

Java

Introducción a Java

Objetivos

- Explicar las **características principales** del lenguaje de programación JAVA.
- **Ejecutar códigos de ejemplo.**
- Explicar qué es la **API de JAVA e investigar su documentación.**
- **Importar bibliotecas Java API en un proyecto JAVA.**
- **Compilar y ejecutar programas JAVA desde la terminal.**



¿Qué es JAVA?

Java es un **lenguaje de programación basado en clases y orientado a objetos**, diseñado para tener la menor cantidad posible de dependencias de implementación.

Las aplicaciones Java se ejecutan en una **Máquina Virtual Java** que permite la portabilidad a diferentes plataformas.



Lenguaje de alto nivel vs Lenguaje de bajo nivel

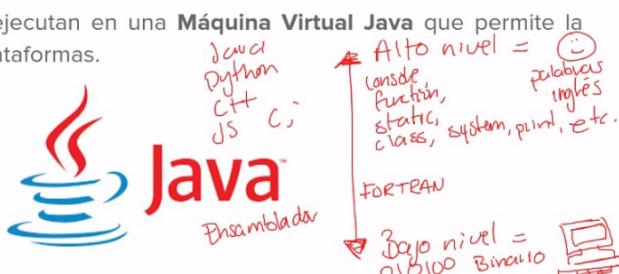
- Es un lenguaje que parece el lenguaje natural con expresiones como console, function, class, static

- Es más tirando a Obj (como lenguaje ensamblador)

¿Qué es JAVA?

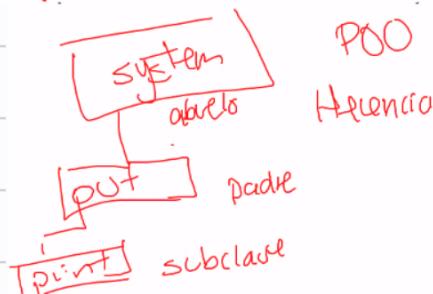
Java es un **lenguaje de programación basado en clases y orientado a objetos**, diseñado para tener la menor cantidad posible de dependencias de implementación.

Las aplicaciones Java se ejecutan en una **Máquina Virtual Java** que permite la portabilidad a diferentes plataformas.



Con Java vamos a tener que ser bastante específicos

System.out.print →
public static void



Java: Creada en 1995 por James Gosling por una empresa llamada Sun - microsystem

La versión más utilizada de Java es la V8 o la V17

Existen 2 versiones SE, standard Edition
EE Enterprise edition
mismas bases distintas funciones/capacidades
mejor para iniciar en el desarrollo web

Creando para evitar el problema de tener que programar solo una vez ya que tienes una máquina virtual

- Comprada por Oracle en 2009, esto impacta en el número de versiones ya que sacan una cada 6 meses

¿Por qué aprender JAVA?

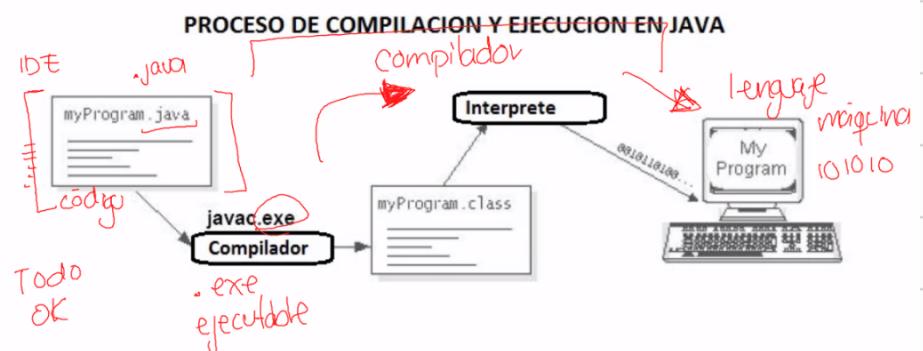
- Es uno de los **lenguajes de programación más populares**.
- Es un lenguaje de programación muy **demandado** por empleadores, y por tanto, **bien remunerado**.
- **JAVA** está en todas partes.
- Es **utilizado en aplicaciones del mundo real**.
- Cuenta con un **conjunto de herramientas sólidas** que ayudan a la productividad.

se puede utilizar
JAVA para ciencia de
datos ML y mucho
más

Java es un
lenguaje compilado
Si no está bien todo no
hace nada

- termina es un lenguaje
interpretado.

JVM = Java Virtual Machine Proceso de compilación en JAVA

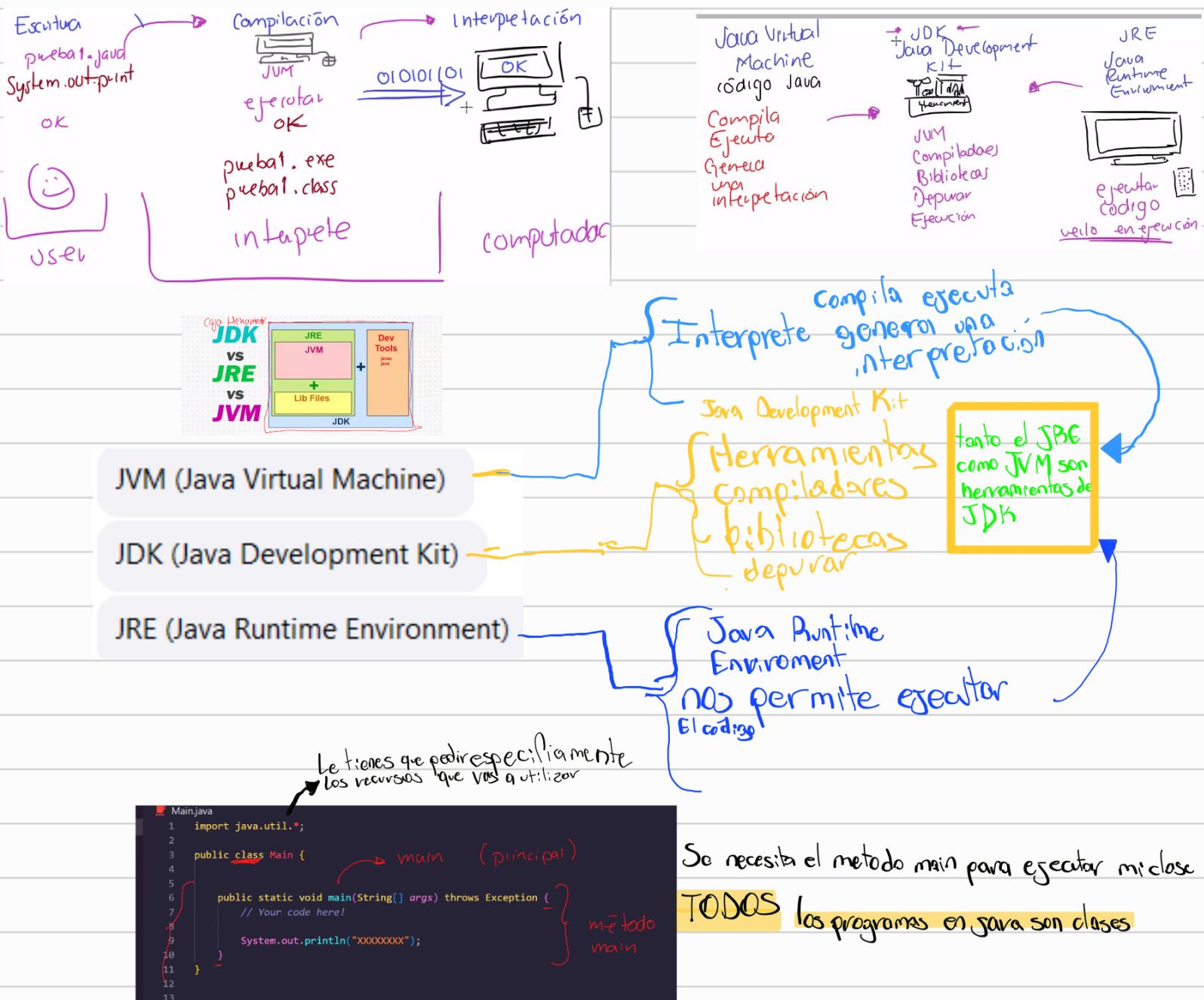


(binario)
pero también utiliza un intérprete para traducirlo a 1-0

IDE

- Netbeans
- IntelliJ
- Eclipse

editor texto → escritura
+ compilador → construcción
+ intérprete → ejecución



Estructura basica de un programa de JAVA

- TODOS los programas de Java, son una clase.
- Al hablar de clases, puedo tener objetos
- Para poder ejecutar mi clase, necesito un metodo main (predefinido)
- El nombre de la clase tiene que ser identico al nombre de mi archivo
- Es similar a JS, solo que con otro lenguaje (sintaxis)
- En ocasiones, sera necesario importar bibliotecas o funcionalidades

Documentación

<https://docs.oracle.com/en/java/javase/17/docs/api/>

```

package Restaurante;
public class Usuario {
    private String username;
    private String password;
    private String rol;
}

public Usuario(String username, String password, String rol) {
    this.username = username;
    this.password = password;
    this.rol = rol;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

```

class Persona

① - Atributos

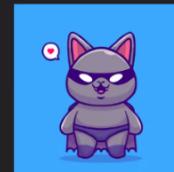
③ - Constructor
this (Juancho)

Método
imprimirInfo

Ejemplo de un código en JAVA

Variables en Java

- Lenguaje de programación estático.
- Lenguaje de programación de tipado fuerte.



O diferencia de JS que tiene tipado dinámico y una variable le podemos asignar cualquier otro tipo de dato

Variables en Java

```

•••
→ String name = "Mariana";
→ int age = 28;
→ double salary = 52500.50;
→ boolean lead;
lead = true;

```

En cambio Java si declaras una variable del tipo string tienes que asignarle el mismo tipo de dato

Tipado dinámico: puedes reasignarle un valor de otro tipo de dato

Tipado Fuerte: para declarar una variable debes de iniciar declarando el tipo

Variables en Java

el punto y coma es obligatorio siempre

```

•••
long product_id = 0152354;

```

Datos primitivos

- Enteros:** byte, short, int y long. Permiten trabajar con números enteros ya sean positivos o negativos.
- Flotantes/Decimales:** float y double. Permiten trabajar con números los cuales posean punto decimal ya sean positivos o negativos.
- Caracteres:** char. Este tipo de datos nos permitirá trabajar con caracteres (ASCII).
- Booleanos:** Permite trabajar con valores lógicos: verdadero o falso.

Enteros

Datos primitivos: Enteros

Nombre	bytes	Valor Mínimo	Valor Máximo
byte	1	-128	127
short	2	-32768	32767
int	4	-2147483648	2147483647
long	8	-922337203685477575808	9223372036854775807

Flotantes decimales

Datos primitivos: Flotantes/Decimales

Nombre	bytes	Valor Mínimo	Valor Máximo
float (f)	4	1.4 e-45	3.4 e+38
double	8	4.9 e-324	1.8 e+308

Datos primitivos: Caracteres

Código estándar americano para el intercambio de información ASCII - Table
<https://www.asciitable.com/>

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr			
0	0 000	NUL (null)			32	20 040	 	Space		64	40 100	@	§	96	60 140	`	¤
1	1 001	SOH (start of heading)			33	21 041	!	!	!	65	41 101	A	A	97	61 141	a	a
2	2 002	STX (start of text)			34	22 042	"	"	"	66	42 102	B	B	98	62 142	b	b
3	3 003	ETX (end of text)			35	23 043	#	#	#	67	43 103	C	C	99	63 143	c	c
4	4 004	ETB (end of transmission)			36	24 044	$	\$	\$	68	44 104	D	D	100	64 144	d	d
5	5 005	ENQ (enquiry)			37	25 045	%	%	%	69	45 105	E	E	101	65 145	e	e
6	6 006	ACK (acknowledge)			38	26 046	&	&	&	70	46 106	F	F	103	66 146	f	f
7	7 007	BEL (bell)			39	27 047	'	'	'	71	47 107	G	G	105	67 147	g	g
8	8 010	BS (backspace)			40	28 050	(((72	48 110	H	H	108	68 150	h	h
9	9 011	TAB (horizontal tab)			41	29 051)))	73	49 111	I	I	105	69 151	i	i
10	A 012	LF (NL line feed, new line)			42	2A 052	*	*	*	74	4A 112	J	J	106	6A 152	j	j
11	B 013	VT (vertical tab)			43	2B 053	+	+	+	75	4B 113	K	K	107	6B 153	k	k
12	C 014	FF (NP form feed, new page)			44	2C 054	,	,	,	76	4C 114	L	L	108	6C 154	l	l
13	D 015	CR (carriage return)			45	2D 055	-	-	-	77	4D 115	M	M	109	6D 155	m	m
14	E 016	SO (shift out)			46	2E 056	.	.	.	78	4E 116	N	N	110	6E 156	n	n
15	F 017	SI (shift in)			47	2F 057	/	/	/	79	4F 117	O	O	111	6F 157	o	o
16	10 020	DLE (data link escape)			48	30 060	0	0	0	80	50 120	P	P	112	70 160	p	p
17	11 021	DC1 (device control 1)			49	31 061	1	1	1	81	51 121	Q	Q	113	71 161	q	q
18	12 022	DC2 (device control 2)			50	32 062	2	2	2	82	52 122	R	R	114	72 162	r	r
19	13 023	DC3 (device control 3)			51	33 063	3	3	3	83	53 123	S	S	115	73 163	s	s
20	24 024	DC4 (device control 4)			52	34 064	4	4	4	84	54 124	T	T	116	74 164	t	t
21	15 025	NAN (negative acknowledge)			53	35 065	5	5	5	85	55 125	U	U	117	75 165	u	u
22	16 026	SYN (synchronous idle)			54	36 066	6	6	6	86	56 126	V	V	118	76 166	v	v
23	17 027	FTB (end of trans. block)			55	37 067	7	7	7	87	57 127	W	W	119	77 167	w	w
24	18 030	CAN (cancel)			56	38 070	8	8	8	88	58 130	X	X	120	78 170	x	x
25	19 031	EM (end of medium)			57	39 071	9	9	9	89	59 131	Y	Y	121	79 171	y	y
26	1A 032	SUB (substitute)			58	3A 072	:	:	:	90	5A 132	Z	Z	122	7A 172	z	z
27	1B 033	ESC (escape)			59	3B 073	;	;	;	91	5B 133	[[123	7B 173	{	[
28	1C 034	FS (file separator)			60	3C 074	<	<	<	92	5C 134	\	\	124	7C 174	|	\
29	1D 035	GS (group separator)			61	3D 075	=	=	=	93	5D 135]]	125	7D 175	}]
30	1E 036	RS (record separator)			62	3E 076	>	>	>	94	5E 136	^	_	126	7E 176	~	_
31	1F 037	US (unit separator)			63	3F 077	?	?	?	95	5F 137	_	DEL	127	7F 177		DEL

El char se declara con una sola comilla

Casteo

- Convertir directamente un tipo de datos a otro tipo se conoce como **casting**.
- La conversión de tipos sucede cuando se asigna un valor de un tipo de datos primitivo a otro tipo.

✓ Ampliación de conversión (**automáticamente**):

→ byte > short > char > int > long > float > double

✓ Estrechamiento de conversión (**manualmente**):

double > float > long > int > char > short > byte

Tipos de notación

En java se utiliza camelCase para las variables y PascalCase para el nombre de las clases

Método main

```
1 package holaMundo;  
2  
3 public class HolaMundo {  
4     //Escribir el método main  
5     public static void main(String[] args) {  
6         } tipo Puerto  
7     public  
8     static  
9     y no tiene retorno
```

el nombre main es una convención

args: es el nombre de un array de strings

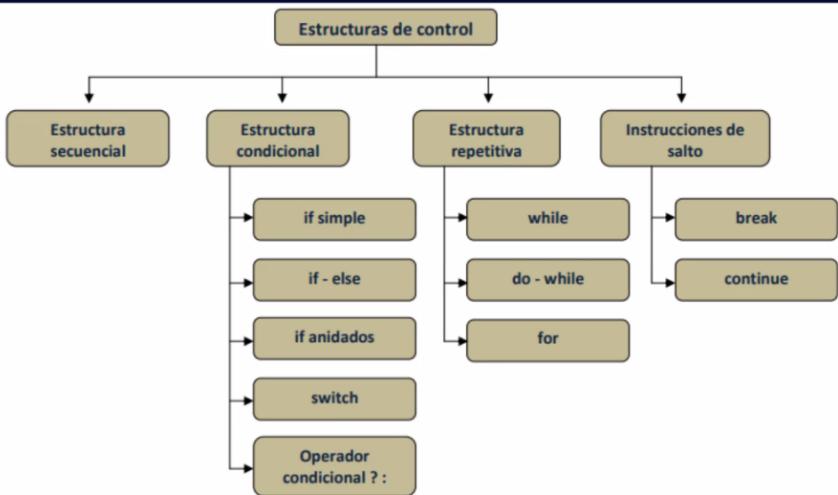
Flujo de control I

#controlDeFlujoJava



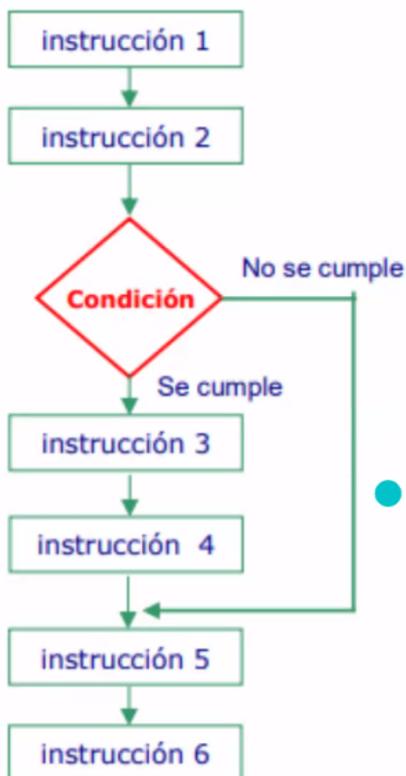
Objetivos de aprendizaje:

- Describir qué es el flujo de control y por qué es importante para un programador.
- Definir y usar declaraciones condicionales.
- Definir y usar la declaración if-else.
- Definir y usar la declaración Switch.
- Definir y usar expresiones condicionales



Las estructuras nos permiten hacer cambios bruscos

Estructuras condicionales



if

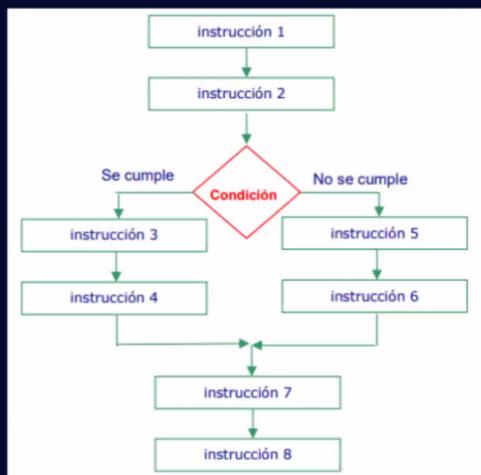
Sintaxis:

```
instrucción 1;  
instrucción 2;  
if(condición){    //inicio de la condición  
    instrucción 3;  
    instrucción 4;  
}  
                                //fin de la condición  
instrucción 5;  
instrucción 6;
```

ko

if else

if - else

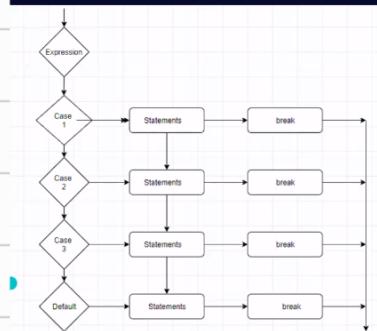


Sintaxis:

```
instrucción 1;  
instrucción 2;  
if(condición){    //instrucciones que se ejecutan  
    // se cumple la condición  
    instrucción 3;  
    instrucción 4;  
} else{      //instrucciones que se ejecutan si,  
    // no se cumple la condición  
}  
instrucción 5;  
instrucción 6;
```

Switch

switch



```
public class EjemploSwitch {  
    public static void main(String[] args) {  
        int opción = 2; // Cambia el valor de 'opción' según tus necesidades  
  
        switch (opción) {  
            case 1:  
                System.out.println("Seleccionaste la opción 1");  
                break;  
  
            case 2:  
                System.out.println("Seleccionaste la opción 2");  
                break;  
  
            case 3:  
                System.out.println("Seleccionaste la opción 3");  
                break;  
  
            default:  
                System.out.println("Opción no válida");  
        }  
    }  
}
```

No admite float ni double puros números enteros

ya que el conjunto de enteros es más limitado que los datos de punto decimal

No puede haber 2 case iguales.

Ciclos

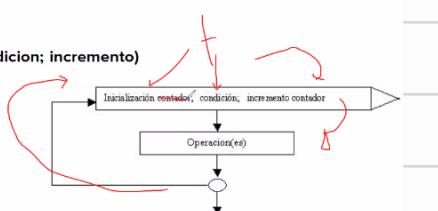
Conceptos básicos

- Los ciclos también son conocidos como bucles (loops).
- Son estructuras de control, también llamadas estructuras repetitivas (hay estructuras repetitivas y secuenciales).
- Es importante identificar cuándo deseamos terminar nuestros ciclos.
- Los ciclos evalúan la condición y cuando se deje de cumplir, ya no ejecutarán más el código.
- Hay que evitar los ciclos infinitos.

For

Ciclo For

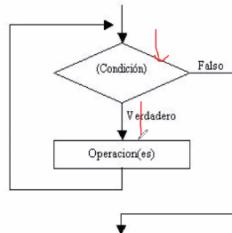
```
for (initialización; condición; incremento)
{
    //instrucciones
}
```



While

Ciclo While

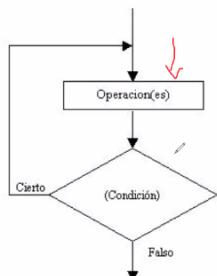
```
while (condición a evaluar)
{
    //instrucciones
}
```



Do while

Ciclo Do-While

```
do {
    //instrucciones
} while (condición)
```



Reposo POO #POO

```
1 package POO;
2
3 public class POO {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8     }
9
10}
11
```

4 pasos para crear clases

- ① Atributos (edad, nombre, apellido)
- ② Constructor (atributos como parámetro)
- ③ Métodos (funciones de un objeto)
- ④ Instancia del objeto

Java = POO

- class → plantilla
- object → elementos
- instance → crear
- method → funcionalidad del objeto
- constructor → constuye
- herencia → pillar POO
- extends → pasar info de una clase a otra

Pasos para crear una clase

2.- Constructor

1. Atributos

```
public class POO {
    //1.- Atributos
    String nombre;
    String apellido;
    byte edad; //el valor de B
    String telefono;
    String email;
```

```
//2.- Constructor
POO(String nombre, String apellido, byte edad, String telefono, String email){}
    //se utilizan la palabra reservada constructor
    this.nombre=nombre;
    this.apellido=apellido;
    this.edad=edad;
    this.telefono=telefono;
    this.email=email;
} //cierre de constructor
```

A diferencia de JS
no utilizaremos la
palabra constructor

3.- Métodos

```
//3.- Métodos
void imprimirInfo() {
    System.out.println("El nombre es: " + nombre);
    System.out.println("El apellido es: " + apellido);
    System.out.println("La edad es: " + edad);
    System.out.println("El telefono es: " + telefono);
    System.out.println("El email es: " + email);
}

void saludar() {
    System.out.println("Hola te mando saludos ");
}
```

4.- Instancia

```
public class Test {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //instancia de persona
        Persona Felipe = new Persona("Felipe", "Maqueda", (byte)31, "332892847887", "Felipe@gmail.com");
        System.out.println(Felipe);
        Felipe.imprimirInfo();
        Felipe.saludar();
    }
} // cierre del metodo main
```

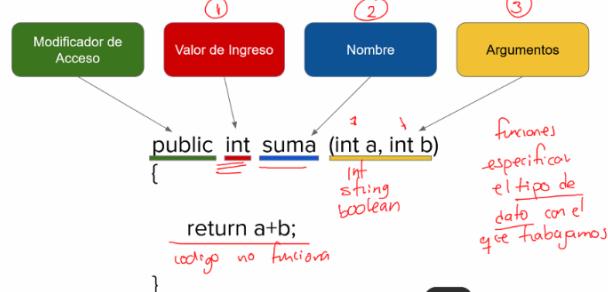
Instancia persona

```
//instancia de un dentista
Dentista simi = new Dentista("Simi", "Lares", (byte)50, "339823659729",
"simi@lares.com", "odontologia", "849894845", "500", "avanzado", "matutino", 500);
simi.calcularSalario();
simi.imprimirInfoDentista();

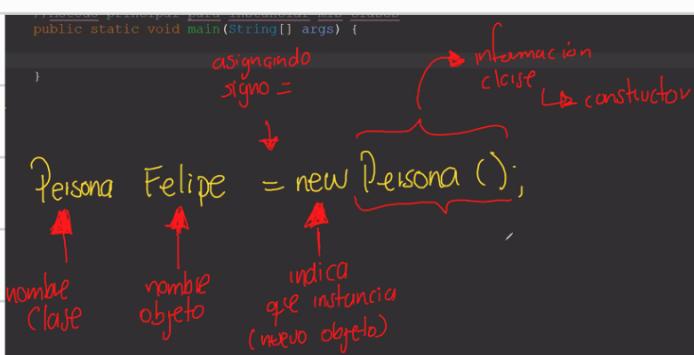
} // cierre del metodo main
```

Instancia dentista

¿Cómo se ve una función en JAVA?



Como instanciar



usando Refactor puedo buscar
y cambiar en todos los lados
donde aparece

Memoria Heap y Memoria Stack (asignación de memoria)

```
void calcularSalario() {
    //Logica de programacion
    if(experiencia.equals("basico")) {
        salario = 30000;
    }else if(experiencia == "intermedio") {
        salario = 40000;
    }else if(experiencia == "avanzado") {
        salario = 50000;
    }
}
```

Cuando utilizo operador de igualdad
evaluo experiencia = experiencia
(referencia de mi variable)

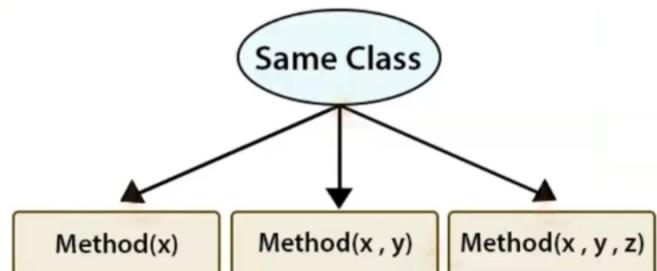
Cuando utilizo el metodo .equals
evaluar el contenido de experiencia =
contenido de experiencia
experiencia.equals

Con el uso del equals puedes
chequear el contenido de la
variable

pregunta de examen:

la sobrecarga de métodos es poder utilizar el mismo nombre de mi constructor/función pero con distintos parámetros

Method Overloading in Java



#programacionOrientadaAOBJETOSAvanzada

Programación orientada a objetos Avanzada

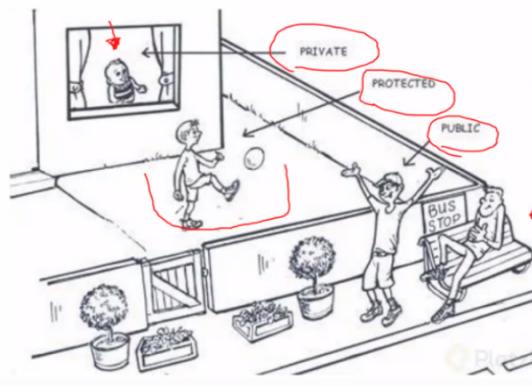
Objetivos

- Usar paquetes JAVA para organizar proyectos
- Explicar los modificadores de acceso JAVA
- Explicar y usar la encapsulación
- Declarar, modificar e iterar colecciones de objetos
- Describir y usar excepciones JAVA
- Usar bloques try/catch

#modificadoresDeAcceso Modificadores de acceso

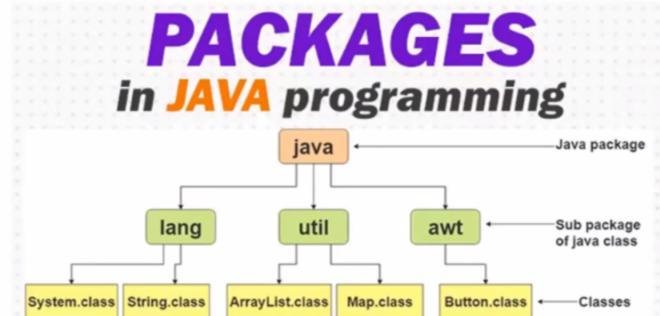
Modificadores de Acceso

Access Levels					
Modifier	Class	Package	Subclass	World	
public	Y	Y	Y	Y	
protected	Y	Y	Y	N	
no modifier	Y	Y	N	N	
private	Y	N	N	N	

El **encapsulamiento** es usado para proteger información

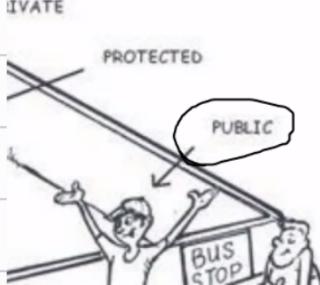
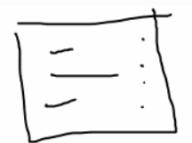
El encapsulamiento necesita del manejo de paquetes

Manejo de paquetes Packages



Si estan en publicos, es facil de modificar

getters y setters
Metodos especiales para acceder y modificar atributos de una clase



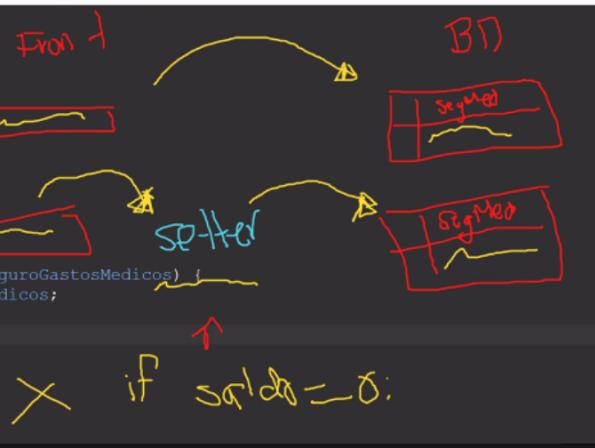
BD

```
} //toString
```

```
//getter | Seguro
public boolean getSeguroGastosMedicos() {
    return seguroGastosMedicos;
}

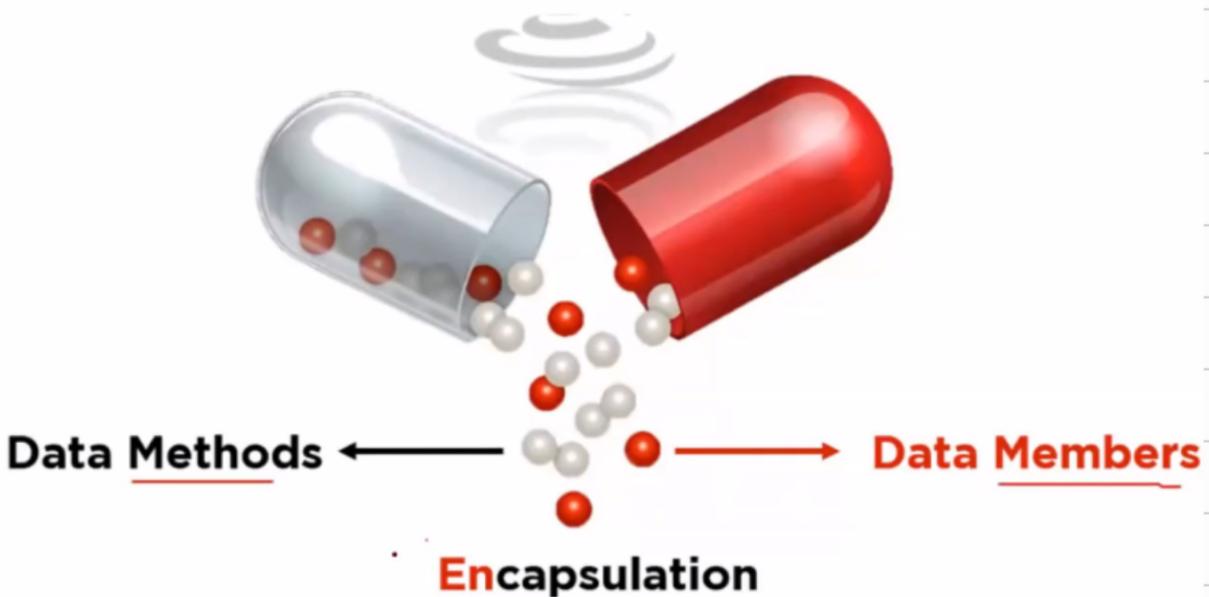
//setter | Seguro
public void setSeguroGastoMedicos(boolean segMedico) {
    this.seguroGastosMedicos = segMedico;
}

//class Paciente
```



El **setter** es como un filtro entre los cambios que puede pedir el usuario. y así no pase a la base de datos

Encapsulamiento



Se usan para acceder a info de datos privados a los cuales se desea acceder (se agregan en la misma clase)

Getters y setters

```
public double getSalario(){  
    return salario;  
}  
  
public String getFechaDeIngreso() {  
    return fechaIngreso;  
}
```

Declaración de los getters

// forma normal de accder a la información con
metodos, pero si queremos acceder al atributo sin
pasar por la función no vamos a poder por lo cual

vamos a necesitar los getters y los setters

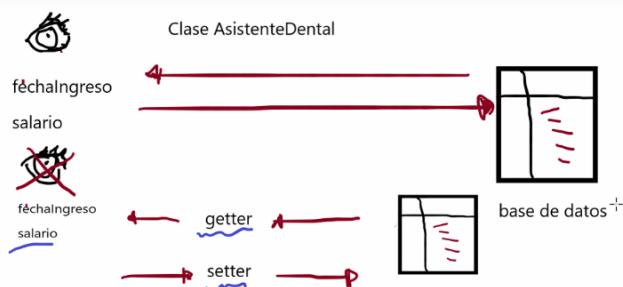
package persona;

```
public class TestAsistenteDental {  
    public static void main(String[] args) {  
        AsistenteDental Francisca= new  
        AsistenteDental("05/10/2023",875.40d);  
        //Impresión de la información de nuestro  
        asistente  
        Francisca.mostrarDatosAsistente();  
    }  
}
```

Accede a la info a través del getter

}

```
public static void main(String[] args) {  
    AsistenteDental Francisca= new AsistenteDental("05/10/2023",875.40d);  
    System.out.println(Francisca.getSalario());  
}  
|
```



Declaración setter

```
public void setSalario(double nuevoSalario) {  
    if(nuevoSalario>0) {  
        salario=nuevoSalario;  
    }  
}
```

Void solo genera cambios
Sin retorno



Conoceremos que es la memoria stack vs heap

Estructura de Datos

Colecciones

Los datos que agrupamos tienen características en común

Sigue habiendo almacen de datos, sigue habiendo estructura de organización y se accede a los datos de mejor forma



Estructuras de datos

Almacenar
Organizar
Acceder a datos de una forma mucho mas rápida

Iterar o hacer operaciones con los datos

Colecciones fueron introducidas a Java en 1998 con el lanzamiento de la versión 1.2 del JDK

Las colecciones se encuentran en el paquete java.util (listas, conjuntos, mapas, arrays, etc.).

Las estructuras de datos son:

- Arreglos (Arrays)
- Listas Enlazadas (Linked Lists)
- Árboles (Trees)
- Grafos (Graphs)
- Conjuntos (Sets)
- Mapas (Maps)

Estructura de datos != Colección

pueden tener algo en común
Pero no son el mismo tipo de dato

tienen características en común lo que almacena

Las estructuras de datos son abstracciones de cómo se almacenan y organizan los datos en la memoria de nuestra computadora, mientras que las colecciones son objetos que permiten almacenar y manipular conjuntos de elementos en un programa.

En cambio, las colecciones son implementaciones concretas de estructuras de datos que proporcionan métodos y operaciones específicas para trabajar con conjuntos de elementos.

las 2 colecciones más comunes :

set: guarda elementos sin repetir

maps: objeto que tiene una "estructura" de llave valor

arrays: esta clase contiene varios métodos para manipular arreglos

link de documentación de arreglos: <https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/package-summary.html>

