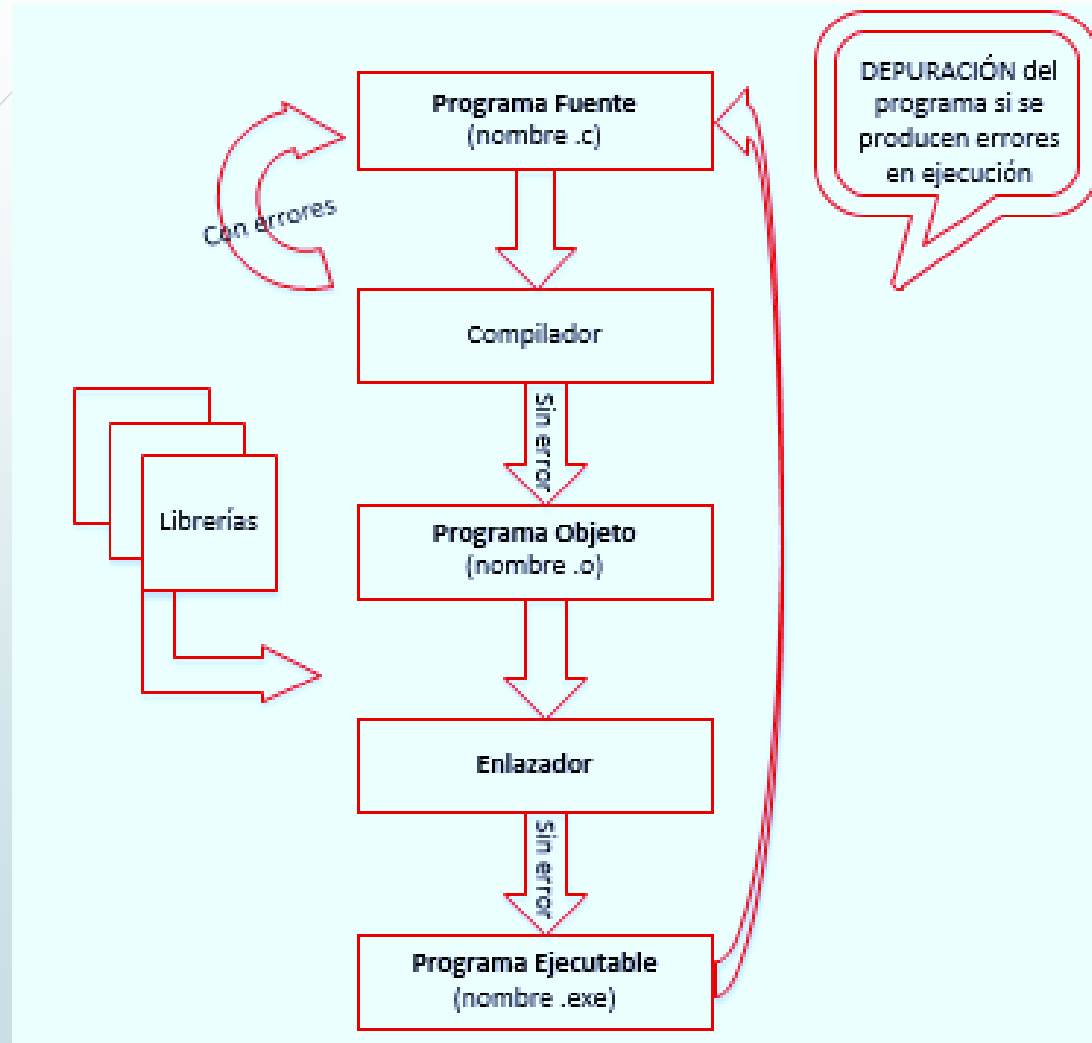




Programación en Java

Tipos de datos y argumentos

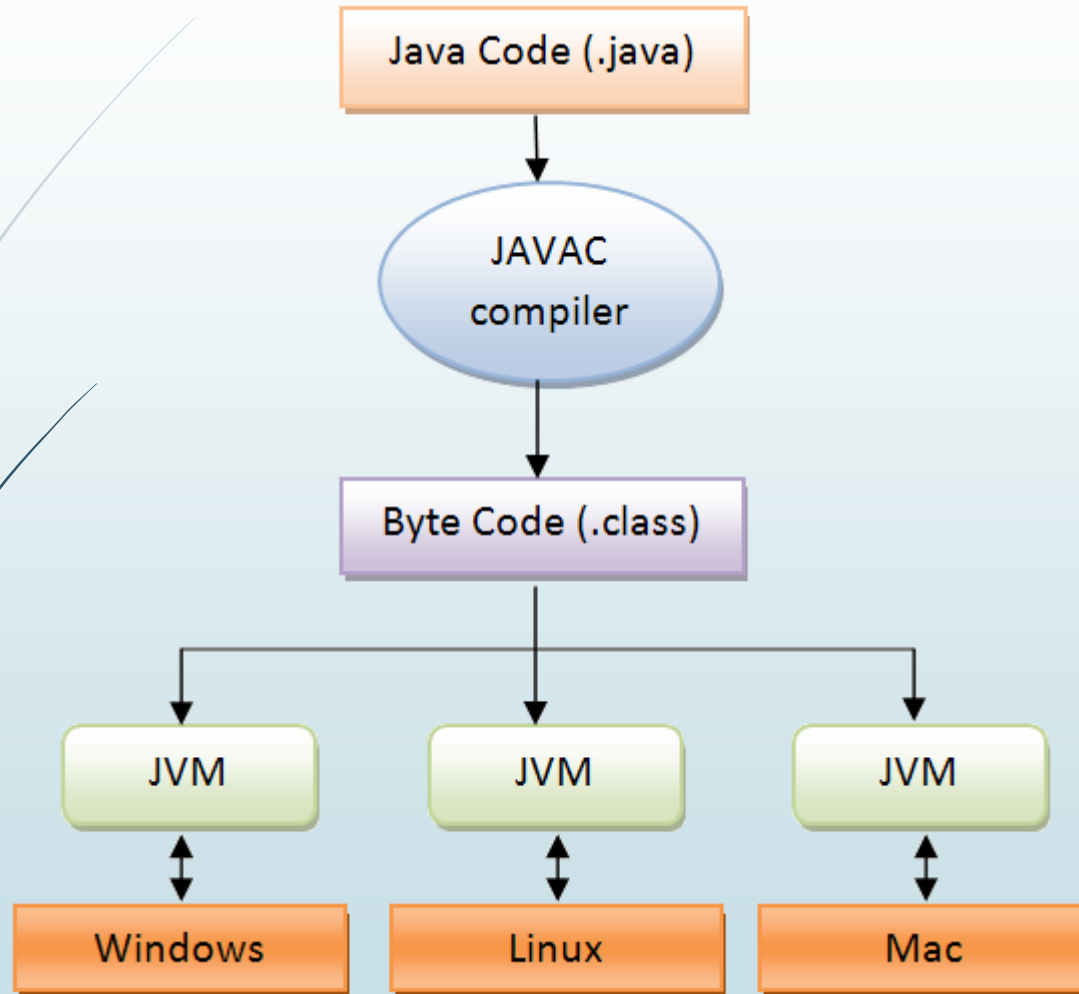
Compilación en lenguaje C



Hecho por Huicho :)

- El código fuente se guarda como .c
- Es dependiente del sistema operativo y arquitectura de la máquina
- Se compila
- Se obtiene un ejecutable .exe compuesto de unos y ceros
- Se basa en funciones

Compilación e interpretación en Java



Hecho por Huicho :)

- El código fuente se guarda como .java
- Es independiente del sistema operativo y arquitectura
- Se pre-compila y después es interpretado por la máquina virtual
- Se obtiene un código intermedio .class conocido como bytecode
- Se basa en clases

Máquina Virtual de Java

- Conocida también como: Java Runtime Environment, jre, Java Virtual Machine o JVM
- Permite escribir y compilar el programa una sola vez en lugar de varias veces y ejecutar ese código en cualquier plataforma
- Para correr el programa es necesario instalarla
- Para que la “Máquina Real” (nuestra computadora) ejecute el programa la máquina virtual debe “interpretar” (traducir) el archivo .class a un código en “Lenguaje de Máquina Real”

Tipos de datos

En Java toda la información que maneja un programa está representada por dos tipos principales de datos:

a) Datos de tipo básico o primitivo.

- *sin métodos*
- *no son objetos*
- *no necesitan una invocación para ser creados*

b) Referencias a objetos.

- *con métodos*
- *son objetos*
- *necesitan una invocación para ser creados*

Tipos de datos

Tipo de dato	Representación	Tamaño (Bytes)	Rango de Valores	Valor por defecto	Clase Asociada
byte	Numérico Entero con signo	1	-128 a 127	0	Byte
short	Numérico Entero con signo	2	-32768 a 32767	0	Short
int	Numérico Entero con signo	4	-2147483648 a 2147483647	0	Integer
long	Numérico Entero con signo	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Numérico en Coma flotante de precisión simple Norma IEEE 754	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0	Float
double	Numérico en Coma flotante de precisión doble Norma IEEE 754	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0	Double
char	Carácter Unicode	2	\u0000 a \uFFFF	\u0000	Character
boolean	Dato lógico	-	true ó false	false	Boolean
void	-	-	-	-	Void

Datos tipo char

- Contiene un número entero para representar un carácter dentro del rango \u0000 a \uFFFF (números desde 0 hasta 65535) en **Unicode**.
- Los 127 primeros caracteres de Unicode corresponden al código ASCII.
- El código Unicode es más grande y permite representar cualquier carácter de cualquier idioma como el Japonés, Chino o árabe.
- **El código Unicode proporciona una única representación numérica para cada símbolo, independientemente del ordenador, el programa o el lenguaje de programación que se use.**

Código Unicode

0020	0	0030	@	0040	P	0050	`	0060	p	0070		00A0	°	00B0	À	00C0	Ð	00D0	à	00E0	ð	00F0
0021	!	0031	1	0041	A	0051	Q	0061	a	0071	q	00A1	±	00B1	Á	00C1	Ñ	00D1	á	00E1	ñ	00F1
0022	"	0032	2	0042	B	0052	R	0062	b	0072	r	00A2	¢	00B2	Â	00C2	Ò	00D2	â	00E2	ò	00F2
0023	#	0033	3	0043	C	0053	S	0063	c	0073	s	00A3	£	00B3	Ã	00C3	Ó	00D3	ã	00E3	ó	00F3
0024	\$	0034	4	0044	D	0054	T	0064	d	0074	t	00A4	¤	00B4	Ä	00C4	Ô	00D4	ä	00E4	ô	00F4
0025	%	0035	5	0045	E	0055	U	0065	e	0075	u	00A5	¥	00B5	Å	00C5	Ö	00D5	å	00E5	ö	00F5
0026	&	0036	6	0046	F	0056	V	0066	f	0076	v	00A6		00B6	Æ	00C6	Ö	00D6	æ	00E6	ö	00F6
0027	'	0037	7	0047	G	0057	W	0067	g	0077	w	00A7	§	00B7	Ç	00C7	×	00D7	ç	00E7	÷	00F7
0028	(0038	8	0048	H	0058	X	0068	h	0078	x	00A8	¨	00B8	È	00C8	Ø	00D8	è	00E8	ø	00F8
0029)	0039	9	0049	I	0059	Y	0069	i	0079	y	00A9	©	00B9	É	00C9	Ù	00D9	é	00E9	ù	00F9
002A	*	003A	:	004A	J	005A	Z	006A	j	007A	z	00AA	ª	00BA	Ê	00CA	Ú	00DA	ê	00EA	ú	00FA
002B	+	003B	;	004B	K	005B	[006B	k	007B	{	00AB	«	00BB	Ë	00CB	Û	00DB	ë	00EB	û	00FB
002C	,	003C	<	004C	L	005C	\	006C	l	007C		00AC	¬	00BC	Ì	00CC	Ü	00DC	ì	00EC	ü	00FC
002D	-	003D	=	004D	M	005D]	006D	m	007D	}	00AD	-	00BD	Í	00CD	Ý	00DD	í	00ED	ý	00FD
002E	.	003E	>	004E	N	005E	^	006E	n	007E	~	00AE	®	00BE	Î	00CE	Þ	00DE	î	00EE	þ	00FE
002F	/	003F	?	004F	O	005F	_	006F	o	007F		00AF	™	00BF	Ï	00CF	ß	00DF	ï	00EF	ÿ	00FF

Código Unicode

```
System.out.println("Soy ni \u00F1o");
```

Caracter	Código UNICODE	Caracter	Código UNICODE
á	\u00E1	Á	\u00C1
é	\u00E9	É	\u00C9
í	\u00ED	Í	\u00CD
ó	\u00F3	Ó	\u00D3
ú	\u00FA	Ú	\u00DA
ñ	\u00F1	Ñ	\u00D1

Tipos de datos

```
int entero= 5;
```

```
float flota= 4.3f; //se coloca una f al final del número
```

```
double doble= 5.87;
```

```
boolean verdad= true;
```

```
char letra= 'c';
```

```
System.out.println(entero + ", " + flota);
```

```
System.out.println(doble + ", " + verdad + ", " + letra);
```

```
System.out.printf("%.2f", flota);
```

Constantes

- Produce un valor a ser usado en el programa pero no modificable (final)

```
final float pi = 3.14159
```

Arreglos

- Crear un arreglo vacío de enteros y asignarle después valores:

```
int lista[] = new int[2];  
lista[0]= 1;  
lista[1]= 2;
```

- Crear un arreglo vacío de objetos String y después asignar:

```
String nombres[];  
nombres= new String[3];  
  
nombres[0] = "Hugo";  
nombres[1] = "Paco";  
nombres[2] = "Luis";
```

Arreglos inicializados

- Crear un arreglo con valores iniciales de enteros:

```
int arreglo[]= {1,2,3};  
int a[]= new int[]{4,5,6};
```

- Inicializar arreglo de objetos String:

```
String nombres[]= {"Hugo", "Paco", "Luis"};  
String nombres[]= new String[]{"Hugo","Paco","Luis"};
```

Arreglos (forma incorrecta)

- **NO** se pueden crear arreglos estáticos en tiempo de compilación:

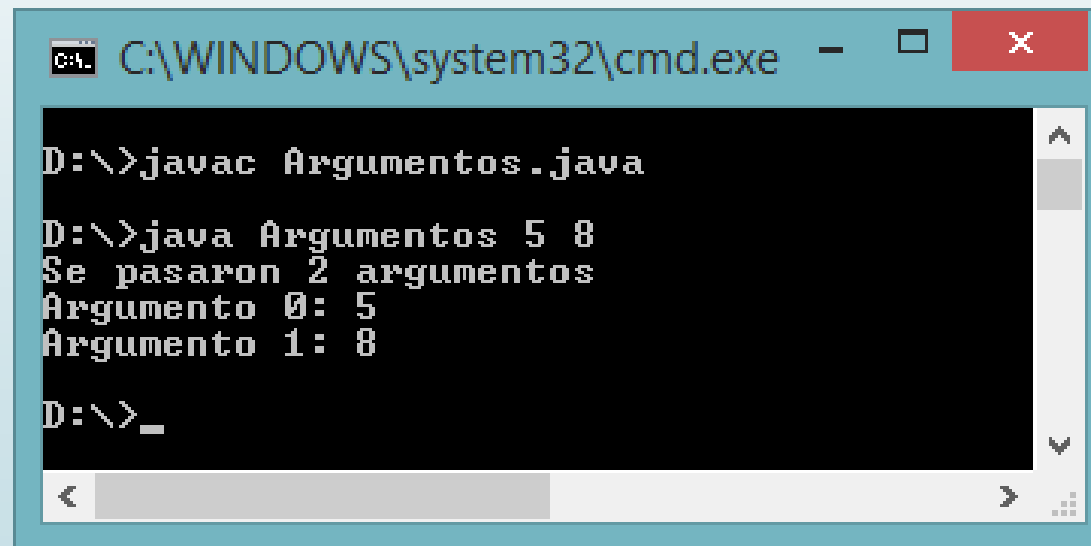
```
int lista[50];
```

- **NO** se puede rellenar un arreglo sin declarar el tamaño con el operador new:

```
int lista[];  
for(int i=0; i<9; i++)  
    lista[i]= i;
```

Argumentos desde consola

Al ejecutar desde la línea de comandos pueden pasarse datos después del nombre del ejecutable separados por un espacio.



```
C:\WINDOWS\system32\cmd.exe

D:\>javac Argumentos.java

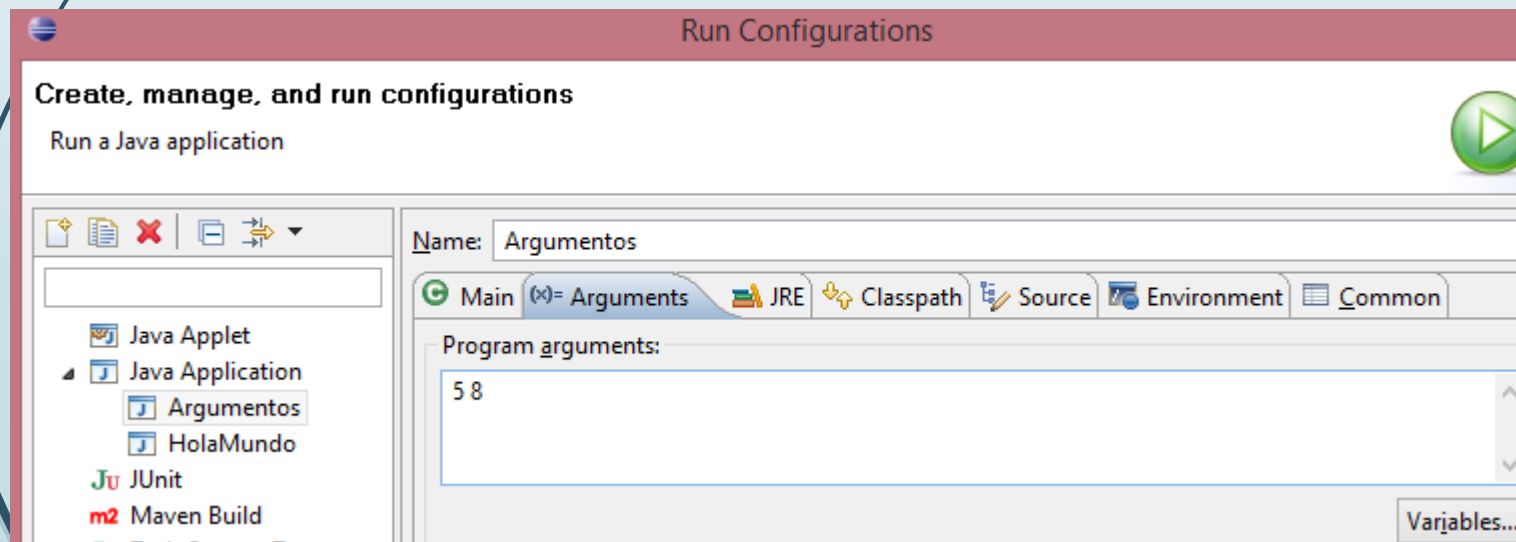
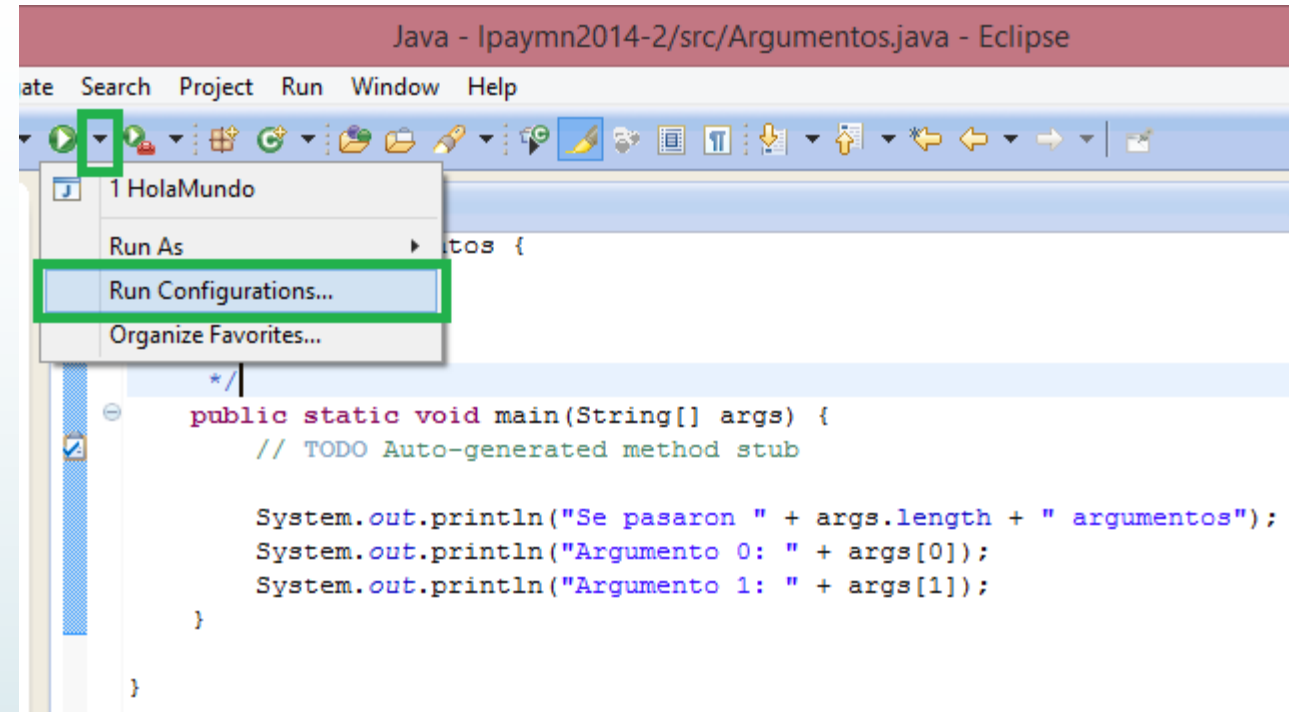
D:\>java Argumentos 5 8
Se pasaron 2 argumentos
Argumento 0: 5
Argumento 1: 8

D:\>_
```

Argumentos en eclipse

Pueden pasarse argumentos desde eclipse como si hiciera desde la línea de comandos:

Run -> Run Configurations



Argumentos en C y en Java

	C	Java
main	<code>int main(int argc, char*argv[])</code>	<code>public static void main(String[] args)</code>
Número de argumentos	Hay una variable entera contador: argc	Se sabe por el método: args.length
Almacenaje de argumentos	Se guardan en arreglo de caracteres con apuntadores	Se guardan en un arreglo de objetos String
Primer argumento	Se guardan desde el elemento argv[1] porque el elemento 0 es el nombre del programa	El primer elemento pasado se ubica desde args[0]
Tipo de dato	Deben convertirse a un tipo numérico para ser operados	Deben convertirse a un tipo numérico para ser operados

Argumentos en C y en Java

	C	Java
main	<code>int main(int argc, char*argv[])</code>	<code>public static void main(String[] args)</code>
Manejo de argumentos	Deben convertirse a un tipo numérico para ser operados: <code>int num=0;</code> <code>num= atoi(argv[1]);</code>	Deben convertirse a un tipo numérico para ser operados: <code>int num=0;</code> <code>num= Integer.parseInt(args[0]);</code>
Impresión en pantalla	Decimos donde colocar la variable dentro de la cadena: <code>printf("El area es: %f", area);</code>	Se concatena (une) la variable con la cadena: <code>System.out.println("El area es " + area);</code>
Lectura del teclado	De acuerdo al formato o tipo de dato deseado: <code>scanf("%f", &variable);</code>	Se toma la línea completa como un flujo de datos string y después de convierte: <code>entrada.readLine();</code>

Referencias

- Impresión con formato

- <http://puntocomnoesunlenguaje.blogspot.mx/2012/08/java-printf.html>