

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

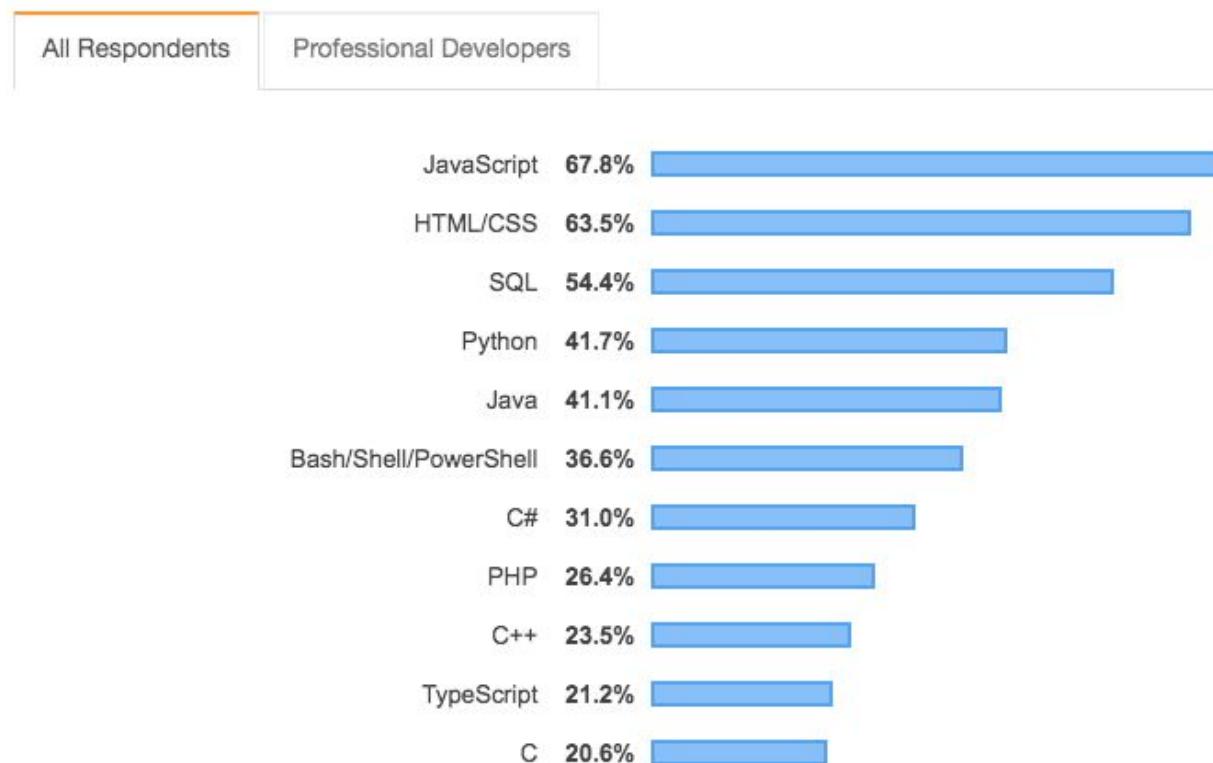
Si deseas entrar al desarrollo móvil, debes saber JAVA.

¿Por qué JAVA?

Jun 2019	Jun 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.004%	-0.36%
2	2		C	13.300%	-1.64%
3	4	▲	Python	8.530%	+2.77%
4	3	▼	C++	7.384%	-0.95%
5	6	▲	Visual Basic .NET	4.624%	+0.86%
6	5	▼	C#	4.483%	+0.17%
7	8	▲	JavaScript	2.716%	+0.22%
8	7	▼	PHP	2.567%	-0.31%
9	9		SQL	2.224%	-0.12%
10	16	▲	Assembly language	1.479%	+0.56%

¿Por qué JAVA?

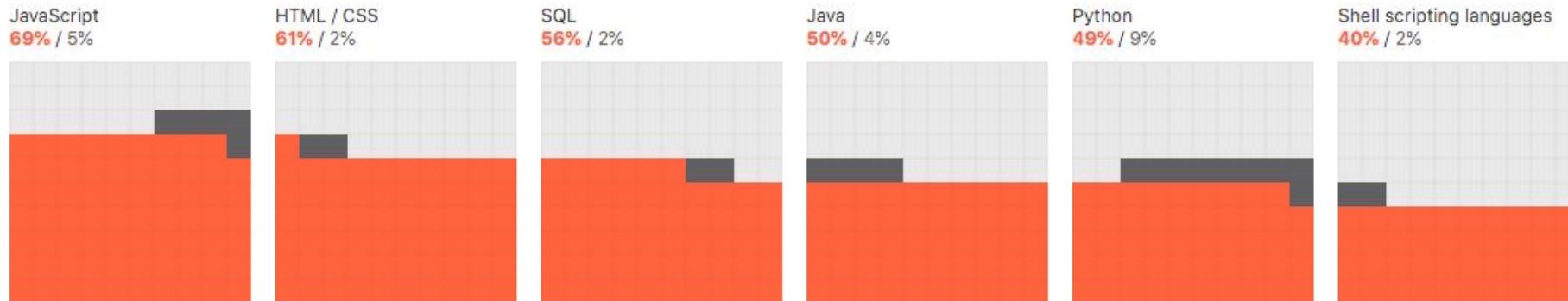
Programming, Scripting, and Markup Languages



¿Por qué JAVA?

What programming languages have you used in the last 12 months?

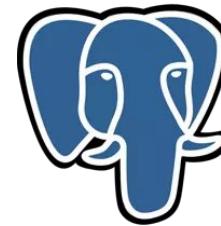
- Used in the last 12 months
- Plan to adopt / migrate



Proyectos que están desarrollados en JAVA



NetBeans



PostgreSQL



2. Configuración del Entorno de Desarrollo

Editor: Visual Studio Code

Visual Studio Code

Docs

Updates

Blog

API

Extensions

FAQ

Search Docs

Download

Version 1.35 is now available! Read about the new features and fixes from May.

Code editing. Redefined.

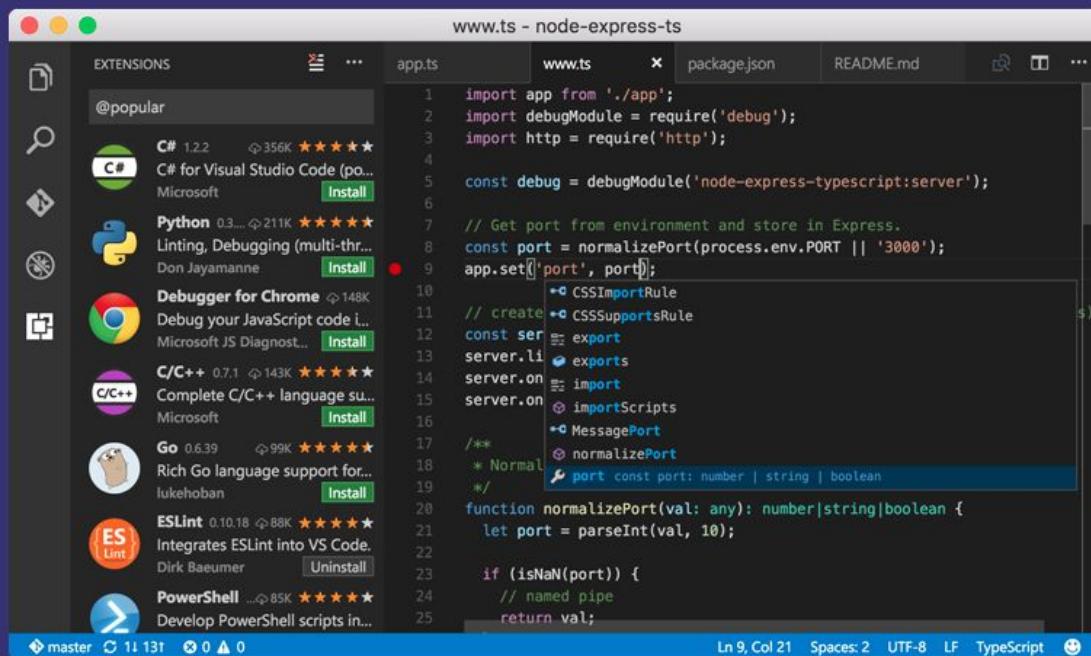
Free. Built on open source. Runs everywhere.

Download for Mac

Stable Build

Other platforms and Insiders Edition

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



www.ts - node-express-ts

```
import app from './app';
import debugModule = require('debug');
import http = require('http');

const debug = debugModule('node-express-typescript:server');

// Get port from environment and store in Express.
const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

// create
const server = app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});

server.on('error', (err) => {
  if (err.syscall === 'listen') {
    const address = typeof port === 'string' ? 'localhost' : `0.0.0.1`;
    const port = port === '' ? 3000 : port;
    const error = new Error(`Port ${port} is already in use on ${address}`);
    error.code = 'EADDRINUSE';
    throw error;
  } else {
    throw err;
  }
});

server.on('listening', () => {
  const address = server.address();
  const port = typeof address === 'string' ? 'localhost' : `0.0.0.1`;
  const port = address === '' ? 3000 : address;
  console.log(`Server listening on ${port}`);
});

// importScripts
// MessagePort
// normalizePort
// port const port: number | string | boolean

function normalizePort(val: any): number|string|boolean {
  const port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port < 0) {
    throw new Error(`Port ${port} must be a positive integer`);
  }

  return port;
}

// export
// exports
// import
// importScripts
// MessagePort
// normalizePort
// port const port: number | string | boolean

if (isNaN(port)) {
  // named pipe
  return val;
}
```

EXTENSIONS

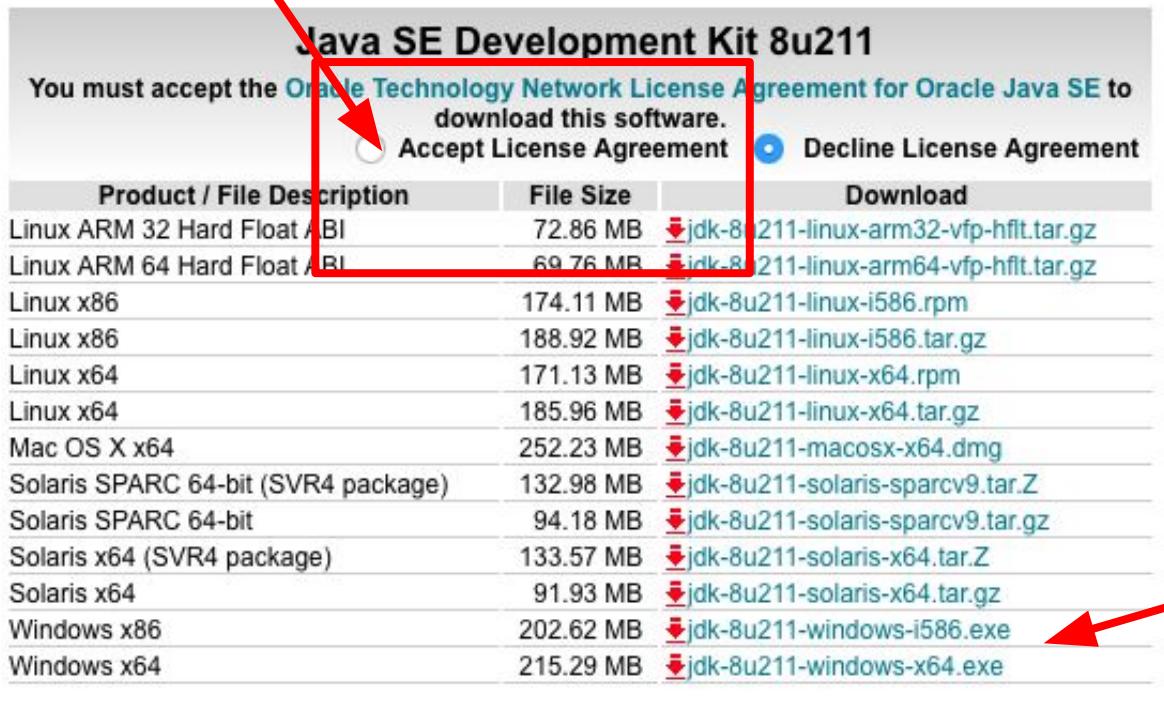
- C#** 1.2.2 356K ★★★★★ Microsoft [Install](#)
- Python** 0.3... 211K ★★★★★ Don Jayamanne [Install](#)
- Debugger for Chrome** 148K Microsoft JS Diagnos... [Install](#)
- C/C++** 0.7.1 143K ★★★★★ Microsoft [Install](#)
- Go** 0.6.39 99K ★★★★★ lukehoban [Install](#)
- ESLint** 0.10.18 88K ★★★★★ Integrates ESLint into VS Code. Dirk Baeumer [Uninstall](#)
- PowerShell** ... 85K ★★★★★ Develop PowerShell scripts in...

master 11 131 0 0

Ln 9, Col 21 Spaces: 2 UTF-8 LF TypeScript

Kit de Desarrollo de JAVA (JDK)

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



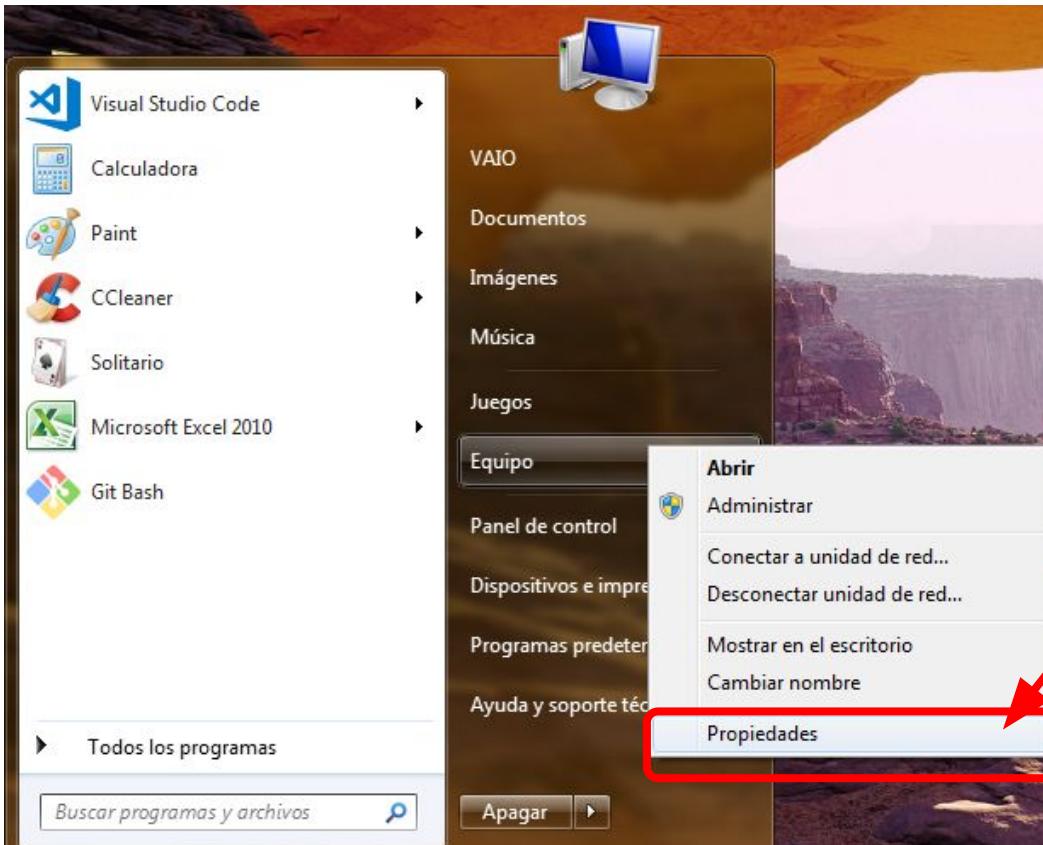
Java SE Development Kit 8u211

You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

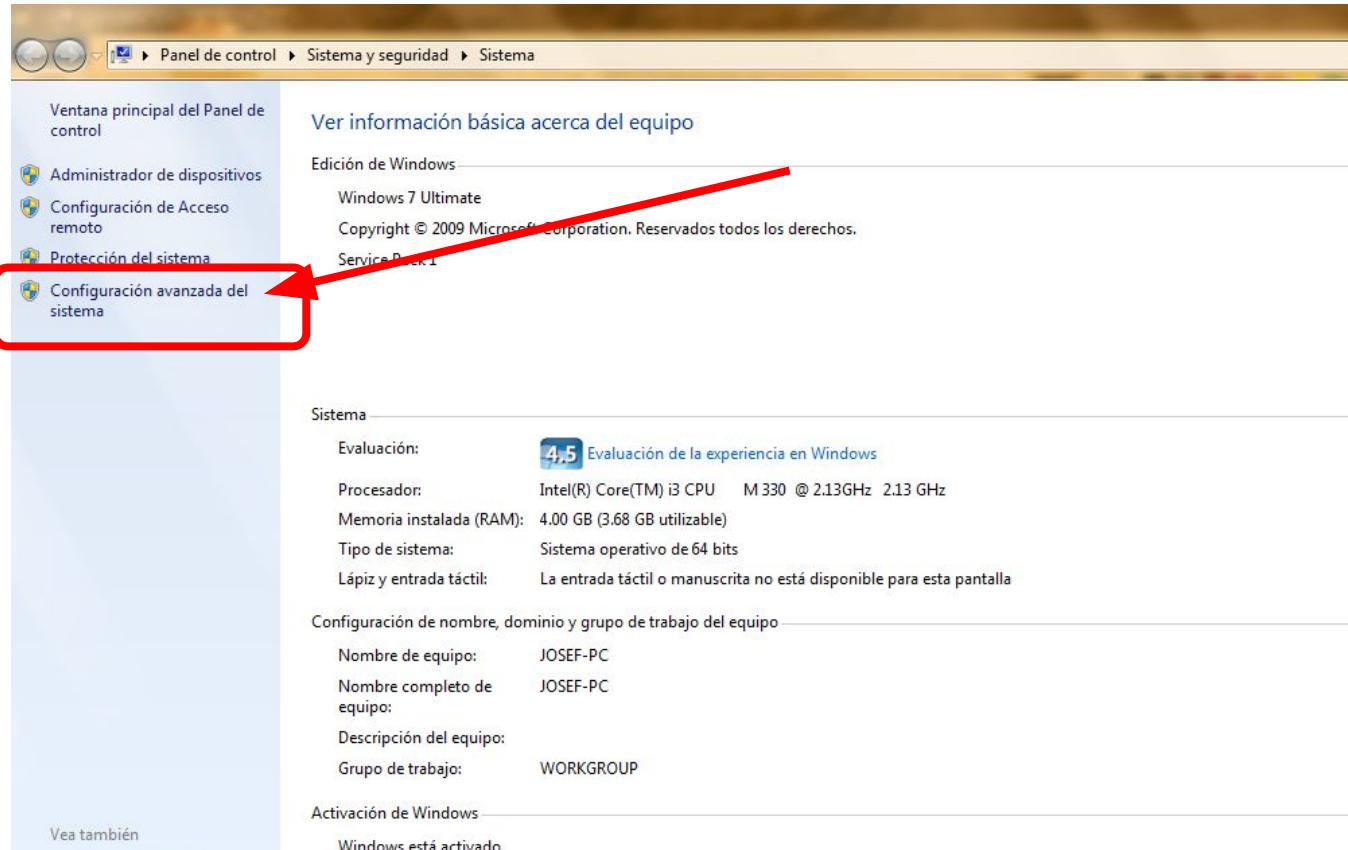
Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.86 MB	jdk-8u211-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.76 MB	jdk-8u211-linux-arm64-vfp-hflt.tar.gz
Linux x86	174.11 MB	jdk-8u211-linux-i586.rpm
Linux x86	188.92 MB	jdk-8u211-linux-i586.tar.gz
Linux x64	171.13 MB	jdk-8u211-linux-x64.rpm
Linux x64	185.96 MB	jdk-8u211-linux-x64.tar.gz
Mac OS X x64	252.23 MB	jdk-8u211-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	132.98 MB	jdk-8u211-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.18 MB	jdk-8u211-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.57 MB	jdk-8u211-solaris-x64.tar.Z
Solaris x64	91.93 MB	jdk-8u211-solaris-x64.tar.gz
Windows x86	202.62 MB	jdk-8u211-windows-i586.exe
Windows x64	215.29 MB	jdk-8u211-windows-x64.exe

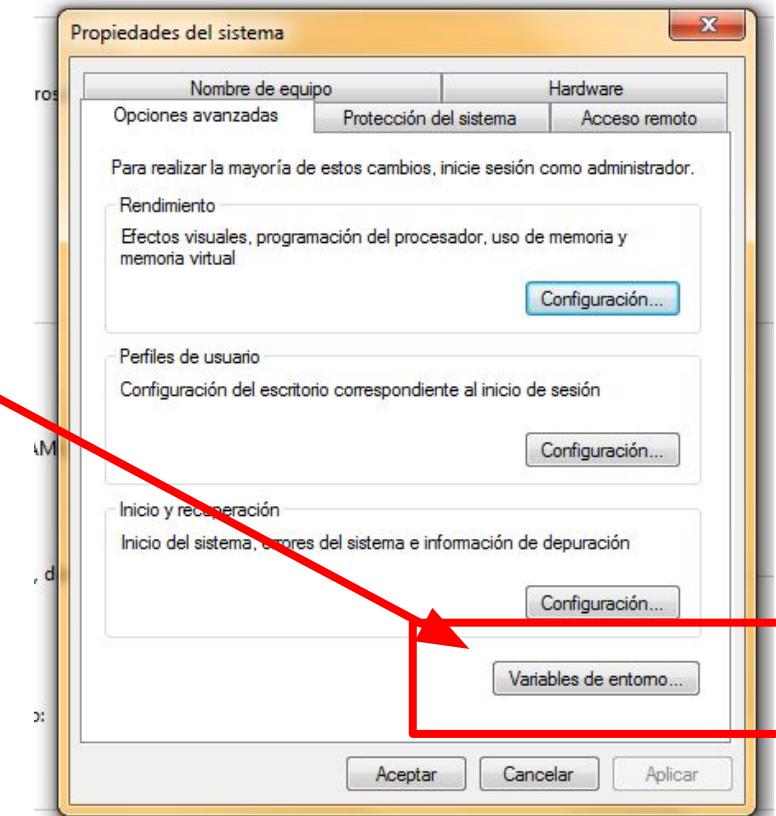
Configuración del CLASSPATH (Windows)



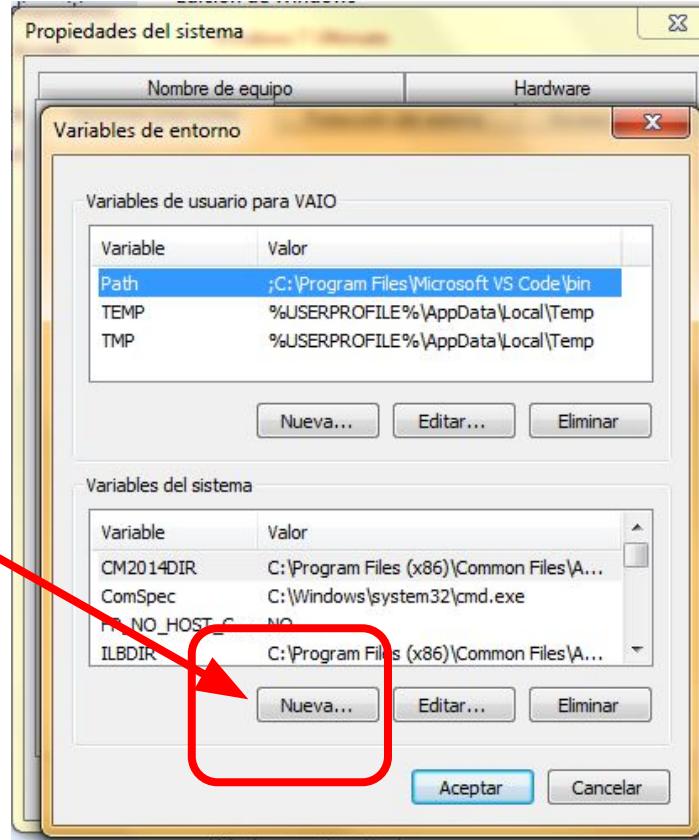
Configuración del PATH (Windows)



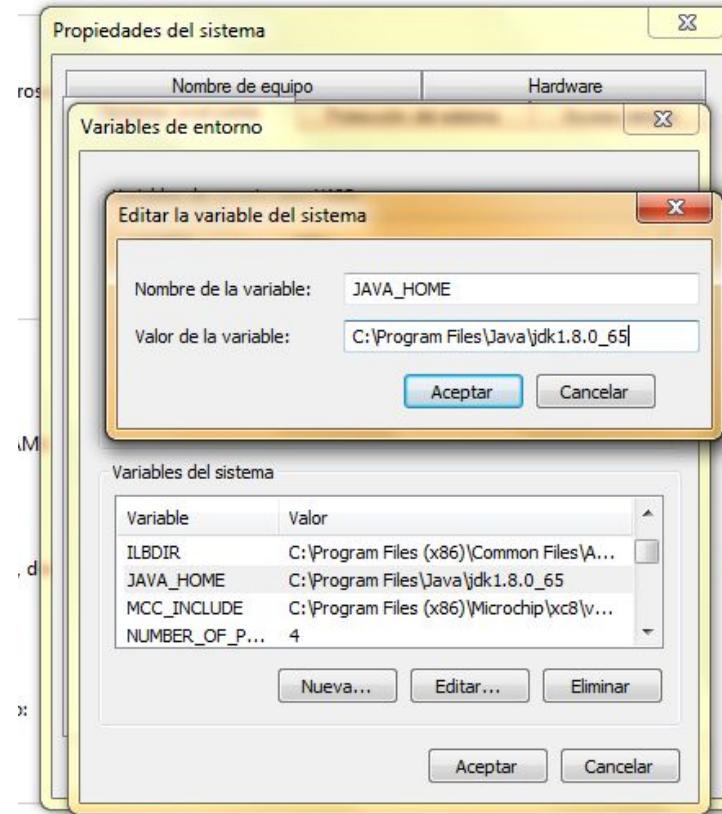
Configuración del PATH (Windows)



Configuración del PATH (Windows)

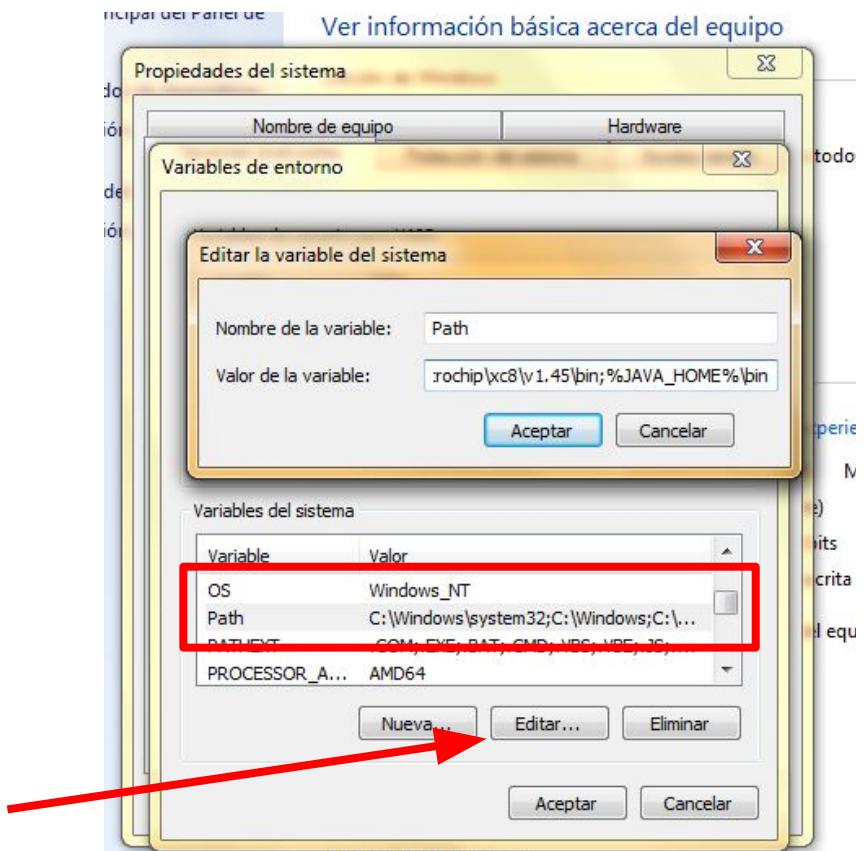


Configuración del PATH (Windows)



JAVA_HOME

Configuración del PATH (Windows)



;%JAVA_HOME%\bin

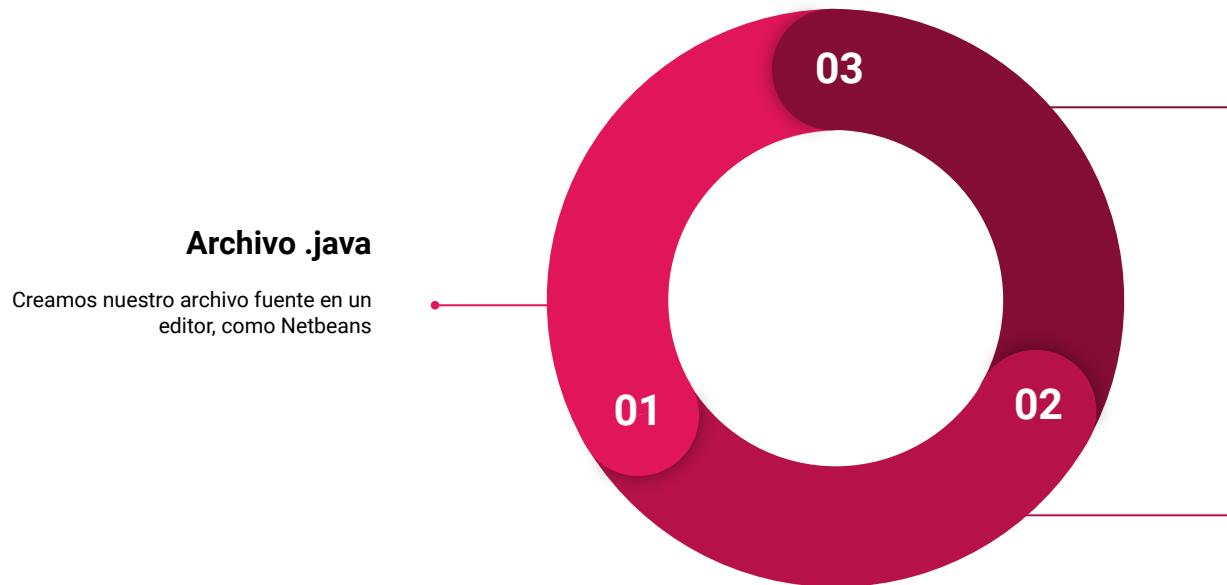
Comprobar que la JVM está configurada

```
java --version
```

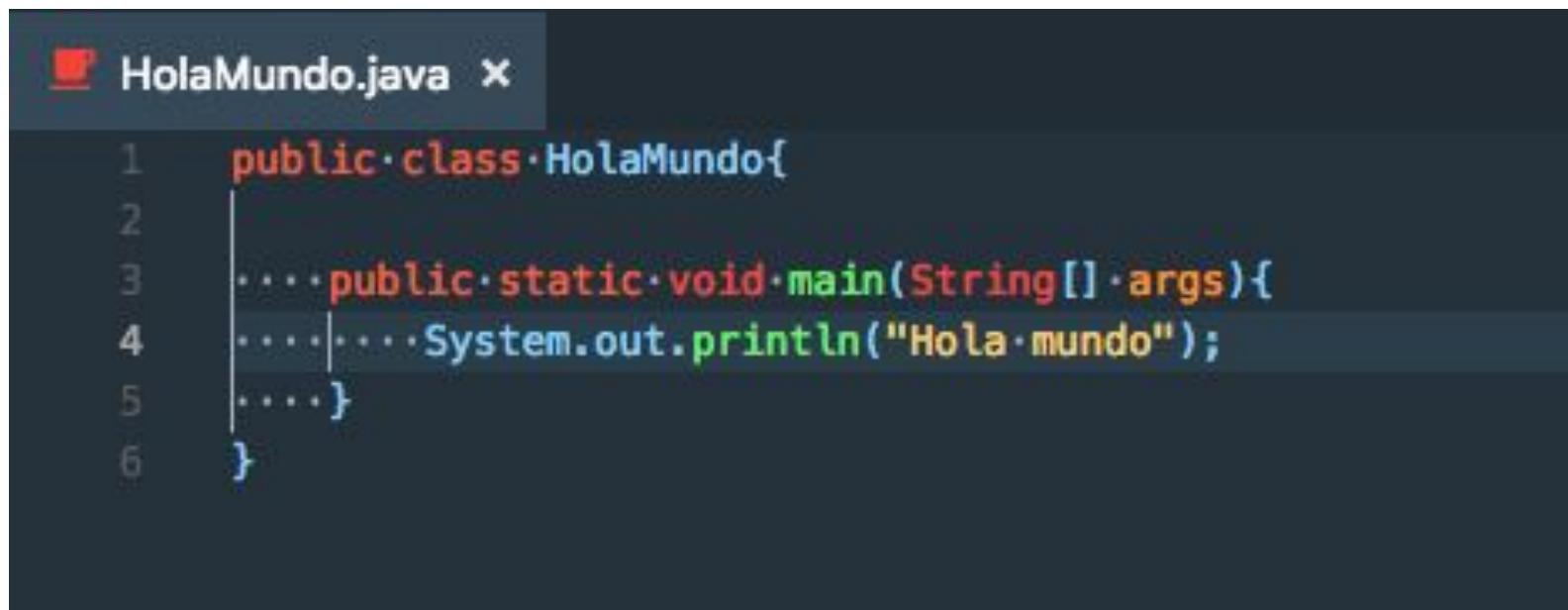
```
→ ~ java --version
java 9.0.4
Java(TM) SE Runtime Environment (build 9.0.4+11)
Java HotSpot(TM) 64-Bit Server VM (build 9.0.4+11, mixed mode)
→ ~
```

```
→ ~ 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”



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

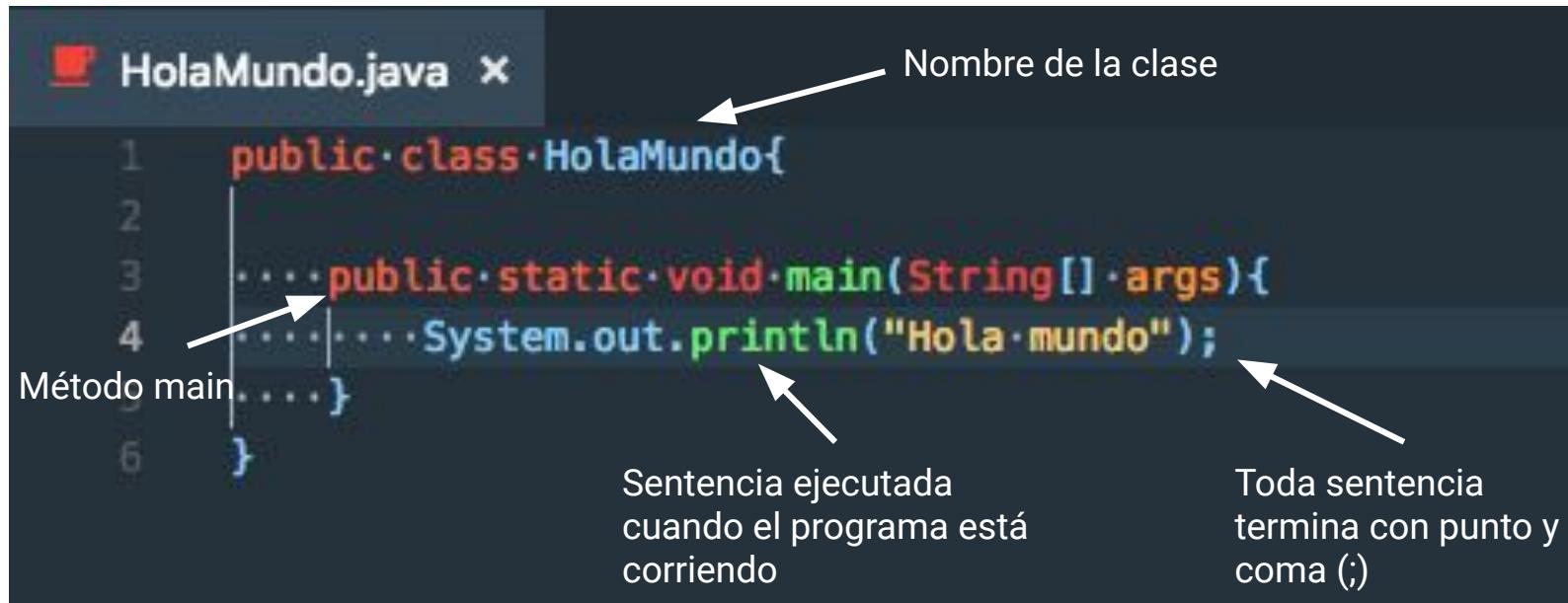
Proceso de compilación en terminal

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



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

Nombre de la clase

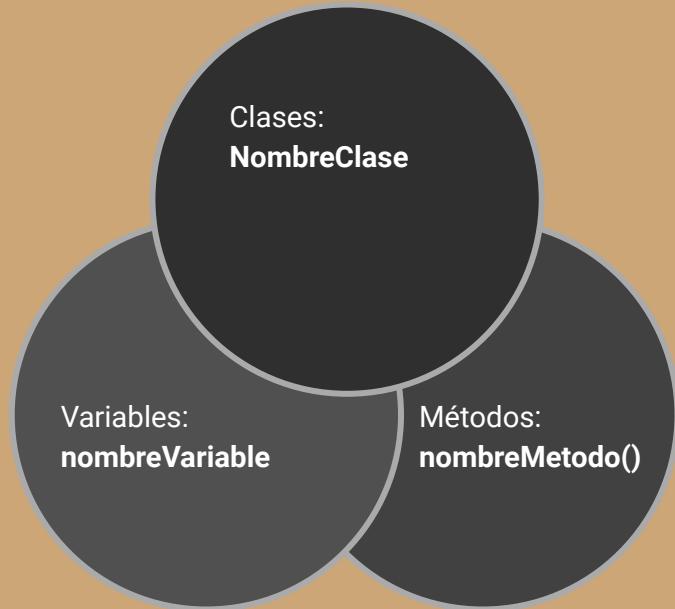
Método main

Sentencia ejecutada cuando el programa está corriendo

Toda sentencia termina con punto y coma (;

Buenas prácticas

Camel Case



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*

Secuencias de Escape

Nombre	Sintaxis
Salto de línea (Enter)	\n
Tabulador	\t
Retorno de carro	\r
Diagonal invertida	\\\
Doble comilla	\"

¿Qué es una variable?

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



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

Declaración de variables

tipoDeVariable *nombreVariable*; //declaración de variable

tipoDeVariable *nombreVariable* = *valorAsignado*;

int *miVariableEntera* = 5;

double *miVariableDoble* = 4.32;

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

Operadores Aritméticos combinados

Nombre	Símbolo	Descripción
Suma	$+=$	$x = x + 3 \rightarrow x+=3$
Resta	$-=$	$x = x - 3 \rightarrow x-=3$
Multiplicación	$*=$	$x = x * 3 \rightarrow x*=3$
División	$/=$	$x = x / 3 \rightarrow x/=3$
Residuo	$\%=$	$x = x \% 3 \rightarrow x\%=3$
Incremento	$++$	$x = x + 1 \rightarrow x++$
Decremento	$--$	$x = x - 1 \rightarrow x--$

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 .

$$F = m a$$



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	Carácter	%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 éste 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	<code>&&</code>	<code>(5 == 5) && (4==4)</code>	True
OR	<code> </code>	<code>(9 > 3) false</code>	True
NOT	<code>!</code>	<code>!false</code>	True

Ejercicio - Aprobado-Reprobado con mensaje

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

- Si obtuvo menos de 6 -> *“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 falto 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
```

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, 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

```
}
```

Sentencia de decisión SWITCH

```
switch(variable){  
    case opcion1:  
        //código  
        break;  
    case opcion2:  
        //código  
        break;  
    default:  
        //código  
}  
}
```

Diagrama que explica la estructura del switch en C. Los comentarios y las instrucciones break están en azul. Los demás textos y las flechas están en negro.

- La flecha apunta a "variable" con la etiqueta "Opción a comparar".
- Las flechas apuntan a "opcion1" y "opcion2" con la etiqueta "Caso".
- La flecha apunta a "default:" con la etiqueta "Rompe el switch".
- La flecha apunta a "código" con la etiqueta "Ejecuta ésta sección si no se cumple ningún caso".

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.



Sentencias de control - FOR

Separado por punto y coma (;

```
for( inicio; condicion ; Δ ){
```

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

}

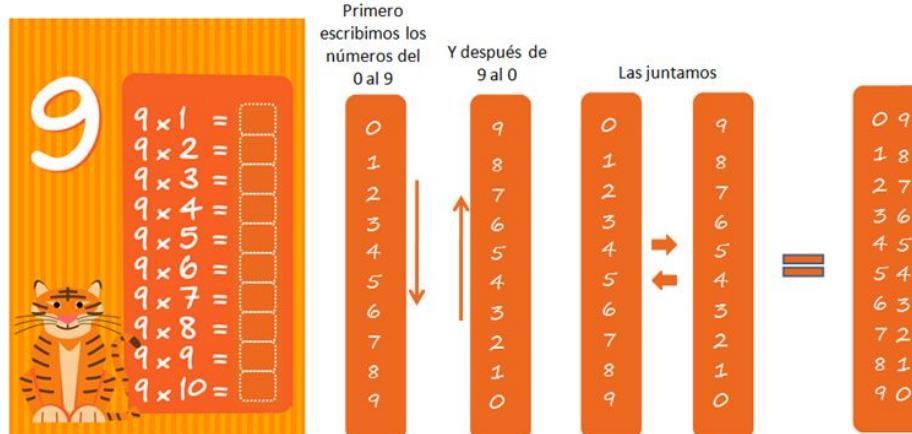
```
for( inicio ; tope ; incremento/decremento ){
```

//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.



TablaFor.java, TablaDinamica.java

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 los números de un conjunto, e ir pidiendo uno a uno, al final dar el resultado de la media, para salir debe dar el valor de -1.

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

Arreglos (array)

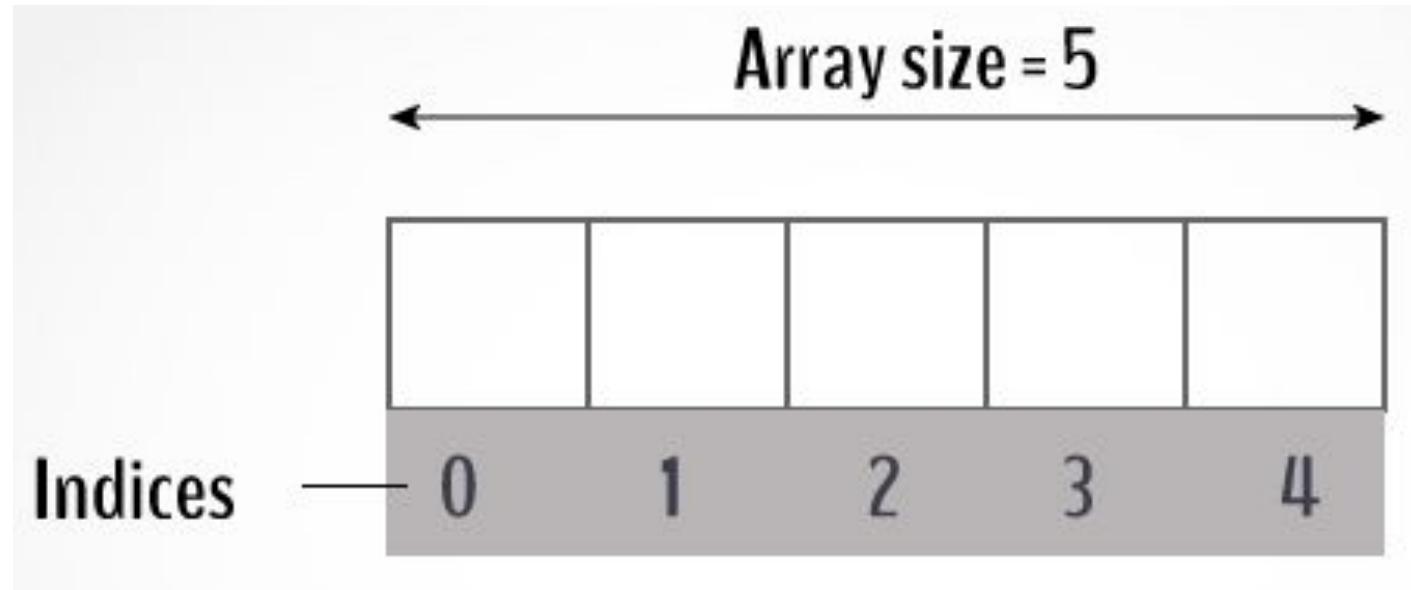
Es una estructura de datos, una colección de elementos, en éste caso es una colección de referencias.

Características:

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



Arreglos (array)



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.

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

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

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*

Array $n \times m$

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

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

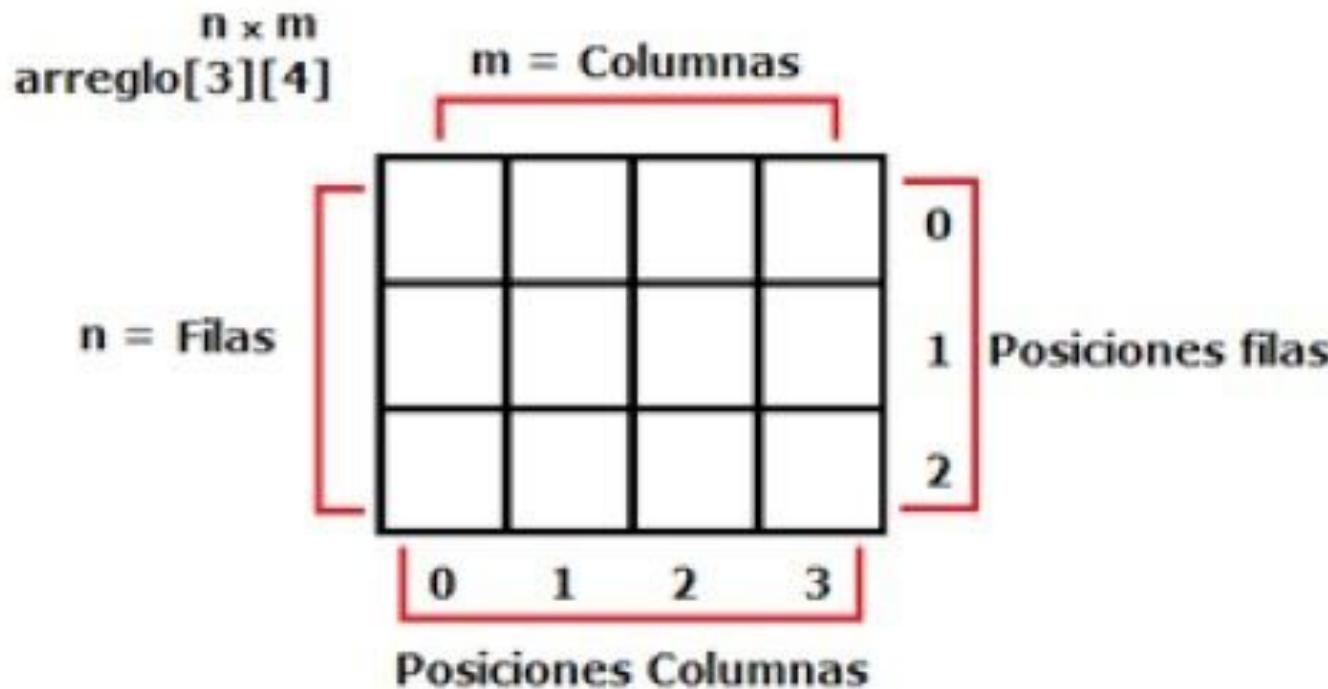
```
int segundoArreglo[] = {
```

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

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

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

Array $n \times m$



Array $n \times m$

- 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 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)

}

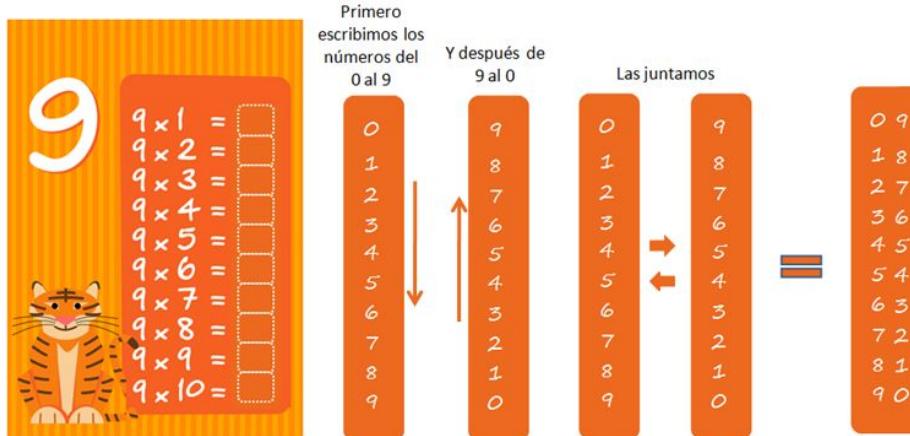
do{

 //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.



Entorno de Desarrollo Integrado (IDE)



NetBeans

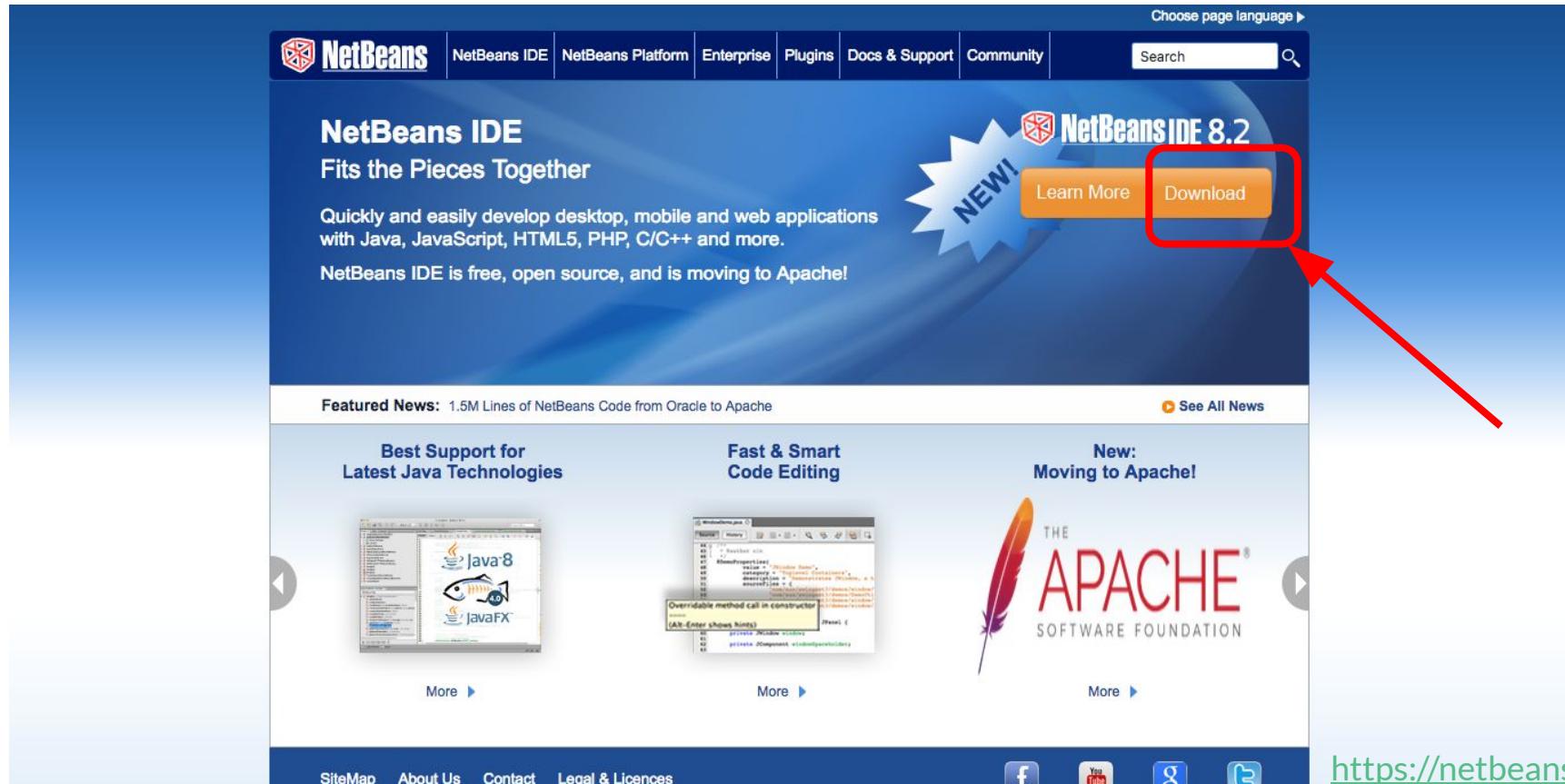


eclipse



IntelliJ IDEA

Netbeans



Choose page language ▾

NetBeans

NetBeans IDE NetBeans Platform Enterprise Plugins Docs & Support Community

Search

NetBeans IDE 8.2

NEW!

Learn More Download

Featured News: 1.5M Lines of NetBeans Code from Oracle to Apache See All News

Best Support for Latest Java Technologies

Fast & Smart Code Editing

New: Moving to Apache!

THE APACHE SOFTWARE FOUNDATION

More ▶

More ▶

More ▶

SiteMap About Us Contact Legal & Licences

<https://netbeans.org/>

NetBeans IDE 8.2 Download

8.1 | 8.2 | Development | Archive

Email address (optional):

Subscribe to newsletters: Monthly Weekly

NetBeans can contact me at this address

IDE Language: ▼

Platform: ▼

Note: Greyed out technologies are not supported for this platform.

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	●	●				●
Java SE	●	●				●
Java FX	●	●				●
Java EE		●				●
Java ME						●
HTML5/JavaScript		●	●	●		●
PHP			●	●		●
C/C++					●	●
Groovy						●
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		●				●
Apache Tomcat 8.0.27	●					●

Download **Download** **Download x86** **Download x86** **Download x86** **Download**
Download x64 **Download x64** **Download x64** **Download x64**

Free, 94 MB Free, 196 MB Free, 116 - 119 MB Free, 116 - 119 MB Free, 115 - 117 MB Free, 214 MB

* You can add or remove packs later using the IDE's Plugin Manager (Tools | Plugins).

HTML/JS, PHP and C/C++ NetBeans bundles include Java Runtime Environment and do not require a separate Java installation.

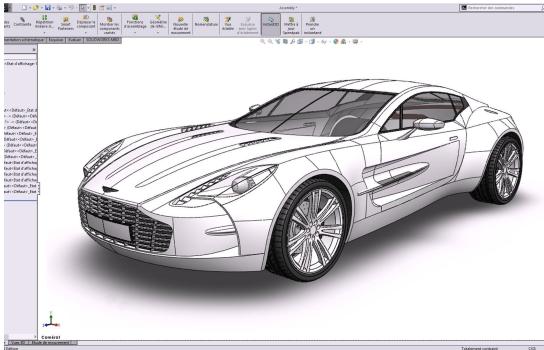
JDK 8 is required for installing and running the Java SE, Java EE and All NetBeans Bundles. NetBeans 8.2 does not run on JDK9! You can download standalone JDK or download the latest JDK with NetBeans IDE Java SE bundle.

Important Legal Information:

NetBeans Community Distributions are available under a Dual License consisting of the Common Development and Distribution License (CDDL) v1.0 and GNU General Public License (GPL) v2. Such distributions include additional components under separate licenses identified in the License file. See the [Third Party License](#) file for external

4. Programación Orientada a Objetos

Objetos



Clase



Objeto



Atributos y
Comportamientos

Objetos



¿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
```

Atributo = Campo

```
    int campo1;
```

Método = Comportamientos

```
    public void miMetodo( ){
```

//cuerpo del método

```
}
```

//fin de la clase

Campos y Métodos

public **int** **noPuertas**; //campo de tipo entero

Nivel de acceso Tipo Nombre Parentesis

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

System.out.println("acelerando");

}

Cuerpo del método
(Lo que hará el método)

Tipos de métodos

Nombre	Descripción
void	No devuelve ningún dato, sólo 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 Objeto

Comportamientos:

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



Creación de una nueva instancia de un Objeto

```
TipoObjeto nombreInstancia = new TipoObjeto( );
```

Debe terminar con paréntesis



Nos indica la creación de un nuevo objeto

Llamada métodos y campos

`<nombreInstancia> . <método>`

`<nombreInstancia> . <campo>`

Niveles de acceso (Encapsulamiento)

Nombre	Clase	Package	Subclase	Todos
public	Sí	Sí	Sí	Sí
protected	Sí	Sí	Sí	No
No especificado	Sí	Sí	No	No
private	Sí	No	No	No

Creando Objeto

Atributos:

- noPuertas: + int
- kilometraje: + long

Comportamientos:

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



Crear objeto - Persona

Atributos:

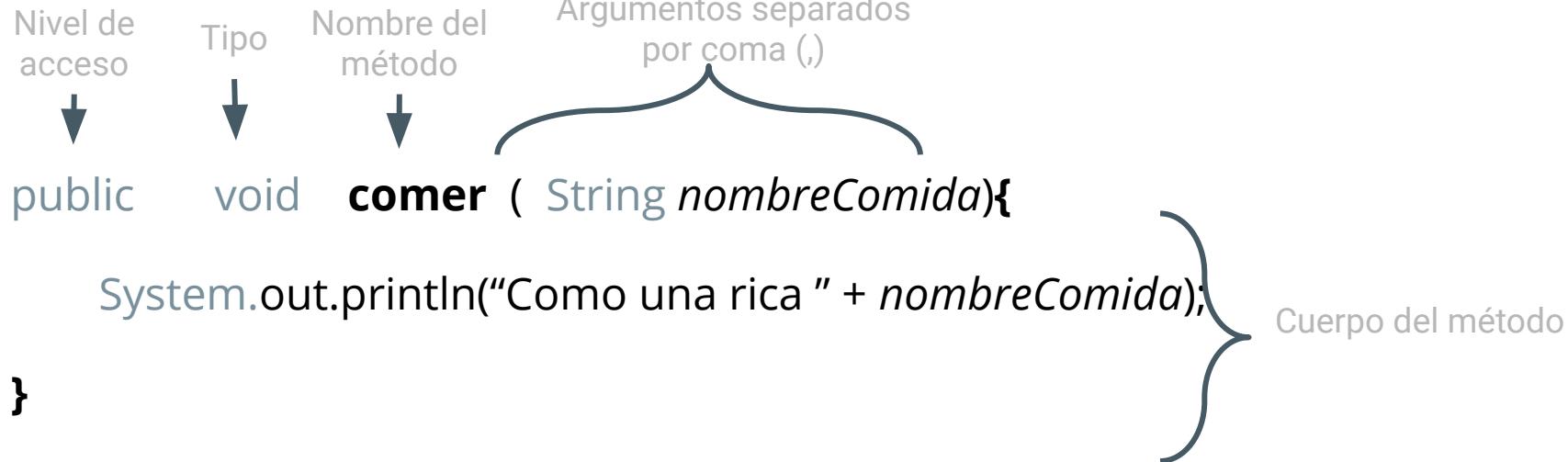
- nombre: + String
- edad: + int

Métodos:

- saludar(): + void
- decirEdad(): +void



Métodos con *argumentos*



Crear objeto - Persona

Atributos:

- nombre: + String
- edad: + int

Métodos:

- saludar(): + void
- decirEdad(): +void
- *calcularEdadActual(int anio): +void*
- *cenar(String comida, String bebida): + void*



Métodos con *return*

Nivel de acceso Tipo Nombre del método

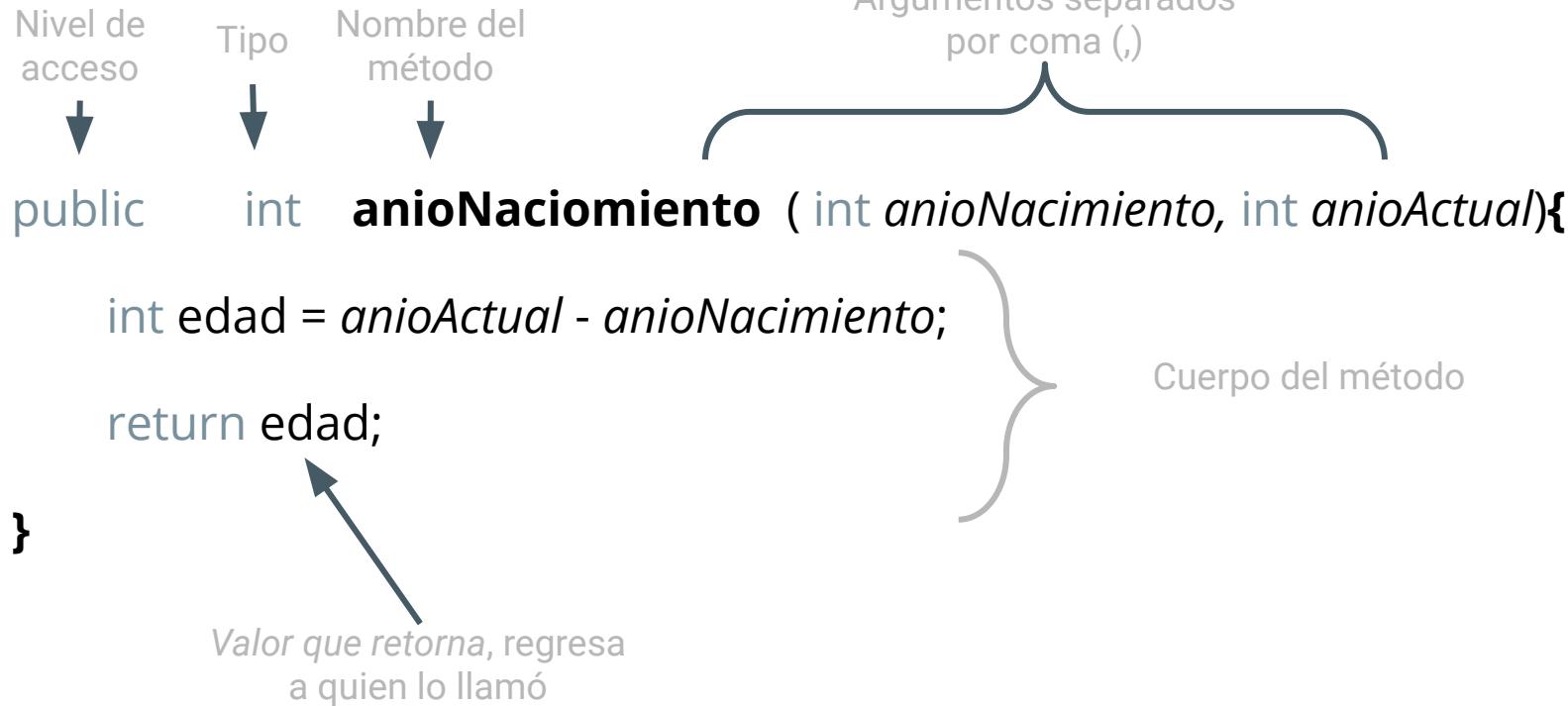
↓ ↓ ↓

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

Argumentos separados por coma (,)

Cuerpo del método

Valor que retorna, regresa a quien lo llamó



Objeto - Perro



Atributos:

Perro.java

- nombre : -String
- raza : -String

Métodos:

- ladear(): + void
- correr(int velocidad):
+ void
- jugar(): + String :
(pelota, hueso,
chancla)

Objeto - Producto (*set, get*)

Campos:

- precio: - double
- nombre: - String
- cantidad : - String

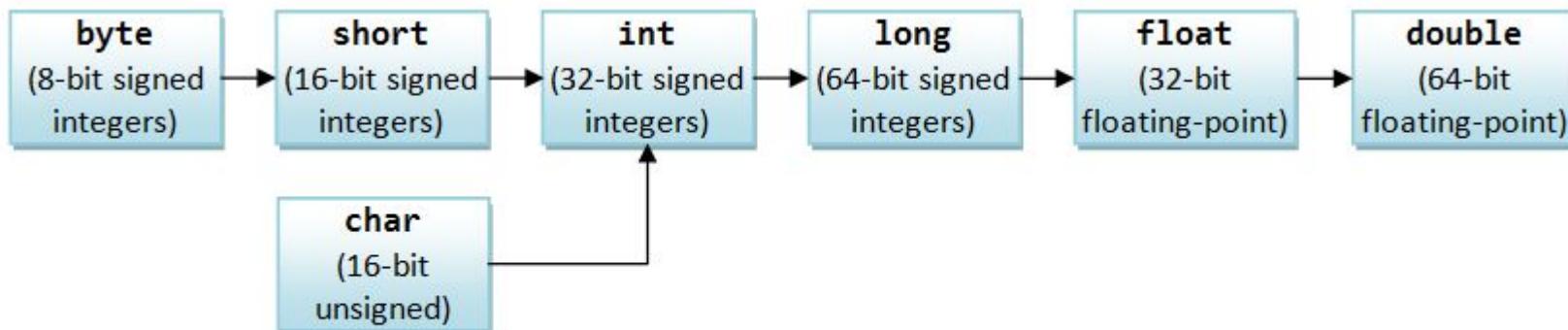
Comportamientos:

- descripcionProducto(): + void

Casting de tipos primitivos

Hacer conversión de tipos de datos primitivos.

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



Orders of Implicit Type-Casting for Primitives

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  
        }  
    }  
}
```

this

Hace referencia a si mismo.

```
String nombre;  
public void setNombre(String nombre){  
    this.nombre = nombre;  
}
```

Objeto - Auto (*this, scope*)

Atributos:

- nombre: - String
- Color: - String
- encendido: -boolean

Comportamientos

- arrancar: + void
- apagar: + void
- descripcion(): +void

Plain Old Java Object (POJO)



Una clase para un objeto básico, con sus métodos de asignación (set) y obtención (get), que describe algún tipo de estructura o dato genérico.

Alumno.java

Sobrecarga de métodos

```
public double setAceleracion(){
```

```
}
```

```
public double setAceleracion(int tiempo){
```

```
}
```

```
public double setAceleracion(double tiempo){
```

```
}
```

```
public double setAceleracion(double tiempo, boolean turbo){
```

```
}
```

Constructores

```
public class MiObjeto{
```

Nivel de
acceso



Mismo nombre de la clase



```
public MiObjeto( ){ //constructor por default
```

//contenido del constructor

```
}
```

- Es el primero que se manda a llamar al crear una instancia
- Debe tener el mismo nombre que la clase

```
}
```

Constructores con argumentos

```
public class MiObjeto{  
    public MiObjeto(){ //constructor vacío  
    }  
    public MiObjeto(tipo argumento){ //constructor con argumento  
    }  
}
```

Constructores (this)

Con la palabra reservada this se puede hacer la llamar a otro constructor. Esto se realiza cuando se desea construir el objeto con datos por default desde el constructor vacío.

```
public MyObjeto(){  
    this("Sin nombre");  
}  
  
public MyObjeto(String nombre){  
    this.nombre = nombre;  
}
```

Metodos estáticos

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

//cuerpo del método

```
}
```

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

//cuerpo del método

```
}
```

- Se requiere que se genere una **instancia**
- No se crea uno nuevo aunque se genere una instancia

Llamada métodos y campos *static*

<Clase> . <método>

<Clase> . <campo>

Objeto - Ley de Ohm

Crear una librería para la solución de Ley de Ohm.



Librería Math

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

SPHERE

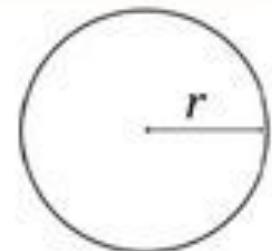
$$S = 4\pi r^2$$
$$V = \frac{4\pi r^3}{3}$$



CIRCLE

$$P = 2\pi r$$

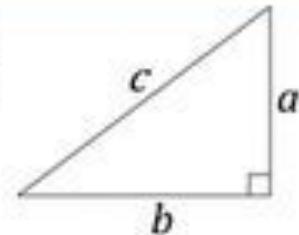
$$A = \pi r^2$$



PYTHAGOREAN THEOREM

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

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



Campos *final*

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

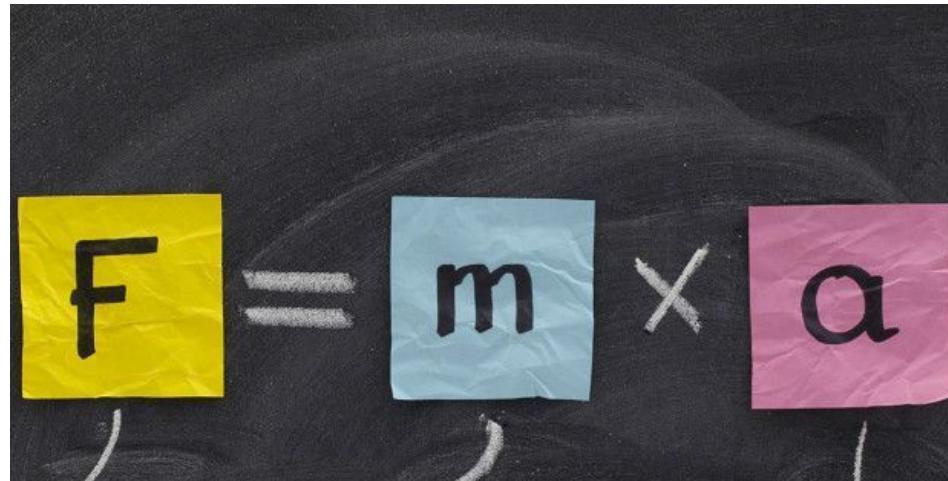
La forma de declararlo es en Mayúsculas y separados por guión bajo (_)

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

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

Objeto - Segunda Ley de Newton

Crear un objeto de la segunda Ley de Newton, teniendo como campo static y final la gravedad.



Clases envolturas (Wrapper)

Las clases envolturas son la equivalencia de los tipos primitivos en Objetos

- Integer
- Double
- Float
- Character
- Boolean

Métodos static de Wrappers

`Double.parseDouble("98.56");`

`Double.toString(43.54);`

`Integer.parseInt(34);`

`Integer.toString("5");`

Null (*comparación, argumento*)

Objeto `null` hace referencia a que el objeto *no tiene ninguna referencia*. Es decir, sólo tiene asignado un espacio en memoria listo para usarse pero aún no contiene nada.

Objeto myObjeto = `null`; -> Objeto myObjeto;

`if(objeto == null){`

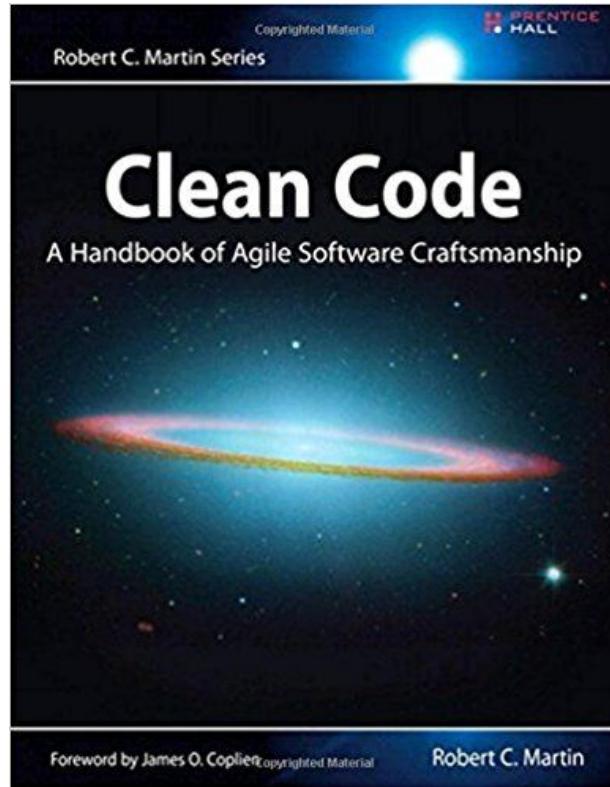
//significa que no existe el objeto, no está definido o no contiene ninguna referencia

`}`

Cuando se instancia una clase, todos los objetos se inicializan con null, igual a los primitivos que se inicializan con 0.

Buenas prácticas en métodos

- Debe realizar **una sola acción**.
- Debe recibir máximo 3 argumentos, si no es posible puede llegar hasta 5, no más.
- El nombre del método debe describir la acción que realiza.
- No debe tener más de 15 líneas de código.
- Un array es el último argumento que debe recibir el método.



5. POO Avanzado

Herencia

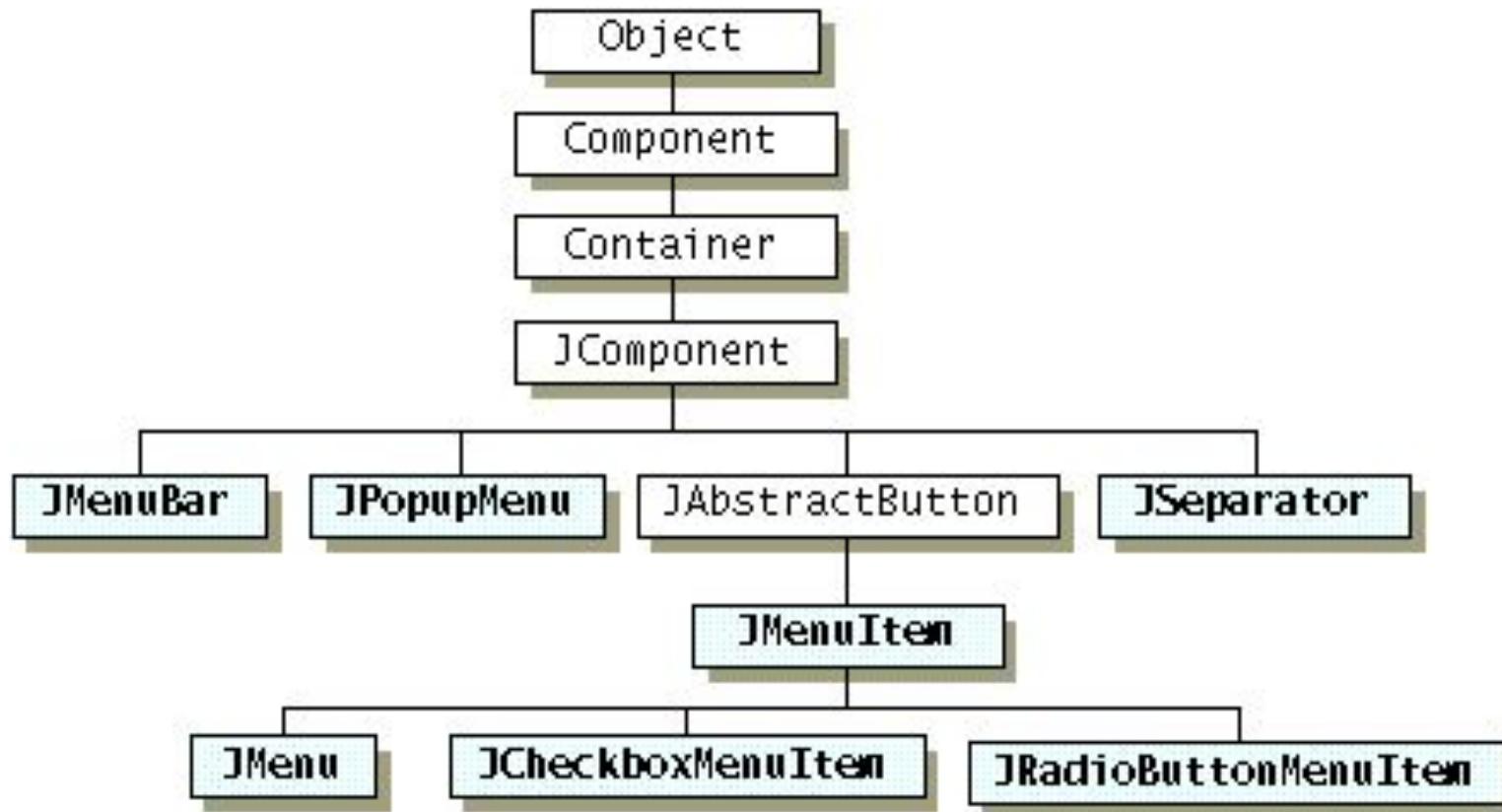
Es una propiedad que existe en la POO, en la cual se puede crear una clase general y conforme se va *heredando* se va haciendo especializada en una o más clases.

Clase Padre --> Clase hija (*heredada*)

Super clase --> Subclase

Al hacer la herencia, todos los campos y métodos son pasados a la clase hija.

Herencia



Extends

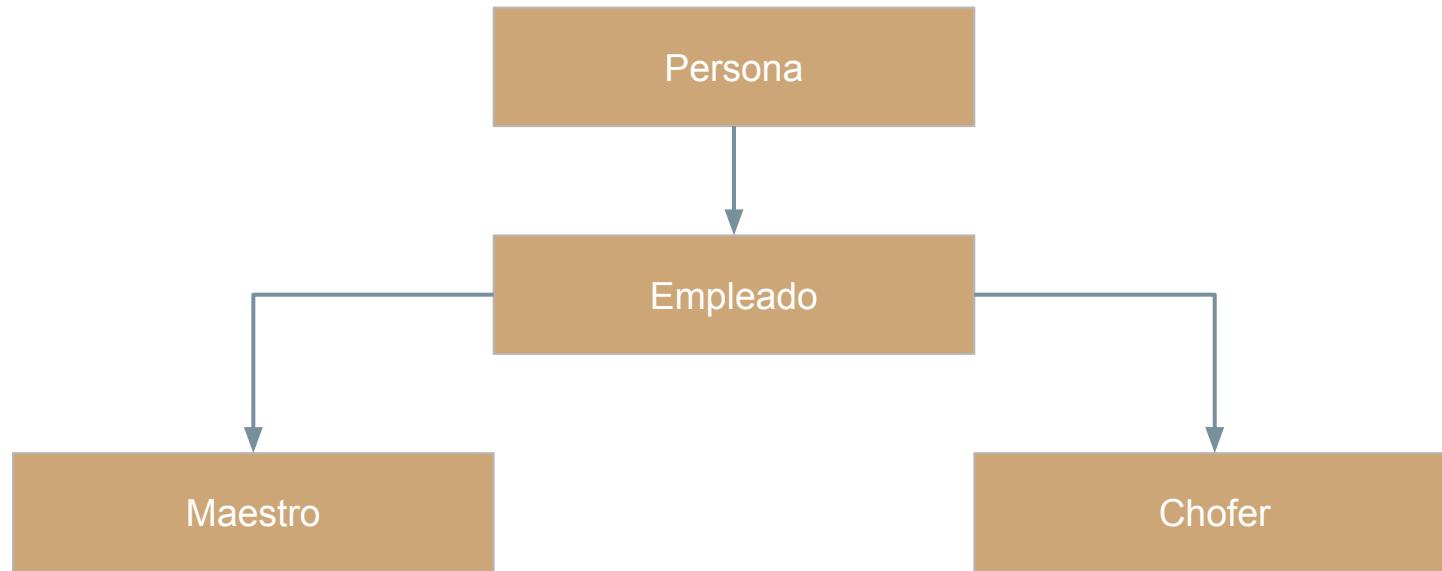
La forma de generar la herencia es usando la palabra reservada *extends*.

```
public class ClaseHija extends ClasePadre{
```

//cuerpo de la clase

```
}
```

Extends



Clase Object

Toda clase al ser creada por default está heredado de Object.

La clase object tiene los métodos:

- `clone();`
- `toString();`
- `finalize();`
- `getClass();`
- `equals();`

Sobre escritura de métodos (Override)

Dentro de la herencia se pueden modificar los comportamientos; es decir, las acción que realiza un método o adicionar acciones diferentes en la subclase. Ésto se conoce como sobre escritura de métodos.

Se identifica con una *anotación* (@) y la palabra override (sobre escritura).

@Override

```
public String toString( ){
```

//cuerpo del método

```
}
```

toString()

Cuando se manda a imprimir un objeto, lo que realmente se manda a llamar es el método **toString()** que contiene dicho objeto.

Es una buena práctica siempre sobre escribir el método **toString()** en todas las clases que creemos.

```
System.out.println(myObjeto); <=> System.out.println(myObjeto.toString());
```

super

Usar la palabra reservada **super** es muy similar a usar la palabra **this**, la diferencia que **super** hace referencia a la clase padre y **this** a la clase actual. De esta manera podemos acceder a los métodos y campos de *clase padre siempre y cuando sean publics o protected*.

@Override

```
public String toString( ){  
    return super.toString() + "contenido de la clase actual";  
}
```

Sobrecarga de constructores por herencia (super)

Sobrecarga de constructores por herencia se realiza siempre y cuando exista un constructor con parámetros en su clase padre.

```
public class Aguila extends Ave{  
    public Aguila(){  
        super(); //Llama al constructor de Ave por default  
    }  
}
```

Sobrecarga de constructores por herencia (super)

Si existe un constructor con parámetros forzosamente debe ser implementado.

```
public class Aguila extends Ave{  
    private String color;  
    public Aguila(String tipo, String color){  
        super(tipo); //Llama al constructor de Ave por default  
        this.color = color;  
    }  
}
```

Final class

Al poner final a la clase lo que genera es que **no se pueda hacer herencia de ésta clase**; es decir, *no puede tener clases hijas o subclases*.

```
public final class MyClase{  
    //cuerpo de la clase  
}
```

Final Method

Al poner final un método ***no podrá sobre escribirse en clases hijas***. Esto se hace cuando no se quiere permitir que modifiquen las acciones del método aplicando herencia.

```
public final void miMetodo{  
    //cuerpo del método  
}
```

6. Manejo de Errores

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 éste no tienen (*null*).

Manejo de Excepciones

```
try{
```

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

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

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

```
}
```

Multiple manejo de Excepciones

```
try{  
    //ejecución de código normal  
}catch(Exception nombre){  
    //si sucede un error y es del tipo de Exception  
}catch(ExceptionN nombreN){  
    //si sucede un error y es del tipo de Exception  
}
```

Manejo de Excepciones (*finally*)

```
try{  
    //ejecución de código normal  
}catch(Exception){  
    //si sucede un error, se trata el error aquí  
}  
finally{  
    // se ejecuta exista o no un error  
}
```

7. Colecciones

List, ArrayList

Es una colección ordena 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

```
}
```

Map, *HashMap*

Map es una interfaz.

Los objetos Maps asocian claves a valores.

No pueden contener llaves duplicadas.

La clase más utilizada es *HashMap*.



8. Interfaz Gráfica

Elementos de Interfaz - Ventana



Elementos de Interfaz

Botones(Button)

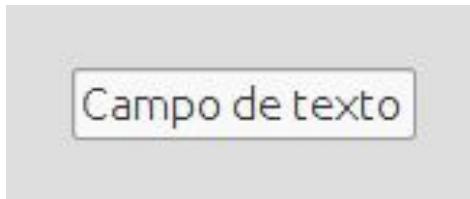


Etiquetas (Label)

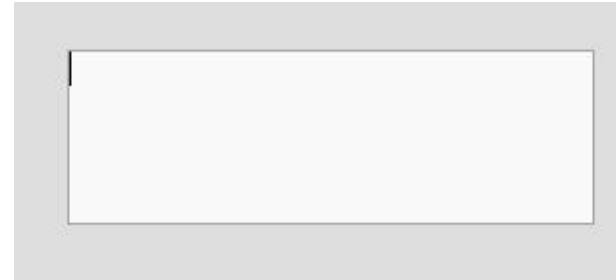


Elementos de Interfaz

Campo de texto



Área de texto



Elementos de Interfaz

Botón tipo radio

RadioButton



Caja de chequeo

CheckBox



Eventos

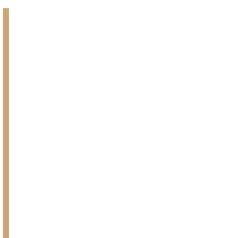
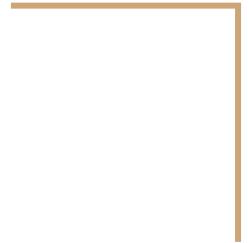
Un evento es un método que se dispara cuando sucede un cambio en un elemento.

Eventos:

- *Click* (Action)
- *Mouse*
 - Entrar el mouse
 - Sale el mouse
- *Teclado*
 - Teclas
- ...



Apps UI



Apps

Generador de Números aleatorios, con la opción de límite.

Generador de contraseña segura.

9. Bases de Datos

Base de Datos

SQLite



Base de datos

CRUD

Crear (Create)

Leer (Read)

Actualizar (Update)

Borrar (Delete)

Lenguaje de Consulta (SQL)

CREATE

CREATE TABLE [name_table]
([columns_name] [data_type],...);

CREATE TABLE employee (_id
INTEGER, name **TEXT**, age
INTEGER);

Lenguaje de Consulta (SQL)

INSERT

INSERT INTO [name_table]
VALUES(name_columns);

INSERT INTO employee **VALUES**(
4, "Juan", 28);

Lenguaje de Consulta (SQL)

READ

SELECT [columns] **FROM** table;

SELECT name, age **FROM**
employee;

Lenguaje de Consulta (SQL)

UPDATE

UPDATE name_table **SET**
column_name,... **WHERE**
condition;

UPDATE employee **SET**
name = "Juan Carlos", age = 29
WHERE id = 2 ;

Lenguaje de Consulta (SQL)

DELETE

DELETE FROM name_table
WHERE condition;

DELETE FROM employee **WHERE**
id = 2;

Documentación

<http://www.sqlitetutorial.net/>

<https://www.sqlite.org/>

<https://github.com/xerial/sqlite-jdbc>



10. Aplicación Java con DB

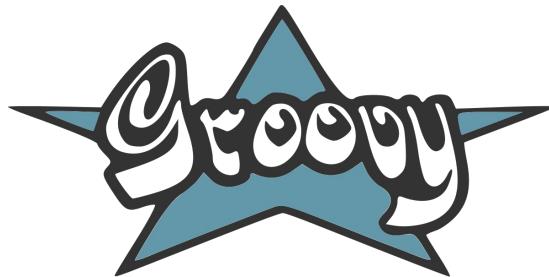




Imagen de la app

Extras

Lenguajes que has aprendido



Clojure

Herramientas que operan con JVM



maven



spark

El curso que ha tomado cuesta

Java SE7 Fundamentals

Itinerario / Compra	Formatos Para La Capacitación <small>?</small>	Precio	Duration	Materiales del Curso	Language
▶ Ver detalles	Training On Demand	\$ 15045	5 días	Español	Español
▶ Ver Itinerario	Live Virtual Class	\$ 16375	5 días	Inglés	Inglés
▶ Ver Itinerario	Classroom Training	\$ 17700	5 días	Inglés	Español
▶ Ver detalles	Capacitación de Autoestudio en CD-ROM	\$ 10658	0 días	Inglés	Inglés

http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=609&get_params=dc:D67234,clang:EN#tabs-3

Contacto

Ing. Alejandro Leyva

<http://www.alejandro-leyva.com>

contacto@alejandro-leyva.com

<https://www.linkedin.com/in/alejandro-leyva-consult/>

<https://www.facebook.com/leyva.consult/>

