

Java SE

Curso básico de Java Standard Edition

Arturo Rivas Rojas

23 de diciembre de 2018

1. Introducción	2
1.1.¿Qué es?	2
1.2.Filosofía	2
1.3.Origen	2
1.3.1Clasificación	3
1.4.Herramientas de desarrollo	3
1.4.1Maquina Virtual	3
1.5.Metodo Main	3
1.6.Entorno de Desarrollo Integrado (IDE)	3
2. Tipos de Datos	4
2.1.Primitivos	4
2.1.1Enteros	4
2.1.2Flotantes	5
2.1.3Texto	5
2.1.4Logicos	5
2.2.Objetos	5
3. Variables	6
3.1.Naming	6
3.2.Cast	6
3.3.Arreglos (Arrays)	7
3.4.Declaración e Inicialización	8
3.5.Consultar	8
4. Operadores	9

1.1. ¿Qué es?

Java es un lenguaje de programación de **alto nivel**.

Características:

- Simple
- Orientado a Objetos
- Distribuido
- Multihilo
- Arquitectura Neutral
- Portable
- Alto desempeño
- Seguro

Java es un lenguaje compilado e interpretado.

1.2. Filosofía

Escribe una vez y correlo donde quieras.

1.3. Origen

Surge en 1991 creado por James Goslin, y fue absorbido por la compañía Sun microsystems.

Nacio con el proposito de comunicar dispositivos de forma configurable y portable.

En el 2009 Java llego a su apogeo y se volvio propiedad de Oracle. Nacieron las certificaciones Java.

1.3.1. Clasificación

- Java SE (Standard Edition) : base y sintaxis del lenguaje.
- Java EE (Enterprise Edition) : ultima generación.

1.4. Herramientas de desarrollo

Para poder trabajar con Java es necesario tener estos 2 componentes:

- JDK (Java Developer Kit): kit de desarrollo.
- JRE (Java Run Enviroment): para correr las aplicaciones.

1.4.1. Maquina Virtual

Una vez que tenemos nuestro codigo java y lo compilamos con el comando **javac** se genera una archivo .class que contine ByteCode.

La maquina virtual de Java a la que se llama con el comando **java** recibe de entrada ByteCode y lo interpreta.

1.5. Metodo Main

Es el punto de entrada de toda aplicación Java. los argumentos que se envian al programa se reciben en el main.

1.6. Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado es una herraminta que tiene todo lo necessario en una sola aplicación la terminal, editor de texto , buscador de archivos, etc.

Saber que tipo de dato ocupar nos ayuda a evitar el desperdicio de memoria.

2.1. Primitivos

Se pueden identificar porque se escriben con minúsculas.

2.1.1. Enteros

- **byte** (1 byte) -2^7 a 2^7-1 .

```
byte age = 127;
```

- **short** (2 bytes) -2^{15} a $2^{15}-1$.

```
short year = 2019;
```

- **int** (4 bytes) -2^{32} a $2^{32}-1$.

```
int id_user = 102301;
```

- **long** (8 bytes) -2^{64} a $2^{64}-1$.

```
long id_twitter = 123423552532L;
```

Es necesario agregar una **L** al final del dato para señalar que es del tipo long.

2.1.2. Flotantes

- **float** (4 bytes).

```
float diameter = 34.25f;
```

Se debe añadir una **f** al final para decirle al compilador que es un float y no un double.

- **double** (8 bytes).

```
double price = 1234.256618177892;
```

2.1.3. Texto

- **char** (2 bytes) unicode.

```
char gender = 'M';
```

Se deben trabajar entre comillas simples **'**.

2.1.4. Logicos

- **boolean** (2 bytes) true or false.

```
boolean isVisible = true;
```

2.2. Objetos

Una variable es un espacio de memoria a la que se le asigna un valor.

3.1. Naming

- Sensible a minúsculas y mayúsculas.
- Su expresión regular seria `^[a-z$_][\w$]*`

Las clases se escriben con el estandar **upper camel case** mientras que los metodos y variables con **lower camel case**.

3.2. Cast

Es hacer una conversión entre tipos de datos.

De un tipo de dato que ocupa menos memoria a uno con mayor memoria el casteo es automatico.

En el caso opuesto se debe de definir a que tipo de dato se desea castear.

```
// primer caso
byte b = 6;
short s = b;
// segundo caso
s = 100;
b = (byte) s;
```

Cuando se castea un float a entero este se trunca, es decir desaparece lo que exista después del punto decimal.

En el caso de los char, castear un entero a un char nos regresa el caracter que ese entero representa en el código ASCII.

Caracteres ASCII de control			Caracteres ASCII imprimibles			ASCII extendido (Página de código 437)										
00	NULL	(carácter nulo)	32	espacio	64	@	96	`	128	Ç	160	Á	192	Ł	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ù	161	í	193	ł	225	ô
02	STX	(inicio texto)	34	"	66	B	98	b	130	é	162	ó	194	Ł	226	õ
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	ö
04	EOT	(fin transmisión)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ø
05	ENQ	(consulta)	37	%	69	E	101	e	133	å	165	Ñ	197	†	229	õ
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	ä	166	*	198	‡	230	µ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	°	199	Ä	231	þ
08	BS	(retroceso)	40	(72	H	104	h	136	ë	168	¿	200	Å	232	ß
09	HT	(tab horizontal)	41)	73	I	105	i	137	ê	169	®	201	Æ	233	Ü
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	¬	202	Ɔ	234	Ù
11	VT	(tab vertical)	43	+	75	K	107	k	139	í	171	½	203	Ɔ	235	Ú
12	FF	(nueva página)	44	,	76	L	108	l	140	î	172	¾	204	Ɔ	236	Ý
13	CR	(retorno de carro)	45	-	77	M	109	m	141	ï	173	ı	205	Ɔ	237	Ÿ
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Ā	174	«	206	Ɔ	238	—
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Ā	175	»	207	Ɔ	239	—
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	É	176	—	208	Ɔ	240	±
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	—	209	Ɔ	241	±
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	—	210	Ɔ	242	—
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ð	179	—	211	Ɔ	243	¼
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	ó	180	—	212	Ɔ	244	½
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	ô	181	—	213	Ɔ	245	¾
22	SYN	(inactividad sinc)	54	6	86	V	118	v	150	ù	182	—	214	Ɔ	246	—
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	û	183	—	215	Ɔ	247	—
24	CAN	(cancelar)	56	8	88	X	120	x	152	ÿ	184	—	216	Ɔ	248	—
25	EM	(fin del medio)	57	9	89	Y	121	y	153	Ō	185	—	217	Ɔ	249	—
26	SUB	(sustitución)	58	:	90	Z	122	z	154	Ū	186	—	218	Ɔ	250	—
27	ESC	(escape)	59	;	91	[123	{	155	ø	187	—	219	Ɔ	251	—
28	FS	(sep. archivos)	60	<	92	\	124		156	£	188	—	220	Ɔ	252	—
29	GS	(sep. grupos)	61	=	93]	125	}	157	Ø	189	—	221	Ɔ	253	—
30	RS	(sep. registros)	62	>	94	^	126	~	158	×	190	—	222	Ɔ	254	—
31	US	(sep. unidades)	63	?	95	_			159	ƒ	191	—	223	Ɔ	255	nbsp
127	DEL	(suprimir)														

Figura 3.1: código ASCII

3.3. Arreglos (Arrays)

Un arreglo es un objeto donde se pueden guardar multiples variables de un mismo tipo.

Entonces podemos decir que un arreglo es un contenedor finito de un tipo de variable definido.

Un arreglo de una solo dimensión es una fila, uno de dos dimensiones es una matriz, uno de tres dimensiones es un cubo.

Para poder imaginarnos un array de n-dimensiones lo haremos visualizando a cada array como una matryoshka de n-1 y una caja donde hay varias de estas matryoshka.



Figura 3.2: matryoshka

3.4. Declaración e Inicialización

Para declarar un arreglo se utilizan los corchetes “[]”, un par por cada dimensión.

- único arreglo

```
// solo la variable con los corchetes a su lado
// sera un arreglo del tipo de dato definido.
int arregloInt[] = new int[5], entero = 4;
```

- múltiples Arreglos

```
// todas variables son arreglos del tipo de dato definido.
char[] [] arreglo1, arreglo2[], arreglon;
arreglo1 = new char[12][2];
// notamos que arreglo2 es un arreglo de arreglos
// de 2 dimensiones eso lo vuelve un arreglo de 3 dimesiones
arreglo2 = new char[3][2][3];
arreglo3 = new char[21][4];
```

3.5. Consultar

Para consultar los elementos de un arreglo se utilizan indices que comienzan en el 0 y van al tamaño del arreglo-1.

```
// un arreglo de tamaño 2
int[2] arreglo = new int[2];
// inicializariamos sus elementos de la siguiente manera:
// primera posición
arreglo[0] = 5;
// segunda posición
arreglo[1] = 87;
// podemos obtener e imprimir el valor
System.out.println(arreglo[0]);
System.out.println(arreglo[1]);
```

CAPÍTULO 4

Operadores
