



# Conceptos de Java

# Índice



## Conceptos de Java

1   Evolución de Java	3
1.1   Ediciones de Java	5
1.2   Instalación de Java	7
1.4   Funcionamiento de Java	12
1.5   Compilar y ejecutar programas en Java	14
1.6   Entornos de desarrollo	18

# 1. Evolución de Java

El proyecto precursor al nacimiento de Java, se desarrollo a principio de los años 90, por los ingenieros de Sun Microsystems Patrick Naughton, James Gosling y Mike Sheridadm, dicho proyecto fue llamado "El proyecto verde", que trataba de desarrollar una tecnología nueva para programar nuevos dispositivos inteligentes. En dicho proyecto trabajaron 13 ingenieros de Sun Microsystems.

En un principio pensaron en la utilización de C++, pero la descartaron porque querían un lenguaje de programación más fácil de usar y aprender. Se decidieron por crear uno que no estuviese ligado a ningún tipo de CPU en concreto, y adoptando características y funcionalidades de C, C++ y Objective C, crearon Oak, al que poco tiempo después llamaron Java, quizás por las iniciales de los ingenieros James Gosling, Arthur Van Off, y Andy Bechtolsheim o en honor al sabroso café de la isla de Java.

El principio en el que se basaron para crea Java, fue que el código resultante pudiese ser usado bajo cualquier arquitectura "*Write Once, Run Anywhere*" (escríbelo una vez, ejecútalo en cualquier sitio).

Hasta la fecha de su nacimiento oficial, su evolución brevemente es la siguiente:

- En 1992 se presenta el proyecto verde, con los prototipos a bajo nivel.
- Entre 1993 y 1994 se trabaja para poder presentar un prototipo funcional "HotJava", donde se empieza a vislumbrar todo el

potencial que se puede ofrecer, haciendo su presentación en septiembre de 1994.

- El 23 de mayo de 1995, en la conferencia SunWorld, John Gage, de Sun Microsystems, y Marc Andreessen, cofundador y vicepresidente de Netscape, anunciaron la versión alpha de Java, este fue el día que se toma como nacimiento de Java, a partir de ese momento la versión se incorporó en Netscape Navigator, que en ese año era el navegador más utilizado en la web.

El llamado JDK, conjunto de herramientas software de desarrollo para la creación de programas en Java, se presento oficialmente en su primera versión JDK 1.0 el 23 de enero de 1996. En febrero de 1997 se lanzó la versión JDK 1.1

Pero en diciembre de 1998 la siguiente versión paso a llamarse J2SE 1.2 (Java Standard Edition), que reemplazó a JDK para distinguirlo de las ediciones J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition).

A esta versión le siguieron las JS2E 1.3 en mayo de 2000 y la J2SE 1.4 en febrero de 2002. Para la siguiente que surgió en septiembre de 2004, cambiando la forma de numerar las versiones llamándose J2SE 5.0.

Desde esa fecha hasta hoy mismo se han sucedido las versiones JSE 6, JSE 7 y JSE 8, en las que se omite el número 2 en el nombre y se pasan a denominar las diferentes ediciones JSE, JEE y JME.

Hay que destacar que en Abril de 2009 Oracle adquirió Sun Microsystems.

Se espera que la nueva versión JSE 9 esté disponible para su utilización en septiembre de 2016.

En los enlaces siguientes se puede encontrar dos infografías sobre la evolución de Java:

<http://www.techsagar.com/the-evolution-of-java-programming-language/>

<https://www.exoplatform.com/blog/en/2015/03/26/infographic-history-java-programming-language-happy-20th-birthday>

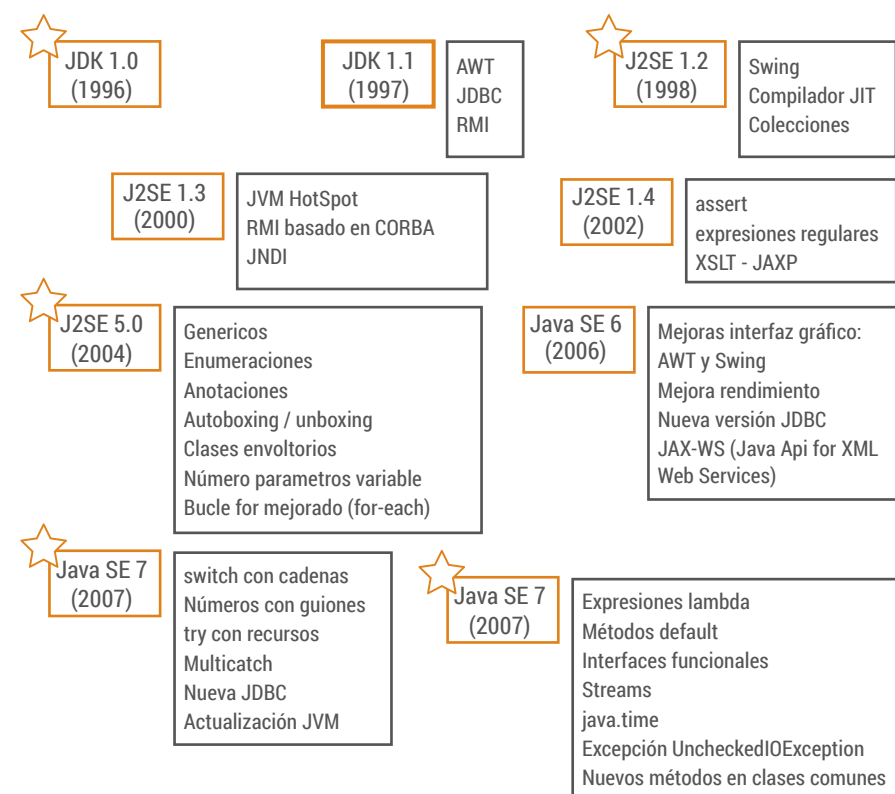


FIGURA 1.1: VERSIONES DE JAVA

## 1.1 | Ediciones de Java

Como se comentó en apartado anterior desde diciembre de 1998 existen tres ediciones de Java:

- **JSE 8:** Forman parte de este grupo los paquetes de clases de uso general, es decir, aquellos que se utilizan en cualquier tipo de aplicación (tratamiento de cadenas, colecciones, acceso a datos, etc...), además de los paquetes para la creación de entornos gráficos y aplicaciones para navegadores Internet (applets).
- **JEE 7:** En esta edición encontramos las clases e interfaces que posibilitan la creación de aplicaciones Empresariales de tres capas, entre otros, las aplicaciones para la Web.
- **JME 7:** Orientado a pequeños dispositivos móviles (teléfonos, tabletas, etc.).

Centrándonos un poco más en la edición JSE, objeto de este manual, Oracle tiene dos productos que la implementan:

- Java SE Development Kit (JDK) 8.
- Java SE Runtime Environment (JRE) 8.

JDK 8 es un superconjunto de JRE 8, y contiene todo lo que está en JRE 8, además de herramientas como los compiladores y depuradores necesarios para el desarrollo de applets y aplicaciones.

JRE 8 proporciona las bibliotecas, la máquina virtual de Java (JVM) y otros componentes para ejecutar applets y aplicaciones escritas en el lenguaje de programación Java.



El siguiente diagrama conceptual ilustra los componentes de los productos de Oracle Java SE:

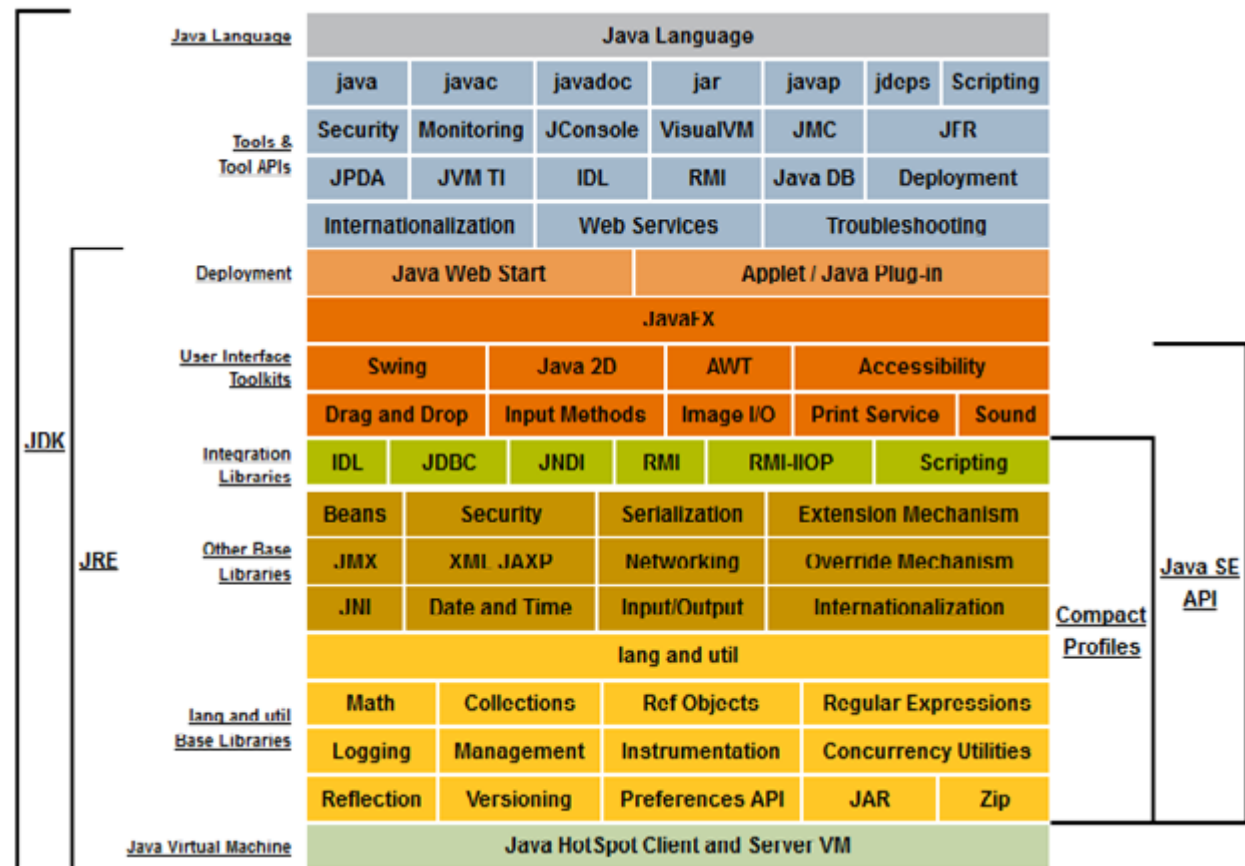


FIGURA 1.2: [HTTP://DOCS.ORACLE.COM/JAVASE/8/DOCS/INDEX.html](http://docs.oracle.com/javase/8/docs/index.html)

## 1.2 | Instalación de Java

Lo primero que hay que hacer es descargar el Java Development Kit (JDK) desde el sitio de Oracle, en este caso y para la edición actual JDK 8 es:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

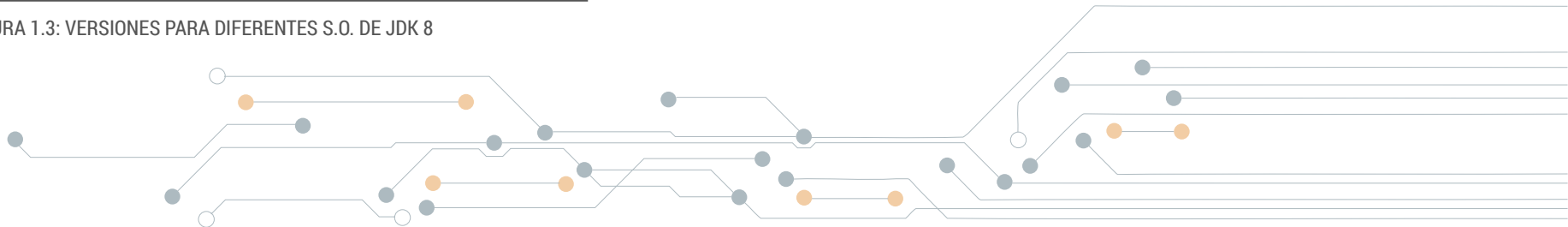
En dicha página, hay que seleccionar la versión del sistema operativo en el que se va a efectuar la selección, de acuerdo con lo que se muestra en la imagen siguiente:

Una vez descargado el archivo correspondiente, se procederá a la descompresión e instalación de acuerdo a las instrucciones particulares para cada S.O., indicadas en la página del enlace siguiente:

[https://docs.oracle.com/javase/8/docs/technotes/guides/install/install\\_overview.html#A1097144](https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html#A1097144)

Java SE Development Kit 8u60		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM v6v7 Hard Float ABI	77.69 MB	<a href="#">jdk-8u60-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM v8 Hard Float ABI	74.64 MB	<a href="#">jdk-8u60-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	154.66 MB	<a href="#">jdk-8u60-linux-i586.rpm</a>
Linux x86	174.83 MB	<a href="#">jdk-8u60-linux-i586.tar.gz</a>
Linux x64	152.67 MB	<a href="#">jdk-8u60-linux-x64.rpm</a>
Linux x64	172.84 MB	<a href="#">jdk-8u60-linux-x64.tar.gz</a>
Mac OS X x64	227.07 MB	<a href="#">jdk-8u60-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	139.67 MB	<a href="#">jdk-8u60-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.02 MB	<a href="#">jdk-8u60-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	140.18 MB	<a href="#">jdk-8u60-solaris-x64.tar.Z</a>
Solaris x64	96.71 MB	<a href="#">jdk-8u60-solaris-x64.tar.gz</a>
Windows x86	180.82 MB	<a href="#">jdk-8u60-windows-i586.exe</a>
Windows x64	186.16 MB	<a href="#">jdk-8u60-windows-x64.exe</a>

FIGURA 1.3: VERSIONES PARA DIFERENTES S.O. DE JDK 8



Por ejemplo para Windows, el archivo descargado es un instalador ejecutable, que ira guiando sobre los pasos que hay que ir siguiendo, en la mayoría de las ocasiones con aceptar las opciones que ofrece es suficiente.

En las imágenes siguientes se tratan de recoger los más significativos.

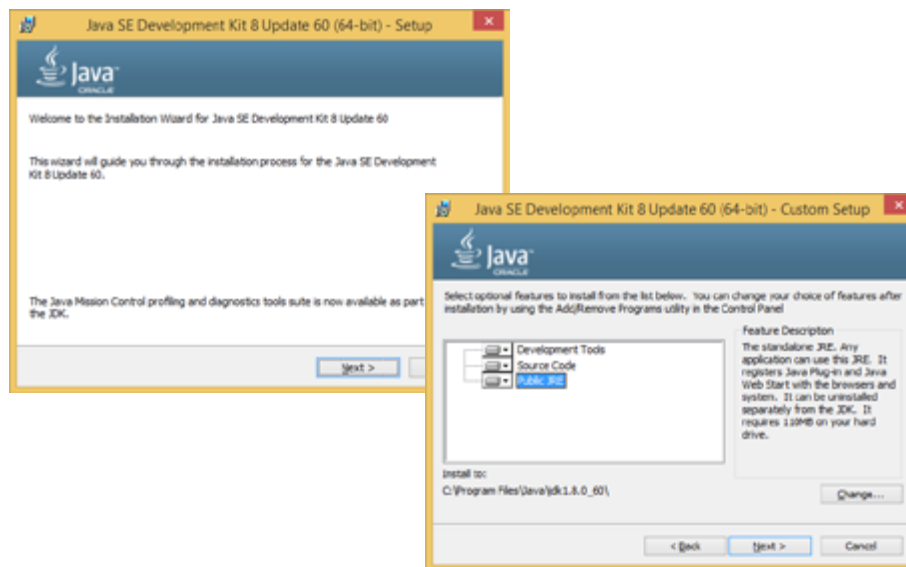


FIGURA 1.4: INICIO DEL INSTALADOR DE JDK Y SIGUIENTE PASO INDICANDO QUE INSTALAR Y DÓNDE.

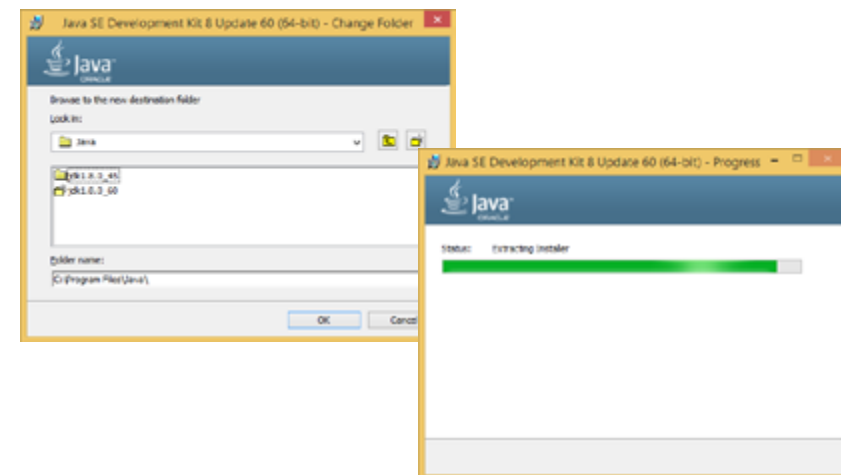


FIGURA 1.5: ELECCIÓN DE CARPETA DÓNDE INSTALAR JDK Y COMIENZO DE LA EXTRACCIÓN DE ARCHIVOS.

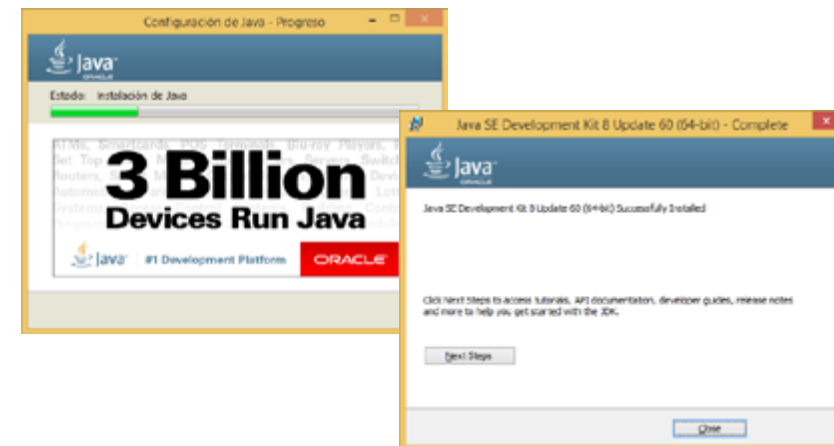


FIGURA 1.6: FINALIZACIÓN DE LA INSTALACIÓN.



Una vez finalizada la instalación es conveniente comprobar dónde ha quedado instalado el JDK y como está configurado en el sistema operativo Java.

Para ello en Windows, se ejecuta la opción "Java", del "Panel de control". Pulsando el botón "Acerca de...", de la pestaña "General", se muestra la versión instalada de JDK, tal y como muestra la imagen siguiente:

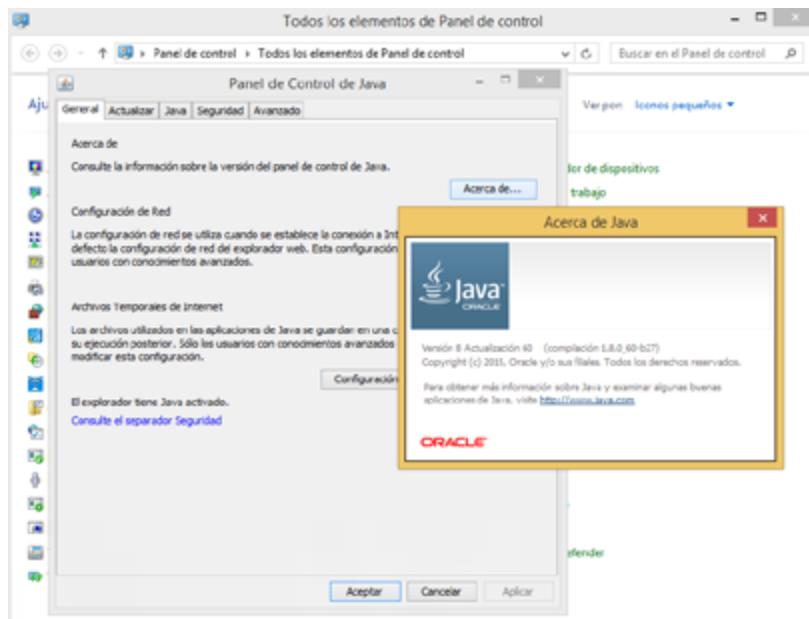
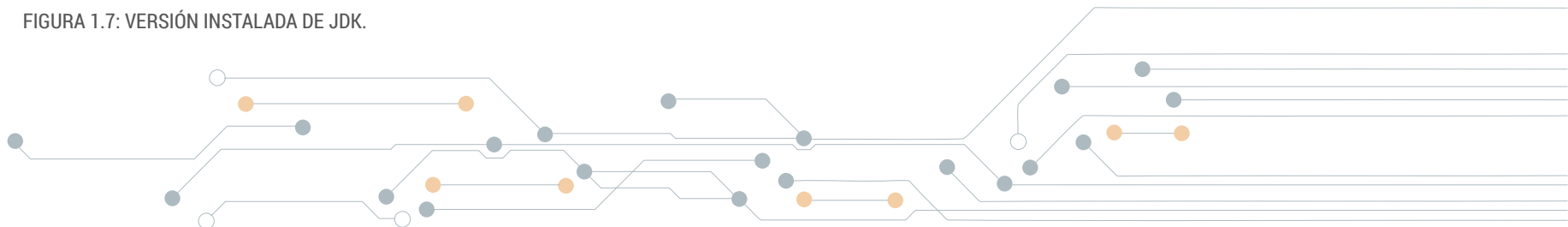


FIGURA 1.7: VERSIÓN INSTALADA DE JDK.

Para poder trabajar con las herramientas de compilación y ejecución de programas en Java, es fundamental conocer el valor de dos variables de entorno (de sistema o de usuario de sesión), como son:

- PATH, con la información de las carpetas en las que el sistema operativo buscara un archivo si no lo encuentra en la que se esté posicionado.
- JAVA\_HOME, con la información de la carpeta en la que se ha instalado el JDK.



En la imagen siguiente se muestra cómo se pueden obtener y cambiar los valores de estas variables de entorno.

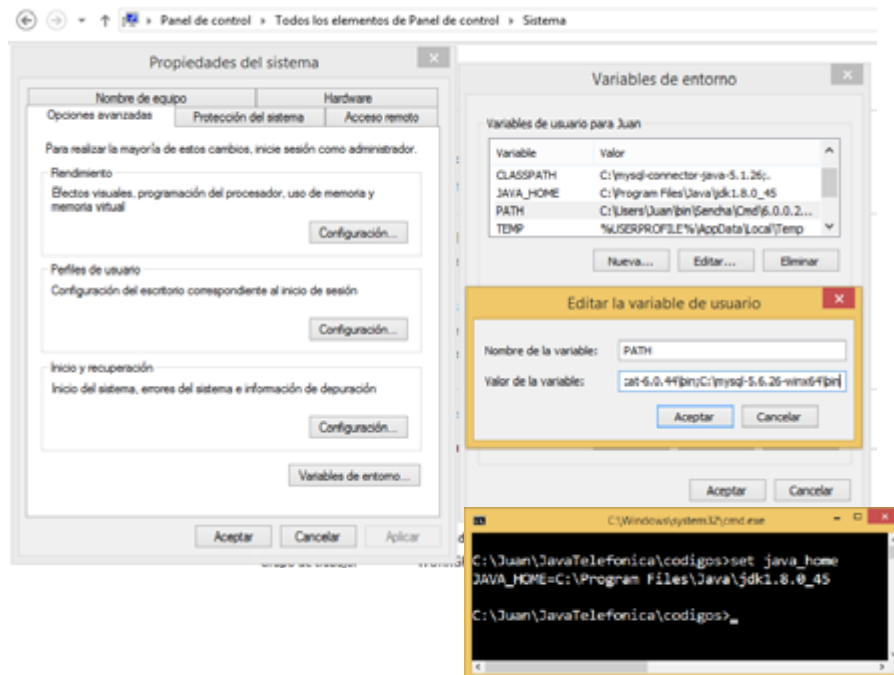
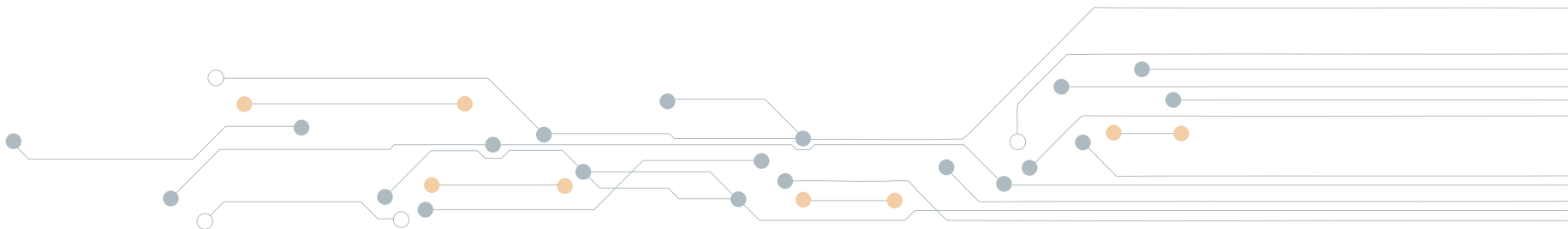


FIGURA 1.8: VARIABLES DE ENTORNO PATH Y JAVA\_HOME.

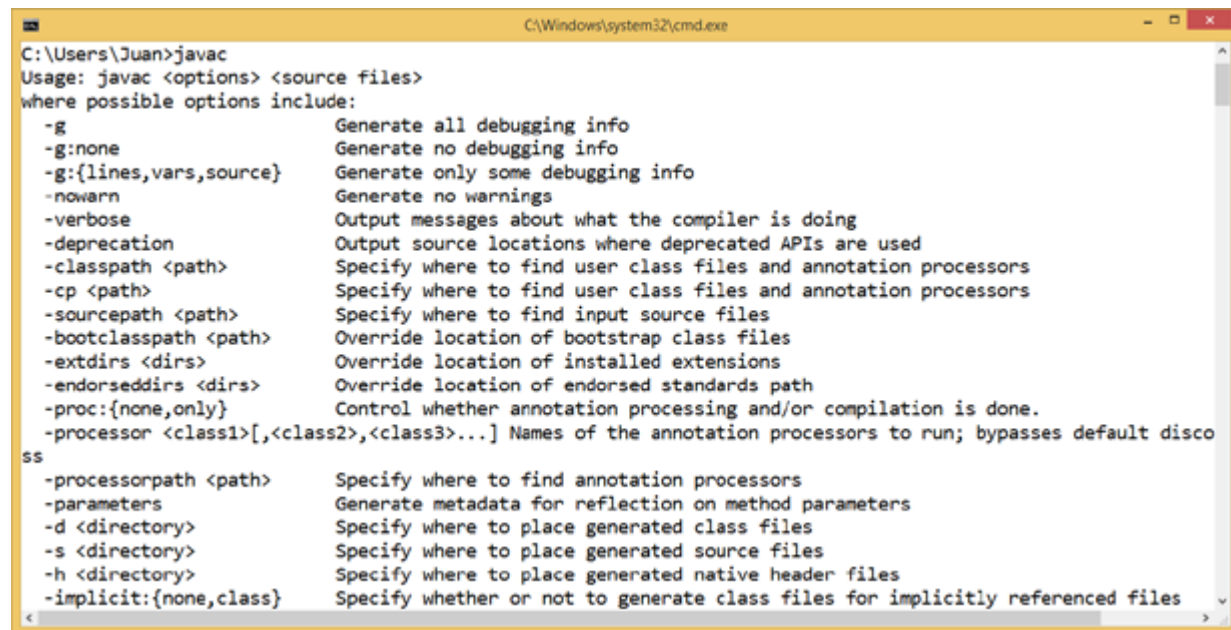
Una vez comprobado que dichas variables tienen los valores adecuados, se puede comprobar ejecutando desde una sesión de consola, comando del sistema "cmd", que cualquier programa de la carpeta "bin" del directorio del JDK se ejecuta correctamente.

Los primeros programas que se van a utilizar son:

- javac, el compilador de Java.
- java, para ejecutar las aplicaciones Java.



En la imagen siguiente se puede visualizar lo que se muestra si se ejecuta el compilador "javac" sin indicar ningún parámetro a su derecha. Este programa, como "java", cuando se ejecuta sin parámetros muestra la ayuda del propio programa.



```
C:\Windows\system32\cmd.exe
C:\Users\Juan>javac
Usage: javac <options> <source files>
where possible options include:
  -g               Generate all debugging info
  -g:none          Generate no debugging info
  -g:{lines,vars,source}  Generate only some debugging info
  -nowarn          Generate no warnings
  -verbose         Output messages about what the compiler is doing
  -deprecation     Output source locations where deprecated APIs are used
  -classpath <path>  Specify where to find user class files and annotation processors
  -cp <path>        Specify where to find user class files and annotation processors
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path>  Override location of bootstrap class files
  -extdirs <dirs>      Override location of installed extensions
  -endorseddirs <dirs>  Override location of endorsed standards path
  -proc:{none,only}  Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery
  -processorpath <path> Specify where to find annotation processors
  -parameters      Generate metadata for reflection on method parameters
  -d <directory>    Specify where to place generated class files
  -s <directory>    Specify where to place generated source files
  -h <directory>    Specify where to place generated native header files
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
```

FIGURA 1.8: AYUDA DE JAVAC, AL EJECUTARLO SIN NINGÚN PARÁMETRO.

## 1.4 | Funcionamiento de Java

En el lenguaje de programación Java, el código fuente se escribe en archivos con extensión ".java". El compilador "javac" convierte los códigos fuentes en código binario con extensión ".class", si no hay ningún error ni léxico, ni sintáctico, ni semántico en el código fuente.

El código binario de los archivos ".class", no es nativo para el procesador en el que se está trabajando, son códigos binarios denominados "bytecode" para la máquina virtual de Java (Java Virtual Machine), a partir de ahora será llamada JVM. Esta es la razón por la que el código Java es portable, de uno sistemas a otros, la razón por la que sus creadores lo definieron como *"Write Once, Run Anywhere"*.

Para poder ejecutar el código para un sistema operativo en concreto hay que utilizar el programa "java" y lanzar el archivo ".class", con esto se está convirtiendo el código "bytecode" al código binario nativo del procesador correspondiente. Cuando se instaló Java, se eligió un S.O., por tanto se instaló la JVM apropiada para dicho S.O.

La imagen siguiente ilustra este proceso:

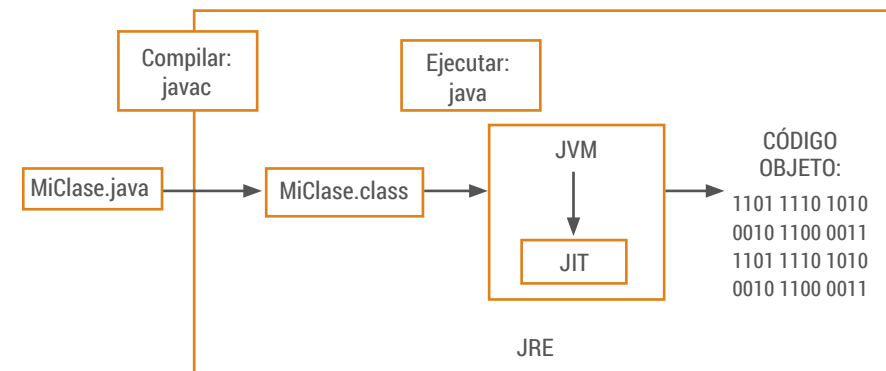


FIGURA 1.9: COMPILAR Y EJECUTAR EN JAVA.

Los ficheros ".class" son interpretados por JVM y compilados a código nativo por JIT (just in time), las primeras JVM sólo eran interpretadas.

HotSpot es el compilador JIT que traduce los "bytecodes" al lenguaje máquina nativo. Cuando JVM encuentra el código compilado, se ejecuta directamente el código en lenguaje máquina, que es más rápido.

Por tanto, los programas en Java pasan por dos fases de compilación:

- En la propia compilación (javac) código fuente se traduce a "bytecodes", para tener portabilidad a través de las diferentes JVMs de los distintos procesadores y sistemas operativos.
- En la ejecución (java), los "bytecodes" se traducen en lenguaje máquina nativo para el procesador en el que se ejecuta el programa.

En la imagen siguiente, se muestra como en la segunda fase se genera el código nativo por el compilador "en ejecución" JIT:

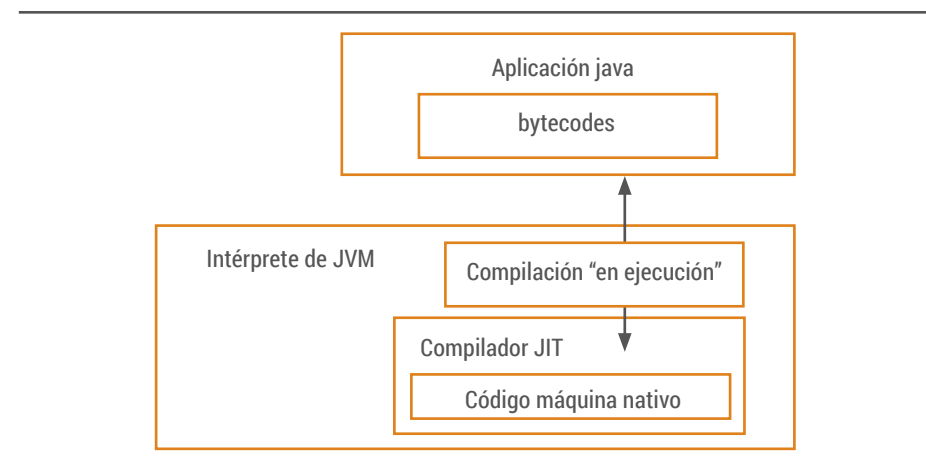


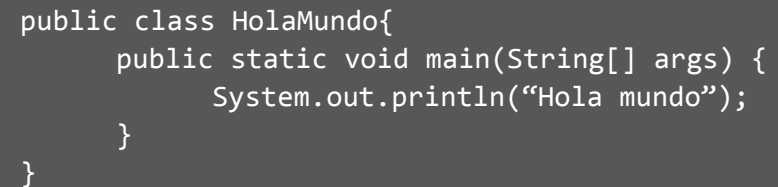
FIGURA 1.9: COMPILAR Y EJECUTAR EN JAVA.

## 1.5 | Compilar y ejecutar programas en Java

Para poder escribir el código fuente de un programa en Java, no se necesita nada más que un editor de texto, como en Windows puede ser "Notepad.exe". Pero es normal que en vez de utilizar esta herramienta tan básica, se utilicen entornos de desarrollo que incorporan e integran funcionalidades y utilidades que mejoran el rendimiento en la creación y actualización de los códigos Java de las aplicaciones, porque permiten editar, compilar, ejecutar, depurar y chequear todos los elementos que en Java forman parte de una aplicación.

En próximo apartado se hará mención de dos de los más utilizados entornos integrados de desarrollo (IDEs), como son Eclipse y NetBeans. Pero en este momento se va a estudiar cómo crear, compilar y ejecutar un pequeño programa Java, el clásico "Hola mundo".

Con el editor mencionado (Notepad.exe), se crea el archivo siguiente:



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

El archivo tiene que tener como nombre de forma obligada "HolaMundo.java". Un programa en Java, tiene que tener una clase que tenga definida la función main. Esta función es por donde empieza a ejecutarse el código.

El compilador "javac" compila el código fuente del archivo "HolaMundo.java" y si no hay errores crea el archivo en código binario "bytecode" en el archivo "HolaMundo.class".

El programa "java" cuando se invoque con "HolaMundo.class", estará invocando a JVM para que ejecute en el código nativo el "bytecode" del fichero ".class".

La JVM, buscara en dicho código una clase que se llame igual que el archivo, en este caso "HolaMundo" y que tenga definida la función "main". Si la encuentra ejecutara las sentencias que contenga.

De momento y en este punto, una clase es un código Java encerrado entre llaves, las llaves siempre definen un conjunto de sentencias. En este caso el bloque de la clase "HolaMundo" contiene la función "main".

Una función es un conjunto de sentencias, encerradas entre llaves (bloque de sentencias de la función) que se ejecutan cuando dicha función es llamada. En este ejemplo, la función "main" es invocada por la JVM.

La imagen siguiente muestra como se llega a ejecutar el programa Java, empezando con su edición.

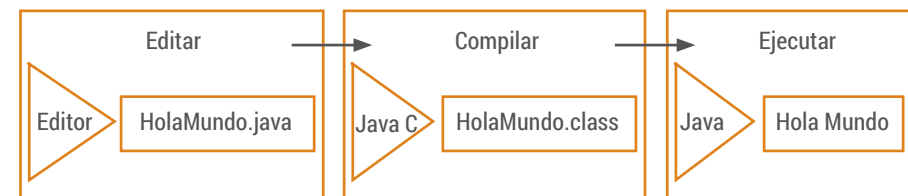
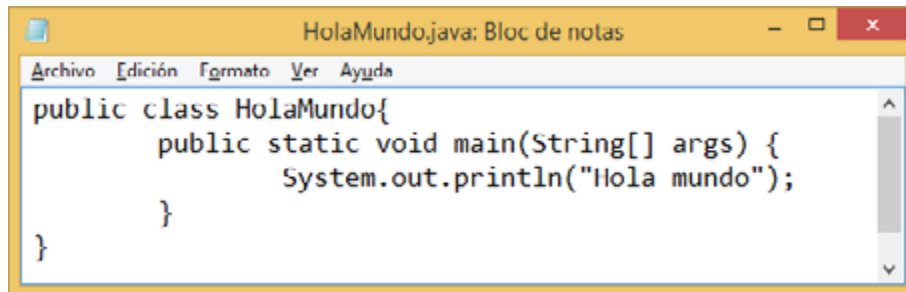


FIGURA 1.9: COMPILAR Y EJECUTAR EN JAVA.

En las imágenes siguientes, se ilustra el proceso paso a paso.

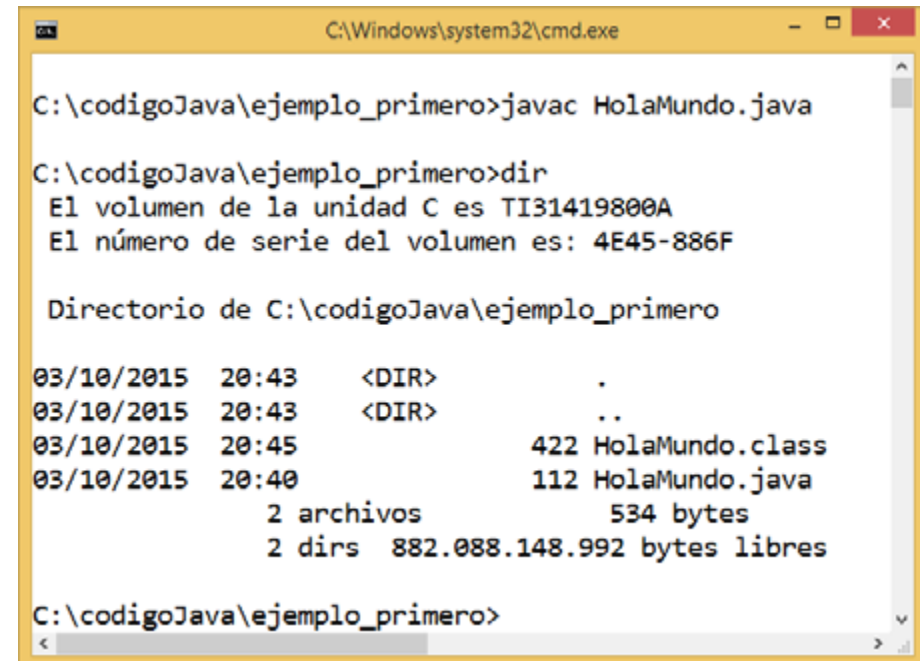
- Editar el código con "Notepad" y guardarlo en el archivo "HolaMundo.java":



```
HolaMundo.java: Bloc de notas
Archivo Edición Formato Ver Ayuda
public class HolaMundo{
    public static void main(String[] args) {
        System.out.println("Hola mundo");
    }
}
```

FIGURA 1.12: ESCRIBIR EL CÓDIGO FUENTE EN EDITOR "NOTEPAD".

- Compilar el código con "javac":



```
C:\Windows\system32\cmd.exe

C:\codigoJava\ejemplo_primer0>javac HolaMundo.java

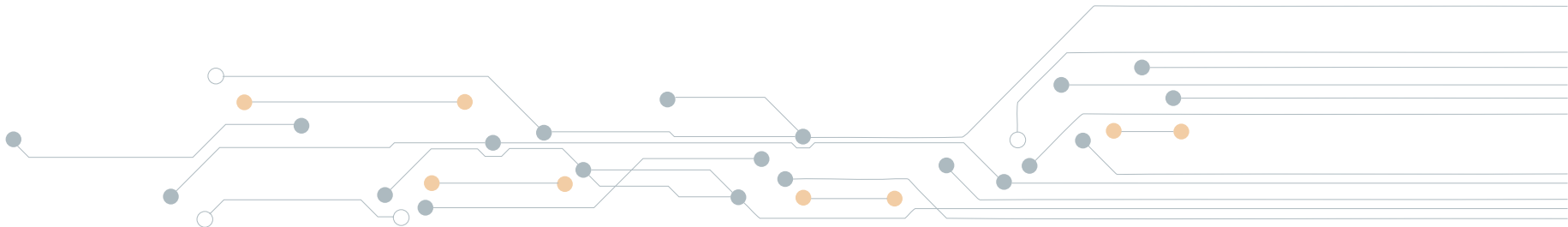
C:\codigoJava\ejemplo_primer0>dir
El volumen de la unidad C es TI31419800A
El número de serie del volumen es: 4E45-886F

Directorio de C:\codigoJava\ejemplo_primer0

03/10/2015  20:43    <DIR>          .
03/10/2015  20:43    <DIR>          ..
03/10/2015  20:45                422 HolaMundo.class
03/10/2015  20:40                112 HolaMundo.java
                                2 archivos          534 bytes
                                2 dirs 882.088.148.992 bytes libres

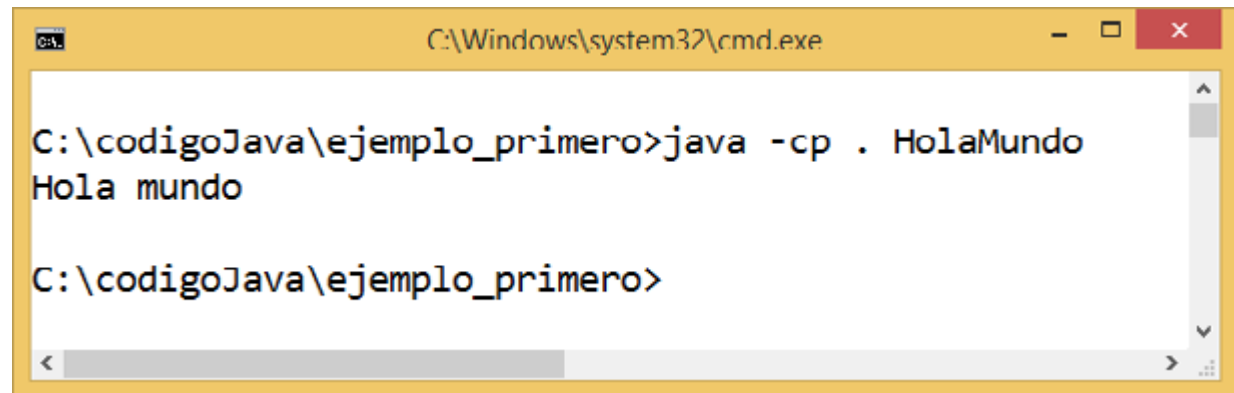
C:\codigoJava\ejemplo_primer0>
```

FIGURA 1.13: COMPILAR CON "JAVAC".





- Ejecutar con "java". La opción "-cp .", indica que las clases que se necesitan para que la JVM puede generar y ejecutar el código, están en el mismo directorio desde el que se invoca a "java". Esto es lo que definiremos más adelante con la tercera variable de entorno importante, la variable CLASSPATH:



```
C:\Windows\system32\cmd.exe

C:\codigoJava\ejemplo_primer>java -cp . HolaMundo
Hola mundo

C:\codigoJava\ejemplo_primer>
```

FIGURA 1.14: EJECUTAR EL CÓDIGO CON "JAVA".

## 1.6 | Entornos de desarrollo

Un entorno de desarrollo es una herramienta software que facilita la organización, creación, compilación, depuración y pruebas de las aplicaciones Java.

Incluyen administradores de archivos y proyectos. Permiten trabajar con diferentes tipos de proyectos, por ejemplo proyectos para aplicaciones Java, para aplicaciones web con tecnología JEE e incluso otras, creación de componentes con tecnologías como JEE.

También ayudan a las aplicaciones que tienen que acceder a bases de datos, con la infraestructura para hacer las conexiones apropiadas, con las principales y más conocidas bases de datos como Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Microsoft Access, Apache Derby.

En las aplicaciones web permiten ejecutar dichas aplicaciones desde el IDE, mientras se está construyendo la versión definitiva, integrando servidores web para poder desplegar y ejecutar dichas aplicaciones. Por ejemplo, permiten integrar el popular servidor web Apache Tomcat.

En aplicaciones de tipo empresarial JEE (aplicaciones distribuidas en la red), permiten integrar los servidores de aplicaciones para desplegar los componentes e ir ejecutando dichas aplicaciones en sus diferentes fases, hasta llegar a la versión final. Entre los servidores de aplicaciones que suelen permitir su integración los IDEs son JBoss, WebLogic de Oracle, GlashFish de Oracle, WebSphere de IBM o Geronimo de Apache.



Una de las características más importantes de estas herramientas de desarrollo, es la poder extender sus funcionalidades con plugins que se pueden ir instalando o desinstalando según las necesidades.

Por último una de sus ventajas fundamentales, es la de poder trabajar con los principales “frameworks” (estructuras de trabajo especializadas), como por ejemplo Spring, Struts, Hibernate, JSF, etc.

Los dos IDEs, de distribución gratuita más utilizados por la comunidad de desarrolladores de Java son:

- NetBeans de Oracle, en la actualidad está disponible la versión 8, y se puede descargar desde cualquiera de estas direcciones:

<https://netbeans.org/>

<http://www.oracle.com/technetwork/developer-tools/netbeans/downloads/index.html>

<http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>

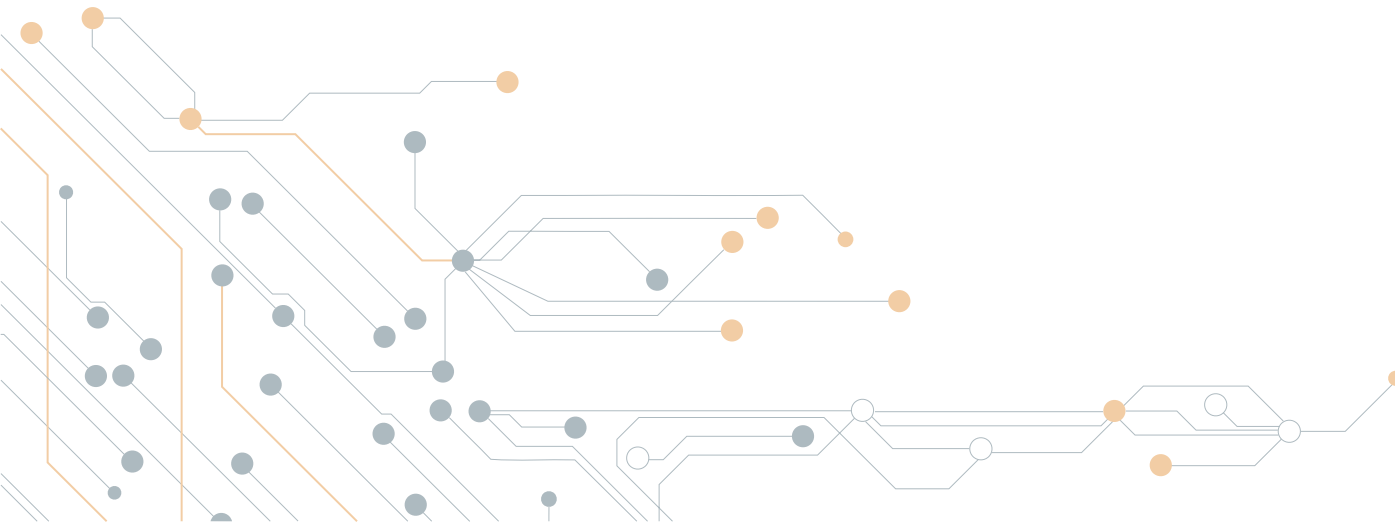
Con esta última dirección se instala después de hacerlo con la edición de Java JSE.

- Eclipse, la versión actual es la denominada “Mars”, y se puede descargar desde la dirección:

<https://eclipse.org/>

También se pueden instalar versiones anteriores o versiones diferentes de esta última edición, accediendo a la dirección siguiente:

<https://eclipse.org/downloads/>



Cualquiera de estos dos IDEs tiene versiones tanto para sistemas operativos Windows, como Linux, como Mac OS.

Para poder instalar Eclipse se deben seguir los pasos siguientes:

- Acceder al enlace:

<https://eclipse.org>

- Pulsar sobre el botón **“Download”** para empezar la descarga e instalación, en el momento actual es la versión “Mars” (4.5)
- Elegir la opción **“Eclipse IDE for Java EE Developers”** para el sistema operativo que corresponda.
- Seguir todos los pasos e indicaciones y una vez terminada la instalación, ejecutar Eclipse.
- Para conocer y estudiar este IDE se puede consultar la guía de inicio del enlace:

[help.eclipse.org/mars/index.jsp](http://help.eclipse.org/mars/index.jsp)

- Cuando se ejecuta Eclipse lo primero que hay que definir es el **“workspace”** en el que se van a almacenar o recuperar los proyectos. Un “workspace” no es nada más que un directorio en disco, donde se almacenan todos los elementos que necesitan los proyectos de Eclipse. se pueden tener tantos “workspaces” como se considere necesario.

- Si se necesita crear un “workspace” nuevo, simplemente hay que posicionarse en una carpeta determinada, después de pulsar sobre el botón “Browse”.

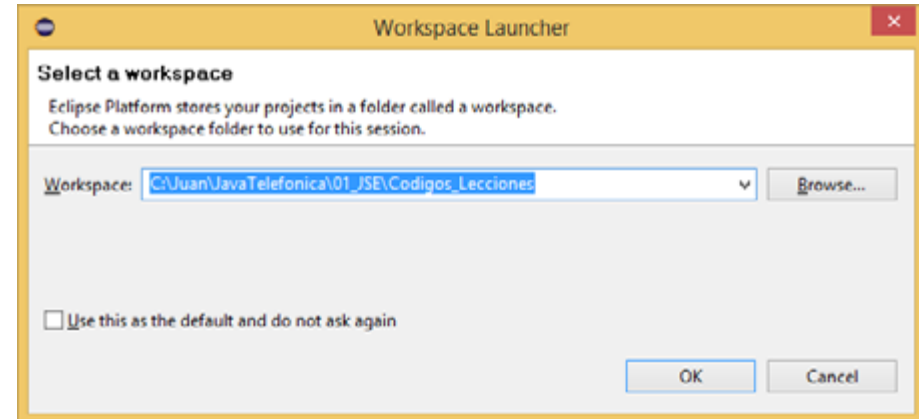
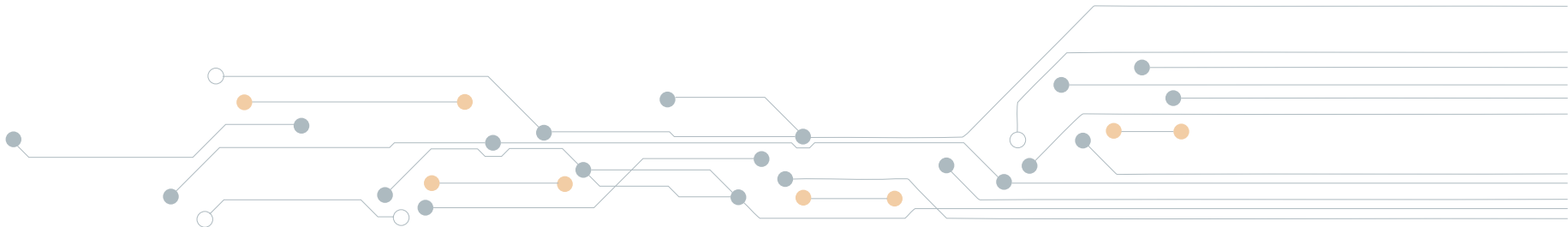


FIGURA 1.15: ELEGIR “WORKSPACE” EN ECLIPSE.

- La primera vez que se accede a un “workspace” aparece una ventana de bienvenida “Welcome to Eclipse” que se puede cerrar para que no vuelva a aparecer en este “workspace”.



- Una vez abierto Eclipse, como en casi todos los IDEs, de forma muy genérica se encuentran cinco zonas, indicadas en la figura siguiente.

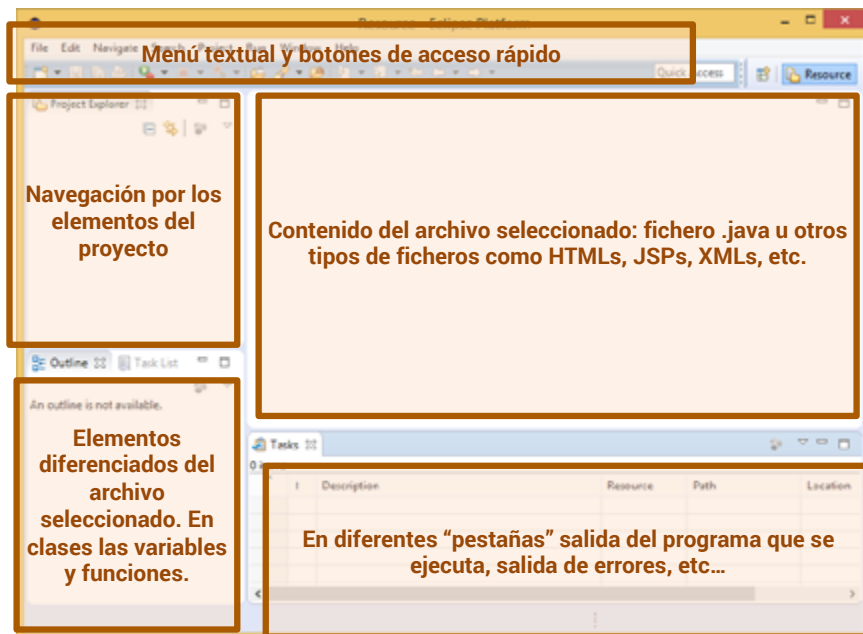


FIGURA 1.16: VENTANA DE ECLIPSE.

- Para poder crear un programa o aplicación en Eclipse, hay que ejecutar el comando **"New"** (nuevo) del menú **"File"** (archivo) y elegir la opción **"Project"**.

Aparece una ventana que permite elegir uno de los tipos de proyectos que se pueden utilizar en este IDE. Para poder escribir un programa Java se debe seleccionar "Java Project", tal y como indica la figura siguiente.

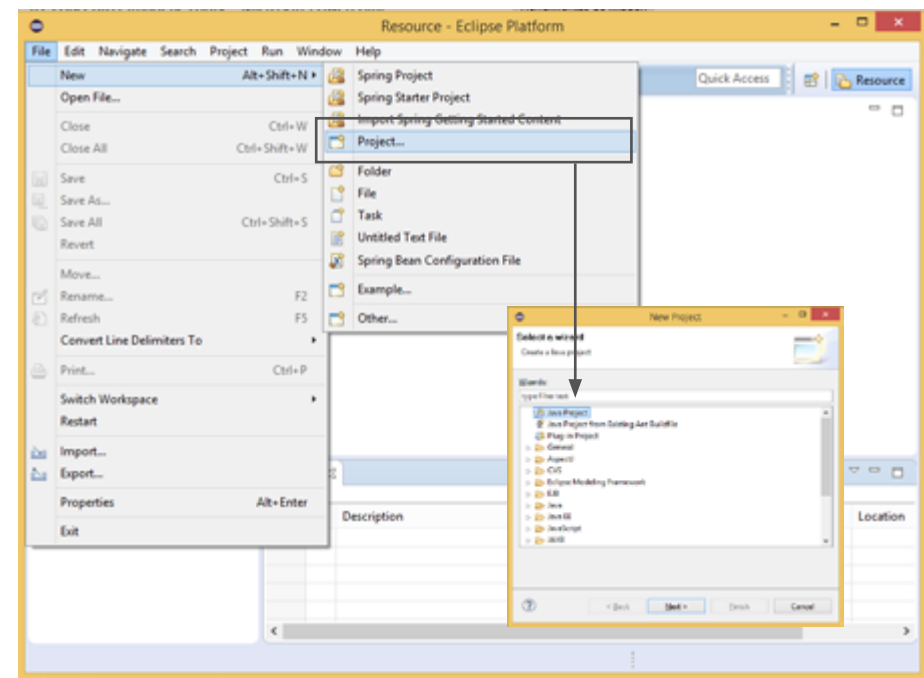


FIGURA 1.17: CREAR UN PROYECTO NUEVO.

- En la ventana que aparece a continuación se deberá poner un nombre al proyecto, que será una carpeta almacenada en el directorio del "workspace". Esta carpeta de proyecto, dependiendo del tipo del mismo tendrá una determinada estructura. Para proyectos java, básicamente consistirá en una carpeta "src" para los archivos fuente (.java) y otra "bin" para los archivos "bytecode" (.class), tal y como se muestra en la imagen siguiente:

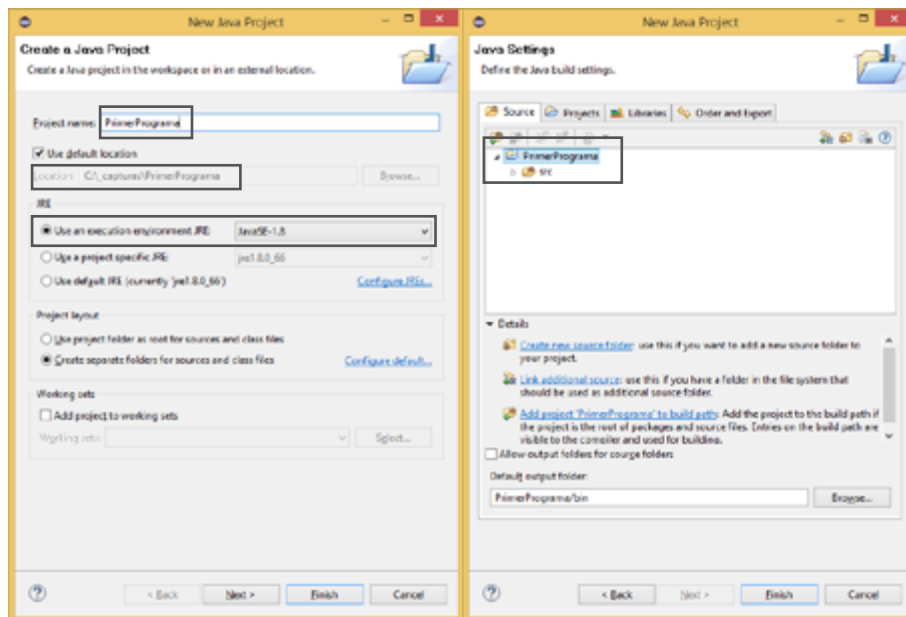


FIGURA 1.18: VENTANAS PARA CREAR UN NUEVO PROYECTO.

- En este punto, se deben ir añadiendo los elementos que necesite el proyecto. En el caso de una aplicación simple de Java habrá que añadir por lo menos una clase que tenga la función main. Una forma de hacerlo es pulsar con el botón secundario del ratón sobre el nombre del proyecto, apareciendo menú, cuya primera opción es **"New"** (nuevo), al pulsar sobre esta aparece otro menú desde el que se puede elegir el tipo de elemento que se quiere añadir al proyecto. En los proyectos Java se deberá seleccionar **"Class"** y más adelante también se añadirán elementos **"Package"**, **"Interface"**, **"Enum"** y **"Annotation"**. La imagen siguiente muestra esta situación.

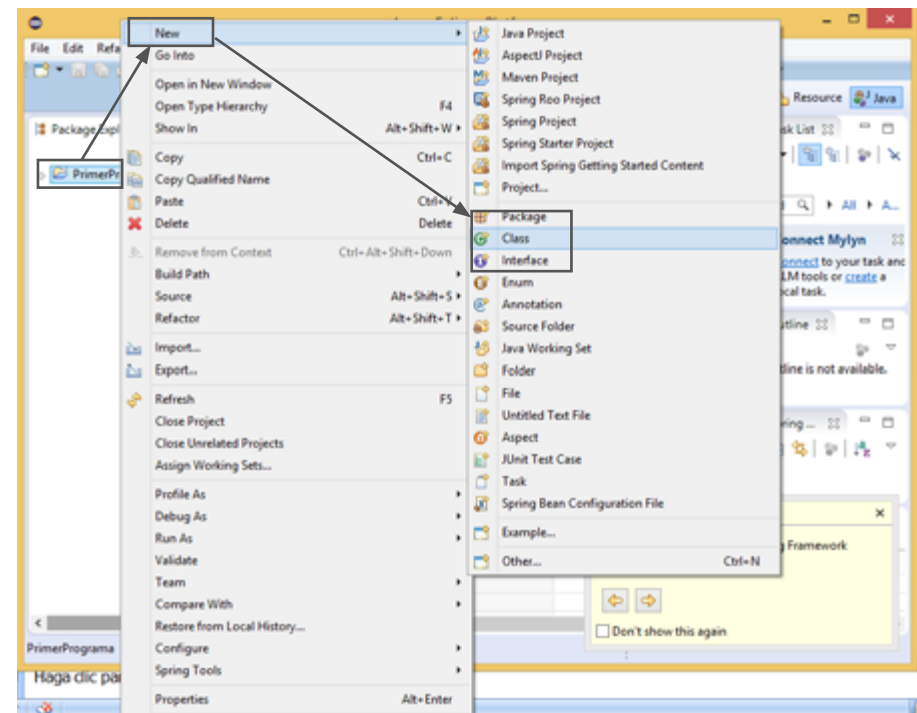


FIGURA 1.19: AÑADIR ELEMENTOS A UN PROYECTO.

- En el caso de una clase Java con función main, habrá que escribir el nombre de la clase, el paquete donde ubicarla e indicar que se quiere incluir la función main, tal y como indica la imagen.

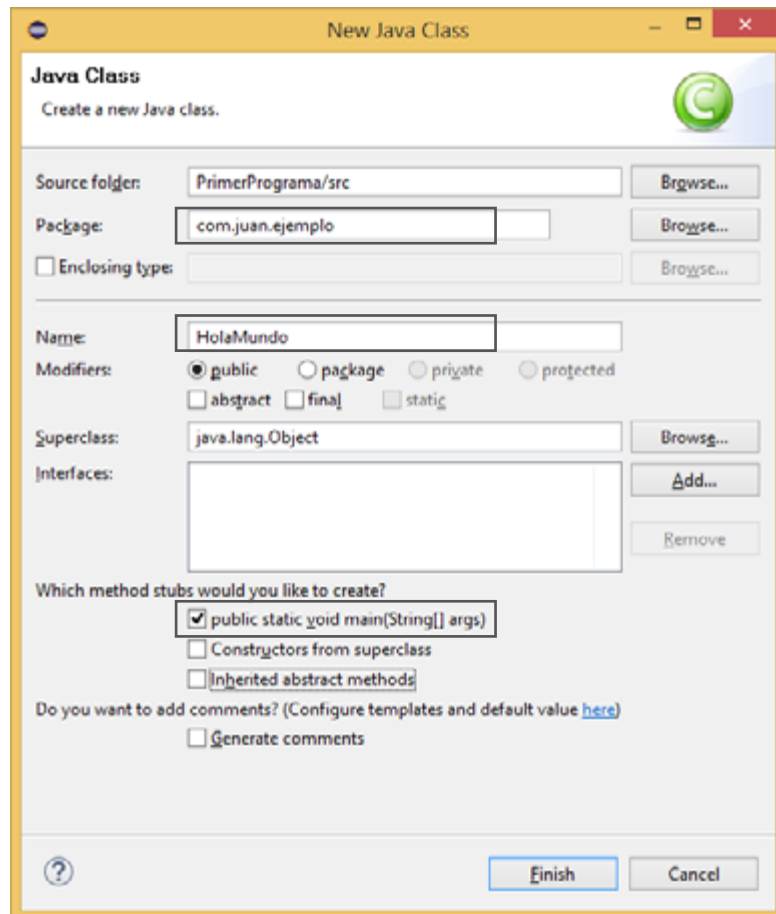


FIGURA 1.19: DEFINIR LOS DATOS DE LA CLASE.

- Una vez realizados los pasos anteriores ya esta creada la clase con la función main, esperando que le programador añada el código necesario.

En Eclipse, no se compila explícitamente, los errores se van señalando según se va escribiendo el código. El IDE ira ayudando y sugiriendo al programador para que escriba un código lo más libre posible de errores léxicos y sintácticos.

No hay que olvidar que de vez en cuando hay que ir guardando lo realizado en disco, para ello se utilizan las opciones clásicas en todas las aplicaciones para este propósito.

La imagen siguiente ilustra cómo es la ventana para una clase recién creada.

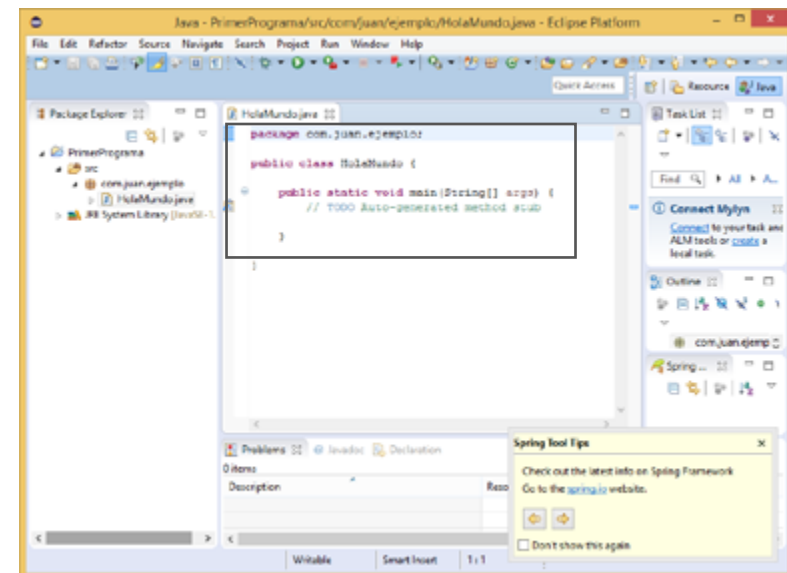


FIGURA 1.20: VISTA DEL PROYECTO CON LA CLASE CREADA.

- Para ejecutar una aplicación Java en Eclipse se tienen varios procedimientos, dos se detallan en las imágenes siguientes. También se puede pulsar con botón secundario sobre nombre del proyecto y elegir en menú contextual la opción **"Run As"** y **"Java Application"**. Lo mismo sucede si se pulsa con botón secundario del ratón sobre el código de la clase.

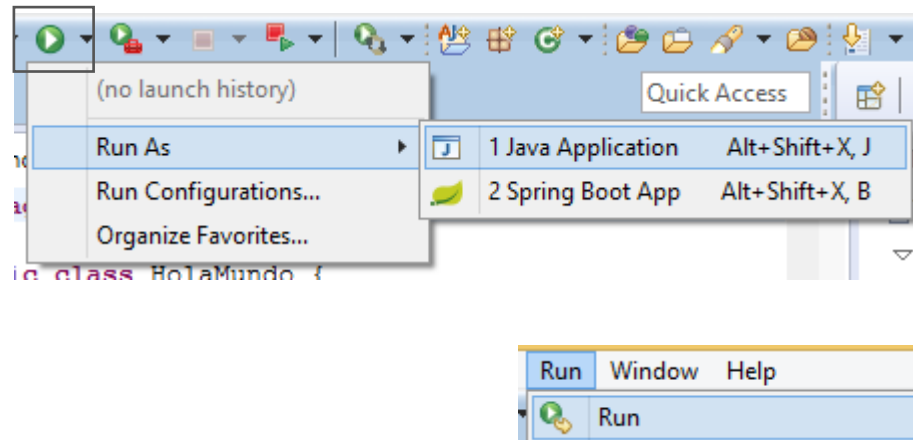


FIGURA 1.20: VISTA DEL PROYECTO CON LA CLASE CREADA.



*Telefonica*

---

EDUCACIÓN DIGITAL