Introducción a la Programación con Java

Lecturas recomendadas, Libro1 capítulos: antes de empezar, capitulo 1, capitulo 2, capitulo3, capitulo 4, capitulo 5 y , capitulo 6.

Estructura de una aplicación básica en Java

```
public class NombreClase {
    public static void main(String[] args) {
        instruccion1;
        instruccion2;
        instruccionN
        …
    }
}
```

Estructura de una aplicación básica en Java

```
public class NombreClase {...}
```

- Encabezado de una clase. El código fuente de un programa en java se encuentra dentro de una **archivo de texto** llamado CLASE .
- **Bloque de instrucciones**: todo lo que se encuentra entre una llave de inicio “{“ y una llave de fin “}” es un bloque de código.
- El nombre de una clase debe iniciar con MAYÚSCULA. Ejemplo: **H**olaMundo

Estructura de una aplicación básica en Java

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

- El **método principal o main()**: Es el primer bloque de instrucciones que se ejecutan en cualquier programa en java.
- Todo programa en java tiene obligatoriamente un único método main.
- Este método es ejecutado por el programa interprete: “java”.

Tipos de datos

- Son las categorías en las cuales se clasifica la información según el uso que se le da a esta.
- Por ejemplo: valores enteros, reales, caracteres, hileras de texto, entre otros.

Tipos de datos

- Para determinar el tipo de dato se deben tomar en cuenta algunos aspectos de la información a clasificar, incluyendo:
  - El conjunto de valores que se le asocia (si son valores **numéricos**, **caracteres** o **lógicos**).
  - El rango de valores válidos para el tipo (Ej. dentro de los números reales).
  - El **tamaño necesario** para representar cada valor en la memoria de la computadora (medido en bytes).

    Nota: es importante utilizar la cantidad **mínima** necesaria de memoria para almacenar el valor **máximo** posible).

Tipos primitivos de datos

- Los lenguaje de programación representan o encasillan los datos de formas simples y elementales mediante tipos primitivos de datos.
- Por ejemplo:
  - Variables de tipo Lógicas → **boolean**
  - Variables de tipo Numéricas enteros → **int**
  - Variables de tipo Numéricas de punto flotante → **double**
  - Caracteres → **char**
  - Otros tipos de datos..

Nota: las hileras de texto **NO** son tipos primitivos. → **String,** ya que son hileras o cadenas de caracteres.

Datos Booleanos (boolean)

- Se utilizan para representar condiciones lógicas que pueden tomar el valor true o el valor false únicamente.
  - Un valor **verdadero** se representa con el literal **true**.
  - Un valor **falso** se representa con el literal **false**.

Datos Numéricos en Java

- Datos enteros

Tipo	Nombre	Tamaño	Rango
Byte	byte	8 bits	Desde -128 hasta +127
Entero corto	short	16 bits	Desde -32768 hasta +32767
Entero	int	32 bits	Desde -2 147 483 648 hasta +2 147 483 548
Entero largo	long	64 bits	Desde -9 223 372 036 854 775 808 hasta + 9 223 372 036 854 775 807

Datos Numéricos en Java

- Datos flotantes

Tipo	Nombre	Tamaño	Rango
Punto flotante simple	float	32 bits	Desde -3.4028234663852886E+38 Hasta -1.4012984632481707E-45 Desde +1.4012984632481707E-45 Hasta +3.4028234663852886E+38
Punto flotante doble	double	64 bits	Desde -1.7976931348626157E+308 Hasta -4.94065645841246544E-324 Desde +4.94065645841246544E-324 Hasta +1.7976931348626157E+308

Datos Caracter en java

- Son aquellos que se utilizan para representar: letras, dígitos, símbolos especiales y caracteres de escape.
- En Java, un caracter se representa mediante un símbolo encerrado entre comillas **SIMPLES**.
  - Ejemplos: ‘a’, ‘A’, ‘9’.
- También existen caracteres especiales conocidos como secuencias de escape. En Java, los caracteres de escape van precedidos por un **“backslash” **
-

Caracteres de escape en Java

Secuencia	Significado
‘\n’	Cambio de línea
‘\t’	Tabulador
‘\\’	El caracter \
‘\”’	El caracter “
‘\uXXXX’	El caracter UNICODE asociado al valor XXXX

Caracteres UNICODE

- UNICODE es un conjunto de caracteres estándares creados para asociar cada caracter a un valor específico.
- En UNICODE un caracter ocupa 16 bits de memoria (2 bytes).
- Su rango va desde el hexadecimal 0000 hasta el hexadecimal FFFF.
- Tienen la forma ‘\uXXXX’, donde XXXX es un valor UNICODE

Caracteres UNICODE

- Sirven para representar símbolos especiales y caracteres en otros idiomas, tales como: Árabe, Chino, Japonés.
-
- Ejemplos en Java:
  - ‘\u0040’ equivale al valor decimal 64 que es ‘@’.
  - ‘\u00D1’ y ‘\u00F1’ equivalen a Ñ y ñ, cuyos valores decimales son 209 y 241.

Hileras de caracteres (String)

- Debido a que una hilera está “compuesta” de varios caracteres, las variables de tipo **String** no son consideradas como tipos primitivos de datos.
- Sin embargo se busca que su manejo sea tan simple como el manejo de tipos primitivos.

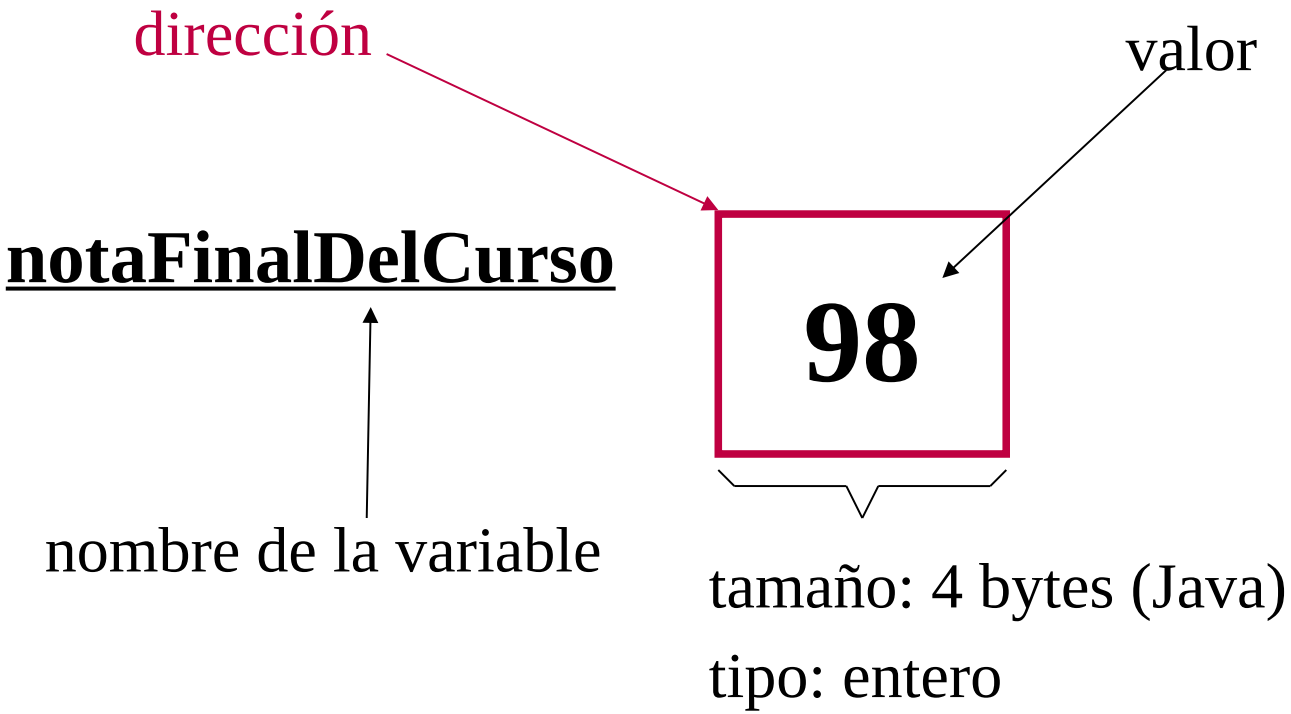
Hileras de caracteres

- En Java las hileras de caracteres se conocen como “String”.
- Los literales se representan con caracteres encerrados por comillas **DOBLES**. Ej: “¡Hola Mundo!”.
- El String también soporta secuencias de escape en su contenido.

Variables

- Son espacios en memoria asignados para almacenar valores.
- A cada variable se le asocia:
  - Un **nombre** que es único dentro de cada bloque de instrucciones {}.
  - Un **Tipo** de dato.
  - Un **Valor** inicial.
  - Un **Tamaño** o espacio que requiere en memoria.
  - Una **Dirección** de su ubicación en la memoria o referencia.

Características de las variables



Creación de una variable

- Requiere de dos pasos: **declarar** la variable e **inicializar** la variable.
- La declaración es la **definición** del tipo y del nombre de la variable.
- La inicialización es la **asignación** del valor inicial.

Declaración de variables en Java

- Se debe escribir el **tipo** seguido del **nombre** que se le quiere dar. Por ejemplo:

```
int variableEntera;
```

- Se pueden declarar varias variables a la vez separadas por **comas**. Por ejemplo:

```
char primeraLetra, segundaLetra;
```

Inicialización de variables en Java

- Se debe escribir el nombre de la variable, seguido del operador de **asignación** “=“, seguido del valor a asignar. Por ejemplo:

```
variableEntera = 234;
primeraLetra = ‘a’;
```

- En algunos casos, se puede mezclar la declaración y la inicialización. Por ejemplo:

```
int variableEntera = 234;
```

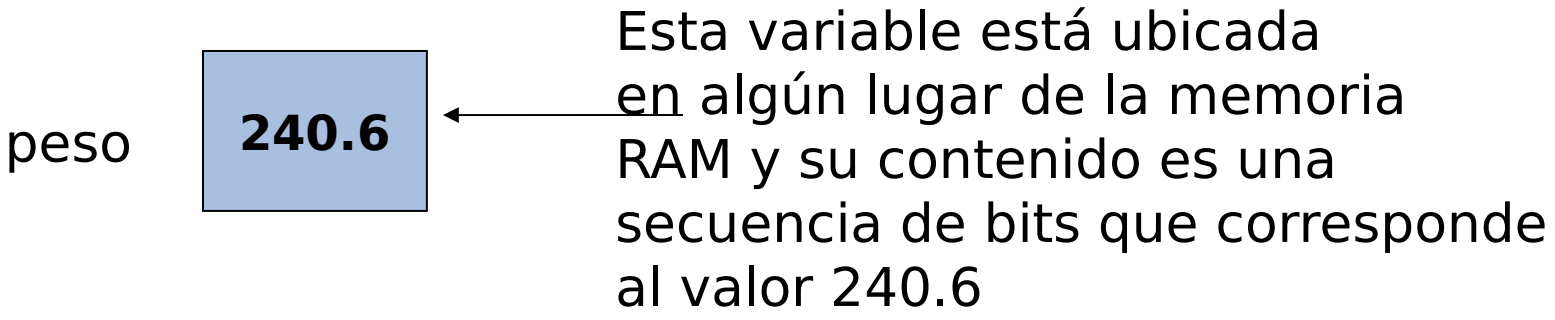
Nota: se recomienda crear las variables al inicio del programa, pero también esta permitido realizar esta acción en cualquier lugar del programa

La memoria al crear una variable

- Al crear la variable:

```
double peso = 240.6;
```

- Se crea en memoria una variable cuyo valor es la representación correspondiente a un 240.6.



Variables no primitivas

- Una variable puede ser de tipo primitivo o pueden ser de tipo No primitiva llamada: “**referencia a instancia”**.

- Ejemplo:
  - `String nombre = “Ana”;`

    Nota: La variable nombre **NO** almacena el valor de la cadena “Ana”, en su lugar guarda la dirección en memoria para encontrar el valor (“Ana”).

Nombres de variables en Java

- Para los nombres de variables se puede usar los siguientes caracteres: ‘a’, ‘b’, …, ‘z’, ‘A’, ‘B’, …, ‘Z’, ‘0’, ‘1’, …, ‘9’, ‘#’, ‘_’.
-
- Ejemplos:

```
int variableEntera1, variableEntera2,
  nombreUsuario, edadCliente;
double aproximacionDelValorPi;
char letra2, letraIngresada;
```

Recomendaciones para nombrar variables

- Utilice palabras significativas.
- Iniciar siempre los nombres de variable con una letra minúscula y la segunda o demás palabras dentro del nombre deben ir con la primera letra en mayúscula .

- No se deben utilizar tildes, ni simbolos, ni caracteres especiales como nombres de variables.

Ejemplos de nombres de variables apropiados

- Note que los siguientes nombres dan una idea del uso que se pretende dar a cada una de las variables:

```
double productoInternoBruto;
float saldoMensual;
int cantidadesEstudiantes;
char digitoInicialPlaca;
```

Constantes

- Son variables cuyo valor es asignado una única vez, y no puede ser cambiado posteriormente.

- Su declaración e inicialización se debe realizar en conjunto.

Constantes en Java

- Las constantes en Java se declaran con el calificativo final

- Ejemplo:

```
final double PI = 3.1415926536;
```

- Se recomienda nombrar las constantes utilizando solamente **letras MAYÚSCULAS**.

Convertir un valor de un tipo de dato a otro

- Se puede convertir un valor de un tipo de dato a otro mediante un mecanismo llamado "**casting**".
- Se antepone, entre paréntesis, el nombre del tipo al que se desea convertir el valor.
- Ejemplos:
  - `(int) 32.24` produce el entero 32.
  - `(double) 3` produce el double 3.0.
  - `(byte) 120` produce el byte 120.

Nota: No funciona para todos los casos y es posible perder parte de la información inicial

Entrada y Salida de datos

- Los programas frecuentemente requieren datos de entrada para realizar sus funciones.
- También requieren mostrar los resultados de su ejecución.
- La entrada / salida de los programas puede darse por muchos medios distintos: en consola o linea de comandos, con ventanas, páginas web, la impresora o el sistema de archivos, entre otros.

Entrada y Salida de datos en Java

- **Ejemplo**: Se pueden utilizar las ventanas de interacción que existen en JOptionPane.
- JOptionPane es una librería de Java que tiene varios tipos de pantalla de diálogo y que sirve tanto para **leer** como para **desplegar** datos ej:
  - El **InputDialog** lee datos digitados por el usuario.
  - El **MessageDialog** despliega mensajes en ventanas independientes.

Salida de datos en Java

- La salida de datos se puede llevar a cabo con el método showMessageDialog que presenta una pantalla para despliegue de datos.

- **`JOptionPane.showMessageDialog(null, “salida”, “titulo”, JOptionPane.tipoDeMensaje);`**

  - null en el primer parámetro indica que no se relacionará esta ventana con ninguna otra.
  - “salida” es el String que va a presentarse con la salida deseada.
  - “titulo” es el título que aparece en el borde de la ventana.
  - tipoDeMensaje determina las opciones o botones, y el icono o dibujo, que tendrá la ventana.

Entrada de datos en Java

- **Ejemplo**: La entrada de datos se puede llevar a cabo con el método showInputDialog que presenta una pantalla para entrada de datos.

- ```
  String nombreVariable =
  JOptionPane.showInputDialog(null, “salida”,
  “titulo”, JOptionPane.tipoDeMensaje);
  ```
  ```

Operadores aritméticos

Operaciones con valores numéricos

- Los tipos numéricos primitivos (suma, resta, producto, división y módulo (o residuo)) permiten realizar las operaciones aritméticas básicas.

	Operador	Ejemplo	Resultado
Suma	+	a = 1 + 2	a vale 3
Resta	-	a = 7 – 2	a vale 5
Producto	*	a = 2 * 7	a vale 14
División	/	a = 35 / 4	a vale 8
Residuo	%	a = 35 % 4	a vale 3

Prioridad entre operadores aritméticos

- El orden de prioridad de los operadores aritméticos es el siguiente:
  1. Paréntesis: ( )
  2. Multiplicativos: *, /, %
  3. Aditivos: +, -
  4. Asignación: =
- Si existen varios operadores del mismo tipo al mismo nivel, estos se realizan de izquierda a derecha.

Pasos que sigue el compilador para evaluar una expresión

Expresión: y = ((2 + 4 * 2) - 5 * (3 - 5)) / 3

Secuencia de evaluación:

y = ((2 + 8) - 5 * (3 - 5)) / 3
y = ((10) - 5 * (3 - 5)) / 3
y = (10 - 5 * (3 - 5)) / 3
y = (10 - 5 * (-2)) / 3
y = (10 - 5 * -2) / 3

y = (10 --10)/3
y = (20) / 3
y = 20 / 3
y = 6

Operadores Unarios

Operadores Unarios

- Algunos operadores unarios sirven para sumar o restar directamente el valor de una variable.
- El efecto de la operación es diferente si se coloca el símbolo **antes** o **después** de la variable.
- Por ejemplo:
  - En el pre incremento y pre decremento primero modifican la variable y luego permiten tomar su valor
  - En el post incremento y post decremento primero se usa el valor de la variable y luego la modifica la misma variable.

Operadores de incremento y decremento.

Suponga que c tiene un valor de 2.

Operador	Ejemplo	Resultado
Postincremento	d = c++;	d vale 2, c vale 3
Preincremento	d = ++c;	d vale 3, c vale 3
Postdecremento	d = c--;	d vale 2, c vale 1
Predecremento	d = --c;	d vale 1, c vale 1

Operadores
Asignación Directa

Operadores Asignación Directa

- Toman el valor de una variable como primer operando, le aplican una operación utilizando un valor dado como segundo operando, y finalmente dejan el resultado en la misma variable de donde se tomó el primer operando.

Operadores Asignación Directa en Java

Operador	Valor previo	Operación	Resultado
=	c vale 30	c = 15;	c vale 15
+=	c vale 10	c += 12;	c vale 22
-=	c vale 4	c -= 6;	c vale -2
*=	c vale 2	c *= 15;	c vale 30
/=	c vale 27	c /= 3;	c vale 9
%=	c vale 9	c %= 4;	c vale 1

Operadores Relacionales

Operadores Relacionales

- Estos operadores permiten evaluar relaciones entre valores. Por ejemplo:
  - ¿Es el valor A mayor que el valor B?
- El resultado de estas operaciones es un valor de tipo booleano.
- Estos operadores se utilizan para escribir expresiones que permitan controlar el flujo de ejecución de un programa.

Operadores Relacionales en Java

Operador	Ejemplo	Significado
==	X == Y	¿ es X igual a Y?
!=	X != Y	¿ es X distinto de Y?
>	X > Y	¿ es X mayor que Y?
<	X < Y	¿ es X menor que Y?
>=	X >= Y	¿ es X mayor o igual que Y?
<=	X <= Y	¿ es X menor o igual que Y?

Prioridad entre operadores de comparación

- La precedencia entre estos operadores es la siguiente:
  - Los operadores: <, >, <=, y >=, tienen prioridad.
  - Luego siguen los de igualdad y desigualdad: ==, !=
- Si existen varios operadores del mismo tipo al mismo nivel, se evalúan de izquierda a derecha.
- Todos los operadores aritméticos preceden a los relacionales.

Operadores Lógicos

Operadores Lógicos

- La operaciones lógicas básicas son:
  - Y (AND).
  - O inclusivo (OR).
  - Negación (NOT).

Operador Y (AND)

- El resultado del Y lógico entre dos operandos es verdadero solamente cuando ambos valores son verdaderos.
- Cuando alguno de los operandos es falso, el resultado del Y siempre es falso.

Y	**Verdadero**	**Falso**
Verdadero	Verdadero	Falso
Falso	Falso	Falso

Operador Y (AND) en Java

- El Y en Java se representa por el símbolo &&.
- Por ejemplo:
  - (true && false) produce false.

Operador O (OR)

- El resultado del O entre dos operandos es verdadero siempre que alguno de los dos operandos es verdadero.
- Solamente cuando ambos valores son falsos, el resultado del O puede ser falso

OR	**Verdadero**	**Falso**
Verdadero	Verdadero	Verdadero
Falso	Verdadero	Falso

Operador O (OR) en Java

- El O en Java se representa por el símbolo ||
- Por ejemplo:
  - (true || false) produce true.

Operador de negación (NOT)

- La negación se aplica sobre un único operando booleano y permite obtener su valor opuesto.
- La negación de un operando produce un valor verdadero solamente cuando el operando es falso y viceversa.

NOT	
Verdadero	Falso
Falso	Verdadero

Operador de negación (NOT) en Java

- La negación o NOT en Java se representa por el símbolo ! precediendo al operando booleano.
- Por ejemplo:
  - !(true) produce false.

Tabla de precedencia de operadores

Orden	Tipo	Operadores	Asociatividad
1	Posfijos	( ) var++ var--	Izquierda a derecha.
2	Unarios	++var --var	Izquierda a derecha.
3	Multiplicativos	* / %	Izquierda a derecha.
4	Aditivos	+ -	Izquierda a derecha.
5	Relacionales	< > <= >=	Izquierda a derecha.
6	Igualdad	== !=	Izquierda a derecha.
7	AND lógico	&&	Izquierda a derecha.
8	OR lógico	||	Izquierda a derecha.
9	Asignación	= += -= /= %=	Derecha a izquierda.

Palabras Reservadas de Java

- Aquellas que emplea el lenguaje Java y que el programador no puede utilizar como identificadores.
- Están marcadas en azul dentro del código.
- Deben ir en minúscula.
- En la siguiente lista las palabras reservadas señaladas con un asterisco (*) no se utilizan

Palabras Reservadas de Java

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

* No se utilizan actualmente

Ej de agrupación por categoría

- **Tipos de datos:** boolean, float, double, int, char
- **Sentencias condicionales:** if, else, switch
- **Sentencias iterativas:** for, do, while, continue
- **Tratamiento de las excepciones:** try, catch, finally, throw
- **Estructura de datos:** class, interface, implements, extends
- **Modificadores y control de acceso:** public, private, protected, transient
- **Otras:** super, null, this.