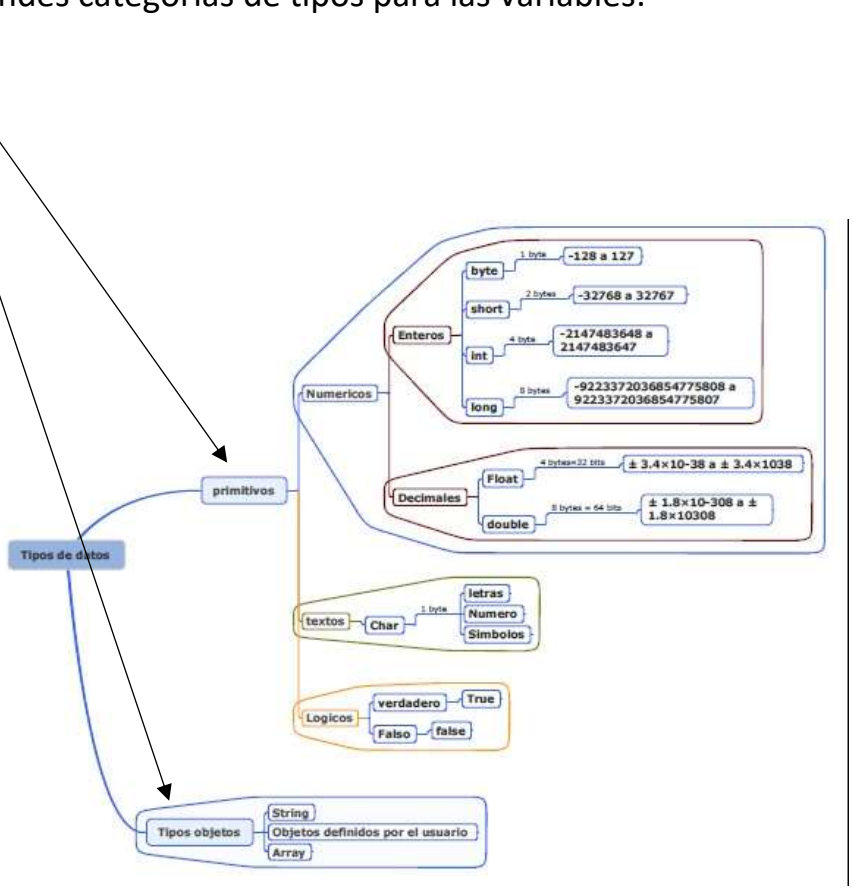
Variables ,Tipos y Operadores Lógicos

Una variable en Java es un identificador que representa una palabra de memoria que contiene información. El tipo de información almacenado en una variable sólo puede ser del tipo con que se declaró esa variable.

Una variable se declara usando la misma sintaxis de C. Por ejemplo la siguiente tabla indica una declaración, el nombre de la variable introducida y el tipo de información que almacena la variable:

Declaración	identificador	tipo
int i;	i	entero
String s;	s	referencia a string
int a[];	a	referencia a arreglo /arrays/ de enteros
int[] b;	b	referencia a arreglo/arrays de enteros

Java posee dos grandes categorías de tipos para las variables:



Tipos Primitivos	referencias a Objetos
int, short, byte, long	Strings
char, boolean	Arreglos
float, double	otros objetos

Las variables de tipos primitivos almacenan directamente un valor que siempre pertenece al rango de ese tipo. Por ejemplo una variable **int** almacena un valor entero como 1, 2, 0, -1, etc.

Esto significa que al asignar una variable entera a otra variable entera, se copia el valor de la primera en el espacio que ocupa la segunda variable.

Las variables de tipo referencia a objetos en cambio almacenan direcciones y no valores directamente. Una referencia a un objeto es la dirección de un área en memoria destinada a representar ese objeto. El área de memoria se solicita con el operador **new.**

Por ejemplo cuando escribimos una expresión de este tipo:

El operador new sirve para asignar memoria y crear un array de una dimensión.

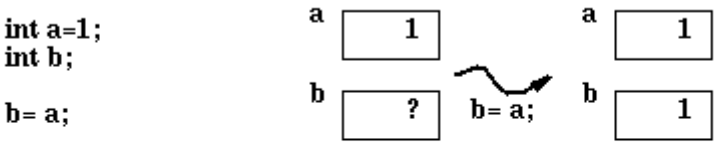
Var_array=new tipo[longitud];

Dia_mes=new int [12]; //array de 12 enteros

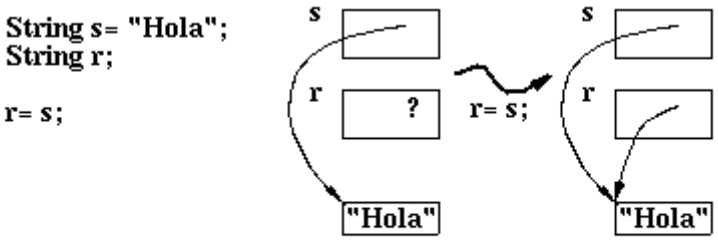
Dia_mes[1]=28, //array 28 a dia_mes

Al asignar una variable de tipo referencia a objeto a otra variable se asigna la dirección y no el objeto referenciado por esa dirección. Esto significa que ambas variables quedan referenciando el mismo objeto.

La diferencia entre ambas asignaciones se observa en la siguiente figura



Asignacion de variables de tipo primitivo



Asignacion de variables de tipo compuesto

Esto tiene implicancias mayores ya que si se modifica el objeto referenciado por r, entonces también se modifica el objeto referenciado por s, puesto que son el mismo objeto.

En Java una variable no puede almacenar directamente un objeto, como ocurre en C y C++.

Por lo tanto cuando se dice en Java que una variable es un string, lo que se quiere decir en realidad es que la variable es una referencia a un string.

Tipos primitivos

En la siguiente tabla se indica para cada tipo primitivo el número de bits que se emplea en su representación y el rango de valores que se puede almacenar en las variables de estos tipos.

Tipo	Bits	Rango	Ejemplos
int	32	-2**31 .. 2**31-1	0, 1, 5, -120, ...
short	16	-2**15 .. 2**15-1	0, 1, 5, -120, ...
byte	8	-2**7 .. 2**7-1	0, 1, 5, -120, ...

long	64	-2**63 .. 2**63-1	0, 1, 5, -120, ...
boolean	1	n/a	false, true
char	16	n/a	'a', 'A', '0', '*', ...
float	32	IEEE	1.2
double	64	IEEE	1.2

Se dice que un tipo A es de mayor rango que un tipo B si A es un superconjunto de B. Esto quiere decir que las variables de tipo B siempre se pueden asignar a variables de tipo A (eventualmente con pérdida de significancia).

Por ejemplo int es de mayor rango que short, que a su vez es de mayor rango que byte. Float y double son de mayor rango que int. Double es de mayor rango que float.

Esto se puede resumir en:

double > float > long > int > short > byte

Expresiones

En Java cada expresión tiene un tipo que se determina durante la compilación, es decir es independiente del valor durante la ejecución. Una expresión puede ser:

- Una constante: 1, 1.0, true, etc.

  El tipo de esta expresión es el tipo de la constante. En el ejemplo los tipos son int, double y boolean respectivamente.

- Una variable: i, s, a

El tipo de esta expresión es el tipo con que se declaró aquella variable. En el ejemplo los tipos son entero, referencia a string y referencia a arreglo (vectores o matrices).

- El resultado de una operación: i+1, a[1], etc.

  El tipo se determina en función de los tipos de las expresiones que se indican como argumentos de la operación.

Java tiene prácticamente los mismos operadores de C y C++. Cada operador acepta una, dos o tres expresiones de diversos tipos. A veces un operador no está definido para argumentos de ciertos tipos. Por ejemplo la resta entre strings no esta definida. Pero la suma de strings esta definida como la concatenación. A continuación veremos los operadores más frecuentes en Java.

Operadores binarios entre expresiones numéricas

Símbolo	Operación
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resto

Los operandos y el resultado de estos operadores pueden ser:

Dominio	Rango
int*int	int
long*long	long
float*float	float
double*double	double

- Al operar con byte y short, estos se convierten implícitamente a int.
- Cuando los tipos de los operandos no coinciden, el operando de menor rango se convierte implícitamente al tipo de mayor rango. El resultado de la operación es del tipo de mayor rango.

Ejemplos:

```
int a=1, b=2;
int c= a + b;

short s= 1;
int d= s + c; // s se convierte a int

float f= 1.0 + a; // a se convierte a float
```

Operadores unarios sobre expresiones numéricas

Símbolo	Operación	Argumento
+ exp	nada	int | long | float | double
- exp	cambio de signo	int | long | float | double

El resultado de estas operaciones es siempre del mismo tipo del operando.

Si el operando es byte o short entonces se convierte a int, se realiza la operación y el resultado es un int.

Operadores sobre variables enteras

Símbolo	Operación	Argumento
++ var	preincremento	int | short | byte | long
-- var	predecremento	int | short | byte | long
var ++	postincremento	int | short | byte | long
var --	postdecremento	int | short | byte | long

El resultado de estas operaciones es siempre del mismo tipo del operando.

Ejemplos:

```
int a=1, b=2;
int c= +a;
int d= -(c+1);

b= a++; // b= 1, a=2
a= --b; // a=b= 0

(a+1)++; // error a+1 no es una variable
float f= a;
f++;     // error f no es entero
```

Operadores binarios relacionales

Símbolo	Operación
>	mayor
<	menor
>=	mayor igual
<=	menor igual
==	igual
!=	distinto

Los operandos y el resultado de estos operadores pueden ser:

Dominio	Rango
int*int	boolean
long*long	boolean
float*float	boolean
double*double	boolean

Operadores entre valores booleanos

Operación	Significado
exp && exp	y-lógico
exp || exp	o-lógico
! exp	negación

Los operandos y el resultado de estos operadores son siempre booleanos.

Los operadores && y || evalúan la expresión del lado derecho sólo si es necesario.

Ejemplos:

```
int a=1, b=2;

boolean v= ! a<=b && a==5 || b!=4;
boolean w= ((! a<=b) && a==5) || b!=4;

boolean w2= a<=b && a; // error a no es boolean
```

Precedencia de Operadores

(Nivel de Jerarquías)

. () []
unarios: - + (cast)
* / %
+ -
< > <= >= == !=
!
||
&&