CURSO
BACKEND 1

Introducción a Java

Introducción a Java

OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

- Utilizar el nuevo IDE.
- Comprender la estructura básica de Java
- Comprender los tipos de datos en Java
- Definir y operar variables.
- Utilizar los métodos de escritura y salida
- Utilizar las clases de utilidad

egg | Argentina programa 4.0

```
package <nombre_paq_sup>[.<nombre_sub_paq>]*;
```

La declaración package, en caso de existir, debe estar al principio del archivo fuente y sólo la declaración de un paquete está permitida. Los nombres de los paquetes los pondrá el programador al crear el programa y son jerárquicos (al igual que una organización de directorios en disco) además, están separados por puntos. Es usual que sean escritos completamente en minúscula.

¿Cuál es la clase?

La primera línea del programa, después del package. Define una clase que se llama HolaMundo. En el mundo de orientación a objetos, todos los programas se definen en términos de objetos y sus relaciones. Las clases sirven para modelar los objetos que serán utilizados por nuestros programas. Los objetos, las clases y los paquetes **son conceptos que serán abordados con profundidad más adelante en el curso.**

Una clase está formada por una parte correspondiente a la declaración de la clase, y otra correspondiente al cuerpo de la misma:

```
Declaración de clase {

Cuerpo de clase

}
```

En la plantilla de ejemplo se ha simplificado el aspecto de la Declaración de clase, pero sí que puede asegurarse que la misma contendrá, como mínimo, la palabra reservada class y el nombre que recibe la clase. La definición de la clase o cuerpo de las comienza con una llave abierta (**{**) y termina con una llave cerrada (**}**). El nombre de la clase lo define el programador.

¿Cuál es el Método?

Después de la definición de clase se escribe la definición del método main(). Pero que es un método?. Dentro del cuerpo de la clase se declaran los atributos y los métodos de la clase. Un método es una secuencia de sentencias ejecutables. Las sentencias de un método quedan delimitadas por los caracteres { y } que indican el inicio y el fin del método, respectivamente. Si bien es un tema sobre el que se profundizará más adelante en el curso, los métodos son de vital importancia para los objetos y las clases. En un principio, para dar los primeros pasos en Java nos alcanza con esta definición.

Método main()

Ahora sabemos lo que es un método, pero en el ejemplo podemos ver el método main(). El main() sirve para que un programa se pueda ejecutar, este método, vendría a representar el Algoritmo / FinAlgoritmo de PseInt y tiene la siguiente declaración:

```
public static void main(String[] args){
```

A continuación, describiremos cada uno de los modificadores y componentes que se utilizan ***siempre*** en la declaración del método ***main():***

public: es un tipo de acceso que indica que el método ***main()*** es público y, por tanto, puede ser llamado desde otras clases. Todo método ***main()*** debe ser público para poder ejecutarse desde el intérprete Java (JVM).

static: es un modificador el cual indica que la clase no necesita ser instanciada para poder utilizar el método. También indica que el método es el mismo para todas las instancias que pudieran crearse.

void: indica que la función o método ***main()*** no devuelve ningún valor.

El método ***main()*** debe aceptar siempre, como parámetro, un vector de strings, que contendrá los posibles argumentos que se le pasen al programa en la línea de comandos, aunque como es nuestro caso, no se utilice.

Luego, al indicarle a la máquina virtual que ejecute una aplicación el primer método que ejecutará es el método ***main()***. Si indicamos a la máquina virtual que corra una clase que no contiene este método, se lanzará un mensaje advirtiendo que la clase que se quiere ejecutar no contiene un método ***main()***, es decir que dicha clase no es ejecutable.

Si no se han comprendido hasta el momento muy bien todos estos conceptos, los mismos se irán comprendiendo a lo largo del curso.

Tipos de Datos Primitivos

Primitivos: Son predefinidos por el lenguaje. La biblioteca Java proporciona clases asociadas a estos tipos que proporcionan métodos que facilitan su manejo.

byte	Es un entero con signo de 8 bits, el mínimo valor que se puede almacenar es -128 y el máximo valor es de 127 (inclusive).
short	Es un entero con signo de 16 bits. El valor mínimo es -32,768 y el valor máximo 32,767 (inclusive).
int	Es un entero con signo de 32 bits. El valor mínimo es -2,147,483,648 y el valor máximo es 2,147,483,64 (inclusive). Generalmente es la opción por defecto.
long	Es un entero con signo de 64 bits, el valor mínimo que puede almacenar este tipo de dato es -9,223,372,036,854,775,808 y el máximo valor es 9,223,372,036,854,775,807 (inclusive).
float	Es un número decimal de precisión simple de 32 bits (IEEE 754 Punto Flotante).
double	Es un número decimal de precisión doble de 64 bits (IEEE 754 Punto Flotante).

boolean	Este tipo de dato sólo soporta dos posibles valores: verdadero o falso y el dato es representado con tan solo un bit de información.
char	El tipo de dato carácter es un simple carácter unicode de 16 bits. Su valor mínimo es de '¥u0000' (En entero es 0) y su valor máximo es de '¥uffff' (En entero es 65,535). Nota: un dato de tipo carácter se puede escribir entre comillas simples, por ejemplo 'a', o también indicando su valor Unicode, por ejemplo '¥u0061'.
String	Además de los tipos de datos primitivos el lenguaje de programación Java provee también un soporte especial para cadena de caracteres a través de la clase String. Encerrando la cadena de caracteres con comillas dobles se creará de manera automática una nueva instancia de un objeto tipo String. String cadena = “Sebastián”; Los objetos String son inmutables, esto significa que una vez creados, sus valores no pueden ser cambiados. Si bien esta clase no es técnicamente un tipo de dato primitivo, el lenguaje le da un soporte especial y hace parecer como si lo fuera.

Valores por defecto

En Java no siempre es necesario asignar valores cuando nuevos atributos son declarados. Cuando los atributos son declarados, pero no inicializados, el compilador les asignará un valor por defecto. A grandes rasgos el valor por defecto será cero o null dependiendo del tipo de dato. La siguiente tabla resume los valores por defecto dependiendo del tipo de dato.

short	**0**
int	**0**
long	**0**
double	**0.0**
boolean	**False**
char	**'\u0000'**
String	**Null**
Objetos	**Null**

Operadores

Los operadores son símbolos especiales de la plataforma que permiten especificar operaciones en uno, dos o tres operandos y retornar un resultado. También aprenderemos qué operadores poseen mayor orden de precedencia. Los operadores con mayor orden de precedencia se evalúan siempre primero.

Primeramente, proceden los operadores unarios, luego los aritméticos, después los de bits, posteriormente los relacionales, detrás vienen los booleanos y por último el operador de asignación. La regla de precedencia establece que los operadores de mayor nivel se ejecuten primero. Cuando dos operadores poseen el mismo nivel de prioridad los mismos se evalúan de izquierda a derecha.

Operadores Aritméticos	
+	Operador de Suma
-	Operador de Resta
*	Operador de Multiplicación
/	Operador de División
%	Operador de Módulo
Operadores Unarios	
+	Operador Unario de Suma; indica que el valor es positivo.
-	Operador Unario de Resta; indica que el valor es negativo.
++	Operador de Incremento.
--	Operador de Decremento.

Operadores de Igualdad y Relación	
==	Igual
!=	Distinto
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

Tipos de Instrucciones

Además de los elementos de un programa/algoritmo, tenemos las instrucciones que pueden componer un programa. Las instrucciones —acciones— básicas que se pueden implementar de modo general en un algoritmo y que esencialmente soportan todos los lenguajes son las siguientes:

- ✓ **Instrucciones de inicio/fin**, son utilizadas para delimitar bloques de código.
- ✓ **Instrucciones de asignación**, se utilizan para asignar el resultado de la evaluación de una expresión a una variable. El valor (dato) que se obtiene al evaluar la expresión es almacenado en la variable que se indique.
- ✓ **Instrucciones de lectura**, se utilizan para leer datos de un dispositivo de entrada y se asignan a las variables.
- ✓ **Instrucciones de escritura**, se utilizan para escribir o mostrar mensajes o contenidos de las variables en un dispositivo de salida.
- ✓ **Instrucciones de bifurcación**, mediante estas instrucciones el desarrollo lineal de un programa se interrumpe. Las bifurcaciones o al flujo de un programa puede ser según el punto del programa en el que se ejecuta la instrucción hacia adelante o hacia atrás.

Entrada y Salida de Información

Los cálculos que realizan las computadoras requieren, para ser útiles la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados, es decir, salida.

Las operaciones de entrada permiten leer determinados valores y asignarlos a determinadas variables y las operaciones de salida permiten escribir o mostrar resultados de determinadas variables y las operaciones, o simplemente mostrar mensajes.

Clases de utilidad

Se acuerdan que en PseInt vimos una serie de funciones y dijimos que las funciones, son herramientas que nos proporciona PSeInt y cumplen el propósito de ayudarnos a resolver ciertos problemas. Bueno, en Java existe algo muy parecido que se llama **Clases de utilidad.**

¿Qué son las clases de utilidad?

Las clases de utilidad son clases que definen un conjunto de métodos/funciones que realizan operaciones, normalmente muy utilizadas o necesarias. Lo que va a facilitar las clases es no tener que escribir dicha operación nosotros, por ejemplo, supongamos que tenemos que calcular la raíz cuadrada de un número, Java no va a dar un método/función, que pasándole un número, nos devuelve el resultado de su raíz cuadrada.

Entre las clases de utilidad de Java más utilizadas y conocidas están las siguientes: Arrays, String, Integer, Math, Date, Calendar y GregorianCalendar.

En esta guía solo vamos a ver a **Math** y **String** para hacer algunos ejercicios y después veremos el resto en mayor profundidad. Estas nos van a ayudar junto con Java, a lograr resolver problemas de manera más sencilla.

Clase String

Método	Descripción.
charAt(int index)	Retorna el carácter especificado en la posición index
equals(String str)	Sirve para comparar si dos cadenas son iguales. Devuelve true si son iguales y false si no.
equalsIgnoreCase(String str)	Sirve para comparar si dos cadenas son iguales, ignorando la grafía de la palabra. Devuelve true si son iguales y false si no.

compareTo(String otraCadena)	Compara dos cadenas de caracteres alfabéticamente. Retorna 0 si son iguales, entero negativo si la primera es menor o entero positivo si la primera es mayor.
concat(String str)	Concatena la cadena del parámetro al final de la primera cadena.
contains(CharSequence s)	Retorna true si la cadena contiene la secuencia tipo char del parámetro.
endsWith(String suffix)	Retorna verdadero si la cadena es igual al objeto del parámetro
indexOf(String str)	Retorna el índice de la primera ocurrencia de la cadena del parámetro
isEmpty()	Retorna verdadero si la longitud de la cadena es 0
length()	Retorna la longitud de la cadena
replace(char oldChar, char newChar)	Retorna una nueva cadena reemplazando los caracteres del primer parámetro con el carácter del segundo parámetro
split(String regex)	Retorna un arreglo de cadenas separadas por la cadena del parámetro

startsWith(String prefix)	Retorna verdadero si el comienzo de la cadena es igual al prefijo del parámetro.
substring(int beginIndex)	Retorna la sub cadena desde el carácter del parámetro
substring(int beginIndex, int endIndex)	Retorna la sub cadena desde el carácter del primer parámetro hasta el carácter del segundo parámetro
toCharArray()	Retorna el conjunto de caracteres de la cadena
toLowerCase()	Retorna la cadena en minúsculas
toUpperCase()	Retorna la cadena en mayúsculas

Java al ser un lenguaje de tipado estático, requiere que para pasar una variable de un tipo de dato a otro necesitemos usar un conversor. Por lo que, para convertir cualquier tipo de dato a un String, utilicemos la función valueOf(n).

Ejemplo:

```
int numEntero = 4;
String numCadena = String.valueOf(numEntero);
```

Si quisiéramos hacerlo al revés, de String a int se usa el método de la clase Integer, parseInt().

Ejemplo:

```
String numCadena = "1";
int numEntero = Integer.parseInt(numCadena);
```

Clase Math

En ocasiones nos vemos en la necesidad de incluir **cálculos, operaciones, matemáticas, estadísticas**, etc en nuestros programas Java.

Es cierto que muchos cálculos se pueden hacer simplemente utilizando los operadores aritméticos que **Java** pone a nuestra disposición, pero existe una opción mucho más sencilla de utilizar, sobre todo para ***cálculos complicados***. Esta opción es la **clase Math** del paquete **java.lang**.

La clase Math nos ofrece numerosos y valiosos métodos y constantes estáticos, que podemos utilizar tan sólo anteponiendo el nombre de la clase.

Método	Descripción.
abs(double a)	Devuelve el valor absoluto de un valor double introducido como parámetro.
abs(int a)	Devuelve el valor absoluto de un valor Entero introducido como parámetro.
abs(long a)	Devuelve el valor absoluto de un valor long introducido como parámetro.
max(double a, double b)	Devuelve el mayor de dos valores double
max(int a, int b)	Devuelve el mayor de dos valores Enteros.
max(long a, long b)	Devuelve el mayor de dos valores long.

min(double a, double b)	Devuelve el menor de dos valores double.
min(int a, int b)	Devuelve el menor de dos valores enteros.
min(long a, long b)	Devuelve el menor de dos valores long.
pow(double a, double b)	Devuelve el valor del primer argumento elevado a la potencia del segundo argumento.
random()	Devuelve un double con un signo positivo, mayor o igual que 0.0 y menor que 1.0.
round(double a)	Devuelve el long redondeado más cercano al double introducido.
sqrt(double a)	Devuelve la raíz cuadrada positiva correctamente redondeada de un double.
floor(double a)	Devuelve el entero más cercano por debajo.

Método random() de la clase Math

El **método random** podemos utilizarlo para generar **números al azar**. El rango o margen con el que trabaja el método random oscila entre 0.0 y 1.0 (Este último no incluido)

Por lo tanto, para generar un número entero entre 0 y 9, hay que escribir la siguiente sentencia:

```
int numero = (int) (Math.random() * 10);
```