



Escuela Superior de Ingenieros Industriales Industri Injineruen Goimailako Eskola

UNIVERSIDAD DE NAVARRA - NAFARROAKO UNIBERTSITATEA

Aprenda Java *como si estuviera en primero*

San Sebastián, Enero 2000




Javier García de Jalón • José Ignacio Rodríguez
Iñigo Mingo • Aitor Imaz
Alfonso Brazález • Alberto Larzabal • Jesús Calleja • Jon García



Apredna Java

como si estuviera en primero



Javier García de Jalón
José Ignacio Rodríguez
Iñigo Mingo
Aitor Imaz
Alfonso Brazález
Alberto Larzabal
Jesús Calleja
Jon García

Perteneiente a la colección : *“Apredna , como si estuviera en primero”*

ÍNDICE

| | |
|---|-----------|
| 1. INTRODUCCIÓN A JAVA | 1 |
| 1.1 QUÉ ES JAVA 2 | 2 |
| 1.2 EL ENTORNO DE DESARROLLO DE JAVA..... | 2 |
| 1.2.1 <i>El compilador de Java</i> | 3 |
| 1.2.2 <i>La Java Virtual Machine</i> | 3 |
| 1.2.3 <i>Las variables PATH y CLASSPATH</i> | 3 |
| 1.3 UN EJEMPLO COMPLETO COMENTADO | 4 |
| 1.3.1 <i>Clase Ejemplo1</i> | 4 |
| 1.3.2 <i>Clase Geometria</i> | 8 |
| 1.3.3 <i>Clase Rectangulo</i> | 9 |
| 1.3.4 <i>Clase Circulo</i> | 11 |
| 1.3.5 <i>Interface Dibujable</i> | 12 |
| 1.3.6 <i>Clase RectanguloGrafico</i> | 13 |
| 1.3.7 <i>Clase CirculoGrafico</i> | 14 |
| 1.3.8 <i>Clase PanelDibujo</i> | 15 |
| 1.3.9 <i>Clase VentanaCerrable</i> | 17 |
| 1.3.10 <i>Consideraciones adicionales sobre el Ejemplo1</i> | 18 |
| 1.4 NOMENCLATURA HABITUAL EN LA PROGRAMACIÓN EN JAVA | 19 |
| 1.5 ESTRUCTURA GENERAL DE UN PROGRAMA JAVA..... | 19 |
| 1.5.1 <i>Concepto de Clase</i> | 20 |
| 1.5.2 <i>Herencia</i> | 20 |
| 1.5.3 <i>Concepto de Interface</i> | 20 |
| 1.5.4 <i>Concepto de Package</i> | 20 |
| 1.5.5 <i>La jerarquía de clases de Java (API)</i> | 20 |
| 2. PROGRAMACIÓN EN JAVA | 22 |
| 2.1 VARIABLES..... | 22 |
| 2.1.1 <i>Nombres de Variables</i> | 22 |
| 2.1.2 <i>Tipos Primitivos de Variables</i> | 23 |
| 2.1.3 <i>Cómo se definen e inicializan las variables</i> | 23 |
| 2.1.4 <i>Visibilidad y vida de las variables</i> | 24 |
| 2.1.5 <i>Casos especiales: Clases BigInteger y BigDecimal</i> | 25 |
| 2.2 OPERADORES DE JAVA | 25 |
| 2.2.1 <i>Operadores aritméticos</i> | 25 |
| 2.2.2 <i>Operadores de asignación</i> | 26 |
| 2.2.3 <i>Operadores unarios</i> | 26 |
| 2.2.4 <i>Operador instanceof</i> | 26 |
| 2.2.5 <i>Operador condicional ?:</i> | 26 |
| 2.2.6 <i>Operadores incrementales</i> | 26 |
| 2.2.7 <i>Operadores relacionales</i> | 27 |
| 2.2.8 <i>Operadores lógicos</i> | 27 |
| 2.2.9 <i>Operador de concatenación de cadenas de caracteres (+)</i> | 27 |
| 2.2.10 <i>Operadores que actúan a nivel de bits</i> | 28 |
| 2.2.11 <i>Precedencia de operadores</i> | 28 |
| 2.3 ESTRUCTURAS DE PROGRAMACIÓN | 29 |
| 2.3.1 <i>Sentencias o expresiones</i> | 29 |
| 2.3.2 <i>Comentarios</i> | 29 |
| 2.3.3 <i>Bifurcaciones</i> | 30 |
| 2.3.3.1 <i>Bifurcación if</i> | 30 |
| 2.3.3.2 <i>Bifurcación if else</i> | 30 |
| 2.3.3.3 <i>Bifurcación if elseif else</i> | 30 |
| 2.3.3.4 <i>Sentencia switch</i> | 31 |
| 2.3.4 <i>Bucles</i> | 31 |
| 2.3.4.1 <i>Bucle while</i> | 32 |
| 2.3.4.2 <i>Bucle for</i> | 32 |
| 2.3.4.3 <i>Bucle do while</i> | 32 |
| 2.3.4.4 <i>Sentencias break y continue</i> | 32 |
| 2.3.4.5 <i>Sentencias break y continue con etiquetas</i> | 33 |

| | | |
|-----------|---|-----------|
| 2.3.4.6 | Sentencia return | 33 |
| 2.3.4.7 | Bloque try {...} catch {...} finally {...} | 33 |
| 3. | CLASES EN JAVA | 35 |
| 3.1 | CONCEPTOS BÁSICOS | 35 |
| 3.1.1 | Concepto de Clase | 35 |
| 3.1.2 | Concepto de Interface | 36 |
| 3.2 | EJEMPLO DE DEFINICIÓN DE UNA CLASE | 36 |
| 3.3 | VARIABLES MIEMBRO | 37 |
| 3.3.1 | Variables miembro de objeto | 37 |
| 3.3.2 | Variables miembro de clase (static) | 38 |
| 3.4 | VARIABLES FINALES | 38 |
| 3.5 | MÉTODOS (FUNCIONES MIEMBRO) | 39 |
| 3.5.1 | Métodos de objeto | 39 |
| 3.5.2 | Métodos sobrecargados (overloaded) | 40 |
| 3.5.3 | Paso de argumentos a métodos | 40 |
| 3.5.4 | Métodos de clase (static) | 41 |
| 3.5.5 | Constructores | 41 |
| 3.5.6 | Inicializadores | 42 |
| 3.5.6.1 | Inicializadores static | 42 |
| 3.5.6.2 | Inicializadores de objeto | 43 |
| 3.5.7 | Resumen del proceso de creación de un objeto | 43 |
| 3.5.8 | Destrucción de objetos (liberación de memoria) | 43 |
| 3.5.9 | Finalizadores | 43 |
| 3.6 | PACKAGES | 44 |
| 3.6.1 | Qué es un package | 44 |
| 3.6.2 | Cómo funcionan los packages | 45 |
| 3.7 | HERENCIA | 45 |
| 3.7.1 | Concepto de herencia | 45 |
| 3.7.2 | La clase Object | 46 |
| 3.7.3 | Redefinición de métodos heredados | 46 |
| 3.7.4 | Clases y métodos abstractos | 47 |
| 3.7.5 | Constructores en clases derivadas | 47 |
| 3.8 | CLASES Y MÉTODOS FINALES | 48 |
| 3.9 | INTERFACES | 48 |
| 3.9.1 | Concepto de interface | 48 |
| 3.9.2 | Definición de interfaces | 49 |
| 3.9.3 | Herencia en interfaces | 49 |
| 3.9.4 | Utilización de interfaces | 49 |
| 3.10 | CLASES INTERNAS | 50 |
| 3.10.1 | Clases e interfaces internas static | 50 |
| 3.10.2 | Clases internas miembro (no static) | 52 |
| 3.10.3 | Clases internas locales | 54 |
| 3.10.4 | Clases anónimas | 56 |
| 3.11 | PERMISOS DE ACCESO EN JAVA | 57 |
| 3.11.1 | Accesibilidad de los packages | 57 |
| 3.11.2 | Accesibilidad de clases o interfaces | 57 |
| 3.11.3 | Accesibilidad de las variables y métodos miembros de una clase: | 57 |
| 3.12 | TRANSFORMACIONES DE TIPO: CASTING | 58 |
| 3.12.1 | Conversión de tipos primitivos | 58 |
| 3.13 | POLIMORFISMO | 58 |
| 3.13.1 | Conversión de objetos | 59 |
| 4. | CLASES DE UTILIDAD | 61 |
| 4.1 | ARRAYS | 61 |
| 4.1.1 | Arrays bidimensionales | 62 |
| 4.2 | CLASES STRING Y STRINGBUFFER | 62 |
| 4.2.1 | Métodos de la clase String | 63 |
| 4.2.2 | Métodos de la clase StringBuffer | 64 |
| 4.3 | WRAPPERS | 64 |

| | | |
|-----------|---|-----------|
| 4.3.1 | <i>Clase Double</i> | 64 |
| 4.3.2 | <i>Clase Integer</i> | 65 |
| 4.4 | CLASE MATH | 65 |
| 4.5 | COLECCIONES | 66 |
| 4.5.1 | <i>Clase Vector</i> | 66 |
| 4.5.2 | <i>Interface Enumeration</i> | 67 |
| 4.5.3 | <i>Clase Hashtable</i> | 68 |
| 4.5.4 | <i>El Collections Framework de Java 1.2</i> | 68 |
| 4.5.4.1 | Elementos del Java Collections Framework..... | 70 |
| 4.5.4.2 | Interface Collection..... | 71 |
| 4.5.4.3 | Interfaces Iterator y ListIterator | 71 |
| 4.5.4.4 | Interfaces Comparable y Comparator..... | 72 |
| 4.5.4.5 | Sets y SortedSets..... | 73 |
| 4.5.4.6 | Listas..... | 74 |
| 4.5.4.7 | Maps y SortedMaps | 74 |
| 4.5.4.8 | Algoritmos y otras características especiales: Clases Collections y Arrays..... | 75 |
| 4.5.4.9 | Desarrollo de clases por el usuario: clases abstract..... | 76 |
| 4.5.4.10 | Interfaces Cloneable y Serializable | 76 |
| 4.6 | OTRAS CLASES DEL PACKAGE JAVA.UUTIL..... | 77 |
| 4.6.1 | <i>Clase Date</i> | 77 |
| 4.6.2 | <i>Clases Calendar y GregorianCalendar</i> | 77 |
| 4.6.3 | <i>Clases DateFormat y SimpleDateFormat</i> | 79 |
| 4.6.4 | <i>Clases TimeZone y SimpleTimeZone</i> | 80 |
| 5. | EL AWT (ABSTRACT WINDOWS TOOLKIT) | 81 |
| 5.1 | QUÉ ES EL AWT..... | 81 |
| 5.1.1 | <i>Creación de una Interface Gráfica de Usuario</i> | 81 |
| 5.1.2 | <i>Objetos “event source” y objetos “event listener”</i> | 81 |
| 5.1.3 | <i>Proceso a seguir para crear una aplicación interactiva (orientada a eventos)</i> | 82 |
| 5.1.4 | <i>Componentes y eventos soportados por el AWT de Java</i> | 82 |
| 5.1.4.1 | Jerarquía de Componentes | 82 |
| 5.1.4.2 | Jerarquía de eventos..... | 83 |
| 5.1.4.3 | Relación entre Componentes y Eventos..... | 84 |
| 5.1.5 | <i>Interfaces Listener</i> | 85 |
| 5.1.6 | <i>Clases Adapter</i> | 86 |
| 5.2 | COMPONENTES Y EVENTOS..... | 87 |
| 5.2.1 | <i>Clase Component</i> | 88 |
| 5.2.2 | <i>Clases EventObject y AWTEvent</i> | 89 |
| 5.2.3 | <i>Clase ComponentEvent</i> | 89 |
| 5.2.4 | <i>Clases InputEvent y MouseEvent</i> | 89 |
| 5.2.5 | <i>Clase FocusEvent</i> | 90 |
| 5.2.6 | <i>Clase Container</i> | 90 |
| 5.2.7 | <i>Clase ContainerEvent</i> | 91 |
| 5.2.8 | <i>Clase Window</i> | 91 |
| 5.2.9 | <i>Clase WindowEvent</i> | 91 |
| 5.2.10 | <i>Clase Frame</i> | 92 |
| 5.2.11 | <i>Clase Dialog</i> | 92 |
| 5.2.12 | <i>Clase FileDialog</i> | 93 |
| 5.2.13 | <i>Clase Panel</i> | 93 |
| 5.2.14 | <i>Clase Button</i> | 94 |
| 5.2.15 | <i>Clase ActionEvent</i> | 94 |
| 5.2.16 | <i>Clase Canvas</i> | 94 |
| 5.2.17 | <i>Component Checkbox y clase CheckboxGroup</i> | 95 |
| 5.2.18 | <i>Clase ItemEvent</i> | 96 |
| 5.2.19 | <i>Clase Choice</i> | 96 |
| 5.2.20 | <i>Clase Label</i> | 96 |
| 5.2.21 | <i>Clase List</i> | 97 |
| 5.2.22 | <i>Clase Scrollbar</i> | 97 |
| 5.2.23 | <i>Clase AdjustmentEvent</i> | 98 |
| 5.2.24 | <i>Clase ScrollPane</i> | 99 |
| 5.2.25 | <i>Clases TextArea y TextField</i> | 99 |

| | | |
|-----------|---|------------|
| 5.2.26 | Clase <i>TextEvent</i> | 100 |
| 5.2.27 | Clase <i>KeyEvent</i> | 101 |
| 5.3 | MENUS..... | 102 |
| 5.3.1 | Clase <i>MenuShortcut</i> | 102 |
| 5.3.2 | Clase <i>MenuBar</i> | 102 |
| 5.3.3 | Clase <i>Menu</i> | 103 |
| 5.3.4 | Clase <i>MenuItem</i> | 103 |
| 5.3.5 | Clase <i>CheckboxMenuItem</i> | 103 |
| 5.3.6 | Menús <i>pop-up</i> | 104 |
| 5.4 | LAYOUT MANAGERS..... | 104 |
| 5.4.1 | Concepto y Ejemplos de <i>LayoutManagers</i> | 104 |
| 5.4.2 | Ideas generales sobre los <i>LayoutManagers</i> | 105 |
| 5.4.3 | <i>FlowLayout</i> | 105 |
| 5.4.4 | <i>BorderLayout</i> | 106 |
| 5.4.5 | <i>GridLayout</i> | 106 |
| 5.4.6 | <i>CardLayout</i> | 106 |
| 5.4.7 | <i>GridBagLayout</i> | 107 |
| 5.5 | GRÁFICOS, TEXTO E IMÁGENES..... | 108 |
| 5.5.1 | Capacidades gráficas del AWT: Métodos <i>paint()</i> , <i>repaint()</i> y <i>update()</i> | 108 |
| 5.5.1.1 | Método <i>paint(Graphics g)</i> | 108 |
| 5.5.1.2 | Método <i>update(Graphics g)</i> | 108 |
| 5.5.1.3 | Método <i>repaint()</i> | 109 |
| 5.5.2 | Clase <i>Graphics</i> | 109 |
| 5.5.3 | Primitivas gráficas..... | 110 |
| 5.5.4 | Clases <i>Graphics</i> y <i>Font</i> | 110 |
| 5.5.5 | Clase <i>FontMetrics</i> | 111 |
| 5.5.6 | Clase <i>Color</i> | 112 |
| 5.5.7 | Imágenes..... | 112 |
| 5.6 | ANIMACIONES..... | 113 |
| 5.6.1 | Eliminación del parpadeo o <i>flicker</i> redefiniendo el método <i>update()</i> | 114 |
| 5.6.2 | Técnica del doble <i>buffer</i> | 114 |
| 6. | THREADS: PROGRAMAS MULTITAREA..... | 116 |
| 6.1 | CREACIÓN DE THREADS..... | 116 |
| 6.1.1 | Creación de <i>threads</i> derivando de la clase <i>Thread</i> | 117 |
| 6.1.2 | Creación de <i>threads</i> implementando la interface <i>Runnable</i> | 117 |
| 6.2 | CICLO DE VIDA DE UN THREAD..... | 118 |
| 6.2.1 | Ejecución de un nuevo <i>thread</i> | 119 |
| 6.2.2 | Detener un <i>Thread</i> temporalmente: <i>Runnable</i> - <i>Not Runnable</i> | 119 |
| 6.2.3 | Finalizar un <i>Thread</i> | 121 |
| 6.3 | SINCRONIZACIÓN..... | 121 |
| 6.4 | PRIORIDADES..... | 124 |
| 6.5 | GRUPOS DE THREADS..... | 124 |
| 7. | APPLETS..... | 126 |
| 7.1 | QUÉ ES UN APPLET..... | 126 |
| 7.1.1 | Algunas características de las <i>applets</i> | 126 |
| 7.1.2 | Métodos que controlan la ejecución de un <i>applet</i> | 127 |
| 7.1.2.1 | Método <i>init()</i> | 127 |
| 7.1.2.2 | Método <i>start()</i> | 127 |
| 7.1.2.3 | Método <i>stop()</i> | 127 |
| 7.1.2.4 | Método <i>destroy()</i> | 127 |
| 7.1.3 | Métodos para dibujar el <i>applet</i> | 127 |
| 7.2 | CÓMO INCLUIR UN APPLET EN UNA PÁGINA HTML..... | 128 |
| 7.3 | PASO DE PARÁMETROS A UN APPLET..... | 128 |
| 7.4 | CARGA DE APPLETS..... | 129 |
| 7.4.1 | Localización de <i>ficheros</i> | 129 |
| 7.4.2 | Archivos <i>JAR</i> (<i>Java Archives</i>)..... | 129 |
| 7.5 | COMUNICACIÓN DEL APPLET CON EL BROWSER..... | 129 |
| 7.6 | SONIDOS EN APPLETS..... | 130 |
| 7.7 | IMÁGENES EN APPLETS..... | 131 |

| | | |
|------------|---|------------|
| 7.8 | OBTENCIÓN DE LAS PROPIEDADES DEL SISTEMA | 132 |
| 7.9 | UTILIZACIÓN DE THREADS EN APPLETS | 132 |
| 7.10 | APPLETS QUE TAMBIÉN SON APLICACIONES | 133 |
| 8. | EXCEPCIONES | 135 |
| 8.1 | EXCEPCIONES ESTÁNDAR DE JAVA | 135 |
| 8.2 | LANZAR UNA EXCEPTION | 136 |
| 8.3 | CAPTURAR UNA EXCEPTION | 137 |
| | 8.3.1 Bloques try y catch | 137 |
| | 8.3.2 Relanzar una Exception | 138 |
| | 8.3.3 Método finally {...} | 138 |
| 8.4 | CREAR NUEVAS EXCEPCIONES | 139 |
| 8.5 | HERENCIA DE CLASES Y TRATAMIENTO DE EXCEPCIONES | 139 |
| 9. | ENTRADA/SALIDA DE DATOS EN JAVA 1.1 | 140 |
| 9.1 | CLASES DE JAVA PARA LECTURA Y ESCRITURA DE DATOS | 140 |
| | 9.1.1 Los nombres de las clases de java.io | 141 |
| | 9.1.2 Clases que indican el origen o destino de los datos | 142 |
| | 9.1.3 Clases que añaden características | 143 |
| 9.2 | ENTRADA Y SALIDA ESTÁNDAR (TECLADO Y PANTALLA) | 143 |
| | 9.2.1 Salida de texto y variables por pantalla | 144 |
| | 9.2.2 Lectura desde teclado | 144 |
| | 9.2.3 Método práctico para leer desde teclado | 145 |
| 9.3 | LECTURA Y ESCRITURA DE ARCHIVOS | 146 |
| | 9.3.1 Clases File y FileDialog | 146 |
| | 9.3.2 Lectura de archivos de texto | 148 |
| | 9.3.3 Escritura de archivos de texto | 148 |
| | 9.3.4 Archivos que no son de texto | 148 |
| 9.4 | SERIALIZACIÓN | 149 |
| | 9.4.1 Control de la serialización | 150 |
| | 9.4.2 Externalizable | 150 |
| 9.5 | LECTURA DE UN ARCHIVO EN UN SERVIDOR DE INTERNET | 151 |
| 10. | OTRAS CAPACIDADES DE JAVA | 152 |
| 10.1 | JAVA FOUNDATION CLASSES (JFC) Y JAVA 2D | 152 |
| 10.2 | JAVA MEDIA FRAMEWORK (JMF) | 152 |
| 10.3 | JAVA 3D | 152 |
| 10.4 | JAVABEANS | 153 |
| 10.5 | JAVA EN LA RED | 153 |
| 10.6 | JAVA EN EL SERVIDOR: SERVLETS | 153 |
| 10.7 | RMI Y JAVA IDL | 154 |
| 10.8 | SEGURIDAD EN JAVA | 154 |
| 10.9 | ACCESO A BASES DE DATOS (JDBC) | 154 |
| 10.10 | JAVA NATIVE INTERFACE (JNI) | 155 |

1. INTRODUCCIÓN A JAVA

Java surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, era importante conseguir una herramienta independiente del tipo de CPU utilizada. Desarrollaron un código “neutro” que no dependía del tipo de electrodoméstico, el cual se ejecutaba sobre una “*máquina hipotética o virtual*” denominada **Java Virtual Machine (JVM)**. Era la **JVM** quien interpretaba el código neutro convirtiéndolo a código particular de la CPU utilizada. Esto permitía lo que luego se ha convertido en el principal lema del lenguaje: “*Write Once, Run Everywhere*”. A pesar de los esfuerzos realizados por sus creadores, ninguna empresa de electrodomésticos se interesó por el nuevo lenguaje.

Como lenguaje de programación para computadores, **Java** se introdujo a finales de 1995. La clave fue la incorporación de un intérprete **Java** en la versión 2.0 del programa Netscape Navigator, produciendo una verdadera revolución en Internet. **Java 1.1** apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. **Java 1.2**, más tarde rebautizado como **Java 2**, nació a finales de 1998.

Al programar en **Java** no se parte de cero. Cualquier aplicación que se desarrolle “cuelga” (o se apoya, según como se quiera ver) en un gran número de **clases** preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el **API** o **Application Programming Interface** de **Java**). **Java** incorpora en el propio lenguaje muchos aspectos que en cualquier otro lenguaje son extensiones propiedad de empresas de software o fabricantes de ordenadores (threads, ejecución remota, componentes, seguridad, acceso a bases de datos, etc.). Por eso muchos expertos opinan que **Java** es el lenguaje ideal para aprender la informática moderna, porque incorpora todos estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo.

El principal objetivo del lenguaje **Java** es llegar a ser el “nexo universal” que conecte a los usuarios con la información, esté ésta situada en el ordenador local, en un servidor de **Web**, en una base de datos o en cualquier otro lugar.

Java es un lenguaje muy completo (de hecho se está convirtiendo en un macro-lenguaje: **Java 1.0** tenía 12 packages; **Java 1.1** tenía 23 y **Java 1.2** tiene 59). En cierta forma casi todo depende de casi todo. Por ello, conviene aprenderlo de modo *iterativo*: primero una visión muy general, que se va refinando en sucesivas iteraciones. Una forma de hacerlo es empezar con un ejemplo completo en el que ya aparecen algunas de las características más importantes.

La compañía **Sun** describe el lenguaje **Java** como “*simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico*”. Además de una serie de halagos por parte de **Sun** hacia su propia criatura, el hecho es que todo ello describe bastante bien el lenguaje **Java**, aunque en algunas de esas características el lenguaje sea todavía bastante mejorable. Algunas de las anteriores ideas se irán explicando a lo largo de este manual.

1.1 QUÉ ES JAVA 2

Java 2 (antes llamado **Java 1.2** o **JDK 1.2**) es la tercera versión importante del lenguaje de programación **Java**.

No hay cambios conceptuales importantes respecto a **Java 1.1** (en **Java 1.1** sí los hubo respecto a **Java 1.0**), sino extensiones y ampliaciones, lo cual hace que a muchos efectos –por ejemplo, para esta introducción– sea casi lo mismo trabajar con **Java 1.1** o con **Java 1.2**.

Los programas desarrollados en **Java** presentan diversas ventajas frente a los desarrollados en otros lenguajes como C/C++. La ejecución de programas en **Java** tiene muchas posibilidades: ejecución como aplicación independiente (**Stand-alone Application**), ejecución como **applet**, ejecución como **servlet**, etc. Un **applet** es una aplicación especial que se ejecuta dentro de un navegador o browser (por ejemplo *Netscape Navigator* o *Internet Explorer*) al cargar una página HTML desde un servidor **Web**. El **applet** se descarga desde el servidor y no requiere instalación en el ordenador donde se encuentra el browser. Un **servlet** es una aplicación sin interface gráfica que se ejecuta en un servidor de Internet. La ejecución como aplicación independiente es análoga a los programas desarrollados con otros lenguajes.

Además de incorporar la ejecución como **Applet**, **Java** permite fácilmente el desarrollo tanto de arquitecturas cliente-servidor como de aplicaciones distribuidas, consistentes en crear aplicaciones capaces de conectarse a otros ordenadores y ejecutar tareas en varios ordenadores simultáneamente, repartiendo por lo tanto el trabajo. Aunque también otros lenguajes de programación permiten crear aplicaciones de este tipo, **Java** incorpora en su propio **API** estas funcionalidades.

1.2 EL ENTORNO DE DESARROLLO DE JAVA

Existen distintos programas comerciales que permiten desarrollar código **Java**. La compañía **Sun**, creadora de **Java**, distribuye gratuitamente el *Java(tm) Development Kit (JDK)*. Se trata de un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en **Java**. Incorpora además la posibilidad de ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables (con el denominado **Debugger**). Cualquier programador con un mínimo de experiencia sabe que una parte muy importante (muchas veces la mayor parte) del tiempo destinado a la elaboración de un programa se destina a la **detección y corrección de errores**. Existe también una versión reducida del **JDK**, denominada **JRE (Java Runtime Environment)** destinada únicamente a ejecutar código **Java** (no permite compilar).

Los **IDEs (Integrated Development Environment)**, tal y como su nombre indica, son entornos de desarrollo integrados. En un mismo programa es posible escribir el código **Java**, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Algunos incluyen una herramienta para realizar **Debug** gráficamente, frente a la versión que incorpora el **JDK** basada en la utilización de una consola (denominada habitualmente ventana de comandos de MS-DOS, en **Windows NT/95/98**) bastante difícil y pesada de utilizar. Estos entornos integrados permiten desarrollar las aplicaciones de forma mucho más rápida, incorporando en muchos casos librerías con **componentes** ya desarrollados, los cuales se incorporan al proyecto o programa. Como inconvenientes se pueden señalar algunos fallos de compatibilidad entre plataformas, y ficheros resultantes de mayor tamaño que los basados en clases estándar.

Gracias por visitar este Libro Electrónico

Puedes leer la versión completa de este libro electrónico en diferentes formatos:

- HTML(Gratis / Disponible a todos los usuarios)
- PDF / TXT(Disponible a miembros V.I.P. Los miembros con una membresía básica pueden acceder hasta 5 libros electrónicos en formato PDF/TXT durante el mes.)
- Epub y Mobipocket (Exclusivos para miembros V.I.P.)

Para descargar este libro completo, tan solo seleccione el formato deseado, abajo:

