CURSO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Entornos de desarrollo

INICIATIVA Y COORDINACIÓN
IFP Innovación en Formación Profesional

Supervisión editorial y metodológica:
Departamento de Producto de Planeta Formación

Supervisión técnica y pedagógica:
Departamento de Enseñanza de **IFP** Innovación en Formación Profesional

Módulo: Entornos de desarrollo / Desarrollo de aplicaciones web

Barcelona (España), 2017

INTRODUCCIÓN AL MÓDULO

Un **entorno de desarrollo** es el conjunto de procedimientos y herramientas utilizadas para la creación o mantenimiento de programas informáticos (software). Si bien un entorno de desarrollo debe abarcar la entrada de código de programa a través de un editor de código, las pruebas a través de un compilador y un depurador y su empaquetado final para la entrega al usuario utilizando un constructor de interfaz gráfica (GUI), hay otros aspectos que debe tener en cuenta para ser un nexo útil entre la fase de análisis y la fase de instalación que un desarrollo de software debe contemplar. En ocasiones, el concepto puede también aplicarse a un entorno y elementos físicos que permiten alcanzar los objetivos planteados por un requerimiento de cliente.

Se etiqueta como IDE (*Integrated Development Environment* o Entorno de Desarrollo Integrado) al entorno de programación que contiene un conjunto de procesos, instrumentos y normas que trabajan de forma coordinada para facilitar al programador el control de las diferentes etapas del desarrollo en las que está directamente implicado: entrada de código de programa y pruebas y preparación (empaquetado) del producto final para su inmediata utilización. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien pueden utilizarse para varios.

En algunos lenguajes de programación, un IDE puede funcionar como un sistema en tiempo de ejecución donde se permite utilizar el lenguaje de programación de forma interactiva sin necesidad de trabajo orientado a archivos de texto.

UNIDAD FORMATIVA 1

- Desarrollo de software
- Instalación y uso de entornos de desarrollo

1. Desarrollo Desoftware

Desarrollo de software es el proceso que tiene lugar desde que un usuario plantea una mejora de su actual sistema informático hasta que dicha solución (nuevo software creado o modificado de uno ya existente) le es entregado para su inmediata utilización. Este proceso incluye: actividades, como la investigación del problema; la creación o modificación de un nuevo software que lo solucione; la presentación de un prototipo para la aceptación del usuario, mediante pruebas, ajustes últimos; e implementación en el sistema real de explotación y diseño de los métodos de mantenimiento. Por lo que respecta a la gestión de las tareas mencionadas, también suelen incluirse todas aquellas actividades que, aunque no están directamente relacionadas con el departamento técnico, son igualmente necesarias para que un proyecto pueda llevarse a cabo de forma satisfactoria para todas las partes implicadas. Estos aspectos, que están más relacionados con el negocio del cliente, son las fechas de entrega, la valoración económica, el cálculo de recursos necesarios y acciones de ajuste de personal, y el presupuesto. El desarrollo de software, por lo tanto, va más allá de una visión puramente tecnológica y contempla aspectos tan humanos como la capacidad de gasto, la disponibilidad de tiempo, el carácter del personal involucrado y la naturaleza de la empresa cliente (familiar, multinacional, local, etcétera).

1.1 Concepto de programa informático

Un **programa informático** es un conjunto de instrucciones que indican y permiten a un ordenador realizar una tarea determinada. La mayoría de programas informáticos intentan simular secciones de actividades desarrolladas en nuestra realidad. Si somos personas observadoras nos daremos cuenta de que estamos rodeados de numerosos grupos de instrucciones con un objetivo común. A este conjunto de instrucciones con normas claras, finitas y ordenadas en el tiempo se le conoce como **algoritmo.**

Las actividades cotidianas pueden ser también definidas mediante algoritmos. Así, acciones sencillas como la de preparar una taza de té puede desglosarse en más pasos de los que a priori podría pensarse, desde comprar las bolsitas de té hasta calentar el agua, verterla en la tetera, echar el azúcar en la taza, etc.

Los algoritmos podemos complicarlos tanto como deseemos, incluso determinados elementos de un algoritmo pueden ser en sí mismos nuevos algoritmos. De este modo, la acción de comprar té en bolsitas podría, a su vez, ser un nuevo algoritmo que definiera las acciones necesarias para acercarnos a un supermercado a comprar el té. Será decisión nuestra alcanzar el nivel de especificidad necesario para nuestro proyecto.

El **lenguaje máquina** es el propio de los ordenadores, por lo que con él podríamos dar las instrucciones a un ordenador. Sin embargo, debido a su complejidad, se han desarrollado lenguajes más cercanos a los hábitos de comunicación humana (**lenguajes de alto nivel** o llamados de **cuarta generación**) que permiten la entrada de código mediante instrucciones comprensibles por nosotros (Figura 1.1). Una serie de procesos posteriores traducirán nuestras instrucciones al lenguaje máquina que el ordenador es capaz de procesar, aunque también es posible la traducción hacia lenguajes intermedios.

Figura 1.1
El desarrollo de software es una equilibrada combinación de tecnología y manifestaciones humanas.

1.2 Código fuente, código objeto y código ejecutable; máquinas virtuales

Desde que un programador entiende cómo debe resolver un problema planteado hasta que se entrega el producto que lo soluciona, pasa por varias fases que van coincidiendo con diferentes "estados" de los elementos que el programador ha creado. El desarrollador introduce código en el ordenador mediante un lenguaje que es comprensible para aquellas personas con conocimientos de la tecnología que se esté utilizando (código fuente), que luego se traducirá a un lenguaje propio del ordenador (código objeto). Como si de un director de orquesta se tratara, el enlazador dará sentido a los objetos creados y, por lo tanto, el código objeto pasará a ser código ejecutable. Por último, las máquinas virtuales permitirán que un mismo programa pueda ser ejecutado, sin necesidad de cambios, en diferentes sistemas operativos.

Recuerda

El lenguaje de alto nivel se diseñó para facilitar la tarea de codificación.

1.2.1 Código fuente

El código fuente es un conjunto de instrucciones creadas por un programador mediante un editor de texto o herramienta de edición y desarrollo de software. Suelen ser **sentencias**: sustantivos, verbos y preposiciones en inglés con sus correspondientes parámetros que para el ordenador serán órdenes concretas en un lenguaje de alto nivel que no puede ser entendido directamente por él (Figura 1.2).

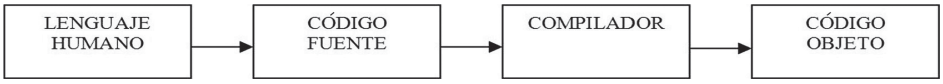

Figura 1.2
Flujo del código de programación.

1.2.2 Código objeto

El código objeto es el resultado de traducir el código fuente, que anteriormente ha introducido un programador, a un lenguaje que el ordenador ya puede entender. Código fuente y código objeto son la misma cosa pero con diferente lenguaje.

El primero es comprensible para las personas con conocimientos técnicos. El segundo lo es para un ordenador. Al programa traductor de código fuente a código objeto se le llama **compilador**. El hecho de ser un código objeto, y, por tanto, comprensible por el ordenador, no significa que pueda ser ejecutado directamente.

1.2.3 Código ejecutable

Se llama código ejecutable a uno o varios códigos objetos enlazados que pueden ser ejecutados por un ordenador. El código o códigos objetos que conforman un código ejecutable, a pesar de estar creados con lenguaje entendible por el ordenador, no pueden ser ejecutados por separado, siendo el enlace mencionado entre ellos el que permitirá al ordenador su uso. La creación del código ejecutable (Figura 1.3) es también una función propia del compilador. Código ejecutable puede referirse también a un lenguaje interpretado, es decir, un lenguaje que necesita de un "intérprete" para poder ser entendido por el ordenador.

Cada vez que se ejecuta el lenguaje interpretado necesita que el intérprete vuelva a "traducir" a un lenguaje entendible por la máquina. Debemos diferenciarlo del lenguaje compilado ya que éste último realiza la traducción al lenguaje de la máquina una sola vez (proceso de compilación).

1.2.4 Máquinas virtuales

Una **máquina virtual** es un software que simula un entorno de ordenador concreto en el que podemos instalar un sistema operativo, o programas, y ejecutarlos.

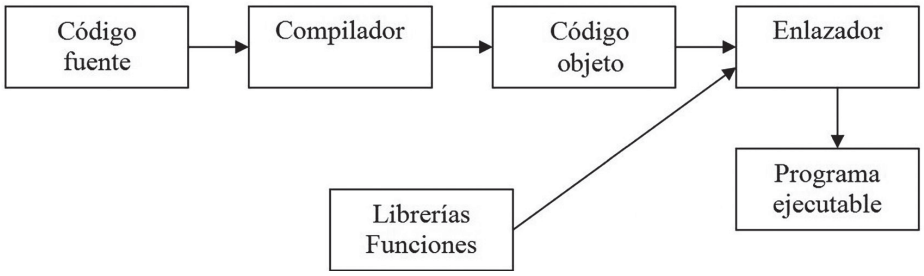

Figura 1.3
Código ejecutable.

Sería como un ordenador simulado dentro de otro ordenador. El primer uso de una máquina virtual puede ser probar un software de un sistema operativo diferente al que tenemos instalado en nuestro ordenador sin que peligre la configuración de nuestro actual sistema operativo (Figura 1.4).

El lenguaje Java utiliza una máquina virtual para ejecutarse. Sin embargo, aunque el concepto de máquina virtual es más amplio, en este módulo deberemos entender la máquina virtual como JVM (Máquina Virtual de Java).

Puesto que el lenguaje Java es a la vez compilado e interpretado, nuestro programa se compila a un conjunto de instrucciones (*bytecodes*) que se guardan en un fichero de extensión .class. Estos ficheros son entonces interpretados independientemente del sistema operativo que se tenga (siempre y cuando la Máquina Virtual de Java esté instalada).

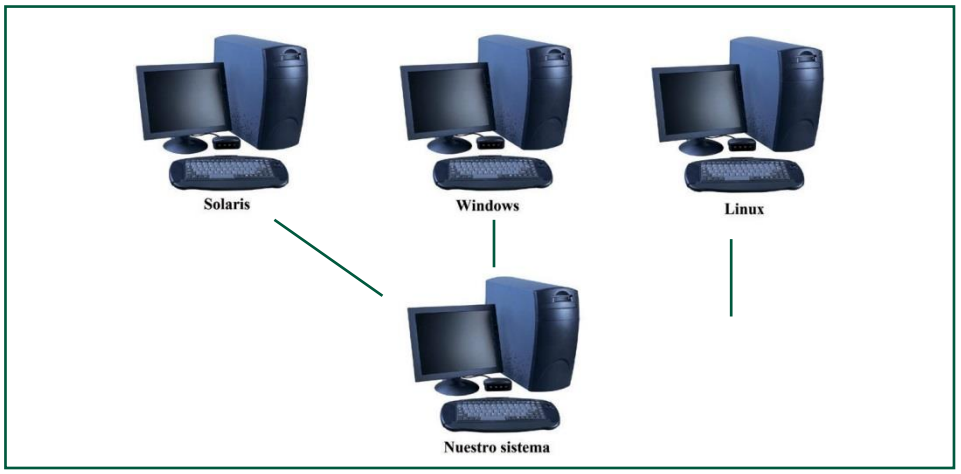

Figura 1.4
Nuestro sistema operativo puede contener otros sistemas virtualizados.

Al generar los compiladores de Java un lenguaje intermedio, que puede ser entendido por una máquina virtual de Java, podemos desarrollar software olvidándonos del tipo de ordenador en que vaya a ejecutarse, ya que el único requisito será que tenga instalada dicha máquina virtual (JVM).

1.3 Tipos de lenguajes de programación

Según los aspectos más o menos específicos de los lenguajes, podemos clasificarlos en tres grandes grupos en función de su forma de ejecución y aparición histórica.

1.3.1 Lenguajes de procedimiento

Un programa de procedimiento (también conocido como *procedural*) contiene una lista de acciones que deben ser completadas para conseguir el objetivo por el que el programa fue creado. Éste se caracteriza por: un punto de comienzo, un proceso y un final. Los procedimientos pueden ser funciones, rutinas o métodos. Como ejemplos de lenguajes de procedimiento podemos mencionar BASIC y FORTRAN (Figura 1.5).

```
10 REM Morse code - MORSE.BAS
20 CLEAR :CLS :KEY OFF
30 PRINT "Type a message and see it appear as Morse Code." :PRINT
40 DIM CODE$(122)
50 RESTORE
60 FOR A = 32 TO 122
70   READ CODE$(A)
80   PRINT CHR$(A);"  ";
90 NEXT A
100 PRINT :PRINT
110 INPUT "Use any of the above - What is your message "; MESSAGE$
120 PRINT
130 L = LEN(MESSAGE$)
140 FOR B = 1 TO L
150   PRINT CODE$(ASC(MID$(MESSAGE$,B,1)));" ";
160 NEXT B
170 PRINT :PRINT
180 END
```

Figura 1.5
Ejemplo de código de procedimiento línea a línea.

1.3.2 Lenguajes de programación estructurada

Se trata de una variante del anterior que nace como respuesta a la dificultad de modificar los programas realizados con lenguajes de procedimiento, sobre todo cuando éstos eran más grandes y complejos. Con este tipo de lenguajes se pretende romper la estructura monolítica de los programas en módulos más pequeños y manejables así como diferenciar variables globales y locales visibles desde un módulo en concreto. Dos ejemplos de lenguajes de programación estructurada son C y Pascal.

1.3.3 Lenguajes de Programación Orientada a Objetos (OOP)

En este tipo de lenguajes, el programador diseña la estructura de datos y el tipo de operaciones que acepta dicha estructura. Al conjunto de estructura de dato y operación se le llama **objeto**. El programa ya no es una lista de acciones de orden fijo si no un conjunto de objetos (datos y posibles operaciones que puede realizar) que cooperan entre ellos para conseguir un objetivo. Veamos a continuación los nuevos conceptos que se aplican a los objetos mencionados:

- **Clase.** Modelo desde el que los objetos son creados (conjunto de variables y métodos). El objeto creado puede posteriormente ser modificado mediante nuevas estructuras de datos y métodos (Figura 1.6).

- **Encapsulación.** Los datos y métodos de una estructura se protegen de aquellos accesos que no sigan las normas especificadas para el objeto. Asimismo el programador decide qué información del objeto puede ser compartida por otros objetos.

- **Polimorfismo.** Aquellos objetos de diferente tipo que pueden recibir el mismo mensaje pero responder de diferente modo.

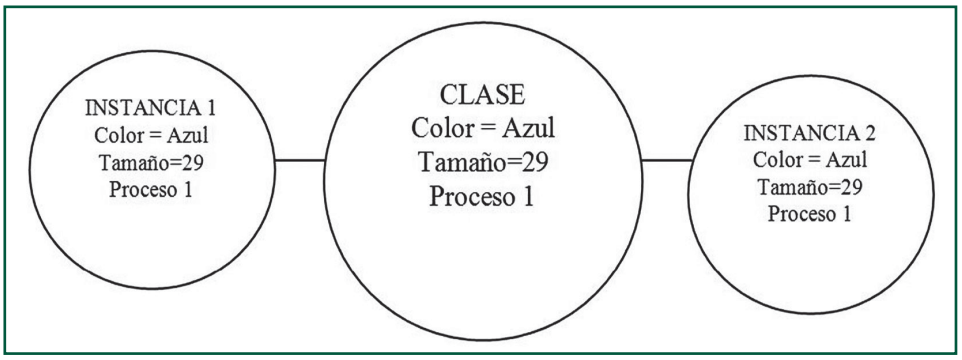

Figura 1.6
Derivación de clase. A partir de un modelo, los nuevos objetos se crean con las mismas características, que luego pueden variarse.

1.4 Características de los lenguajes más difundidos

Los lenguajes de programación más comunes son Java, C, C++, PHP, C# y Visual Basic. Veamos a continuación las características principales de cada uno de ellos:

- **Java**
  - Lenguaje orientado a objetos.
  - Más sencillo de manejar que otros lenguajes orientados a objetos.
  - Manejo de computación distribuída (varios ordenadores trabajando juntos en una red).
  - Portabilidad (la Máquina Virtual de Java permite que un mismo programa pueda ser utilizado en diferentes ordenadores con distintos sistemas operativos).
  - Buenas perspectivas profesionales.

Para saber más

A diferencia de la Programación de Procedimiento, que ejecuta el código siempre en el mismo orden, la Programación Orientada a Objetos puede no ejecutar el código en el orden que lo hizo la última vez.

- **C**
  - Lenguaje estructurado.
  - Reducido número de comandos.
  - Existencia de compiladores para casi cualquier ordenador.
  - Larga presencia en el tiempo.

- **C++**
  - Lenguaje orientado a objetos.
  - Concisión: utiliza muchos caracteres especiales en vez de comandos, por lo que un programador codifica a mucha velocidad.
  - Portabilidad: para compilar en casi todos los ordenadores y sistemas operativos.
  - Programación modular: varias fuentes son compiladas independientemente para luego enlazarlas.
  - Compatibilidad: código escrito en C puede ser incluido con facilidad en C++.
  - Eficiencia en la ejecución de programas.

```
// File name: HelloWorld.cpp
// Purpose:   A simple C++ program which prints "Hello World!" on the screen

#include <iostream>  // need this header file to support the C++ I/O system
using namespace std; // telling the compiler to use namespace "std",
                     // where the entire C++ library is declared.

int main()
{
        // Print out a sentence on the screen.
        // "<<" causes the expression on its right to
        // be directed to the device on its left.
        // "cout" is the standard output device -- the screen.
        cout << "Hello World!" <<  endl;
        return 0; // returns 0,which indicate the successful
                  // termination of the "main" function

}
```

Figura 1.7
Ejemplo de código con presentación de las palabras "Hello World" en lenguaje C++.

- **PHP**
  - Lenguaje de *script* (permite el control de una o más aplicaciones).
  - Utilizado en el desarrollo web (Ámbito del Servidor).
  - Práctico: su evolución se orientó hacia la utilidad de cara al usuario.
  - Muy bajo coste económico.
  - Páginas web dinámicas (uso embebido en código HTML).

- **C#**
  - Lenguaje orientado a objetos.
  - Incorpora características de otros lenguajes (Java, Visual Basic).
  - Forma parte de la plataforma .NET de Microsoft.
  - Su sintaxis deriva de C y C++.

- **Visual Basic**
  - Lenguaje orientado a objetos.
  - Fácil de utilizar.
  - Deriva del BASIC de procedimiento.
  - Forma parte de la plataforma .NET de Microsoft.

Los entornos visuales y el lenguaje orientado a objetos dominan el mercado. Sin embargo, existen nichos de mercado de costosa actualización en los que antiguos lenguajes no visuales de procedimiento, o estructurados, se resisten a desaparecer. Ese es el caso del lenguaje COBOL en el sector de la banca o del lenguaje RPG en el sector industrial (Figura 1.8).

Para saber más

Un Entorno de Desarrollo debe encajar dentro de una metodología de trabajo. PRINCE2 (Projects IN Controlled Environments o Proyectos en Entornos Controlados) es una buena metodología de gestión de proyectos con certificación de calidad.

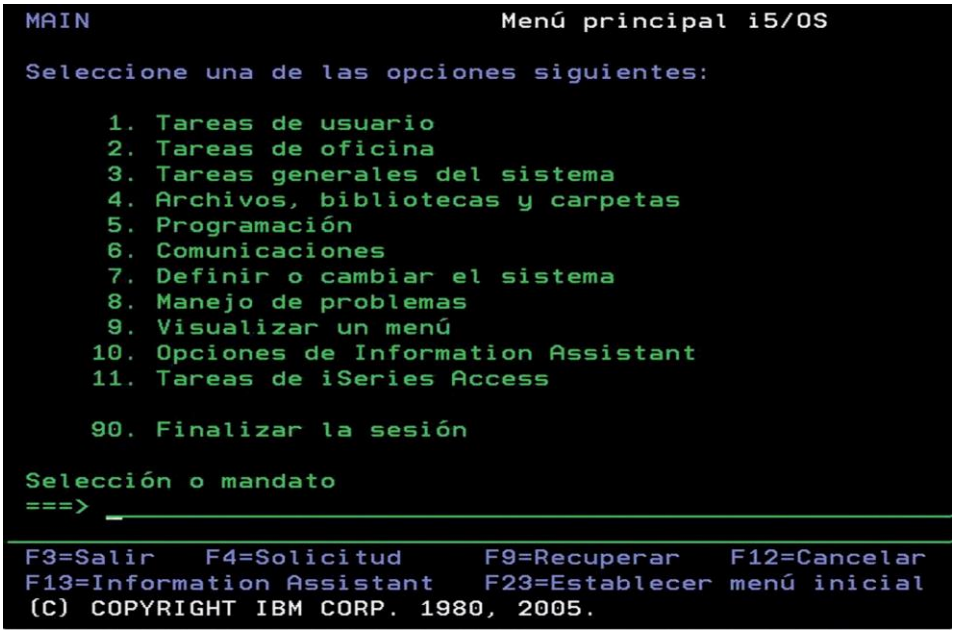

Figura 1.8
Entorno no visual de un sistema AS400 de IBM.

1.5 Fases del desarrollo de una aplicación

El desarrollo de una aplicación informática debe estar controlada por un buen gestor de proyectos que conozca perfectamente las etapas por las que debe pasar para poder llegar a la entrega del producto en el plazo previsto, con la calidad pactada y con el coste razonablemente calculado. Veamos a continuación cuáles son estas fases:

- **Planificación del proyecto.** Estudio de viabilidad. Se trabaja con una visión global. Dependiendo del ámbito y complejidad del proyecto se debate con la dirección de la empresa y se perfilan los objetivos a alcanzar. Es preciso crear un documento explicativo de las primeras valoraciones del proyecto.

- **Documento del presupuesto.** Después de la planificación, el responsable del proyecto debe valorarlo económicamente. Si bien existen fórmulas matemáticas de cálculo de un presupuesto, suele ser una buena práctica añadir un 10 % de tiempo y dinero al importe ya calculado para cubrir imprevistos.

- **Firma del documento de iniciación del proyecto.** Una vez el presupuesto es aceptado, se precisa la firma de los directivos de la empresa para establecer las fechas de entrega para afianzar el compromiso (Figura 1.9).

Figura 1.9
Un acuerdo claro entre las partes implicadas en el proyecto puede evitar futuros conflictos.

- **Análisis del sistema y recogida de requerimientos.** Se pretende entender la situación actual mediante investigaciones técnicas y de negocio, además de reuniones con los usuarios. Se identifican todas las funciones y operaciones existentes, las que se van a modificar y las que se van a crear.

- **Diseño del sistema.** El diseño de sistemas consiste en definir la arquitectura de hardware y software, componentes, módulos y datos de un sistema informático para satisfacer ciertos requerimientos. Es la etapa posterior al análisis de sistemas. El diseño de sistemas tiene un rol más respetado y crucial en la industria de procesamiento de datos. La importancia del software multiplataforma ha incrementado la ingeniería de software a costa de los diseños de sistemas (Figura 1.10).

Figura 1.10
El análisis del sistema busca la relación entre departamentos y funciones de usuarios evitando los elementos aislados de funciones.

- **Codificación.** Mediante un entorno de desarrollo, el programador realizará las modificaciones y la creación de nuevos objetos informáticos que ayuden a conseguir los objetivos del proyecto aplicando las especificaciones técnicas pertinentes.

- **Pruebas técnicas.** El programador realizará las pruebas necesarias para demostrar que sus objetos realizan correctamente las tareas demandadas. Las pruebas técnicas deben ir acompañadas de un documento en el que se detalle cómo se realizó la prueba y los datos involucrados.

- **Pruebas de usuario.** El usuario debe realizar las pruebas que crea conveniente con los nuevos programas en un entorno con datos ficticios y protegido de la explotación en real para dar el visto bueno al desarrollo. Estas pruebas suelen ir acompañadas de un documento firmado por el usuario, lo que atestigua por tanto de que se han hecho los test necesarios.

- **Documentación.** El responsable de proyecto reúne toda la documentación generada hasta el momento y añade la información que falte. Esta fase adquiere mayor importancia si la empresa está bajo el control de normas de calidad. Si las pruebas de usuario están firmadas y han sido satisfactorias, se creará el documento de cambio del sistema en el que se enumeran los pasos a seguir para una buena implementación, así como la vuelta atrás (*roll back*) de la instalación en caso de fiasco (mal funcionamiento de lo instalado) (Figura 1.11).

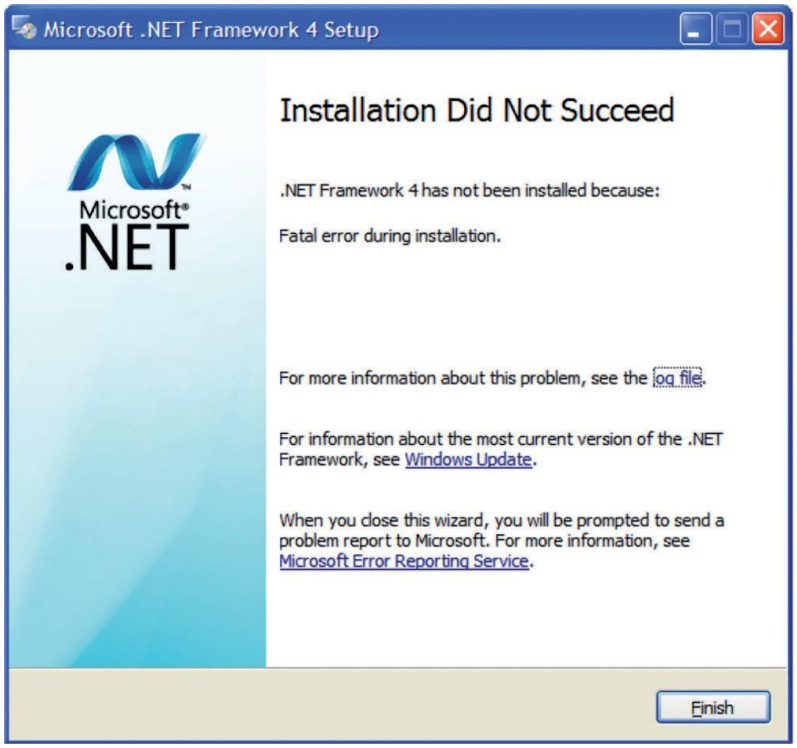

Figura 1.11
Siempre debe haber un plan de retroceso de una instalación si algo va mal.

- **Implementación en entorno de explotación (Real)**. Los objetos del proyecto pasan a explotación. Se abre un período de ventana de monitorización de la instalación y de posibles ajustes o vuelta atrás por mal funcionamiento.

- **Plan de mantenimiento.** Se agrupa toda la documentación técnica necesaria para realizar un buen mantenimiento de los objetos. En el futuro habrá cambios, correcciones, añadidos, migraciones hacia otras plataformas informáticas, etc. A la hora de confeccionar los documentos técnicos, es una buena práctica pensar en todo momento que aquellos que los leerán en un futuro serán técnicos que no han participado en el proyecto (Figura 1.12), por lo que es preciso tener en cuenta la claridad de la información.

Figura 1.12
Disponer de una buena documentación es básico para futuros mantenimientos.

1.6 Proceso de obtención de código ejecutable a partir del código fuente

Los elementos y fases implicadas en la obtención de un código ejecutable son los siguientes:

- **Fichero fuente.** La creación de uno o más ficheros fuente en el que son almacenadas las instrucciones en un lenguaje de alto nivel y que puede realizarse con un sencillo procesador de textos.

- **Preprocesamiento.** Se hace un rastreo del código fuente para detectar todos los posibles errores de entrada de código (paréntesis sin cerrar, asignaciones de variables inválidas, etcétera).

- **Compilador.** Este programa lee el fichero fuente y lo traduce en un tipo de código entendible por el ordenador en el que se ha realizado la compilación. Cada tipo de ordenador tiene su propio compilador. Esta traducción a un código entendible por el ordenador se le llama código objeto. Este nuevo código aún no es ejecutable.

- **Enlazador.** Después de que el compilador haya creado todos los objetos necesarios, el enlazador los procesará junto con las librerías de rutinas / funciones / datos, que el propio lenguaje aporta al proceso de desarrollo para la creación del ejecutable (Figura 1.13).

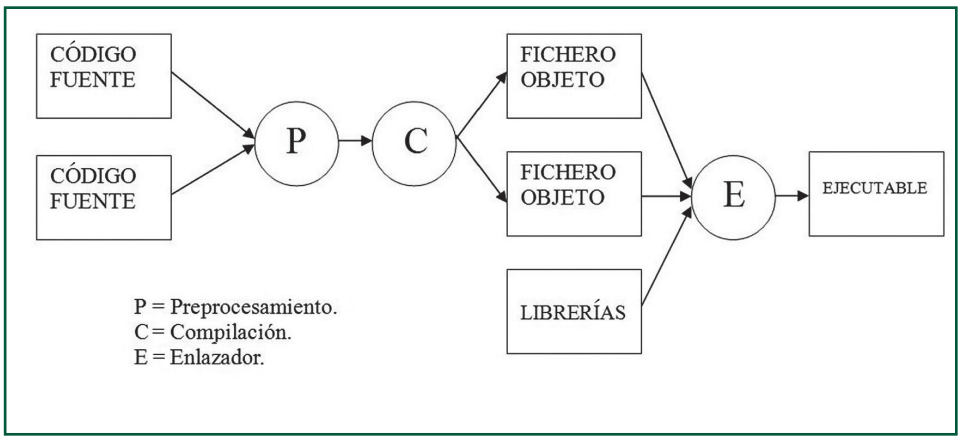

Figura 1.13
Ruta y transformaciones del código fuente hasta llegar a ser código ejecutable.

Resumen

Desarrollo de Software es el proceso que tiene lugar desde que un usuario plantea una mejora de su actual sistema informático hasta que dicha solución (nuevo software creado o modificado de uno ya existente) le es entregado para su inmediata utilización. La solución a los problemas planteados por un usuario se aborda mediante cambios en los métodos de trabajo y en la creación de herramientas informáticas (programas) que van evolucionando en función del momento de desarrollo en el que nos encontremos.

Existen diferentes lenguajes de programación que nos permiten la creación de códigos ejecutables (entendibles por un ordenador) a partir de una evolución desde un código fuente (entrado en el sistema por un programador). Esta evolución parte de un Fichero fuente que evoluciona a fichero objeto mediante la acción de un compilador y a un fichero ejecutable mediante la acción de un enlazador.

Dividimos los lenguajes de programación en tres grandes grupos:

- Lenguaje procedural (lista de acciones a realizar siempre en el mismo orden)

- Lenguaje estructurado (variante del anterior con mayor organización interna).

- Lenguaje orientado a objetos (Se crean estructuras de datos y operaciones que aceptan dichas estructuras)

Todos los lenguajes de programación deben formar parte de un proceso de desarrollo que parte del análisis del problema pasando por el diseño, codificación, pruebas y puesta en real.

Algunos de los lenguajes más difundidos son Java, C, C++, PHP, C# y Visual Basic.

Los principales hitos de un desarrollo de software son: la planificación del proyecto, el documento del presupuesto, la firma del documento de inicio del proyecto, el análisis del sistema, el diseño del sistema, la codificación, las pruebas técnicas, las pruebas de usuario, la publicación de la documentación, la implementación en el entorno de explotación y la creación de un plan de mantenimiento.

ejercicios deautocomprobación

Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):

1. El desarrollo de software es el proceso que tiene lugar desde que un usuario plantea una mejora de su actual sistema informático hasta que dicha solución (nuevo software creado o modificado de uno ya existente) le es entregado para su inmediata utilización.

2. Los algoritmos podemos complicarlos tanto como deseemos, incluso determinados elementos de un algoritmo pueden ser en sí mismos nuevos algoritmos.

3. Los lenguajes de cuarta generación son aquellos que no permiten la entrada de códigos mediante instrucciones comprensibles por nosotros.

4. Código fuente y código objeto son la misma cosa, pero con diferente lenguaje.

5. Al programa traductor de código fuente a código objeto se le llama complemento. El hecho de ser un código objeto, y, por tanto, comprensible por el ordenador, significa que puede ser ejecutado directamente

6. El código o códigos objetos que conforman un código ejecutable, a pesar de estar creados con lenguaje entendible por el ordenador, sí pueden ser ejecutados por separado.

7. Una máquina virtual es un hardware que simula un entorno de ordenador concreto en el que podemos instalar un sistema operativo, o programas, y ejecutarlos.

Completa las siguientes afirmaciones:

8. El_____es un conjunto de instrucciones creadas por un programador mediante un_____de texto o herramienta de edición y desarrollo de_____.

9. Mediante la_____, los datos y métodos de una estructura se protegen de aquellos_____que no siguen las normas especificadas para el_____.

10. Después de que el compilador haya creado todos los objetos necesarios, el _____ los procesará junto con las_____de rutinas / funciones / datos, que el propio lenguaje aporta al proceso de desarrollo para la creación del_____.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

2. INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

Escoger un **entorno de desarrollo** determinado dependerá en gran medida del tipo de solución y mercado que deseemos abordar. Sin embargo, casi siempre tendremos varias alternativas para seleccionar. Deberemos preguntarnos acerca de su uso intuitivo, su rápida curva de aprendizaje, así como sobre su perfecto ajuste en un ciclo de desarrollo más amplio, que empieza con el requerimiento del cliente y termina con el diseño de un plan de mantenimiento.

La mayoría de los entornos de desarrollo se utilizan de forma parecida y presentan un aspecto similar. Las acciones que realizan se hallan todas enmarcadas en el **ciclo de desarrollo**, que es idéntico para todo el mundo. Por lo que respecta al aprendizaje de entornos adicionales sólo deberemos prestar atención a las particularidades del lenguaje a tratar y cuál es su funcionamiento dentro del nuevo entorno seleccionado.

Por último, el esfuerzo de inversión, el coste por cada línea de código entrado, será un importante factor a tener en cuenta en la toma de decisiones, aunque sin olvidar, no obstante, la caprichosa influencia de las modas, que también existen en el ámbito de la informática.

2.1 Funciones de un entorno de desarrollo

Si abrimos las opciones desplegables que forman parte del menú principal de un entorno de desarrollo, podremos ver las principales funciones habilitadas para el desarrollador (Figura 2.1). Veamos cuáles son:

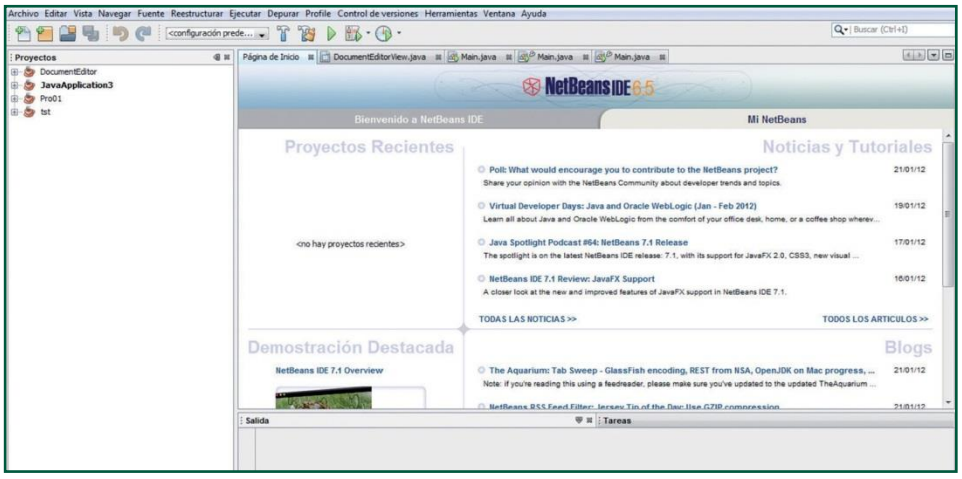

Figura 2.1
El menú desplegable del entorno de desarrollo nos ofrece las principales actividades que es posible realizar.

- Entrada de codificación (normalmente en un lenguaje de alto nivel) mediante un editor de texto.
- Derivación natural de la codificación a partir de un diseño previo.
- Ejecución del programa en pruebas, línea a línea o por secciones.
- Compilación y enlace de objetos mediante un sistema de empaquetado de la instalación.
- Asistencia de errores y coherencia interna del código entrado.
- Asistencia en la documentación y mantenimiento posteriores.

2.2 Herramientas y asistentes

Las **herramientas** y los **asistentes** ofrecen numerosas ventajas y facilitan el trabajo. Sin embargo, conocer exactamente el problema a resolver es mucho más importante que poseer un arsenal de ayudas adicionales (Figura 2.2).

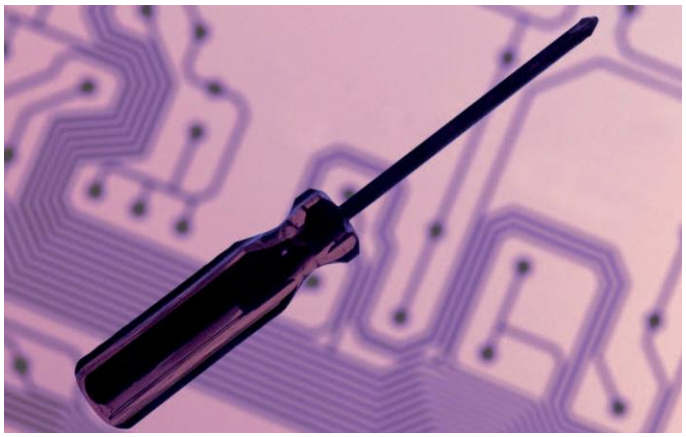

Figura 2.2
La resolución del problema siempre debe estar presente aunque tengamos muchas herramientas a nuestra disposición.

- **Asistente de aplicaciones.** Generador de programas tipo al que se le deberán aplicar las modificaciones necesarias para adaptarlo al requerimiento concreto del cliente.
- **Edición de formularios (pantalla, impresión).** El programador accede a múltiples formularios tipo.
- **Servidores web locales.** Simulan el funcionamiento de un servidor real de Internet y permite el desarrollo sin acceder a la red.

- **Entornos de varios lenguajes.** Permiten el desarrollo de una aplicación utilizando varios lenguajes.
- **Acceso a objetos externos.** Utilización de otros programas como si fueran propios (Microsoft Excel, Microsoft Word).
- **Motores de acceso a bases de datos.** Facilita la gestión de diferentes bases de datos sin cambios agresivos en el código de programación.
- **Depuración.** Asistencias extras y depuraciones en remoto.
- **Asistencia en diseño gráfico.** Presentación de datos e interacción. Soporte de creación de pantallas e informes.

Para saber más

En nuestro caso, hemos optado por trabajar con NeatBeans; no obstante, existen otras aplicaciones, tanto comerciales como libres.

2.3 Instalación de un entorno de desarrollo

Existen diversas maneras de instalar un entorno de desarrollo según el sistema operativo que vayamos a utilizar y de si emplearemos un instalador propio o el que nos ofrezca el producto. En nuestro caso, pensaremos siempre en la instalación de un entorno para trabajar con lenguaje Java en un sistema operativo Windows y que, si es posible, esté traducido a nuestro idioma (Figura 2.3).

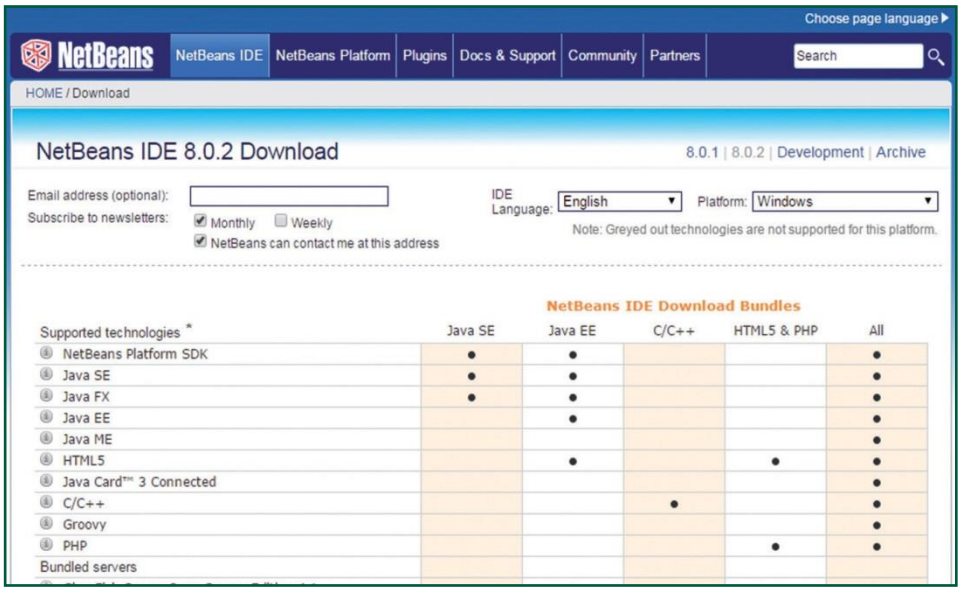

Figura 2.3
NetBeans es un ejemplo de entorno de desarrollo creado principalmente para el lenguaje de programación Java.

- **Instalación de JDK (*Java Development Kit* - Equipo de Desarrollo en Java).** Conjunto de herramientas para desarrolladores de Java. Puede instalarse a parte o junto con el entorno de desarrollo (NetBeans) utilizado en este ejemplo.

- **Instalación de NetBeans IDE (*Integrated Development Environment*) con JDK.** NetBeans es creado principalmente para el lenguaje de programación Java. Existe, además, un número importante de módulos para extenderlo.

En ocasiones, aunque el entorno de desarrollo esté en español, las pantallas que veremos hasta descargar el instalador están en inglés. En cualquier caso, siempre podemos recurrir a herramientas de traducción existentes en la red (Figura 2.4).

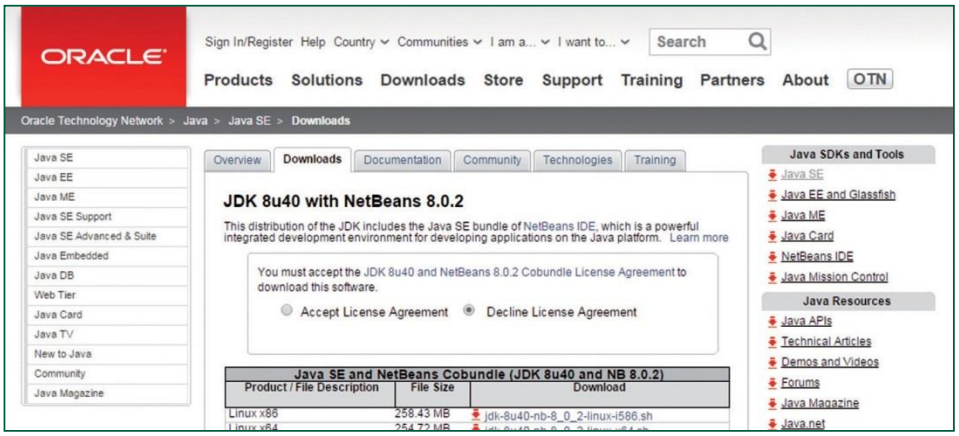

Figura 2.4
Imagen de ejemplo del entorno de desarrollo jdk-8u40-nb-8_0_2-windows-x64.exe.

El fichero del sistema Windows, jdk-8u40-nb-8_0_2-windows-x64.exe, deberemos guardarlo en un directorio local de nuestro ordenador, puesto que hemos escogido una instalación para sistema Windows. Veamos a continuación cuáles son los pasos previos que debemos realizar para instalar el programa.

- **Abrir archivo.** Para instalar un entorno de desarrollo en un sistema operativo Windows hacemos doble clic en el fichero .exe que hemos guardado en un directorio local. Puede aparecer un aviso que nos indique que la web desde donde hacemos la descarga no posee una **firma digital** (conjunto de datos asociados a otros que sirve para identificar al firmante; en este caso a la empresa de la cual vamos a realizar la descarga). Si confiamos en la web, deberemos proseguir clicando en "Ejecutar" (Figura 2.5).

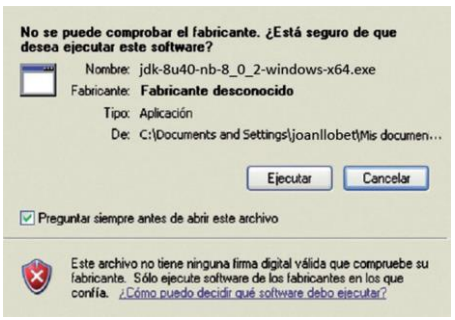

Figura 2.5
El software libre nos remite con mucha frecuencia a páginas en inglés por lo que deberemos estar atentos a las instrucciones de pantalla.

- **Inicio.** Una vez hemos aceptado progresar en la instalación de nuestro entorno de desarrollo, deberemos seguir las instrucciones que aparecen en pantalla, y que nos irán guiando a lo largo del proceso. Una primera pantalla nos invita a continuar y nos proporciona una información importante acerca de empresas vinculadas, así como de la versión exacta del producto que vamos a instalar. Estamos a tiempo de cancelar la instalación si éste no es el entorno o versión que necesitamos (Figura 2.6).

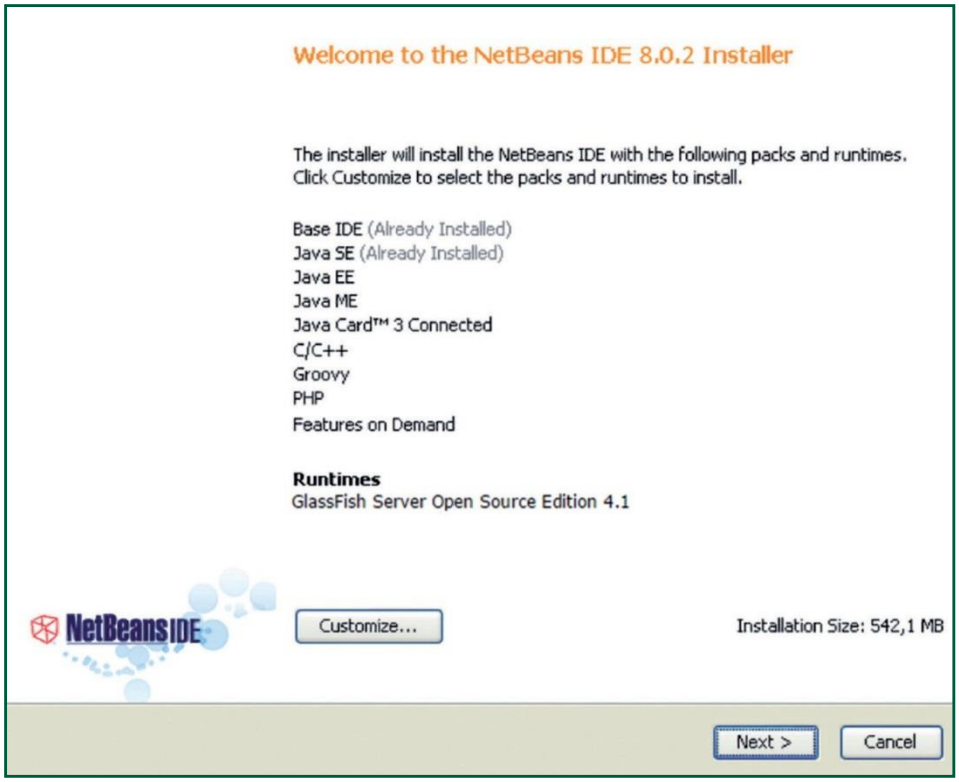

Figura 2.6
Información de Java previa a la instalación.

- **Desarrollo.** Si progresamos en nuestra instalación, se nos informará qué carpetas de nuestro ordenador van a ser utilizadas para guardar los objetos que el entorno necesita para funcionar. Normalmente, se recomienda dejar las sugeridas ("Archivos de Programa y subcarpetas"); así, cuando trabajemos en otro ordenador que no sea el nuestro, podremos intuir con facilidad la localización física del entorno. Las ubicaciones informadas por el proceso de instalación se refieren a JDK (Kit de Desarrollo de Java) y a los elementos propios del entorno (interacción con el usuario). El desarrollo de la instalación puede durar varios minutos según la potencia de nuestro ordenador. Una vez confirmadas las pantallas con la información básica de configuración (Figura 2.7), una barra de progreso nos indicará el porcentaje de la instalación que se ha realizado hasta el momento.

Para ampliar este tema, puedes ver el videotutorial *Instalación de un entorno Netbeans*, que encontrarás en el Campus online.

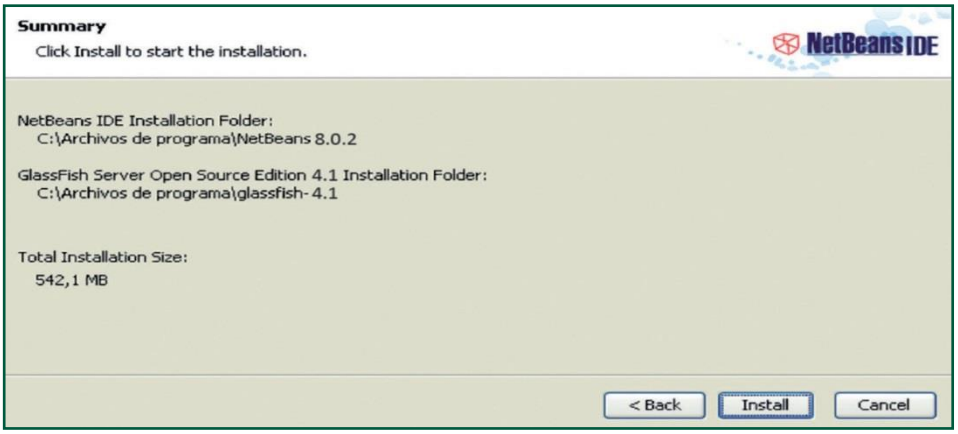

Figura 2.7
Información previa que el usuario deberáconfirmar.

- **Finalización.** Una vez concluida la instalación, aparecerá una pantalla informativa. En algunos casos, y con el único propósito de mejorar el producto, se nos puede invitar a participar en el envío de información técnica relativa al funcionamiento de nuestro software. No estamos obligados a participar en la recogida de datos. Si desmarcamos la casilla de "Contribute to the NetBeans..." y hacemos clic en botón "Finish", la instalación finalizará y se creará un icono de llamada al entorno en nuestro escritorio.

2.4 Instalación y desinstalación de módulos adicionales

NetBeans es un software libre, por lo que cualquier desarrollador puede crear complementos adicionales o existentes que pueden ser publicados y utilizados por otros programadores. La opción "Herramientas", ubicada en el "Menú Principal", muestra un desplegable con las diferentes tareas que permite realizar. Si escogemos la opción "Complementos", accederemos a una serie de pestañas que nos permitirán seleccionar qué grupo de complementos deseamos ver:

- Complementos disponibles.
- Complementos descargados (en los que podremos agregar nuevos).
- Complementos instalados (listos para su uso por nuestro entorno).

Para añadir un complemento, abriremos el menú "Herramientas", haremos clic en "Complementos" y seleccionaremos la pestaña "Complementos disponibles". A continuación, seleccionaremos el complemento que deseemos instalar y haremos clic en "Instalar". Para desinstalar un complemento la acción es similar, pero, en ese caso, cambiaremos la pestaña; es decir, haremos clic en "Instalado" y desmarcaremos el complemento que deseemos desinstalar. En el botón donde antes podía leerse "Instalar" se leerá "Desinstalar" (Figura 2.8).

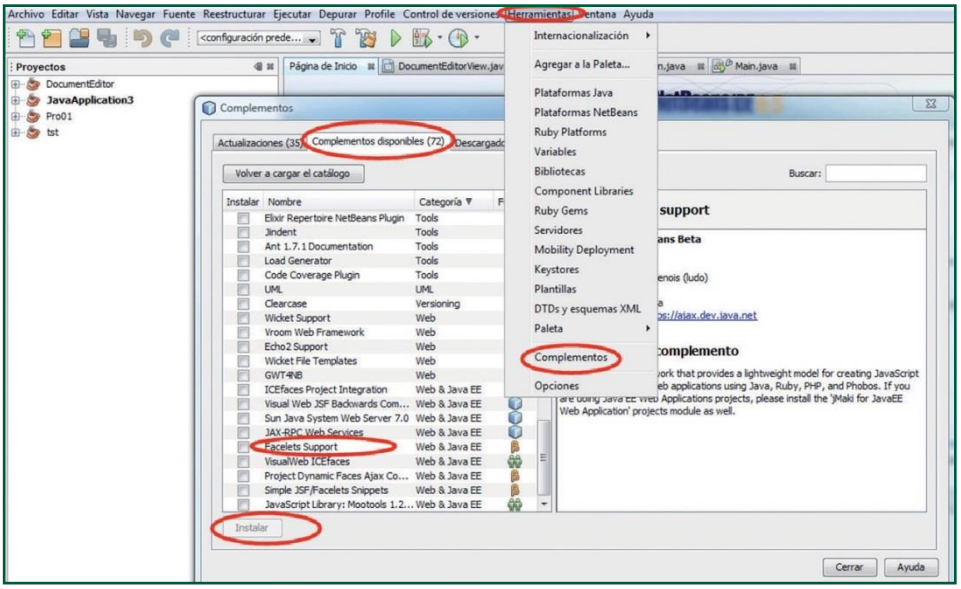

Figura 2.8
Opciones involucradas en la instalación y desinstalación de complementos.

2.5 Mecanismos de actualización

En el menú "Herramientas", opción "Complementos" se mantiene una actualización del catálogo de actualizaciones de los complementos instalados. La forma de proceder será idéntica a la instalación y desinstalación, con la diferencia de que la pestaña "Actualizaciones" es la que deberemos tener activa. A continuación, seleccionaremos todas las actualizaciones que deseemos arrancar y clicaremos el botón "Actualizar" (Figura 2.9).

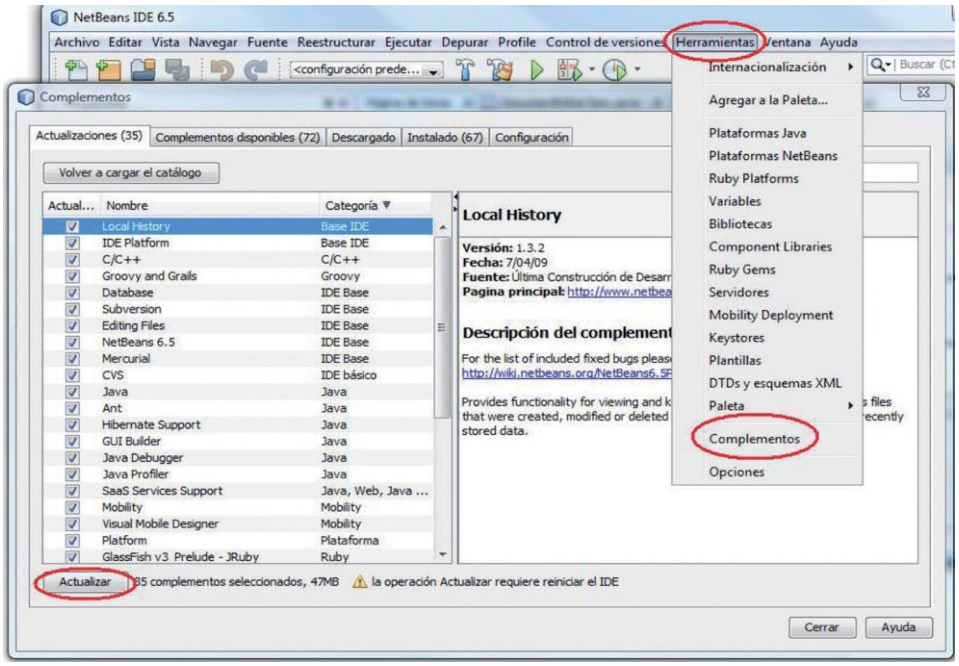

Figura 2.9
A diferencia de la figura anterior, en este caso la casilla de la parte inferior izquierda no sería "Instalar", sino "Actualizar".

Recuerda

El editor de NetBeans puede ser utilizado para otros lenguajes diferentes al Java.

2.6 Entornos de desarrollo libres y comerciales más usuales

A la hora de escoger un entorno de desarrollo, deberemos tener en cuenta aspectos como el esfuerzo de inversión o las garantías formales que un producto debe tener para un responsable de informática (asistencia técnica y gastos por el mantenimiento). En ocasiones, se paga por algunos productos debido a las políticas de empresa relacionadas con la evitación de riesgos, si bien existen soluciones mejores en el mercado de software libre. ¿Comercial o libre? En la figura 2.10 puede verse una relación de los entornos propios de cada grupo.

LIBRE	Sistema Operativo	Características
Aptana Studio	Windows/Mac/Linux	Soporte HTML5, CSS3, JavaScript, Ruby, Rails, PHP and Python
Qt Creator	Windows/Mac/Linux	Acceso a bibliotecas Qt multiplataforma
Code::Blocks	Windows/Mac/Linux	Orientado a C++
NetBeans	Windows / Linux / Mac / Solaris	Aplicaciones Web con Java, PHP, C/C++ y más.
Eclipse	Windows / Mac / Linux	IDE de propósito general
Pspad	Windows	Soporte a múltiples lenguajes
Microsoft Visual Studio Express	Windows	Soporte a múltiples lenguajes
COMERCIAL		
Microsoft Visual Studio	Windows	Soporte a C, C++ y lenguajes basados en .NET
Embarcadero Delphi (antes Borland Delphi y C++ Builder	Windows / Mac /iOS	Pascal y variante de C++
Wing IDE	Windows / Mac / Linux	Lenguaje Python
Komodo IDE	Windows / Mac / Linux	Python, PHP, Ruby, JavaScript, Perl and Web Dev

Figura 2.10
Entornos integrados libres y comerciales.

Un software libre, a diferencia del comercial o privado, proporciona libertad al usuario para copiar, ejecutar, modificar y distribuir el propio producto. No es sinónimo de gratis, aunque la mayoría del software de este tipo suele serlo.

2.7 Uso básico de un entorno de desarrollo

La pantalla principal del entorno de desarrollo puede ser distribuida en varias áreas de trabajo (Figura 2.11). La primera tarea que debemos realizar es la de poner nombre a nuestro trabajo. El entorno lo reconocerá como un "Proyecto". Para crear un proyecto, clicaremos menú "Archivo" del área 1 e iremos a la opción "Proyecto Nuevo".

Veamos a continuación cuáles son las principales áreas de trabajo:

- **Área 1.** Menú principal, desde donde se realizan las principales acciones del entorno (ficheros, depuración y control del proyecto).
- **Área 2. (Ventana de proyecto).** Muestra una visión de conjunto del proyecto (ficheros, librerías necesarias para el código entrado, etcétera).
- **Área 3 (Ventana de navegación).** Permite acceder de forma rápida a los elementos de las clases que tengamos seleccionadas.
- **Área 4 (Ventana de edición).** Editor de texto a través del cual entramos en el código del desarrollo de nuestro proyecto.
- **Área 5 (Ventana de salida / Tareas).** Muestra los errores de compilación e información seleccionada por el programador mediante palabras clave del código.

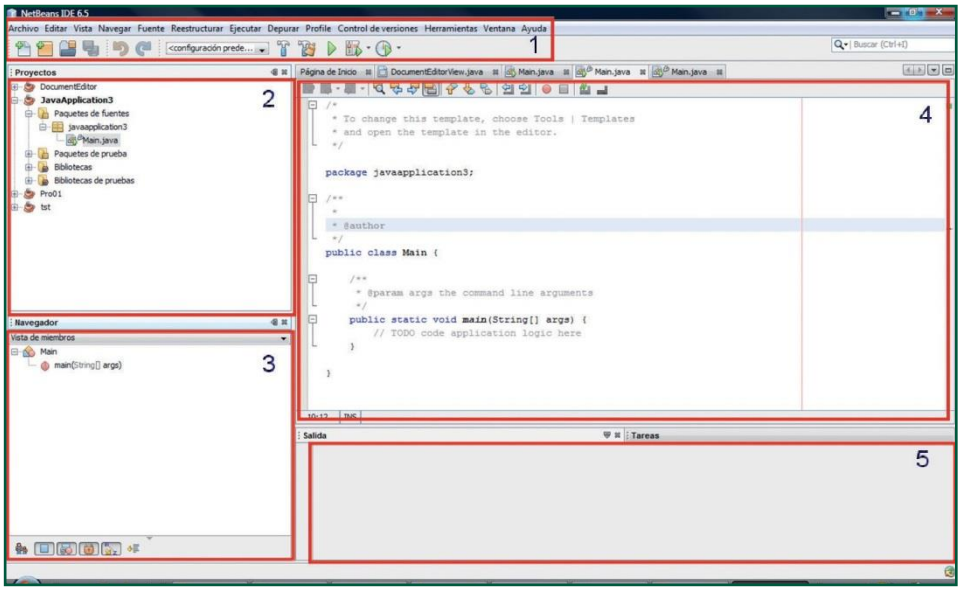

Figura 2.11
Áreas de trabajo de la pantalla principal.

2.8 Edición de programas

La **edición de programas** es una entrada directa de texto, con sentencias permitidas para un determinado lenguaje que luego el ordenador deberá comprender. En teoría, cualquier editor de textos sencillo (como el Bloc de notas de Windows) nos permitirá entrar código en cualquier lenguaje y, luego, pasarlo al compilador.

Sin embargo, esta forma de proceder debe tener un motivo claro, ya que en dicho editor no disponemos de las herramientas de depuración y corrección sintáctica que ofrecen los entornos de desarrollo (Figura 2.12).

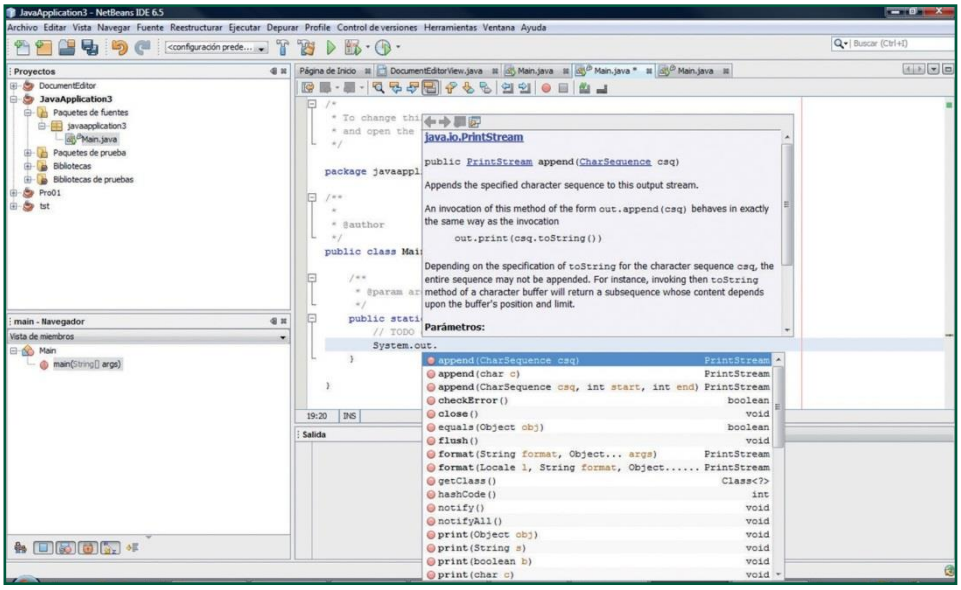

Figura 2.12
Plantillas de ayuda con sugerencias y parámetros que podremos utilizar al entrar código.

Veamos a continuación las características principales del editor del entorno de desarrollo con respecto a un editor de texto convencional:

- Sangrado de líneas para una mayor claridad de lectura.
- Encaja paréntesis y palabras automáticamente armonizando el aspecto.
- Muestra avisos de código erróneo mediante signos y contrastes. Permite utilizar diferentes lenguajes.
- Completa comandos tecleando los primeros caracteres.
- Navegación interna del código fuente mediante palabras clave.

2.9 Generación de ejecutables

Después de las correspondientes pruebas necesarias para comprobar el buen funcionamiento del desarrollo, ha llegado el momento de crear el **ejecutable** para su posterior distribución. Para ello, es preciso clicar la opción "Ejecutar" del menú principal, que muestra la opción "Clean and Build Main Project" y que realiza las siguientes acciones (Figura 2.13):

- Borra los ficheros compilados y el resto de elementos de pruebas previas del mismo proyecto.
- Vuelve a compilar la aplicación y crea un fichero de extensión JAR (Java Archive, archivo comprimido que contiene los ficheros compilados) en el caso de que trabajemos con Java de escritorio (J2SE). Otras extensiones posibles (war, exe) o cualquier otra extensión, quedarán definidas por el empaquetador del lenguaje y estarán listas para distribuirse.

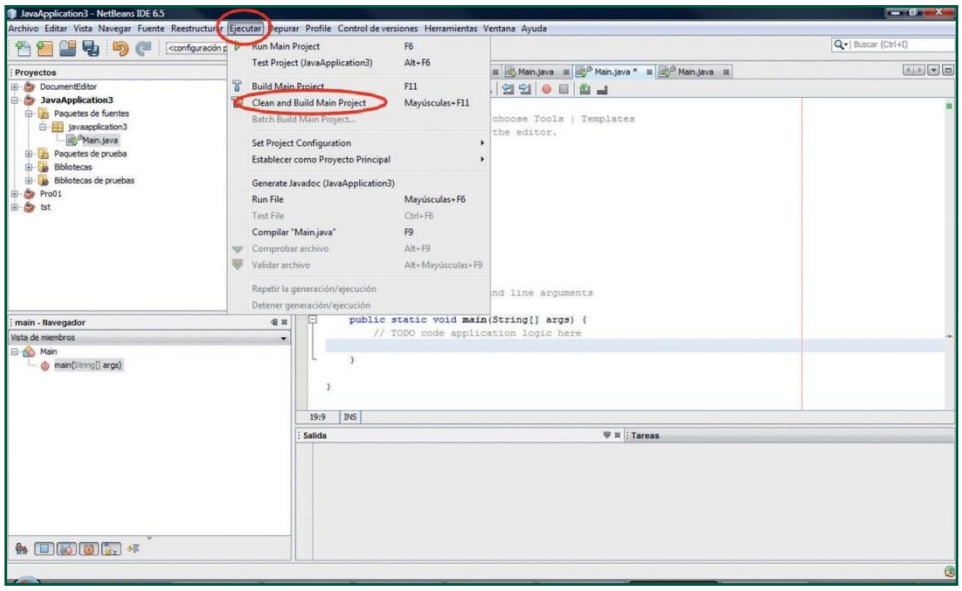

Figura 2.13
Generación de objetos ejecutables listos para usar.

Para saber más

La comunidad NetBeans es muy interesante para saber un poco más acerca del entorno de desarrollo NetBeans. Desde netbeans.org es posible acceder a la última versión del software, así como tener acceso a tutoriales y al resto de la comunidad.

Resumen

Escoger un entorno de desarrollo determinado depende en gran medida del tipo de solución y del mercado que deseemos abordar.

Es importante conocer la funcionalidad y las herramientas y asistentes que el entorno pone en manos del desarrollador: la facilidad de entrada de código, sus características de depuración, la asistencia frente a errores y la facilidad de generar documentación son aspectos a tener en cuenta.

La instalación del entorno de desarrollo de software libre NetBeans parte de la descarga de su instalador en la web del fabricante. Además de instalarlo, podremos actualizarlo o añadirle nuevos módulos que nos ofrecerán mayor potencia y funcionalidad a nuestro entorno de trabajo.

A diferencia del software comercial, que nos obliga a pagar por su uso y que no podemos modificar, el software libre no suele ser de pago y podemos modificar el producto si así lo deseamos creando nuevas funcionalidades que otros programadores pueden utilizar.

La pantalla principal de diseño de NetBeans, y su proceso de creación de código ejecutable, no suele ser demasiado diferente del resto de entornos más usuales, por lo que su aprendizaje nos permitirá abordar otros entornos con mayor facilidad.

Una pantalla básica de desarrollo contempla varias áreas de trabajo: el Área de Menús, donde se ejecutan las acciones más generales; el Área de Proyecto (visión de conjunto); el Área de Navegación, como camino rápido para acceder a algún componente; el Área de Edición, lugar de entrada del código; y el Área de Salida / Tareas, en la que se nos presentarán mensajes informativos desde el entorno.

La Edición de programas mediante un código conocido por el programador puede ser realizada por cualquier procesador de textos; sin embargo, conviene utilizar el editor del entorno debido a todas las herramientas específicas que posee.

Una vez depurado nuestro código, es momento de compilarlo y enlazarlo para poder distribuirlo a nuestros clientes. Eso se consigue mediante la opción '*Build Main Project*' del menu 'Ejecutar' del Área de Menús de NetBeans.

ejercicios deautocomprobación

Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):

1. Las acciones que realizan los entornos de desarrollo se hallan todas enmarcadas en el ciclo de desarrollo, que es idéntico para todo el mundo.

2. Los servidores web locales simulan el funcionamiento de un servidor imaginario de Internet y permiten el desarrollo sin acceder a la red.

3. Los motores de acceso a bases de datos facilitan la gestión de diferentes bases de datos, pero con cambios agresivos en el código de programación.

4. NetBeans es un software libre, por lo que cualquier desarrollador puede crear complementos adicionales o existentes que pueden ser publicados y utilizados por otros programadores.

5. La edición de programas es una entrada indirecta de texto, con sentencias permitidas para un determinado lenguaje que luego el ordenador deberá comprender.

6. La edición de programas mediante un código conocido por el programador no puede ser realizada por cualquier procesador de textos.

7. Una vez depurado un código, es momento de compilarlo y enlazarlo para poder distribuirlo a los clientes. Eso se consigue mediante la opción '*Build Main Project*' del menú 'Ejecutar' del Área de Menús de NetBeans.

Completa las siguientes afirmaciones:

8. El_____de aplicaciones es un_____de programas tipo al que se le deberán aplicar las modificaciones necesarias para _____ al requerimiento concreto del cliente.

9. _____ es un entorno de desarrollo integrado libre, creado principalmente para el lenguaje de programación_____.

10. Para añadir un complemento, abriremos el menú "_____", haremos clic en "_____" y seleccionaremos la pestaña "Complementos _____".

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

soluciones de los ejercicios de autocomprobación

Unidad 1

1. V
2. V
3. F. Los lenguajes de cuarta generación son aquellos que permiten la entrada de códigos mediante instrucciones comprensibles por nosotros.
4. V
5. F. Al programa traductor de código fuente a código objeto se le llama compilador. El hecho de ser un código objeto, y, por tanto, comprensible por el ordenador, no significa que pueda ser ejecutado directamente
6. F. El código o códigos objetos que conforman un código ejecutable, a pesar de estar creados con lenguaje entendible por el ordenador, no pueden ser ejecutados por separado.
7. F. Una máquina virtual es un software que simula un entorno de ordenador concreto en el que podemos instalar un sistema operativo, o programas, y ejecutarlos.
8. código, fuente, editor, software.
9. encapsulación, accesos, objeto.
10. enlazador, librerías, ejecutable.

Unidad 2

1. V
2. F. Los servidores web locales simulan el funcionamiento de un servidor real de Internet y permiten el desarrollo sin acceder a la red.
3. F. Los motores de acceso a bases de datos facilitan la gestión de diferentes bases de datos sin cambios agresivos en el código de programación.
4. V
5. F. La edición de programas es una entrada directa de texto, con sentencias permitidas para un determinado lenguaje que luego el ordenador deberá comprender.
6. F. La edición de programas mediante un código conocido por el programador puede ser realizada por cualquier procesador de textos.
7. V
8. asistente, generador, adaptarlo.
9. NetBeans, Java.
10. Herramientas, Complementos, disponibles.

Índice

MÓDULO: ENTORNOS DE DESARROLLO

UNIDAD FORMATIVA 1