APÉNDICE W4
Glosario de términos de Programación

Abstraction *(abstracción)*	Propiedad y/o técnica de software que oculta los detalles de la implementación. Java soporta abstracción de clases y abstracción de métodos. La *abstracción de métodos* se define separando el uso de un método sin conocer como está implementado ese método. Si decide combinar la implementación, el programa cliente será afectado. De modo similar la *abstracción de clases* oculta la implementación de la clase del cliente.
Acoplamiento *(coupling)*	Medida del grado en el que un objeto o componente depende de otro. Bajo acoplamiento minimiza las dependencias y es una indicación de un buen diseño
Agregación (*aggregation)*	Relación en la que un objeto se compone o está construido de uno o más objetos, de modo que la colección completa representa un todo. Las relaciones de agregación se especifican entre clases y se reflejan en instancias de objetos
Algoritmo *(algorithm)*	Método que describe cómo se resuelve un problema en término de las acciones que se ejecutan y especifica el orden en que se ejecutan estas acciones. Los algoritmos ayudan al programador a planificar un programa antes de su escritura en un lenguaje de programación.
Ámbito de clase *(scope class)*	Las variables *privadas* definidas fuera de los *método*s internos a la clase tienen ámbito de clase. Son accesibles desde todos los métodos del interior de la clase, con independencia del rden en que están definidas. Los métodos privados también tiene ámbito de clase.
Análisis *(análysis)*	Proceso de identificación, modelado y descripción de lo que hace un sistema y de cómo trabaja
Aplicación *(application)*	Programa autónomo Java tal como cualquier programa escrito utilizando un lenguaje de alto nivel. Las aplicaciones se pueden ejecutar desde cualquier computadora con un interprete Java. Las aplicaciones no están sometidas a las restricciones impuestas los *applets* de

	Java. Una clase aplicación debe contener un método `main`. Se utiliza como sinónimo de *programa.*
Applet	Tipo especial de programa Java que se puede ejecutar (correr) directamente en un navegador Web o en un visualizador *applet.* A un *applet* se le imponen diversas restricciones de seguridad. Por ejemplo, un *applet* no se puede ejecutar operaciones de entrada/salida en un sistema de usuario y por consiguiente no puede leer o escribir archivos o transmitir virus de computadora.
Argumento *(argument)*	Información pasada a un método. Los argumentos se suelen llamar también parámetros. Unmétodo que espera recibir argumentos debe contener una declaración de *argumentos formales* por cada *argumento actual* como parte de la cabecera del mismo. Cuando se invoca a un método, los valores de los argumentos actuales 8reales) se copia en los correspondientes argumentos formales. *Vease* parámetro actual *(actual parameter).*
Array *(array, vector, lista)*	Objeto contenedor que almacena una secuencia indexada de los mismos tipos de datos. Normalmente los elementos individuales se referencian por el valor de un índice. El índice es un valor entero que , suele comenzar, en 0 para el primer elementos, 1 para el segundo y así sucesivamente.
Asignación *(assignment)*	Almacenamiento de un valor en una variable. La sentencia de asignación es aquella que implementa la asignación y utiliza un operador de asignación
Asociación *(association)*	Una relación entre dos clases tales como una instancia de una clase referencia a una instancia de otra clase.
Asociatividad *(associativity)*	Orden en que se envalúan operadores de igual precedencia o prioridad dentro de una expresión. La asociatividad por la izquierda produce una evaluación de izquierda a derecha y la asociatividad por la derecha conduce a una evaluación de derecha a izquierda.
AWT (ABSTRACT WINDOW TOOLKIT)	Colección de clases (`java.awt.*`) que se utiliza para implementar interfaces gráficas de usuario. Contiene componentes tales como botones, etiquetas, campos de texto, áreas de texto, barras de desplazamiento, cajas de verificación y menús. Las clases de AWT proporcionan una interfaz independiente de la plataforma para desarrollo de programas visuales e interfaces gráficas de usuario.
Biblioteca de clases *(class library)*	Colección organizada de clases que proporciona un conjunto de componentes y abstracciones reutilizables
Binario *(binary)*	Representación numérica en base 2. En esta base sólo se utilizan los dígitos 0 y 1. Las posiciones de los d´digitos representan potencias sucesivas de 2. *Véase* bit.
Binding *(ligadura)*	
Bit	Dígito binario que puede tomar dos valores posibles: 0 y 1. Los bits son elementos básicos de construcción de programas y datos
Bloque *(block)*	Sentencias y declaraciones encerradas entre una pareja de llaves (apertura y cierre, ´{´ y ´}´. Por ejemplo, un *cuerpo*

	de una clase, es un bloque, al igual que el *cuerpo de un método*, Un bloque delimita un nivel de ámbito.
***Bolean** (bolean, lógico)*	Tipos primitivos de datos en Java. El tipo bolean puede tomar sólo dos valores: `true` *(verdadero)* y `false` *(falso)*.
***Bytecode** (códigos de byte)*	Resultado de la compilación del código fuente Java. La JVM (Java Virtual Machine) interpreta los *bytecodes* con la finalidad de ejecutar un programa Java. El *bytecode* es independiente de la máquina y se puede ejecutar en cualquier máquina que tenga un entorno de ejecución. Los *bytecodes* se almacenan en archivos `class`
Cabecera de la clase *(class header)*	Cabecera de la definición de la clase. La cabecera proporciona un nombre a la clase y define sus accesos. También describe si es una clase ampliada `(extends)` de una superclase o implementa interfaces `(implements)`
Clase *(clase)*	Colección encapsulada de datos y operaciones que actúan sobre los datos. El concepto de clase es fundamental en programación orientada a objetos. Una clase consta demétodos y datos. Los métodos de una clase definen el conjunto de operaciones permitidas sobre los datos de una clase (sus atributos). Una clase puede tener muchas instancia de la clase u objetos.
Clase abstracta *(abstract class)*	Superclase que contiene características comunes compartidas por las subclases. Se declaran utilizando la palabra reservada `abstract`. Las clases abstractas pueden contener datos y métodos, pero no se pueden *instanciar (*crear objetos); es decir, no se pueden crear objetos de esta clase.
Clase cliente *(client class)*	Clase que hace uso de otra clase.
Clase concreta *(concrete class)*	Una clase diseñada para crear (tener) instancias de objetos
Clase hija *(child class)*	*Véase* **subclase.**
Clase interna *(inner class)*	Una clase interna es una clase empotrada en otra clase. Las clases internas permiten definir pequeños objetos auxiliares y unidades de comportamiento que hacen a los programas más simples y concisos.
clase interna (*inner class***)**	Término utilizado para describir una clase declarada dentro de otra declaración de clases.
Clase miembro *(member class)*	Término general utilizado para describir una clase declarada dentro de otra declaración de clases.
Cohesivo *(cohesive)*	Modo de describir una clase que tiene partes fuertemente integradas, cada una de las cuales contribuye a describir las mismas abstracciones.
Cometario *(comment)*	Trozo de texto que tienen como objetivo documentar el programa y mostrar como se ha construido. Los comentarios no son sentencias de programación y son ignorados por el compilador. En Java los comentarios están precedidos por dos barras `(//)` en una línea o encerrados

	entre `/+` y `*/` en múltiples líneas.
Compilación (compilation)	Proceso de traducción de un lenguaje de programación. Normalmente este proceso implica la traducción de un *lenguaje de programación de alto nivel* a *lenguaje de programación de bajo nivel,* o el formato binario de un *conjunto de instrucciones específicas.* La traducción dse realiza con un programa denominado *compilador.* Un compilador java traduce los programas en *bytecodes.*
Compilación *(compiling)*	Nombre dado al proceso de traducción del código fuente a *bytecodes.*
Compilador *(compiler)*	Programa de software que realiza un proceso de compilación (traducción del lenguaje fuente a lenguaje máquina) de un programa escrito en un lenguaje de programación de alto nivel. En el caso de Java, es un programa que traduce el código fuente Java en *bytecode.* El compilador de J2SDK se denomina `javac`.
Compilador en tiempo de ejecución (*inst-in.time compiler)*	Compilador capaz de compilar cada b*ytecode* de una vez, y a continuación se reinicia al código compilado repetidamente cuando se ejecuta el *bytecode.*
Constante *(constant)*	Una variable declarada en final en Java. Una constante de la clase normalmente está compartida por todos los objetos de la misma clase; por consiguiente, una constante de clase se declara normalmente como `static`. Una constante local es una constante declarada dentro de un método.
C**onstante de la clase** *(class constant)*	Variable definida como `final` y `static`.
Constructor *(constructor)*	Método especial utilizado para inicializar el estado de un nuevo objeto. El constructor permite crear objetos utilizando el operador `new`. El constructor tiene exactamente el mismo nombre que la clase que lo contiene. Los constructores se pueden sobrecargar con el objetivo de facilitar la construcción de objetos con diferentes tipos de valores iniciales.
Constructor por defecto *(default constructor)*	Constructor que no tiene parámetros y sirve para inicalizar un objeto
Contenedor *(container)*	Clase que implementa una estructura de datos que contiene una colección de objetos. Se utiliza también para representar un componente IGU , Interfaz Gráfica de Usuario (GUI; Graphical User Interface) que contiene una colección de otros componentes IGU
Cuerpo de la clase *(class body)*	Cuerpo de una definición de una clase que agrupa las definiciones de los miembros de la clase: *campos, métodos* y *clases anidadas.*
Declaración *(declaration)*	Define las variables, métodos yc lasees en un programa.
Definición *(defnition)*	Término sinónimo de **declaración,** aunque en el proceso de escritura de un programa se suele diferenciar
Depuración *(debugging)*	Proceso de encontrar, fijar y eliminar errores en un programa. Para estas tareas se suele utilizar una herramienta de programación conocida como *depurador.*

Depurador *(debugger)*	Herramienta para ayudar a la localización de errores de un programa: `jdbc` se proporciona como parte del J2SDK. Un depurador puede establecer puntos de interrupción *(breakpoint),* parada simple a través de un programa e inspecciona el estado de las variables.
Diagrama de clases *(class diagram).*	Una representación gráfica construida utilizando una notación formal para visualizar y documentar las relaciones entre clases de un sistema.
Diseño *(diseño)*	Actividad de definir como se debe estructurar e implementar un programa.
Encapsulamiento, encapsulación *(encapsulation)*	Localización y protección de las características internas y estructura de un objeto. Combinación de métodos y datos en una única estructura de datos. En Java se conoce como **clase**
Entero *(integer)*	Un número completo (no es un número real con coma decimal) tal como –5, 1, 10 y 2002. Los enteros se pueden representar en Java de dos formas: utilizando el tipo primitivo `int` o utilizando una instancia de una clase `integer`.
Excepción *(exception)*	Un suceso (evento) no previsto que indica que un programa ha fallado en alguna forma. Las excepciones se representan por objetos excepción en java. Las excepciones se manejan con un bloque de sentencias `try/catch`.
Expresión *(expresión)*	Una subparte de una sentencia que representa un valor. Por ejemplo, la expresión aritmética ´2+5´ representa el valor 7. En Java, cualquier construcción sintáctica legal que represente un valor es una expresión.
Expresión booleana , lógica (*Bolean expresión*)	Una expresión cuyo resultado es del tipo lógico (bolean, bol), Operadores tales como && y || toman operandos lógicos y producen un resultado lógico. Los operadores relacionales toman operandos de tipos diferentes y producen un resultado lógico.
Final *(final)*	Modificador de clases, datos, métodos y variables locales. Una clase final no se puede extender, un dato final o variable local es una constante y un método final no sepuede anular (sustituir) en una subclase.
Formal parameter *(parámetro formal)*	Parámetros definidos en la signatura o declaración del método.
Fuente del suceso *(event source)*	El objeto que genera el suceso.
Función *(function)*	Construcción matemática a la que se pueden aplicar valores y que devuelve un resultado.
Herencia *(inheritance)*	Una relación entre clases en que una subclase se extiende desde una superclase.
HTML (Hypertext Markup Language)	Lenguaje de *´script´* o de marcas para diseñar páginas Web para creación y compartición de documentos electrónicos integrados preparados para multimedia e Internet.
I2SK	El Java 2 Software Kit distribuido por Inn proporciona el conjunto de herramientas para escribir programas Java,

	contiene las bibliotecas de calse Java, el compilador Java (`javac`) y una colección de otras utilidades. Las versiones se numeran en secuencia con 1.2,1.3,1.4(la más reciente dentro de la implementación de la plataforma Java 2.
IDE (*integrated development)*	Software para ayudar a lso programadores a escribir código eficientemente.
Identificador *(identifier)*	Nombre de una variable, método, clase, interfaz o paquete.
IGU, Interfaz Gráfica de Usuario *(GUI, Graphical User Interface)*	Una interfaz es un programa que se implementa utilizando componentes AWT tales como cuadros, botones, etiquetas, campos de texto, etc.
Implementación *(implementation)*	La actividad de escribir, compilar, probar y depurar el código de un programa.
Instancia *(instance)*	Objeto de una clase
Instanciación *(instantiation)*	Proceso de creación de un objeto de una clase.
Instanciación (*instantion)*	Proceso de crear un objeto de una clase.
Interfaz (*interface)*	Una interfaz se trata como una clase especial de Java. Cada interface se compila en un archivo independiente de bytecode, tal como una clase ordinaria. No se puede crear un instancia de la interfaz. La estructura de una interfaz Java es similar al de una clase abstracta en la que se puede tener datos y métodos. Los datos ,sin embargo ,deben ser constantes y los métodos pueden tener sólo declaraciones sin implementación. En Java existe sólo herencia simple y una clase puede heredar de una supereclase. Esta restricción se puede superar por el uso de una interfaz.
Interprete *(Interpreter)*	Software que interpreta y ejecuta *bytecode* de Java. La máquina virtual Java (JVM) es un interprete de bytecodes de Java que proporciona una emulación de software de un procesador de máuina.
JDK (*Java development kit) vease J25KJ*	Define el APJ de Java y contiene un conjunto de utilidades de líneas de órdenes tales como `Javac` (compilador) y `Java` (interprete).
Jerarquía de clases *(class hierarchy)*	Colección de clases organizadas en términos de relaciones de superclases y subclases.
JVM, Máquina Virtual Java *(Java Virtual Machine)*	Una emulación de software de una máquina que puede ejecutar *bytecodes* de Java. Proporciona una implementación del procesador, sistema dememoria e interfaces a dispositivos hardware. Todos los programs Java se compilan a *bytecodes* que se ejecutan por una JVM.
Ligadura dinámica *(dynamic binding)*	Ligadura o enlace del nombre de un método al cuerpo de dicho método que se ejecuta mientras que un programa se está ejecutando , al contrario del enlace que se produce cuando se compila el programa.
Llamada por referencia *(call-by-reference)*	Término utilizado cuando yuna referencia de un objeto se pasa como un parámetro de un método. La referencia se copia (llamada por valor) pero no el objeto refereciado
Llamada por valor *(call-*	Paso de un *argumento* a un método en el que una *copia* del

by.value)	valor del *argumento real* se toma y se sitúa en una posición de memoria independiente, representada por el correspondiente *argumento formal*. Todos los parámetros se pasan en Java por valor, pero hay otros lenguajes de programación que proporcionan también el método de paso por referencia.
Manejador de sucesos *(event handler)*	Un método en el que el objeto "oyente" se ha diseñado para hacer algún proceso especificado cuando ocurre un suceso determinado.
Marco de trabajo *(framework)*	
Mensaje *(message)*	Una petición enviada a un objeto que solicita ejecutar una operación determinada. El mensaje incluye un nombre y una lista opcional de parámetros.
Método abstracto (*abstract method)*	Método que sólo tiene signatura y no tiene cuerpo, y debe estar contenido dentro de una clase abstracta. Su implementación se realiza en la subclase. Se repreenta mediante el modificador `abstract`. Los métodos abstractos deben implementarse en una subclase no abstracta incluso aunque no se utilicen.
Método de la clase *(class method)*	Sinónimo de método estático. Un método que se puede invocar sin crear una instancia de la clase. Para definir métodos de clases, se ha de poner un modificador `static` en la declaración del método.
Método de la instancia (*Instance method)*	Un método (o procedimiento)declarado por un clase que se llama por sus objetos de instancias (o los de las subclases).
Moldeado (*casting, conversión)*	Proceso de convertír un valor de un tipo de dato primitivo en otro tipo primitivo o conversión de un objeto de un tipo de dato en otro tipo de objeto. Por ejemplo, `(int) 4.5` convierte `4.5` en un valore entero y `(cuadrado)c` convierte un objeto `c` en uno de tipo `cuadrado`
Moldear *(cast,, convertir)*	Cambiar explícitamente el tipo de una expresión utilizando una expresión de conversión (*cast).*
Objeto instancia (*instance object)*	Un objeto instancia es un representación de un valor del tipo implementado por su clase. La clase declara un objeto de variables, instancia que forman la estructura de un objeto y un conjunto de métodos que se pueden llamar en un objeto.
Ocultación de la información *(information hiding)*	Un concepto de ingeniería de software que se refiere a la ocultación y protección de las características internas y la estructura de un objeto.
Oyente de sucesos *(event listener)*	El objeto que recibie y maneja el suceso.
Palabra clave, reservada *(keyword)*	En Java, una palabra clave (o palabra reservada) es una palabra definida como parte del lenguaje de programación, Un nombre de palabra reservada no se puede utilizar para ningún otro propósito.
Palabra reservada,	Palabra definida como parte del lenguaje Java /(*vease en*

palabra clave (keyword)	*Apéndice A ,la lista de palabras reservadas Java).*
Parámetro actual o real *(actual parameter)*	Valor que se pasa a un método cuando se invoca ese método. Los parámetros reales (actuales) deben concordar en tipo, orden y número con los parámetros formales. Cuando se invoca a un método, los valores de los *argumentos actuales* se copian en los correspondientes argumentos formales.
Parámetro formal *(formal parameter)*	Declaración de una variable parámetro en una lista de parámetros de un método.
Plataforma de Java 2. (*Java 2 Plataform)*	Nombre de la versión más reciente de Java.
Programación controlada por sucesos *(event-drive programming)*	La programación de gráficos en Java está controlada por sucesos. En programación controlada por sucesos (o enventos) los códigos se ejecutan por activación de sucesos, tales como pulsar un botón o mover el ratón
Programación imperativa *(imperative programming)*	Programación basada en los principios de instrucción o secuencias de órdenes, selección, repetición, variables y asignación. También se conoce a esta programación como *procedimental* o *por procedimientos*. Java es un lenguaje imperativo.
Recolección de basura *(garbage collection)*	
Sentencia compuesta (*compound statement)*	Sentencia contenedora que consta de una secuencia de otras sentencias y declaraciones. En Java se utilizan llaves (`{ y }`) para delimitar una sentencia compuesta.
Suceso *(event)*	Un tipo de señal que indica ha ocurrido alguna acción. Normalmente se asocia con sucesos de entrada de interfaces gráficas de usuario (p.e. el "clic" de un ratón, pulsación de una tecla, etc.) El programa puede responder o ignorar el suceso. *Véase* **evento.**
Tipo abstracto de datos , TAD (ADT, *Abstract Data Type)*	Especificación formal de un tipo de dato que consta de un nombre, un conjunto de operaciones y una descripción algebraica del comportamiento de las operaciones.
Tipo de datos *(data type)*	Los tipos de datos se utilizan para definir variables. Java soporta los tipos de datos primitivos y tipos de datos objeto.
Tipo de datos *(data type)*	Tipo de dato que se utiliza para definir variables. Java soporta tipos primitivos de datos y tipos de datos objeto.
Variable de clase *(class variable)*	Sinónimo de variable estática.
Variable de instancia *(instance variable)*	Una variable declarada en una clase. Un miembro dato no estático de una clase. Una copia de un método de una instancia existe en cada instancia de la clase que se crea.
Variable local *(local variable)*	Variable definida en el interior de una definición de un método.

Clase Principal *(main class)*	Una clase que contiene un método principal (`main`)**.**
Mensaje (*message)*	Petición enviada a un objeto que solicita realizar una operación con nombre. El mensaje incluye un nombre y

	una lista opcional de parámetros.
Método (*method)*	Una colección de sentencias que se agrupan juntos para ejecutar una operación.
Method object	
Sobrecarga de un método (*method overloading)*	La sobrecarga de n método significa que se puede definir los métodos con el mismo nombre de una clase siempre que haya diferencia en sus parámetros.
Nented class (*Nented class)*	Una clase estática declarad dentro de otra clase. Denominada también una clase anidad de nivel superior.
Anulación de métodos (*method overriding)*	La anulación o sustitución de métodos significa que se puede modificar el método de una subclase que está definida originalmente en una superclase.
Modificador (*Modifer)*	Una palabra reservada en Java que especifica las propiedades de los datos, métodos y clases, y como se pueden utilizar. Ejemplos de modificaciones son `public,private y static.`
Multihilo *(mulithreading)*	Propiedad de un programa para ejecutar diversas tareas simultáneamente dentro de un programa**.**
Red (*network)*	Infraestructura que permite a los ordenadores comunicarse unos con otros.
En red *(networking)*	Propiedad de los ordenadores y programas de ordenador que las permiten comunicarse unos con otros a través de una red.
Objeto *(object)*	*Vease* instancia. Una instancia de una *clase* específica. En general, se puede construir cualquier número de objetos a partir de una clase.
Análisis orientado a objetos OOA *(objetct-oriented Analysis)*	Análisis realizado en términos de objetos, clase y relaciones de clases.
Diseño orientado a objetos OOD (*object.oriented design)*	Diseño realizado en términos de objetos, clases y selecciones de clases.
Operador *(operator)*	Operaciones para valores de tipos primitivos de datos. Ejemplos de operadores son +,-,*,/ y %
Programación orientada a objetos OOP *(object-oriented programming)*	Un enfoque de programación que implica organización de objetos y sus comportamiento en clases de componentes realizables.
Asociatividad de operadores *(operator associativity)*	Orden en que se evalúan operadores de igual procedencia dentro de una expresión. La asociatividad a izquierda produce una evaluación de izquierda a derecha, la asociatividad derecha es derecha a izquierda.
Precedencia de operadores *(operator precedence)*	Prioridad de un operador dentro de una expresión utilizando para determinar en que orden de evaluarán los operadores.
Sobrecarga (*overload)*	Proporciona dos o más métodos con el mismo nombre en el mismo ámbito ,diferenciado por tener listas de parámetros deferentes.
Anular o sustituir *(override)*	Donde un método de una subclase redefine y especializa un método del mismo tipo heredado de una superclase.

Paquete *(package)*	Colección de clases agrupadas juntas.
Parámetro *(parameter)*	Los parámetros formales se especifican en una declaración de un método en una llamada a un método,
Paso por referencia *(pass-by-refernce)*	Un término utilizado cuando una referencia de un objeto se pasa como un parámetro de un método. Cualquier cambio al objeto local que ocurre dentro del cuerpo del método afectará el objeto original que se pasará como argumento.
Paso por valor *(pass-by-value)*	Un término utilizado cuando una copia de una variable de un tipo primitivo de dato se pasa a un parámetro de un método. La variable real externa al método no está afectado, con independencia de los cambios hechos al parámetro formal dentro del método.
Lista de parámetros *(parameter list)*	Lista de valores dados a un método para inicializar sus parámetros o la lista de parámetros de las declaraciones de variables.
Variable parámetro *(parameter variable)*	Variable declarad en una lista de parámetros formados de un método y que se inicializa cuando se llama al método. Los bloques `catch` también utilizan variables parámetro.
Clase padre *(parent class)*	Igual concepto que superclase
Patrón *(pattern)*	Disposición avanzada de clases y objetos
Precedencia, prioridad *(precedence)*	Prioridad de un operador en una expresión utilizada para determinar el orden en que se evalúan los operadores.
Tipo primitivo (*primitive type)*	Un tipo definido como parte del lenguaje Java en vez del declarado por una clase o intefaz. Se denomina también tipos incorporados.
Privado *(private)*	Un modificador de miembros de una clase, un miembro privado sólo puede ser referenciado en el interior de la clase.
Programa *(program)*	Un conjunto de instrucciones (o sentencias) que describen alguna aplicación o actividad ejecutada en una computadora.
Tipo de dato primitivo *(primitive data type)*	Los tipos de datos primitivos son `byte, short, int, long, float,double,bolean y chaw.`
Programador *(progammer)*	Personas que diseña, escribe, prueba y depura programas.
Lenguaje de programación (programming language)	Notación utilizada por los programadores para escribir programas . un lenguaje tiene una sintaxis (las palabras y símbolos utilizadas para escribir códigos de programa), una gramática (las reglas que definen una secuencia de palabras y símbolos significativos y correctos) y semántica. Java es un lenguaje de programación.
Protegido *(protected)*	Un modificador para los miembros de una clase. Un miembro protegido de una clase que puede ser utilizado en la clase que está declarad o cualquier subclase derivada de esa clase.
Público *(public)*	Un modificador de clases, datos y métodos a los que se puede acceder por todos los programas.
Palabra reservada *(reserved word)*	Véase **palabra clave.**

Ejecutar, ejecución *(run)*	Hacer funcionar un programa instrucción a instrucción.
Escenario *(scenario)*	Descripción o conjunto de secuencias de sucesos que se utilizan para describir parte del comportamiento de un programa.
Semántica *(semantics)*	Conjunto de reglas que definen el significado de un programa sintácticamente válido. Java toma un enfoque operacional en semántica de modo que el comportamiento y por consiguiente el significado de un programa se define por la máquina sobre la que esta ejecutando el programa.
Signatura *(signature)*	
Socket `(socket)`	Término que describe la facilitación de comunicación entre un servidor y un cliente.
Inferencia de software *(software engineering)*	Conjunto de etapas en la realización de un programa. Estas etapas suelen ser de análisis, diseño implementación , pruebas, entregas y mantenimiento.
Código fuente *(source code)*	Texto de un programa antes de ser complilado. El texto se crea y edita utilizando en editor ordinario y contiene caracteres normales, legibles. El código fuente ser utiliza para las personas para describir programas y sus componentes han de ser lo más legibles y comprensibles posibles.
Software engineering	
Source code	
Source text	
Specification	
Lenguaje de consulta *(DQL stanadr query langage)*	Lenguaje de computadora para realizar consultas y actualizaciones en una base de datos.
Sentencia *(statement)*	Una unidad de código que representa una acción o una secuencia de acciones. Las sentencias se ejecutan en el orden en que están escritas y siempre terminan en un punto y coma.
Ligadura estática *(static briding)*	Enlace o conexión de un nombre de un método a un cuerpo del método ejecutados por el compilador mediante el análisis léxico del texto de un programa,
Método estático *(static method)*	Véase **método de una clase.** Método declarado en una clave que se llama directamente sin necesidad de que el objeto sea llamado.
Variable estática *(static variable)*	Véase **variable de clase.**
Flujo ***(Stream)***	Término que describe el flujo de datos continuo de una dirección entre un emisor y un receptor.
Subclase *(subclass)*	Una clase que hereda o se extiende de una superclase.
Superclase ***(superclass)***	Una clase que puede ser heredada de otra clase.
Subtipo *(subtype)*	Un tipo que hereda o se extiende de un supertipo.
Superclase *(superclass)*	Una clase que es heredada por una subclase.
Supertipo *(supertype)*	Un tipo que es heredado por un subtipo.
Sintaxis (*Syntax)*	Un conjunto de reglas que especifica la composición de

	programas a partir de palabras reservadas, símbolos y caracteres. La sintaxis define la estructura de los programas legales en términos de cómo las palabras reservadas y otros caracteres se pueden escribir y en qué orden.
Etiqueta *(tag)*	Una instrucción HTML que indica a un navegador Web como visualizar un documento. Las etiquetas se encierran entre corchetes tales como `<html>`, `<i>`, `<b>`, y `</html>`.
Prueba/ probar *(test)*	En términos de programación, la actividad de verificación sistemática de que un programa funciona correctamente.
Prueba *(testing)*	Véase **prueba**
Hilo *(thread)*	Un flujo de ejecución de una tarea que tiene un principio y un fin, en un programa.
UML *(UML)*	Lenguaje unificado de modelado que proporciona notación estándar visual para documentar el análisis y diseño de sistemas orientados a objetos.
Unicode *(unicode)*	Un sistema de codificación de caracteres internacionales gestionados por el consorcio Unicode, Java soporta Unicode.