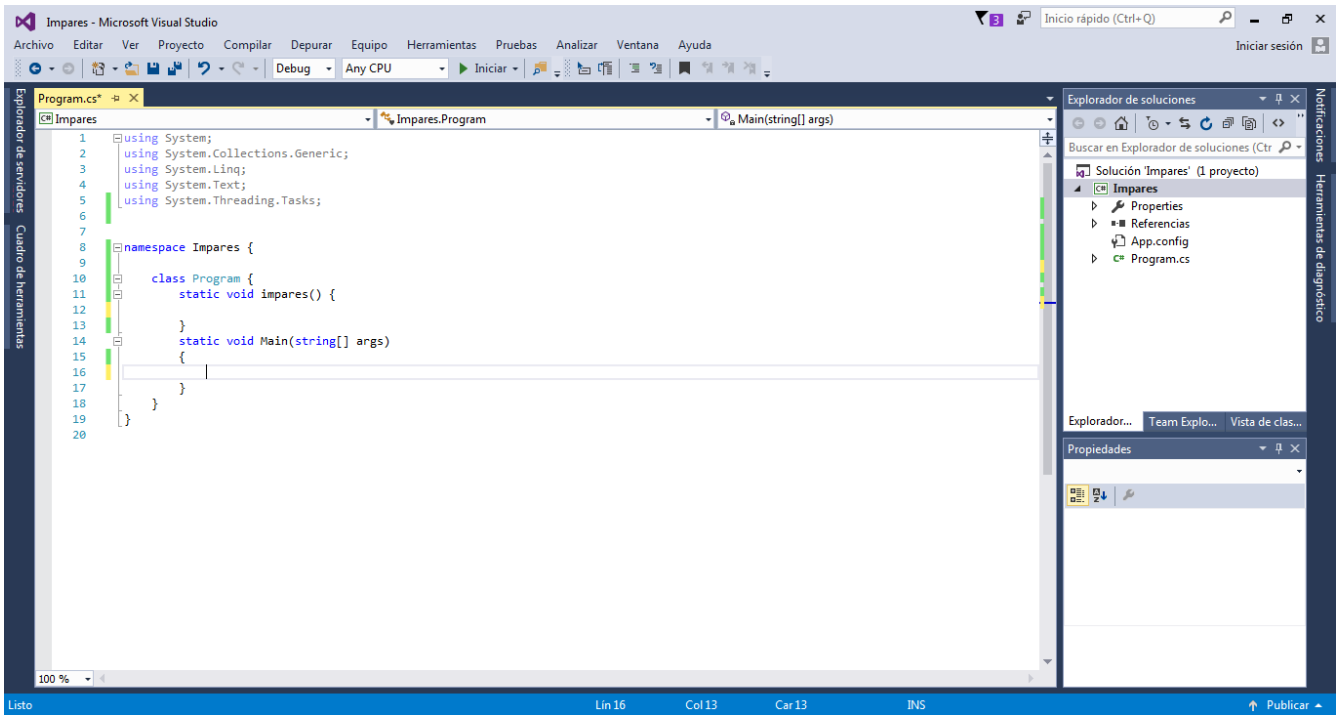


Se nos abrirá la siguiente ventana en la cual debemos de indicarle que en este caso se trata de una aplicación de consola en lenguaje C#. A continuación, introduciremos el nombre del proyecto, la ubicación del mismo.

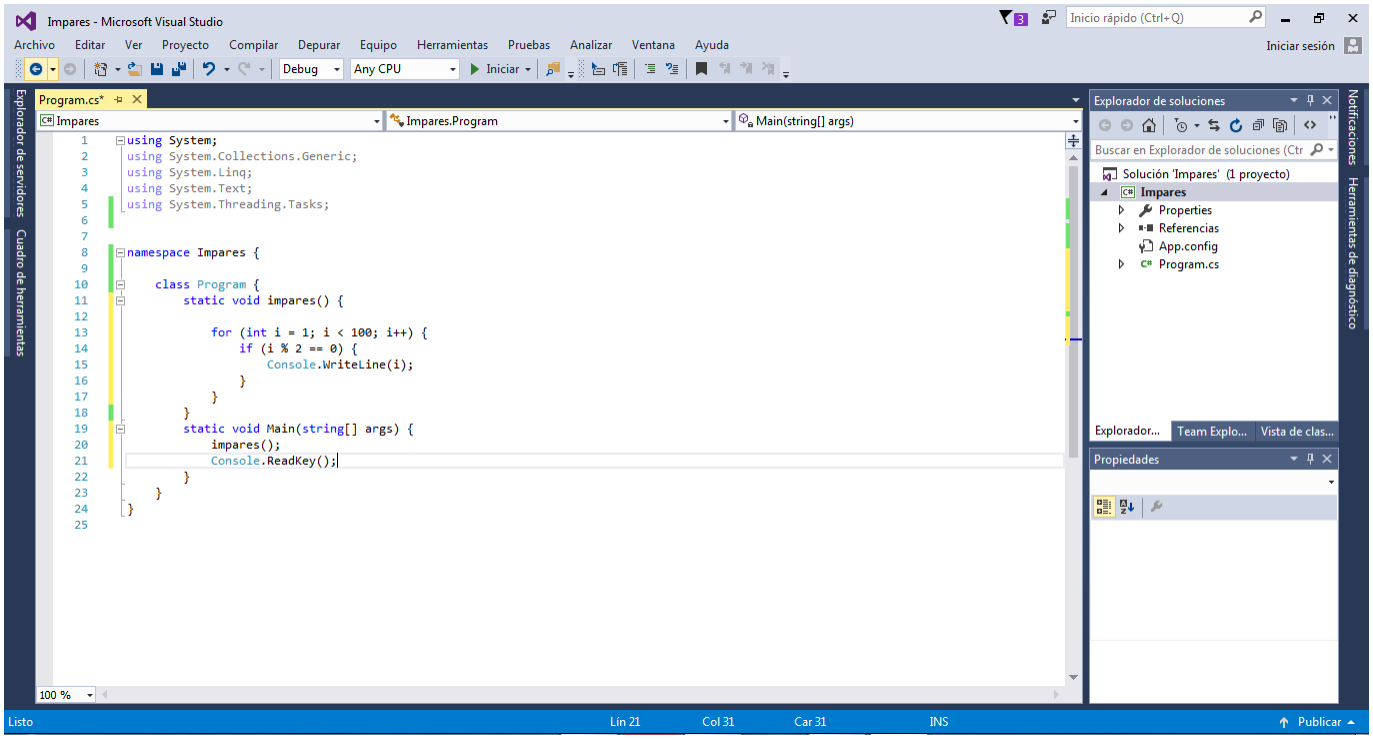


3.2.2 EDITOR DE TEXTO

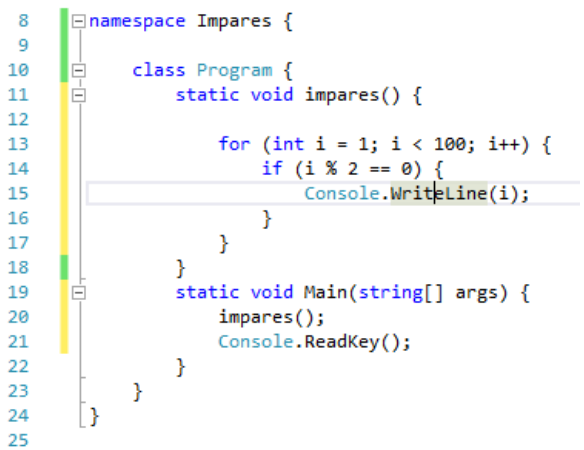
Ya tenemos aquí nuestro editor de texto, que por defecto es una plantilla con la clase main ya construida.

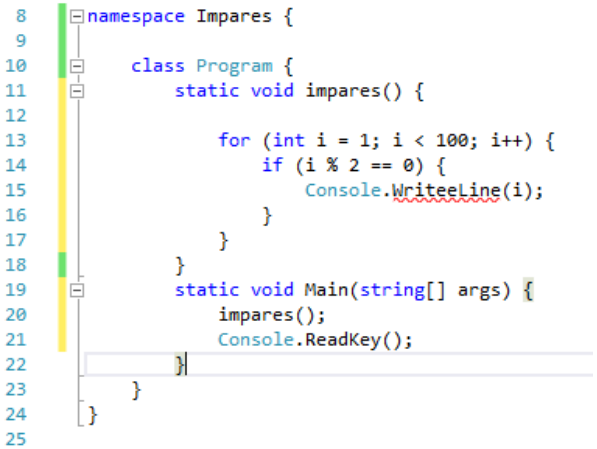


A continuación, debemos escribir nuestro programa. El editor de texto nos ayuda según vamos escribiendo e irá marcando con un subrayado rojo los posibles errores de sintaxis y posibles errores léxicos.

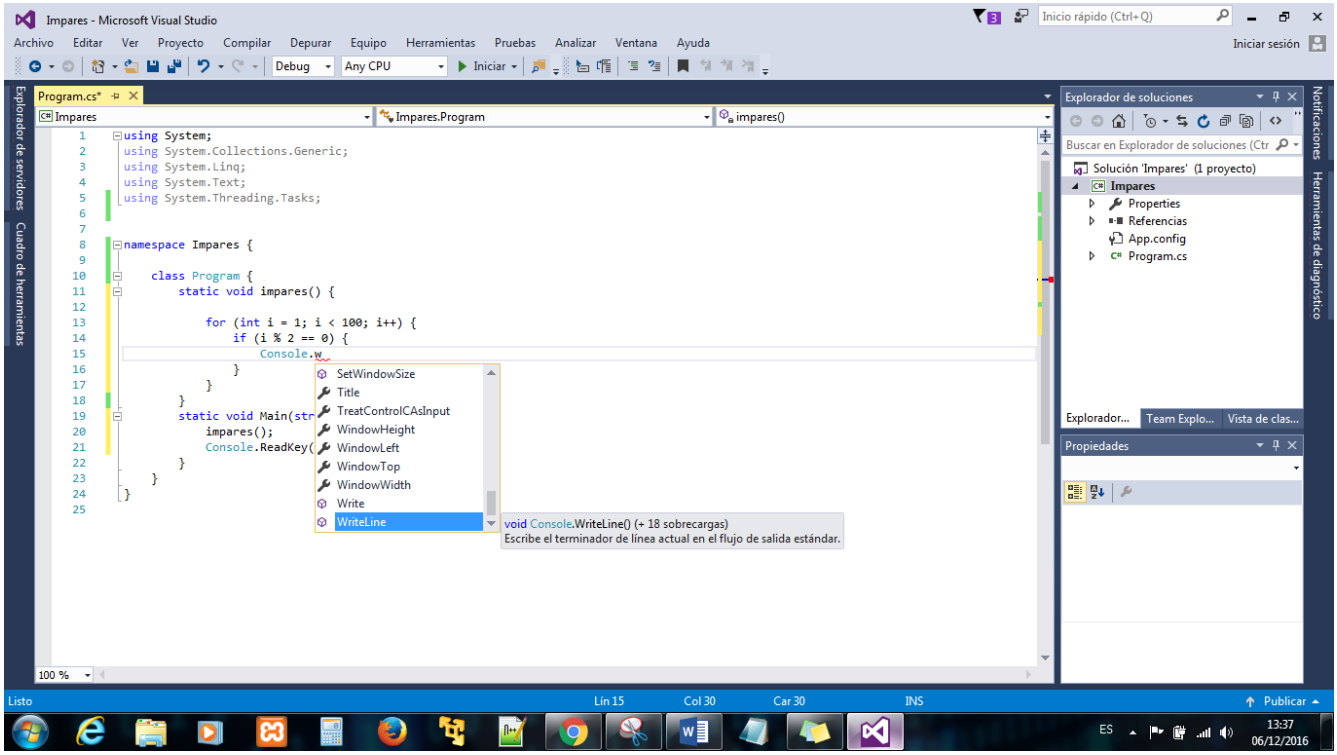


Aquí podemos ver un ejemplo con la diferencia de un código bien escrito frente a otro que no lo está, de forma que nuestro editor nos muestra un error léxico en la palabra mal escrita. También podemos distinguir que nos muestra de forma resaltada las palabras reservadas del lenguaje.





El error descrito anteriormente es difícil de cometer, puesto que como podemos ver en la imagen inferior, el propio editor nos muestra las posibles opciones disponibles según vamos escribiendo.



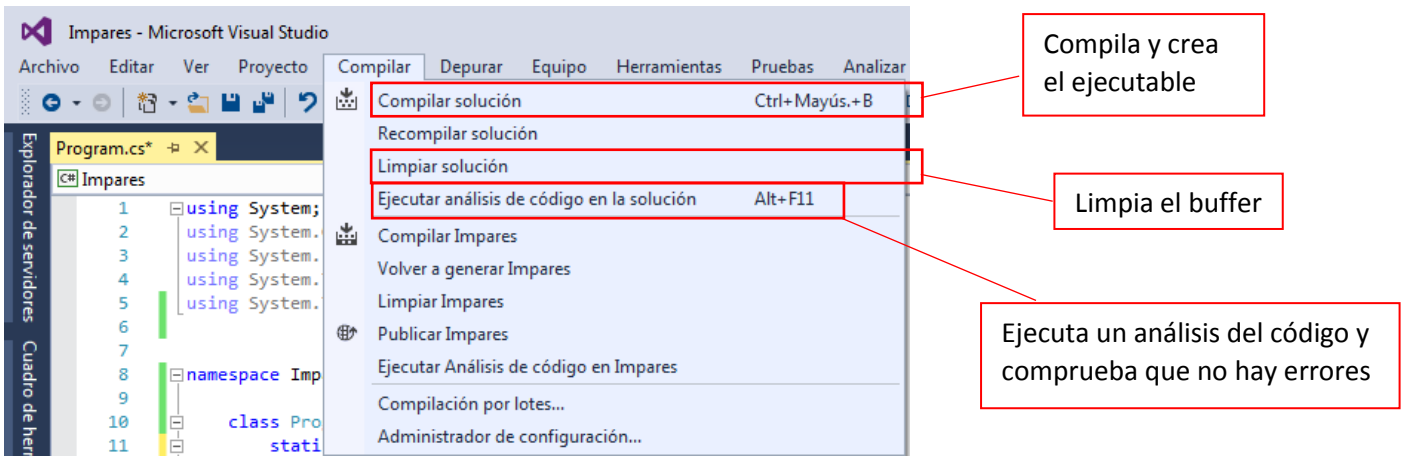
3.2.3 COMPILADOR

Para ejecutar el programa debemos compilar nuestro código, es decir, transformar nuestro código fuente en un código binario o código máquina y para ello tenemos varias formas de hacerlo. Una de las formas y la más rápida, sería con el atajo de teclado F5, este método de compilación nos permite ejecutar el programa en una consola del sistema, pero he observado que la ejecución en este lenguaje se realiza y desaparece sin darnos tiempo a ver el resultado.

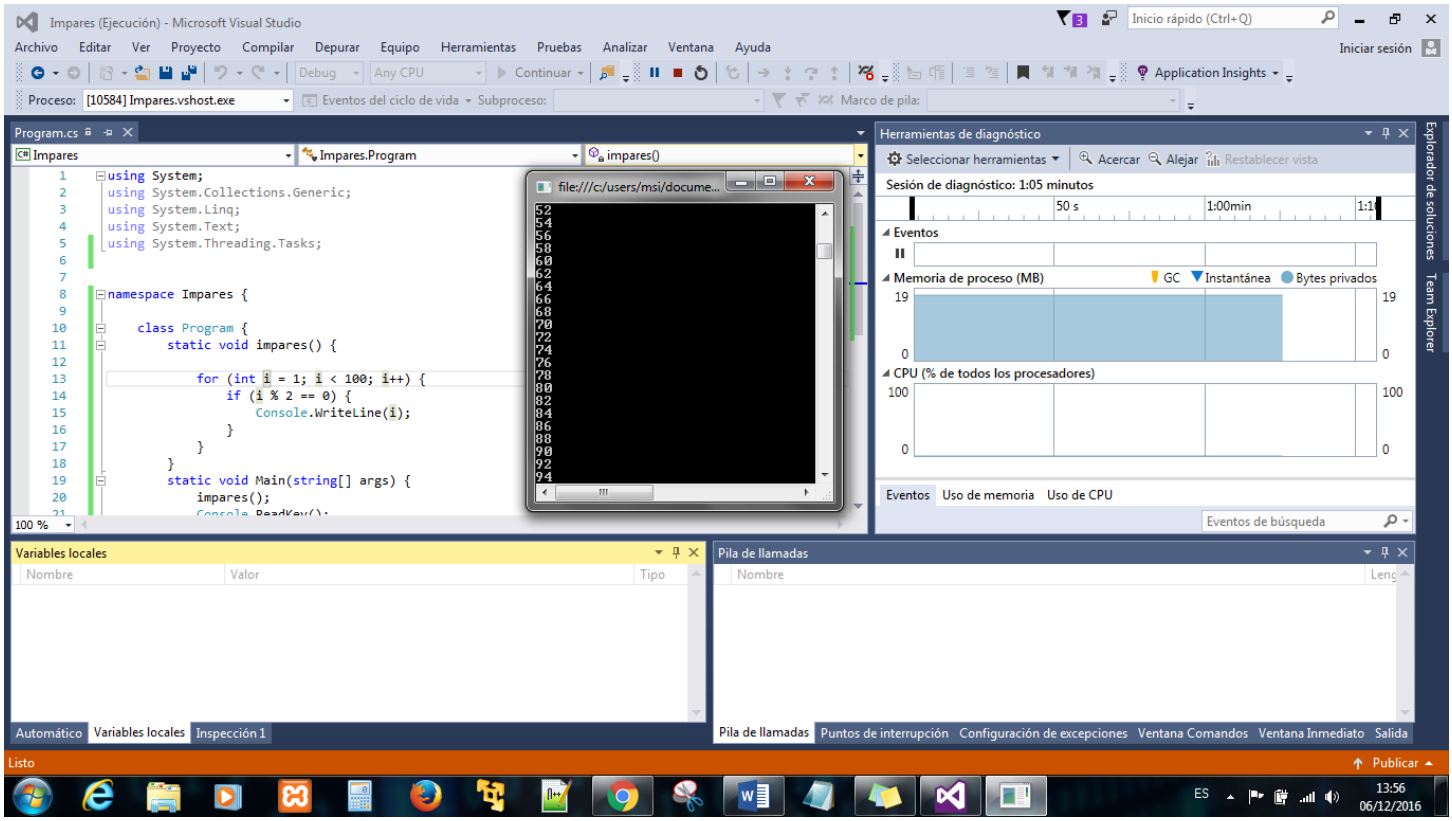
Para poder ver el resultado obtenido en la consola es necesario detener el sistema, nosotros lo hemos hecho introduciendo una línea de código al final del código fuente en el que el programa se detiene a la espera de introducir un dato.

```
21          Console.ReadKey();
```

Otras de las maneras para compilar nuestro algoritmo son las siguientes:



Cuando pulsamos F5 para que inicie el programa y podamos visualizar los resultados obtenidos, por defecto el programa nos lo inicia en modo depuración, de forma que podemos ver los recursos consumidos, las variables, la pila y la consola de salida del sistema.



Para iniciar el programa sin entrar en el modo depuración debemos ir a la pestaña **Depurar** → **Iniciar sin depurar** u otra manera es con el atajo del teclado **Ctrl+F5**.

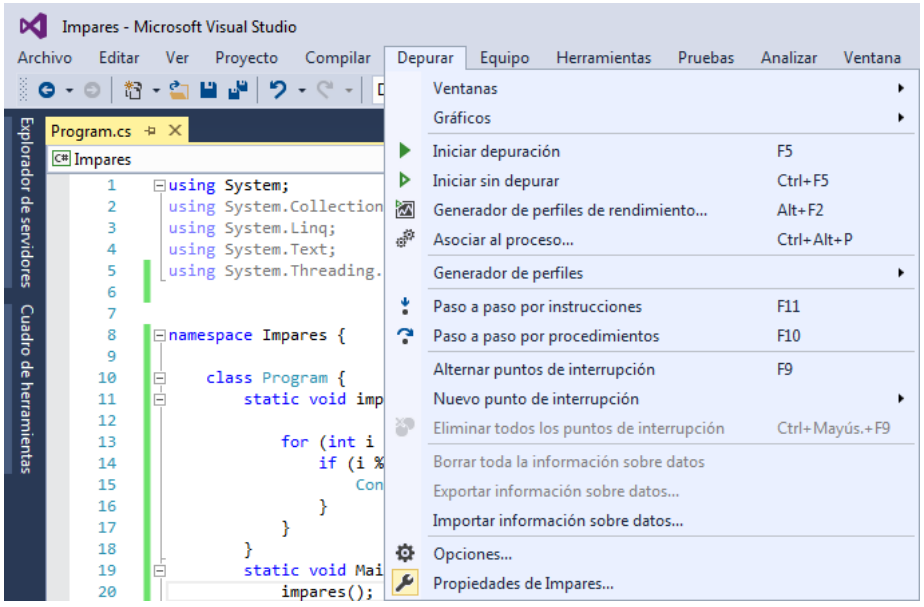
3.2.4 DEPURADOR

El depurador o debugger nos permite ejecutar el código fuente paso a paso para identificar posibles errores de forma más sencilla y rápida facilitándonos su búsqueda y obteniendo una forma más eficiente de desarrollo.

Para ello, nos permite utilizar puntos de ruptura o breakpoint para detener la ejecución del programa y poder observar el estado de las variables, su valor, e incluso modificarlo sobre la marcha y continuar la ejecución.

Hay varias formas de iniciar la ejecución del programa en modo depuración:

Por defecto, en Visual Studio cuando pulsamos sobre el botón **iniciar** o **F5** nos lo inicia en el modo depuración. Otra manera de llegar al depurador es a través de la pestaña **depurador**;



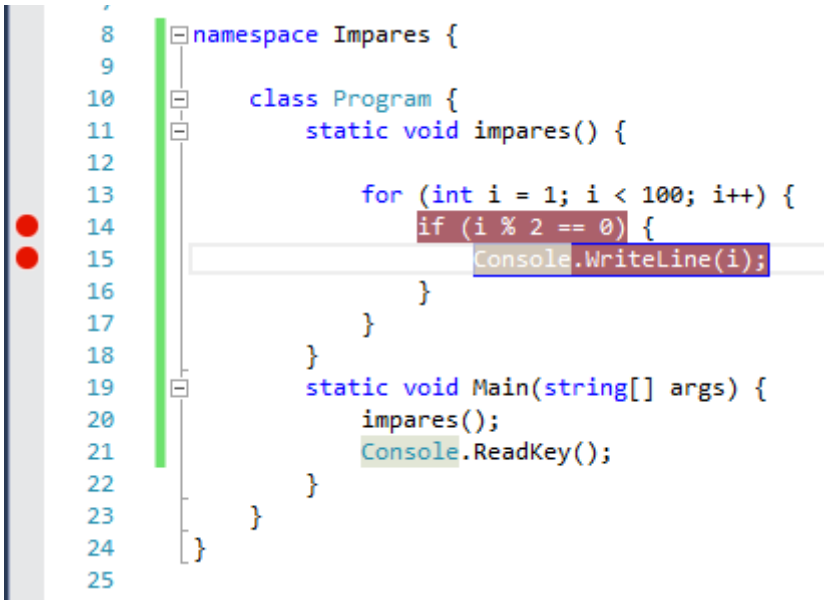
En esta pestaña disponemos de distintas opciones de depuración, como son, la depuración estándar que ejecuta el algoritmo hasta encontrarse con un punto de ruptura.

Depurar el código paso a paso por instrucción

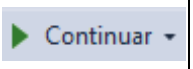
Una vez iniciada la depuración, el depurador se detiene en la siguiente línea de código que se va a ejecutar y esta aparece en verde, con una flecha verde a su izquierda.

Depurar el código paso a paso por procedimiento

Una vez iniciada la depuración, el depurador se detiene en el primer procedimiento y lo comienza a ejecutar.



Para el uso del depurador manejamos distintas opciones a través de botones:

	**Step Over (F10)** Ejecuta una línea de código.
	**Step Into (F11)** Ejecuta una línea de código. Si la instrucción es una llamada a un método, salta al método y continúa la ejecución por la primera línea del método.
	**Step Out (Mayus+ F11)** Ejecuta una línea de código. Si la línea de código actual se encuentra dentro de un método, se ejecutarán todas las instrucciones que queden del método y se vuelve a la instrucción desde la que se llamó al método.
	**Continue** La ejecución del programa continúa hasta el siguiente punto de ruptura. Si no existe un punto de ruptura se ejecuta hasta el final.
	**Finish Debugger Session (Mayús + F5).** Termina la depuración del programa.
	**Reboot (Ctrl+Mayus+F5).** Reinicia la ejecución del depurador.

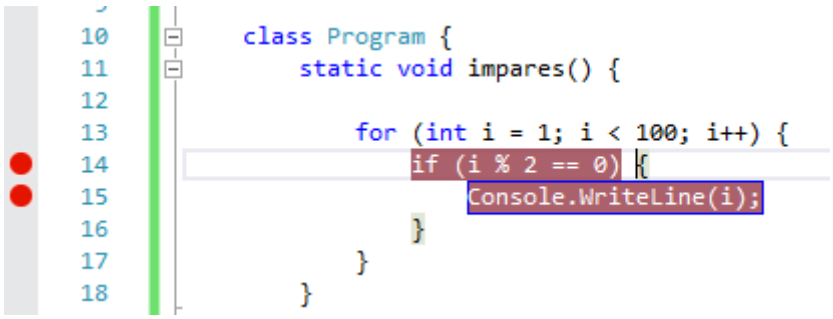
3.2.4.1 ESTABLECER PUNTOS DE RUPTURA

Un punto de ruptura es una marca que indica al depurador que debe detenerse cuando la ejecución del programa llegue a ella.

Cuando el programa se detiene podemos:

- Examinar los valores actuales de las variables.
- Continuar la depuración línea a línea del programa.

Para fijar un punto de ruptura simplemente pulsamos sobre la parte izquierda del número de línea donde se desea colocar. La línea queda resaltada en color rojo con una marca del mismo color en el margen izquierdo.

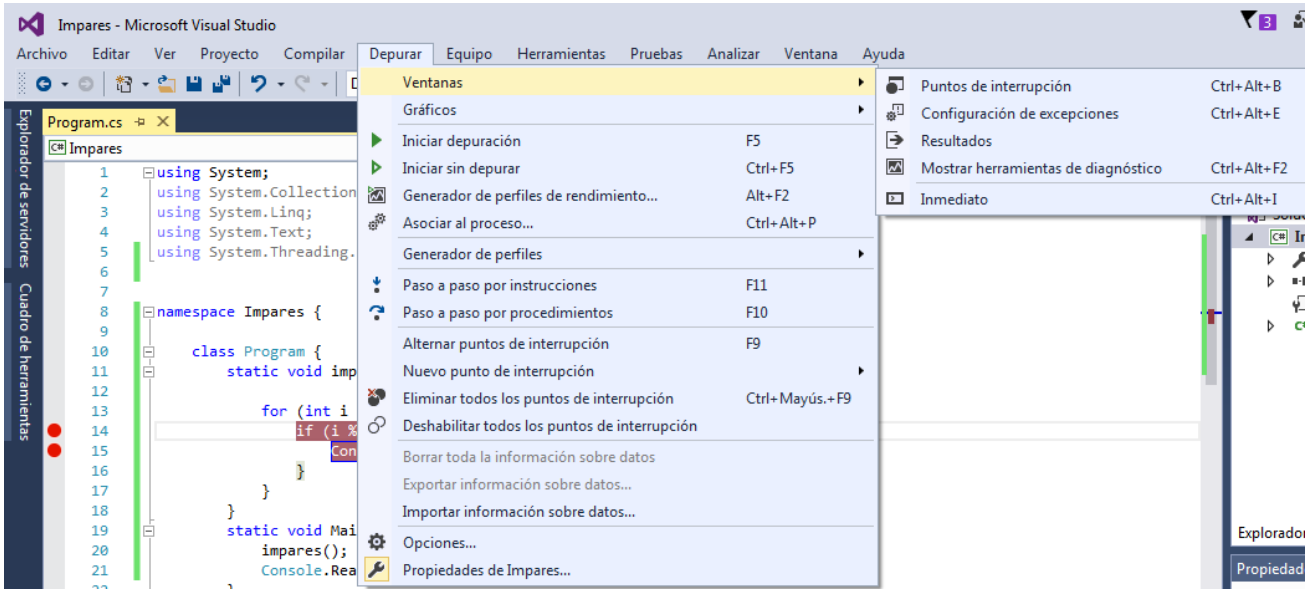


Para eliminar un punto de ruptura se pulsa sobre el cuadrado rojo.

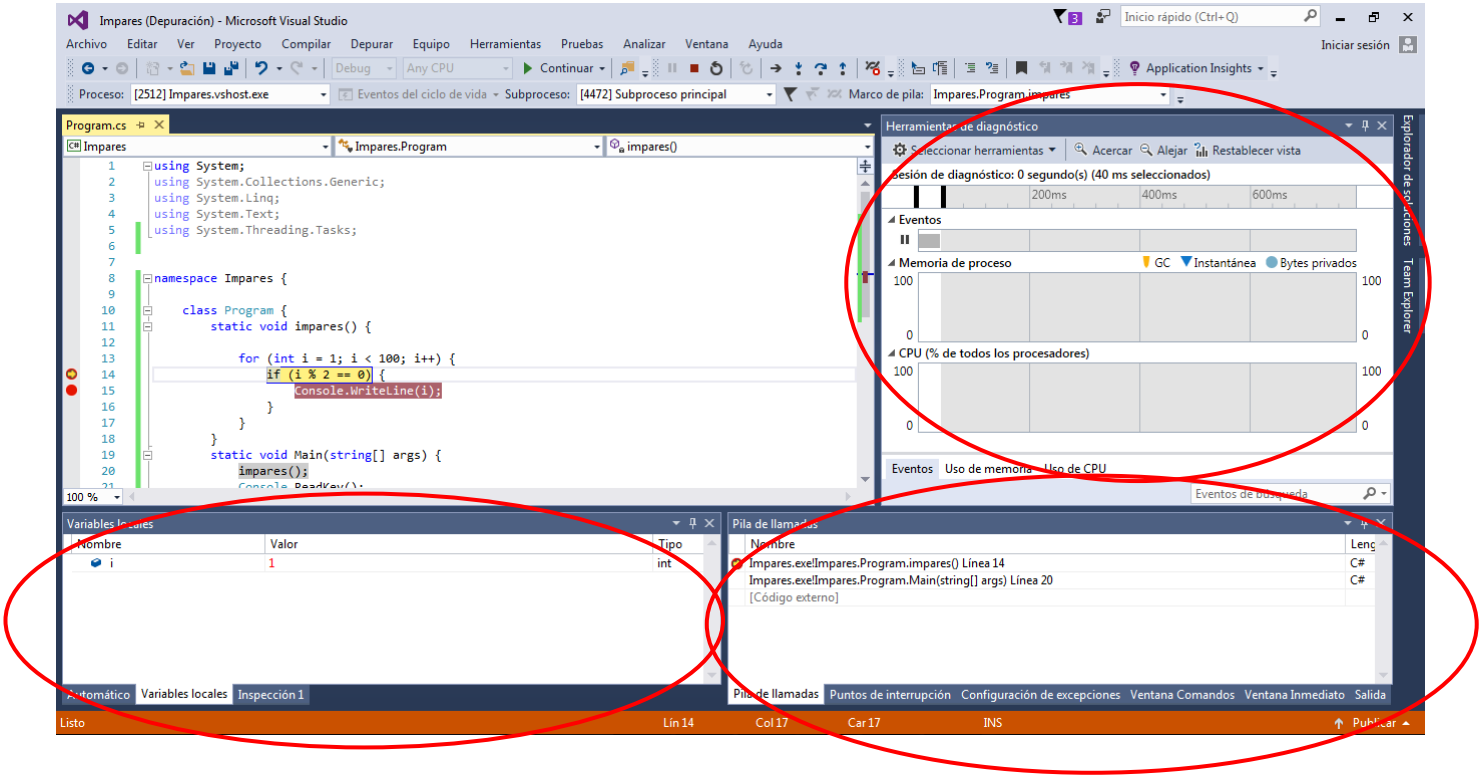
3.2.4.2 VENTANAS DEL DEPURADOR

Para el proceso de depuración se usan distintas ventanas que en Visual Studio casi todas aparecen automáticamente.

Para mostrarlas vamos a **Depurar-> Ventana** y seleccionar la que queremos.



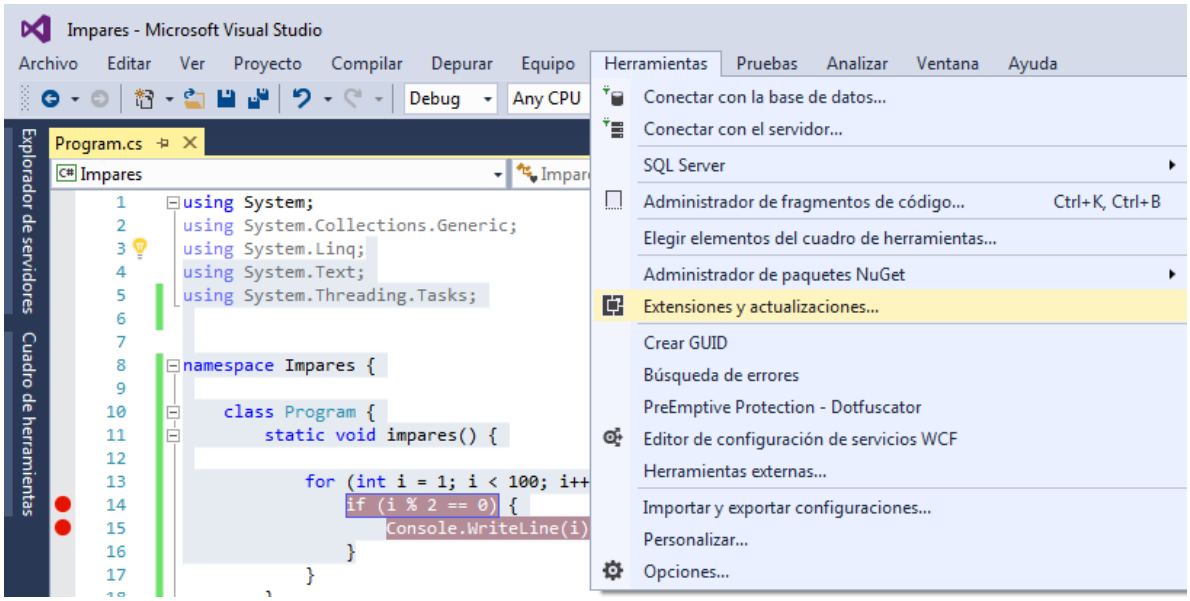
Las más importantes y utilizadas son: puntos de interrupción, Variables y Pila de llamadas, en el caso de Visual Studio, también podemos ver la herramienta de diagnostico.



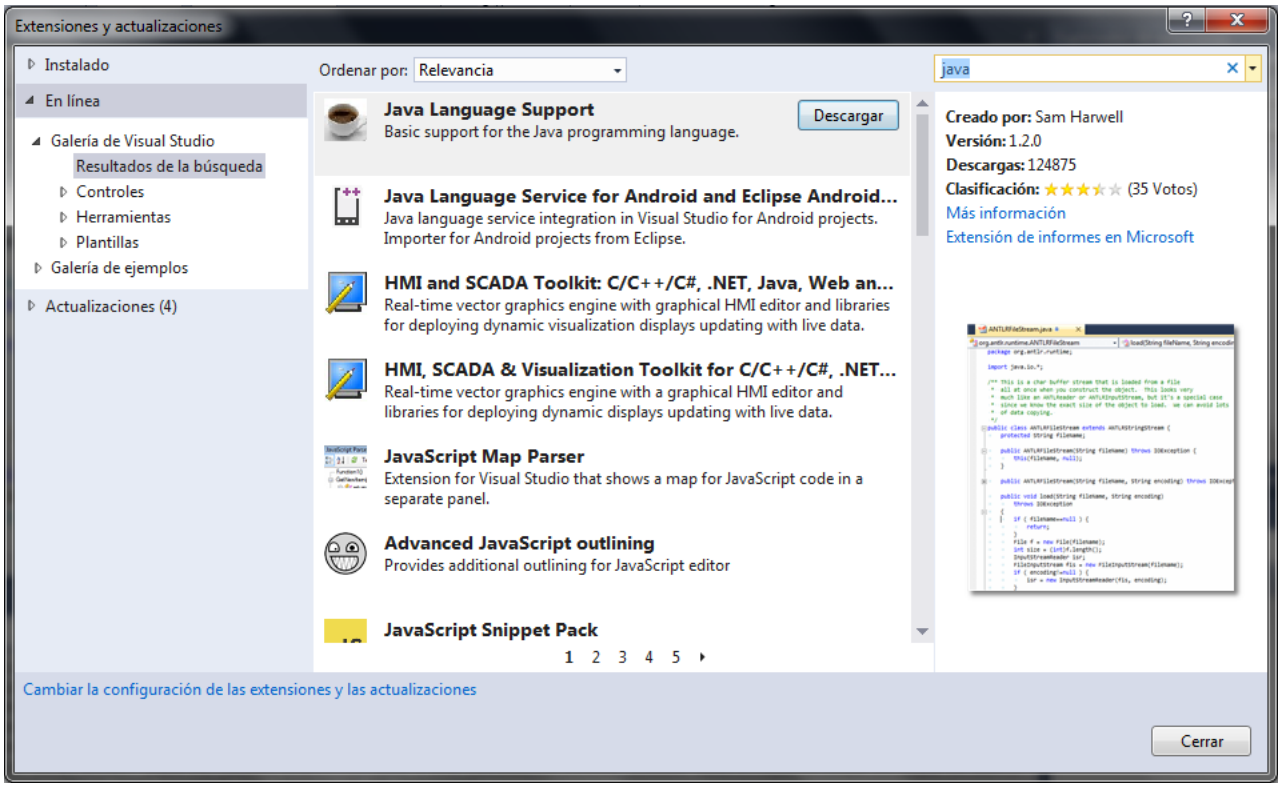
3.2.5 PLUGINS

En el IDE de Visual Studio los plugin son llamados extensiones y para poder descargar e instalar cualquier extensión debemos seguir los siguientes pasos:

En la barra de menú, elija Herramientas, Extensiones y actualizaciones.

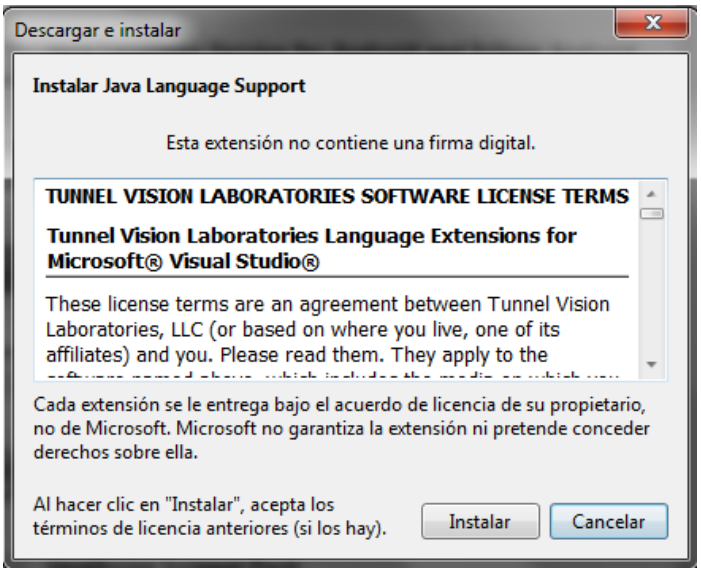


En la lista izquierda, elegiremos la opción **En línea** y Galería de Visual Studio o Galería de ejemplos, y utilizaremos las opciones de *Ordenar por* o *Buscar* para encontrar la extensión que desea en la lista de resultados.



A continuación, elegimos la extensión que deseamos agregar y hacemos clic en el botón Descargar para instalar la extensión.

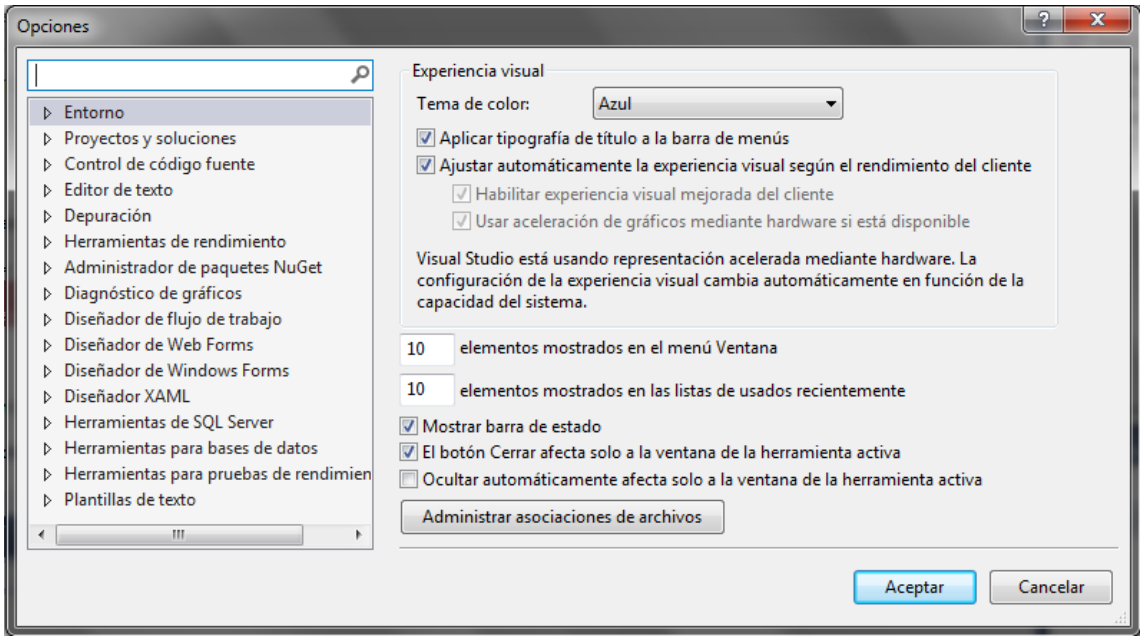
Nos aparecerá la siguiente ventana en la que debemos aceptar los términos de licencia.



Por último, reiniciamos la aplicación para que surjan efecto los cambios.

3.2.6 PANEL DE CONFIGURACION

Para acceder a la configuración de Visual Studio debemos ir a la pestaña **Herramientas → opciones.**



El panel de opciones está dividido en dos partes: un panel de navegación a la izquierda y un área de presentación a la derecha. El panel de navegación está formado por un árbol de

directorios, como Entorno, Editor de texto, Proyectos y soluciones, y Control de código fuente. Para acceder a las opciones de cada carpeta debemos expandir cualquier la carpeta deseada para ver las páginas de opciones que contiene. Cuando selecciona una de las carpetas de una página determinada, sus opciones aparecen en el área de presentación.

Las opciones para una característica del IDE no aparecen en el panel de navegación hasta que la característica no está cargada en memoria. Por tanto, puede que no se muestren las mismas opciones cuando inicia una nueva sesión que cuando finalizó la última sesión. Cuando crea un proyecto o ejecuta un comando que utiliza una aplicación determinada, se agregan carpetas para las opciones correspondientes al cuadro de diálogo Opciones. Estas opciones agregadas permanecerán disponibles mientras la característica del IDE esté en memoria.

Al hacer clic en *Aceptar*, en el cuadro de diálogo *Opciones* se guardan todas las configuraciones de todas las páginas.

3.3 ECLIPSE

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

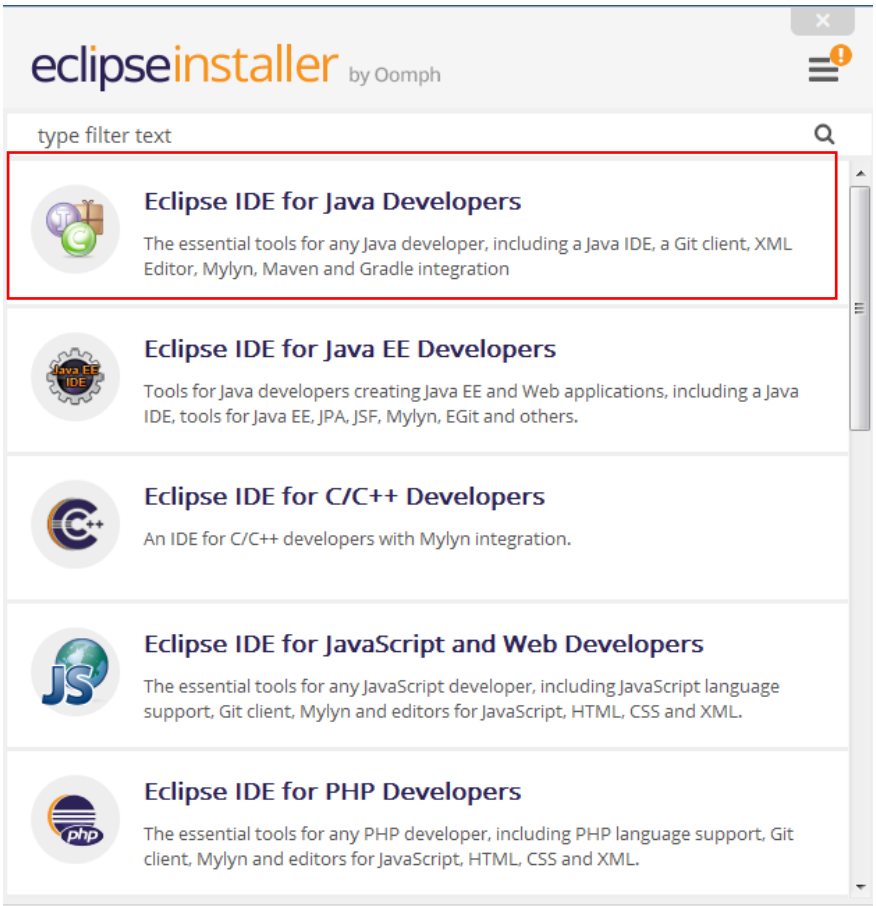
Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

3.1.3 INSTALACION

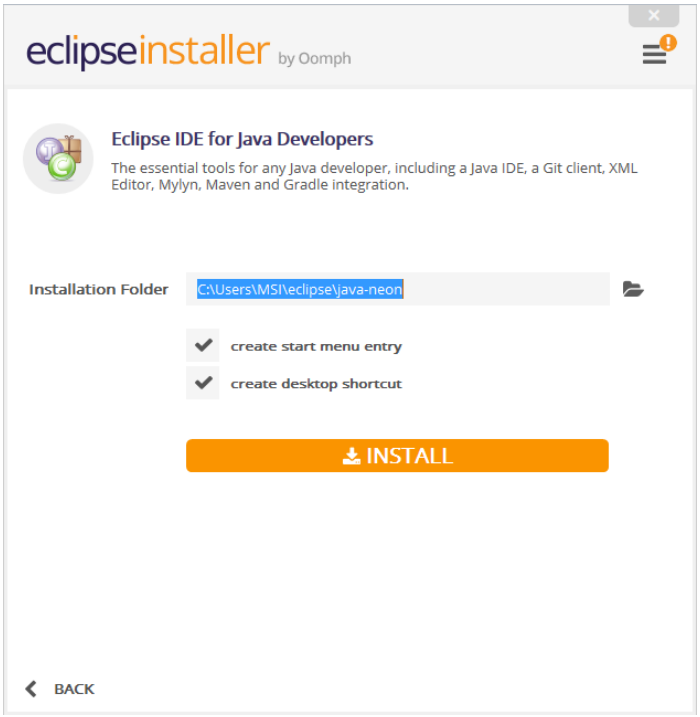
A continuación, veremos el proceso de instalación de Eclipse Neón de 64 bit en español paso a paso.

Una vez descargado, hacemos doble clic sobre el paquete y se nos abrirá la siguiente ventana, donde prepara todo lo necesario para su correcta instalación.

A continuacion se nos abrira un asistente diferente empaquetado que incorporar funcionalidades según el lenguae a desarrollar, nosotros hemos elegido la primera opcion.

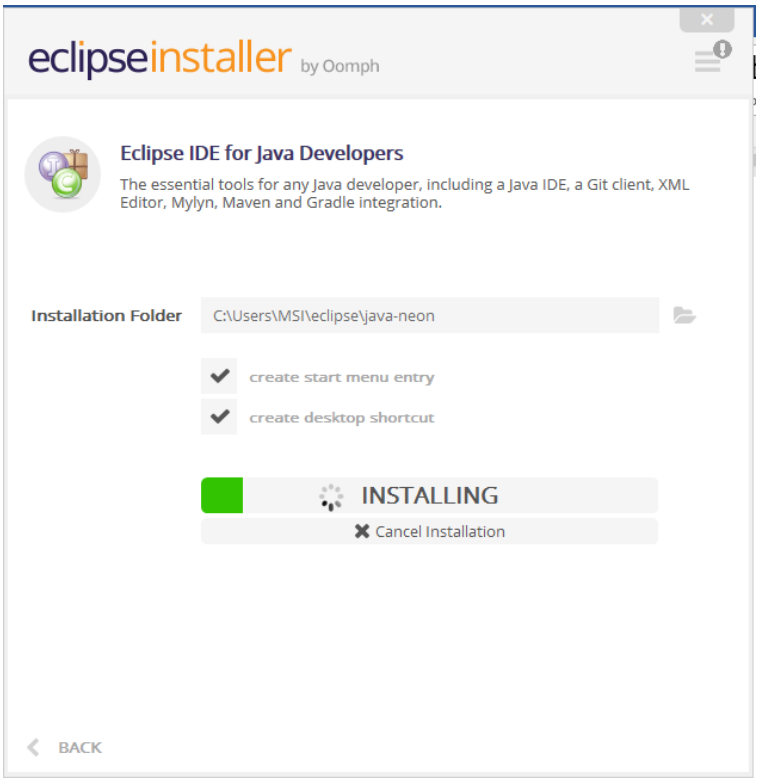


A continuacion elegiremos la carpeta de ubicación de nuestro IDE, y marcaremos la opcion de que nos cree el acceso directo en el escritorio.

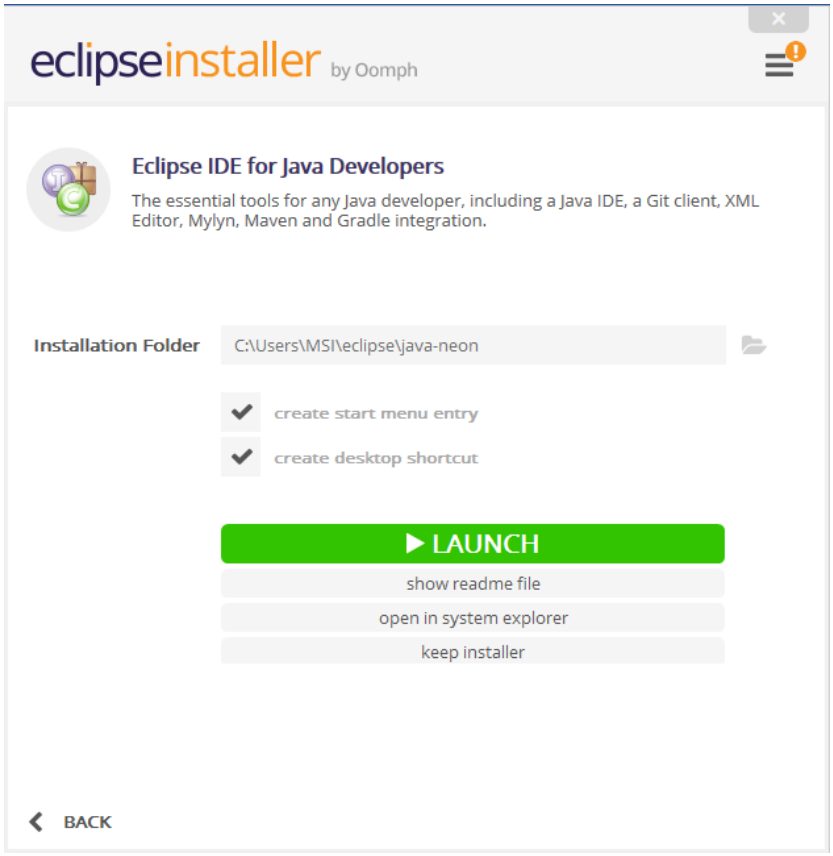


Posteriormente aceptaremos los términos de uso si estamos de acuerdo.

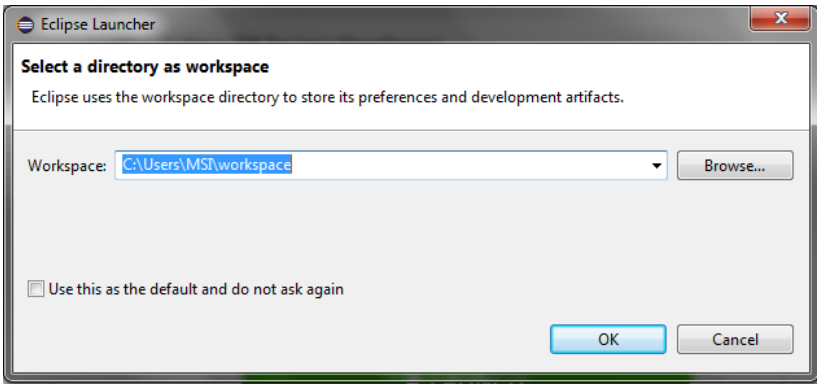
Y comenzaremos con la instalación de nuestro entorno



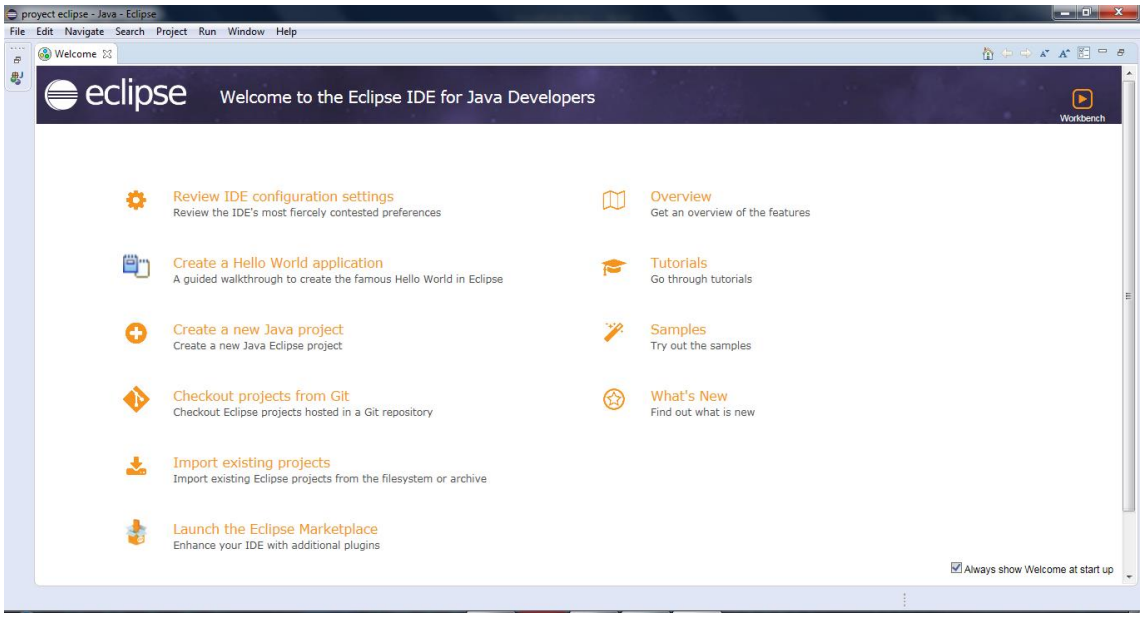
Una vez instalados lo inicializamos



Antes de abrirse nos pedira que le indiquemos donde guardar los proyectos que realicemos.

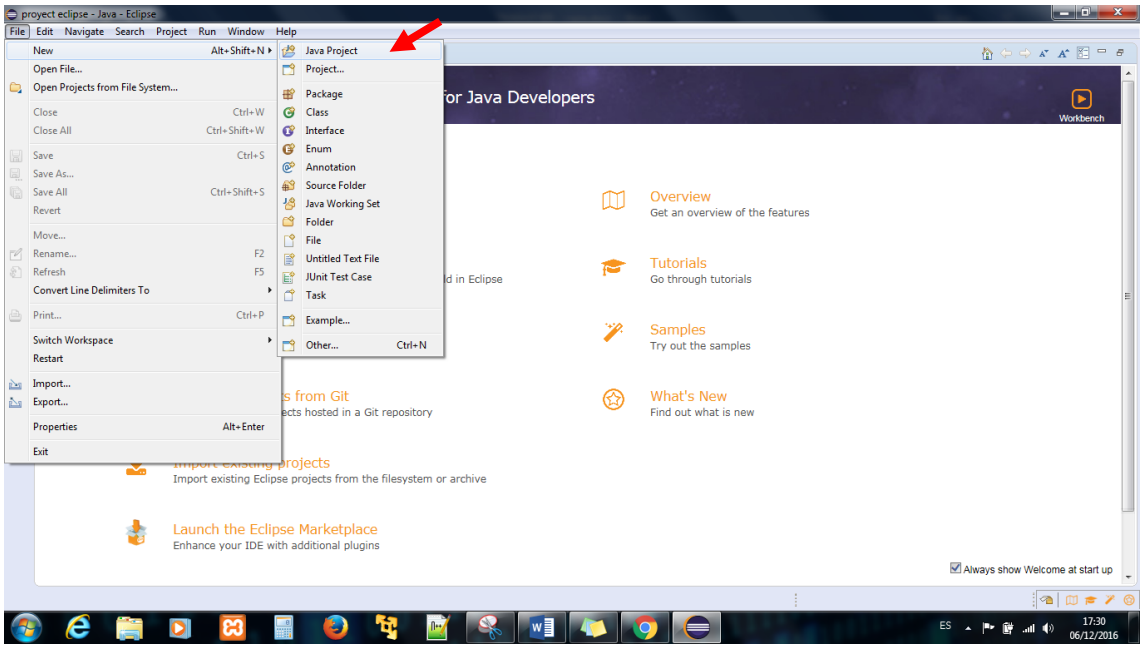


Ya tenemos nuestro entorno de desarrollo.

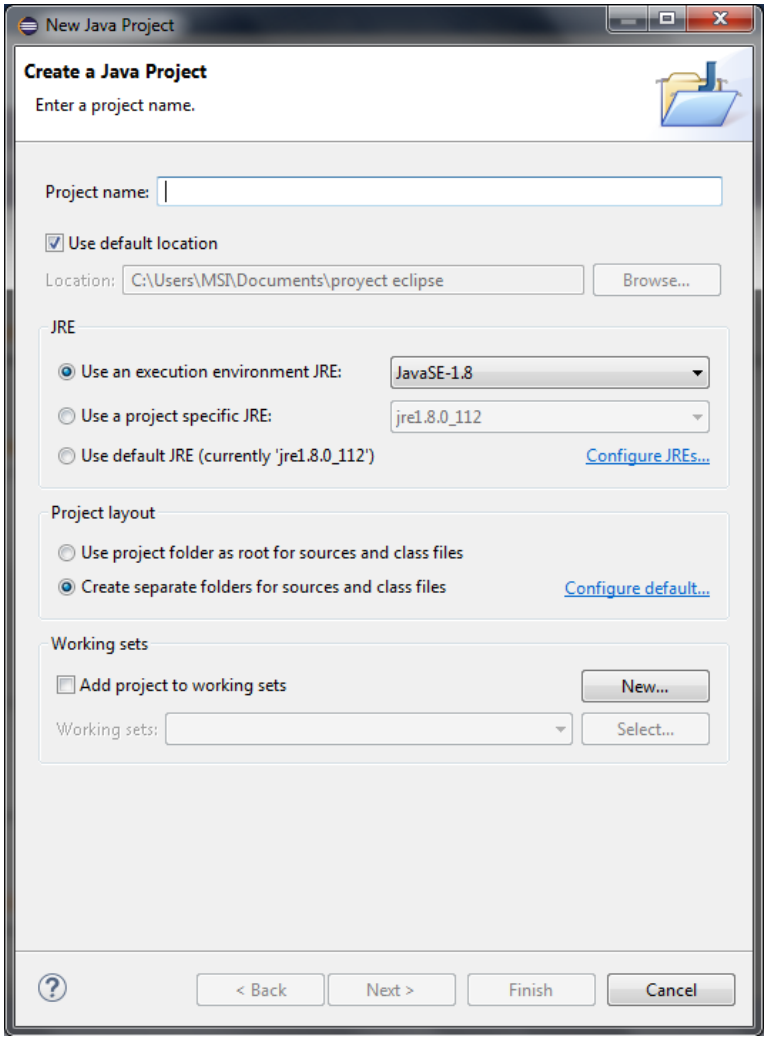


Ahora vamos a crear un nuevo proyecto para ver algunas de sus herramientas.

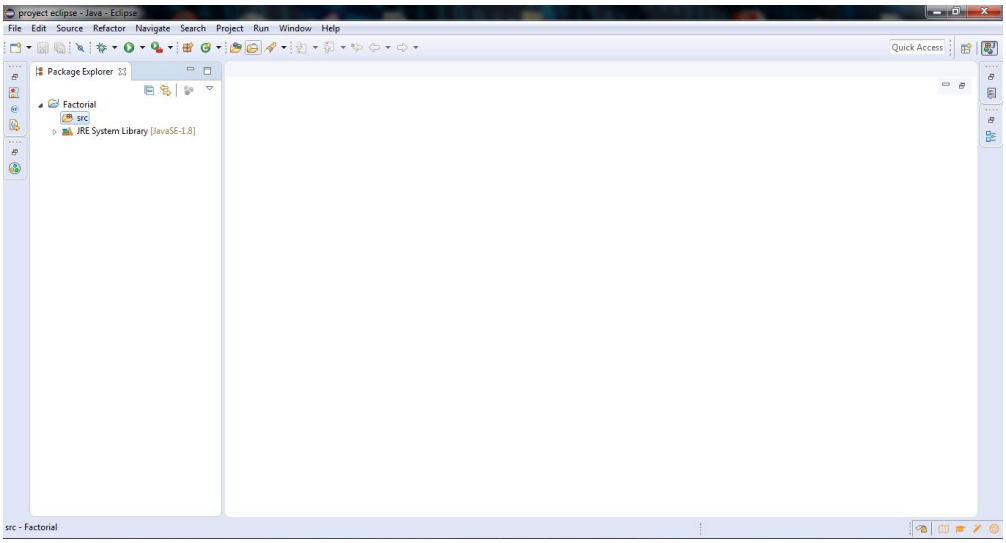
Para iniciar un nuevo proyecto iremos a la pestaña **File → New → Java Project.**



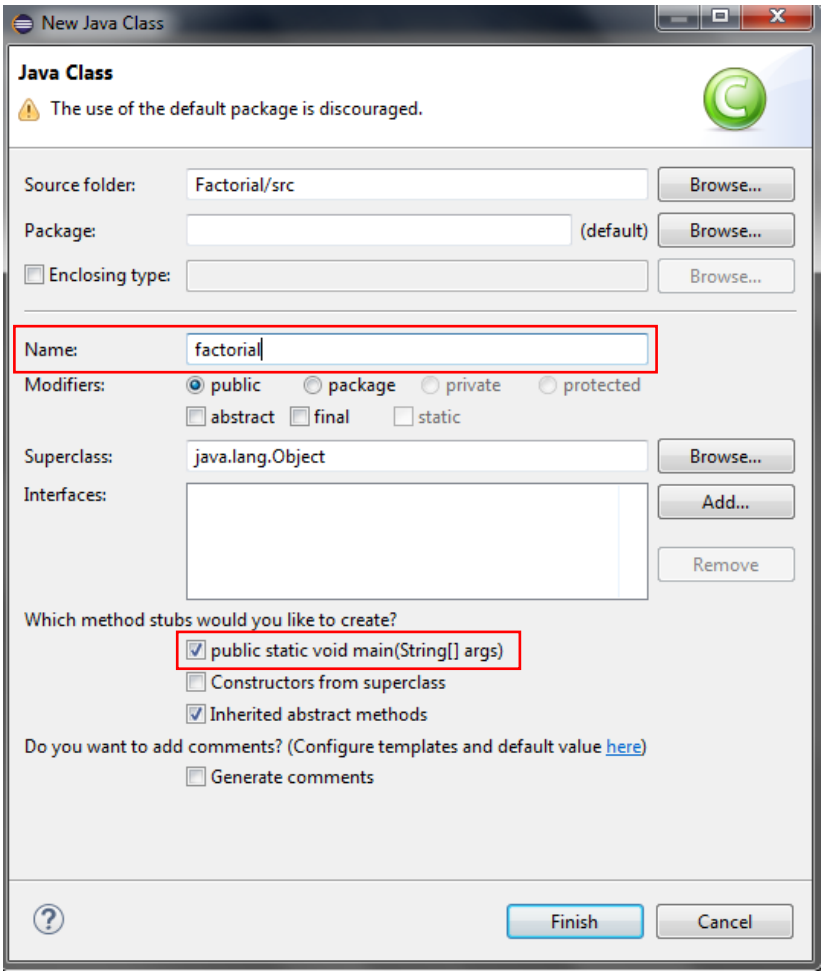
Se nos abrirá la siguiente ventana en la que debemos indicar el nombre del proyecto y si modificar la ubicación de este si lo deseamos.



Una vez creado el proyecto, nos aparecerá la pantalla principal del proyecto.

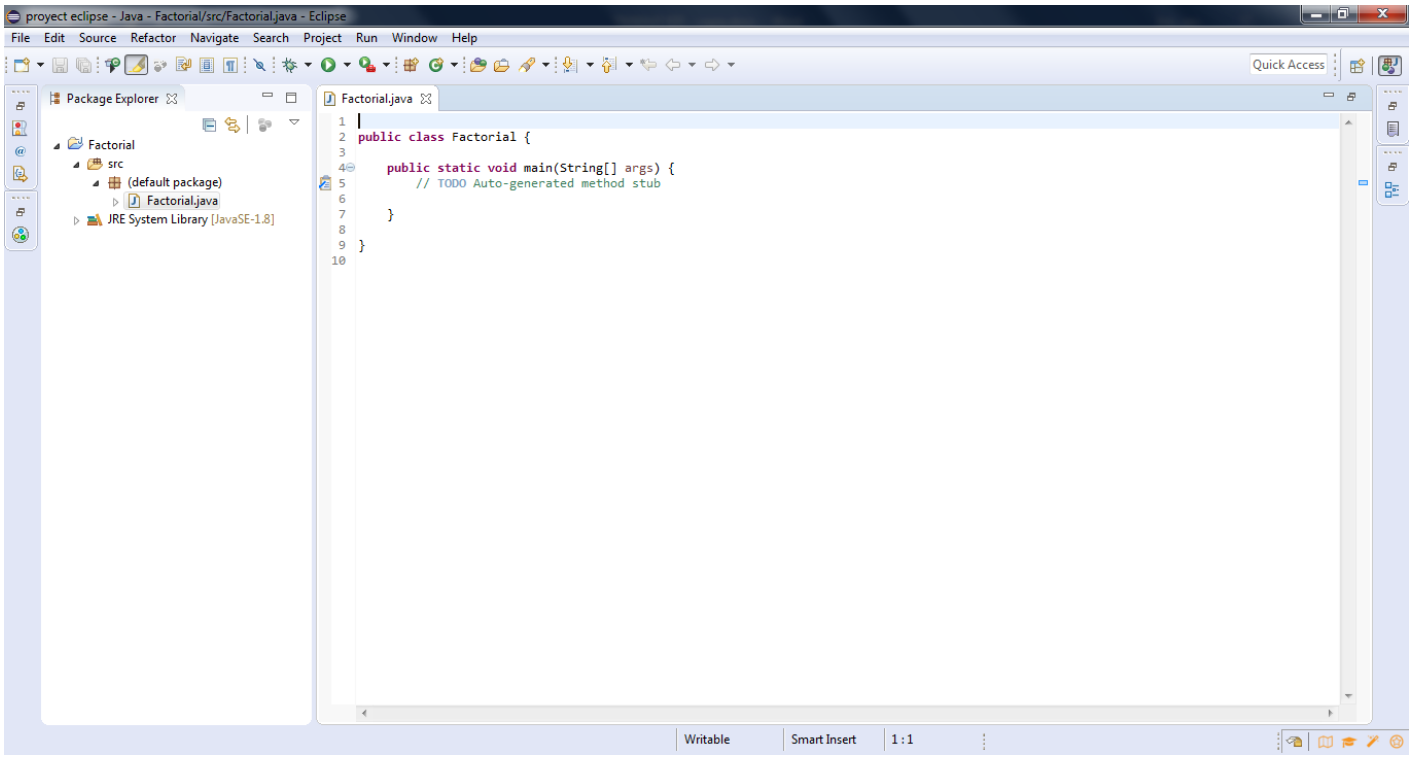


Para poder comenzar a escribir nuestro programa lo primero que debemos hacer es crear la clase principal de nuestro algoritmo. Para ello, simplemente hacemos clic en el siguiente icono  y se nos abrirá una ventana donde debemos indicar el nombre el nombre de la clase y marcar la casilla que creará el método.

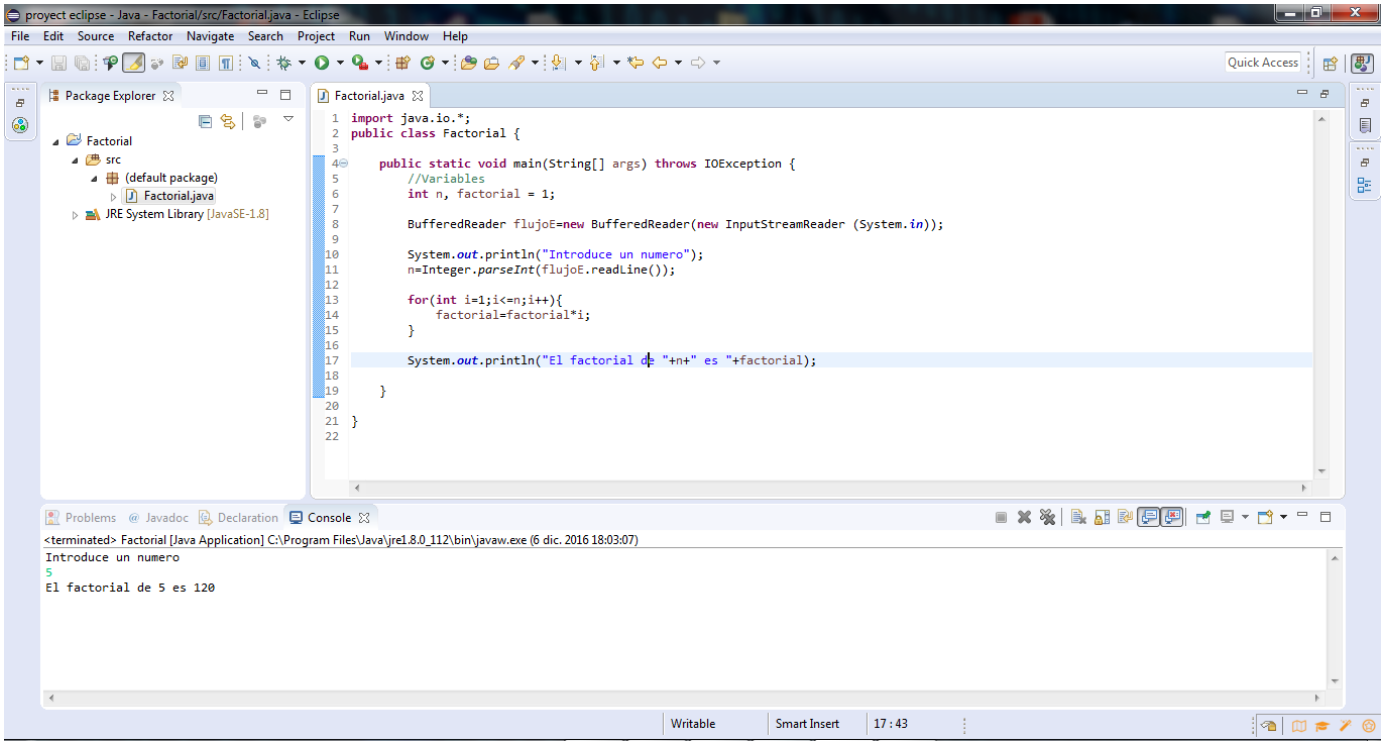


3.3.2 EDITOR DE TEXTO

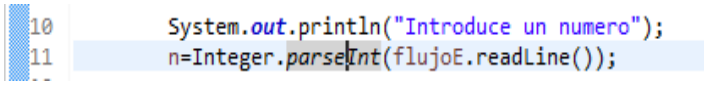
Ya tenemos aquí nuestro editor de texto con la clase main ya construida.

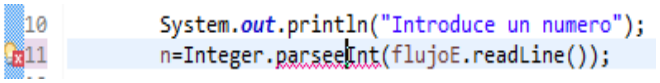


A continuación, debemos escribir nuestro programa. El editor de texto nos ayuda según vamos escribiendo, e irá marcando con un subrayado rojo los posibles errores de sintaxis y posibles errores léxicos.

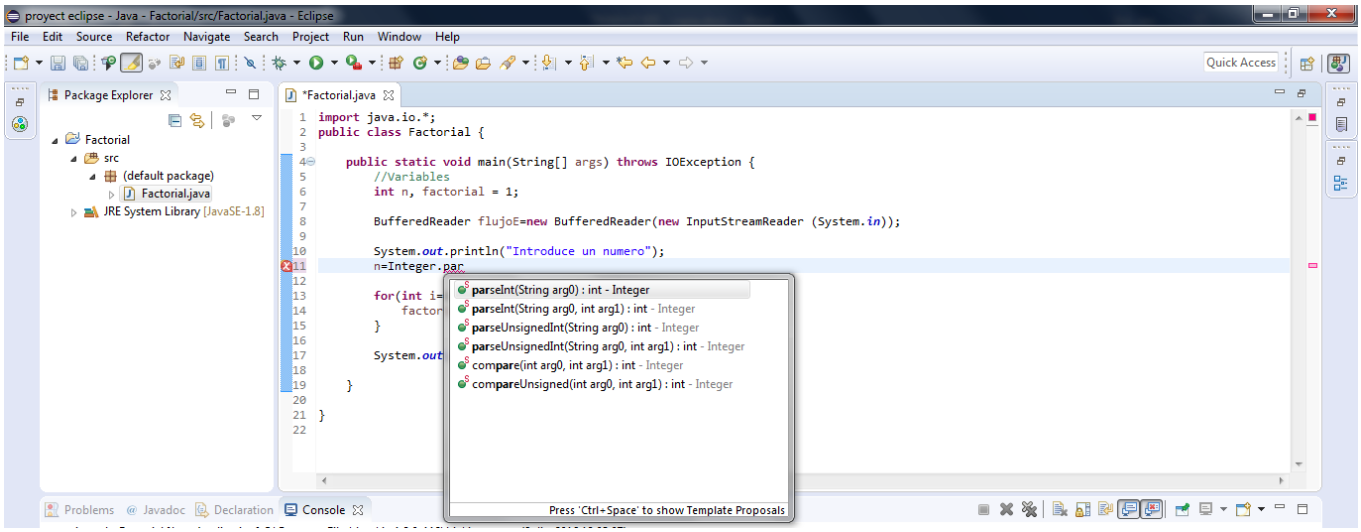


También podemos distinguir que nos muestra de forma resaltada las palabras reservadas del lenguaje.



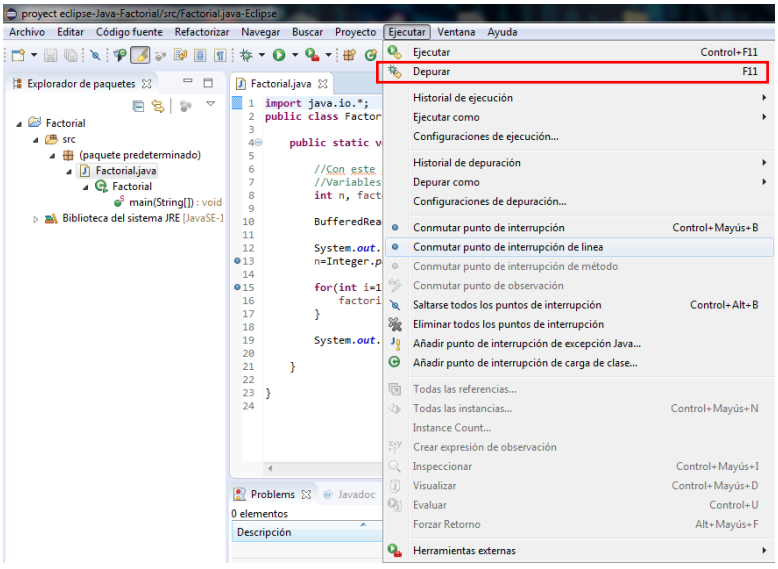


El error descrito anteriormente es difícil de cometer puesto que como podemos ver en la imagen inferior, el propio editor nos muestra las posibles opciones disponibles según vamos escribiendo.



3.3.3 COMPILADOR

Para ejecutar el programa debemos compilar nuestro código, es decir, transformar nuestro código fuente en un código binario o código máquina y para ello, tenemos varias formas de hacerlo, a través del siguiente botón  o desde la pestaña ejecutar o con Ctrl+F11



En el caso de eclipse solo hay una forma de compilar y es que cuando hacemos clic en ejecutar para observar el resultado del código, este mismo por debajo está creando el archivo .java.

3.3.4 DEPURADOR

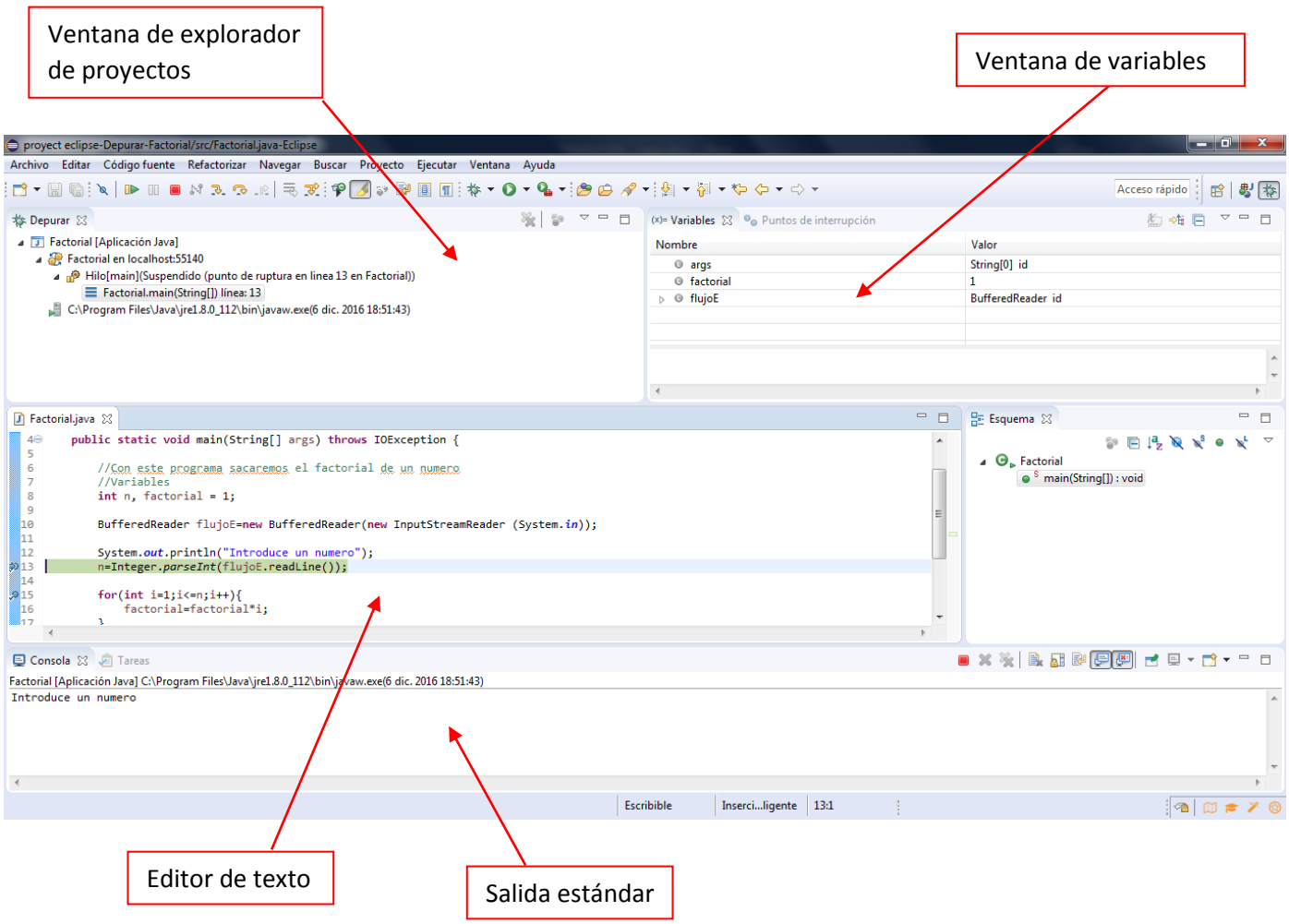
El depurador o debugger nos permite ejecutar el código fuente paso a paso para identificar posibles errores de forma más sencilla y rápida facilitándonos su búsqueda y obtener una forma más eficiente de desarrollo.

Para ello, nos permite utilizar puntos de ruptura o breakpoint, para detener la ejecución del programa y poder observar el estado de las variables, su valor e incluso modificarlo sobre la marcha y continuar la ejecución.

Hay varias formas de iniciar la ejecución del programa en modo depuración:

Podemos hacerlo mediante la pestaña **Ejecutar → Depurar.**

Podemos hacerlo con un atajo de teclado (**F11**) o pulsando sobre el botón: 



Para el uso del depurador manejamos distintas opciones a través de botones:

	**Step Over (F6)** Ejecuta una línea de código.
	**Step Into (F5)** Ejecuta una línea de código. Si la instrucción es una llamada a un método, salta al método y continúa la ejecución por la primera línea del método.
	**Reanudar (F8)** La ejecución del programa continúa hasta el siguiente punto de ruptura. Si no existe un punto de ruptura se ejecuta hasta el final.
	**Detener (Ctrl + F2).** Termina la depuración del programa.

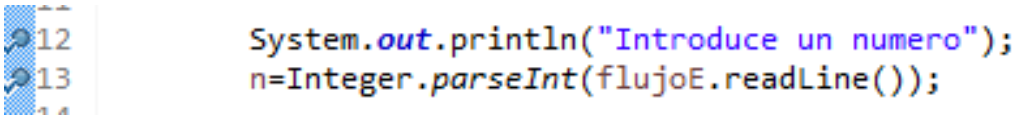
3.3.4.1 ESTABLECER PUNTOS DE RUPTURA

Un punto de ruptura es una marca que indica al depurador que debe detenerse cuando la ejecución del programa llegue a ella.

Cuando el programa se detiene podemos:

- Examinar los valores actuales de las variables.
- Continuar la depuración línea a línea del programa.

Para fijar un punto de ruptura simplemente pulsamos sobre la parte izquierda del número de línea donde se desea colocar.



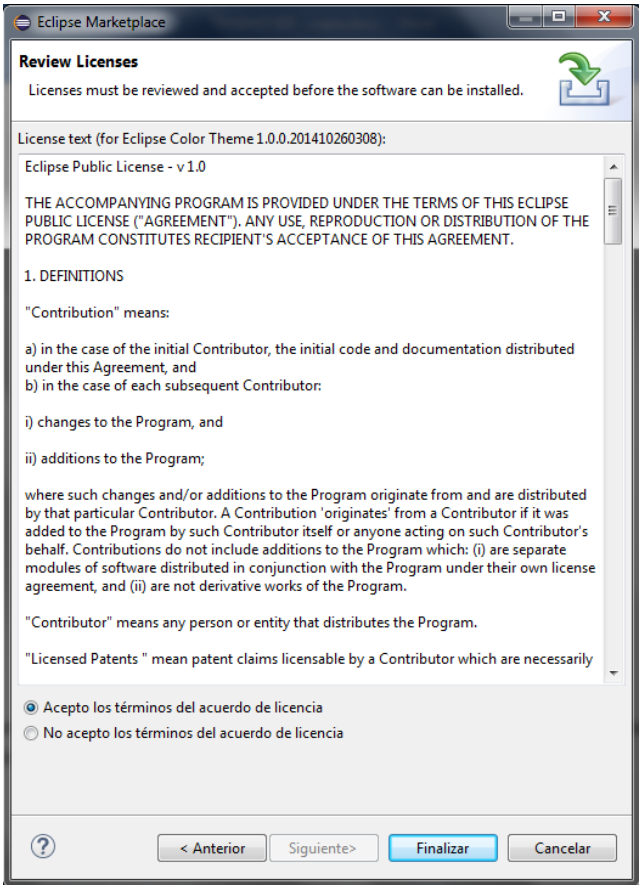
3.3.5 PLUGINS

Eclipse nos ofrece una serie de plugins con los que podremos trabajar (la lista de plugins que podemos instalar la podemos encontrar en: https://marketplace.eclipse.org/).

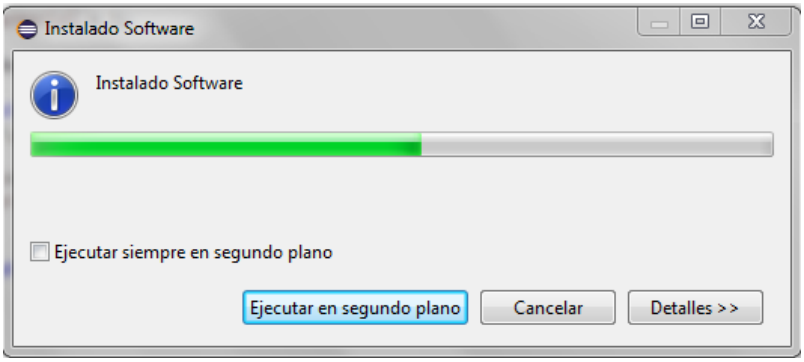
Para descargar e instalar cualquier plugin en el entorno de eclipse tenemos que ir a **Ayuda → eclipseMarketplace.**

Podríamos decir que es como una tienda de plugins, tan solo debemos buscar el plugin que queramos y hacer clic en instalar.

Cuando hagamos clic en aceptar nos aparecerá una nueva ventana en la cual tenemos que aceptar los términos de licencia.



Y comenzará la instalación de nuestro plugin.

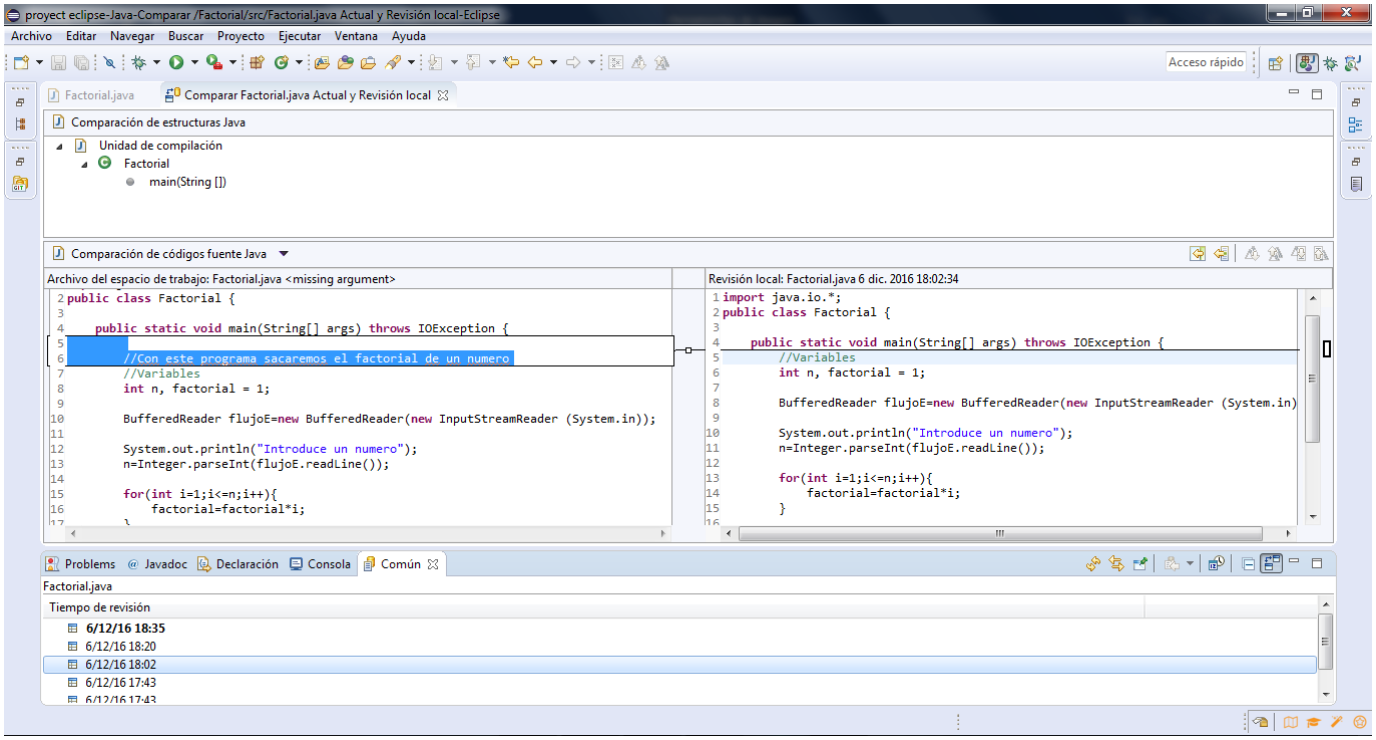


Por último, solo nos queda reiniciar el IDE para aplicar los cambios.

3.3.6 CONTROL DE VERSIONES

Eclipse dispone de una herramienta muy útil para los desarrolladores de software, es el control de versiones. Esta herramienta nos muestra los cambios sufridos en nuestro código fuente a lo largo de su codificación.

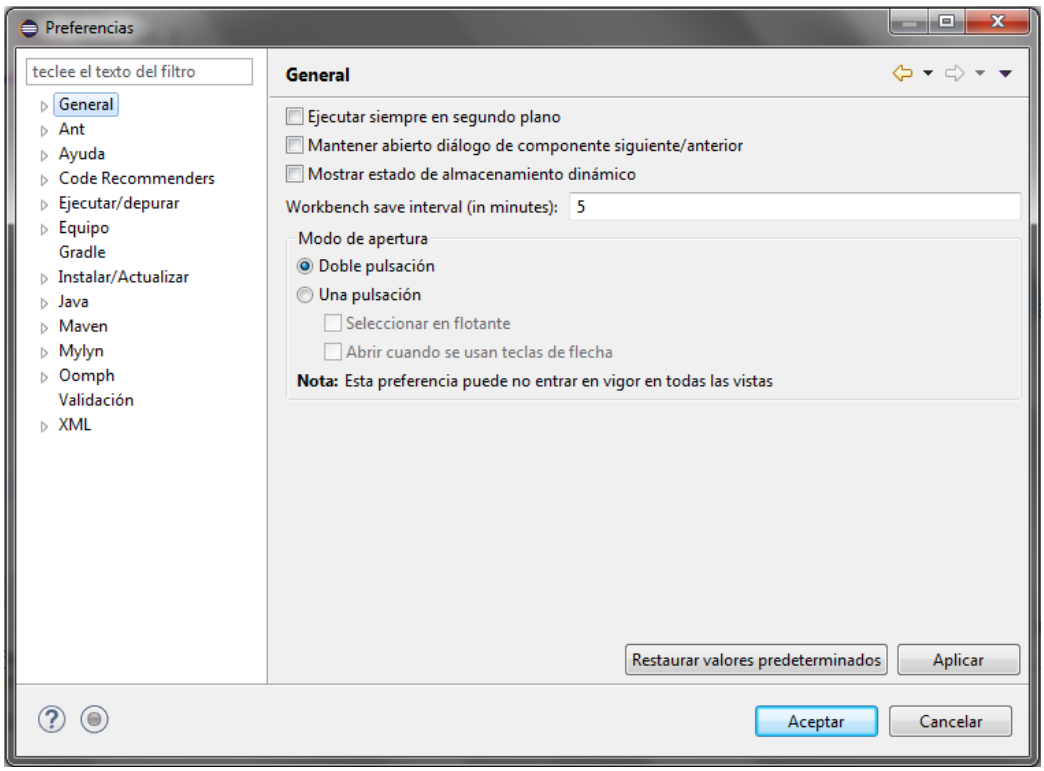
Para acceder a dicha herramienta simplemente pulsamos clic derecho sobre el paquete que queremos ver y hacemos clic en la opción **comparar**. Nos aparecerán las distintas versiones que han sido guardadas a lo largo del desarrollo, ahora solo tenemos que seleccionar la versión que queremos mostrar.



Como podemos observar en la imagen de arriba, al seleccionar una de las versiones nos destaca visualmente cuales han sido los cambios entre la versión seleccionada y la última disponible.

3.3.7 PANEL DE CONFIGURACION

Para acceder a la configuración de Eclipse debemos ir a la pestaña **Ventanas → preferencias.**

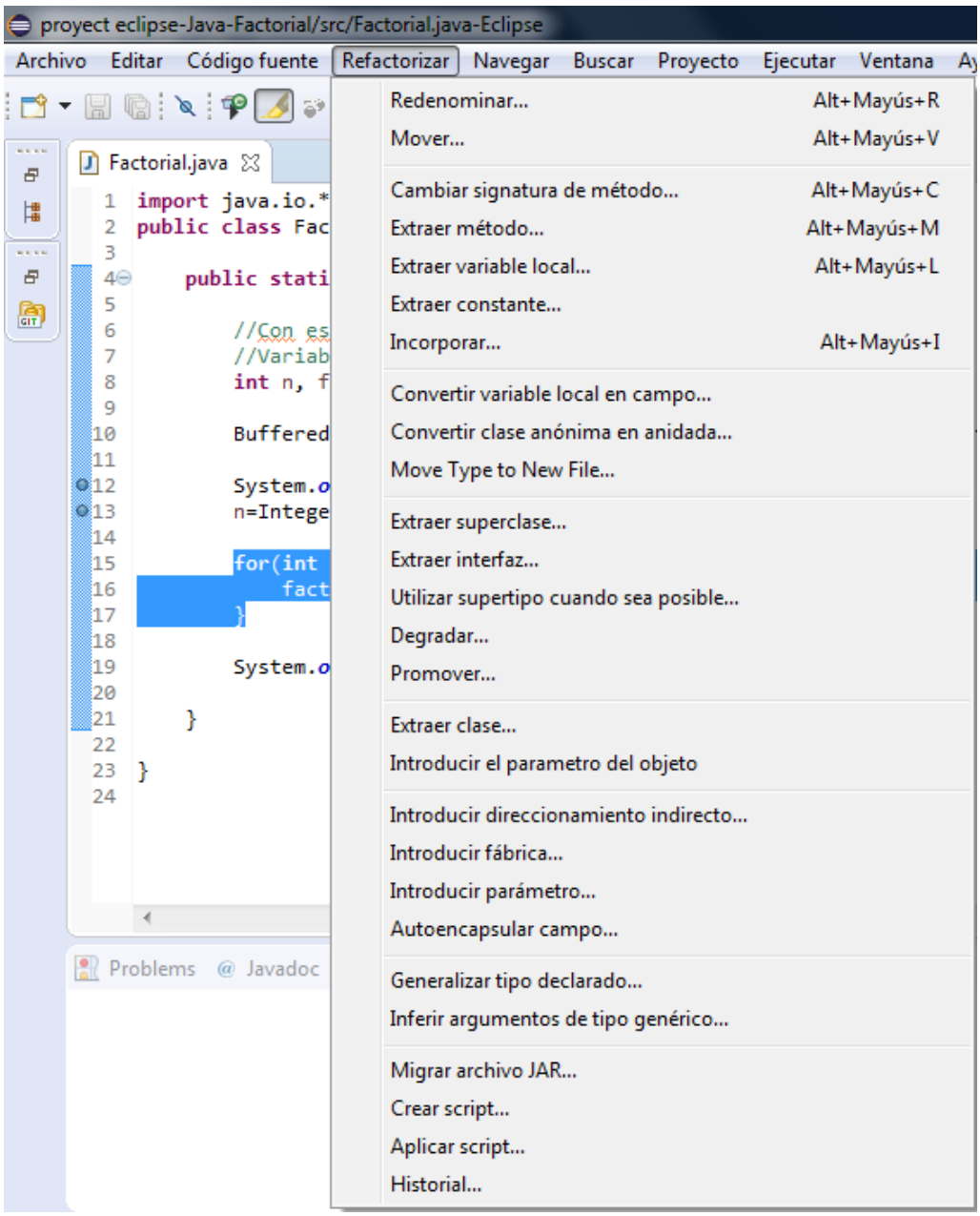


El panel de opciones está dividido en dos partes: un panel de navegación a la izquierda y un área de presentación a la derecha. El panel de navegación está formado por un árbol de directorios, como General, Validación, Equipo, Ayuda... Para acceder a las opciones de cada carpeta debemos expandir cualquier la carpeta deseada para ver las páginas de opciones que contiene. Cuando selecciona una de las carpetas de una página determinada, sus opciones aparecen en el área de presentación.

Al hacer clic en Aplicar en el cuadro de diálogo se guardan todas las configuraciones de todas las páginas y tan solo nos queda hacer clic en aceptar para salir.

3.3.8 REFACTORIZACION

La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. El objetivo, por el contrario, es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto, para facilitar el mantenimiento en el futuro. Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede refactorizarlo primero para facilitar esta tarea y luego añadir el nuevo comportamiento.



En esta pestaña disponemos de opciones como cambiar el nombre del paquete principal, mover, copiar y eliminar código, seleccionando una parte de código lo podemos reestructurar creando métodos, etc.

3.4 INTELLIJ IDEA

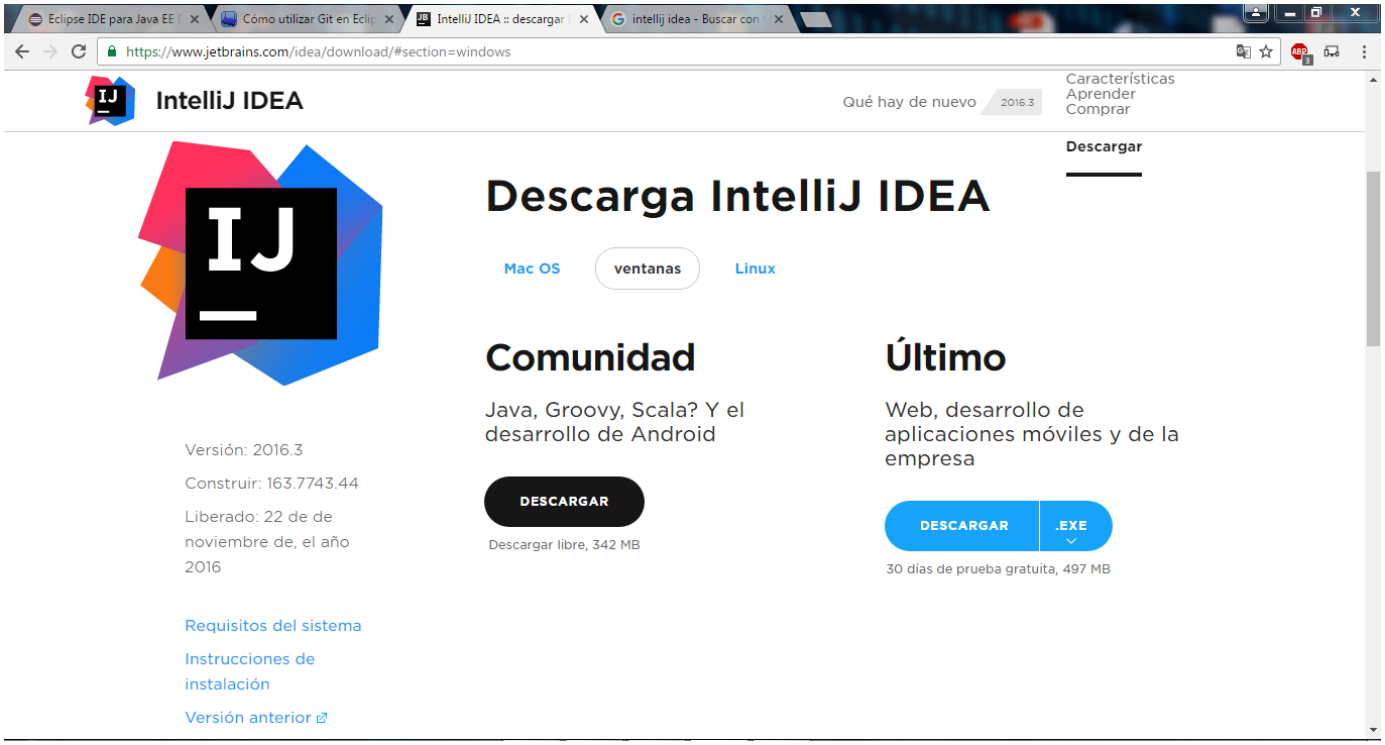
IntelliJ IDEA es un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Es desarrollado por JetBrains (anteriormente conocido como IntelliJ), y está disponible en dos ediciones: community edition, y edición comercial.

La primera versión de IntelliJ IDEA fue liberada en enero de 2001, y en ese tiempo fue uno de los primeros IDE para Java, estaba disponibles con avanzada navegación de código y capacidades de refactorización de código integrado.

En el 2010, IntelliJ recibió la puntuación de centro de prueba más alta fuera de las tres herramientas de programación superiores de Java: Eclipse, NetBeans y Oracle JDeveloper.

Su última versión es la 2016.3 y podemos encontrarla en su página oficial
https://www.jetbrains.com/idea/download/#section=windows

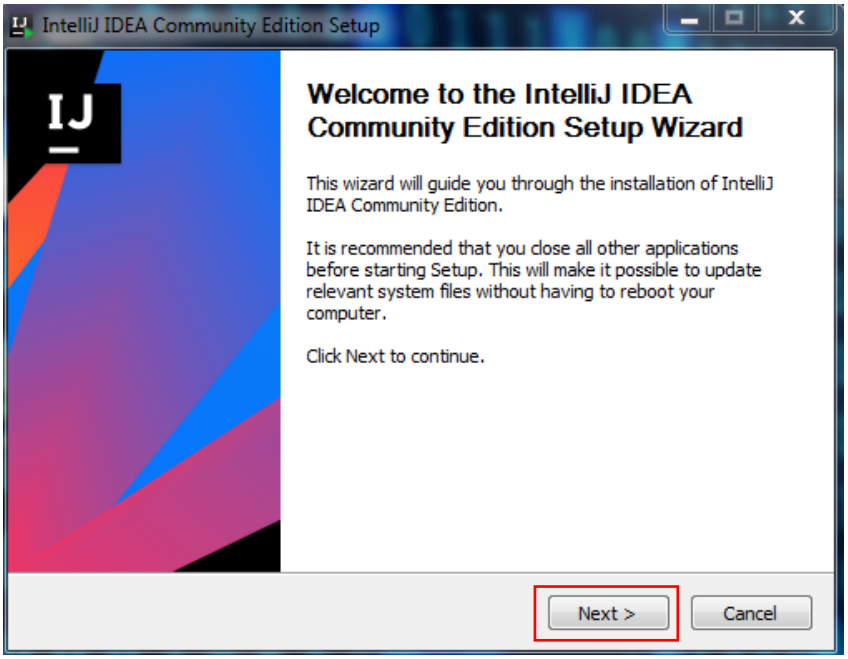
IntelliJ IDEA está disponible para Windows, Linux y Mac OS.



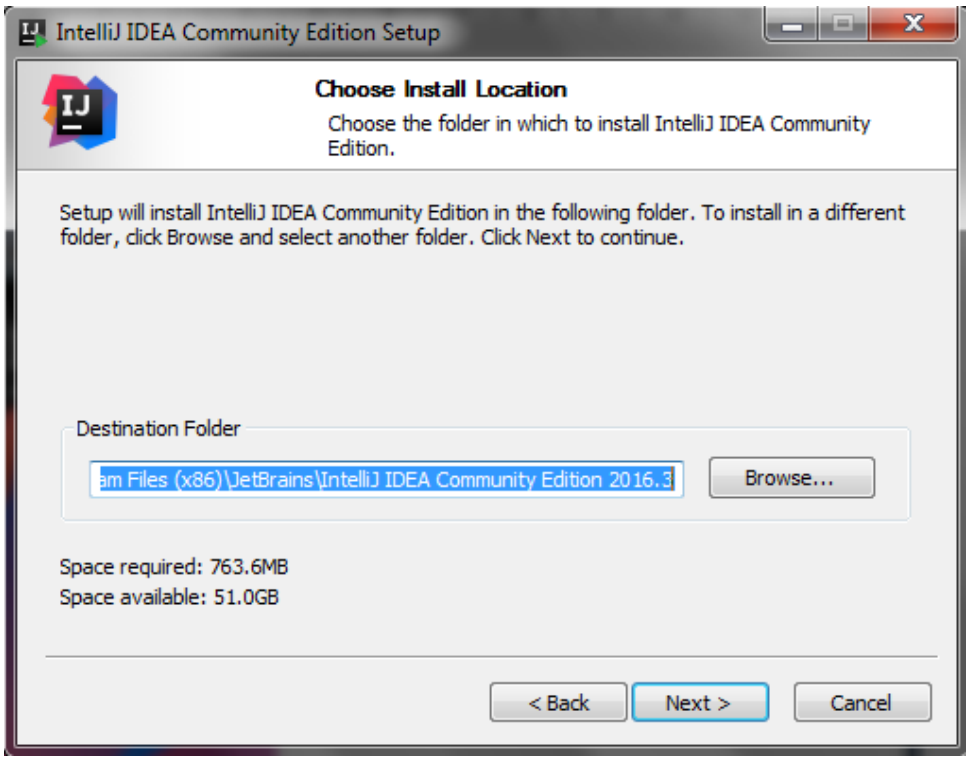
3.4.1 INSTALACION

A continuación, veremos el proceso de instalación de IntelliJ IDEA 2016.3 paso a paso.

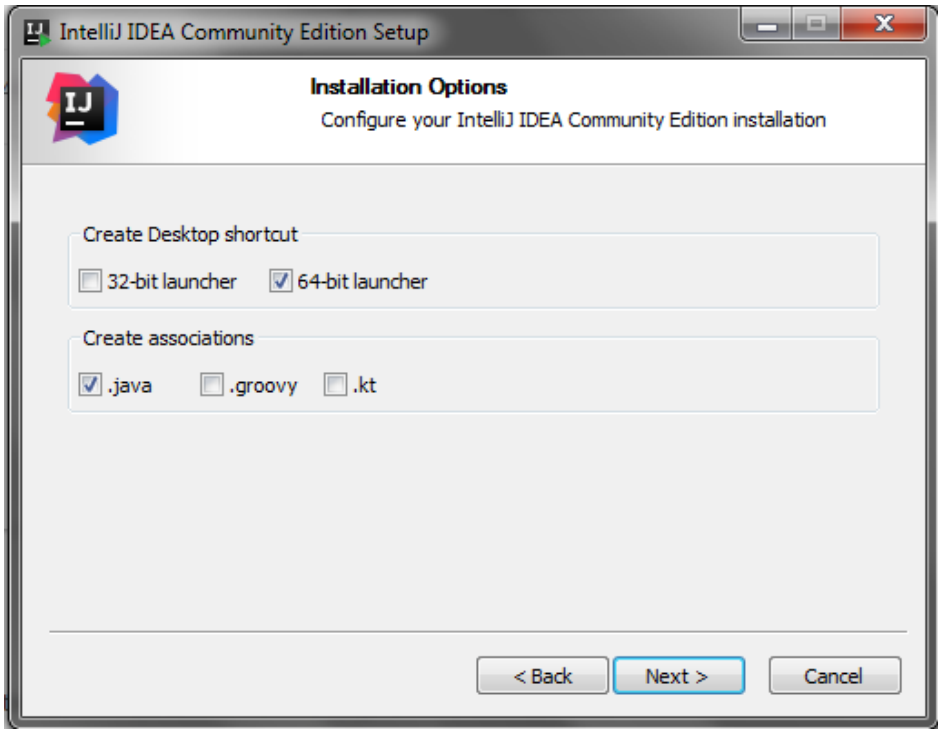
Una vez descargado, hacemos doble clic sobre el paquete y se nos abrirá la siguiente ventana, donde prepara todo lo necesario para su correcta instalación, nosotros seguiremos el asistente.



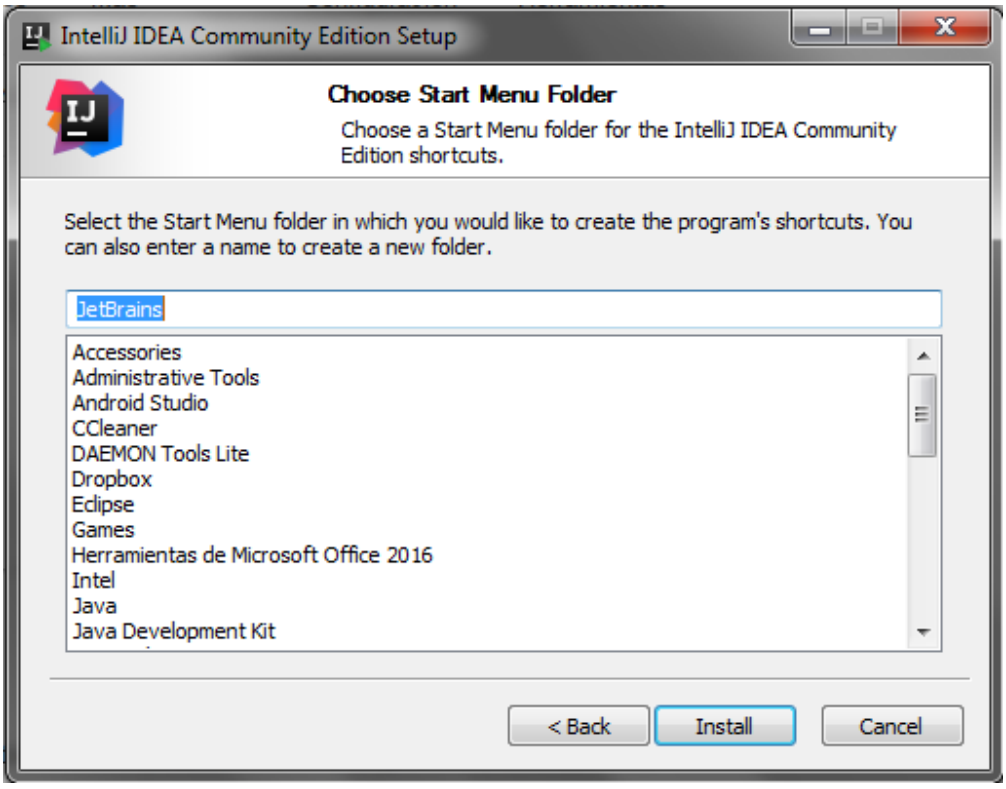
A continuación, elegiremos la ubicación donde va a ser instalado nuestro entorno. Nosotros lo dejaremos en la carpeta por defecto.



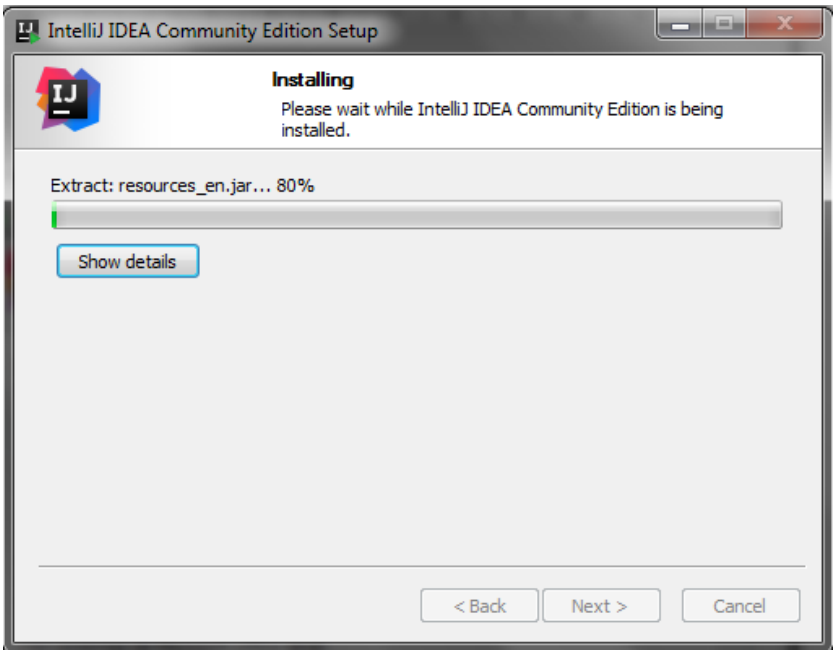
Posteriormente marcaremos la casilla de 64 bit para que nos cree el acceso directo en el escritorio y en mi caso, he asociado los archivos .java al programa.



Finalmente hacemos clic en instalar.

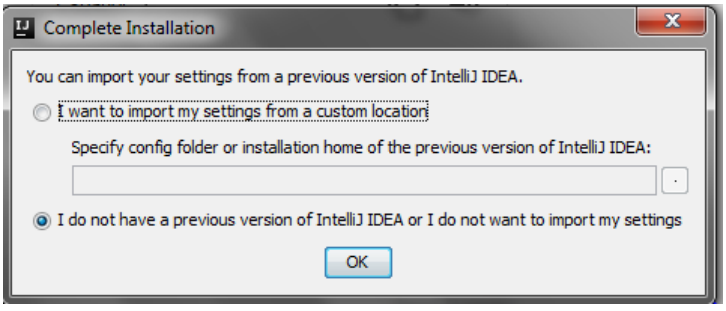


Como podemos ver en la imagen inferior, comienza el proceso de instalación.

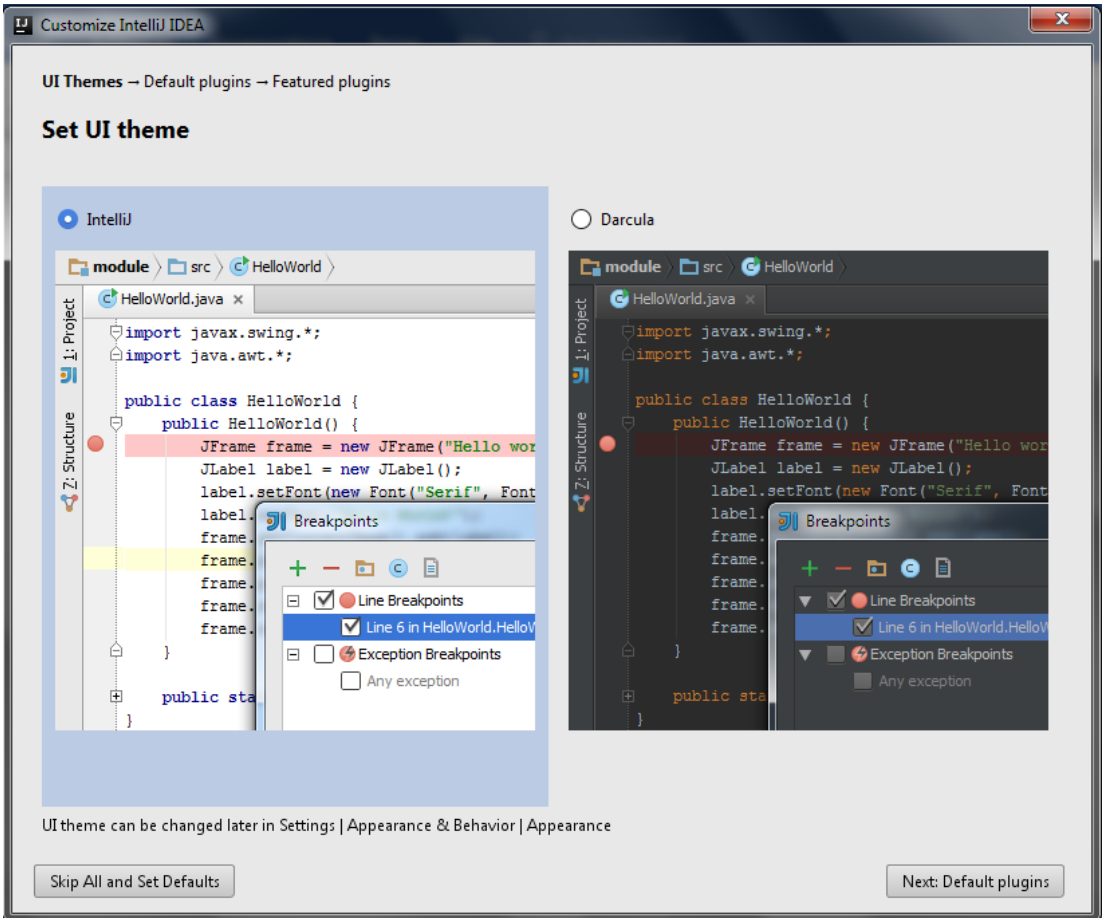


Una vez ha finalizado la instalación, marcamos la casilla de comenzar con el programa.

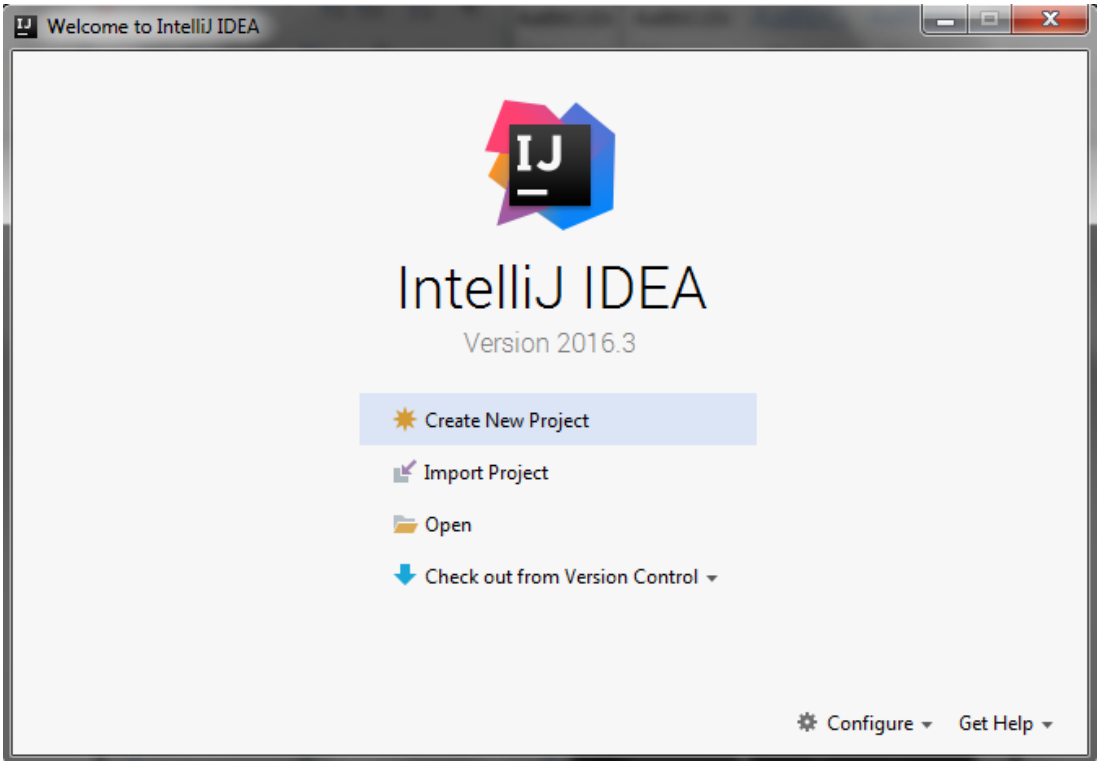
Antes de arrancar el programa nos pregunta si hemos tenido alguna versión anterior instalada, esto se debe a que, si la hubiésemos tenido, podríamos mantener la configuración.



Una vez arranca el programa nos pide que elijamos un tema para la apariencia de nuestro entorno.



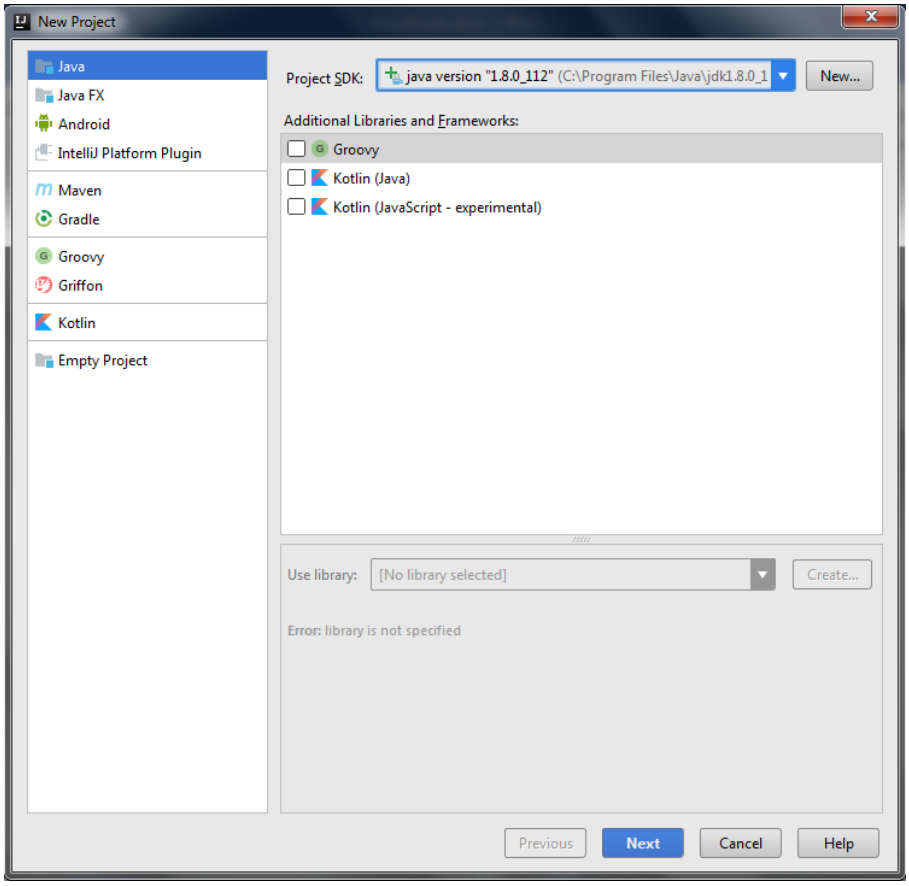
Y ya tenemos la pantalla principal de nuestro entorno.



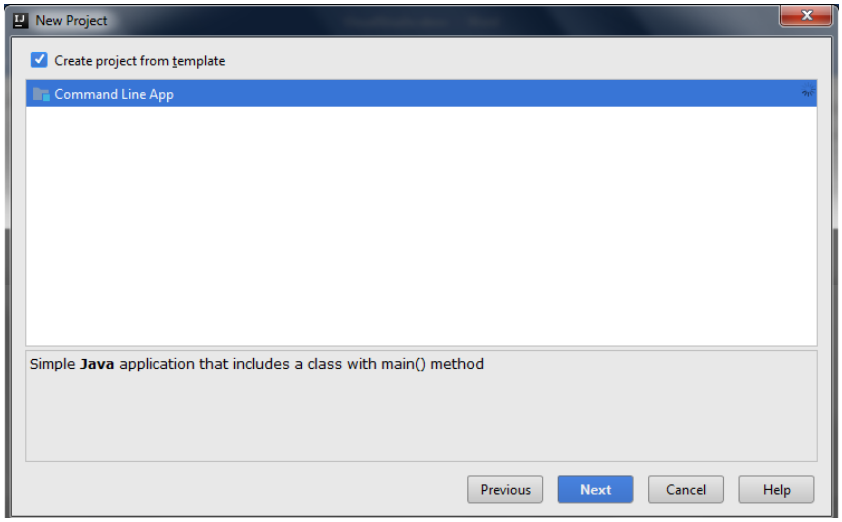
Ahora vamos a crear un nuevo proyecto para ver algunas de sus herramientas.

Para iniciar un nuevo proyecto haremos clic en **Create New Project.**

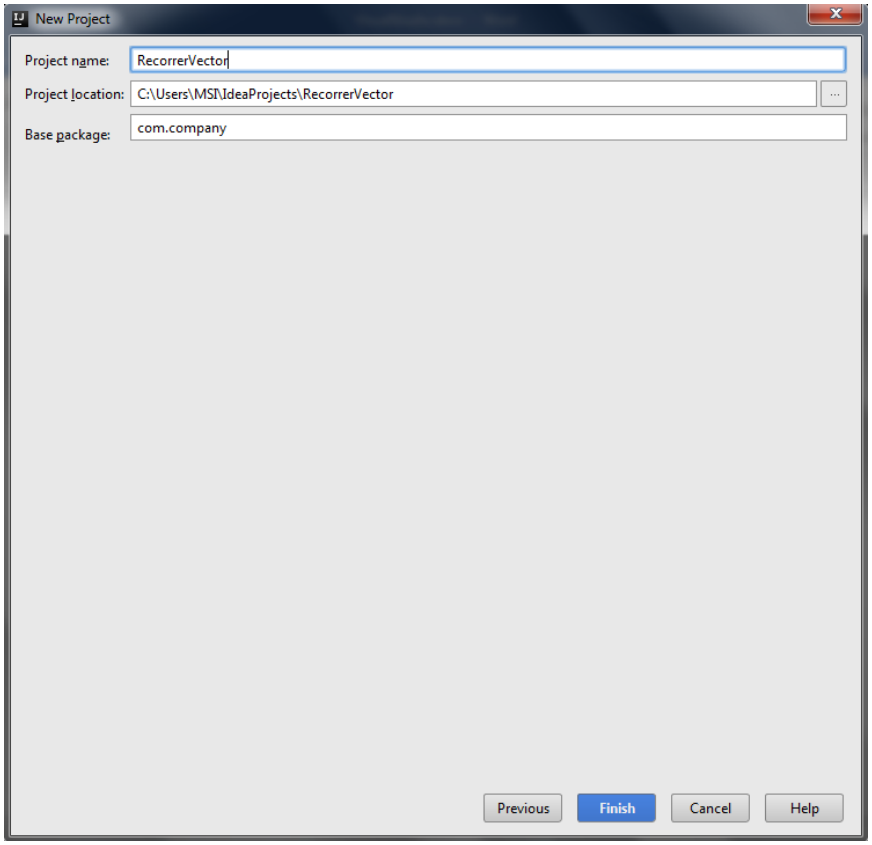
Nos aparecerá la siguiente ventana en la que debemos elegir el tipo de proyecto a realizar, en mi caso será un proyecto en el lenguaje java, por ello marco la opción java y pulso en Next.



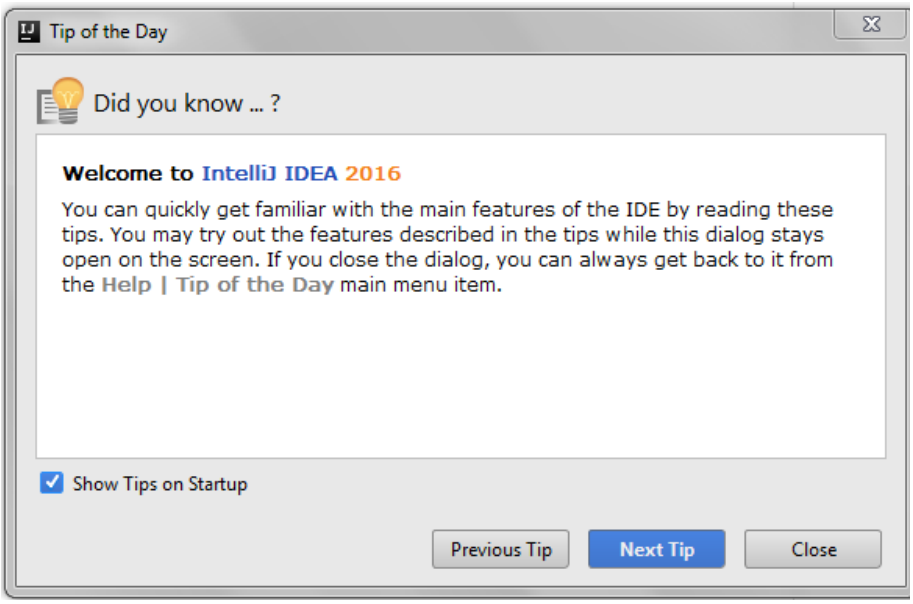
A continuación, no aparecerá otra ventana en la que viene marcada por defecto utilizar una plantilla, tan solo debemos hacer clic en Next.



En la siguiente ventana introducimos el nombre del proyecto y cual sera su ubicación.

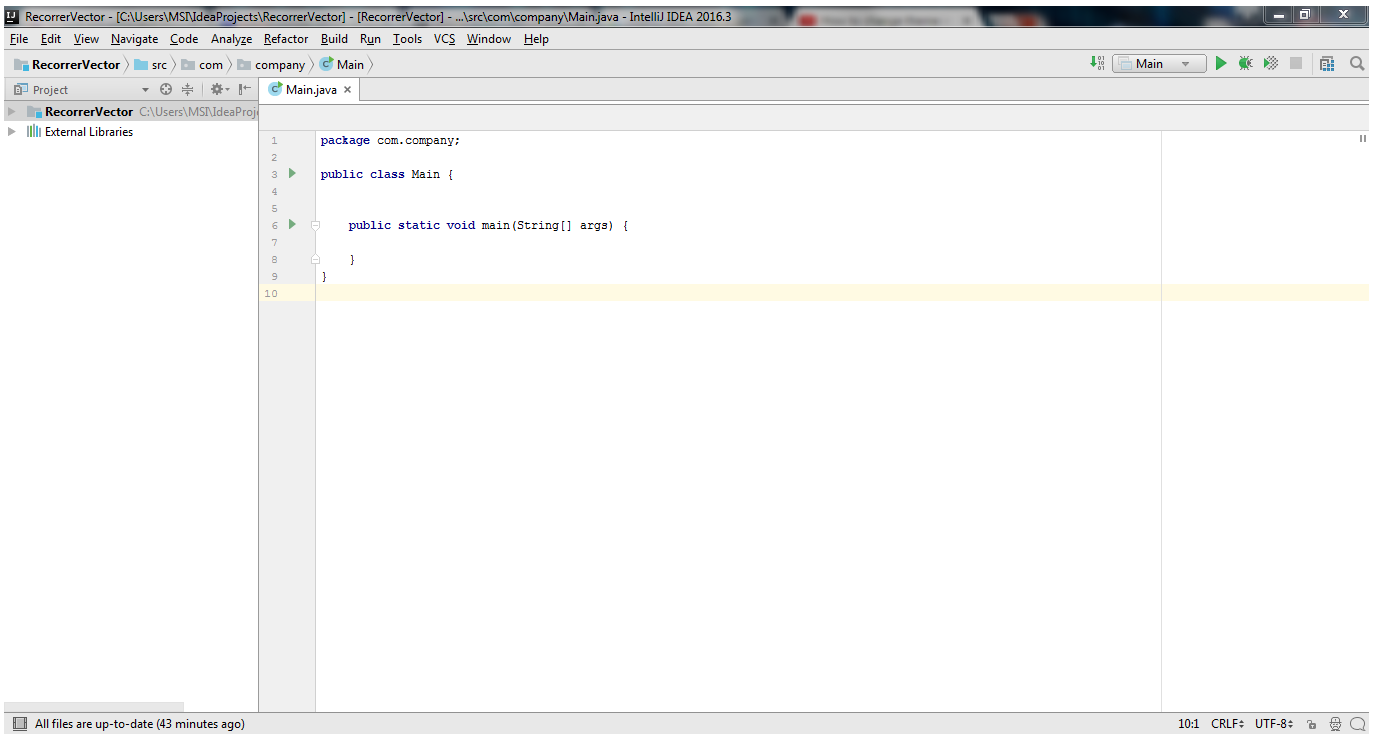


Antes de empezar nos da la bienvenida y muestra un pequeño tutorial.

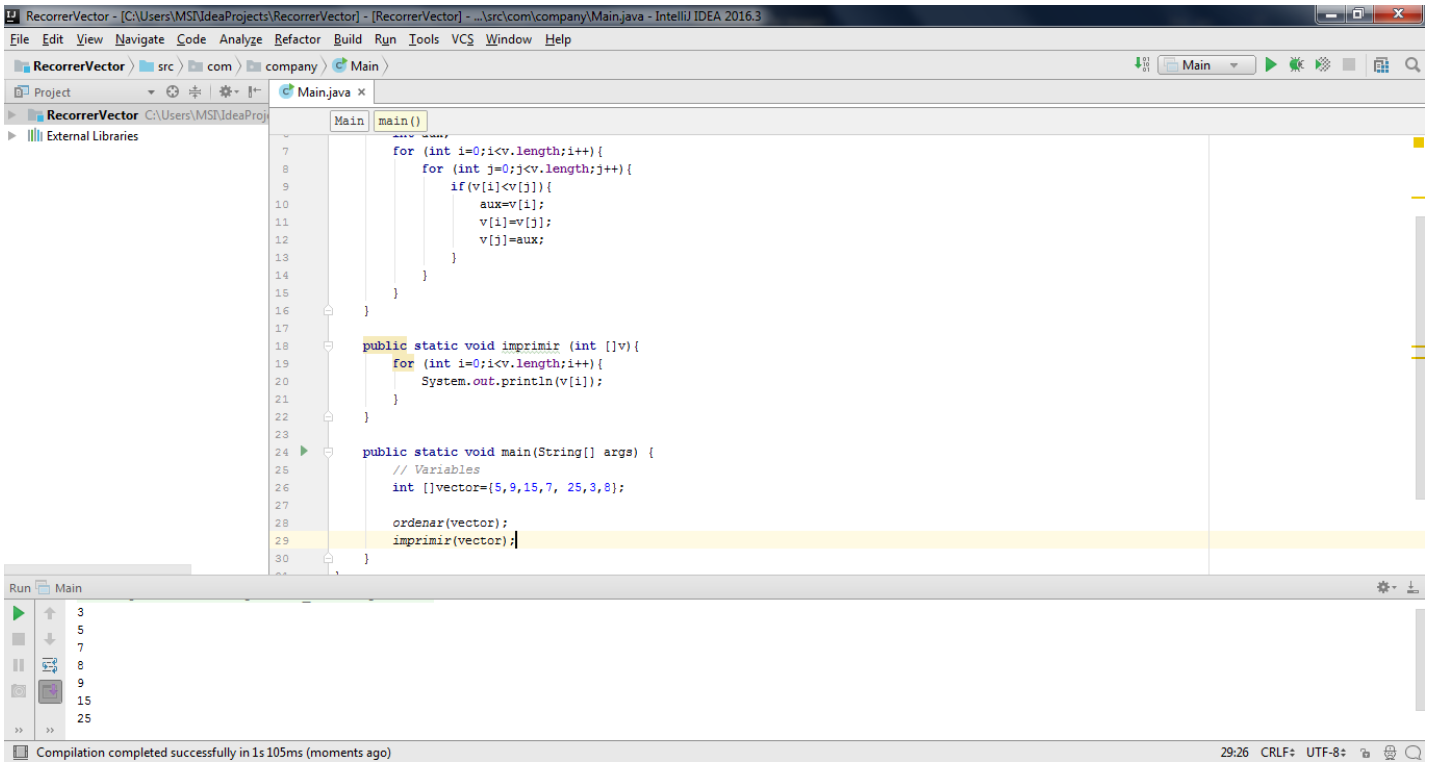


3.4.2 EDITOR DE TEXTO

Ya tenemos aquí nuestro editor de texto, que por defecto es una plantilla con la clase **main** ya construida.



A continuación, debemos escribir nuestro programa. El editor de texto nos ayuda según vamos escribiendo e ira marcando con un subrayado rojo los posibles errores de sintaxis y posibles errores léxicos.

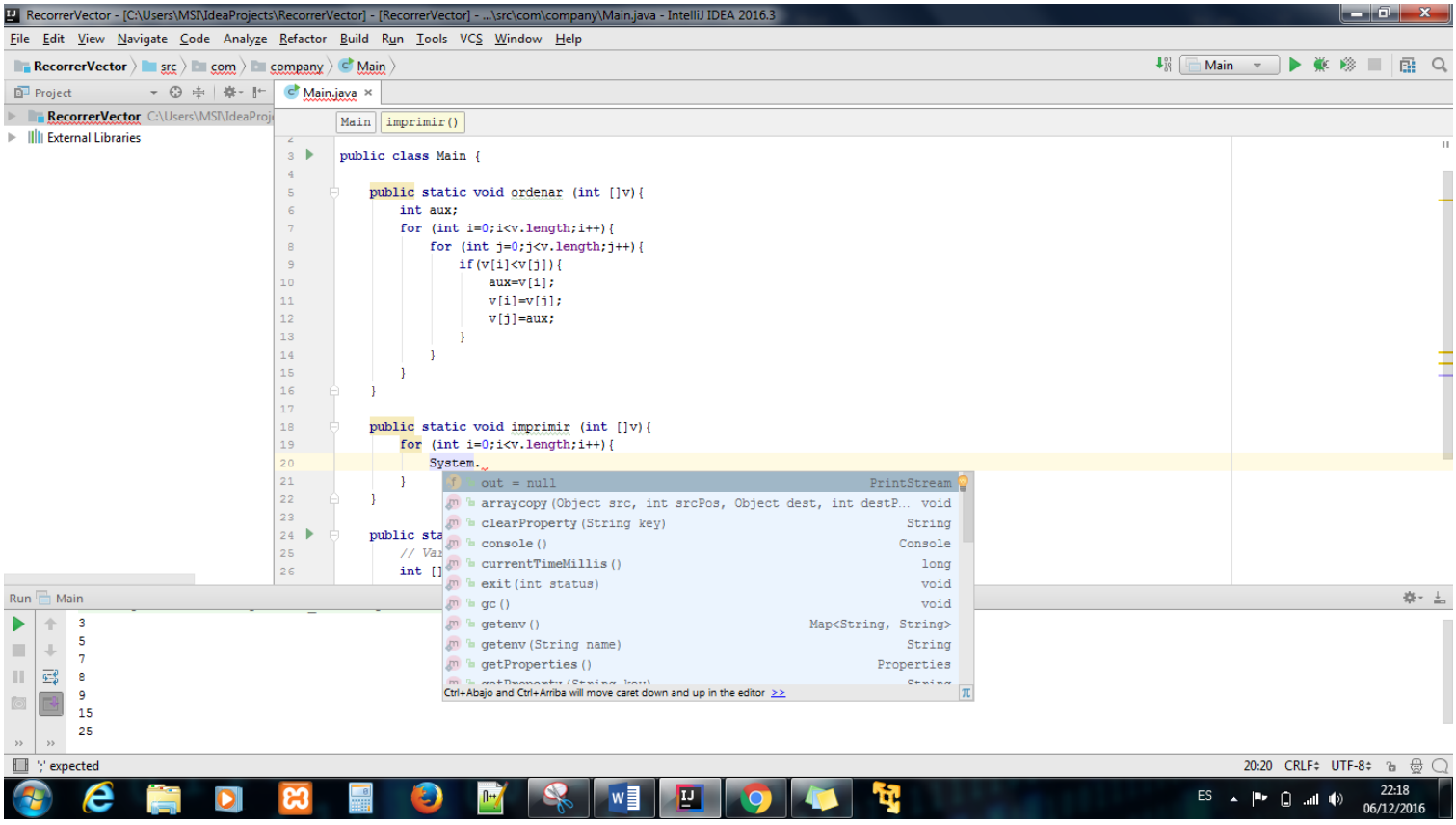


También podemos observar que nos muestra de forma resaltada las palabras reservadas del lenguaje.

```
5   public static void ordenar (int []v){
6       int aux;
7       for (int i=0;i<v.length;i++){
8           for (int j=0;j<v.length;j++){
9               if(v[i]<v[j]){
10                  aux=v[i];
11                  v[i]=v[j];
12                  v[j]=aux;
13              }
14          }
15      }
16  }
```

```
5   public static void ordenar (int []v){
6       int aux;
7       for (int i=0;i<v.length;i++){
8           for (int j=0;j<v.length;j++){
9               if(v[i]<v[j]){
10                  aux=v[i];
11                  v[i]=v[j];
12                  v[j]=au;
13              }
14          }
15      }
16  }
```

El error descrito anteriormente es difícil de cometer puesto que como podemos ver en la imagen inferior, el propio editor nos muestra las posibles opciones disponibles según vamos escribiendo.

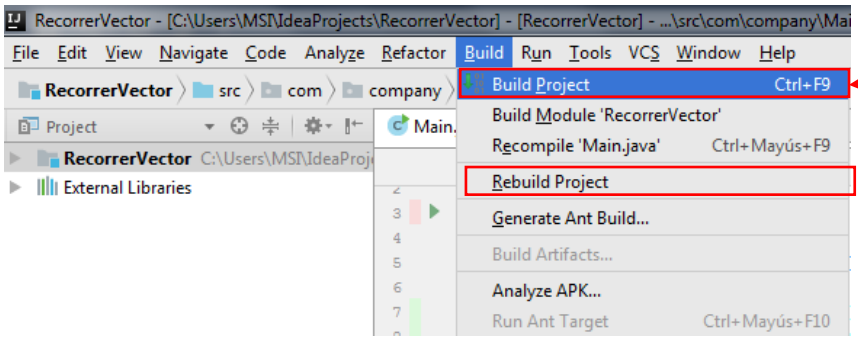


3.4.3 COMPILADOR

Para ejecutar el programa debemos compilar nuestro código, es decir, transformar nuestro código fuente en un código binario o código máquina y para ello tenemos varias formas de hacerlo a través de las siguientes opciones:

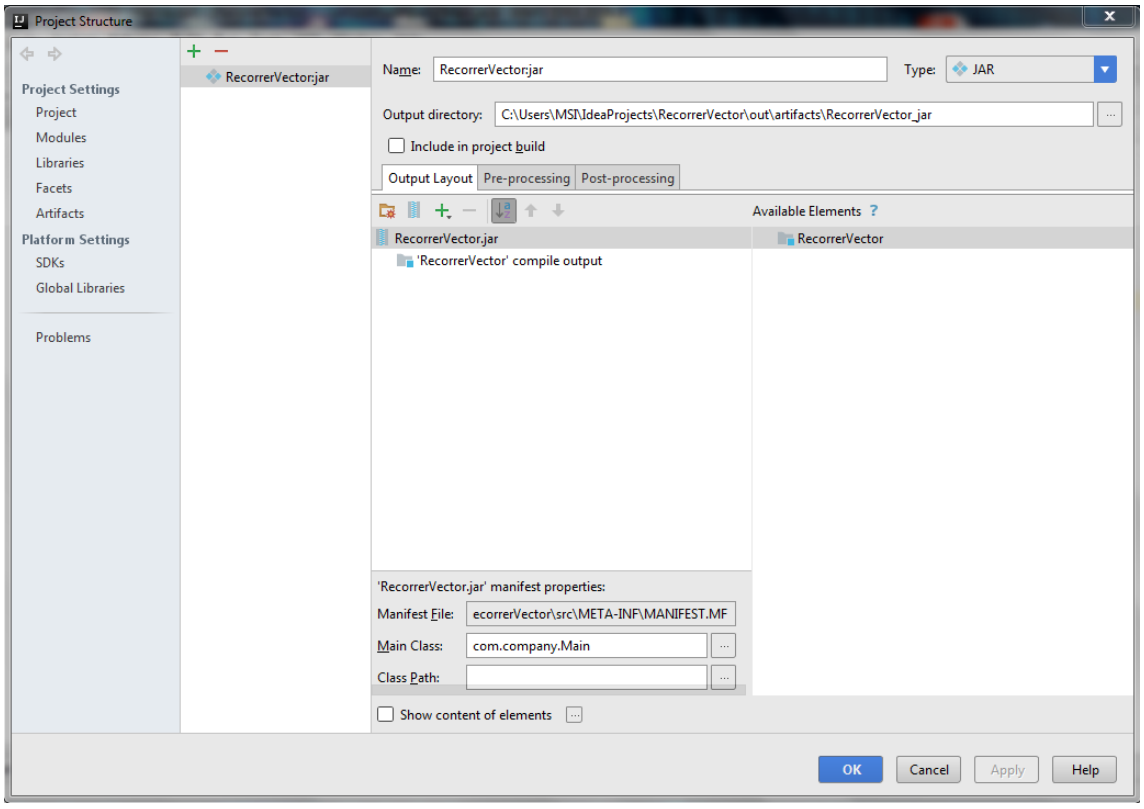
Dicho botón compila y ejecuta por la consola de nuestro entorno

Otra forma de compilar nuestro algoritmo es a través de las pestañas de la barra de menús.

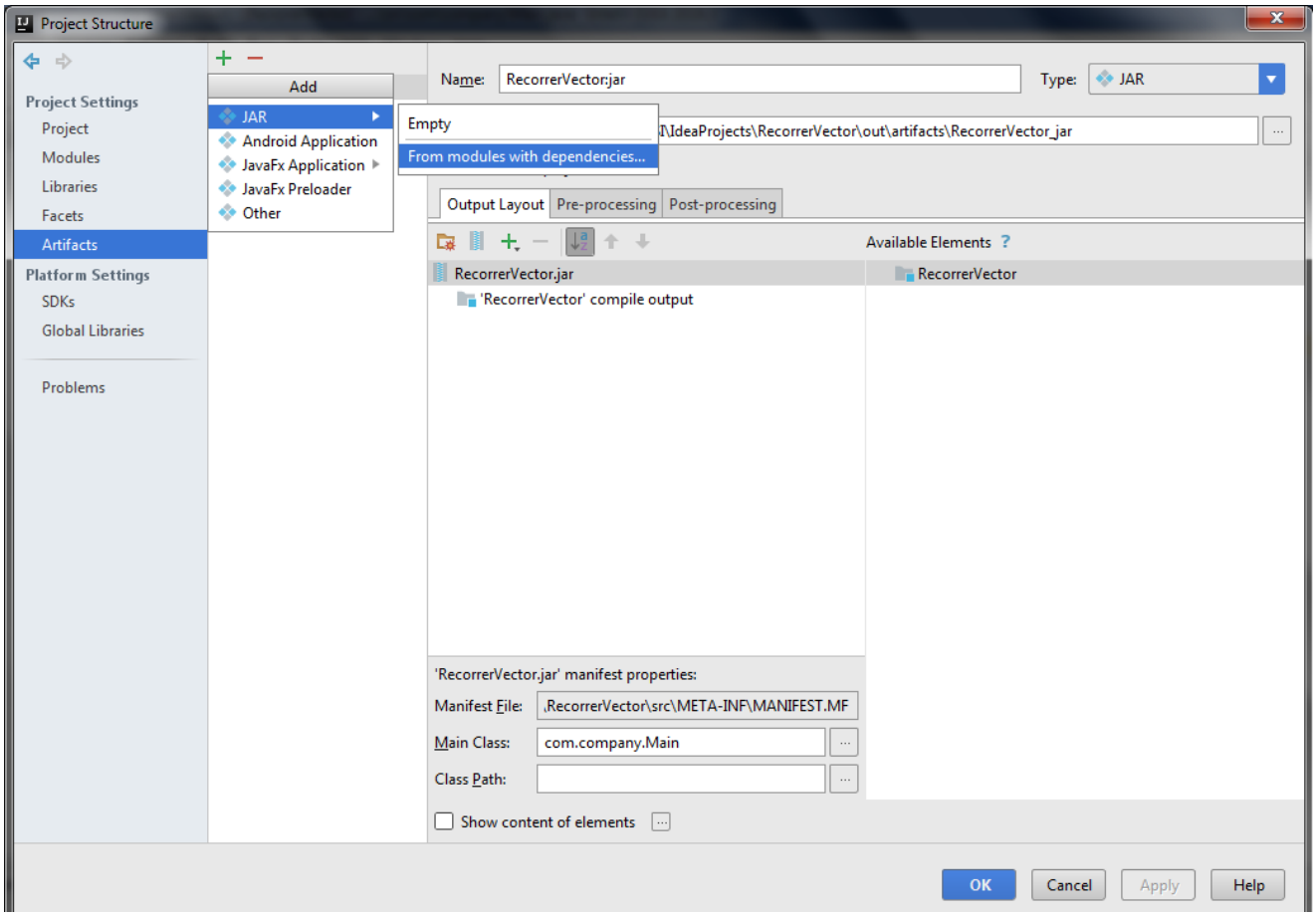


Tenemos que tener en cuenta que ninguna de estas compilaciones crea el ejecutable. Para obtener nuestro ejecutable debemos ir a **File → Project structure** y nos aparecerá la siguiente ventana.

En ella debemos ir a la opción **Artifac**.

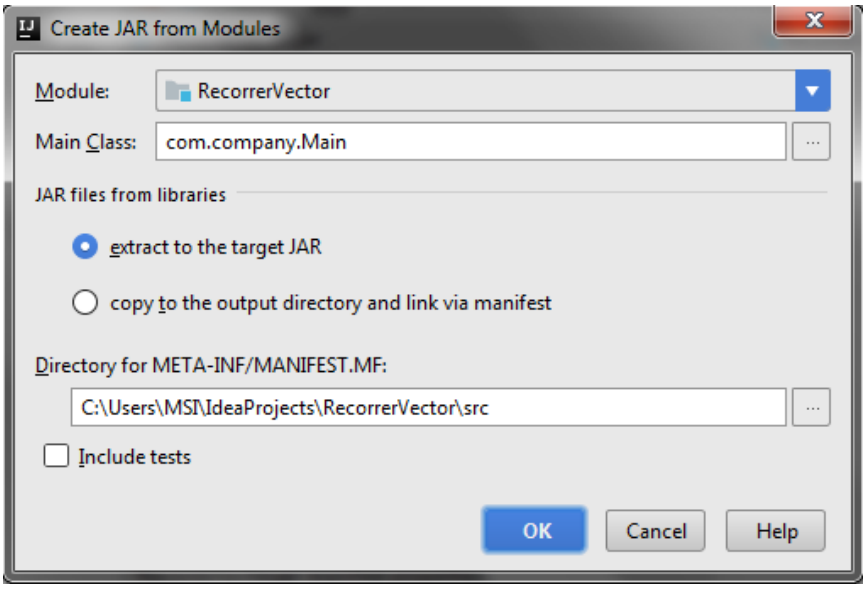


A continuación, vamos hacemos clic en **+ → JAR → from modules with dependecies**.



Posteriormente se nos abrirá la siguiente ventana en la que debemos seleccionar el proyecto, la clase principal y la ubicación de donde se alojado el modulo.

Si vamos a la siguiente ruta encontraremos el ejecutable (C:\Users\MSI\IdeaProjects\RecorrerVector\out\artifacts\RecorrerVector_jar)



3.4.4 DEPURADOR

El depurador o debugger nos permite ejecutar el código fuente paso a paso para identificar posibles errores de forma más sencilla y rápida facilitándonos su búsqueda y obtener una forma más eficiente de desarrollo.

Para ello, nos permite utilizar puntos de ruptura o breakpoint, para detener la ejecución del programa y poder observar el estado de las variables, su valor e incluso modificarlo sobre la marcha y continuar la ejecución.

Hay varias formas de iniciar la ejecución del programa en modo depuración:

Podemos hacerlo mediante la pestaña **Run → debug.**

Podemos hacerlo con un atajo de teclado (**Mayus + F9**) o pulsando sobre el botón:

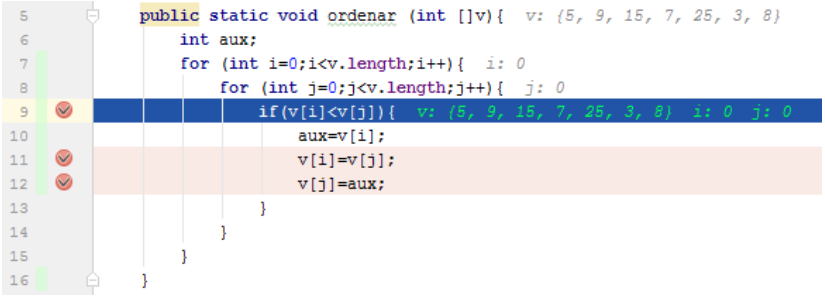
Cuando arranca, el programa se ejecuta hasta llegar al primer punto de ruptura. Si no han establecido dichos puntos, el programa se ejecutará normalmente hasta el final.

Otra forma de depurar el código es **Depurar → Ejecutar hasta el cursor** o con el atajo **F4**

El depurador empieza e ejecutar el programa hasta la instrucción donde se encuentra el cursor.

Comienza la depuración desde la primera línea del método main. El depurador se detiene esperando que decidamos el modo de depuración.

Una vez iniciada la depuración, el depurador se detiene sobre la siguiente línea de código que se va a ejecutar y esta aparece en azul.



Para el uso del depurador manejamos distintas opciones a través de botones:

	**Rerun (Ctrl + F5)** Volver a comenzar desde la primera línea
	**Mute Breakpoint (F7)** Ignora los puntos de ruptura
	**View Breakpoint (Ctrl+Mayus+F8)** Ver panel de puntos de ruptura
	**Continue (F5)** La ejecución del programa continúa hasta el siguiente punto de ruptura. Si no existe un punto de ruptura se ejecuta hasta el final.
	**Finish Debugger (Ctrl + F2).** Termina la depuración del programa.

3.4.4.1 ESTABLECER PUNTOS DE RUPTURA

Un punto de ruptura es una marca que indica al depurador que debe detenerse cuando la ejecución del programa llegue a ella.

Cuando el programa se detiene podemos:

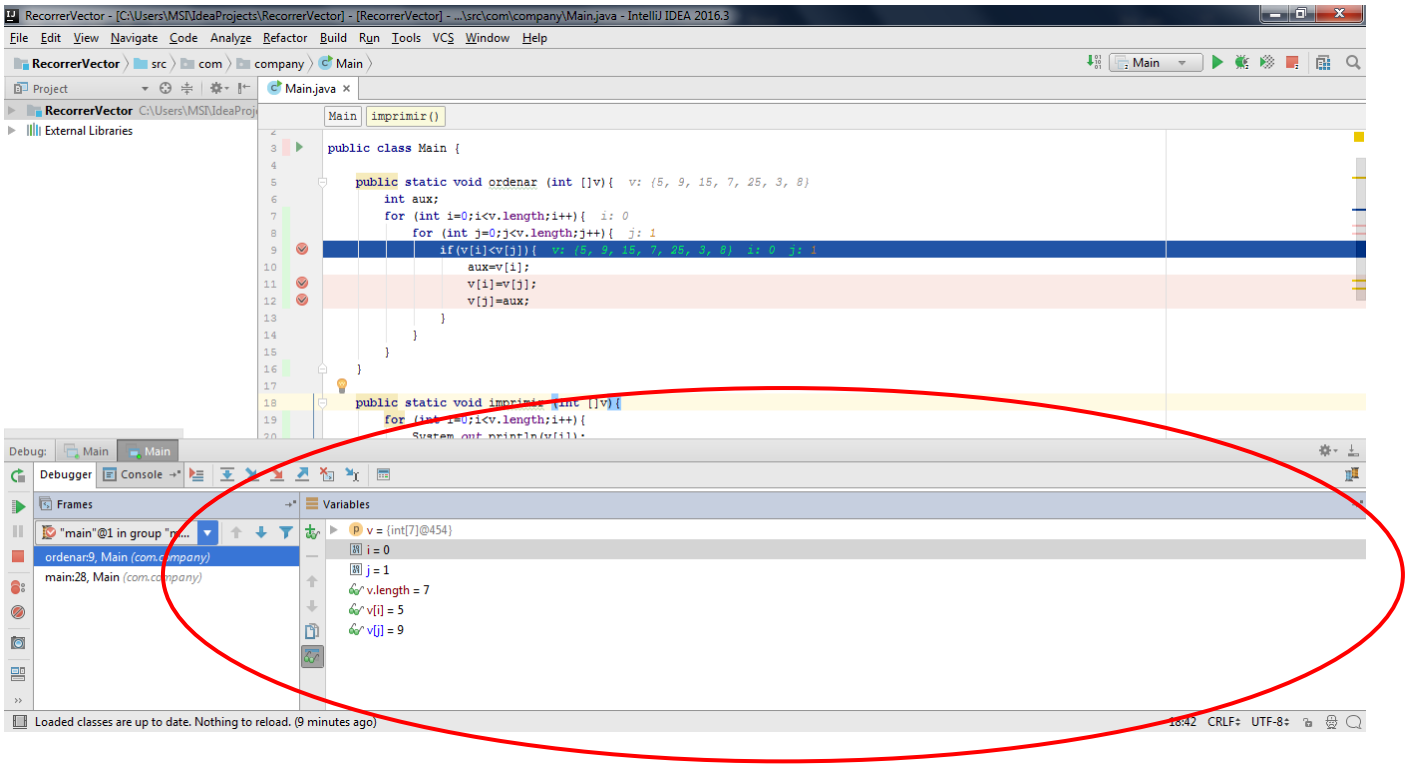
- Examinar los valores actuales de las variables.
- Continuar la depuración línea a línea del programa.

Para fijar un punto de ruptura simplemente pulsamos sobre el número de línea donde se desea colocar. La línea queda resaltada en color rojo con una marca del mismo color en el margen izquierdo.

Para eliminar un punto de ruptura se pulsa sobre el cuadrado rojo.

3.4.4.2 LA VENTANA DE VARIABLES LOCALES

En esta ventana se muestran las variables, su tipo y su valor actual.

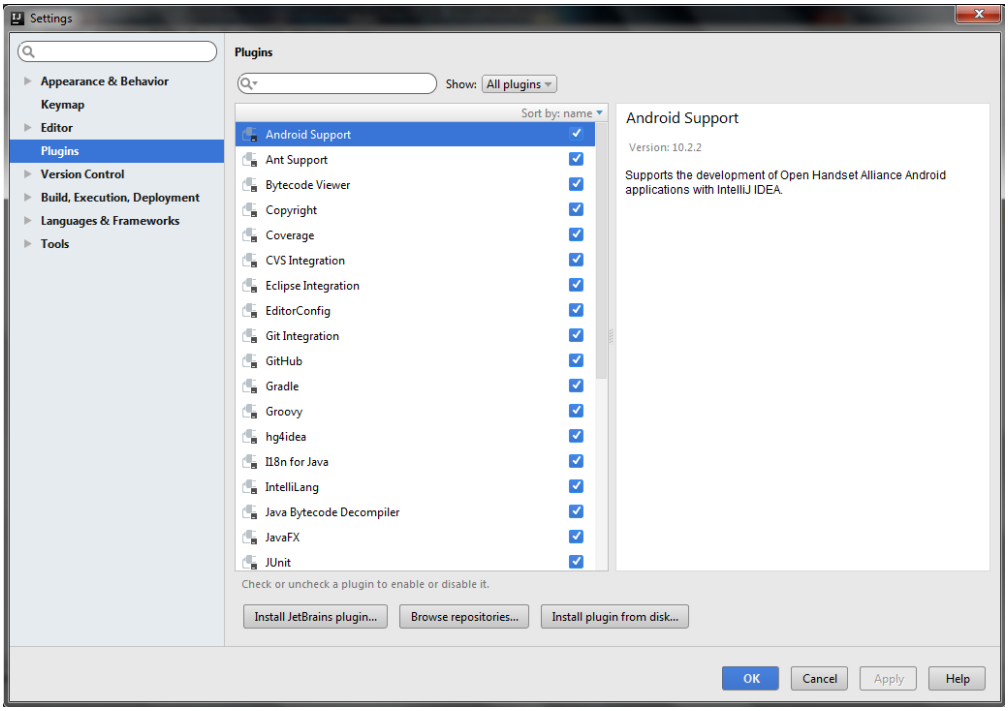


El depurador permite cambiar el valor de una variable local en esta ventana y continuar la ejecución del programa usando el nuevo valor de la variable.

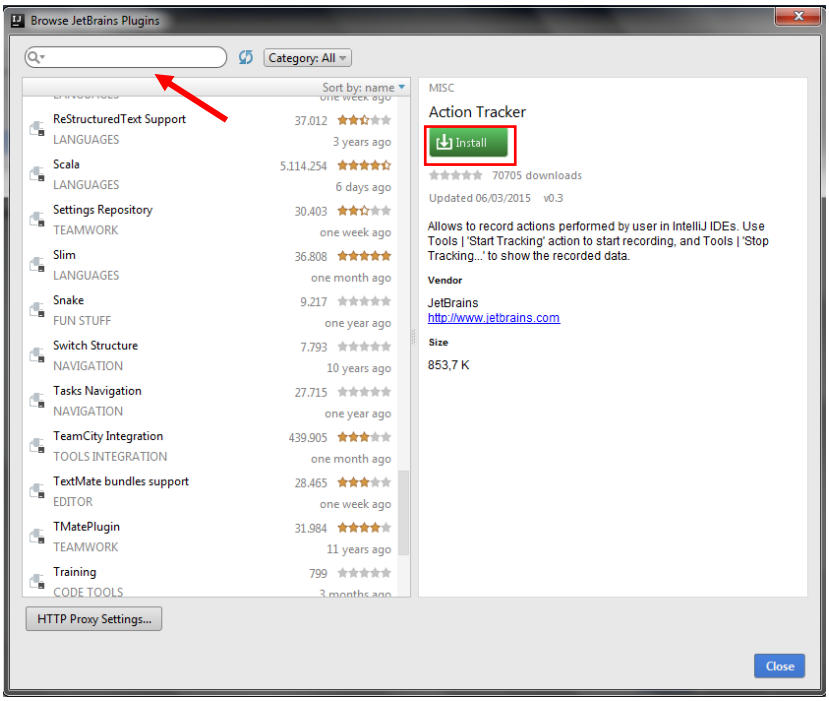
3.4.5 PLUGINS

IntelliJ IDEA nos ofrece una serie de plugins con las que podremos trabajar y ampliar nuestra herramienta de trabajo.

Para descargar e instalar cualquier plugin en el entorno de IntelliJ IDEA tenemos que ir a **File → Settings→ Plugins** o con el atajo de teclado **Ctrl+Alt+S** y nos aparecerá la siguiente ventana:



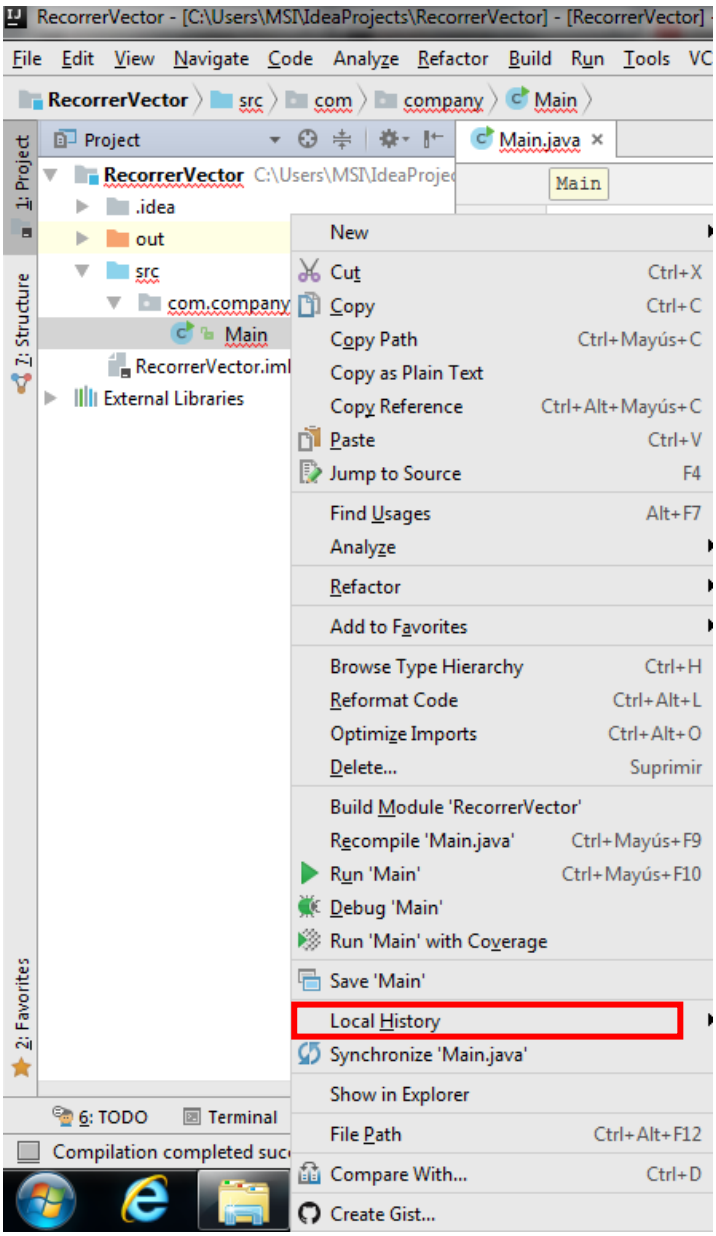
Para buscar los plugins hacemos clic en **Install JetBrains plugins** y nos aparecerá una nueva ventana, donde podemos realizar búsquedas por nombre o por categoría, una vez tengamos localizado nuestro plugins haremos clic en **install** y solo debemos reiniciar nuestro IDE.



3.4.6 CONTROL DE VERSIONES

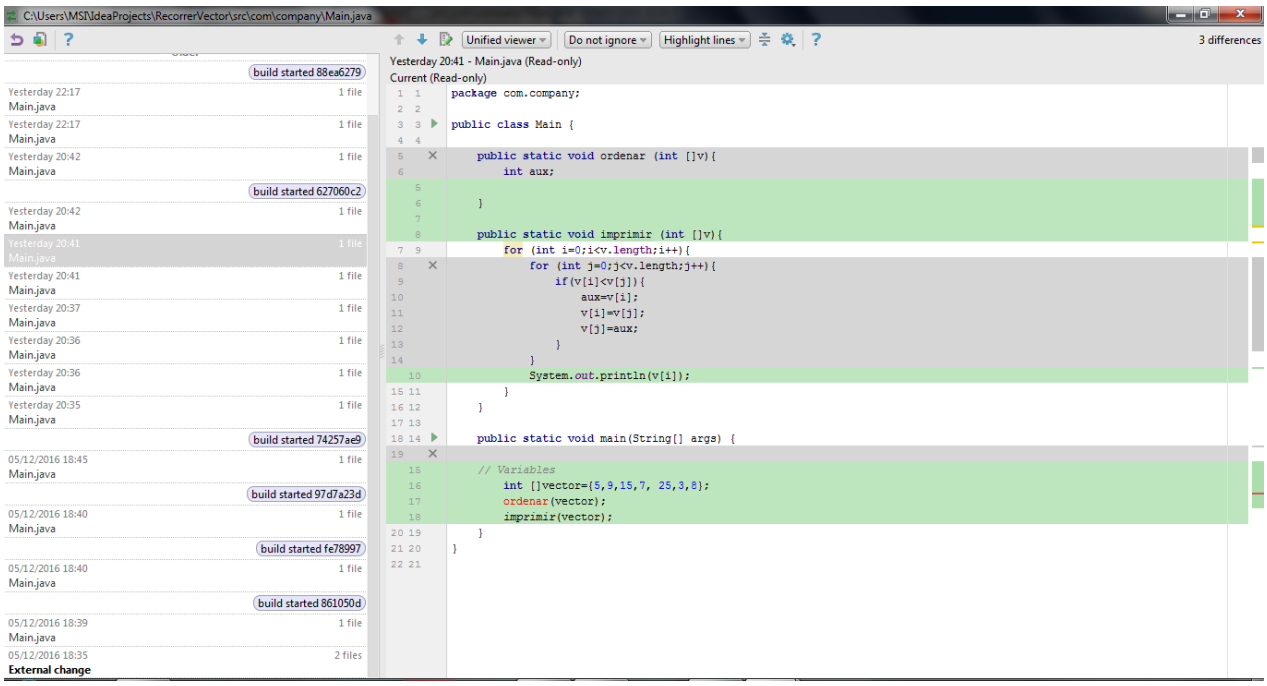
IntelloJ IDEA dispone de una herramienta muy útil para los desarrolladores de software, es el control de versiones. Esta herramienta nos muestra los cambios sufridos en nuestro código fuente a lo largo de su desarrollo.

Para acceder a dicha herramienta hacemos clic derecho sobre el método main la ventana de estructura de proyecto y elegimos la opción **Local History**.



Se nos abrirá la siguiente ventana en la que nos aparecerán las distintas versiones que han sido guardadas a lo largo del desarrollo, ahora solo tenemos que seleccionar la versión que queremos mostrar.

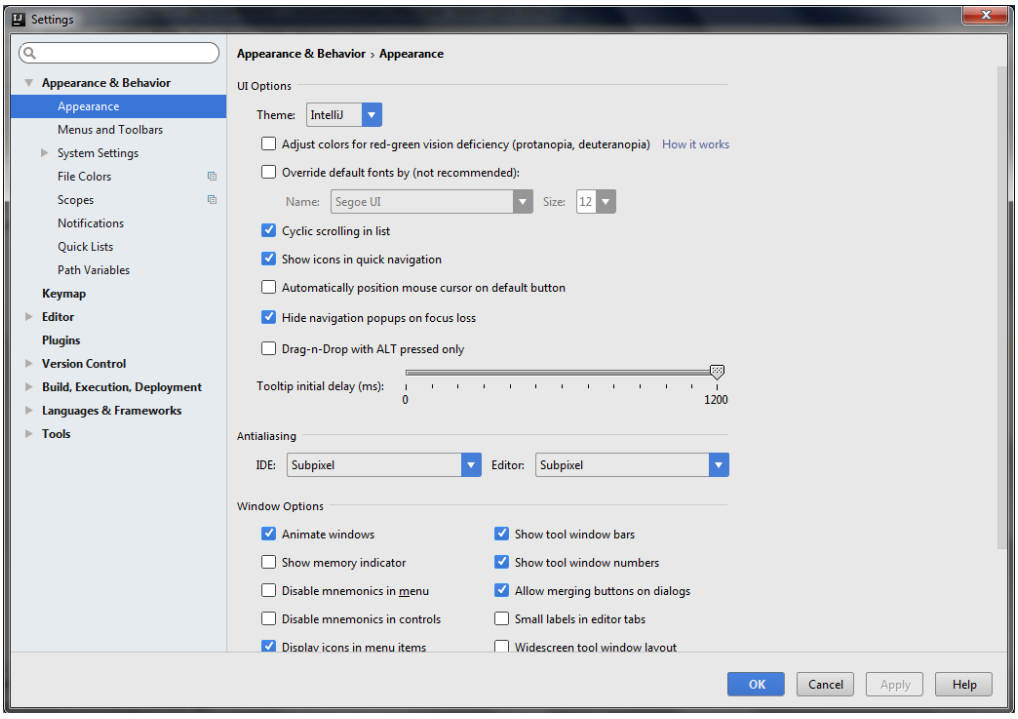
En la parte izquierda podemos ver un despliegue de todas las versiones guardas y en la parte derecha vemos señalados en color verde los cambios que ha sufrido nuestro algoritmo.



Como podemos observar en la imagen de arriba, al seleccionar una de las versiones nos destaca visualmente cuales han sido los cambios entre la versión seleccionada y la última disponible.

3.4.7 PANEL DE CONFIGURACION

Para acceder a la configuración de InteliJ IDEA tenemos que ir **a File → Settings→ Plugins** o con el atajo de teclado **Ctrl+Alt+S** y nos aparecerá la siguiente ventana

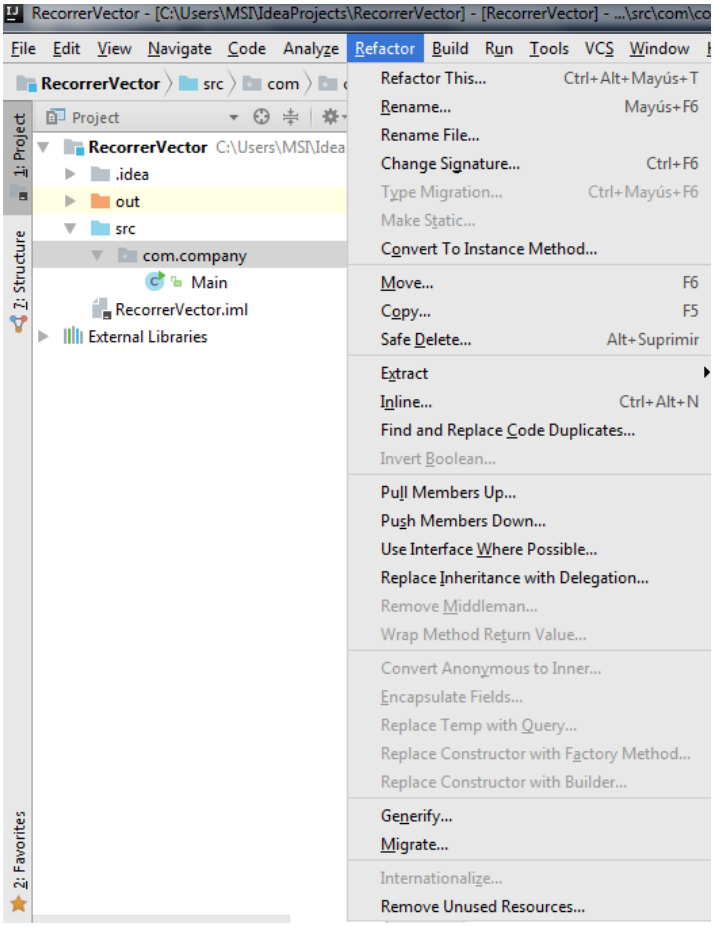


El panel de opciones está dividido en dos partes: un panel de navegación a la izquierda y un área de presentación a la derecha. El panel de navegación está formado por un árbol de directorios que dispone de opciones como la apariencia, Plugins, Control de Versiones, Lenguajes... Para acceder a las opciones de cada carpeta debemos expandir la carpeta deseada para ver las páginas de opciones que contiene. Cuando selecciona una de las carpetas de una página determinada, sus opciones aparecen en el área de presentación.

Al hacer clic en Aplicar en el cuadro de diálogo se guardan todas las configuraciones de todas las páginas y tan solo nos queda hacer clic en OK para salir.

3.4.8 REFACTORIZACION

La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. El objetivo, por el contrario, es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño, y eliminar código muerto para facilitar el mantenimiento en el futuro. Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede refactorizarlo primero para facilitar esta tarea y luego añadir el nuevo comportamiento.



En esta pestaña disponemos de opciones como cambiar el nombre del paquete principal, mover, copiar y eliminar código, seleccionando una parte de código lo podemos reestructurar creando métodos, etc.

4. CONCLUSION

Tras analizar los distintos entornos de desarrollo (IDE) como han sido NetBeans 8.2, Eclipse Neon1, Visual Studio 2015 y IntelliJ IDEA y haber probado algunas de sus herramientas, asignaré una puntuación a cada uno de ellos, bajo mi criterio de evaluación según sus funcionalidades:

NetBeans 8.2	9
Eclipse Neon1	7
Visual Studio 2015	6
IntelliJ IDEA	8

A continuación, explicaré brevemente el motivo de dichas puntuaciones con los pros y contras más llamativos que he deducido de cada una de estas herramientas.

He de decir que la puntuación más alta ha sido para NetBeans debido a ser el entorno que más conozco, ya que es el que más utilizo a diario, pero aparte de eso, me parece la herramienta más completa junto con IntelliJ IDEA, ya que ambos son de los más completos. Del mismo modo, he de decir que es multiplataforma y además está disponible en varios idiomas, es intuitivo y fácil de configurar.

Sin embargo, IntelliJ IDEA, me parece un entorno en el que la distribución de ventanas es limpia y clara, es un entorno agradable mientras desarrollamos nuestros proyectos, y esto de extrema importancia puesto que todo programador está constantemente viendo la misma ventana durante mucho, ya que es donde escribirá su código fuente. Otra de las cosas que me ha llamado la atención es la herramienta de depuración. Sin duda, para mi parecer es el mejor de los cuatro entornos probados puesto que es muy intuitivo y tiene detalles como mostrar los valores de las variables en la misma línea de código que se está ejecutando, es un detalle que hace más fácil el descubrimiento de posibles errores. El principal inconveniente que le encuentro es que solo está disponible en inglés, aunque eso no debe ser un problema.

Eclipse es el entorno más usado a nivel empresarial, pero su principal problema es que debemos de instalar extensiones dependiendo del lenguaje o las funcionalidades que vayamos a utilizar, o el tipo de aplicación que vayamos a diseñar. Por defecto, se encuentra en inglés, aunque es posible cambiar el idioma descargando la extensión del lenguaje correspondiente según la versión de la que dispongamos. La distribución de las ventanas no facilita el trabajo agradable, ya que abre demasiadas y se descolocan fácilmente.

Visual estudio es una herramienta muy potente, aunque muy poco o nada intuitiva. Es necesaria la instalación de extensiones para casi cualquier función que deseemos hacer, y creo que es una aplicación para trabajar constantemente en línea, ya que parece que es lo que quiere Microsoft, como hace con todas sus herramientas. La instalación de esta herramienta es muy lenta y muy pesada (8 GB) ya que debe descargar millones de librerías (o eso parece).

Si tuviese que elegir uno definitivamente para el uso diario, mi elección sería claramente IntelliJ IDEA, sin duda me parece el mejor de los entornos en cuanto a diseño y es bastante completo junto a funcionalidades. Nos obstante, debería estudiar más a fondo dicho entorno y para así poder conocer todas las herramientas que incorpora, aunque para eso he de tener más conceptos en programación y así poder entender cada una de las opciones de las que dispone.

5. CLASIFICACION DE LOS ENTORNOS

IDE	Plataforma (Windows, Mac, Linux…)	Lenguajes que soporta	Libres o propietarios	Tipo de aplicación	Fabricante o empresa
Eclipse	Windows, Linux, Mac OS	Java, ANCI C, C++, JSP, sh, perl, php, sed	Libre	Empresarial, de uso general y en red	Fabricante
NetBeans	Windows, Linux, Mac OS	Java, php, C, C++, JS, HTML, JavaScript, Groovy	Libre	Empresarial, de uso general y en red	Empresa
IntelliJ IDEA	Windows, Linux, Mac OS	Java, perl, XML/XSL, Python, Lua, haskell, SQL, ruby y Groovy	Tiene una versión libre y otra de propietaria	Empresarial, de uso general y en red	Fabricante
JBuilder	Windows, Linux, Mac OS	java	Tiene una versión libre y otra de propietaria	Empresarial, de uso general y en red	Fabricante
Bluefish	Windows, Linux, Mac OS	HTML, PHP, Javascript, JSP, SQL, XML, Python, Perl, CSS, ColdFusion, Pascal, R, Octave/Matlab	Libre	web	Fabricante
Visual C++	Windows	C, C++ y C++/CLI.	Tiene una versión libre y otra de propietaria	Empresarial, de uso general y en red	Empresa
MS Visual Studio	Windows	C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP	Tiene una versión libre y otra de propietaria	Empresarial, de uso general y en red	Empresa
Clarion	Windows	SQL, ADO, y XML	Tiene una versión libre y otra de propietaria	BBDD	Empresa
Anjuta	Linux	C, C++, Java, Python y Vala	Libre	Empresarial, de uso general y en red	Fabricante
KDevelop	Linux	C, C++, PHP y Python	Libre	Empresarial, de uso general y en red	Fabricante
JDeveloper	Windows, Linux, Mac OS	Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML	Libre	Empresarial, de uso general y en red	Empresa
Dreamweaver	Windows, Mac OS	ActionScript, ASP, ASP.NET, C#, CSS, ColdFusion, XHTML, XML, XSLT	propietaria	Web	Empresa