Tema 1: Introducción la programación en Java

Programación

ÍNDICE DE CONTENIDOS

1. Conceptos Básicos

- **Algoritmo:** Es un conjunto de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Dados un estado inicial y una entrada, siguiendo unos pasos sucesivos se llega a un estado final y se obtiene una solución. **(Wikipedia)**

- **Algoritmo:** Es un conjunto ordenado de reglas o instrucciones tal que siguiéndolas paso a paso se obtiene la respuesta a un problema dado, sean cuales sean los datos o circunstancias particulares del mismo. **(Introducción a la Informática, Alberto Prieto, McGrawHill)**

- **Algoritmo (RAE):** Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

1. Conceptos Básicos

- **Programa:** Un programa es una serie de órdenes o instrucciones ordenadas con una finalidad concreta que realizan una función determinada. **(Programación, Juan Carlos Moreno, Ed RA-MA)**

- **Programa:** Es un conjunto ordenado de instrucciones que se dan a la computadora indicándole las operaciones o tareas que se desea realice. **(Introducción a la Informática, Alberto Prieto, McGrawHill)**

- **Programa informático (RAE):** Conjunto unitario de instrucciones que permite a un ordenador realizar funciones diversas, como el tratamiento de textos, el diseño de gráficos, la resolución de problemas matemáticos, el manejo de bancos de datos, etc.

1. Conceptos Básicos

- **¿Cuál es la diferencia entre algoritmo y programa informático?**

  Los algoritmos pueden ser expresados de muchas maneras incluyendo el lenguaje natural, pseudocódigo, diagramas de flujo y lenguajes de programación entre otros.

  En definitiva los programas informáticos son la implementación de un algoritmo en un lenguaje de programación determinado para implantarse sobre una plataforma o plataformas determinadas .

1. Conceptos Básicos

- **Programación (RAE):** Acción y efecto de programar

- **Programación:** Consiste en describir los módulos o programas, definidos mediante análisis, por medio de instrucciones del lenguaje de programación que se utiliza. **(Introducción a la Informática, Alberto Prieto, McGrawHill)**

1. Conceptos Básicos

- **Lenguaje de Programación:** Es un idioma artificial diseñado para expresar tareas que pueden ser llevadas a cabo por máquinas como las computadoras.

- Está formado por un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

- Permite especificar de manera precisa sobre que datos debe operar una computadora, como deben ser almacenados o transmitidos, y que acciones debe tomar bajo una determinada gama de circunstancias.

1. Conceptos Básicos

- Codificar un algoritmo en un determinado lenguaje y la posterior ejecución y verificación del programa elaborado no es una tarea trivial y depende en parte del lenguaje de programación elegido.

- En este sentido hay que diferenciar entre:

    a) Lenguajes de programación compilados
    b) Lenguajes de programación interpretados
    c) Lenguajes de programación semi interpretados

1. Conceptos Básicos

a) **Lenguajes de programación compilados.**

Una vez escrito el código fuente de un programa , este se traduce por medio de un **compilador** en un archivo ejecutable para una determinada plataforma.

Los lenguajes compilados son leguajes de programación en los que las instrucciones se traducen del lenguaje utilizado a código máquina para una ejecución rápida.

Ejemplos:

- Fortran
- C / C++ / C#
- Ada
- Pascal

1. Conceptos Básicos

b) Lenguajes de programación interpretados

Por el contrario, un lenguaje de programación interpretado es aquel en el que las instrucciones se traducen o interpretan una a una .

Los lenguajes interpretados son compilados línea por línea; es decir cada línea del programa se compila cada vez que va a ser ejecutada, y si por algún motivo alguna línea se ejecuta más de una vez, se compila tantas veces como fuera necesario.

Ejemplos:

- JavaScript
- PHP
- Lisp

1. Conceptos Básicos

c) Lenguajes de programación seminterpretados

Se trata de un termino medio entre los dos anteriores. Estos lenguajes utilizan una representación intermedia, que combina tanto la interpretación como la compilación.

En este caso el compilador produce una representación intermedia del programa o bytecode, que después es ejecutado por un interprete. O como es el caso de Java, una máquina virtual.

Ejemplos:

- Java
- Ruby
- Python

2. El lenguaje JAVA

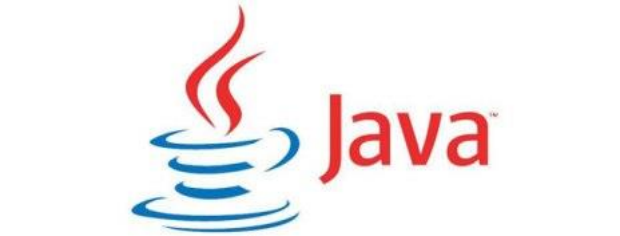
INTRODUCCIÓN

- Desarrollado por los laboratorios Sun, (hoy en día pertenece a Oracle), es uno de los lenguajes de programación orientados a objetos de mayor repercusión. Se emplea tanto en la programación tradicional, distribuida, GUI, Web, dispositivos móviles etc.

- Esta basado en C++ pero simplificado, por tanto es mucho más fácil de usar, de más alto nivel y menos propenso a errores. Posee una amplísima biblioteca estándar de clases predefinida.

- Las aplicaciones de java pueden ser ejecutadas indistintamente en cualquier plataforma sin necesidad de recompilación. Las aplicaciones están típicamente compiladas en un bytecode. En tiempo de ejecución el bytecode es interpretado a código nativo para su ejecución

2. El lenguaje JAVA



CARACTERÍSTICAS

- Gestión avanzada de memoria mediante el recolector de basura.
- Gestión avanzada de errores, tanto en tiempo de compilación como de ejecución.
- Soporte sencillo para múltiples hebras de ejecución.
- Pueden integrarse módulos escritos en C/C++
- Entre diciembre de 2006 y mayo de 2007, SUN Microsystem liberó la mayor parte de sus tecnologías bajo licencia GNU GPL , de acuerdo con las especificaciones del Java Comunity Process, de tal forma que todo el Java de Sun es ahora software libre.

2. El lenguaje JAVA

El JRE y el JDK.

- El JRE (Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada con Java. El usuario final usa el JRE como paquetes software o plugins (o conectores) de un navegador web o Sistema Operativo.

http://Java.com/es/download/

2. El lenguaje JAVA

El JRE y el JDK.

- El JDK (Java Developement Kit, o Kit de desarrollo Java) Incluye:
  - El JRE
  - Herramientas de desarrollo para crear de programas en java como el compilador y el depurador de java, Javadoc para generar documentación, etc

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. El lenguaje JAVA

El JRE y el JDK.

- En Linux podemos usar el compilador intermedio de JAVA y compilar y ejecutar directamente sobre el terminal. En Microsoft Windows también es posible hacerlo desde la consola de línea de comandos de Windows, pero es necesario decirle a Windows donde está dicho compilador.
- Esto se hace a través de las "variables de entorno " del sistema operativo. Para ello debemos añadir:
  - A la variable de entorno "Path" la siguiente línea:

    ; C:\Program Files\Java\jdk1.7.0_67\bin ;
  - A la variable de entorno "ClassPath" la siguiente línea:

    ; C:\Program Files\Java\jdk1.7.0_67\src.zip\ ;

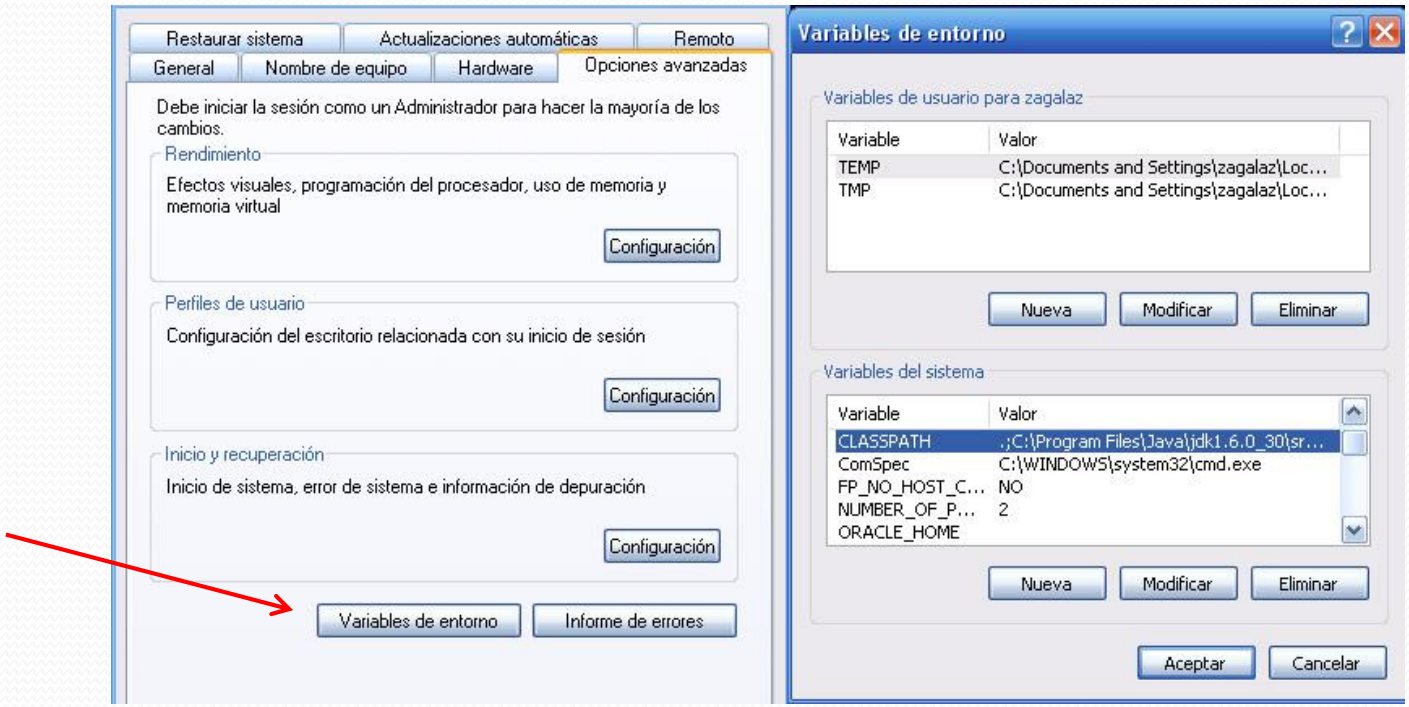
Donde "C:\Program Files\Java\jdk1.7.0_67\" es el directorio donde se ha instalado el JDK de JAVA. Este directorio puede cambiar en función de la versión instalada

2. El lenguaje JAVA

El JRE y el JDK.

- La configuración de las variables de entorno en windows debemos realizarla en las opciones avanzadas de la configuración del sistema. Es decir:

  Mi PC, clic derecho -> propiedades, opciones avanzadas -> variables de entorno



3. Mi primer programa en JAVA

¡¡HOLA MUNDO!!

```
//Importamos la librería java.lang que nos permite imprimir mensajes por pantalla
import java.lang.*;

//Definimos la clase principal de nuestro programa
public class Programa1{

    /*
     Definimos el método main. Este es el método principal
     de cualquier programa ejecutable en java
     */

    public static void main (String [] args){
        //System.out es una clase de java.lang que permite mostrar mensajes por pantalla
        System.out.println("Hola Mundo");
    }
}
```

¡¡¡¡Atención a las mayúsculas y las minúsculas!!!

3. Mi primer programa en JAVA

Crear, Compilar y Ejecutar un programa en Java

- Paso 1: Abrimos el editor de texto “gedit”
- Paso 2: Creamos un nuevo fichero llamado “Programa1.java” y lo guardamos.
- Paso 3: Escribimos el código anterior en el fichero “Programa1.java”
- Paso 4: Tras copiar el código lo guardamos y comprobamos que el código se ha resaltado y el icono del fichero ha cambiado al icono de Java.

3. Mi primer programa en JAVA

Crear, Compilar y Ejecutar un programa en Java

- Paso 5: La compilación del código se realiza desde el terminal. Abre un terminal en el mismo directorio en el que se encuentra el archivo .java.

- Paso 6: Compila el programa “Programa1.java” que hemos creado. Para ello usa el comando javac y el nombre del programa:

  javac Programa1.java

  Si no hay ningún error en el código del programa, se producirá la compilación y se generará el fichero intermedio “Programa1.class” con el código portable que se puede ejecutar en cualquier máquina virtual.

  Si hemos cometido algún error este nos aparecerá en la consola, revisa lo que has escrito en el fichero en busca del error que se indica en consola.

3. Mi primer programa en JAVA

Crear, Compilar y Ejecutar un programa en Java

- Paso 8: Comprueba que efectivamente se ha generado el fichero "Programa1.class".

- Paso 9: Ejecutamos el programa enviándolo a la máquina virtual de java. Para ello escribimos en el terminal:

```
java  Programa1
```

Observa que para ejecutar un programa en Java no se escribe la extensión .class

4. Estructura de un programa en JAVA

- Java es un lenguaje de programación O.O. Con lo cual nuestros programas en realidad son clases, una clase se indica mediante la palabra reservada **class**.

- El código fuente se guarda en archivos con el mismo nombre que la clase que contienen y con extensión ".java"

- El compilador genera un archivo intermedio de clase (".class")por cada una de las clases definidas en el archivo fuente.

- Los programas "ejecutables" deben contener forzosamente el método **main**.

4. Estructura de un programa en JAVA

- Vamos a analizar cada uno de los elementos que componen el siguiente código antes de seguir avanzando:
  - Librerías y/o programas importados.
  - Comentarios
  - Bloques de código
  - Identificadores
  - Sentencias
  - Metacaracteres

4. Estructura de un programa en JAVA

- Librerías y/o programas importados:

  Una librería es un conjunto de recursos. Si queremos utilizar estos recursos en nuestras aplicaciones, es necesario importar la librería de java en la que se encuentran. Para esto se utiliza la palabra reservada **import** seguido del nombre de la librería a importar**.**

  Cuando importamos una librería podemos importar sólo una parte de ella indicando que recurso concreto queremos importar, o podemos importarla entera indicándolo con un *****

  Además de importar librerías propias de java, también podemos importar nuestros propios programas para poder reutilizarlos

4. Estructura de un programa en JAVA

- Comentarios:

  Son líneas de texto insertadas en el programa para documentarlo y facilitar su lectura y comprensión por parte del programador. Los comentarios en Java se indican de 2 formas:

  - Comentarios de una sola línea: //comienzan por 2 barras inclinadas

  - De varias líneas: /* La primera línea comienza por una barra inclinada y un asterisco y la última termina por un asterisco y una barra inclinada */

4. Estructura de un programa en JAVA

- Bloques de código:

  Son el principal mecanismo de encapsulamiento, y se forman con un grupo de sentencias y otros bloques de código delimitados por una llave de apertura y una de cierre: **{}**

  Para mejorar la legibilidad de un código son imprescindibles el uso de comentarios y de las tabulaciones. Los comentarios aclaran la utilizad de cada método o programa, incluso instrucciones complejas. Mientras que las tabulaciones ayudan a entender que partes del código pertenecen a otras.

  Por tabulaciones entendemos a sangrar las líneas de código que están dentro de un mismo juego de **{}**

4. Estructura de un programa en JAVA

- Identificadores:

  Son los nombres que se asigna a las clases, métodos, atributos, objetos, variables etc, para poder diferenciarlos y referirse a ellos dentro del programa. Deben respetar las siguientes normas:

  - No pueden contener ni espacios ni caracteres especiales salvo "_".
  - No pueden empezar por un número.
  - Java distingue mayúsculas de minúsculas, por lo que "Nombre" y "nombre" son identificadores diferentes.
  - Se evita el uso de acentos y la letra ñ.

4. Estructura de un programa en JAVA

- Identificadores:

  Además de las normas anteriores, suelen respetarse las siguientes convenciones para asignar identificadores:

  - El nombre de una clase siempre empieza por mayúscula: Programa1
  - El nombre de un método, variable, atributo etc siempre empieza por minúscula: main
  - Si el identificador está formado por más de un vocablo, a partir del segundo las iniciales deberán ser mayúsculas
  - Los nombres de las clases deberán ser sustantivos, los de los métodos verbos.
  - Los nombres de las variables y atributos deben expresar con claridad su contenido.

4. Estructura de un programa en JAVA

- Sentencias:

  Son las distintas ordenes que debe ejecutar el programa y terminan siempre con un punto y coma “;” podemos distinguir las siguientes:

  - **E/S:** Piden o muestran algún dato por un dispositivo de entrada / salida.
  - **Asignaciones y Expresiones:** Las expresiones y sentencias de asignación guardan un valor o el resultado de una operación en un atributo o variable.
  - **Condicionales:** Expresan una condición para definir el flujo del programa. **If, if-else y switch.**
  - **Bucles:** Sentencias que se encargan de repetir una o varias sentencias un número determinado de veces. **For, while, do-while.**
  - **Saltos:** Llevan al compilador a un punto específico del programa. **Break, continue, return**

4. Estructura de un programa en JAVA

- Metacaracteres:

  Son una serie de caracteres especiales, que sirven para el control y la significación puntual en las sentencias y los bloques de código:

  ( ) [ ] { } \ ^ $?

5. INTRODUCCIÓN A LA PROGRAMACIÓN EN JAVA

- Java, como ya hemos visto anteriormente, surge como una evolución, un intento de simplificar y mejorar C/C++ con lo cual estos lenguajes coinciden en muchos aspectos básicos:

  2.1 Expresiones.

  2.2 Variables.

  2.3 Operadores.

5.1 EXPRESIONES

- Las expresiones son fragmentos de código formadas por 2 o más miembros separados entre si por operadores que los evalúan y los relacionan.
  Ejemplo:

  x = 3 + 2;

- El orden es importante, ya que el resultado de una expresión se guarda en la parte izquierda

  3 + 2 = x; → esta expresión produciría un error al compilar el programa

5.2 VARIABLES

- Una variable, es una zona de la memoria del computador que se reserva a lo largo de un programa informático donde se puede almacenar un valor. Algunas de las características de una variable son las siguientes:

  - Poseen un identificador para poder referirse y acceder a ellas.

  - Su valor puede cambiarse o modificarse a lo largo de un programa cuantas veces sea necesario.

  - Son de un tipo determinado que debe indicarse al comenzar a trabajar con ella

5.2 VARIABLES

- Según el tipo de dato que pueden almacenar, podemos distinguir 2 tipos principales de variables:

a) Variables de tipos primitivos: Poseen un valor único y pueden ser entero, decimal, carácter, lógico…

b) Variables de referencia: Las variables de referencia son objetos de una determinada clase, pudiendo hacer referencia a más de un valor.

5.2 VARIABLES

- Desde el punto de vista de la vida de una variable, estas pueden ser:

a) Variables miembro de una clase(atributos): Se definen en una clase, fuera de cualquier método.

b) Variables locales: Se definen dentro de un método o un bloque entre llaves {}. Se crean en el interior del bloque y el recolector de basura las destruye al terminar este.

5.2 VARIABLES

- El nombre de una variable es su identificador, y como tal cumple con las reglas de los identificadores

- Además en Java existe una serie de palabras reservadas que tienen un significado específico dentro de un programa y por lo tanto no se pueden utilizar como nombres de variables. Dichas palabras son:

5.2 VARIABLES

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

5.2 VARIABLES

- **Tipos primitivos:** Son aquellas variables sencillas que contienen tipos de información más habituales: booleanos, caracteres, enteros y reales. Para ello en Java existen los siguientes tipos primitivos:
  - **Enteros:**
    - byte(8 bits) valores entre -128 y 127.
    - short(16 bits) valores entre -32768 y 32767.
    - int (32 bits) valores entre -2147483648 y 2147483647.
    - long (64 bits) valores entre -9223372036854775808 y 9223372036854775807.
  - **Reales:**
    - float (32 bits) hasta 7 cifras decimales
    - double(64 bits) hasta 15 cifras decimales
  - **Caracteres:**
    - char (16 bits) caracteres (65536 caracteres diferentes) (entre comillas simples ‘a’)
  - **Booleanos**
    - boolean (1 bit) true o false

5.2 VARIABLES

- **Cadenas de Texto:** Las cadenas de texto no son tipos primitivos, en realidad son objetos. Pero debido a su amplio uso, java nos permite trabajar con ellas como si fueran tipos primitivos:

    - **String:**
        - Para diferenciarlas de los caracteres van entre comillas dobles “a”.
        - “a” es una cadena de texto, ‘a’ es un carácter
        - Ejemplos: “hola”, “Esto es una cadena”, “123”, “ ”, “”
        - Su valor por defecto es “” o null

5.2 VARIABLES

- ¿Cuál sería el tipo de dato más correcto para los siguientes casos?
  - Edad de una persona
  - Número de teléfono
  - Sueldo en euros
  - Respuesta a una pregunta con ‘s’ o ‘n’
  - El valor de PI
  - Campo que indique si una persona está casada o soltera

5.2 VARIABLES

- **Definición de variables**

  Una variable se define especificando el tipo y el nombre de dicha variable. Estas variables pueden ser tanto de tipos primitivos como referencias a objetos de alguna clase.

  Cuando se define una variable se puede definir su valor inicial. Si no se especifica estas se inicializan a 0, salvo boolean y char que se inicializan a false y a '\0'.

  Las variables de tipo referencia se inicializan por defecto a null (nulo).

  int edad =1;

  double iva =0.21;

5.2 VARIABLES

- **Constantes**

  Las constantes son un tipo especial de variable, poseen un valor fijo que no se puede cambiar en el transcurso de la ejecución de un programa. Estas se declaran mediante el modificador **final.**

  ```
  final double PI = 3.1416159265;
  ```

  Como regla adicional, las constantes suelen declararse en mayúsculas para indicar que lo son. Ejemplos de constantes existente en JAVA:

  Math: PI, E //numeros pi y E
  Double: MAX_VALUE, MIN_VALUE //valores máximo y mínimo de un double
  Integer: MAX_VAULE, MIN_VALUE//valores máximo y mínimo de un int

5.2 VARIABLES

- **Conversión de tipos**

  En algunos casos suele ser necesario convertir un tipo de dato a otro. Este proceso se conoce como conversión de tipos.

  Se debe tener en cuenta el tipo de dato que se va a convertir, ya que si se convierte un dato a otro que tenga una cantidad menor de bits puede haber perdida de información.

  El ejemplo más claro de esto es si se quiere guardar un real a un entero. Con lo cual se producirá la perdida de los números decimales.

  Otro caso puede ser convertir un entero en un short. Lo cual puede producir que se eliminen las primeras cifras del mismo.

5.2 VARIABLES

- **Conversión de tipos**

  Si la conversión se realiza a un tipo de dato con mayor número de bits, la conversión no supone ningún tipo de conflicto y se realiza automáticamente. Este tipo de conversión se llama **conversión implícita.**

  Por el contrario si la conversión es a un tipo de dato con menor número de bits que el original, se puede producir perdida de información y hay que indicárselo explícitamente a java, se la conoce como **conversión explícita.**

  Para realizar una conversión explícita se tiene que poner el tipo de dato al que se desea convertir entre paréntesis y luego el dato en cuestión:

5.2 VARIABLES

- **Conversión de tipos**

  Un ejemplo de conversión explícita puede ser:

```
double numero1 = 32.17;
int numero2;
numero2= (int) numero1;  //con esto numero2 pasará a valer 32
```

  Un ejemplo de conversión implícita puede ser:

```
int numero1 = 32;
double numero2;
numero2= numero1;  //con esto numero2 pasará a valer 32.0
```

5.3 OPERADORES

Los operadores en Java son casi idénticos a los operadores de C y C++

- **Operadores aritméticos**

  Son operadores binarios (requieren siempre al menos 2 operandos) y corresponden a las operaciones aritméticas habituales:
  - Suma +
  - Resta –
  - Multiplicación *
  - División /
  - Resto de la división %

- **Operadores de asignación**

  Permiten asignar un valor a una variable:

  variable = expresión;

5.3 OPERADORES

- **Otros Operadores de asignación**

Operador	Utilización	Expresión equivalente
+=	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

5.3 OPERADORES

- **Operadores incrementales**

  Los operadores incremento (++) y decremento (--) incrementan o decrementan en una unidad la variable a la que se aplica.

  x++ → x = x + 1

  x-- → x = x - 1

- **Operador de concatenación**

  El operador + se utiliza también para unir cadenas de caracteres

  ```
  int anio = 2017;
  System.out.println("hola mundo del "+anio+" ¿que tal?");
  ```

5.3 OPERADORES

- **Operadores relacionales**

Sirven para realizar comparaciones de igualdad, desigualdad y relación de mayor o menor. El resultado de estos operadores es siempre un valor booleano (true o false)

Operador	Utilización	El resultado es true
>	op1 > op2	si op1 es mayor que op2
>=	op1 >= op2	si op1 es mayor o igual que op2
<	op1 < op2	si op1 es menor que op2
<=	op1 <= op2	si op1 es menor o igual que op2
==	op1 == op2	si op1 y op2 son iguales
!=	op1 != op2	si op1 y op2 son diferentes

5.3 OPERADORES

- **Precedencia de operadores**

  El orden en que se realizan las operaciones es fundamental para determinar el resultado de una operación. Por ejemplo el resultado de la operación 3+2*5 dependerá de que operación se realice primero.

  El mejor método es indicarle a Java en que orden queremos que se realicen las operaciones. Para esto Java utiliza los paréntesis (). En una expresión con paréntesis siempre se evaluará primero los operadores que estén dentro del paréntesis.

5.3 OPERADORES

- **Precedencia de operadores**

  En la siguiente tabla se puede observar cual es la precedencia de operadores en java. Teniendo siempre más prioridad de arriba abajo y de izquierda a derecha

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

5.3 OPERADORES

- **Operadores lógicos**

Los operadores lógicos se utilizan para construir expresiones lógicas. Combinando valores lógicos o resultados de los operadores relacionales.

Operador	Nombre	Utilización	Resultado
&&	AND	op1 && op2	true si op1 y op2 son true. Si op1 es false ya no se evalúa op2
||	OR	op1 || op2	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2
!	negación	! op	true si op es false y false si op es true
&	AND	op1 & op2	true si op1 y op2 son true. Siempre se evalúa op2
|	OR	op1 | op2	true si op1 u op2 son true. Siempre se evalúa op2

6. ENTRADA Y SALIDA DE DATOS

- **Salida de datos**

  Para poder realizar una salida de datos por la pantalla es necesario usar la clase **System**. Esta clase tiene el atributo **out** que hacer referencia a la salida estándar (pantalla o monitor).

  Por tanto para mostrar un mensaje por pantalla en primer lugar es necesario importar la librería que contiene a la clase System:

  import java.lang.*; //ya no es necesario

  Después cuando se desee mostrar un mensaje por pantalla bastará con escribir la siguiente instrucción:

  System.out.print(mensaje_a_mostrar);

  Donde el mensaje a mostrar puede ser una variable o una cadena de caracteres, Si se trata de una cadena debe ir entre comillas dobles “ ”

6. ENTRADA Y SALIDA DE DATOS

- Salida de datos:

  Caracteres de Escape

  Los caracteres de escape son pequeñas constantes de gran utilizad para formatear las salidas de datos:

  “\”” escribir comillas dobles: “hola \“mundo\” ”: hola “mundo”

  “\\” permite escribir una \ : “hola mundo \\”: hola mundo \

  “\b” borra el último carácter: “ hola mundo\b”: hola mund

  “\r” retorno de carro: “hola \r mundo”: mundo

  “\t” introduce un tabulador: “hola \t mundo” : hola mundo

  “\n” introduce un salto de línea: “hola \n mundo”: hola
  mundo

6. ENTRADA Y SALIDA DE DATOS

- Entrada de datos

La lectura de datos de teclado en un programa en java no es una operación trivial. Para esto son necesarios ciertos conocimientos de programación dirigida a objetos, que permitan configurar la comunicación con el teclado.

Para simplificar esta tarea utilizaremos una de las utilidades propias de java. La clase scanner. Esta clase se encuentra dentro del paquete **util** de java, por lo que será necesario importarlo en todos nuestros programas que necesiten usar el teclado:

import java.util.Scanner;

6. ENTRADA Y SALIDA DE DATOS

- Entrada de datos

Para poder trabajar con esta utilidad es necesario inicializarla antes de realizar cualquier lectura de la siguiente forma:

Scanner sc = new Scanner(System.in);

Donde **System.in** indica que estamos leyendo del teclado.

sc es el nombre que le queremos dar a la conexión que hemos abierto con el teclado, podemos darle cualquier nombre: sc, leer, teclado…

Esta operación sólo hay que realizarla una vez, independientemente de cuantas lecturas queramos hacer.

6. ENTRADA Y SALIDA DE DATOS

- Entrada de datos

  La clase Scanner dispone de los siguientes métodos para poder leer datos:

```
nextBoolean()                                          //Lee un booleano (True, False)
nextByte()  nextShort()    nextInt()  nextLong()       //Leen enteros
nextFloat()        nextDouble()                        //Leen decimales
nextLine()                                             //Lee cadenas de texto
next().charAt(0)                                       //Lee un carácter
```

  Ejemplo:

```
Scanner sc = new Scanner(System.in);
System.out.println("Dime un número");
int num=sc.nextInt();
System.out.println(num);    //muestra el número leído por pantalla
```

6. ENTRADA Y SALIDA DE DATOS

- Limpiar el Buffer de entrada

Al introducir un dato numérico de cualquier tipo, se introduce dicho número más el carácter de intro (‘\n’), al realizar la lectura, solamente se lee el número y el salto de línea permanece en el buffer de lectura

Esto puede ser problemático si después de leer un número, se lee una cadena de caracteres, ya que esta tomará ese ultimo salto de línea del teclado en lugar de realizar una lectura. Para evitar este problema, basta con realizar una lectura extra para eliminar dicho salto de línea.

6. ENTRADA Y SALIDA DE DATOS

- Ejemplo:

```
 //inicialización del teclado
Scanner sc = new Scanner(System.in);

//pedimos el número
System.out.println("Dime un número");
int num=sc.nextInt();  //lectura del número

//pedimos el nombre
System.out.println("Ahora dime un nombre");
sc.nextLine();     //eliminamos lo que pueda haber en el buffer de lectura
String cadena=sc.nextLine();

//mostramos los dos datos leídos
System.out.println("has introducido el número "+num+" y el nombre "+cadena);
```

6. ENTRADA Y SALIDA DE DATOS

- Entrada de datos

```
/*Importamos la librería java.util.Scanner para poder leer datos de teclado*/
import java.lang.*;

//Definimos la clase de nuestro programa
public class Programa2{
    /**
     Definimos el método main. Este es el método principal
     de cualquier programa ejecutable en java.
     */
     public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);//configuro el teclado para leer
        int anio= 0;

        //Pido el año actual
        System.out.println("¿en que año estamos?");
        anio=sc.nextInt();

        //Muestro el saludo con el año actual
        System.out.println("Hola Mundo del "+anio);
    }
}
```

7. BIBLIOGRAFÍA

- García de Jalón, j.: "*Aprende Java como si estuvieras en primero*". Editorial TECNUN. 2000
- Holzner, S.: "*La biblia de JAVA 2*". Editorial Anaya Multimedia 2000.
- Schildt, H.: "*Java, manual de referencia*". Editorial McGraw-Hill 2009.

7. BIBLIOGRAFÍA

- Oracle and/or its affiliates: "*Java ™ Platform, Standard Edition 9 API Specification*". Última visita: Octubre 2017.
  https://docs.oracle.com/javase/9/docs/api/overview-summary.html

- Documentación oficial Java JSE 9 (y anteriores). Última visita: Octubre 2017.
  http://docs.oracle.com/javase/9/

- Programación en castellano: "Java". Última visita: Octubre 2017.
  http://www.programacion.net/java

- López, J.C.: "*Curso de JAVA* Última visita: Octubre 2015.
  http://www.cursodejava.com.mx