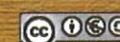


Unidad 3: Programación básica en Lenguaje Java

Fundamentos de Programación. 1º de ASI



Esta obra está bajo una licencia de Creative Commons.
Autor: Jorge Sánchez Asenjo (año 2009) <http://www.jorgesanchez.net>
e-mail:info@jorgesanchez.net

Esta obra está bajo una licencia de Reconocimiento-NoComercial-CompartirIgual de Creative Commons
Para ver una copia de esta licencia, visite:
<http://creativecommons.org/licenses/by-nc-sa/2.5/es/legalcode.es>
o envíe una carta a:
Creative Commons, 559 Nathan Abbot



Reconocimiento-NoComercial-CompartirIgual 2.5 España

Usted es libre de:



copiar, distribuir y comunicar públicamente la obra



hacer obras derivadas

Bajo las condiciones siguientes:



Reconocimiento. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).



No comercial. No puede utilizar esta obra para fines comerciales.



Compartir bajo la misma licencia. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

- Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
- Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
- Apart from the remix rights granted under this license, nothing in this license impairs or restricts the author's moral rights.

Advertencia

Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
Esto es un resumen legible por humanos del texto legal (la licencia completa) disponible en los idiomas siguientes:
Catalán Castellano Euskera Gallego

Para ver una copia completa de la licencia, acudir a la dirección <http://creativecommons.org/licenses/by-nc-sa/2.5/es/legalcode.es>

(3)

programación

básica en Java

esquema de la unidad

(3) programación básica en Java	5
(3.1) historia de Java	8
(3.1.1) los antecedentes de Java. influencia de C y C++	8
(3.1.2) la llegada de Java	10
(3.1.3) Java y JavaScript	12
(3.2) características de Java	12
(3.2.1) construcción de programas en Java. bytecodes	12
(3.2.2) seguridad	14
(3.2.3) tipos de aplicaciones Java	15
(3.2.4) plataformas	15
(3.3) empezar a trabajar con Java	16
(3.3.1) el kit de desarrollo Java (SDK)	16
(3.3.2) versiones de Java	16
(3.3.3) instalación del SDK	19
(3.3.4) entornos de trabajo	22
(3.4) escritura de programas Java	24
(3.4.1) codificación del texto	24
(3.4.2) notas previas	24
(3.4.3) el primer programa en Java	25
(3.5) ejecución de programas Java	26
(3.5.1) proceso de compilación desde la línea de comandos	26
(3.6) javadoc	27
(3.7) import	29
(3.8) variables	30
(3.8.1) introducción	30
(3.8.2) declaración de variables	31
(3.8.3) asignación	31

(I.vii.ix) limpiar	67
(I.vii.x) exportar preferencias	69
(I.viii) creación de javadoc con Eclipse	70
Apéndice (II) Netbeans	73
(II.i) introducción	73
(II.ii) instalación	73
(II.ii.i) comprobación de la plataforma Java instalada	76
(II.iii) aspecto inicial de Netbeans	76
(II.iii.i) cerrar y abrir paneles	77
(II.iii.ii) iconizar un panel	77
(II.iii.iii) mover paneles	77
(II.iii.iv) mostrar y quitar barras de herramientas	77
(II.iv) proyectos de Netbeans	78
(II.iv.i) crear proyectos	78
(II.iv.ii) la ventana del proyecto	79
(II.iv.iii) crear paquetes	80
(II.iv.iv) crear nuevas clases	80
(II.iv.v) vistas del proyecto	81
(II.iv.vi) propiedades del proyecto	82
(II.iv.vii) borrar un proyecto	83
(II.iv.viii) importar un proyecto de Eclipse	83
(II.v) compilar y ejecutar programas	84
(II.v.i) compilar	84
(II.v.ii) ejecutar la clase principal del proyecto	84
(II.v.iii) preparar la distribución	84
(II.vi) javadoc	85
(II.vi.i) añadir la documentación Java a Netbeans	85
(II.vi.ii) generar la documentación Javadoc del proyecto	85
(II.vii) edición de código	86
(II.vii.i) aspecto de la ventana de código	86
(II.vii.ii) ayudas al escribir código	87
(II.vii.iii) búsqueda y reemplazo de texto en el código	87
(II.vii.iv) dar formato al código	88
(II.vii.v) modificación de opciones del editor	88
(II.vii.vi) reestructurar	89
(II.vii.vii) plantillas generales	90
(II.vii.viii) comparar archivos	91
(II.vii.ix) exportar e importar las opciones	92

C aportó a los lenguajes existentes las siguientes ventajas:

- ◆ Un lenguaje de nivel medio (más cercano a la forma de pensar del ordenador) que permitía tanto utilizar estructuras de los lenguajes de alto nivel (funciones, bucles avanzados,...) como instrucciones de nivel bajo (punteros)
- ◆ Una sintaxis que permite escribir código de forma rápida
- ◆ Un lenguaje potente capaz de crear todo tipo de aplicaciones
- ◆ Un lenguaje capaz de utilizar todo tipo de estructuras estáticas y dinámicas y de manejar todos los recursos de la máquina.

Sin embargo C también tiene sus problemas. Uno de los principales es que cuando la aplicación crece, el código es muy difícil de manejar. Las técnicas de programación estructurada y programación modular, que en C pueden ser aplicadas, paliaban algo el problema. Pero fue la **programación orientada a objetos (POO u OOP)** la que mejoró notablemente el situación.

No obstante C sigue siendo uno de los lenguajes más utilizados y académicamente sigue utilizándose por su versatilidad, que permite aprender todas las características de la programación clásica. De hecho a un buen programador de lenguaje C no le debería ser difícil aprender a programar en otros lenguajes (una vez que conozca las bases de la programación orientada a objetos).

La influencia de la programación orientada a objetos

La **POO** permite fabricar programas de forma más parecida al pensamiento humano. De hecho simplifica el problema dividiéndolo en objetos y permitiendo centrarse en cada objeto, para de esa forma eliminar la complejidad. Cada objeto se programa de forma autónoma y esa es la principal virtud.

Al aparecer la programación orientada a objetos (en los años setenta), aparecieron varios lenguajes orientados a objetos y también se realizaron versiones orientadas a objetos (o semiorientadas a objetos) de lenguajes clásicos.

Una de las más famosas adaptaciones fue la que capacitó al lenguaje C a utilizar objetos. A ese lenguaje se le llamó **C++** indicando con esa simbología que era un incremento del lenguaje C (en el lenguaje C, como en Java, los símbolos **++** significan incrementar). Las ventajas que añadió C++ a C fueron:

- ◆ Añadir soporte para objetos (POO)
- ◆ Librerías de clases de objetos (como **MFC**¹ por ejemplo) que facilitaban el uso de código ya creado para las nuevas aplicaciones.
- ◆ Todo lo bueno del C (incluso compatibilidad con este lenguaje)

¹ **Microsoft Foundation Classes**, librería creada por Microsoft para facilitar la creación de programas para el sistema Windows.

es tomarse una taza de café (aunque no sea precisamente de Java). Parece que los desarrolladores de Java tomaron muchas tazas de **Java**.²

Ese año se da a conocer al público, y adquiere notoriedad rápidamente, casi desde su lanzamiento. Durante estos años se ha mejorado y se le ha revisado. La versión 1.2 modificó tanto Java que se la llamó **Java 2** y también a sus descendientes (Java 1.3 y Java 1.4). Actualmente el número 2 se ha quitado del nombre y la última versión se conoce como **Java v6**.

En general la sintaxis de Java es similar a C y C++. Pero posee estas diferencias:

- ◆ No hay punteros (lo que le hace más seguro y fácil de manejar)
- ◆ No es híbrido, sino totalmente orientado a objetos (aunque muchos programadores tienen reservas respecto a esta aseveración). Los lenguajes orientados a objetos híbridos permiten crear aplicaciones no orientadas a objetos.
- ◆ Muy preparado para ser utilizado en redes TCP/IP y especialmente en Internet
- ◆ Implementa excepciones (control de errores) de forma nativa
- ◆ Es un lenguaje interpretado (lo que acelera su ejecución remota, aunque provoca que las aplicaciones Java sean más lentas en la ejecución que las aplicaciones escritas en lenguajes compilados como C++).
- ◆ Permite múltiples hilos de ejecución, es decir que se ejecuten varias tareas en paralelo.
- ◆ Admite firmas digitales
- ◆ Tipos de datos y control de sintaxis más rigurosa que los lenguajes C y C++, lo que facilita la gestión de errores
- ◆ Es independiente de la plataforma, ejecutable en cualquier sistema con máquina virtual

La última ventaja (quizá la más importante) se consigue ya que el código Java no se compila, sino que se **precompila**, de tal forma que se crea un **código intermedio** que no es directamente ejecutable. No es código máquina. Para ejecutarle hace falta pasarle por un intérprete que va interpretando cada línea. Ese intérprete suele ser la **máquina virtual de Java**. Por lo que cualquier sistema que posea máquina virtual de Java, podrá ejecutar código precompilado en Java. Más adelante se explica este proceso en detalle.

² La simbología del café sigue presente en Java. El logotipo oficial de Java es una taza humeante de café. Mucho software desarrollado para Java ha mantenido esa simbología: Visual Café, Kawa (*café en ruso*),...

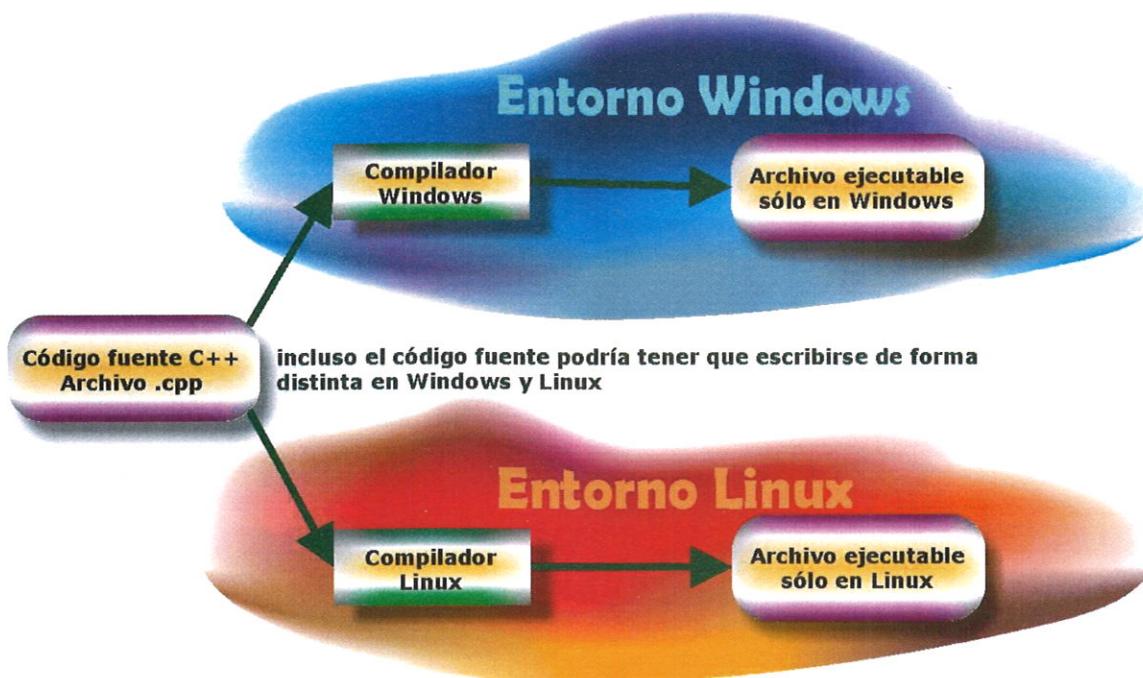


Ilustración 3-1, Proceso de compilación de un programa C++ en Windows y Linux

la "compilación" en Java

En Java el código no se traduce a código ejecutable. En Java el proceso se conoce como **precompilación** y sirve para producir un archivo (de extensión **class**) que contiene código que no es directamente ejecutable (no es código Java). Es un código intermedio llamado **bytecode** (también se le llama **J-code**).

Al no ser ejecutable, el archivo **class** no puede ejecutarse directamente con un doble clic en el sistema. El bytecode tiene que ser **interpretado** (es decir, traducido línea a línea) por una aplicación conocida como la **máquina virtual de Java (JVM)**. Hoy se conoce como **JRE (Java Runtime Environment**, entorno de ejecución de Java).

La gran ventaja es que el entorno de ejecución de Java lo fabrica Sun para todas las plataformas; lo que significa que un archivo **class** se puede ejecutar en cualquier ordenador o máquina que incorpore el JRE. Sólo hay una pega, si programamos utilizando por ejemplo la versión 1.6 de Java, el ordenador en el que queramos ejecutar el programa deberá incorporar el JRE al menos de la versión 1.6.

El JRE o la máquina virtual de Java son un programas muy pequeños y que se distribuyen gratuitamente para prácticamente todos los sistemas operativos.

A la forma de producir código final de Java se la llama **JIT (Just In Time**, justo en el momento) ya que el código ejecutable se produce sólo en el instante de ejecución del programa. Es decir, no hay en ningún momento código ejecutable.

(3.2.3) tipos de aplicaciones Java

applets

Son programas Java pensados para ser colocados dentro de una página web. Pueden ser interpretados por cualquier navegador con capacidades Java. Estos programas se insertan en las páginas usando una etiqueta especial (como también se insertan vídeos, animaciones flash u otros objetos).

Los applets son programas independientes, pero al estar incluidos dentro de una página web las reglas de éstas le afectan. Normalmente un applet sólo puede actuar sobre el navegador.

Hoy día mediante applets se pueden integrar en las páginas web aplicaciones multimedia avanzadas (incluso con imágenes 3D o sonido y vídeo de alta calidad)

aplicaciones de consola

Son programas independientes al igual que los creados con los lenguajes tradicionales.

aplicaciones gráficas

Aquellas que utilizan las clases con capacidades gráficas (como awt por ejemplo).

servlets

Son aplicaciones que se ejecutan en un servidor de aplicaciones web y que como resultado de su ejecución resulta una página web.

midlet

Aplicación creada con Java para su ejecución en sistemas de propósito simple o dispositivos móviles. Los juegos Java creados para teléfonos móviles son midlets.

(3.2.4) plataformas

Actualmente hay tres ediciones de Java. Cada una de ellas se corresponde con una plataforma que incluye una serie de funciones, paquetes y elementos del lenguaje (es decir la **API**, *Application Program Interface*).

Java SE

Java Standard Edition. Antes se la conocía como **J2SE** (el dos se refiere a Java 2). Permite escribir código Java relacionado con la creación de aplicaciones y applets en lenguaje Java común. Es decir, es el Java normal. La última versión del kit de desarrollo de aplicaciones en esta plataforma³ es la JSE 1.8.14.0. *20/06/10/2*

³ En el momento de escribir este manual

Java 1.0 (JDK 1.0)

Fue la primera versión de Java y propuso el marco general en el que se desenvuelve Java. está oficialmente obsoleto, pero hay todavía muchos clientes con esta versión.

Java 1.1 (JDK 1.1)

Mejoró la versión anterior incorporando las siguientes mejoras:

- ◆ JDBC, API de acceso a bases de datos
- ◆ RMI llamadas a métodos remotos. Es una técnica de comunicación de procesos en red
- ◆ JavaBeans, componentes independientes reutilizables.
- ◆ Internacionalización para crear programas adaptables a todos los idiomas
- ◆ Clases internas

Java 2 (J2SE 1.2)

Apareció en Diciembre de 1998 al aparecer el JDK 1.2. Incorporó notables mejoras como por ejemplo:

- ◆ JFC. *Java Foundation classes*. El conjunto de clases de todo para crear programas más atractivos de todo tipo. Dentro de este conjunto están:
 - ◆ El paquete Swing. Mejorando notablemente al anterior paquete AWT. Se trata de todo un conjunto de clases que permiten desarrollar fácilmente entornos de ventanas. Es parte de JFC.
 - ◆ Enterprise Java beans. Para la creación de componentes para aplicaciones distribuidas del lado del servidor
 - ◆ Java Media. Conjunto de paquetes para crear paquetes multimedia:
 - Java 2D. Paquete (parte de JFC) que permite crear gráficos de alta calidad en los programas de Java.
 - Java 3D. Paquete (parte de JFC) que permite crear gráficos tridimensionales.
 - Java Media Framework. Paquete marco para crear elementos multimedia
 - Java Speech. Para reconocimiento de voz.
 - Java Sound. Audio de alta calidad
 - Java TV. Televisión interactiva
- ◆ JNDI. *Java Naming and Directory Interface*. Servicio general de búsqueda de recursos. Integra los servicios de búsqueda más populares (como LDAP por ejemplo).

(3.3.3) instalación del SDK

Es un requisito previo antes de programar en lenguaje Java.

en Windows

Desde la página de descarga <http://java.sun.com/javase/downloads> se elige la versión deseada del entorno de desarrollo. Una vez descargado el programa de instalación del SDK, basta con ejecutarle.

Hay que prestar atención al directorio en el que se ha instalado el SDK. La razón es que debemos modificar tres variables del sistema (variables que utiliza Windows para la configuración correcta de comandos). Son:

- ◆ **PATH**. Variable que contiene rutas por defecto a los programas que indiquemos. La razón es que por ejemplo el comando **java** debe de estar disponible estemos en la carpeta que estemos. Dicho comando (junto con el resto de comandos del SDK) está en la carpeta **bin** dentro de la carpeta en la que hemos instalado el SDK.

Ejemplo de contenido de la variable **path**:

```
PATH=C:\WINNT\SYSTEM32;C:\WINNT;C:\WINNT\SYSTEM32\WBE  
M;C:\Archivos de programa\Microsoft Visual  
Studio\Common\Tools\WinNT;C:\Archivos de programa\Microsoft Visual  
Studio\Common\MSDev98\Bin;C:\Archivos de programa\Microsoft  
Visual Studio\Common\Tools;C:\Archivos de programa\Microsoft Visual  
Studio\VC98\bin;C:\Archivos de programa\java\jsdk1.6.2\bin
```

- ◆ **JAVA_HOME**. Variable utilizada por la mayoría de aplicaciones basadas en Java que contiene la ruta a la carpeta en la que se instaló el SDK.
- ◆ **CLASSPATH**. Se explicara con detalle más adelante en este mismo manual. Es una variable similar al PATH que sirve para indicar rutas a las carpetas en las que se almacenarán aplicaciones Java. Dicho de una manera más técnica: contiene las rutas de todos los **filesystems** de Java.

La forma de configurar estas variables en Windows (si al menos tenemos versión de Windows superior o igual al 2000):

- (1) Señalar al ícono **Mi PC** (o **Equipo** en Windows Vista, Windows 2008 o Windows 7) y elegir Propiedades. Después elegir **Propiedades Avanzadas** y finalmente pulsar en el botón **Variables de entorno**.

en Linux

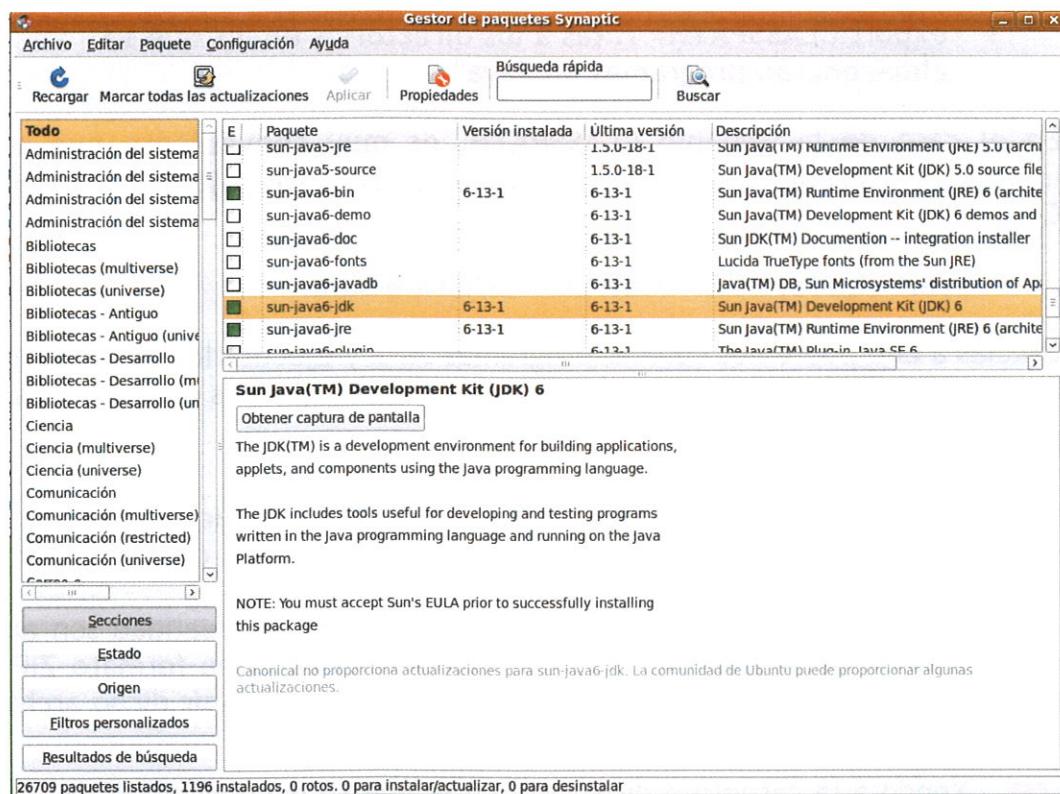


Ilustración 3-5, El gestor de paquetes Synaptic en el sistema Linux Ubuntu, mostrando información sobre los paquetes Java

Casi todas las versiones actuales de Linux incluyen el entorno de ejecución (**JRE**) de Java y la mayoría el entorno de desarrollo (**SDK**). Esto significa que seguramente no haya que instalar nada. Para conocer la versión instalada habría que ejecutar el comando **java -version**.

Si deseamos instalar Java o actualizarlo, hay que instalar el último paquete, una opción es utilizar el gestor **Synaptic** para descargar la última versión del paquete de desarrollo en Java (ver Ilustración 3-5).

También podemos instalar desde la línea de comandos, sería algo como

```
sudo apt-get install sun-java6-jdk
```

Finalmente siempre podemos acudir a la página de descargas de Sun, <http://java.sun.com/javase/downloads> y descargar la versión deseada. El archivo hay que descomprimirlo (si es un paquete **rpm**) o ejecutarlo (si es simplemente un archivo **bin**). Se obtendrá un directorio con todo el SDK de Java. Ahora bastará con colocarlo en el directorio adecuado.

En todo caso, sea cual sea la forma de instalar el SDK, habrá que modificar tres variables de entorno. Para lo cual lo normal es modificar el fichero **/etc/bash.bashrc** y al final añadir las siguientes entradas:

- ◆ **export JAVA_HOME="ruta en la que está el SDK de Java"**
La ruta podría ser algo como **/usr/lib/jvm/java6sun**

entornos de edición, algunos incluyen el compilador y otras utilizan el propio JDK de Sun.

Algunas ventajas que ofrecen son:

- ◆ Facilidades para escribir código: coloreado de las palabras clave, autocorrección al escribir, abreviaturas,...
- ◆ Facilidades de depuración, para probar el programa
- ◆ Facilidad de configuración del sistema. Elección concreta del directorio del SDK, manipulación de la variable CLASSPATH, etc.
- ◆ Facilidades para organizar los archivos de código.
- ◆ Facilidad para exportar e importar proyectos

Los IDE más utilizados en la actualidad son:

- ◆ **Eclipse.** Es un entorno completo de código abierto que posee numerosas extensiones (incluido un módulo para JEE y otro para programar en C++) y posibilidades. Hoy en día es el más utilizado y recomendable. Posee una comunidad muy dinámica y desde luego es uno de los más potentes. Disponible en www.eclipse.org.
- ◆ **NetBeans.** Entorno gratuito de código abierto para la generación de código en diversos lenguajes (especialmente pensado para Java). Contiene prácticamente todo lo que se suele pedir a un IDE, editor avanzado de código, depurador, diversos lenguajes, extensiones de todo tipo (CORBA, Servlets,...). Incluye además un servidor de aplicaciones **Tomcat** para probar aplicaciones de servidor. Se descarga en www.netbeans.org.
- ◆ **JBuilder.** Entorno completo creado por la empresa Borland (famosa por su lenguaje **Delphi**) para la creación de todo tipo de aplicaciones Java, incluidas aplicaciones para móviles. Es un entorno muy potente. *(www.eubarcadero.com/es/products/jbuilder) (PAGO)*
- ◆ **JDeveloper.** De Oracle. Entorno completo para la construcción de aplicaciones Java y XML. Uno de los más potentes y completos (ideal para programadores de Oracle). *(Gratis)* *(www.oracle.com/technetwork)*
- ◆ **IntelliJ Idea.** Entorno comercial de programación bastante fácil de utilizar pero a la vez con características similares al resto. Es menos pesado que los anteriores y muy bueno con el código. *(www.jetbrains.com)*
- ◆ **Microsoft Visual J++ y Visual J#.** Ofrece un compilador recomendable para los conocedores de los entornos de desarrollo de **Microsoft** (como **Visual Basic** por ejemplo), de hecho son lenguajes integrados en el **Visual Studio** y **Visual Studio.Net** respectivamente. Pero el Java de Microsoft no es estándar. El único uso que tiene es para aquellos programadores de Java que necesitan programar para la plataforma .Net de Microsoft y que no desean aprender los lenguajes propios de esta plataforma como **C#** o **Visual Basic** por ejemplo.

◆ **Geany** : www.geany.org
(23)

◆ **Jcreator** : www.jcreator.org/download.htm