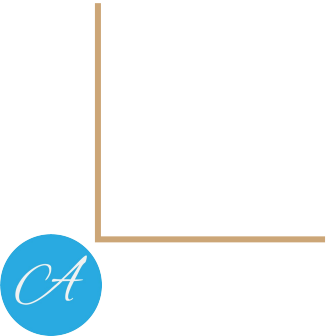


Diplomado de Programación en JAVA

Ing. Alejandro Leyva



1. Introducción al Lenguaje



¿Qué es JAVA?

Java es un lenguaje de programación de **propósito general**, concurrente, orientado a objetos, compilado, multi hilo.

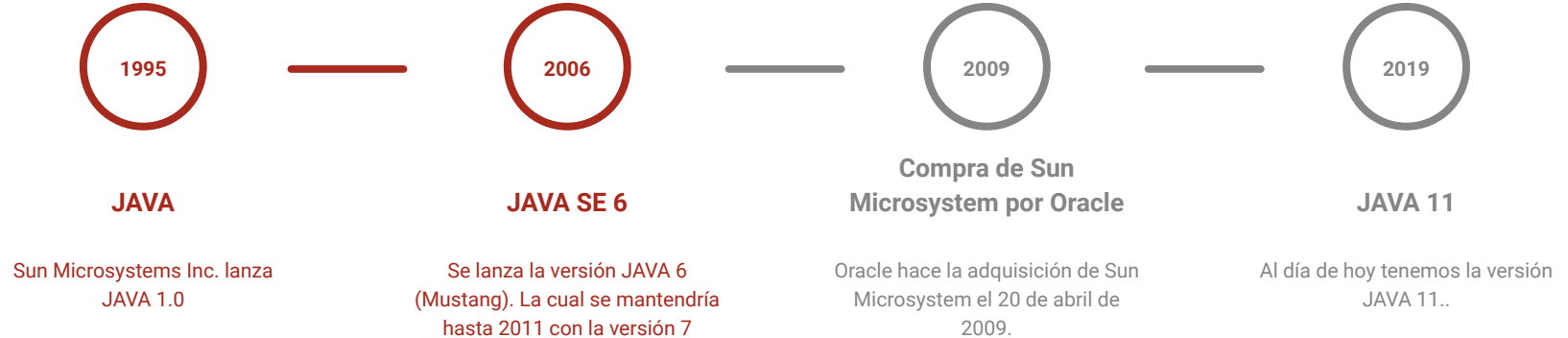
Permiten que los desarrolladores de aplicaciones *escriban el programa una vez y lo ejecuten en cualquier dispositivo* (conocido en inglés como WORA, o "*write once, run anywhere*").



¿Qué es JAVA?



Historia de JAVA



¿Por qué JAVA?

Multiplataforma

El código compilado (bytecode) es ejecutado en una máquina virtual (JVM)











Compilado

JAVA es un lenguaje compilado, esto lo hace rápido y seguro para cualquier tipo de aplicación.

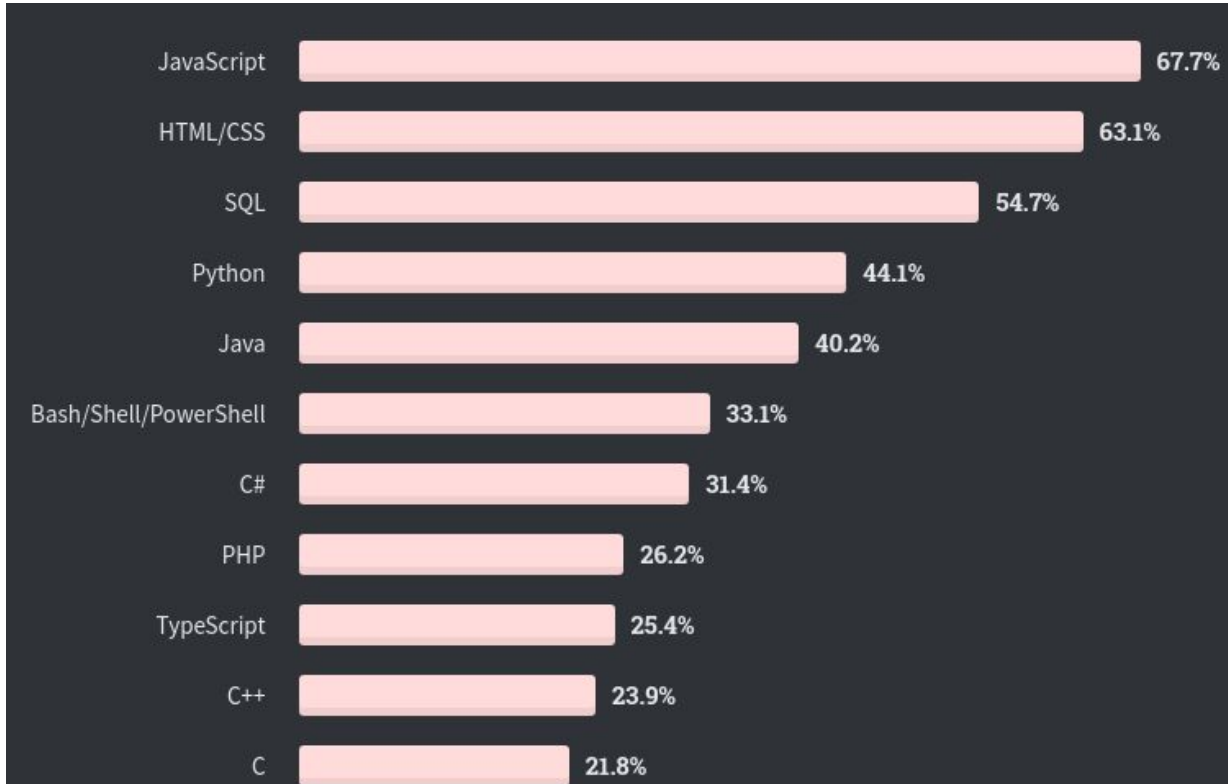
Android

Si deseas entrar al desarrollo móvil, debes saber JAVA.

¿Por qué JAVA?

Jun 2021	Jun 2020	Change	Programming Language		Ratings	Change
1	1			C	12.54%	-4.65%
2	3	▲		Python	11.84%	+3.48%
3	2	▼		Java	11.54%	-4.56%
4	4			C++	7.36%	+1.41%
5	5			C#	4.33%	-0.40%
6	6			Visual Basic	4.01%	-0.68%
7	7			JavaScript	2.33%	+0.06%
8	8			PHP	2.21%	-0.05%
9	14	▲▲		Assembly language	2.05%	+1.09%
10	10			SQL	1.88%	+0.15%

¿Por qué JAVA?



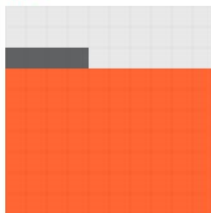
¿Por qué JAVA?

Programming languages

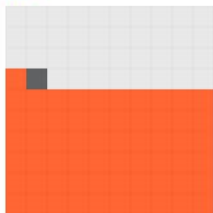
● Used in the last 12 months

● Planning to adopt or migrate

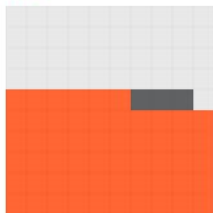
JavaScript
70% / 4%



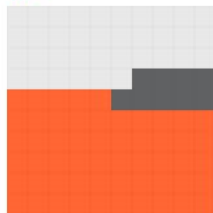
HTML / CSS
61% / 1%



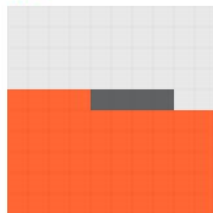
SQL
56% / 3%



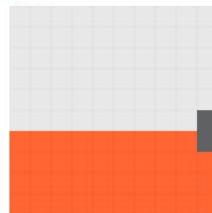
Python
55% / 9%



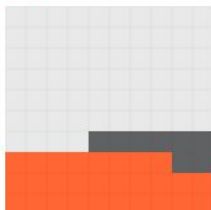
Java
54% / 4%



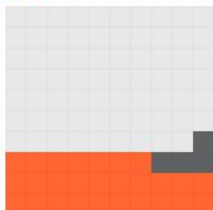
Shell scripting languages
39% / 2%



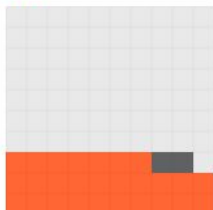
TypeScript
28% / 8%



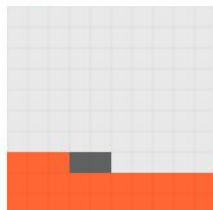
C++
27% / 4%



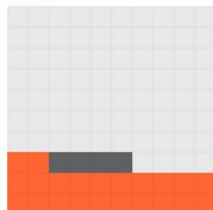
PHP
27% / 2%



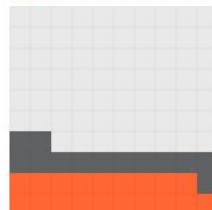
C
23% / 2%



C#
22% / 4%



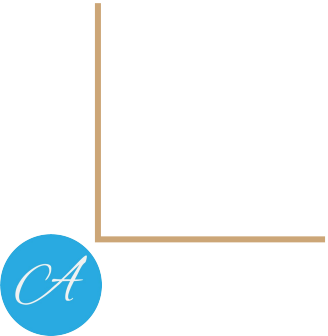
Go
19% / 13%



Proyectos que están desarrollados en JAVA



2. Configuración del Entorno de Desarrollo





Version 1.57 is now available! Read about the new features and fixes from May.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download for
Debian, Ubuntu...
Red Hat, Fedora...

Other platforms and languages

By using VS Code, you agree to its
license and privacy statement.















Editor: Visual Studio Code



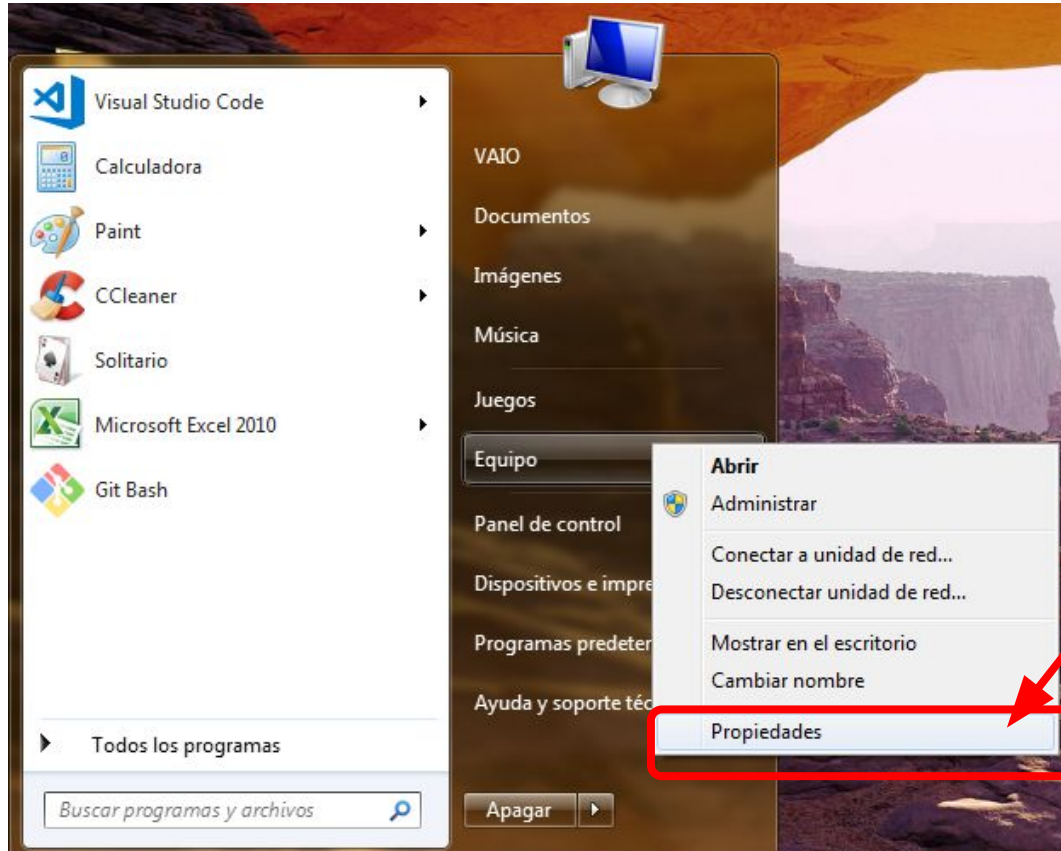
Kit de Desarrollo de JAVA (JDK)

Java SE Development Kit 8u291

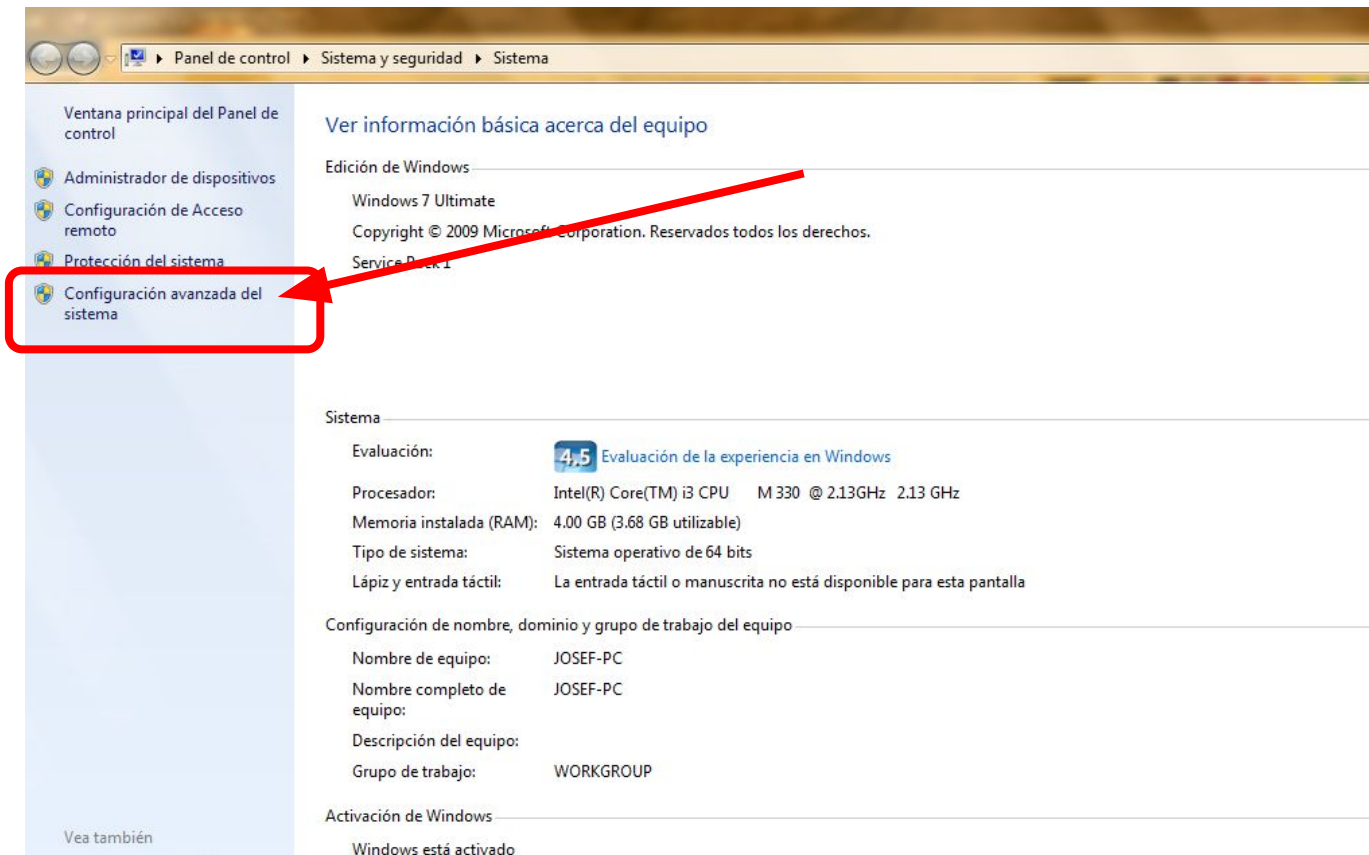
This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	59.1 MB	 jdk-8u291-linux-aarch64.rpm
Linux ARM 64 Compressed Archive	70.79 MB	 jdk-8u291-linux-aarch64.tar.gz
Linux ARM 32 Hard Float ABI	73.5 MB	 jdk-8u291-linux-arm32-vfp-hflt.tar.gz
Linux x86 RPM Package	109.05 MB	 jdk-8u291-linux-i586.rpm
Linux x86 Compressed Archive	137.92 MB	 jdk-8u291-linux-i586.tar.gz
Linux x64 RPM Package	108.78 MB	 jdk-8u291-linux-x64.rpm
Linux x64 Compressed Archive	138.22 MB	 jdk-8u291-linux-x64.tar.gz
macOS x64	207.42 MB	 jdk-8u291-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.69 MB	 jdk-8u291-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.74 MB	 jdk-8u291-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.48 MB	 jdk-8u291-solaris-x64.tar.Z
Solaris x64	92.56 MB	 jdk-8u291-solaris-x64.tar.gz
Windows x86	155.67 MB	 jdk-8u291-windows-i586.exe
Windows x64	168.67 MB	 jdk-8u291-windows-x64.exe

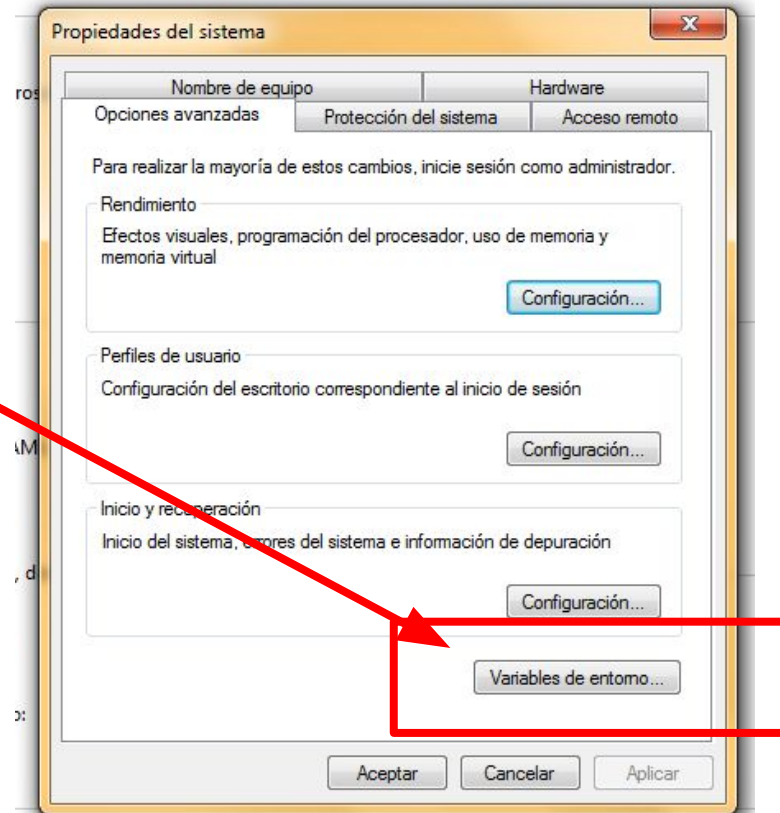
Configuración del CLASSPATH (Windows)



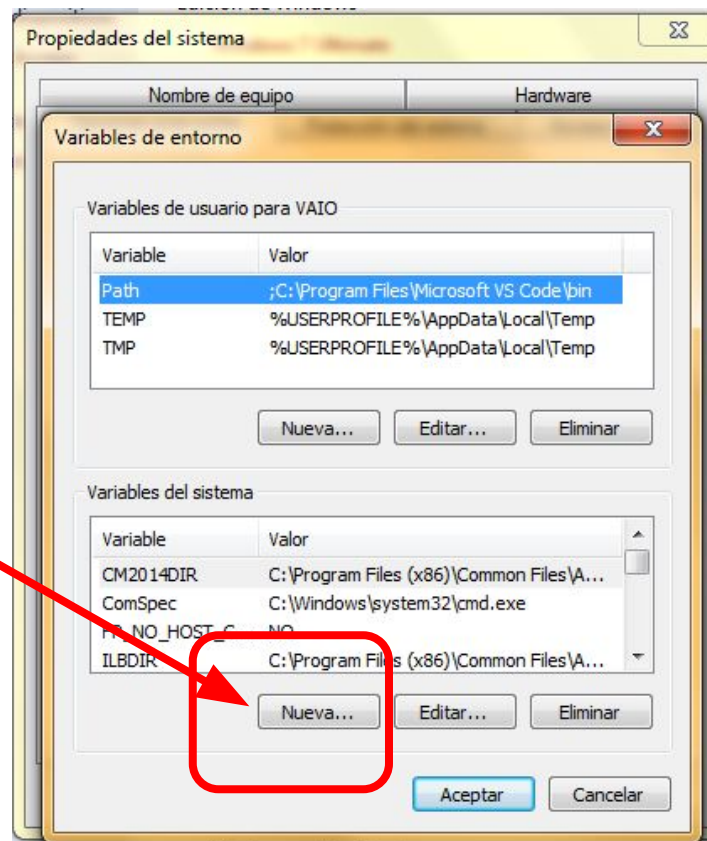
Configuración del PATH (Windows)



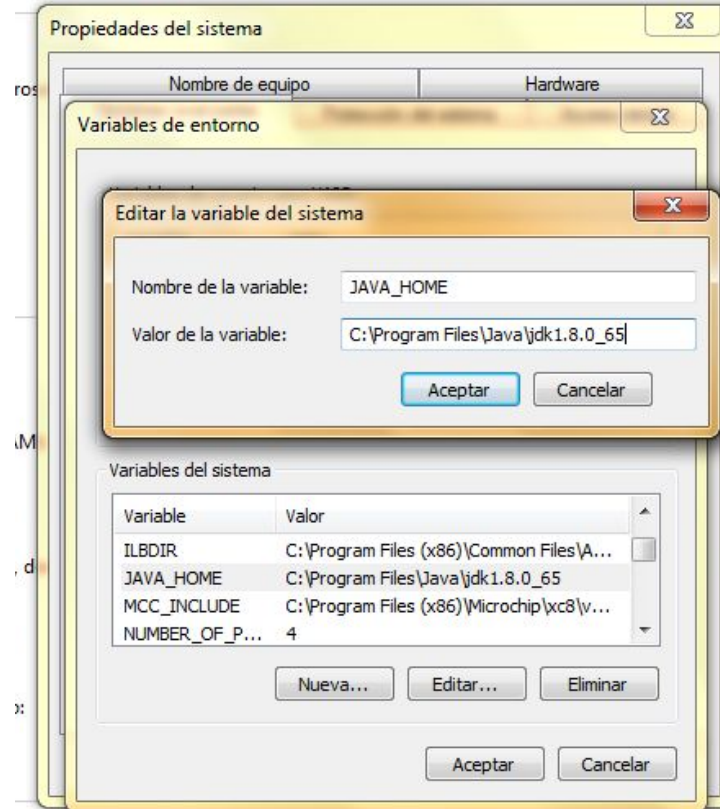
Configuración del PATH (Windows)



Configuración del PATH (Windows)

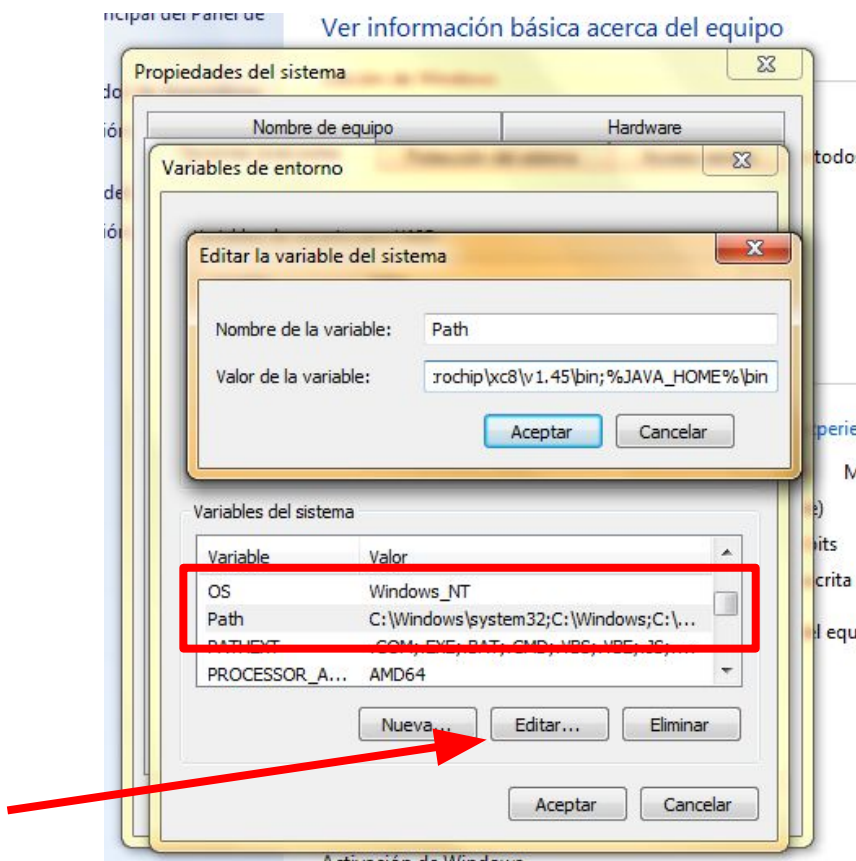


Configuración del PATH (Windows)



JAVA_HOME

Configuración del PATH (Windows)



;%JAVA_HOME%\bin

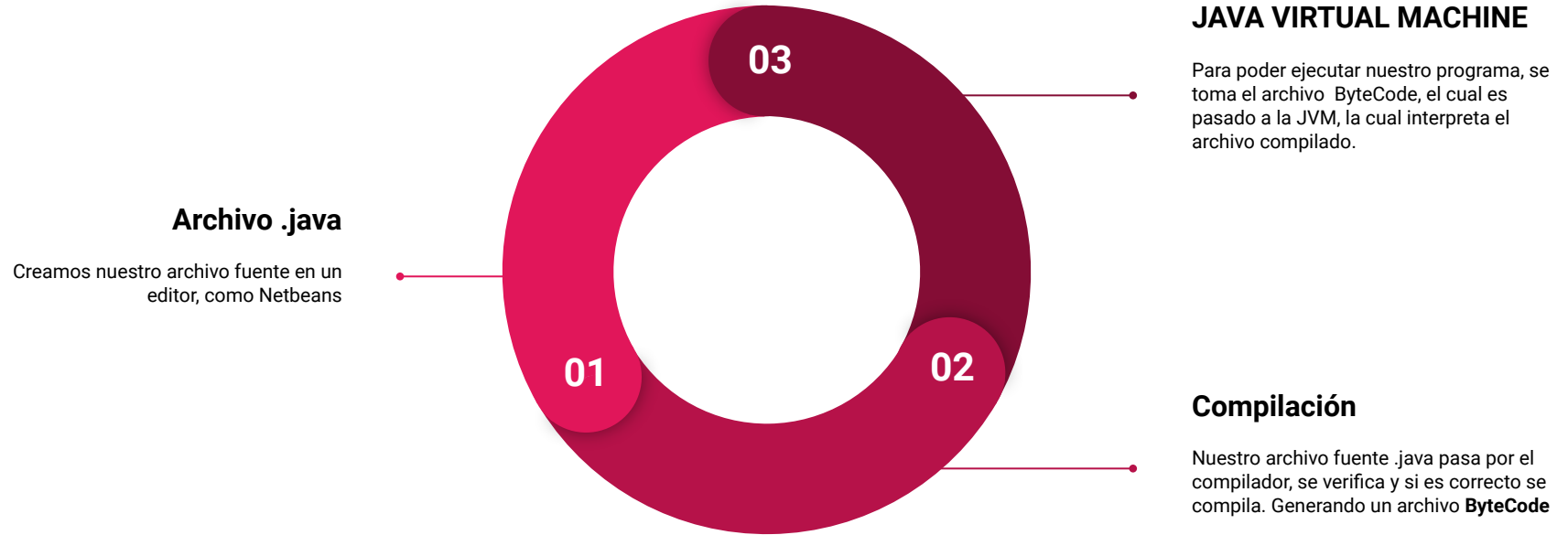
Comprobar que la JVM está configurada

`java --version`


```
→ ~ java --version
java 9.0.4
Java(TM) SE Runtime Environment (build 9.0.4+11)
Java HotSpot(TM) 64-Bit Server VM (build 9.0.4+11, mixed mode)
→ ~
```

```
→ ~ java -version
openjdk version "1.8.0_172"
OpenJDK Runtime Environment (Zulu 8.30.0.1-macosx) (build 1.8.0_172-b01)
OpenJDK 64-Bit Server VM (Zulu 8.30.0.1-macosx) (build 25.172-b01, mixed mode)
→ ~
```

Proceso de compilación y ejecución



Nuestro primer “Hola mundo”

 HolaMundo.java x

```
1  public class HolaMundo{  
2  
3      .... public static void main(String[] args){  
4          .... System.out.println("Hola mundo");  
5      .... }  
6  }
```

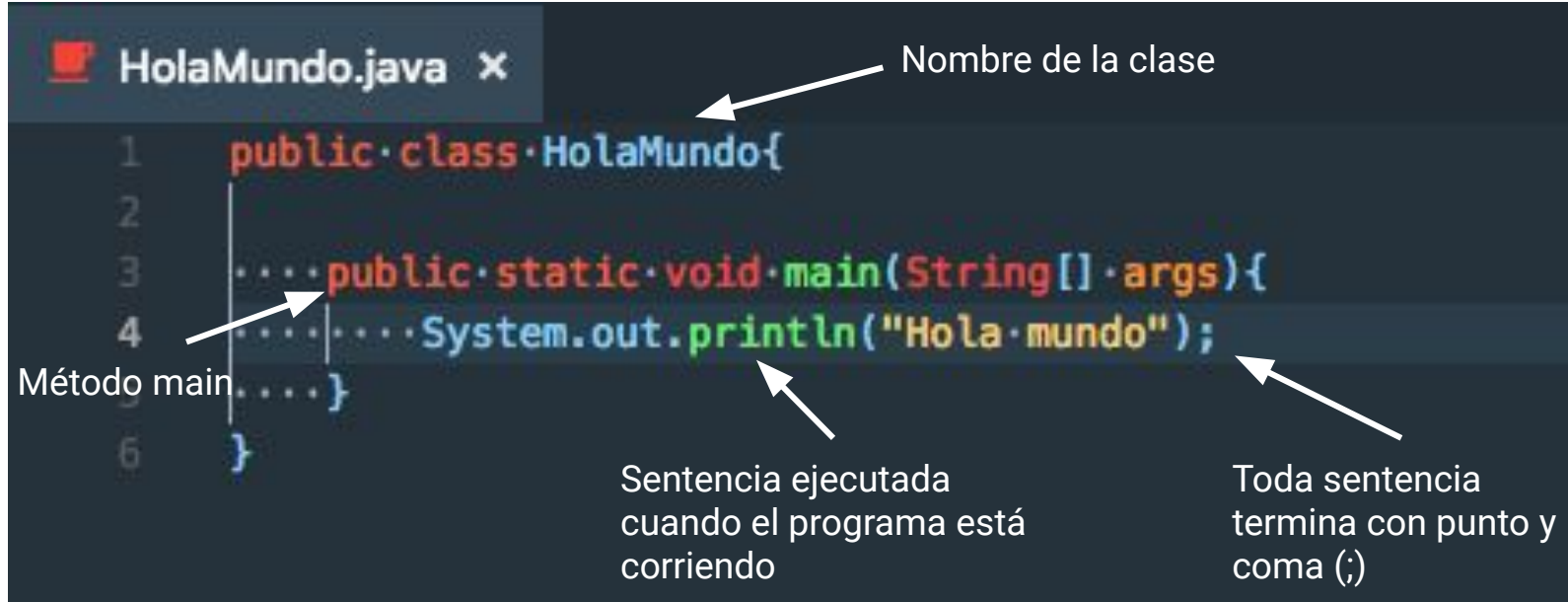
Proceso de compilación en terminal

javac *nombreArchivo.java* //compilación

java *nombreArchivo* //ejecución de programa

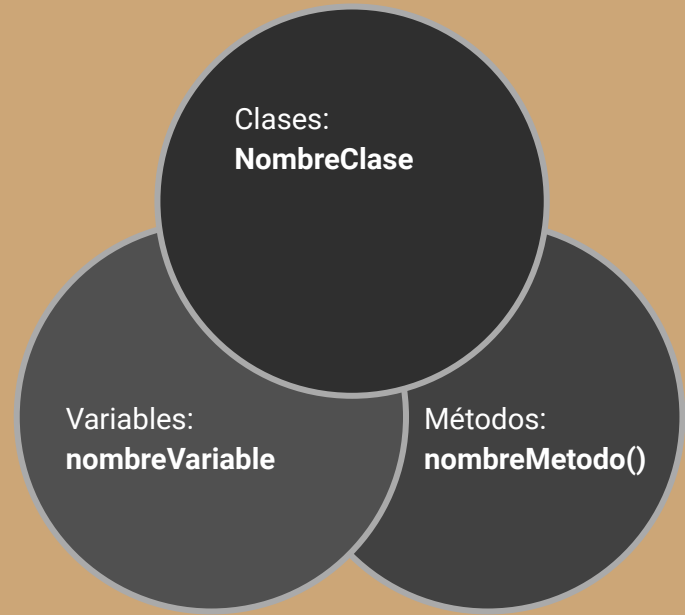
```
→ programas java javac HolaMundo.java
→ programas java java HolaMundo
Hola mundo
→ programas java █
```

Programa en JAVA



Buenas prácticas

Camel Case



¿Qué es una variable?

Es un espacio en memoria, que se usa para almacenar un valor.



Declaración de variables

tipoDeVariable *nombreVariable*; //declaración de variable

tipoDeVariable *nombreVariable* = valorAsignado;

int *miVariableEntera* = 5;

double *miVariableDoble* = 4.32;

Tipos de variables (Datos primitivos)

Nombre	Sintaxis	Rango de valores
booleano	boolean	True, false
byte	byte	-128 a 127
Entero corto	short	-32,768 a 32,767
Entero	int	-2,147,483,648 a 2,147,483,649
Entero largo	long	-9×10^{18} a 9×10^{18}
Doble	double	-1.79×10^{308} a 1.79×10^{308}
Flotante	float	-3.4×10^{38} a 3.4×10^{38}
Carácter	char	Caracteres

Identificadores

- **Sólo puede contener**

- Letras (A,B,C...Z)
- Números (0,1,2,...9)
- Guion bajo (_)
- Signo de peso (\$)

- **No puede**

- Comenzar con número
- Tener espacios
- Tener acentos
- *Si es una clase, debe comenzar con Mayúscula, de lo contrario será con minúscula, ejemplo: HolaMundo.java*
- *Sensible a mayúsculas y minúsculas*

Strings -Cadena de caracteres

Una cadena de caracteres es la unión o concatenación de varios caracteres.

Un carácter se coloca con comillas simples

```
char caracter = 'a';
```

Un string se coloca con comillas dobles

```
String mensaje = "Hoola :D ";
```

Concatenación

La concatenación solo se da con Strings, lo cual es unir o fusionar cadenas de caracteres para formar una más compleja.

```
String mensaje1 = "Hola";
```

```
String mensaje2 = "Mundo";
```

```
String mensajeCompleto = mensaje1 + " " + mensaje2; //concatenación
```

Secuencias de Escape

Nombre	Sintaxis
Salto de línea (Enter)	<code>\n</code>
Tabulador	<code>\t</code>
Retorno de carro	<code>\r</code>
Diagonal invertida	<code>\\</code>
Doble comillas	<code>\"</code>

Operadores Aritméticos

Nombre	Símbolo	Descripción
Asignación	=	Asignar un valor dado
Suma	+	Operación suma
Resta	-	Operación resta
Multiplicación	*	Operación producto
División	/	Operación división
Residuo	%	Retorna el residuo de la división

Ejercicio de Física - Segunda Ley de Newton

Calcular la fuerza si un objeto tiene una aceleración de 10m/s^2 y una masa de 2.5kg . Después, si el objeto incrementa su aceleración a 12.6m/s^2 .

$$F = m a$$



Operadores Aritméticos combinados

Nombre	Símbolo	Descripción
Suma	<code>+=</code>	<code>x = x + 3 -> x+=3</code>
Resta	<code>-=</code>	<code>x = x - 3 -> x-=3</code>
Multiplicación	<code>*=</code>	<code>x = x * 3 -> x*=3</code>
División	<code>/=</code>	<code>x = x / 3 -> x/=3</code>
Residuo	<code>%=</code>	<code>x = x % 3 -> x%=3</code>
Incremento	<code>++</code>	<code>x = x + 1 -> x++</code>
Decremento	<code>--</code>	<code>x = x - 1 -> x--</code>

Ejemplos - Operadores Aritméticos combinados

Realizar las siguientes operaciones e imprimir el resultado, utilizando los operadores aritméticos combinados.

()

Ejercicio - Conversiones

- Realizar programa para conversión de unidades, de centímetros a pulgadas y de pulgadas a centímetros.

$$2.54 \text{ cm} = 1 \text{ inch}$$



Método **print**

- ***print***("Texto"); //impresión básica
- ***println***("Texto"); //impresión con salto de línea
- ***printf***("Texto"); //impresión con formato

Ejemplo de **printf()**; es decir, impresión con formato

```
double peso = 85.3656;
```

```
printf("Mi peso es %.2f kgrs", peso); //impresión con formato
```

-> Mi peso es 85.37 kgrs <- Salida

Printf - Especificadores de formato

Carácter	Tipo de salida	Ejemplo
d	Entero	%d %5d
f	Flotantes y dobles	%f %2.2f
e	Con notación científica	%8.3e %e
s	String (Texto)	%s %12s
c	Caracter	%c %2c

3. Estructuras de Decisión y Control



Sentencia de decisión IF

```
if(condicionVerdadera){//si la condición se cumple entra al bloque del código
```

```
//en caso que sea verdadero, ejecuta este código
```

```
}
```

```
if(5 >= 4){
```

```
System.out.println("5 es mayor o igual a 4");
```

```
}
```

Operadores de relación

Operador	Descripción	Ejemplo	Resultado
<code>==</code>	Igual que	<code>8 == 9</code>	false
<code><</code>	Menor que	<code>9 < 4</code>	false
<code>></code>	Mayor que	<code>0 > -4</code>	true
<code><=</code>	Menor o igual que	<code>9 <= 20</code>	true
<code>>=</code>	Mayor o igual que	<code>3 >= 6</code>	false
<code>!=</code>	Diferente de	<code>4 != 4</code>	false

Ejercicio - Aprobado-Reprobado

Hacer un programa que nos indique si el alumno aprobó o reprobó la materia.



Ejercicio - Conociendo si es número es par o impar

Realizar un programa que diga si el número es par o impar y si el número es mayor 10, que diga un mensaje que el dígito dado es superior a 10.



Operadores lógicos

AND	
Operación	Resultado
False && False	False
False && True	False
True && False	False
True && True	True

OR	
Operación	Resultado
False False	False
False True	True
True False	True
True True	True

NOT	
Operación	Resultado
!True	False
!False	True

Operadores lógicos

Nombre	Símbolo	Aplicación	Resultado
AND	&&	(5 == 5) && (4==4)	True
OR	 	(9 > 3) false	True
NOT	!	!false	True

Ejercicio - Aprobado-Reprobado con mensaje

Realizar un programa que diga una frase dependiendo de su calificación.

- Si obtuvo menos de 6, pero más de 0 -> *"Lastima Margarito"*
- Si obtuvo de 6 hasta menos de 7-> *"De panzazo"*
- Si obtuvo de 7 hasta menos de 8 -> *"Echale más punch"*
- Si obtuvo de 8 hasta menos de 9 -> *"Bien, puedes mejorar"*
- Si obtuvo de 9 hasta menos de 10 -> *"Muy bien, te faltó tantito"*
- Si obtuvo 10 -> *"Excelente, con toda la actitud"*
- Si da otro valor que no esté definido dirá *"No es posible"*

Leyendo datos del teclado

Se importa el objeto Scanner, se genera una instancia.

```
import java.util.Scanner; //se importa la librería, debe ir al inicio del archivo
```

```
Scanner leer = new Scanner(System.in); //crea instancia dentro de main
```

```
int entero = leer.nextInt(); //lee y guarda entero
```

```
double doble = leer.nextDouble(); //lee y guarda doble
```

```
double doble = leer.next(); //lee y guarda String
```


Sentencia de decisión IF-ELSE

```
if(condicionVerdadera){
```

```
    //en caso que sea verdadero, ejecuta éste código
```

```
}else{
```

```
    //en caso contrario, se ejecuta éste código
```

```
}
```

Ejercicio - Qué sexo eres

Generar un programa que pregunte qué sexo eres, 1 si es Hombre que diga “Macho alfa lomo plateado”, de lo contrario que diga “Eres una linda señorita”.



If anidado

```
if(condicionVerdadera){
```

```
    //en caso que sea verdadero, ejecuta éste código
```

```
}else if(condicionVerdadera){
```

```
    //de lo contrario si, se ejecuta
```

```
}else{
```

```
    //en caso contrario, se ejecuta éste código
```

```
}
```

If anidado

```
if(condicionVerdadera){
```

```
    //en caso que sea verdadero, ejecuta éste código
```

```
    if(condicionVerdadera){...}else{...}
```

```
}else if(condicionVerdadera){
```

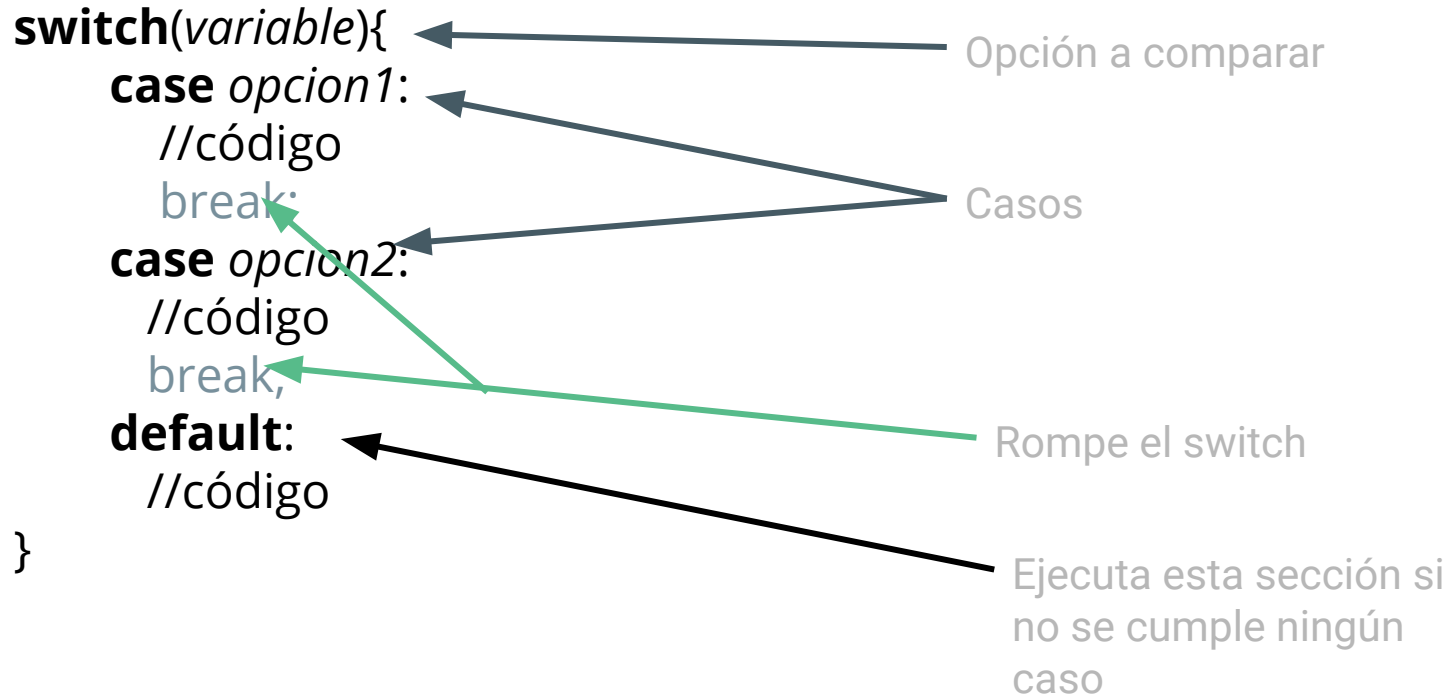
```
    //de lo contrario si, se ejecuta
```

```
    }else{
```

```
        //en caso contrario, se ejecuta este código}
```

```
}
```

Sentencia de decisión SWITCH



Ejercicio - Calculadora básica

Crear un menú dando las opciones para seleccionar que se desea calcular. Opciones: 1. Suma, 2. Resta, 3 Multiplicación, 4 División y al final arrojar el resultado de la operación, en caso que no exista la operación, lanzará el mensaje que no existe dicha operación.



Operador ternario

El operador ternario es una especie de if-else simplificado en una línea, que devuelve de manera directa un resultado. Tiene que devolver algún valor forzosamente.

Sintaxis

tipo *variable* = (**condicion**) ? *cumple* : *no_cumple* ;

Ejemplo:

bool *variable* = (**5 > 6**) ? *true* : *false* ;

Sentencias de control - FOR

Separado por punto y coma (;)

```
for( inicio ; condición ;  $\Delta$  ){
```

//código que se va a repetir hasta que la condición sea falsa

```
}
```

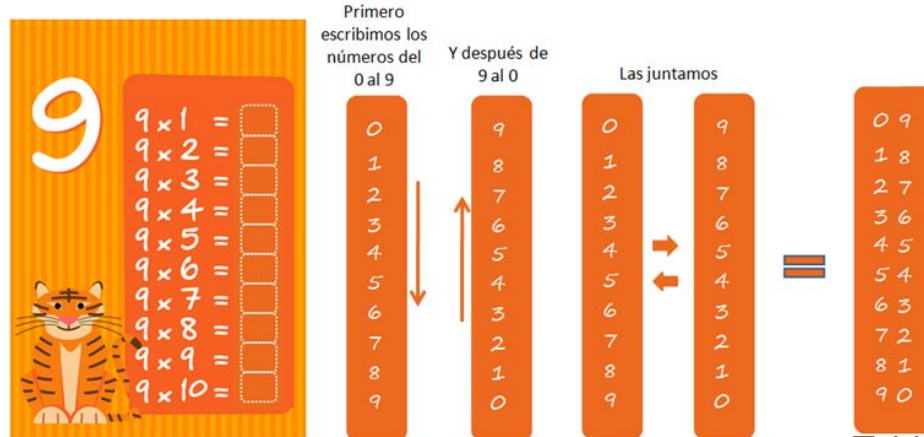
```
for( inicio ; tope ; incremento/decremento ){
```

//código que se va a repetir hasta que la condición sea falsa

```
}
```


Ejercicio - Imprimiendo tablas de multiplicar

- Realizar un programa que imprima la tabla de 7, que hasta la multiplicación hasta el 10.
- Realizar un programa que realice la tabla que el usuario quiera conocer, debe llegar hasta el 10 la multiplicación.



Ejercicio - Media

- Solicitar al usuario la cantidad de números que va a ingresar de un conjunto, e ir pidiendo uno a uno, al final dar el resultado de la media.

$$\bar{X} = \frac{\sum_{i=1}^n x_i}{n}$$

Break y Continue

- Solicitar al usuario 8 números y el programa ignorará los números pares.
- Solicitar al usuario números de un conjunto positivos, e ir pidiendo uno a uno, al final dar el resultado de la media, para salir debe dar el valor de -1 (*for infinito*).

$$\bar{X} = \frac{\sum_{i=1}^n x_i}{n}$$

Arreglos (array)

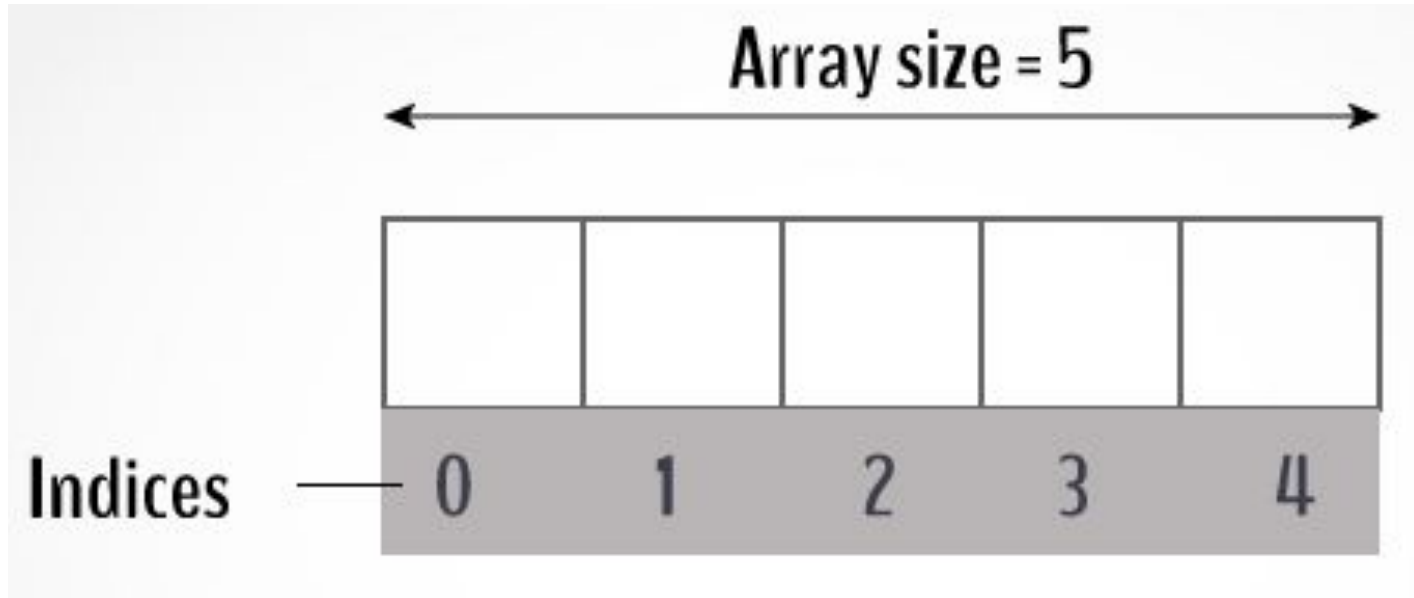
Es una estructura de datos, una colección de elementos, en éste caso es una colección de referencias.

Características:

- Espacio definido
- Índice de posición
- Solo puede contener un tipo de elemento



Arreglos (array)



Arreglos (array)

tipo *nombre*[] = new tipo[tamaño]; //declaración vacío pero su espacio definido

tipo *nombre*[] = {valor1, valor2, valor3}; //asignando los valores

int ***miArreglo***[] = new int[4]; //array con 4 espacios

int ***segundoArreglo***[] = {4, 3, 7, 9}; //array con 4 espacios

Arrays - Ejemplos

Crear un array de tipo String, cargarle 3 textos distintos y después imprimirlos todos a través de un for.

Crear un **array vacío de 6 posiciones** e ingresarle los valores con un *for* en la siguiente secuencia 10, 20...50, 60 en su respectiva posición. *Ejemplo: index 0 el valor 10, index 1 el valor 20*

Crear un array de tipo entero de 4 posiciones con valores predefinidos, imprimir todos sus valores con un for, posteriormente multiplicar todos esos valores por 3 e irlos guardando en el mismo array, sustituyendo los valores anteriores, e imprimir los nuevos valores recorriendo el array de nuevo.

Arrays - Estadística

Realizar programa que calcule la media y la desviación estándar de un conjunto de datos que ingrese el usuario, previamente se solicita el total de datos.

$$\bar{X} = \frac{\sum_{i=1}^n x_i}{n}$$

$$s = \sqrt{\frac{\sum_{i=1}^N (Xi - \bar{X})^2}{N - 1}}$$

Array *nxm*

Array bidimensionales, tridimensionales, de dimensión $n \times m$.

```
int miArreglo[][] = new int[4][4]; //array de 4 x 4
```

```
int segundoArreglo[] = {
```

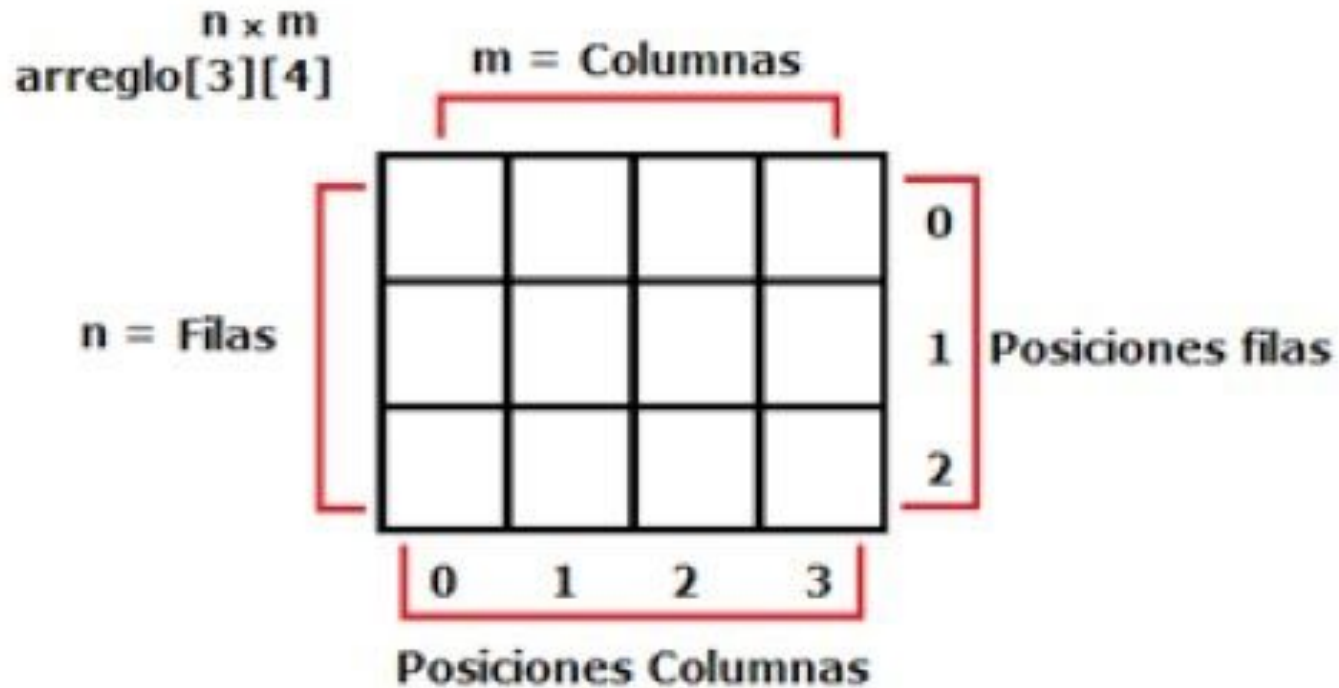
```
    { 4, 3, 7, 9 },
```

```
    { 4, 3, 9 },
```

```
    { 4, 3, 7, 9 }
```

```
}; //array con 2 x 4
```

Array $n \times m$



Array *nxm*

- Realizar la combinación de un nombre con un color al azar, debe estar contenido en un array, tres nombres y 8 colores. Cargando todos los datos desde un inicio.
- Realizar un combinador de 5 parejas, en un array bidimensional, pedir los nombres de los hombres y mujeres, posterior hacer parejas aleatorias. Sin importar que se repitan.



Sentencias de control - WHILE & DO-WHILE

while(*condicionVerdadera*){

 //código que se ejecuta mientras la
 condición se cumpla (true)

}

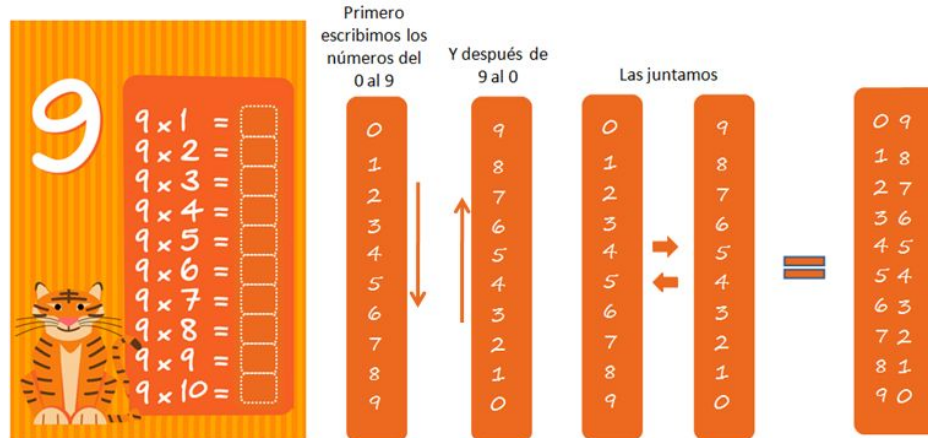
do{

 //código que se ejecuta mientras la
 condición se cumpla (true), pero entra la
 primera vez

}**while**(*condicionVerdadera*);

Ejercicio - Imprimiendo tablas de multiplicar (while)

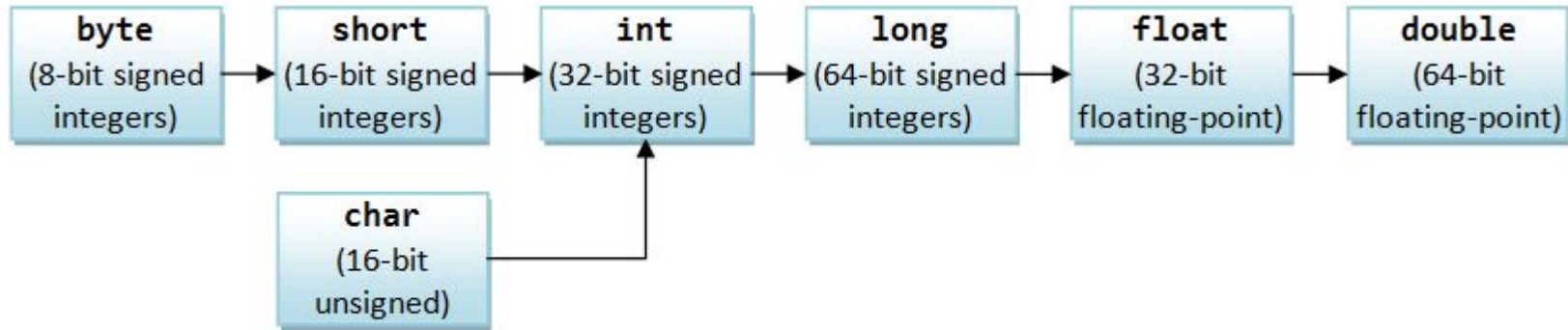
- Realizar un programa que imprima la tabla de 9, que llegue hasta el 10.
- Realizar un programa que realice la tabla que el usuario quiera conocer, debe llegar hasta el 10.
- Juego: Adivina el número



Casting de tipos primitivos

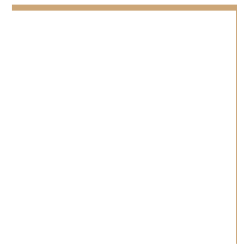
Hacer conversión de tipos de datos primitivos.

```
double suma = (double) 4;
```

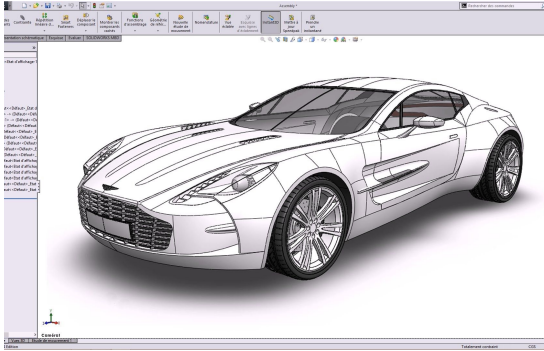


Orders of Implicit Type-Casting for Primitives

4. Objetos



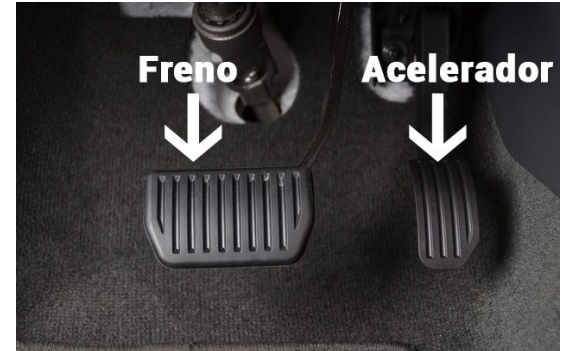
Clases - Objetos -> Instancia



Clase



Objeto (instancia)



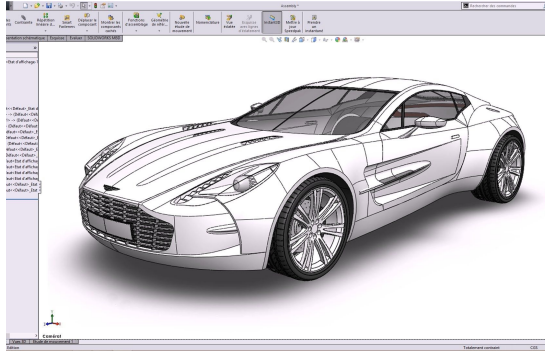
**Atributos y
Comportamientos**

Objetos



Que es una instancia

Es la creación de un objeto que se va a utilizar dentro de una clase o método.



Clase



Objeto (instancia)

¿Qué es un método?

- Es un comportamiento (acción) que realiza un objeto (cosa).
- Una secuencia de pasos ordenados.
- Es un bloque o secuencia de código que se repite continuamente.
- Hace una sola tarea, y lo hace muy bien.
- Su nombre se define con un verbo (acción).
- Funciones de un objeto.
- Modifica estados.

Creación de Objetos

Objeto = Clase

Nombre de clase

public class **MiObjeto**{ //inicia la clase

int campo1;

Atributo = Campo

public void **miMetodo**(){

//cuerpo del método

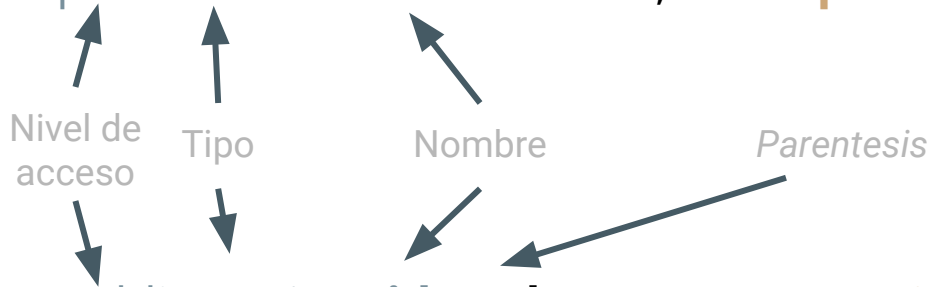
Método = Comportamientos

}

}//fin de la clase

Campos y Métodos Estáticos

`public static int noPuertas; //campo de tipo entero`



`public static void acelerar () { //comienza el método, tipo void`

`System.out.println("acelerando");`

Cuerpo del método
(Lo que hará el método)

`}`

Tipos para métodos

Nombre	Descripción
void	No devuelve ningún dato, solo realiza una acción
int	Devuelve un valor entero
long	Devuelve un valor entero largo
float	Devuelve un valor flotante
double	Devuelve un valor flotante largo
char	Devuelve un valor tipo carácter
byte	Devuelve un valor tipo byte
Objeto	Devuelve el tipo del objeto

Creando Clase

Comportamientos:

- *static* acelerar(); + void
- *static* arrancar(); + void



Metodos estáticos

```
public static void nombreMetodoEstatico( ){
```

```
    //cuerpo del método
```

```
}
```

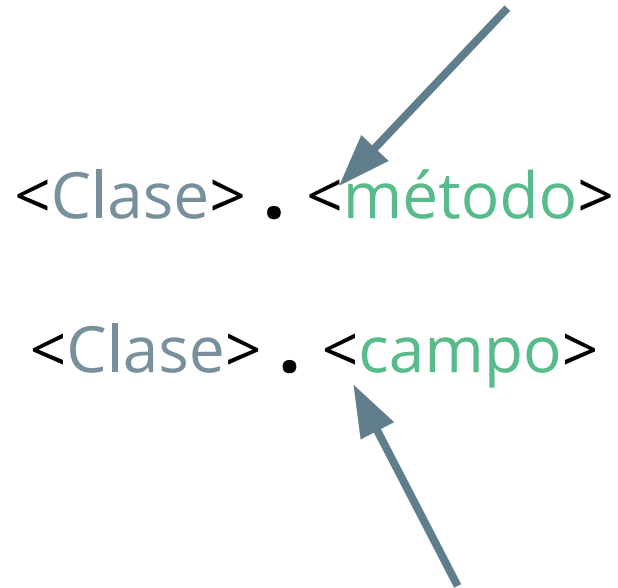
```
public static void nombreMetodoEstatico(tipo argumento ){
```

```
    //cuerpo del método
```

```
}
```

- No se requiere que se genere una **instancia**

Llamada métodos y campos *static*



<Clase> . <método>

<Clase> . <campo>

The diagram illustrates the syntax for calling static methods and accessing static fields. It consists of two lines of code. The first line is '<Clase> . <método>', where '<método>' is highlighted in green and a blue arrow points to it from the top right. The second line is '<Clase> . <campo>', where '<campo>' is highlighted in green and a blue arrow points to it from the bottom right.

Objeto- Saludo

Métodos:

- saludar(): + void
- despedida(): +void



Métodos estáticos con *argumentos*

Nivel de acceso Tipo Nombre del método Argumentos separados por coma (,)

↓ ↓ ↓

```
public static void    comer ( String nombreComida) {  
    System.out.println("Como una rica " + nombreComida);  
}
```

Cuerpo del método

Métodos con *return*

Nivel de acceso Tipo Nombre del método Argumentos separados por coma (,)

↓ ↓ ↓

```
public static int    anioNacimiento ( int anioNacimiento, int anioActual) {  
    int edad = anioActual - anioNacimiento;  
    return edad;  
}
```

Cuerpo del método

Valor que retorna, regresa a quien lo llamó

Objeto - Ley de Ohm

Crear una librería para la solución de Ley de Ohm.



Alcance de variables

```
public class MiClase{
```

```
    public int variable1; //variable global
```

```
    public void metodo1() {
```

```
        int variable2; //variable local
```

```
        for(int variable3 = 0; variable < 6; variable++){
```

```
            //cuerpo del for
```

```
        }
```

```
    }
```

```
}
```

Campos *final*

Una variable tipo FINAL significa que ese valor jamás va a cambiar, no se puede cambiar una vez dado un valor.

La forma de declararlo es en Mayúsculas y separados por guión bajo (_)

```
final double PI = 3.141592653589793;
```

```
final long VELOCIDAD_DE_LA_LUZ = 300000000;
```

Libreria *Math*

- **sqrt()**: Raíz cuadrada
- **pow()**: potencia
- **sin()**: seno
- **cos()**: coseno
- **abs()**: valor absoluto

SPHERE

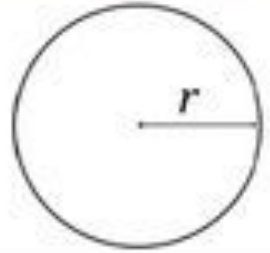
$$S = 4\pi r^2$$
$$V = \frac{4\pi r^3}{3}$$



CIRCLE

$$P = 2\pi r$$

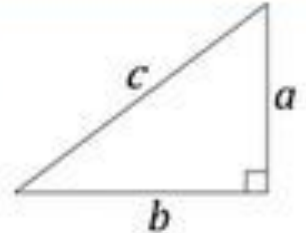
$$A = \pi r^2$$



PYTHAGOREAN THEOREM

$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}$$



FIGURAS GEOMÉTRICAS PL

Objeto - Figuras geométricas

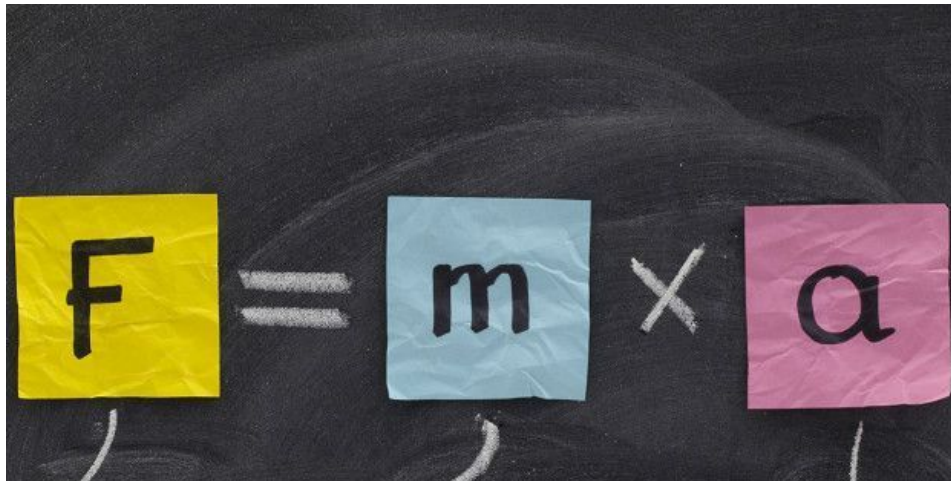


Crear una **librería** para el cálculo de áreas y perímetros de figuras geométricas.

- Cuadrado
- Rectángulo
- Círculo

Objeto - Segunda Ley de Newton

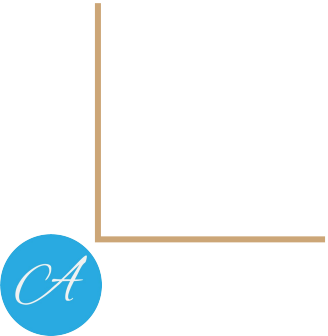
Crear una libreria de la segunda Ley de Newton, teniendo como campo *static final la gravedad* de la tierra (9.81 m/s²).



A photograph of the equation $F = m \times a$ written on a chalkboard. The letters are represented by colored sticky notes: a yellow square for 'F', a light blue square for 'm', and a pink square for 'a'. The equals sign and the multiplication symbol are drawn with white chalk. The background is a dark, textured chalkboard surface.

$$F = m \times a$$

5. Manejo de Errores Básico



Excepciones

Ocurre cuando hay un problema.

Permite escribir programas tolerantes a fallas y robustos.

Sucede cuando se quiere realizar una acción que no es posible.

Se lanza cuando el programa no sabe qué hacer al recibir un tipo diferente.

Se ejecuta cuando no encuentra algún archivo que necesite.

Errores más comunes

Al realizar una división por cero, se genera un error matemático y por consiguiente de lógica de programación.

Al recorrer un array y se intenta acceder **a un índice que no existe**, nos lanza un error de que la posición no existe.

Un error común es cuando se quiere acceder a un objeto (su referencia) y no tienen (*null*).

Manejo de Excepciones

```
try{
```

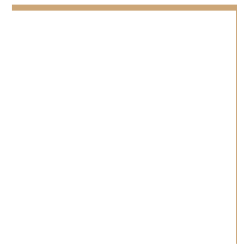
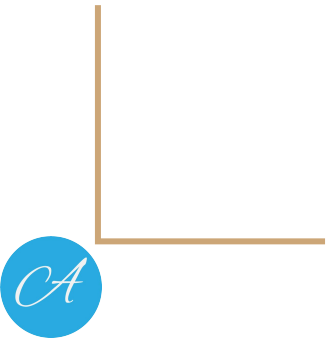
```
    //ejecución de código normal
```

```
}catch(Exception nombre){
```

```
    //si sucede un error, se trata el error aquí
```

```
}
```

6. Colecciones



List, *ArrayList*

Es una colección ordenada que puede contener elementos duplicados.

Es una **interfaz** que implementa ***ArrayList, LinkedList y Vector***.

Es una interfaz genérica.

El más usado por su velocidad es **ArrayList**.

foreach

```
for(TipoBase elemento: Colección ){
```

```
    //Cuerpo del for
```

```
}
```

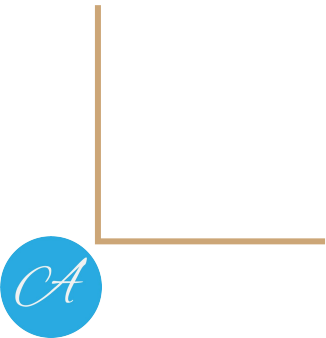
```
ArrayList<String> valores = new ArrayList<>();
```

```
for(String value: names ){
```

```
    //Cuerpo del for
```

```
}
```

7. Aplicación



Definiendo proyecto (yoyo)

Aplicación de terminal para extraer información de una e-commers, guardando esa información en un archivo csv.

