Java

duke

30 de julio de 2015

Índice general

Kernel. ,Nueva API para el manejo de Días y Fechas, la cual reemplazara las antiguas clases Date y Calendar. ,Posibilidad de operar con clases BigDecimal usando operandos. , ,Java SE 8 — lanzada en marzo de 2014. Cabe destacar: Incorpora de forma completa la librería JavaFX. ,Diferentes mejoras en seguridad. ,Diferentes mejoras en concurrencia. ,Añade funcionalidad para programación funcional mediante expresiones Lambda. ,Mejora la integración de JavaScript. ,Nuevas API para manejo de fechas y tiempo (date - time). , , ,En el 2005 se calcula en 4,5 millones el número de desarrolladores y 2.500 millones de dispositivos habilitados con tecnología Java.

Capítulo 1

Historia

Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en el año 1991. El equipo (Green Team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road en Menlo Park en su desarrollo. , ,El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarse Green tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas y finalmente se renombró a Java. , ,Es frecuentada por algunos de los miembros del equipo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus diseñadores: James Gosling, Arthur Van Hoff, y Andy Bechtolsheim. Otros abogan por el siguiente acrónimo, Just Another Vague Acronym ("sólo otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la de que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de java sea una taza de café caliente. Un pequeño signo que da fuerza a esta teoría es que los 4 primeros bytes (el número mágico) de los archivos.class que genera el compilador, son en hexadecimal, 0xCAFEBABE. A pesar de todas estas teorías, el nombre fue sacado al parecer de una lista aleatoria de palabras.3 , ,Los objetivos de Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. Entre junio y julio de 1994, tras una sesión maratoniana de tres días entre John Gaga, James Gosling, Patrick Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web. Sintieron que la llegada del navegador web Mosaic, propiciaría que Internet se convirtiese en un medio interactivo, como el que pensaban era la televisión por cable. Naughton creó entonces un prototipo de navegador, WebRunner, que más tarde sería conocido como HotJava. , ,En 1994, se les hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun. Java 1.0a pudo descargarse por primera vez en 1994, pero hubo que esperar al 23 de mayo de 1995, durante las conferencias de SunWorld, a que vieran la luz pública Java y HotJava, el navegador Web. El acontecimiento fue anunciado por John Gage, el Director Científico de Sun Microsystems. El acto estuvo acompañado por una pequeña sorpresa adicional, el anuncio por parte de Marc Andreessen, Vicepresidente Ejecutivo de Netscape, de que Java sería soportado en sus navegadores. El 9 de enero del año siguiente, 1996, Sun fundó el grupo empresarial JavaSoft para que se encargase del desarrollo tecnológico. [1] Dos semanas más tarde la primera versión de Java fue publicada. , ,La promesa inicial de Gosling era Write Once, Run Anywhere (Escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución (la JVM) ligero y gratuito para las plataformas más populares de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma. , ,El entorno de ejecución era relativamente seguro y

los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web. , ,Java ha experimentado numerosos cambios desde la versión primigenia, JDK 1.0, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar.4 , ,Desde J2SE 1.4, la evolución del lenguaje ha sido regulada por el JCP (Java Community Process), que usa Java Specification Requests (JSRs) para proponer y especificar cambios en la plataforma Java. El lenguaje en sí mismo está especificado en la Java Language Specification (JLS), o Especificación del Lenguaje Java. Los cambios en los JLS son gestionados en JSR 901. ,JDK 1.0 (23 de enero de 1996) — Primer lanzamiento: comunicado de prensa ,JDK 1.1 (19 de febrero de 1997) — Principales adiciones incluidas: comunicado de prensa una reestructuración intensiva del modelo de eventos AWT (Abstract Windowing Toolkit) ,clases internas (inner classes) ,JavaBeans ,JDBC (Java Database Connectivity), para la integración de bases de datos ,RMI (Remote Method Invocation) , ,J2SE 1.2 (8 de diciembre de 1998) — Nombre clave Playground. Esta y las siguientes versiones fueron recogidas bajo la denominación Java 2 y el nombre "J2SE"(Java 2 Platform, Standard Edition), reemplazó a JDK para distinguir la plataforma base de J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition). Otras mejoras añadidas incluían: comunicado de prensa la palabra reservada (keyword) strictfp ,reflexión en la programación ,la API gráfica (Swing) fue integrada en las clases básicas ,la máquina virtual (JVM) de Sun fue equipada con un compilador JIT (Just in Time) por primera vez ,Java Plug-in ,Java IDL, una implementación de IDL (Lenguaje de Descripción de Interfaz) para la interoperabilidad con CORBA ,Colecciones (Collections) , ,J2SE 1.3 (8 de mayo de 2000) — Nombre clave Kestrel. Los cambios más notables fueron:comunicado de prensa lista completa de cambios la inclusión de la máquina virtual de HotSpot JVM (la JVM de HotSpot fue lanzada inicialmente en abril de 1999, para la JVM de J2SE 1.2) ,RMI fue cambiado para que se basara en CORBA ,JavaSound ,se incluyó el Java Naming and Directory Interface (JNDI) en el paquete de bibliotecas principales (anteriormente disponible como una extensión) ,Java Platform Debugger Architecture (JPDA) , ,J2SE 1.4 (6 de febrero de 2002) — Nombre Clave Merlin. Este fue el primer lanzamiento de la plataforma Java desarrollado bajo el Proceso de la Comunidad Java como JSR 59. Los cambios más notables fueron: comunicado de prensalista completa de cambios Palabra reservada assert (Especificado en JSR 41.) ,Expresiones regulares modeladas al estilo de las expresiones regulares Perl ,Encadenación de excepciones Permite a una excepción encapsular la excepción de bajo nivel original. ,non-blocking NIO (New Input/Output) (Especificado en JSR 51.) ,Logging API (Specified in JSR 47.) ,API I/O para la lectura y escritura de imágenes en formatos como JPEG o PNG ,Parser XML integrado y procesador XSLT (JAXP) (Especificado en JSR 5 y JSR 63.) ,Seguridad integrada y extensiones criptográficas (JCE, JSSE, JAAS) ,Java Web Start incluido (El primer lanzamiento ocurrió en marzo de 2001 para J2SE 1.3) (Especificado en JSR 56.) , ,J2SE 5.0 (30 de septiembre de 2004)

— Nombre clave: Tiger. (Originalmente numerado 1.5, esta notación aún es usada internamente.[2]) Desarrollado bajo JSR 176, Tiger añadió un número significativo de nuevas características comunicado de prensa Plantillas (genéricos) — provee conversión de tipos (type safety) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de conversión de tipos (type casting). (Especificado por JSR 14.) ,Metadatos — también llamados anotaciones, permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades de proceso de metadatos. (Especificado por JSR 175.) ,Autoboxing/unboxing — Conversiones automáticas entre tipos primitivos (Como los int) y clases de envoltura primitivas (Como Integer). (Especificado por JSR 201.) ,Enumeraciones — la palabra reservada enum crea una typesafe, lista ordenada de valores (como Dia.LUNES, Dia.MARTES, etc.). Anteriormente, esto solo podía ser llevado a cabo por constantes enteras o clases construidas manualmente (enum pattern). (Especificado por JSR 201.) ,Varargs (número de argumentos variable) — El último parámetro de un método puede ser declarado con el nombre del tipo seguido por tres puntos (e.g. void drawtext(String... lines)). En la llamada al método, puede usarse cualquier número de parámetros de ese tipo, que serán almacenados en un array para pasarlos al método. ,Bucle for mejorado — La sintaxis para el bucle for se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente Iterable, como la clase estándar Collection, de la siguiente forma: , ,Este ejemplo itera sobre el objeto Iterable widgets, asignando, en orden, cada uno de los elementos a la variable w, y llamando al método display() de cada uno de ellos. (Especificado por JSR 201.) ,Java SE 6 (11 de diciembre de 2006) — Nombre clave Mustang. Estuvo en desarrollo bajo la JSR 270. En esta versión, Sun cambió el nombre "J2SE"por Java SE y eliminó el ".0"del número de versión.[3]. Está disponible en http://java.sun.com/javase/6/. Los cambios más importantes introducidos en esta versión son: Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript. ,Incluye el motor Rhino, de Mozilla, una implementación de Javascript en Java. ,Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones para Servicios Web, como JAX-WS 2.0, JAXB 2.0, STAX y JAXP. ,Mejoras en la interfaz gráfica y en el rendimiento. , ,Java SE 7 — Nombre clave Dolphin. En el año 2006 aún se encontraba en las primeras etapas de planificación. Su lanzamiento fue en julio de 2011. Soporte para XML dentro del propio lenguaje. ,Un nuevo concepto de superpaquete. ,Soporte para closures. ,Introducción de anotaciones estándar para detectar fallos en el software. , ,No oficiales: NIO2. ,Java Module System. ,Java

Capítulo 2

Filosofía

El lenguaje Java se creó con cinco objetivos principales: 1.Debería usar el paradigma de la programación orientada a objetos. 2.Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos. 3.Debería incluir por defecto soporte para trabajo en red. 4.Debería diseñarse para ejecutar código en sistemas remotos de forma segura. 5.Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture), Internet Communications Engine o OSGi respectivamente.

Orientado a objetos[editar]

La primera característica, orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, debería en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones. En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su "comportamiento" (soldar dos piezas, etc.), el objeto "aluminio" puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de "código abierto" (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

Independencia de la plataforma[editar]

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run anywhere".

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode" (específicamente Java bytecode)—instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de compiladores sólo puede ejecutarse en un tipo de arquitectura.

La licencia sobre Java de Sun insiste que todas las implementaciones sean "compatibles". Esto dio lugar a una disputa legal entre Microsoft y Sun, cuando éste último alegó que la implementación de Microsoft no daba soporte a las interfaces RMI y JNI además de haber añadido características ''dependientes" de su plataforma. Sun demandó a Microsoft y ganó por daños y perjuicios (unos 20 millones de dólares) así como una orden judicial forzando la acatación de la licencia de Sun. Como respuesta, Microsoft no ofrece Java con su versión de sistema operativo, y en recientes versiones de Windows, su navegador Internet Explorer no admite la ejecución de applets sin un conector (o plugin) aparte. Sin embargo, Sun y otras fuentes ofrecen versiones gratuitas para distintas versiones de Windows.

Las primeras implementaciones del lenguaje usaban una máquina virtual interpretada para conseguir la portabilidad. Sin embargo, el resultado eran programas que se ejecutaban comparativamente más lentos que aquellos escritos en C o C++. Esto hizo que Java se ganase una reputación de lento en rendimiento. Las implementaciones recientes de la JVM dan lugar a programas que se ejecutan considerablemente más rápido que las versiones antiguas, empleando diversas técnicas, aunque sigue siendo mucho más lento que otros lenguajes.

La primera de estas técnicas es simplemente compilar directamente en código nativo como hacen los compiladores tradicionales, eliminando la etapa del bytecode. Esto da lugar a un gran rendimiento en la ejecución, pero tapa el camino a la portabilidad. Otra técnica, conocida como compilación JIT (Just In Time, o çompilación al vuelo"), convierte el bytecode a código nativo cuando se ejecuta la

aplicación. Otras máquinas virtuales más sofisticadas usan una recompilación dinámica.[en] la que la VM es capaz de analizar el comportamiento del programa en ejecución y recompila y optimiza las partes críticas. La recompilación dinámica puede lograr mayor grado de optimización que la compilación tradicional (o estática), ya que puede basar su trabajo en el conocimiento que de primera mano tiene sobre el entorno de ejecución y el conjunto de clases cargadas en memoria. La compilación JIT y la recompilación dinámica permiten a los programas Java aprovechar la velocidad de ejecución del código nativo sin por ello perder la ventaja de la portabilidad en ambos.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en múltiples plataformas de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run anywhereçomo "Write once, debug everywhere"(o "Escríbelo una vez, ejecútalo en cualquier parte" por "Escríbelo una vez, depúralo en todas partes").

El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

El recolector de basura[editar]

Véase también: Recolector de basura

En Java el problema fugas de memoria se evita en gran medida gracias a la recolección de basura (o automatic garbage collector). El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de éste; al salir del método el objeto es eliminado). Aun así, es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios—es decir, pueden aún ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos y mayor seguridad.

Capítulo 3

Sintaxis

La sintaxis de Java se deriva en gran medida de C++. Pero a diferencia de éste, que combina la sintaxis para programación genérica, estructurada y orientada a objetos, Java fue construido desde el principio para ser completamente orientado a objetos. Todo en Java es un objeto (salvo algunas excepciones), y todo en Java reside en alguna clase (recordemos que una clase es un molde a partir del cual pueden crearse varios objetos).

Aplicaciones autónomas[editar]

Este ejemplo necesita una pequeña explicación. Todo en Java está dentro de una clase, incluyendo programas autónomos. El código fuente se guarda en archivos con el mismo nombre que la clase que contienen y con extensión ".java". Una clase (class) declarada pública (public) debe seguir este convenio. En el ejemplo anterior, la clase es Hola, por lo que el código fuente debe guardarse en el fichero "Hola.java" El compilador genera un archivo de clase (con extensión ".class") por cada una de las clases definidas en el archivo fuente. Una clase anónima se trata como si su nombre fuera la concatenación del nombre de la clase que la encierra, el símbolo "$", y un número entero. Los programas que se ejecutan de forma independiente y autónoma, deben contener el método "main()". La palabra reservada "void" indica que el método main no devuelve nada. El método main debe aceptar un array de objetos tipo String. Por acuerdo se referencia como "args", aunque puede emplearse cualquier otro identificador. La palabra reservada "static" indica que el método es un método de clase, asociado a la clase en vez de a una instancia de la misma. El método main debe ser estático o "de clase". La palabra reservada public significa que un método puede ser llamado desde otras clases, o que la clase puede ser usada por clases fuera de la jerarquía de la propia clase. Otros tipos de acceso son "private" o "protected". La utilidad de impresión (en pantalla por ejemplo) forma parte de la biblioteca estándar de Java: la clase "'System"' define un campo público estático llamado "'out"'. El objeto out es una instancia de "'PrintStream"', que ofrece el método "'println (String)"' para volcar datos en la pantalla (la salida estándar). Las aplicaciones autónomas se ejecutan dando al entorno de ejecución de Java el nombre de la clase cuyo método main debe invocarse. Por ejemplo, una línea de comando (en Unix o Windows) de la forma java –cp . Hola ejecutará el programa del ejemplo (previamente compilado y generado "Hola.class"). El nombre de la clase cuyo método main se llama puede especificarse también en el fichero "MANIFEST" del archivo de empaquetamiento de Java (.jar).

Applets[editar]

Las applet Java son programas incrustados en otras aplicaciones, normalmente una página Web que se muestra en un navegador.

Actualmente HTML 5 ha eliminado el uso de la etiqueta <applet>. Pero todavía existe la forma de usarlo en HTML5. (Texto en inglés) Java Applets in HTML5.

La sentencia import indica al compilador de Java que incluya las clases java.applet. Applet y java.awt. Graphics, para poder referenciarlas por sus nombres, sin tener que anteponer la ruta completa cada vez que se quieran usar en el código fuente.

La clase Hola extiende (extends) a la clase Applet, es decir, es una subclase de ésta. La clase Applet permite a la aplicación mostrar y controlar el estado del applet. La clase Applet es un componente del AWT (Abstract Window Toolkit), que permite al applet mostrar una interfaz gráfica de usuario o GUI (Graphical User Interface), y responder a eventos generados por el usuario.

La clase Hola sobrecarga el método paint (Graphics) heredado de la superclase contenedora (Applet en este caso), para acceder al código encargado de dibujar. El método paint() recibe un objeto Graphics que contiene el contexto gráfico para dibujar el applet. El método paint() llama al método drawString (String, int, int) del objeto [4]

Servlets[editar]

Artículo principal: Java Servlet

Los servlets son componentes de la parte del servidor de Java EE, encargados de generar respuestas a las peticiones recibidas de los clientes.

Las sentencias import indican al compilador de Java la inclusión de todas las clases públicas e interfaces de los paquetes java.io y javax.servlet en la compilación.

La clase Hola extiende (extends), es heredera de la clase GenericServlet. Esta clase proporciona la interfaz para que el servidor le pase las peticiones al servlet y el mecanismo para controlar el ciclo de vida del servlet.

La clase Hola sobrecarga el método service (ServletRequest, ServletResponse), definido por la interfaz servlet para acceder al manejador de la petición de servicio. El método service() recibe un objeto de tipo ServletRequest que contiene la petición del cliente y un objeto de tipo ServletResponse, usado para generar la respuesta que se devuelve al cliente. El método service() puede lanzar (throws) excepciones de tipo ServletException e IOException si ocurre algún tipo de anomalía.

El método setContentType (String) en el objeto respuesta establece el tipo de contenido MIME a "text/html", para indicar al cliente que la respuesta a su petición es una página con formato HTML. El método getWriter() del objeto respuesta devuelve un objeto de tipo PrintWriter, usado como una tubería por la que viajarán los datos al cliente. El método println (String) escribe la cadena "Hola, mundo!.[en] la respuesta y finalmente se llama al método close() para cerrar la conexión, que hace que los datos escritos en la tubería o stream sean devueltos al cliente.

Aplicaciones con ventanas[editar]

Swing es la biblioteca para la interfaz gráfica de usuario avanzada de la plataforma Java SE. Las instrucciones import indican al compilador de Java que las clases e interfaces del paquete javax.swing se incluyan en la compilación.

La clase Hola extiende (extends) la clase javax.swing.JFrame, que implementa una ventana con una barra de título y un control para cerrarla.

El constructor Hola() inicializa el marco o frame llamando al método setDefaultCloseOperation (int) heredado de JFrame para establecer las operaciones por defecto cuando el control de cierre en la barra de título es seleccionado al valor WindowConstants.DISPOSE_ON_CLOSE. Esto hace que se liberen los recursos tomados por la ventana cuando es cerrada, y no simplemente ocultada, lo que permite a la máquina virtual y al programa acabar su ejecución. A continuación se crea un objeto de tipo JLabel con el texto "Hola, mundo!", y se añade al marco mediante el método add (Component), heredado de la clase Container. El método pack(), heredado de la clase Window, es invocado para dimensionar la ventana y distribuir su contenido.

El método main() es llamado por la JVM al comienzo del programa. Crea una instancia de la clase Hola y hace la ventana sea mostrada invocando al método setVisible (boolean) de la superclase (clase de la que hereda) con el parámetro a true. Véase que, una vez el marco es dibujado, el programa no termina cuando se sale del método main(), ya que el código del que depende se encuentra en un hilo de ejecución independiente ya lanzado, y que permanecerá activo hasta que todas las ventanas hayan sido destruidas.

Capítulo 4

Sobre el autor: duke

Mascota de java