

# I. Introducción a la programación orientada a objetos y al lenguaje JAVA



## Variables, Operaciones y Jerarquía en las Operaciones

En Java existen dos categorías de tipos de datos:

- Tipos Primitivos. Los cuales se definen abajo.
- Referencias. Las cuales corresponden a clases y objetos que se tratarán posteriormente.

### Tipos primitivos

Los tipos primitivos son los que permiten manipular valores numéricos (con distintos grados de precisión), caracteres y valores booleanos (verdadero / falso). Los Tipos Primitivos son:

- **boolean**: Puede contener los valores **true** o **false**.
- **byte**: Enteros. Tamaño 8-bits. Valores entre -128 y 127.
- **short**: Enteros. Tamaño 16-bits. Entre -32768 y 32767.
- **int**: Enteros. Tamaño 32-bits. Entre -2147483648 y 2147483647.
- **long**: Enteros. Tamaño 64-bits. Entre -9223372036854775808 y 9223372036854775807.
- **float**: Números en coma flotante. Tamaño 32-bits.
- **double**: Números en coma flotante. Tamaño 64-bits.
- **char**: Caracteres. Tamaño 16-bits. Unicode. Desde '\u0000' a '\uffff' inclusive. Esto es desde 0 a 65535

### Variables

Una variable es un área en memoria que tiene un nombre y un Tipo asociado. El Tipo es o bien un Tipo primitivo o una Referencia.

Es obligatorio declarar las variables antes de usarlas. Para declararlas se indica su nombre y su Tipo, de la siguiente forma:

*tipo\_variable nombre ;*

Ejemplos:

```
int i; // Declaracion de un entero
char letra; // Declaracion de un caracter
boolean flag; // Declaracion de un booleano
```

- El **;** es el separador de instrucciones en Java.
- El símbolo **//** indica comentarios de línea, se ponen después de una instrucción para comentarla, el compilador no las toma al detectarlas.
- En Java las mayúsculas y minúsculas son significativas. No es lo mismo el nombre

- letra que Letra.  
• Todas las palabras reservadas del lenguaje van en minúsculas.

Se pueden asignar valores a las variables mediante la instrucción de asignación (=). Por ejemplo:

```
i = 5;           // a la variable i se le asigna el valor 5
letra = 'c';     // a la variable letra se le asigna el valor 'c'
flag = false;    // a la variable flag se le asigna el valor false
```

La declaración y la asignación se pueden combinar en una sola expresión:

```
int i = 5;
char letra = 'c';
boolean flag = false;
```

### **Operaciones Básicas**

En java al igual que en C++ se tienen una serie de operadores que ayudan a obtener cálculos, dependiendo de los valores a utilizar, Java trabaja con los siguientes operadores:

### **Operadores Aritméticos**

Operador en Java	Significado
+	suma
-	resta
*	multiplicación
/	división
%	residuo

Todos los operadores que se muestran en esta tabla son binarios; es decir, trabajan con dos operandos.

Los operadores +, - y \* funcionan de la manera conocida.

El operador / funciona de diferente manera si trabaja con datos de tipo entero o de tipo flotante. Con datos de tipo flotante funciona de la manera tradicional; pero al realizarse una división entre dos números enteros, el operador / regresa el cociente de la división entera; es decir, regresa la parte entera del resultado (si hay fracción la elimina).

Por ejemplo:

2/3 da como resultado 0

pero

2.0/3.0 da como resultado 0.66666

Una manera de visualizar esto es a través de un applet que dibuje el resultado de una operación entera, en este caso utilizaremos el método drawString() como lo hicimos antes, y

hacemos uso del operador + que cuando es utilizado después de un String (cadena de caracteres) en Java funciona como concatenación.

Tenemos entonces que la siguiente codificación:

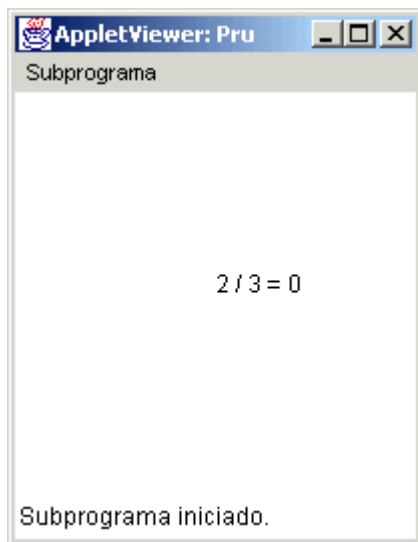
```
import java.awt.*;
import java.applet.*;

// <applet width="200" height="200" code="Pru"></applet>

public class Pru extends Applet {

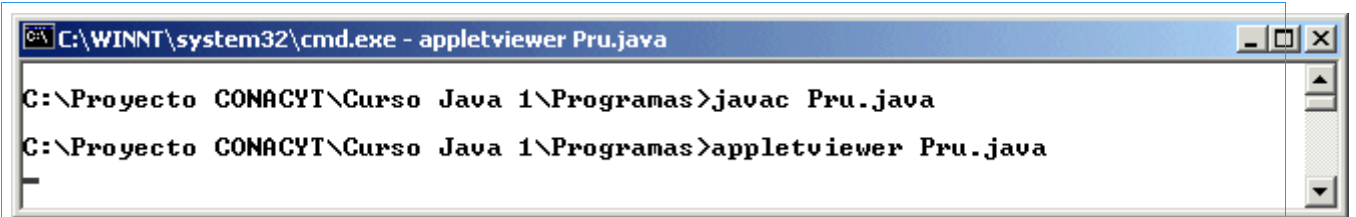
    public void paint(Graphics g) {
        int x;
        x = 2 / 3;
        g.drawString("2 / 3 = "+x, 100, 100);
    }
}
```

Dará como resultado:



Observa como utilizamos dentro del applet Pru.java el comentario `// <applet width="200" height="200" code="Pru"></applet>`

Esto lo hacemos para no tener que hacer un archivo de tipo HTML aparte, ya que un navegador o visualizador de applets lo que requiere para ejecutar el applet es solamente la directiva `<applet>`, hacemos uso de esta línea como comentario dentro del applet, de manera que al compilar no da un error por esta instrucción y al usar el visualizador de applets, se utiliza el mismo archivo, como se muestra en la ventana de comandos de DOS donde se compilo y uso la clase anterior:



Al estar haciendo operaciones, si hay operandos de diferentes tipos de datos, se convierten al tipo de datos más amplio y el tipo del valor resultante es del tipo más amplio. Por ejemplo, si hay enteros y flotantes, todos los números se convierten a flotantes y el resultado se calcula como flotante.

Por ejemplo:

4/3.0 da como resultado 1.3333

El operador **%** calcula el residuo de la división entera y sólo existe para datos de tipo entero

Por ejemplo:

10%3 da como resultado 1

### **Otros operadores de Asignación**

En Java, como en C++, es posible abreviar algunas expresiones de asignación como se muestra en la siguiente tabla:

Operador	Expresión equivalente
v + = e	v = v + e
v - = e	v = v - e
v * = e	v = v * e
v / = e	v = v / e
v % = e	v = v % e

### **Otros Operadores aritméticos**

En Java, al igual que en C++, existen también los siguientes operadores aritméticos:

++ incremento

-- decremento

Es decir:

`x++` ó `++x` es equivalente a `x = x+1`

`x--` ó `--x` es equivalente a `x = x-1`

Estos operadores son unitarios, es decir, trabajan con un solo operando y solamente se pueden utilizar con variables de tipo

entero.

Los operadores se pueden utilizar antes o después del nombre de la variable y funcionan de diferente manera:

- Si se ponen antes, primero se realiza la operación (incremento o decremento) y luego se utiliza el valor de la variable en la expresión en la que se encuentre.
- Si se pone después, primero se utiliza el valor de la variable en la expresión y luego se lleva a cabo la operación (incremento o decremento).

Por ejemplo:

Supón que `a = 10` y `c = 4`

La operación `v = a * c++;` `v` toma el valor de 40 y `c` queda con el valor de 5

La operación `v = a * ++c;` `v` toma el valor de 50 y `c` queda con el valor de 5

### **Jerarquía de los operadores aritméticos**

Prioridad	Operadores	Asociatividad
1	( )	Empezando por los paréntesis más internos
2	++, --, +(positivo), -(negativo)	De derecha a izquierda, ++ y -- dependiendo de la posición
3	*,/,%	De izquierda a derecha
4	+, -	De izquierda a derecha
5	=, +=, -=, *=, /=, %=	De derecha a izquierda

### **Algunos Métodos Matemáticos Predefinidos**

Java contiene una serie de métodos matemáticos que puedes utilizar en tus clases, para realizar algún cálculo, son tomados de la clase `Math`, esta viene dentro del paquete `java.lang`, entonces para poder tomarlos dentro de una clase debes de usar la instrucción

```
import java.lang.Math;
```

Antes de iniciar tu applet o aplicación. Algunos de los métodos a utilizar son:

```
public final static double e    da el valor de e.
```

```
public final static double PI    da el valor de pi.
```

```
public static int abs(int a)  da el valor absoluto de un entero dado.
```

```
public static long abs(float a) da el valor absoluto de un numero de punto flotante.
```

```
public static double cos(double a)  que te da el coseno de un valor de doble precisión.
```

```
public static double exp(double a)  te da el valor de  $e^a$  para un valor a de doble precisión.
```

```
public static double pow(double a, double b) te da el valor de  $a^b$  para a y b de doble precisión.
```

```
public static double sqrt(double a)  obtiene el valor de la raíz cuadrada de un valor a de doble precisión.
```

### ¿Cómo se utilizan?

Lo anterior es la definición de cada uno de los métodos de la clase Math, tu aun no estar familiarizado con esta definición, pero poco a poco entenderás su uso, vamos a entender el uso de uno de ellos.

```
public static double sqrt(double a)  obtiene el valor de la raíz cuadrada de un valor a de doble precisión.
```

La definición anterior describe que sqrt es un método de tipo double y el parámetro que toma es un double, entonces algunos ejemplos de su uso son:

#### Ejemplo1

```
double a, b;  
a = 25.0  
b = Math.sqrt(a);
```

#### Ejemplo 2

```
int a;  
double b;  
a = 25;  
b = Math.sqrt(a);
```

En ejemplo 1 vemos claramente como a b le será asignado el valor real 5.0, ya que esa es la

raíz cuadrada de 25.0

En ejemplo 2 pensaríamos que sería un error el tener el uso de una variable entera en lugar de una real, pero es valido asignarle a una variable real un valor entero, pero lo contrario es un error.