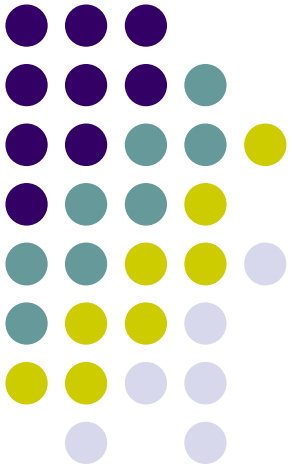
INGENIERÍA EN SISTEMAS Y COMUNICACIONES

UDA: PROGRAMACIÓN ORIENTADA A OBJETOS

TEMA: FUNDAMENTOS DEL LENGUAJE JAVA



ELABORÓ: DR. EN C. HÉCTOR RAFAEL OROZCO AGUIRRE
CU UAEM VM

Programa de Estudios Por Competencias
Programación Orientada a Objetos

IDENTIFICACIÓN DEL CURSO

ISMO ACADÉMICO:
Universitario UEAM Valle de México

ma Educativo: ía en Sistemas y Comunicaciones			**Área de docencia:** Ciencias de la ingeniería						
ción por los H.H. os Académico y de no		**Fecha:** Julio de 2006	**Programa elaborado por:** Morales Escobar Saturnino Job y Rodríguez Pérez Ivonne **Actualizado por:** Orozco Aguirre Héctor Rafael					**Fecha de elaboraci** Julio de 2006 **Fecha de actualiza** 26/Junio/2012	
ve	**Horas de teoría**	**Horas de práctica**	**Total de horas**	**Créditos**	**Tipo de Unidad de Aprendizaje**	**Carácter de la Unidad de Aprendizaje**	**Núcleo de formación**	**Mod**	
	2	4	6	8	Obligatoria	Curso	Básico	Preser	
uisitos entos de Programación y Algoritmos y ras de Datos			**Unidad de Aprendizaje Antecedente** Algoritmos y Estructuras de Datos			**Unidad de Aprendizaje Consecue** Programación Avanzada			

mas educativos en los que se imparte: Ingeniería en Sistemas y Comunicaciones

Historia

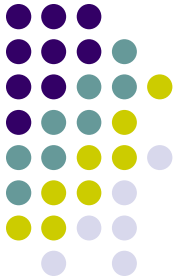
Sun Microsystems, líder en servidores para Internet, uno de cuyos lemas desde hace mucho tiempo es *"the network is the computer"* (lo que quiere dar a entender es que la verdadera computadora es la red en su conjunto y no cada máquina individual), es quien ha desarrollado el lenguaje Java, en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

Historia

Sun Microsystems decidió intentar introducirse en el mercado de la electrónica de consumo y desarrollar programas para pequeños dispositivos electrónicos. Tras unos comienzos dudosos, Sun decidió crear una filial, denominada FirstPerson Inc., para dar margen de maniobra al equipo responsable del proyecto.

Inicialmente Java se llamó Oak (roble en inglés), aunque tuvo que cambiar de denominación, debido a que dicho nombre ya estaba registrado por otra empresa.

Historia



Tres de las principales razones que llevaron a crear Java son:

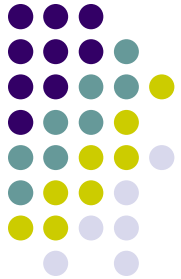
1. Creciente necesidad de interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban hasta el momento.
2. Fiabilidad del código y facilidad de desarrollo. Gosling observó que muchas de las características que ofrecían C o C++ aumentaban de forma alarmante el gran coste de pruebas y depuración. Por ello en los sus ratos libres creó un lenguaje de programación donde intentaba solucionar los fallos que encontraba en C++.

Historia

3. Enorme diversidad de controladores electrónicos. Los dispositivos electrónicos se controlan mediante la utilización de microprocesadores de bajo precio y reducidas prestaciones, que varían cada poco tiempo y que utilizan diversos conjuntos de instrucciones. Java permite escribir un código común para todos los dispositivos.

Por todo ello, en lugar de tratar únicamente de optimizar las técnicas de desarrollo y dar por sentada la utilización de C o C++, el equipo de Gosling se planteó que tal vez los lenguajes existentes eran demasiado complicados como para conseguir reducir de forma apreciable la complejidad de desarrollo asociada a ese campo.

Historia



El proyecto Green fue el primero en el que se aplicó Java, y consistía en un sistema de control completo de los aparatos electrónicos y el entorno de un hogar. Con este fin se construyó una computadora experimental denominada *7 (Star Seven). Más tarde Java se aplicó a otro proyecto denominado VOD (Video On Demand) en el que se empleaba como interfaz para la televisión interactiva

Historia

Finalmente Bill Joy, cofundador de Sun y uno de los desarrolladores principales del Unix de Berkeley, juzgó que Internet podría llegar a ser el campo de aplicación adecuado para disputar a Microsoft su primacía casi absoluta en el terreno del software, y vio en Oak el instrumento idóneo para llevar a cabo estos planes. Tras un cambio de nombre y modificaciones de diseño, el lenguaje Java fue presentado en sociedad en agosto de 1995.

Características

Java ha dado un significativo avance en el mundo del software, y esto viene avalado por tres elementos claves que lo diferencian desde un punto de vista tecnológico:

1. Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
2. Proporciona un conjunto de clases potente y flexible.
3. Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web (aplicaciones denominadas *applets*). aportando a la Web una interactividad que se había buscado durante mucho tiempo entre usuario y aplicación.

Características

Las principales características que nos ofrece Java respecto a cualquier otro lenguaje de programación, son:

- **Simple:** ofrece toda la funcionalidad de un lenguaje potente y es de fácil aprendizaje (requerimiento para aprender Java es tener una comprensión de los conceptos básicos de la programación orientada a objetos), añade características muy útiles cómo el recolector de basura (reciclador de memoria automático).

Características

- **Orientado a objetos:** Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java incorpora funcionalidades inexistentes en C++ como por ejemplo, la resolución dinámica de métodos mediante una interfaz específica llamada RTTI (RunTime Type Identification) que define la interacción entre objetos.

- **Distribuido:** Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con diversos protocolos que permiten a los programadores acceder a la información a través de la red con tanta facilidad como a los archivos locales. La verdad es que Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando entre sí.

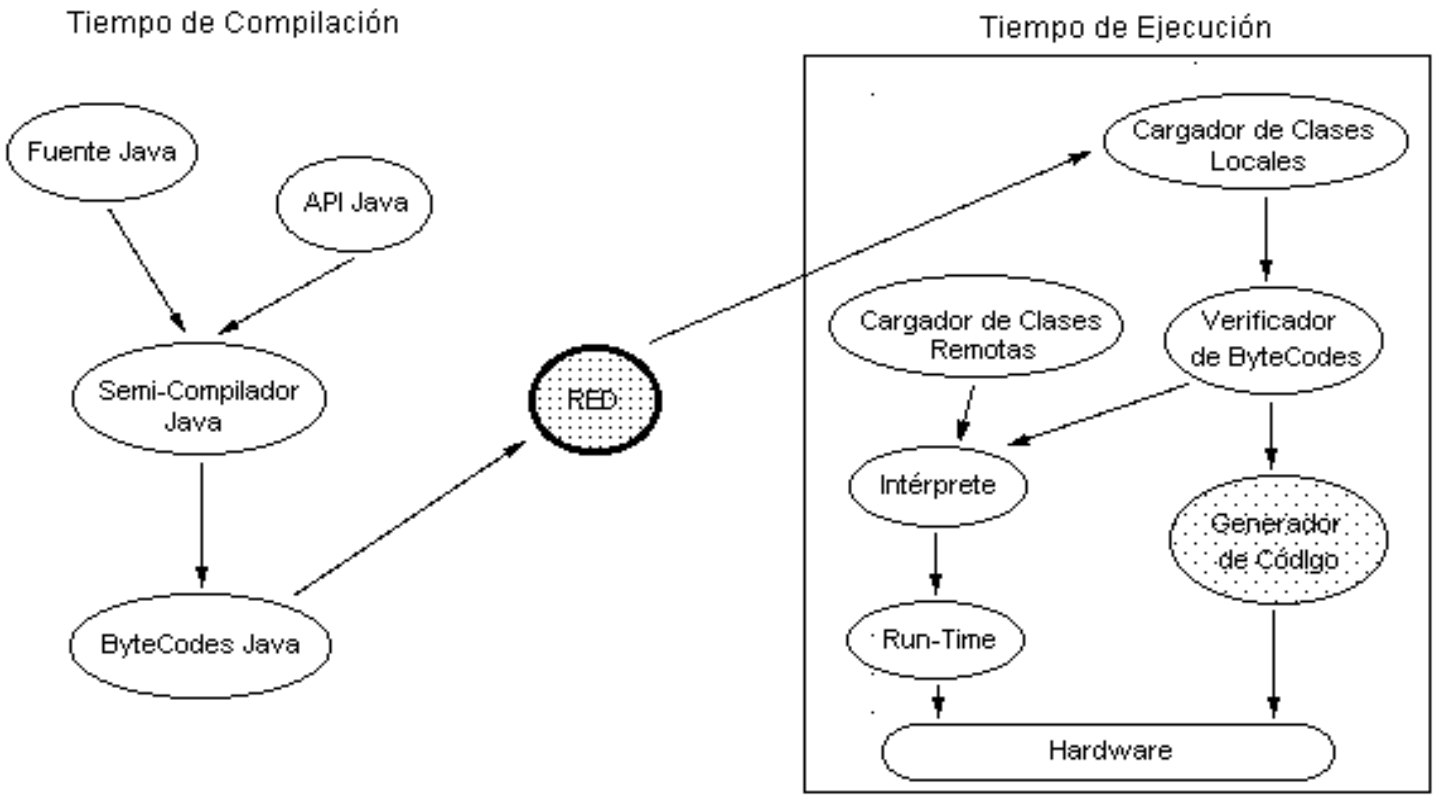
Características

- **Robusto:** Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de la misma. Así proporciona entre otras, comprobación de límites de arreglos, manejo de excepciones y verificación de ByteCodes.

Características

- **Arquitectura Neutral:** Java está diseñado para que un programa escrito en este lenguaje sea ejecutado correctamente independientemente de la plataforma en la que se esté corriendo. Para establecer Java como parte integral de la red, el compilador Java compila su código a un archivo objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (RunTime) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

Características



Máquina hipotética que es implementada por un sistema RunTime, que sí es dependiente de la máquina.

Características

- **Seguro:** La seguridad en Java tiene dos facetas el uso del lenguaje y la memoria para que las aplicaciones sean extremadamente seguras. Los niveles de seguridad que presenta son:

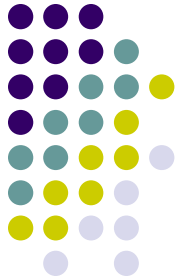
  - Fuertes restricciones al acceso a memoria, como son la eliminación de apuntadores aritméticos y de operadores ilegales de transmisión.
  - Rutina de verificación de los *ByteCode* que asegura que no se viole ninguna construcción del lenguaje.
  - Verificación del nombre de clase y de restricciones de acceso durante la carga.
  - Sistema de seguridad de la interfaz que refuerza las medidas de seguridad en muchos niveles.

Características

- **Portable:** Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos.

- **Interpretado:** Java es más lento que otros lenguajes de programación, como C++, ya que debe ser interpretado y no ejecutado como sucede en cualquier programa tradicional. Con el fin de crear aplicaciones multiplataforma, es necesario que exista el RunTime correspondiente al sistema operativo utilizado.

Características



- **Multihilo:** Java permite muchas actividades simultáneas en un programa. Los threads (a veces llamados, procesos ligeros), son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar los threads construidos en el lenguaje, son más fáciles de usar y más robustos que sus homólogos en C o C++. El beneficio de ser multihilo consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real.

- **Dinámico:** Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las librerías nuevas o actualizadas no paralizarán las aplicaciones actuales (siempre que mantengan el API anterior).

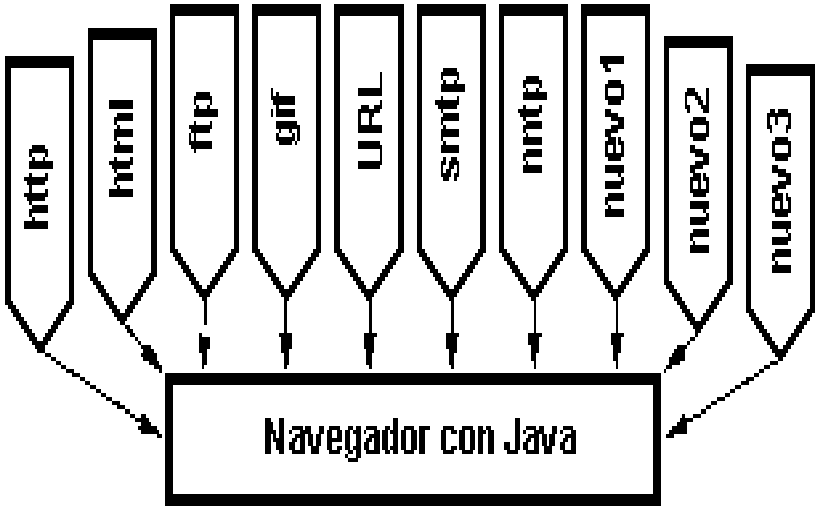
Características

Java también simplifica el uso de protocolos nuevos o actualizados. Si su sistema ejecuta una aplicación Java sobre la red y encuentra una pieza de la aplicación que no sabe manejar, tal como se ha explicado en párrafos anteriores, Java es capaz de traer automáticamente cualquiera de esas piezas que el sistema necesita para funcionar.

Java, para evitar que los módulos de ByteCodes o los objetos o nuevas clases, haya que estar trayéndolos de la red cada vez que se necesiten, implementa las opciones de persistencia, para que no se eliminen cuando se limpie la caché de la máquina.

Características

Monolito: cada pieza de código se compacta dentro del código del navegador



Sistema Federado: el navegador es un coordinador de piezas, y cada pieza es responsable de una función. Las piezas se pueden añadir dinámicamente a través de la red

Características

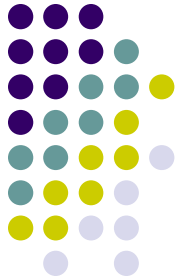
Ahora nos surgen dos grandes interrogantes respecto al uso del lenguaje Java:

1. ***¿Hay ventajas de todo esto?***
2. ***¿Se gana algo con Java?***

Características

- **Primero:** No se debe volver a escribir el código si quieres ejecutar el programa en otra máquina.
- **Segundo:** Java es un lenguaje de programación orientado a objetos, y tiene todos los beneficios que ofrece esta metodología de programación.
- **Tercero:** Un navegador compatible con Java deberá ejecutar cualquier programa hecho en Java, esto ahorra a los usuarios tener que estar insertando "plug-ins" y demás programas que a veces nos quitan tiempo y espacio en disco.
- **Cuarto:** Java es un lenguaje y por lo tanto puede hacer todas las cosas que puede hacer un lenguaje de programación.
- **Quinto:** Si lo que me interesa son las páginas de Web, se le pueden poner toda clase de elementos multimedia y permiten un alto nivel de interactividad, sin tener que gastar en paquetes carísimos de multimedia.

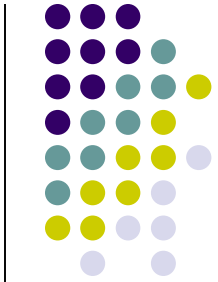
Fundamentos del Lenguaje



Java es un lenguaje orientado a objetos, que se deriva en alto grado de C++, de tal forma que puede ser considerado como un C++ nuevo y modernizado o bien como un C++ al que se le han amputado elementos heredados del lenguaje estructurado C.

Java = ((C++) – C)++

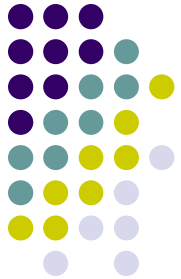
Fundamentos del Lenguaje



Tokens

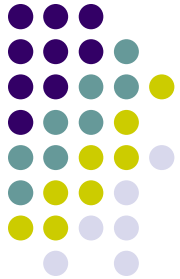
Un *token* es el elemento más pequeño de un programa que es significativo para el compilador. Estos *tokens* definen la estructura de Java.

Cuando se compila un programa Java, el compilador analiza el texto, reconoce y elimina los espacios en blanco y comentarios y extrae *tokens* individuales. Los *tokens* resultantes se compilan, traduciéndolos a código de byte Java, que es independiente del sistema e interpretable dentro de un entorno Java.



Los *tokens* Java pueden subdividirse en seis categorías:

1. Identificadores.
2. Palabras clave o reservadas.
3. Constantes y literales.
4. Operadores.
5. Separadores.
6. Comentarios.

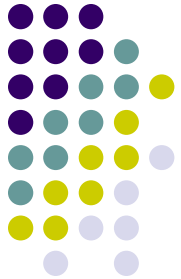


- **Identificadores:** Los identificadores son *tokens* que representan nombres asignables a variables, métodos y clases, para identificarlos de forma única ante el compilador y darles nombres con sentido para el programador.

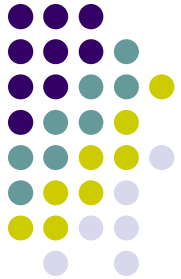
  En Java, un identificador comienza con una letra, un guión bajo (_) o un símbolo de pesos ($). Los siguientes caracteres pueden ser letras o dígitos. Se distinguen las mayúsculas de las minúsculas y no hay longitud máxima. Como nombres de identificadores no se pueden usar palabras claves de Java.

- **Palabras clave o reservadas:** son aquellos identificadores reservados por Java para un objetivo determinado y se usan sólo de la forma limitada y específica.

abstact	*boolean*	*break*	*byte*	*byvalue*
case	*cast*	*catch*	*char*	*class*
const	*continue*	*default*	*do*	*double*
else	*extends*	*false*	*final*	*finally*
float	*for*	*future*	*generic*	*goto*
if	*implements*	*import*	*inner*	*instanceof*
int	*interface*	*long*	*native*	*new*
null	*operator*	*outer*	*package*	*private*
protected	*public*	*rest*	*return*	*short*
static	*super*	*switch*	*syncroniced*	*this*
throw	*throws*	*transient*	*true*	*try*
var	*void*	*volatile*	*while*	



- **Constantes y literales:** Un valor constante en Java se crea utilizando una representación literal de él. Java utiliza cinco tipos de elementos: enteros, reales en coma flotante, booleanos, caracteres y cadenas, que se pueden poner en cualquier lugar del código fuente de Java. Cada uno de estos literales tiene un tipo correspondiente asociado con él.

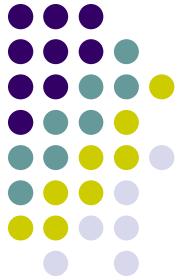


Enteros:

- byte 8 bits complemento a dos
- short 16 bits complemento a dos
- int 32 bits complemento a dos
- long 64 bits complemento a dos

Reales en coma flotante:

- float 32 bits IEEE 754
- double 64 bits IEEE 754



Booleanos:

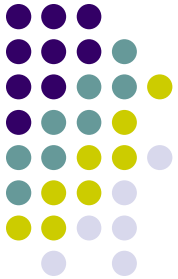
- true
- false

Caracteres:

- ‘caracter’

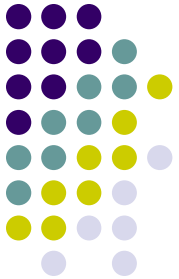
Cadenas:

- "cadena"



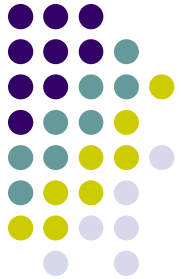
Descripción	Representación	Valor Unicode
Caracter Unicode	*\udddd*	
Numero octal	*\ddd*	
Barra invertida	\\	*\u005C*
Continuación	\	\
Retroceso	*\b*	*\u0008*
Retorno de carro	*\r*	*\u000D*
Alimentación de formularios	*\f*	*\u000C*
Tabulación horizontal	*\t*	*\u0009*
Línea nueva	*\n*	*\u000A*
Comillas simples	*\'*	*\u0027*
Comillas dobles	*\"*	*\u0022*
Números arábigos ASCII	*0-9*	*\u0030 a \u0039*
Alfabeto ASCII en mayúsculas	*A.-Z*	*\u0041 a \u005A*
Alfabeto ASCII en minúsculas	*a.-z*	*\u0061 a \u007A*

Caracteres especiales Java

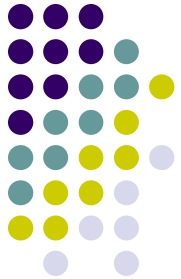


Conversiones sin pérdidas de información

Tipo Origen	**Tipo Destino**
byte	double, float, long, int, char, short
short	double, float, long, int
char	double, float, long, int
int	double, float, long
long	double, float
float	double



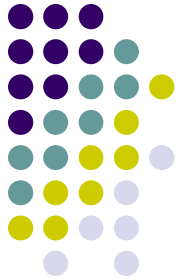
- **Operadores:** Un valor constante en Java se crea utilizando una representación literal de él. Java utiliza cinco tipos de elementos: enteros, reales en coma flotante, booleanos, caracteres y cadenas, que se pueden poner en cualquier lugar del código fuente de Java. Cada uno de estos literales tiene un tipo correspondiente asociado con él.



Operadores que se utilizan en Java, por orden de precedencia:

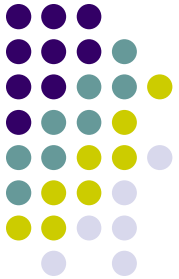
- . [] ()
- ++ --
- ! ~ instanceof
- * / %
- + -
- << >> >>>
- < > <= >= == !=
- & ^ |
- && ||
- ? : (expresion ? sentencia1 : sentencia2)
- = op= (*= /= %= += -= etc.)

Nota: Java no soporta la sobrecarga de operadores, excepto para concatenar cadenas



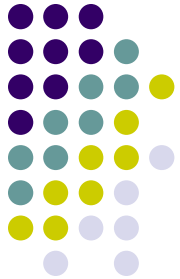
Operador	Uso	Descripción
+	*op1 + op2*	Suma op1 y op2
-	*op1 - op2*	Resta op2 de op1
*	*op1 * op2*	Multiplica op1 por op2
/	*op1 / op2*	Divide op1 por op2
%	*op1 % op2*	Calcula el resto de dividir op1 entre op2

Operadores aritméticos binarios



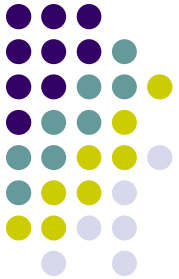
Operador	Uso	Descripción
+	+op	Convierte op a entero si es un byte, short o char
-	-op	Niega aritméticamente op

Versiones unarias de los operadores + y -



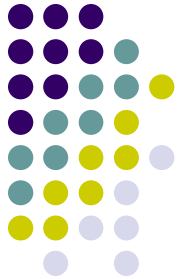
Operador	Uso	Descripción
++	*op*++	Incrementa op en 1; se evalúa al valor anterior al incremento
++	++*op*	Incrementa op en 1; se evalúa al valor posterior al incremento
--	*op*--	Decrementa op en 1; se evalúa al valor anterior al incremento
--	--*op*	Decrementa op en 1; se evalúa al valor posterior al incremento

Operaciones con ++ y --



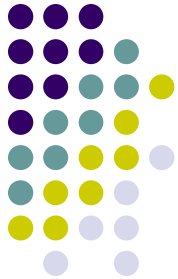
Operador	Uso	Devuelve verdadero si
>	op1 > op2	op1 es mayor que op2
>=	op1 >= op2	op1 es mayor o igual que op2
<	op1 < op2	op1 es menor que op2
<=	op1 <= op2	op1 es menor o igual que op2
==	op1 == op2	op1 y op2 son iguales
!=	op1 != op2	op1 y op2 son distintos

Operadores de comparación



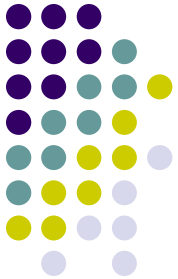
Operador	Uso	Devuelve verdadero si
&&	*op1 && op2*	op1 y op2 son ambos verdaderos, condicionalmente evalúa op2
&	*op1 & op2*	op1 y op2 son ambos verdaderos, siempre evalúa op1 y op2
||	*op1* || *op2*	op1 o op2 son verdaderos, condicionalmente evalúa op2
|	*op1* | *op2*	op1 o op2 son verdaderos, siempre evalúa op1 y op2
!	*! op*	op es falso

Operadores condicionales



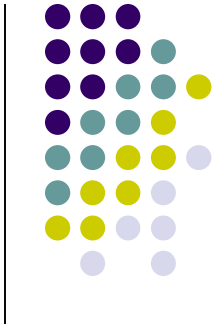
Operador	Uso	Operación
>>	*op1 >> op2*	Desplaza los bits de op1 a la derecha op2 veces
<<	*op1 << op2*	Desplaza los bits de op1 a la izquierda op2 veces
>>>	*op1 >>> op2*	Desplaza los bits de op1 a la derecha op2 veces (sin signo)

Operadores de desplazamiento de bits

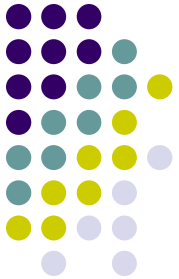


Operador	Uso	Operación
&	*op1 & op2*	AND
|	*op1 | op2*	OR
^	*op1 ^ op2*	OR Exclusivo
~	*~op1*	Complemento a uno

Operadores de lógica de bits

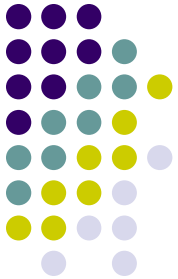


Operador	Uso	Operación
>>	Operando >> Despl	Desplaza bits del operando hacia la derecha las posiciones indicadas (con signo)
<<	Operando << Despl	Desplaza bits del operando hacia la izquierda las posiciones indicadas
>>>	Operando >>> Despl	Desplaza bits del operando hacia la derecha las posiciones indicadas (sin signo)
&	Operando & Operando	Realiza una operación AND lógiga entre los dos operandos
|	Operando | Operando	Realiza una operación OR lógica entre los dos operandos
^	Operando ^ Operando	Realiza una operación lógica OR Exclusiva entre los dos operandos
~	~Operando	Complementario del operando (unario)



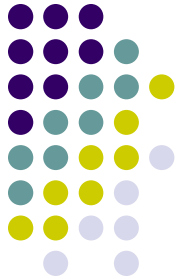
Operador	Uso	Equivalente a
+=	*op1 += op2*	op1 = op1 + op2
-=	*op1 -= op2*	op1 = op1 - op2
*=	*op1 *= op2*	op1 = op1 * op2
/=	*op1 /= op2*	op1 = op1 / op2
%=	*op1 %= op2*	op1 = op1 % op2
&=	*op1 &= op2*	op1 = op1 & op2

Operadores de atajo de asignación

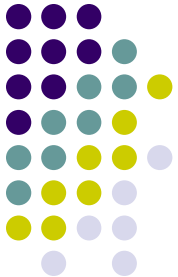


Tipo de operadores	Operadores de este tipo
Operadores posfijos	[ ] . (parametros) expr++ expr--
Operadores unarios	++expr --expr +expr -expr ~ !
Creación o conversión	new (tipo) expr
Multiplicación	* / %
Suma	+ -
Desplazamiento	<<
Comparación	< <= = instanceof
Igualdad	== !=
AND a nivel de bit	&
OR a nivel de bit	^
XOR a nivel de bit	|
AND lógico	&&
OR lógico	||
Condicional	? :
Asignación	= += -= *= /= %= &= ^= |= <<= = =

Precedencia de operadores

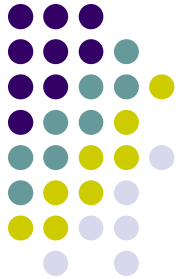


- **Separadores:** Un valor constante en Java se crea utilizando una representación literal de él. Java utiliza cinco tipos de elementos: enteros, reales en coma flotante, booleanos, caracteres y cadenas, que se pueden poner en cualquier lugar del código fuente de Java. Cada uno de estos literales tiene un tipo correspondiente asociado con él.



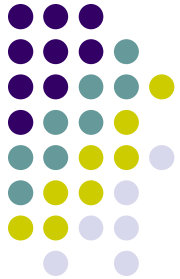
Los separadores admitidos en Java son:

- ***()*** *.-* Para contener listas de parámetros en la definición y llamada a métodos. También se utiliza para definir precedencia en expresiones, contener expresiones para control de flujo y rodear las conversiones de tipo.
- ***{}****.-* Para contener los valores de matrices inicializadas automáticamente. También se utiliza para definir un bloque de código, para clases, métodos y ámbitos locales.
- ***[]*** *.-* Para declarar tipos matriz. También se utiliza cuando se hace referencia a los valores de una matriz.
- ***;*** *.-* Separa sentencias.
- ***,*** *.-* Separa identificadores consecutivos en una declaración de variables. También se utiliza para encadenar sentencias dentro de una sentencia *for*.
- ***.*** *.-* Para separar nombres de paquete de subpaquetes y clases. También se utiliza para separar una variable o método de una variable de referencia.



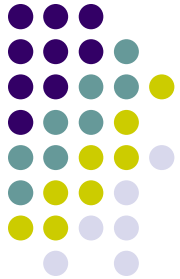
- **Comentarios:** En Java hay tres tipos de comentarios:
  - // comentario para una sola línea
  - /* comentario de una o más líneas */
  - /** comentario de documentación, de una o más líneas */

Control de Flujo



El lenguaje Java soporta las estructuras de control:

Sentencia	Clave
Toma de decisión	if-else, switch-case
Bucle	for, while, do-while
Misceláneo	break, continue, etiqueta:, return, goto
Excepciones	try-catch-throw

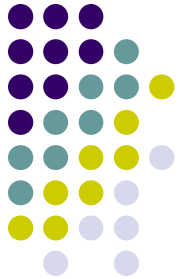


If
if-else
if else-if else

```
if(condición1)  {
    sentencias;
}
[else if (condición2) {
    sentencias;
}]
…
[else {
    sentencias;
}
]
```

switch

```
switch(expresión1) {
    case expresión2:
        sentencias;
    break;
    [case expresión3:
        sentencias;
    break;]
    …
    [default:
        sentencias;
    break;]
}
```



for

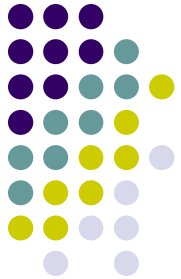
```
for (inicialización; terminación; iteración) {
    sentencias;
}
```

while

```
[inicialización;]
while(condición) {
    sentencias;
    [iteración;]
}
```

do-while

```
[inicialización;]
Do {
    sentencias;
    [iteración;]
 }while(condición);
```



break

break [etiqueta];

continue

continue [etiqueta];

return

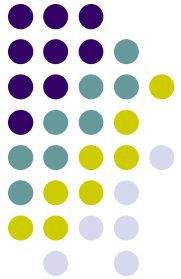
return expresión;

etiqueta

etiqueta: sentencia;

```
uno: for(..) {
    dos: for(..) {
        continue; // seguirá en el bucle interno
        continue uno; // seguirá en el bucle principal
        break uno; // saldrá del bucle principal
    }
}
```

Nota: se debe de evitar el uso de etiquetas en lo posible.



try-catch-finally

```
try {
    sentencias;
} catch (excepción variable1) {
    sentencias;
  }
  [catch(excepción variable2) {
    sentencias
  }]
  …
  [finally {
    sentencias;;
  }]
```

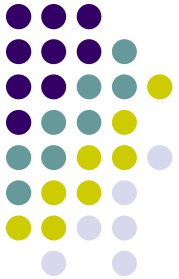
throw

throw excepción;

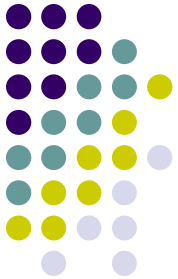
throws

throws excepción1, excepción2, …

Referencias

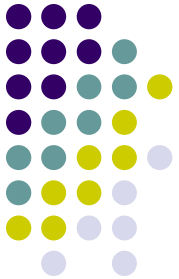


- Deitel Harvey & Deitel Paul, Como Programar en JAVA, 5a edición. Pearson/Prentice Hall, 2008.
- Booch Grady, Rumbaugh James & Jacobson Ivar, UML El Lenguaje Unificado de Modelado (Guía de usuario), Pearson, 2006.
- Luis Joyanes Aguilar, Programación en JAVA 2, 1ª edición. Mc Graw Hill.
- Wang Paul S. Java con Programación Orientada a Objetos y aplicaciones en la WWW.1ª Edición. Thomson Editores.
- Froufe Agustín. JAVA 2 Manual de Usuario y Tutorial. 2ª edición. Alfaomega RA-MA.
- Larman Craig, UML y Patrones (Introducción al análisis y diseño orientado a objetos), Pearson, 2003.
- Herbert Schildt, Java: The Complete Reference, McGraw-Hill Osborne Media, 8 edition, June 22, 2011.
- Wu C. Thomas, Introducción a la Programación Orientada a Objetos. Programación en Java, McGraw-Hill, 2008.



Guión explicativo

- Esta presentación tiene como fin lo siguiente:
  - Historia del lenguaje Java
  - Características del lenguaje Java
  - Fundamentos del lenguaje Java



Guión explicativo

- El contenido de esta presentación contiene temas de interés contenidos en la Unidad de Aprendizaje Inteligencia Artificial.

- Las diapositivas deben explicarse en orden, y deben revisarse aproximadamente en 24 horas, además de realizar preguntas a la clase sobre el contenido mostrado.