Taller de Programación Web

Data Types (Tipos de Datos)

¿Que es un tipo de dato?

Un tipo de dato (o simplemente dato) se puede definir en tres componentes:

- Un conjunto de valores (o objetos de datos)
- Un conjunto de operaciones que pueden ser aplicados a todos los valores del conjunto.
- Una representación del dato, que determina cómo los valores serán almacenados.

Como otros tipos de lenguajes, Java provee algunos tipos de datos predefinidos, que son conocidos como tipos de datos incorporados (built-in).
Un tipo de dato primitivo, es aquel que consiste en un valor atómico, indivisible y que puede ser definido sin la ayuda de otro tipo de dato.
Java provee varios tipos de datos primitivos, tales como `int, float, boolean, char`, etc. Por ejemplo, los tres componentes que definen el dato primitivo int en Java son los siguientes:

- Un tipo de dato **`int`** consiste en el conjunto de todos los valores enteros comprendidos entre -2147483648 y 2147483647.
- Las operaciones que se definen para el tipo de dato *int* son suma, resta, multiplicación, división, comparación y muchas más.
- Un valor para el tipo de dato *int* está representado en memoria de 32 bit en su forma de complemento a dos (binario).

¿Que es un identificador?

Un identificador en Java es una secuencia de caracteres. La secuencia de caracteres incluye todas las letras y dígitos de Java, donde la primera debe ser una letra. Dicha letra puede puede pertenecer a cualquier lenguaje que esté representado en el conjunto de caracteres Unicode. Por ejemplo, A-Z, a-z, _ (guión bajo o underscore), y $ son considerados como letras.

Ejemplos de identificadores válidos:

```
num1                //Puede usar a-z and 0-9 and comenzar con letra
kn                  //Solo letras
_abc                //Puede comenzar con underscore
_                   //Puede tener solo una letra, ej: underscore
suma_de_dos_nros    //Puede contener letras y underscore
Outer$Inner         //Puede contener a-z, A-Z y $
$var                //Puede también comenzar con $
```

Ejemplos de identificadores NO válidos:

```
2 num               //No puede comenzar con un número
mi nombre           //No puede contener espacios
num1+num2           //No puede tener el signo +
num1-num2           //No puede contener el símbolo resta (o hyphen)
```

Java define una lista de palabras reservadas (keywords). Keywords son palabras que tienen significados predefinidos en Java y que solo pueden ser usadas en los contextos definidos por el lenguaje de Java. Los keywords no pueden ser usados como identificadores. Podremos ver una lista de palabras reservadas a continuación:

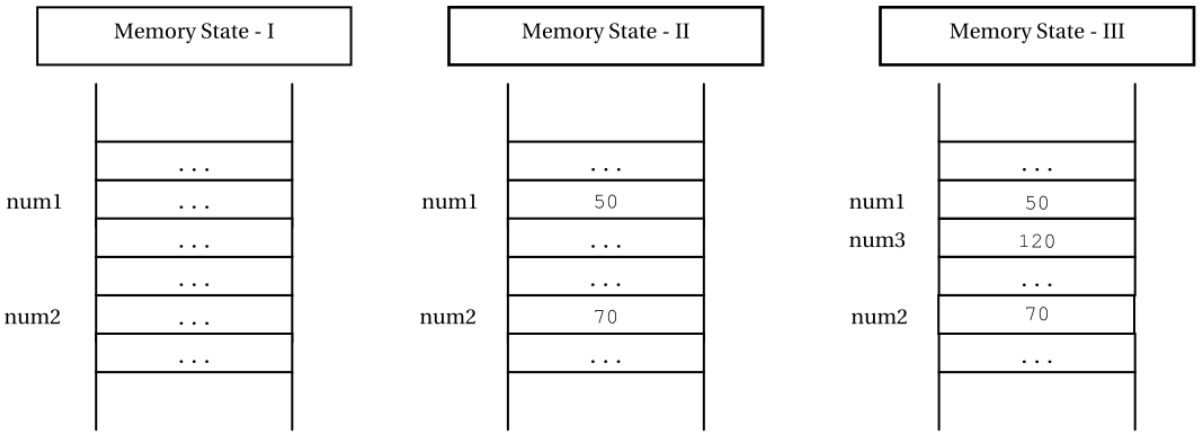
abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Tipos de Datos en Java

Supongamos que queremos realizar la suma de 2 números en un programa de Java. Al ser Java un lenguaje fuertemente tipado lo primero que debemos definir en Java es el tipo de los 2 números que queremos sumar. A modo de ejemplo, diremos que queremos sumar 2 números enteros, 20 y 30. Para poder realizar esta operación en un programa de Java, se le debe designar explícitamente un nombre (también conocido como identificador) para ambos números. Los llamaremos numero1 y numero2. Las siguientes líneas en un programa de Java indican que existen 2 números enteros, numero1 y numero2:

```
int numero1;
int numero2;
```

La palabra **int** es usada para indicar que el nombre que está a continuación representa un valor de tipo entero. Cuando son ejecutadas las líneas de código que están arriba, Java asigna 2 espacios de memoria y los asocia a los nombres numero1 para el primer espacio en memoria y numero2 para el segundo espacio. En el gráfico siguiente podemos ver el estado de la memoria luego de la ejecución de ambas líneas (ver Memory State - I)



Estos espacios de memoria son llamados **variables**. Los nombres numero1 y numero2 son asociados a estos espacios de memoria.

Como estas dos variables fueron declaradas como tipo de datos **int**, no podras almacenar un número real (ejemplo: 11.74). Las siguientes líneas almacenan el valor 20 en numero1 y 30 en numero2:


```
numero1 = 20;
numero2 = 30;
```

El estado de la memoria luego de la ejecución de ambas líneas se puede ver en el gráfico anterior (ver Memory State - II). Ahora, queremos realizar la suma. Antes de esto, asignaremos otro espacio en memoria, que contendrá el resultado. Este espacio de memoria la llamaremos numero3 y el siguiente bloque de código realizará dicha tarea:

```
int numero3;
numero3 = numero1 + numero2;
```

El estado de la memoria luego de la ejecución de ambas líneas se puede ver en el gráfico anterior (ver Memory State - III).

Variables

Podemos decir que una variable tiene tres propiedades:

- Un espacio de memoria que contiene el valor.
- El tipo de dato a almacenar en dicho espacio de memoria.
- Un nombre (llamado identificador) para referirse al espacio de memoria.

El tipo de dato de la variable también determina el rango de valores que el espacio de memoria puede contener. Por lo tanto, 32 bits de memoria son asignados para una variable de tipo `int`.

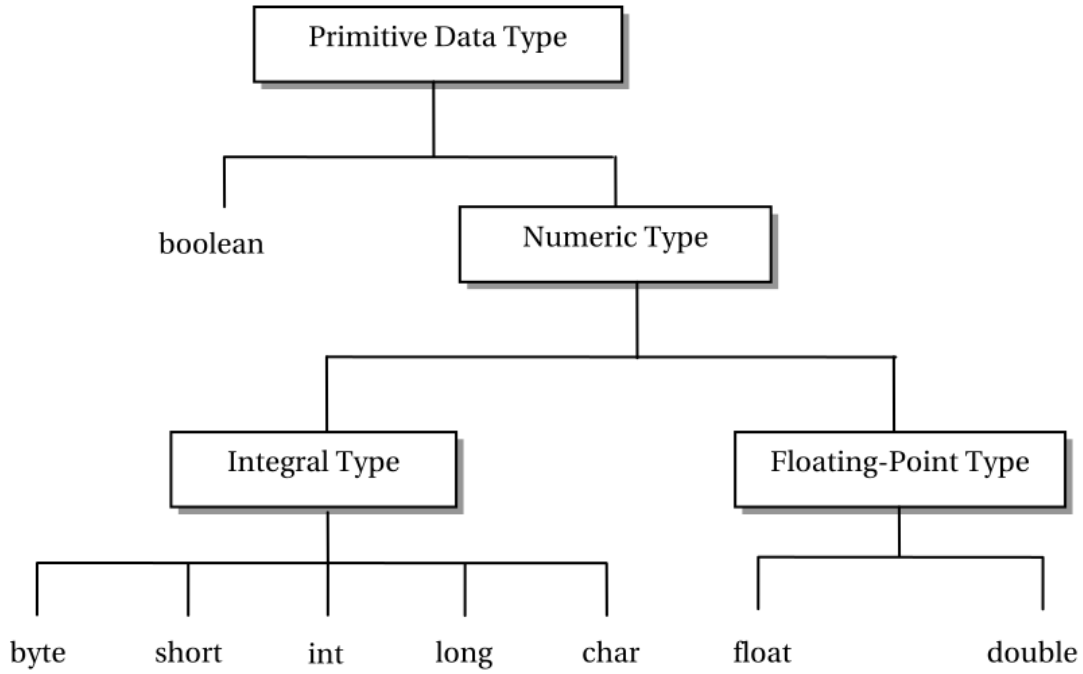
Java soporta 2 tipos de datos:

- Tipo de Dato primitivo
- Tipo de Dato de referencia

Una variable de tipo de dato primitivo contiene un valor mientras que una variable de tipo de dato de referencia contiene la referencia hacia un objeto en memoria. Uno de los tipos de datos de referencia disponibles en Java es ***String.*** String es una clase definida en la librería de Java y que puede usarse para manipular texto (secuencia de caracteres).

Tipos de Datos Primitivos

Dentro de los datos primitivos en Java describiremos ocho: `byte, short, int, long, char, float, double, y boolean`. Divididos en dos categorías tipo de dato booleano y tipo de dato numérico. Los de tipo numérico se subdividen en tipos integrales y tipos de coma-flotante.



Tipos de Datos Integrales

Los tipos de datos integrales es un tipo de dato numérico y sus valores son enteros.

El tipo de dato int

- El tipo de dato **`int`** es un tipo de dato primitivo de 32-bit (con signo).
- Su rango válido es -2147483648 a 2147483647 (-2^{31} *a* $2^{31}-1$). Todos los números en este rango se consideran como enteros literales (o constantes enteras).
- Los literales enteros pueden ser expresados en:
    - Formato de número decimal

- Formato de número octal
- Formato de número hexadecimal
- Formato de número binario

```
int num1 = 021;          //021 es en formato número octal (no decimal)
int num1 = 17;           //No posee 0 adelante, formato decimal
int num1 = 0x123         //Comienza con 0x, formato hexadecimal
int num1 = 0b10101       //Si comienza con 0b o 0B, se considera binario
```

El tipo de dato long

- El tipo de dato `long` es un tipo de dato primitivo de 64-bit (con signo).
- Su rango válido es -9223372036854775808 a 9223372036854775807 (-2^{63} a $2^{63}-1$). Todos los números en este rango se consideran como enteros literales (o constantes enteras).
- Un literal entero de tipo `long` siempre termina con L (o l minúscula, pero la comunidad recomienda el uso de la mayúscula). Ejemplo: 51L.
- También puede expresarse en diferentes formatos numéricos (como vimos con int)

```
long num1;
num1 = 25L;          Formato decimal
num1 = 031L;         Formato Octal
num1 = 0x19L         Formato Hexadecimal
num1 = 0b11001L      Formato Binario
```

Cuando un literal `long` es asignado a una variable de tipo `long`, el compilador de Java evalúa el valor asignado y se asegura que este dentro del rango del tipo de dato long. Caso contrario, generará un error en tiempo de compilación.

Observación:
La asignación de un tipo `int` a un tipo `long` es válida, porque todos los valores que pueden ser almacenados en una variable int también pueden ser almacenados en una variable long. Pero la inversa no es verdad. Veamos:

```
int num1 = 10;
long num2 = 2147483655L;
num1 = num2;  // Que sucederá?
```

Error en tiempo de compilación. El valor almacenado en num2 está fuera del rango que el tipo `int` puede manejar. Para evitar cometer estos errores que pueden pasar inadvertidos, Java no permite la asignación de un `long` a int.

El tipo de dato byte

- El tipo de dato `byte` es un tipo de dato primitivo de 8-bit (con signo).
- Su rango válido es -128 a 127 (-2^7 a 2^7-1). Este es el más pequeño del tipo de datos entero disponibles en Java.
- De manera similar a `int` con `long`. No se puede asignar a una variable byte una de tipo int (error de compilación).
- A diferencia de `int` y `long`, `byte` no posee literales byte.

El tipo de dato short

- El tipo de dato `short` es un tipo de dato primitivo de 16-bit (con signo).
- Su rango válido es -32768 a 32767 (-2^{15} a $2^{15}-1$). Todos los números en este rango se consideran como enteros literales (o constantes enteras).
- A diferencia de `int` y `long`, no existen literales `short`.

El tipo de dato char

- El tipo de dato **char** es un tipo de dato primitivo de 16-bit (sin signo, no posee valores negativos).
- Representa un carácter Unicode.
- Su rango válido es 0 a 65535.
- Un carácter literal representa un valor del tipo de dato char.
- Un carácter literal puede ser expresado en los siguientes formatos:
    - Un carácter encerrado por comillas simples (single quotes)
    - Como carácter de secuencia de escape.
    - Como una secuencia de escape Unicode.
    - Como una secuencia de escape octal.
- Recordar que las comillas dobles (“) encierran un literal String. Un String no puede ser asignado a una variable de tipo char.

Veamos algunos ejemplos:

```
char c1 = 'A';          // OK
String s1 = 'A';        // Error. No se puede asignar char a String
String s2 = "A";        // OK
char c1 = "A";          // Error. No se puede asignar String a char
char c3 =  'AB';        // Error. Un carácter literal debe contener solo un
                        carácter.
```

También se pueden asignar secuencias de escape de caracteres (algunos también llamados caracteres de impresión):

```
'\n';                   // Representa un salto de línea
'\r';                   // Representa retorno de carro
'\f';                   // Representa un form feed
'\b';                   // Representa el retroceso
'\t';                   // Representa Tabulación.
'\\';                   // Símbolo de backslash
```


```
'\"';                    // Símbolo de comillas dobles
'\';                     // Símbolo de comillas simples
```

El tipo de dato boolean

El tipo de dato `boolean` tiene solo dos valores válidos: `true` y `false`. También llamados literales booleanos. Se puede hacer uso de literales booleanos de la siguiente forma:

```
boolean terminado;       // Declara una variable boolean llamada terminado
terminado = true;        //Asigna el valor true a la variable terminado
```

Tipos de Datos Coma-Flotante:

- La parte fraccional de un número se conoce como número real (ejemplo: 8.15, 0.33, -3.14, etc).
- Un computador almacena todos los números en formato binario (que consiste en ceros y unos). Por lo tanto es necesario convertir un número real a su representación en binario antes de ser almacenado. Y de manera inversa si quiere ser leído.
- Las representaciones en coma flotante son más lentas y menos precisas comparadas con los de representación de punto fijo. Sin embargo, las representaciones de coma flotante pueden manejar un rango de valores mucho más amplio.
- Java posee dos tipos numéricos de coma flotante: `float` y `double`.

El tipo de dato float

- Utiliza 32 bits para almacenar números en coma flotante.

- Puede almacenar un número real tan pequeño como $1.4\ x\ 10^{-45}$ y tan grande como $3.4\ x\ 10^{38}$ (aprox) para la magnitud. Donde estos valores pueden ser negativos o positivos.
- Todos los números reales que finalizan con f o F son llamados literales float. Un literal `float` puede ser expresado en los siguientes formatos:
    - Formato de número decimal
    - Notación científica (para no prestar a confusiones. Solo lo mencionaremos aquí).

Ejemplos de literales float en formato decimal:

```
float f1 = 8F;
float f2 = 8.F;
float f3 = 8.0F;
float f4 = 3.51F;
float f5 = 0.0F;
float f6 = 16.78f;
```

El tipo de dato double

- Utiliza 64 bits para almacenar números en coma flotante.
- Puede almacenar un número real tan pequeño como $4.9\ x\ 10^{-324}$ y tan grande como $1.7\ x\ 10^{308}$ (aprox) para la magnitud. Donde estos valores
- Todos los números reales que finalizan con d o D (es opcional) son llamados literales `double`. Un literal double puede ser expresado en los siguientes formatos:
    - Formato de número decimal
    - Notación científica (para no prestar a confusiones. Solo lo mencionaremos aquí).

Ejemplos de literales double en formato decimal:

```
double d1 = 8D;
double d2 = 8.;
double d3 = 8.0;
double d4 = 8.D;
double d5 = 12.9867;
double d6 = 15.0;
```