

```
require 'spec_helper'  
require 'rspec/rails'
```

```
require 'capybara/rspec'  
require 'capybara/rails'
```

```
Capybara.javascript_driver = :webkit  
Category.delete_all; Category.create  
Shoulda::Matchers.configure do |config|  
  config.integrate do |with|  
    with.test_framework :rspec  
    with.library :rails
```

VARIABLES Y ESTRUCTURAS DE CONTROL EN LA PROGRAMACIÓN
ORIENTADA A OBJETOS: JAVA

SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

```
0  
21 # Add additional require files with some support files  
22 # Requires supporting ruby files with support/ and its subdirectories. This will be required by default.  
23 # spec/support/ and its subdirectories. This will be required by default.  
24 # run as spec files by default. This will be required by default.  
25 # in _spec.rb will both be required and run as spec files.  
26 # run twice. It is recommended that you run this command twice.  
27 # end with _spec.rb. You can configure this command to run  
28 # on the command line as follows:   
  'mongoid'
```



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



ESTRUCTURA DE CONTENIDOS

INTRODUCCIÓN.....	3
1. VARIABLES.....	3
1.1 ¿Cómo se declaran?.....	4
1.2 Tipos.....	5
2. TIPOS DE DATOS.....	8
2.1 Primitivos.....	8
2.2 Clases.....	9
3. OPERACIONES BÁSICAS.....	10
3.1 Operadores.....	10
3.2 Jerarquía.....	15
4. FUNCIONES.....	19
5. ENTRADA DE INFORMACIÓN.....	22
GLOSARIO.....	26
REFERENCIAS BIBLIOGRÁFICAS.....	27
CRÉDITOS.....	28



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

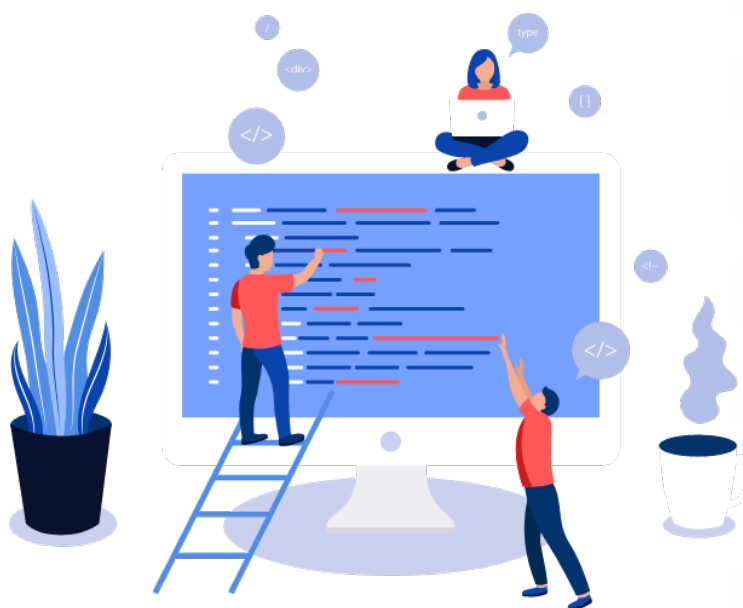
INTRODUCCIÓN



En este material de formación se aborda la sintaxis para la declaración de variables y su respectiva asignación de valores, así como también los tipos de datos y la jerarquía de operaciones, haciendo uso del lenguaje de programación Java.

En el contexto de la programación, la lógica siempre es igual, sin importar el lenguaje que se esté manejando; sin embargo, es esencial conocer la sintaxis específica de las diferentes instrucciones que van a ser escritas, dado que estas, por lo general, si difieren de un lenguaje a otro.

1. VARIABLES



Las variables consisten en espacios de almacenamiento de información en la memoria; están definidas por medio de identificadores que se representan en palabras, los cuales son utilizados durante el proceso de programación cada vez que se requiera almacenar datos en la memoria del programa; cada variable debe tener establecido un tipo de dato que determina la clase de valor que puede almacenar.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

1.1 ¿Cómo se declaran?

Para declarar variables se debe contar con tres aspectos básicos que son: tipo de dato, nombre de la variable y valor; sin embargo, si se trata de variables que sean propias de un objeto, debe tenerse en cuenta también la privacidad.

Algunos ejemplos de declaración de variables son los siguientes:



int	valor = 17;
long	nota = 3.8;
string	texto = "Bienvenidos";
boolean	acepta = false;

En los ejemplos anteriores se observan los tres aspectos básicos mencionados; están los tipos de datos int, long, string y boolean, y están los nombres de variables valor, nota, texto y acepta; asimismo, a cada una de esas cuatro variables de ejemplo, se les ha asignado un valor inicial.

En el caso de las variables que sean propias de un objeto, los ejemplos serían los siguientes:



private static int	valor
static long	nota
public static string	texto
protected static int	nota2

Las palabras private y public hacen referencia a la privacidad de los atributos, entendiéndose de la siguiente forma:

- 1. Private:** el atributo puede ser accedido únicamente por la misma clase.
- 2. Public:** el atributo puede ser accedido desde cualquier clase o instancia.
- 3. Protected:** el atributo puede ser accedido desde la misma clase, desde las clases del mismo paquete, y desde clases que se hayan heredado de la clase original.

En el caso del ejemplo static long nota; la privacidad sería por defecto, lo que significa que el atributo puede ser accedido por la misma clase, y también por las clases del mismo paquete.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

A continuación, se observan las restricciones de acceso, según la privacidad declarada:

Acceso	Clase	Subclase	Paquete	Otros paquetes
por defecto	✓	✓	✗	✗
public	✓	✓	✓	✓
private	✓	✗	✗	✗
protected	✓	✓	✓	✗

1.2 Tipos

Teniendo en cuenta el lugar donde están definidas, la forma de declaración y su accesibilidad, las variables se dividen en tres tipos:

Algunos ejemplos de declaración de variables son los siguientes:

» Locales

Suelen ser declaradas dentro de un método; se crean cuando este es llamado, y se destruyen al salir del mismo. Únicamente es posible acceder dentro del mismo método.

A continuación, se observa un ejemplo donde se define y utiliza una variable local dentro del método:

```
public void EmpleadoEdad()
{
    int edad = 0;
    edad = edad + 7;
    System.out.println("La edad del empleado es: " + edad);
}
```

La variable local es creada y utilizada dentro del método, por lo tanto, la ejecución no presenta ningún inconveniente; sin embargo, si dicha variable intenta utilizarse por fuera del método, como se observa en el siguiente ejemplo, el programa devolvería un error.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

```
public class DetalleEmpleado
{
    public void EmpleadoEdad()
    {
        int edad = 0;
        edad = edad + 7;
    }

    public static void main (String args[])
    {
        System.out.println("La edad del empleado es: " + edad);
    }
}
```

» Estáticas

También llamadas variables de clase, son declaradas dentro de una clase, pero por fuera de cualquier método; se crean al iniciar la ejecución del programa, y se destruyen cuando esta finaliza. Pueden ser accedidas con el nombre de la clase, sin ser necesario ningún objeto; solamente puede haber una variable estática por cada clase.

Se utilizan especialmente cuando se desea que los objetos tengan referencia a algún valor que sea igual en todas las instancias de la clase, y teniendo en cuenta que dicho valor pueda ser cambiado; esto facilita la modificación de información con una sola acción, dado que basta con editar el valor en la clase principal, y todas sus instancias podrán reconocerlo.

```
class Empleado
{
    public static double sueldo;
}

public class EmpleadoUno
{
    public static void main (String args[])
    {
        Empleado.sueldo = 890000;
        System.out.println("El salario es: " + Empleado.sueldo);
    }
}
```



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

En el ejemplo anterior, en la clase Empleado se declara el salario como una variable estática, y posteriormente en el main se utiliza dicha variable, únicamente haciendo uso del nombre de la clase, sin que haya la necesidad de un objeto.

» No estáticas

También conocidas como variables de instancia. Son declaradas dentro de una clase, pero por fuera de cualquier método; se crean al instanciar un objeto de la clase, y se destruyen al destruirse dicho objeto; cada objeto puede tener una copia de la variable no estática.

```
class Empleados {  
    int edad;  
    String área;  
    double sueldo;  
}  
  
class EmpleadosSucursal {  
  
    public static void main(String args[] ) {  
        Empleados emp1 = new Empleados();  
        emp1.edad = 21;  
        emp1.area = "Administración";  
        emp1.sueldo = 3500000;  
        Empleados emp2 = new Empleados();  
        emp2.edad = 34;  
        emp2.area = "Comercial";  
        emp2.sueldo = 1100000;  
        System.out.println("Información del empleado 1");  
        System.out.println("Área: " + emp1.area);  
        System.out.println("Edad: " + emp1.edad);  
        System.out.println("Sueldo: " + emp1.sueldo);  
        System.out.println("Información del empleado 2");  
        System.out.println("Área: " + emp2.area);  
        System.out.println("Edad: " + emp2.edad);  
        System.out.println("Sueldo: " + emp2.sueldo);  
    }  
}
```

En el ejemplo anterior, en la clase Empleados, se declaran la edad, el área y el sueldo como variables de instancia (no estáticas) y posteriormente en el main se imprimen los valores de dichas variables, haciendo evidente que para cada objeto se muestra un valor diferente de cada una de ellas.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

2. TIPOS DE DATOS

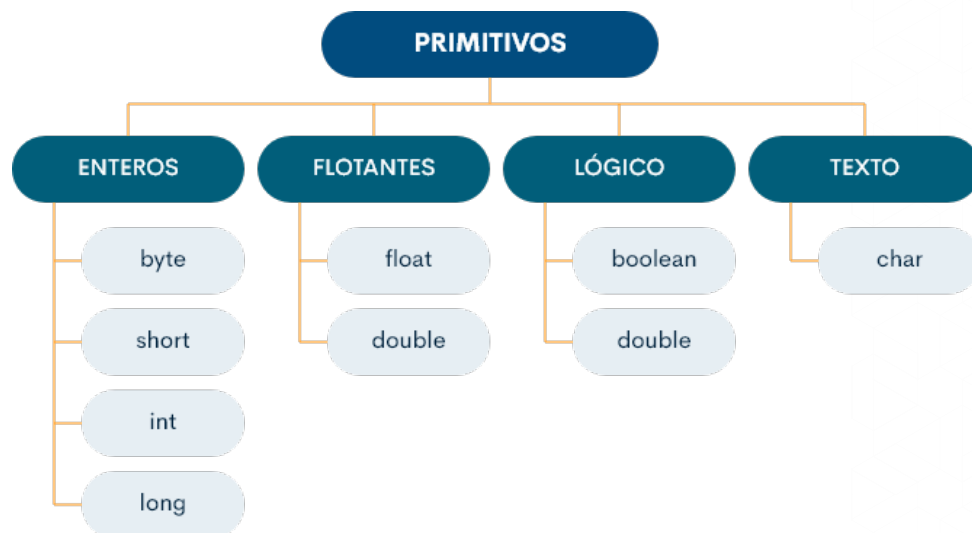


En la programación de software es necesario manipular y almacenar información, la cual es grabada en espacios temporales de memoria del computador; los tipos de datos consisten en la forma en que el lenguaje interpreta las variables de acuerdo con lo que estas contengan o puedan contener.

Cada uno de los tipos de datos tiene un nombre, y además está en capacidad de guardar información de cierta categoría dentro de un rango específico.

2.1 Primitivos

Los tipos de datos primitivos son aquellos que se encuentran predefinidos y están disponibles dentro del lenguaje. El lenguaje Java cuenta con cuatro tipos diferentes para representar números enteros, dos de números reales en coma flotante, uno para caracteres, y uno para valores lógicos. En la siguiente figura se pueden observar dichos tipos, clasificados según la información que almacenan.





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

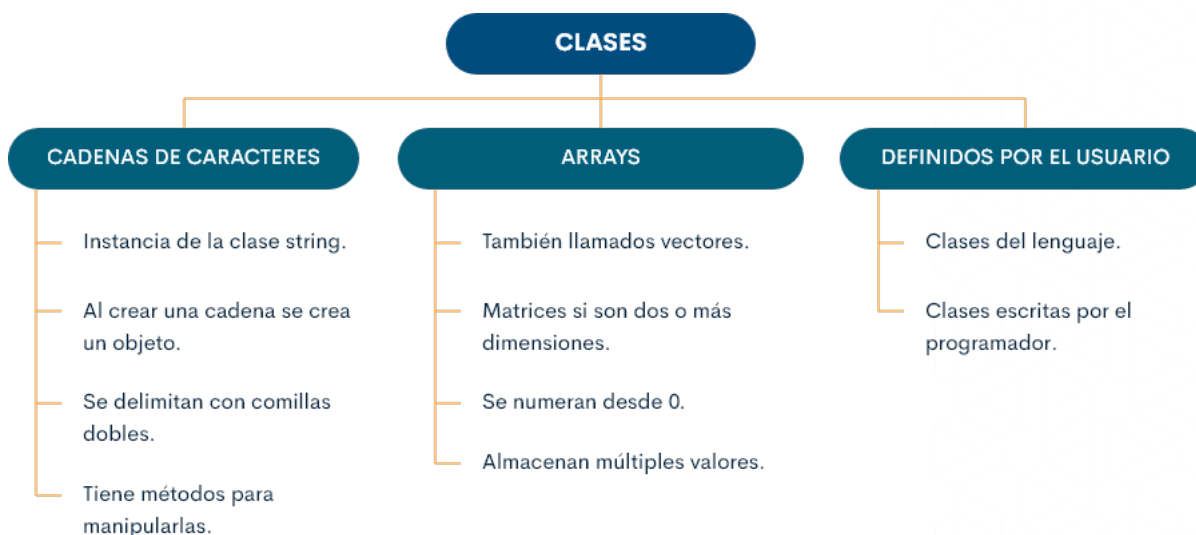
En la siguiente tabla se ilustran los tipos de datos primitivos, sus valores mínimo y máximo y valores por defecto.

TIPO	VALOR MÍNIMO	VALOR MÁXIMO	VALOR POR DEFECTO
byte	-128	128	0
short	-32768	32768	0
int	-2147483648	2147483648	0
long	-9223372036854775808	9223372036854775808	0
float	$\pm 3.4028235 \times 10^{38}$	$\pm 1.4023984 \times 10^{-45}$	0.0
double	$\pm 1.79769313486231570 \times 10^{308}$	$\pm 4.94065645841246544 \times 10^{-324}$	0.0
boolean	true o false	---	false

A manera de ejemplo y para comprender mejor lo que se menciona respecto a los valores mínimo y máximo, puede decirse que en Java un dato de tipo int no podría almacenar el valor de tres mil millones (3.000'000.000) dado que su tope máximo es de dos mil ciento cuarenta y siete millones cuatrocientos ochenta y tres mil seiscientos cuarenta y siete (2.147'483.647).

2.2 Clases

También llamadas tipos estructurados, usualmente contienen múltiples valores y se utilizan en la representación de objetos. En la siguiente figura se pueden observar dichos tipos.





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

3. OPERACIONES BÁSICAS

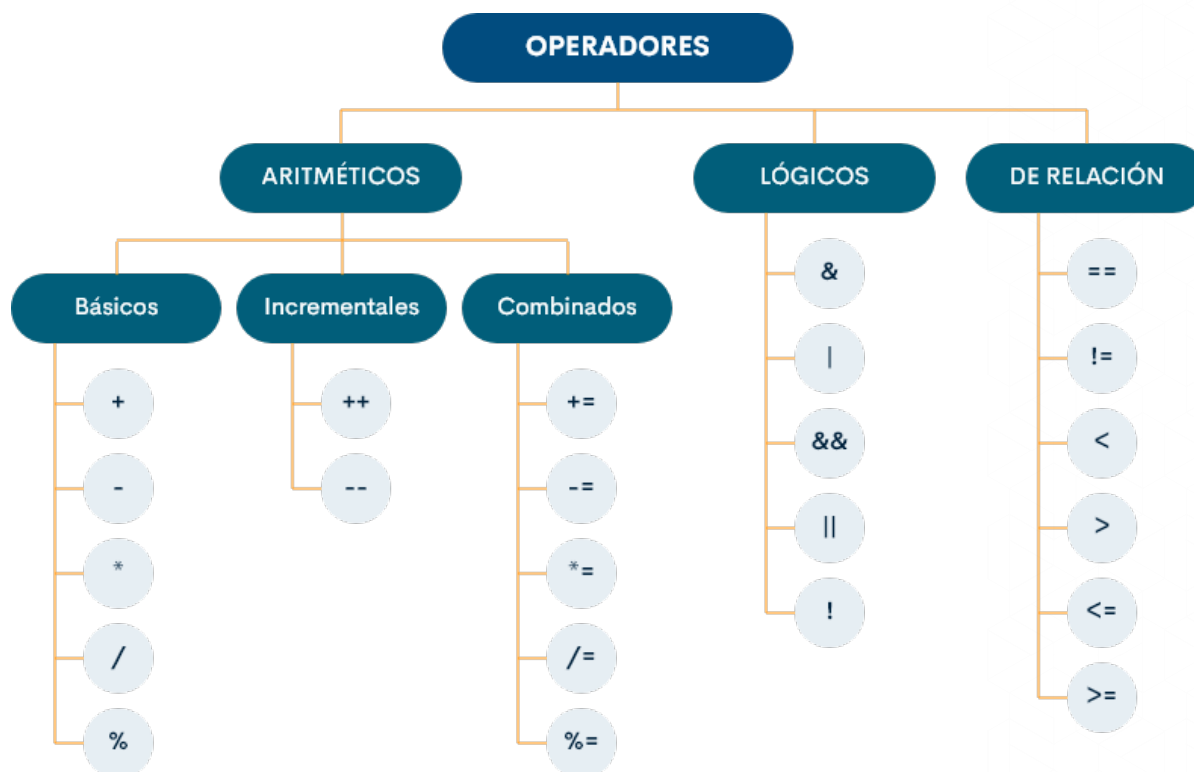


Es usual en la programación de aplicaciones la realización de cálculos matemáticos, los cuales se encuentran normalmente basados en las operaciones aritméticas básicas, y su complejidad depende de las diferentes expresiones creadas por el programador.

Para que las secuencias de programación sean correctas y devuelvan al usuario el resultado esperado, es necesaria la utilización de distintos operadores, y se debe prestar especial atención al orden en que se evalúan las diferentes operaciones mostradas.

3.1 Operadores

En la siguiente figura se observan los operadores utilizados.





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

1. Aritméticos

Como su nombre lo indica, se utilizan para realizar operaciones aritméticas con las variables de tipo numérico; en el caso de Java, se clasifican en tres categorías que son: básicos, incrementales y combinados.

» Básicos

+

Suma. Se utiliza para sumar los valores contenidos en las variables.

-

Resta. Se emplea para restar los valores contenidos en las variables.

*

Multiplicación. Se usa para multiplicar los valores contenidos en las variables.

/

División. Se maneja para dividir los valores contenidos en las variables.

%

Resto de la división. se destina para devolver el residuo de una división entera.

En la siguiente figura se observa un ejemplo de utilización de los operadores aritméticos básicos:

```
public class Main {  
    public static void main (String[] args) {  
        int a = 20, b = 10, c = 0, d = 20, e = 40, f = 30;  
        System.out.println("c + d = " + (c+d));  
        System.out.println("e - f = " + (c+d));  
        System.out.println("b x a = " + (c+d));  
        System.out.println("f / b = " + (c+d));  
    }  
}
```

En la línea de código 3 del ejemplo anterior, se declaran las variables de tipo entero requeridas para realizar operaciones con los operadores aritméticos básicos; posteriormente de las líneas 5 a la 8 se utilizan dichas variables para imprimir la suma, resta, multiplicación y división entre estas.

» Salida:

```
run:  
c + d = 20  
e - f = 10  
b x a = 200  
f / b = 3
```



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

En la anterior figura, se observa la ejecución de dicho código.



NOTA: ejecutar el IDE NetBeans y llevar a cabo todos los ejemplos vistos en el contenido de este documento con el fin de adquirir práctica en la escritura de las secuencias de código en el lenguaje Java.

» Incrementales

++

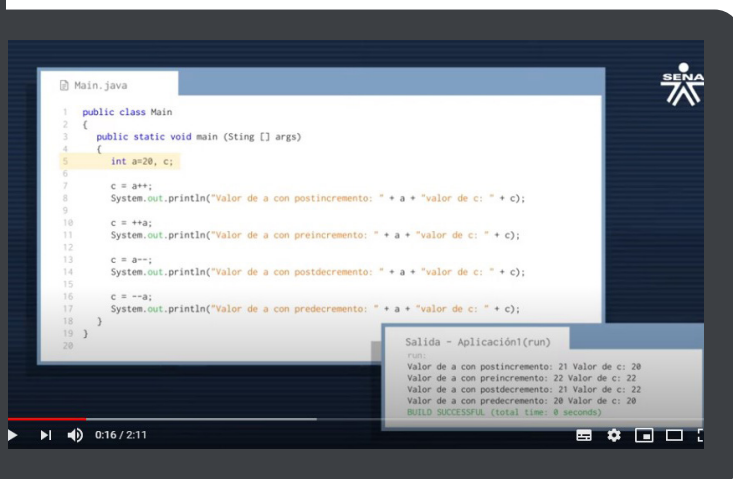
Incremento. Incrementa en 1 el valor de una variable.

--

Decremento. Decrementa en 1 el valor de una variable.

Según la ubicación del operador, la variable se utiliza y luego se incrementa (operador después de la variable **variable++**) o se incrementa y luego se utiliza (operador antes de la variable **++variable**).

A continuación se observa un ejemplo de utilización de los operadores aritméticos incrementales.



```

1 public class Main
2 {
3     public static void main (String [] args)
4     {
5         int a=20, c;
6
7         c = a++;
8         System.out.println("Valor de a con postincremento: " + a + "valor de c: " + c);
9
10        c = ++a;
11        System.out.println("Valor de a con preincremento: " + a + "valor de c: " + c);
12
13        c = a--;
14        System.out.println("Valor de a con postdecremento: " + a + "valor de c: " + c);
15
16        c = --a;
17        System.out.println("Valor de a con predecremento: " + a + "valor de c: " + c);
18    }
19 }
20

```

Salida - Aplicación1(run)

```

run
Valor de a con postincremento: 21 Valor de c: 20
Valor de a con preincremento: 22 Valor de c: 22
Valor de a con postdecremento: 21 Valor de c: 22
Valor de a con predecremento: 20 Valor de c: 20
BUILD SUCCESSFUL (total time: 0 seconds)

```



CLIC PARA
VER VIDEO

» Combinados

+=

Suma combinada. Asigna a la variable 1 la suma de los valores de las variables 1 y 2.

-=

Resta combinada. Designa a la variable 1 la diferencia entre los valores de las variables 1 y 2.

*=

Producto combinado. Establece a la variable 1 la multiplicación de los valores de las variables 1 y 2.

/=

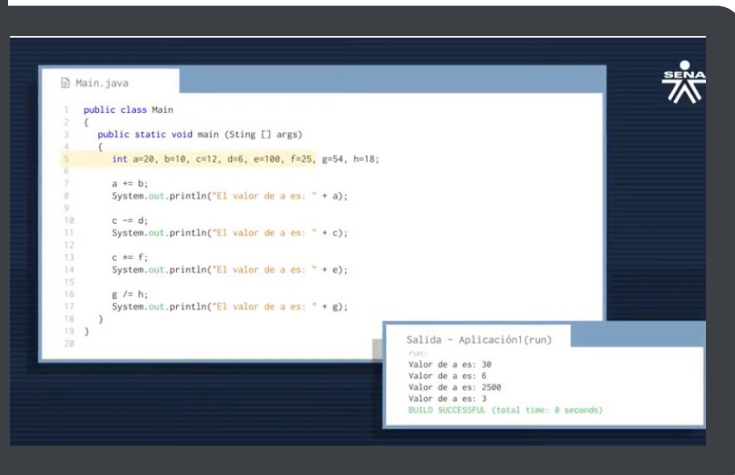
División combinada. Asigna a la variable 1 la división entre los valores de las variables 1 y 2.

%=

Resto combinado. Designa a la variable 1 el residuo de los valores de las variables 1 y 2.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



```

Main.java
1 public class Main
2 {
3     public static void main (String [] args)
4     {
5         int a=20, b=10, c=12, d=6, e=100, f=25, g=54, h=18;
6
7         a += b;
8         System.out.println("El valor de a es: " + a);
9
10        c -= d;
11        System.out.println("El valor de a es: " + c);
12
13        c *= f;
14        System.out.println("El valor de a es: " + e);
15
16        g /= h;
17        System.out.println("El valor de a es: " + g);
18    }
19 }
20
Salida - Aplicación(run)
run:
Valor de a es: 30
Valor de a es: 6
Valor de a es: 2500
Valor de a es: 3
BUILD SUCCESSFUL (total time: 0 seconds)
    
```

A continuación se observa un ejemplo de utilización de los operadores aritméticos combinados.

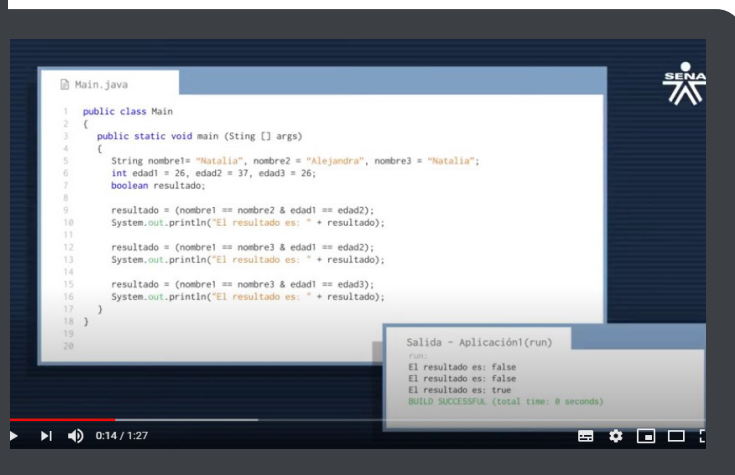


CLIC PARA
VER VIDEO

2. Lógicos

Los operadores lógicos, como su nombre lo indica, permiten construir expresiones lógicas.

- &** **AND lógico.** Evalúa condiciones y devuelve TRUE cuando todas se cumplen.
- |** **OR lógico.** Evalúa condiciones y devuelve TRUE cuando al menos una se cumple.
- &&** **AND en cortocircuito.** Evalúa condiciones y si la primera no se cumple, devuelve FALSO y no evalúa las demás.
- ||** **OR en cortocircuito.** Evalúa condiciones y si la primera se cumple, devuelve VERDADERO y no evalúa las demás.
- !** **Negación.** Niega el valor enviado.



```

Main.java
1 public class Main
2 {
3     public static void main (String [] args)
4     {
5         String nombre1 = "Natalia", nombre2 = "Alejandra", nombre3 = "Natalia";
6         int edad1 = 26, edad2 = 37, edad3 = 26;
7         boolean resultado;
8
9         resultado = (nombre1 == nombre2 & edad1 == edad2);
10        System.out.println("El resultado es: " + resultado);
11
12        resultado = (nombre1 == nombre3 & edad1 == edad2);
13        System.out.println("El resultado es: " + resultado);
14
15        resultado = (nombre1 == nombre3 & edad1 == edad3);
16        System.out.println("El resultado es: " + resultado);
17    }
18 }
19
20
Salida - Aplicación(run)
run:
El resultado es: false
El resultado es: false
El resultado es: true
BUILD SUCCESSFUL (total time: 0 seconds)
    
```

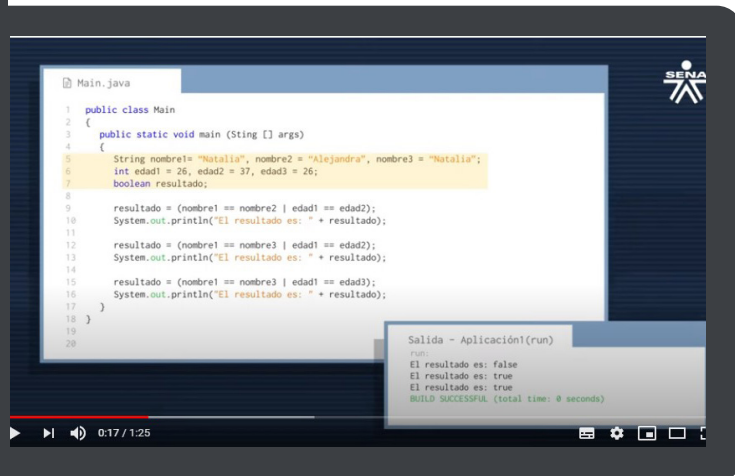
A continuación se observa un ejemplo de utilización del operador lógico AND (&).



CLIC PARA
VER VIDEO



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



```

1 public class Main
2 {
3     public static void main (String [] args)
4     {
5         String nombre1 = "Natalia", nombre2 = "Alejandra", nombre3 = "Natalia";
6         int edad1 = 26, edad2 = 37, edad3 = 26;
7         boolean resultado;
8
9         resultado = (nombre1 == nombre2 | edad1 == edad2);
10        System.out.println("El resultado es: " + resultado);
11
12        resultado = (nombre1 == nombre3 | edad1 == edad2);
13        System.out.println("El resultado es: " + resultado);
14
15        resultado = (nombre1 == nombre3 | edad1 == edad3);
16        System.out.println("El resultado es: " + resultado);
17    }
18 }
19
20

```

Salida - Aplicación(run)

```

run:
El resultado es: false
El resultado es: true
El resultado es: true
BUILD SUCCESSFUL (total time: 0 seconds)

```

A continuación se observa un ejemplo de utilización del operador lógico OR.



CLIC PARA
VER VIDEO



A continuación se observa un ejemplo de utilización del operador lógico AND (&&) y OR (||) en cortocircuito.



CLIC PARA
VER VIDEO

3. De relación

Los operadores de relación permiten comparar datos de tipos primitivos; asimismo devuelven un valor booleano o ejecutan las acciones requeridas por el programador, según la comparación efectuada.

- == Igual a.** Evalúa si una variable contiene el mismo valor que la otra.
- != Diferente a.** Evalúa si una variable contiene un valor diferente que la otra.
- < Menor que.** Evalúa si una variable contiene un valor menor que la otra.
- > Mayor que.** Evalúa si una variable contiene un valor mayor que la otra.
- <= Menor o igual que.** Evalúa si una variable contiene un valor menor o igual que la otra.
- >= Mayor o igual que.** evalúa si una variable contiene un valor mayor o igual que la otra.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



A continuación se observa un ejemplo de utilización de los operadores aritméticos de relación.



CLIC PARA
VER VIDEO

3.2 Jerarquía

En Java se debe tener en cuenta la jerarquía o el orden en que son evaluadas las expresiones matemáticas, según los operadores que estas utilicen; es importante también el empleo del paréntesis para agrupar subexpresiones con el fin de indicar que sean evaluadas primero.

Al igual que en las expresiones algebraicas, los paréntesis son utilizados con el fin de agrupar términos. Por ejemplo, si se desea multiplicar x por el resultado de sumar $y + z$ debe escribirse así:

$$x * (y + z)$$

Se sumaría entonces $x + y$ para posteriormente multiplicar el resultado por z .

Con respecto a la jerarquía que debe analizarse en los operadores, se tienen dos reglas básicas que son las siguientes:

- » Deben realizarse primero las operaciones de multiplicación, división y residuo; en caso de que la expresión contenga varias, se efectúan de izquierda a derecha.
- » Posteriormente se efectúan las operaciones de suma y resta, y de igual manera, si son varias, se aplican de izquierda a derecha.

En la siguiente tabla se puede observar un resumen de lo anterior y se incluyen los operadores combinados y el de asignación que deben ser evaluados en el último orden.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

OPERADOR	OPERACIÓN	ORDEN
*	Multiplicación	Primero
/	División	Primero
%	Residuo	Primero
+	Suma	Segundo
-	Resta	Segundo
=	Asignación	Tercero
+=	Suma combinada	Tercero
-=	Resta combinada	Tercero
*=	Producto combinado	Tercero
/=	División combinada	Tercero
%=	Resto combinado	Tercero

Para comprender y aplicar la jerarquía de las operaciones, se observan a continuación una serie de ejemplos con expresiones escritas teniendo en cuenta la sintaxis del lenguaje Java.



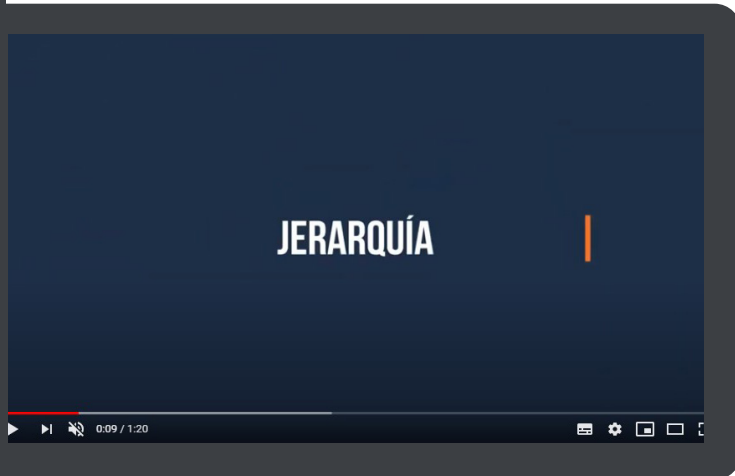
Ejemplo 1. Promedio entre 4 términos

Promedio = (termino1 + termino2 + termino3 + termino4) / 4;

En la expresión previa es obligatorio utilizar los paréntesis ya que, en caso de omitirlos, el resultado devuelto sería totalmente diferente debido a la jerarquía, y por ende la primera operación a efectuar es la división.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



En el siguiente video se observa el ejemplo anterior realizado con y sin paréntesis en NetBeans.



CLIC PARA
VER VIDEO



Ejemplo 2. Hallar el valor de "y" en el polinomio de segundo grado

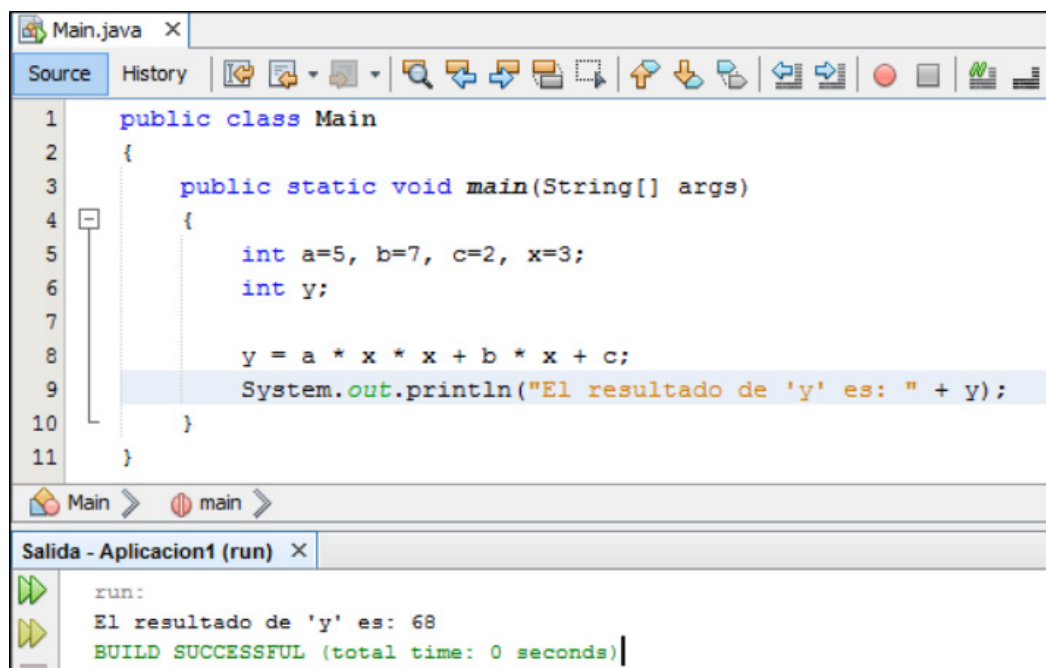
$$y = ax^2 + bx + c;$$

donde $a = 5$, $b = 7$, $c = 2$, $x = 3$

$$y = a * x * x + b * x + c;$$

En la expresión previa no es necesario utilizar paréntesis, únicamente se debe aplicar la jerarquía de las operaciones, efectuando de izquierda a derecha primero las multiplicaciones y luego las sumas.

En la siguiente figura se observa el ejemplo anterior realizado en NetBeans.



```

Main.java x
Source History
1 public class Main
2 {
3     public static void main(String[] args)
4     {
5         int a=5, b=7, c=2, x=3;
6         int y;
7
8         y = a * x * x + b * x + c;
9         System.out.println("El resultado de 'y' es: " + y);
10    }
11 }

Main main
Salida - Aplicacion1 (run) x
run:
El resultado de 'y' es: 68
BUILD SUCCESSFUL (total time: 0 seconds)
    
```

SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

En las líneas 5 y 6 del ejemplo anterior se declaran las variables requeridas para calcular el valor de y. Posteriormente se realiza lo siguiente:

- » En la línea 8 se escribe la expresión que es evaluada de izquierda a derecha; respetando la jerarquía, se realiza inicialmente la multiplicación que aparece de primera a la izquierda $a * x$ que al reemplazar es $5 * 3 = 15$, luego se tendría $15 * x + b * x + c$ y se continúa entonces de nuevo con la primera multiplicación de la izquierda $15 * x$ que reemplazado es $15 * 3 = 45$. A continuación se tiene $45 + b * x + c$ y el siguiente paso es la multiplicación $45 + b * 3 + c$, reemplazando $45 + 7 * 3 + c$ que quedaría $45 + 21 + c$, luego se efectúan las sumas, iniciando también con la primera de la izquierda $45 + 21 = 66$ quedando finalmente con $66 + c$ que si se reemplaza es $66 + 2 = 68$, lo cual corresponde con el valor devuelto por la línea 9.
- » En caso de que la jerarquía no fuera respetada y las operaciones se efectuarán tal como aparecen, de izquierda a derecha, sin tener en cuenta la precedencia de los operadores, el proceso sería: $5 * 3 = 15 * 3 = 45 + 7 = 52 * 3 = 156 + 2 = 158$; para Java esto no es correcto, ya que el lenguaje respeta dicha jerarquía.

Ejemplo 3. Hallar el valor de "b" de acuerdo con la fórmula



$$b = 2 / ((x+y) / (z/a)) + (g/h) / e + 3cdf$$

donde $x=5, y=6, z=3, a=4, c=1, d=7, e=6, f=3, g=2, h=3$

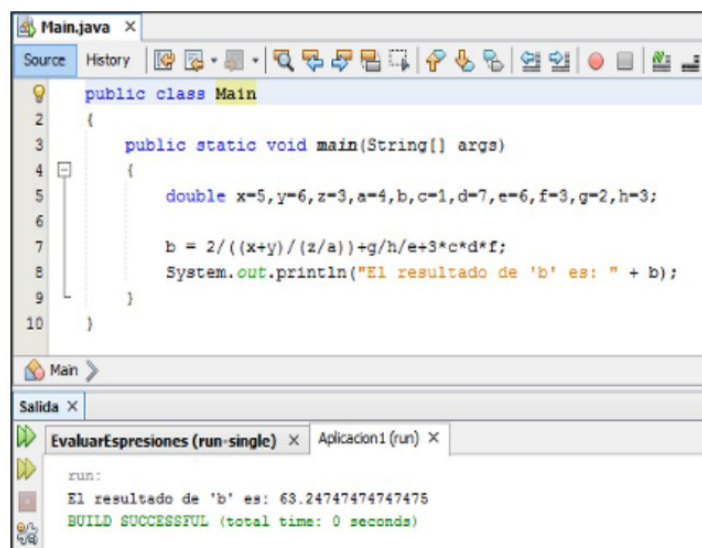
La expresión escrita en forma horizontal para que sea entendible por Java sería:

$$b = 2 / ((x + y) / (z / a)) + g / h / e + 3 * c * d * f$$

En la expresión previa no es necesario utilizar paréntesis, únicamente se debe aplicar la jerarquía de las operaciones, efectuando de izquierda a derecha primero las multiplicaciones y luego las sumas.

En la expresión previa es preciso emplear los paréntesis con el fin de que sea más clara; para resolverla, además de la jerarquía de las operaciones, se debe tener en cuenta que se comienza evaluando los paréntesis más internos.

En la siguiente figura se observa el ejemplo anterior realizado en NetBeans.





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

En la línea 5 del ejemplo anterior se declaran las variables requeridas para hallar el valor de *b* de acuerdo con la fórmula, y posteriormente se realiza lo siguiente:

- » En la línea 7 se escribe la expresión y en la línea 8 se imprime el resultado devuelto.
- » El proceso de evaluación de dicha expresión se realiza en el orden visto en la siguiente tabla.

PASO	ACCIÓN	VALORES	RESULTADO
1	Sumar "x" más "y"	5 + 6	11
2	Dividir "z" entre "a"	3 / 4	0,75
3	Dividir el resultado del paso 1 entre el resultado del paso 2	11 / 0,75	14,66666667
4	Dividir 2 entre el resultado del paso 3	2 / 14,66666667	0,136363636
5	Dividir "g" entre "h"	2 / 3	0,666666667
6	Dividir el resultado del paso 5 entre "e"	0,666666667 / 6	0,11111111
7	Multiplicar 3 por "c"	3 * 1	3
8	Multiplicar el resultado del paso 7 por "d"	3 * 7	21
9	Multiplicar el resultado del paso 8 por "f"	21 * 3	63
10	Sumar el resultado del paso 4 más el resultado del paso 6	0,136363636 + 0,11111111	0,247474747
11	Sumar el resultado del paso 10 más el resultado del paso 9	0,247474747 + 63	63,24747475
12	Asignar a "b" el resultado del paso 11	b = 63,24747475	---

4. FUNCIONES



Las funciones en Java también son conocidas como métodos y deben definirse dentro de las clases. Consisten en una serie de líneas de código que utilizan parámetros para realizar acciones.

Pese a lo anterior, los conceptos de método y función tienen como diferencia que un método no devuelve nada, en cambio una función sí lo hace.

Crear funciones es de gran importancia en el desarrollo de soluciones, ya que permiten separar el código de acuerdo con los requerimientos, implementando cada utilidad del programa en un método independiente.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

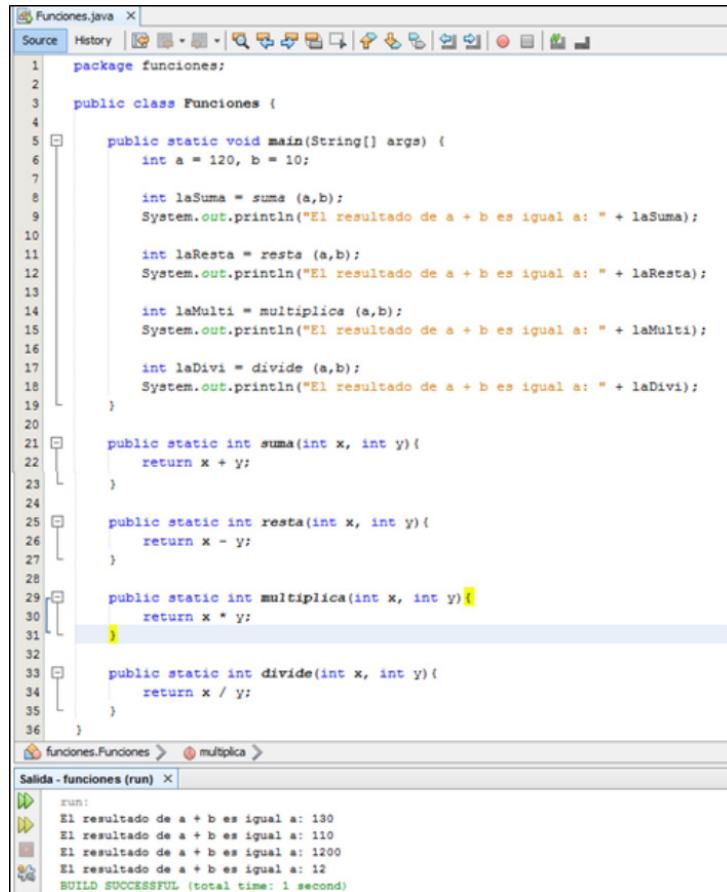
La sintaxis para declarar una función es la siguiente:

```
[ acceso ] [ modificador ] tipo nombre ( tipoParametro1...N nombreParametro1...N )
{
    Instruccion1...
    Instruccion2...
    InstruccionN...
    return valor;
}
```

- » **acceso:** corresponde al modificador de acceso y permite controlar el acceso a los datos que componen un objeto; puede ser public, protected, private o default.
- » **modificador:** es opcional y puede tomar los valores final o static.
- » **tipo:** corresponde al tipo de dato del valor que la función va a retornar; puede ser int, float, booleano o cualquiera de los utilizados según se requiera.
- » **nombre:** nombre que identifica la función para invocarla.
- » **tipoParametro:** el tipo de dato que recibe la función como parámetro.
- » **nombreParametro:** el nombre del parámetro recibido.
- » **instruccion1... instruccionN:** sentencias ejecutadas al invocar la función.
- » **return resultado:** valor devuelto por la función.

Para comprender mejor esta temática, se observa en la siguiente imagen una aplicación creada por NetBeans, que por medio de funciones recibe dos números enteros y realiza con estos las cuatro operaciones matemáticas básicas.

SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



```
1 package funciones;
2
3 public class Funciones {
4
5     public static void main(String[] args) {
6         int a = 120, b = 10;
7
8         int laSuma = suma(a,b);
9         System.out.println("El resultado de a + b es igual a: " + laSuma);
10
11        int laResta = resta(a,b);
12        System.out.println("El resultado de a + b es igual a: " + laResta);
13
14        int laMulti = multiplica(a,b);
15        System.out.println("El resultado de a + b es igual a: " + laMulti);
16
17        int laDivi = divide(a,b);
18        System.out.println("El resultado de a + b es igual a: " + laDivi);
19    }
20
21    public static int suma(int x, int y){
22        return x + y;
23    }
24
25    public static int resta(int x, int y){
26        return x - y;
27    }
28
29    public static int multiplica(int x, int y){
30        return x * y;
31    }
32
33    public static int divide(int x, int y){
34        return x / y;
35    }
36 }
```

funciones.Funciones > multiplica >

Salida - funciones (run) X

run:

El resultado de a + b es igual a: 130
El resultado de a + b es igual a: 110
El resultado de a + b es igual a: 1200
El resultado de a + b es igual a: 12
BUILD SUCCESSFUL (total time: 1 second)

A partir de la línea 21 se declararon las cuatro funciones encargadas de sumar, restar, multiplicar y dividir los dos parámetros que les sean enviados.

Dentro del main, en la línea 6 se declaran las variables que son enviadas como parámetro a cada función, y a partir de la línea 8, también dentro del main, se invocan las funciones y se imprimen los resultados devueltos.

A simple vista pareciera que haber escrito todo el código dentro del main fuese más sencillo; sin embargo si se piensa en el alcance de los programas, la escritura de las rutinas de programación sin utilizar funciones, se torna desordenada y poco funcional, ya que en soluciones de alta envergadura se requiere efectuar acciones repetitivas que son facilitadas con el uso de funciones; pudiendo de esta manera invocarlas en distintas partes, sin que sea necesario repetir la escritura del código completo.



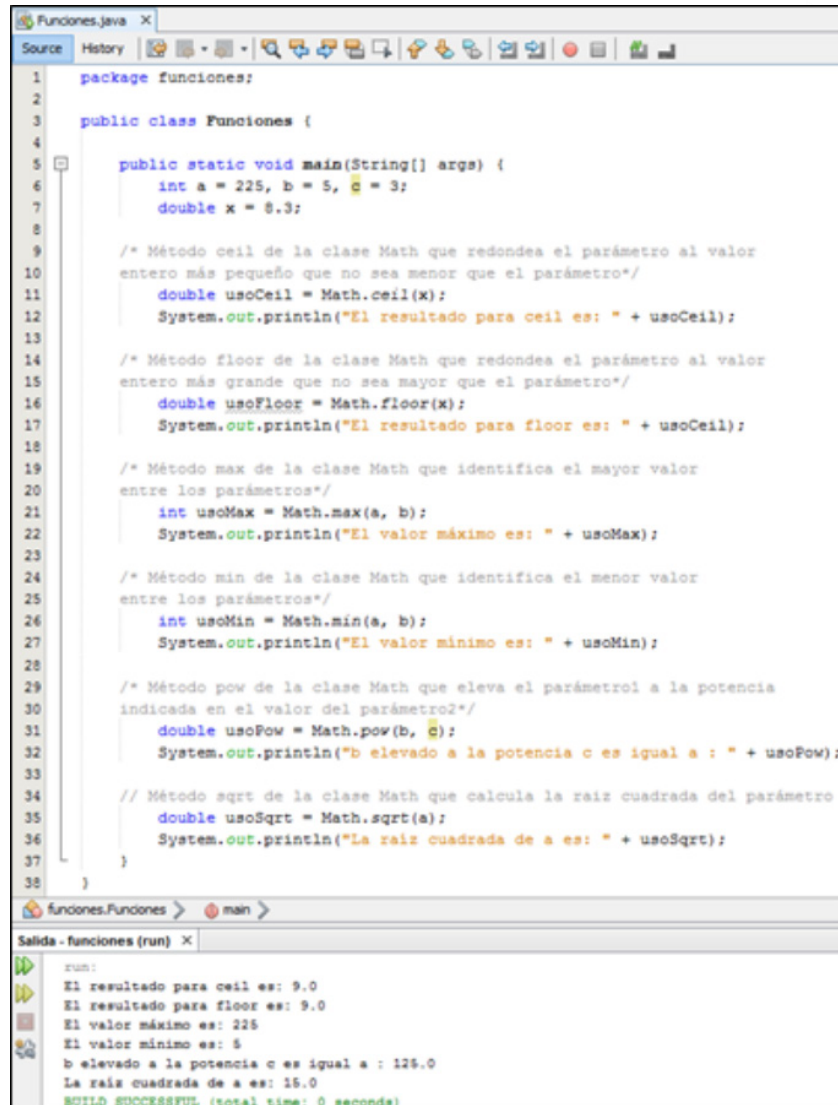
El lenguaje JAVA cuenta con una serie de métodos que ya han sido declarados con anterioridad y por lo tanto para utilizarlos basta con invocarlos con sus respectivos argumentos; varios de esos métodos corresponden a la **clase math**.

Adicionalmente, en la clase math también se encuentra una constante de uso común que es el valor de PI (conocido usualmente como 3,1416), el cual se puede utilizar así: **Math.PI**.



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

En la siguiente imagen se observa un ejemplo que utiliza algunos de los métodos de la clase `math`.



```
1 package funciones;
2
3 public class Funciones {
4
5     public static void main(String[] args) {
6         int a = 225, b = 5, c = 3;
7         double x = 8.3;
8
9         /* Método ceil de la clase Math que redondea el parámetro al valor
10          entero más pequeño que no sea menor que el parámetro*/
11         double usoCeil = Math.ceil(x);
12         System.out.println("El resultado para ceil es: " + usoCeil);
13
14         /* Método floor de la clase Math que redondea el parámetro al valor
15          entero más grande que no sea mayor que el parámetro*/
16         double usoFloor = Math.floor(x);
17         System.out.println("El resultado para floor es: " + usoCeil);
18
19         /* Método max de la clase Math que identifica el mayor valor
20          entre los parámetros*/
21         int usoMax = Math.max(a, b);
22         System.out.println("El valor máximo es: " + usoMax);
23
24         /* Método min de la clase Math que identifica el menor valor
25          entre los parámetros*/
26         int usoMin = Math.min(a, b);
27         System.out.println("El valor mínimo es: " + usoMin);
28
29         /* Método pow de la clase Math que eleva el parámetro1 a la potencia
30          indicada en el valor del parámetro2*/
31         double usoPow = Math.pow(b, c);
32         System.out.println("b elevado a la potencia c es igual a : " + usoPow);
33
34         // Método sqrt de la clase Math que calcula la raíz cuadrada del parámetro
35         double usoSqrt = Math.sqrt(a);
36         System.out.println("La raíz cuadrada de a es: " + usoSqrt);
37     }
38 }
```

Salida - funciones (run) X

```
run:
El resultado para ceil es: 9.0
El resultado para floor es: 9.0
El valor máximo es: 225
El valor mínimo es: 5
b elevado a la potencia c es igual a : 125.0
La raíz cuadrada de a es: 15.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. ENTRADA DE INFORMACIÓN



Hasta el momento, en todos los ejercicios vistos se ha utilizado dentro de las líneas de código la declaración e inicialización de las variables que van a ser manipuladas; la entrada de información consiste en solicitar al usuario por pantalla que ingrese los datos deseados para el contenido de dichas variables; esta funcionalidad proporciona a las soluciones un interesante ingrediente de interactividad porque las operaciones o manipulación de información son realizadas internamente por el programa, pero es el usuario final quien define con qué datos o valores se debe trabajar.

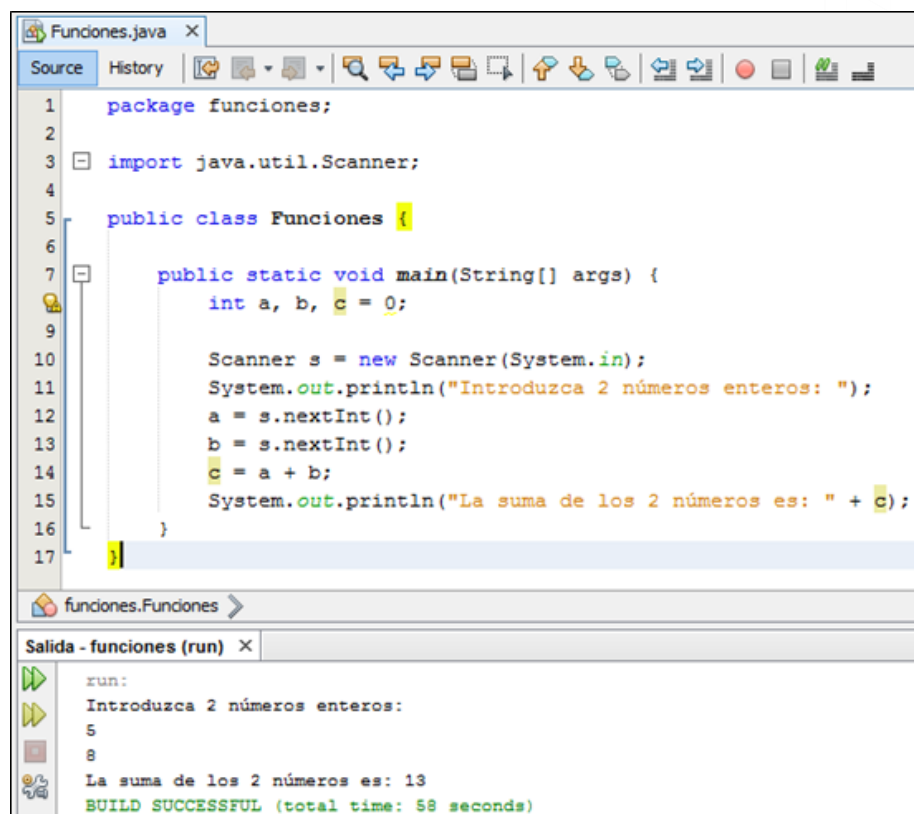


SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

A continuación, se ejemplifica la entrada de datos por teclado de dos maneras; primero escribiendo el código directamente en el main e ingresando los datos por la consola de NetBeans y luego creando un formulario que, al ser ejecutado, solicite la información en diferentes campos de texto.

Al realizar la ejecución del código visto en la siguiente figura, lo primero que se observa es el texto Introduzca 2 números enteros: con el cual el programa está solicitando al usuario que ingrese por teclado dichos valores; el usuario debe digitar cada uno seguido de la tecla Enter y posterior a ello, el sistema le devuelve el resultado, el cual para el caso ejemplificado es: La suma de los 2 números es: 13.

NOTA: La línea 3 del ejemplo visto en la siguiente figura se utiliza para que Java permita leer los valores por parte del usuario.



```
1 package funciones;
2
3 import java.util.Scanner;
4
5 public class Funciones {
6
7     public static void main(String[] args) {
8         int a, b, c = 0;
9
10        Scanner s = new Scanner(System.in);
11        System.out.println("Introduzca 2 números enteros: ");
12        a = s.nextInt();
13        b = s.nextInt();
14        c = a + b;
15        System.out.println("La suma de los 2 números es: " + c);
16    }
17 }
```

funciones.Funciones

Salida - funciones (run) X

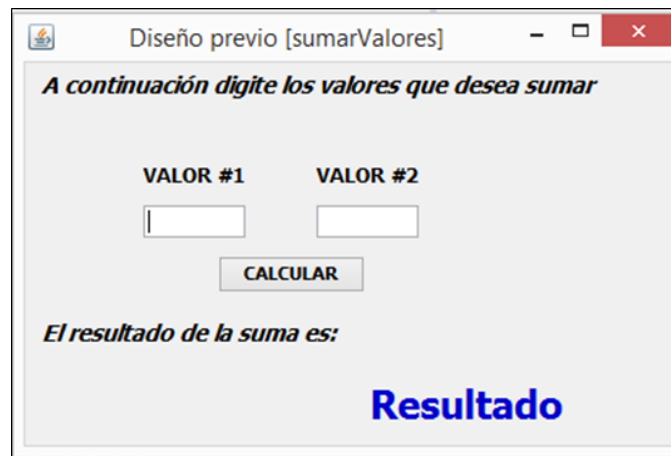
```
run:
Introduzca 2 números enteros:
5
8
La suma de los 2 números es: 13
BUILD SUCCESSFUL (total time: 58 seconds)
```

Para mostrar el ejemplo de ingreso de datos en un formulario, es necesario acceder a NetBeans y crear dicho formulario, arrastrando los controles requeridos para después efectuar la programación de los mismos.

Con el fin de resolver el mismo ejercicio visto en la imagen anterior, pero utilizando el otro método, la interfaz del formulario sería la que se ilustra a continuación:



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



Una vez creada la interfaz, se da doble clic en el botón para programar las acciones que deben ser realizadas cuando el usuario haya digitado los números y oprima dicho botón.

Para escribir esas líneas de código, es importante que al crear la interfaz gráfica, se haya asignado a cada control un nombre reconocido; esto facilitará la escritura de los segmentos de código respectivos.

En el ejemplo, los nombres asignados fueron los siguientes:

```
Campo de texto valor1 = JTextValor1  
Campo de texto valor2 = JTextValor2  
Label para mostrar el resultado = JLabelResultado
```

El código que hay que escribir para generar la suma se observa a continuación:

```
int numero1 = 0, numero2 = 0, resultado = 0;  
numero1 = Integer.parseInt(JTextValor1.getText());  
numero2 = Integer.parseInt(JTextValor2.getText());  
resultado = numero1 + numero2;  
JLabelResultado.setText("" + resultado);
```

Los pasos para la generación de dicho código son los siguientes:

- » Se declaran las variables que van a contener el resultado final y ambos números.
- » Se asigna a la variable **numero1** el contenido del campo de texto llamado **jTextValor1**; esto se realiza utilizando **Integer.parseInt** dado que el contenido de dicho campo de texto es de tipo **texto** y la

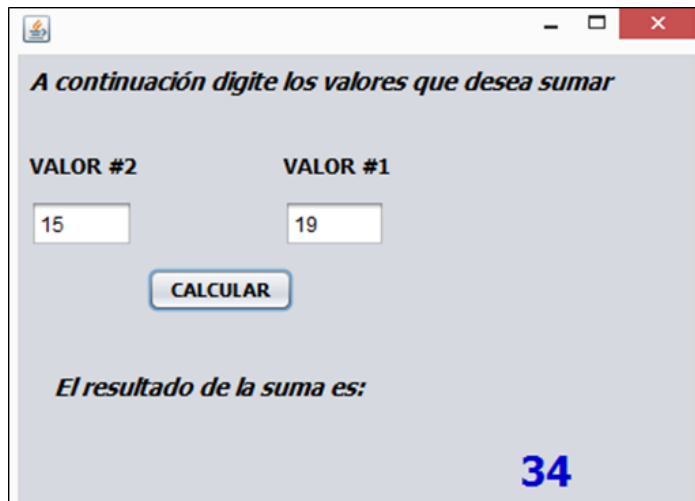


SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

variable **numero1** fue declarada de tipo **int**, por lo tanto, es necesario hacer la conversión y de esto se encarga **Integer.parseInt**. Posteriormente se usa **getText()** que permite obtener el valor de la caja de texto.

- » De la misma manera, se asigna a la variable **numero2** el contenido del campo de texto llamado **jTextValor2**.
- » Ahora se asigna a la variable **resultado** la suma de las variables **numero1** y **numero2**.
- » Finalmente se asigna al label **jLabelResultado** el valor de la variable **resultado** y esto se realiza a través de **setText()** que permite asignar un valor al label; adicionalmente se deben concatenar unas comillas para efectuar de nuevo la conversión, dado que la variable **resultado** tiene un dato tipo **int** y el label para mostrar la información maneja el dato de tipo texto..

El formulario tras ser ejecutado en NetBeans con los valores 15 y 19, devuelve el resultado visto en la siguiente figura:



A continuación digite los valores que desea sumar

VALOR #2 VALOR #1

15 19

CALCULAR

El resultado de la suma es:

34



SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

GLOSARIO

Expresión: Línea de código compuesta por variables y operadores.

Interfaz: Punto de acceso gráfico de un programa.

Jerarquía: Orden en el cual deben ser evaluadas las distintas operaciones de una expresión.

Main: Método principal de Java que actúa como punto de acceso para los programas.

Operador: Carácter utilizado para efectuar diferentes acciones sobre las variables.

Tipo de dato: Nombre que identifica la clase de información que puede almacenar una variable.

Variable: Espacio de memoria utilizado para almacenar un dato.





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA



REFERENCIAS BIBLIOGRÁFICAS

NetBeans. (2019). NetBeans IDE 8.2 Download. <https://netbeans.org/downloads/8.2/>





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

CRÉDITOS

Equipo Contenido Instruccional

» Gloria Matilde Lee Mejía	Responsable equipo	Centro de Comercio y Servicios - Regional Tolima
» Rafael Nelftali Lizcano Reyes	Asesor pedagógico	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» Lucero Montes Arenas	Gestora de Desarrollo de Programas	Centro para la Formación Cafetera (Caldas)
» Julio Alexander Rodriguez Del Castillo	E-pedagogo instruccional	Centro Atención Sector Agropecuario Regional Risaralda
» Rachman Bustillo Martínez	Evaluador de contenido	Centro Atención Sector Agropecuario Regional Risaralda
» Natalia Andrea Bueno Pizarro	Diseñadora instruccional	Centro de Diseño y Metrología - Regional Distrito Capital

Equipo Diseño y Desarrollo

» Francisco José Lizcano Reyes	Responsable Equipo	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» Daniel Ricardo Mutis Gómez	Diagramación web	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» Yazmin Rocio Figueroa Pacheco	Construcción documentos digitales	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» Edgar Mauricio Cortes García	Desarrollo front-end	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» Luis Gabriel Urueta Álvarez	Desarrollo de actividades didácticas	Centro Industrial Del Diseño y La Manufactura - Regional Santander
» Leyson Fabian Castaño Pérez	Integración de recursos y pruebas	Centro Industrial Del Diseño y La Manufactura - Regional Santander





SINTAXIS DE LAS VARIABLES, TIPOS DE DATOS Y OPERACIONES EN JAVA

Equipo de Gestores de Repositorio

» Kely Alejandra Quiros Duarte

Administrador repositorio de contenidos y gestores de repositorio.

Centro de comercio y servicios - Regional Tolima

Recursos gráficos

Fotografías y vectores tomados de www.shutterstock.com y www.freepik.com

creative
commons



BY NC SA

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.

