TÉCNICAS DE PROGRAMACIÓN ORIENTADA A OBJETOS

26/03/2024 y 28/03/2024

UPN.EDU.PE

LOGRO DE SESIÓN

Al término de la sesión el estudiante analiza los conceptos fundamentales de clases, objetos y encapsulamiento de datos, su representación en un ambiente de desarrollo integrado y características, aplicando su razonamiento lógico en el desarrollo de casos básicas.

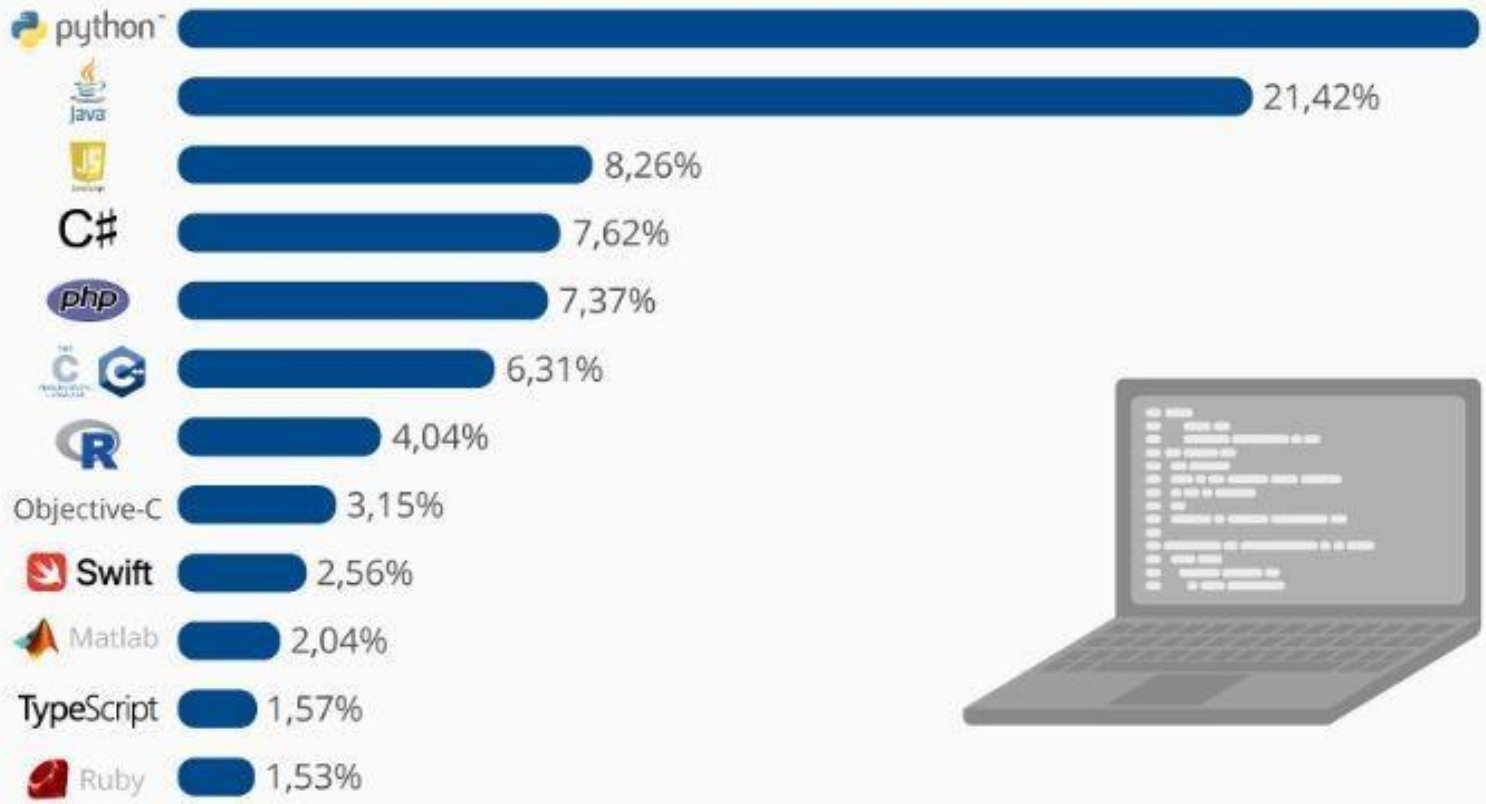
FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS Y RELACIONES BÁSICAS ENTRE CLASES

Sesión 1

- Paradigma Orientado a Objetos.
- Fundamentos de la programación Orientada a Objetos.
- Java como herramienta de programación orientada a objetos.
- Principales sentencias en Java.
- Introducción a la programación orientada a objetos.
- Principios, evolución, abstracción de datos.
- Ocultamiento y encapsulamiento.
- Necesidad de encapsular datos.
- Clases y Objetos.
- Ciclo De Vida.

Los lenguajes de programación más usados

Porcentaje de uso de los lenguajes de programación más populares del mundo*



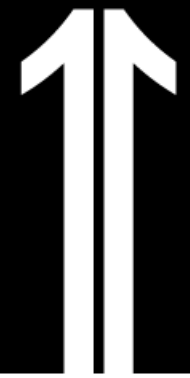
* Datos procedentes del Índice PYPL. Este se basa en las búsquedas en Google de tutoriales de lenguajes de programación.

Datos de enero de 2019

Fuente: PYPL

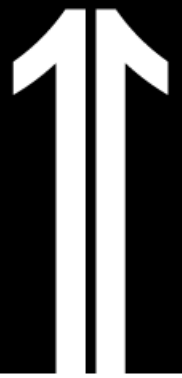
statista

REFLEXIONA



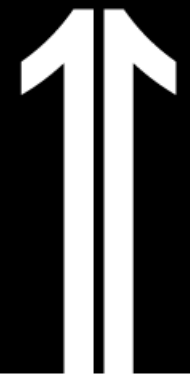
¿Qué es la programación orientada a objetos ?
¿Cuáles son las similitudes y diferencias entre C++ y java?

LOGRO DE SESIÓN



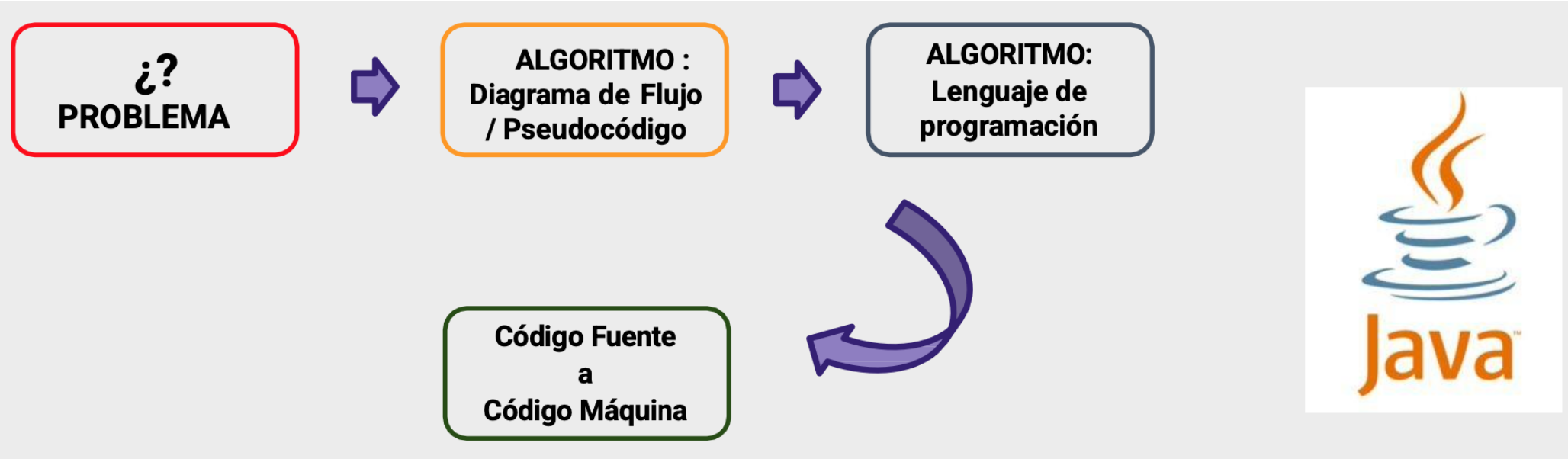
Al término de la sesión el estudiante analiza los conceptos fundamentales de la Programación Orientada a Objetos en Java, su representación en un ambiente de desarrollo integrado y características, aplicando su razonamiento lógico en el desarrollo de casos básicas.

TEMARIO



1. Algoritmos y su representación
2. Introducción al paradigma orientado a objetos.
3. Introducción a Java.
4. Tipos de datos en Java. Clasificación: Primitivas (Simples) y Estructurados.
5. Operadores en Java / Prioridad de operadores
6. Estructuras Básicas de Control
7. Introducción al ambiente de desarrollo: Apache Netbeans.
8. Ejemplo práctico – Estructura Secuencial.

ALGORITMOS Y SU REPRESENTACIÓN



PARADIGMA DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

NECESIDAD DE DOMINAR LA COMPLEJIDAD QUE TIENE LA CONSTRUCCIÓN DE SOFTWARE.

APLICACIONES GRANDES Y DURADERAS EN EL TIEMPO.

APLICACIONES FÁCILES DE MODIFICAR.

LA PROGRAMACIÓN MODULAR SE HACÍA OBSOLETA.

LA POO - DESCOMPOSICIÓN DE OBJETOS.

LA POO SIMULA EL COMPORTAMIENTO DE LOS OBJETOS DE LA VIDA REAL A LA HORA DE PROGRAMAR.

INTRODUCCIÓN A LAS CLASES DE OBJETOS:

CARACTERÍSTICAS	FUNCIONALIDADES
• Sistema Operativo Android • Banda 4G • Pantalla de 5” • Doble cámara	• Llamada a otros celulares • Captura de fotos • Captura Videos • Reproduce música • Accede a redes sociales

En programación orientada a objetos, el teléfono móvil sería un objeto de la clase teléfono, las propiedades serían las características y los métodos serían las funcionalidades que tiene el dispositivo móvil.

CLASE

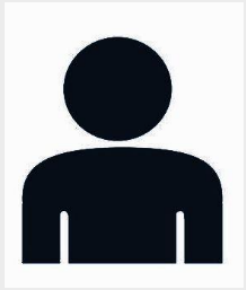
OBJETO

ATRIBUTOS / PROPIEDADES

MÉTODOS

INTRODUCCIÓN A LAS CLASES Y OBJETOS EN JAVA

ES UNA PLANTILLA



PERSONA (CLASE)

TIENE ATRIBUTOS Y MÉTODOS

POSEE UN NOMBRE

UN OBJETO ES UNA INSTANCIA DE UNA CLASE

MARIAN (OBJETO)

MAXIMO (OBJETO)

INTRODUCCIÓN A JAVA

MUY BREVE HISTORIA

1991: Un grupo de ingenieros de Sun Microsystems (liderados por James Gosling y Patrick Naughton) debían desarrollar un lenguaje de programación que se pudiera utilizar en pequeños electrodomésticos. Lo llamaron java.

1991 -1994 se trató de vender la tecnología java SIN EXITO. El proyecto de Gosling y Naughton quedó detenido.

Durante 1994 Internet se expandió y decidieron que java se ajustaba a esta nueva Ola.

Sun

ORACLE

INTRODUCCIÓN A JAVA

CARACTERÍSTICAS:

- SENCILLO: NO en el sentido de aprender el lenguaje, sino de que se quitó características engorrosas de otros lenguajes (aritmética de punteros, encabezados,etc).
- ORIENTADO A OBJETOS
- DISTRIBUIDOS: Buen tratamiento a la hora de programación en red (internet).

INTRODUCCIÓN A JAVA

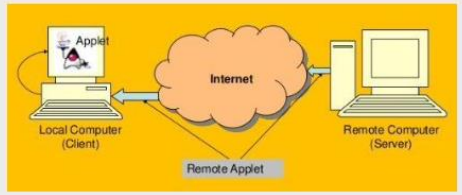
CARACTERÍSTICAS:

- **SEGURO**: Como se creó pensando en un trabajo en red, se hizo seguro.
  NEUTRO: Con respecto a la arquitectura (multiplataforma).
- **INTERPRETADO** : Se compila , y luego interpreta.
- **ADAPTABLE**: Tipo de datos primitivos iguales en todas las plataformas.

¿QUÉ PUEDO PROGRAMAR EN JAVA?

Aplicaciones de Java de Consola

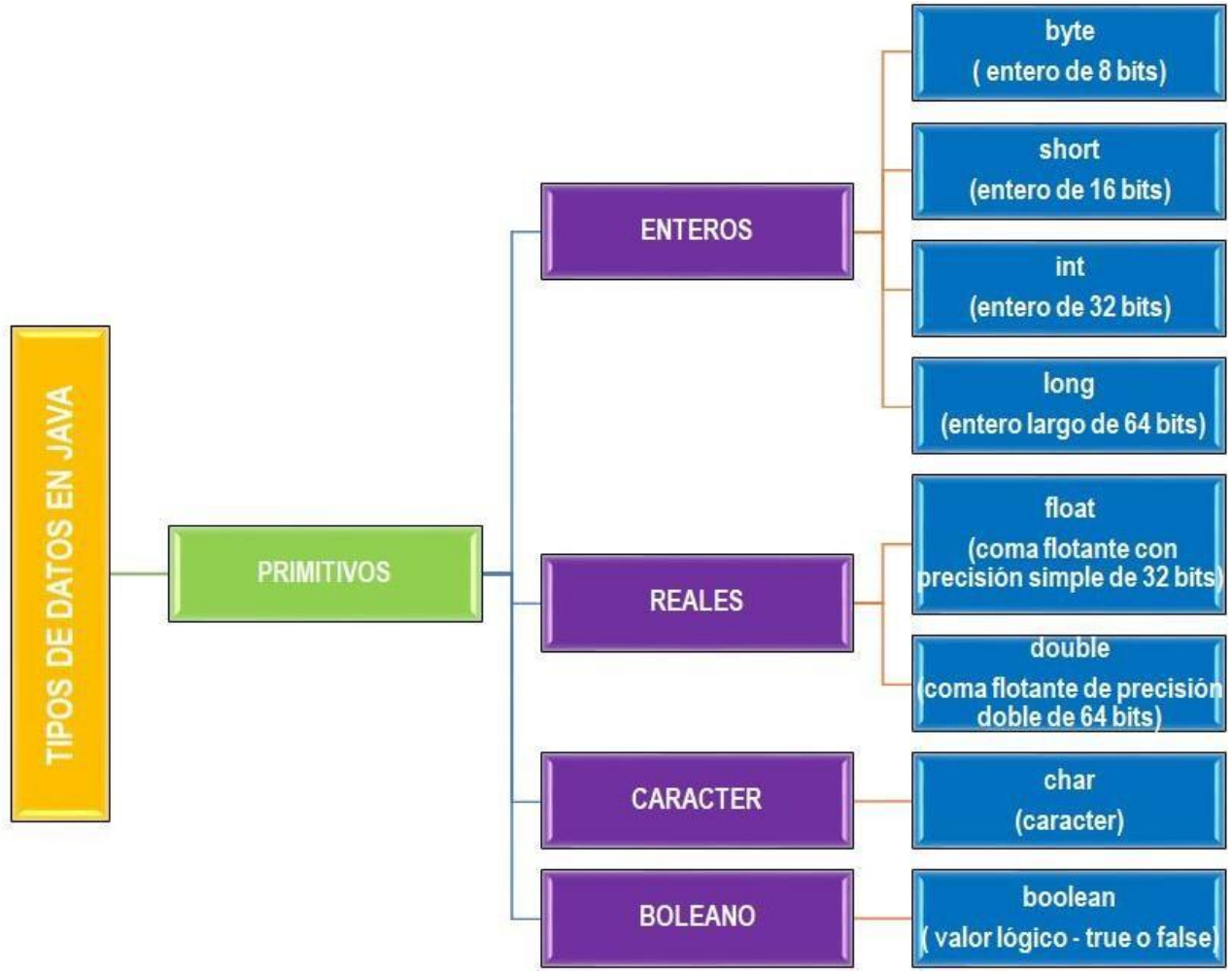
Aplicaciones de Java Applets



Aplicaciones GUI

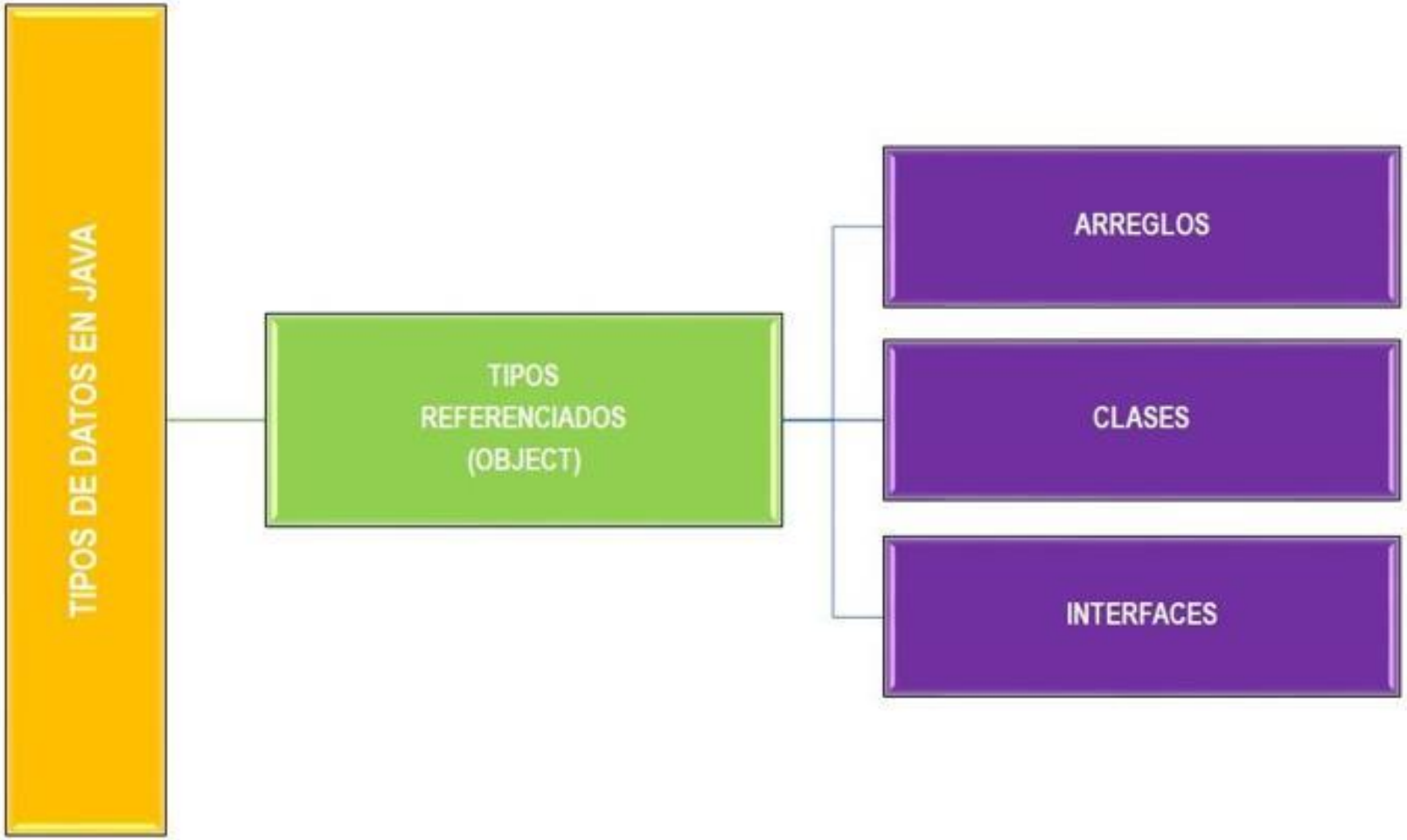
Aplicaciones Servlets

BASES DE PROGRAMACIÓN EN JAVA TIPOS DE DATOS



LA DECLARACIÓN DE LAS VARIABLES TIENEN LA MISMA FORMA QUE C++.

BASES DE PROGRAMACIÓN EN JAVA TIPOS DE DATOS



VARIABLES EN JAVA

FORMATO:

TIPO DE DATOS	IDENTIFICADOR
byte short int long double float char boolean	edad; cantidad; eventos; distancia; sueldo; raiz; categoría; bandera;

Al declarar variables se debe tener en cuenta:

- ❑ Asignar el nombre dependiendo de lo que va almacenar.
- ❑ Si la variable declarada es totalmente en mayúscula así se deberá usarse dentro de la aplicación, esto debido a la sensibilidad de mayúsculas y minúsculas de Java.
- ❑ Toda variable se compone de un nombre , tipo y valor a almacenar en él.

LA CLASE STRING

La clase String es una de las más utilizadas en las aplicaciones Java. Los desarrolladores utilizan cadenas para almacenar y procesar texto, incluyendo el texto capturado de la entrada del usuario o leer fuentes externas. Los objetos String pueden crear y utilizar cualquier aplicación Java. Dicha clase también proporciona una serie de funciones útiles para el acceso y la modificación de caracteres, que pueden incluir letras, números y signos de puntuación.

CONSTANTES EN JAVA

SINTAXIS

```
final nombreConstante = valor;
```

PALABRAS CLAVE - JAVA:

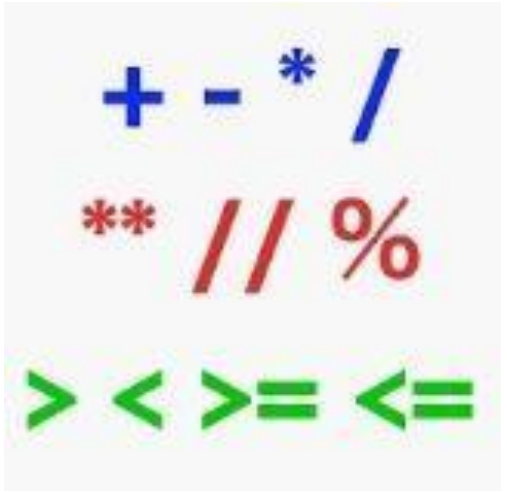
abstract	continue	for	new	switch
boolean	default	goto	null	synchronized
break	do	if	package	this
byte	double	implements	private	threadsafe
byvalue	else	import	protected	throw
case	extends	instanceof	public	transient
catch	false	int	return	true
char	final	interface	short	try
class	finally	long	static	void
const	float	native	super	while

Las siguientes son las palabras clave que están definidas en Java y que no se pueden utilizar como identificadores.

BASES DE PROGRAMACIÓN EN JAVA: OPERADORES EN JAVA

Conocido también como Operandos, y tenemos:

1. Operadores Aritméticos Binarios.
2. Operadores Aritméticos Unarios.
3. Operadores de Incremento y Decremento.
4. Operadores de Asignación.
5. Operadores de Comparación.
6. Operadores Lógicos.
7. Operador de Concatenación.



OPERADORES ARITMÉTICOS BINARIOS

OPERADOR	ACCIÓN	EJEMPLO

+	Suma	int n1=10;
		int n2=15;
		int suma=n1+n2;
		Resp: 25

-	Resta	int n1=20;
		int n2=30;
		int resta=n1-n2;
		Resp; -10

*	Multiplicación	int n1=5;
		int n2=9;
		int multiplicacion=n1*n2;
		Resp: 45

/	División	int n1=10;
		int n2=5;
		int divide=n1/n2;
		Resp: 2

%	Módulo	inr n1=4;
		int 2=3;
		int mod=n1%n2;
		Resp: 1

OPERADORES ARITMÉTICOS UNARIOS

OPERADOR	ACCIÓN	EJEMPLO
+	Positivo	char a='0'; int n=+a; Resp= 48

-	Negativo	int a=10;
		in t n=-a;
		Resp=-10

OPERADORES DE INCREMENTO Y DECREMENTO

OPERADOR	ACCIÓN	EJEMPLO

++	Prefija	int i=0;
		i++;
		Resp: 1
		Incrementa 1 en 1 pero se evalúa al valor anterior al incremento.

++	Posfija	int i=0; ++i; Resp:1 Incrementa 1 en 1 pero se evalúa el valor posterior al incremento.
--	Prefija	int i=10; i--; Resp: 9 Decrementa 1 en 1 pero se evalúa al valor anterior del incremento.
--	Posfija	int i=10; --i; Resp:9 Decrementa 1 en 1 pero se evalúa al valor posterior del incremento.

OPERADOR DE ASIGNACIÓN

OPERADOR	ACCIÓN	EJEMPLO

=	Asignación	int n;
		n=10;
		
		Se declara la variable n y se le asigna el valor de 10, también se podría implementar de la siguiente forma:
		
		int n=10;

OPERADOR DE COMPARACIÓN

OPERADOR	ACCIÓN	EJEMPLO

==	Igualdad Numérica o Caracter	if (n==10)
		if(sueldo==100.00)
		if(estado==true)
		while(n==10)

>	Mayor que	Evalúa si n supera a 10 if (n>10)
>=	Mayor o igual que	Evalúa si el valor de sueldo es mayor o igual a 100.00 if(sueldo>=100.00)
<	Menor	Evalúa si el valor de n es inferior a 10 dentro de un ciclo de repetición. while (n<10)
<=	Menor o igual que	if (sueldo<=100.00)
!=	Diferente	Evalúa si el valor de n no es igual a 10 if(n!=10)

OPERADOR DE LÓGICO

OPERADOR	ACCIÓN	EJEMPLO
&	Y Lógica a nivel de bit	if (n>=1 & n<=10) Devuelve verdadero si ambas condiciones son verdad.El compilador evalúa a ambas condiciones.
&&	Y Lógico	if (n>=1 && n<=10) Devuelve verdadero si ambas condiciones son verdad.
|	O Lógica a nivel de bit	if(n==1 | n<=10) Devuelve verdadero si una de las expresiones es verdadero.
||	O Lógico	if(n==1 || n<=10) Devuelve verdadero si una de las expresiones es verdaderO.
!	Menor o igual que	if !(n==1) Niega el valor obtenido en la condición.

OPERADOR DE CONCATENACIÓN

OPERADOR	ACCIÓN	EJEMPLO
+	Concatenar	Permite unir dos o más expresiones. Ejm: String día=”27”; String mes=”Enero”; String año=”2013”; String fecha=dia+mes+año; **Resp:27Enero2013**

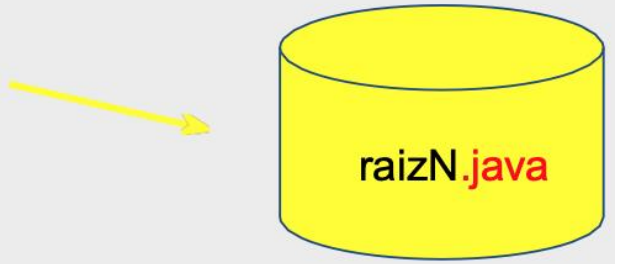
INTRODUCCIÓN A JAVA:

El proceso de programación en Java consta de 5 etapas: Edición, Compilación, Carga, Verificación y Ejecución; de éstos 5 mínimamente podríamos nombrar a Edición, Compilación y Ejecución.

1. **EDICIÓN**: En esta etapa el programador digita las instrucciones en Java en un editor el cual podrá corregir alguna parte del código si fuese necesario o grabar una chivo cuando termine su edición, cuando ello suceda se generará un archivo con extensión .java.

EDITOR DE CÓDIGO:

```
class Main {
    public static void main(String[] args) {
        double numero = 2;
        double elevado = Math.pow(numero, 32);
        System.out.println("El 2 elevado a la potencia 32 es " + elevado);
    }
}
```

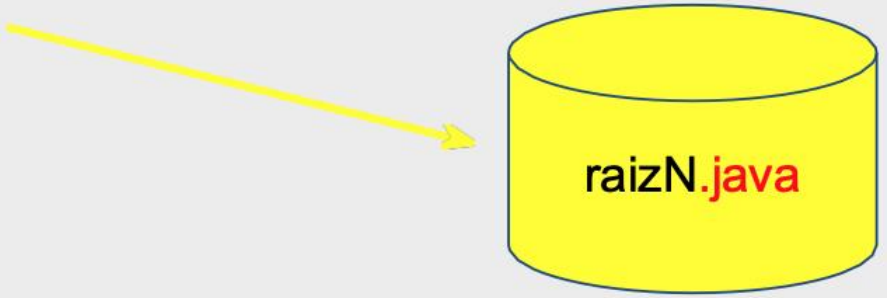


INTRODUCCIÓN A JAVA:

2.COMPILACIÓN: En esta etapa de transición se crean códigos de bytes y al ser guardados se crea el archivo con el mismo nombre del archivo Java pero con extensión .class.

CODEBYTE:

```
CA FE BA BE 00 00 00 34 00 14 0A 00 04 00 10 09
00 03 00 11 07 00 12 07 00 13 01 00 05 76 61 6C
6F 72 01 00 01 49 01 00 06 3C 69 6E 69 74 3E 01
00 03 28 29 56 01 00 04 43 6F 64 65 01 00 0F 4C
69 6E 65 4E 75 6D 62 65 72 54 61 62 6C 65 01 00
12 4C 6F 63 61 6C 56 61 72 69 61 62 6C 65 54 61
62 6C 65 01 00 04 74 68 69 73 01 00 20 4C 63 6F
6D 2F 61 64 69 63 74 6F 73 41 6C 54 72 61 62 61
6A 6F 2F 62 63 2F 42 43 54 65 73 74 3B 01 00 0A
53 6F 75 72 63 65 46 69 6C 65 01 00 0B 42 43 54
65 73 74 2E 6A 61 76 61 0C 00 07 00 08 0C 00 05
00 06 01 00 1E 63 6F 6D 2F 61 64 69 63 74 6F 73
41 6C 54 72 61 62 61 6A 6F 2F 62 63 2F 42 43 54
65 73 74 01 00 10 6A 61 76 61 2F 6C 61 6E 67 2F
4F 62 6A 65 63 74 00 21 00 03 00 04 00 00 00 01
00 00 00 05 00 06 00 00 00 01 00 01 00 07 00 08
```



INTRODUCCIÓN A JAVA:

3. CARGA: En esta etapa los códigos de bytes generados son enviados a la memoria de la computadora por medio del cargador de clases obtenidos desde la unidad donde se guardaron los archivos java y class.

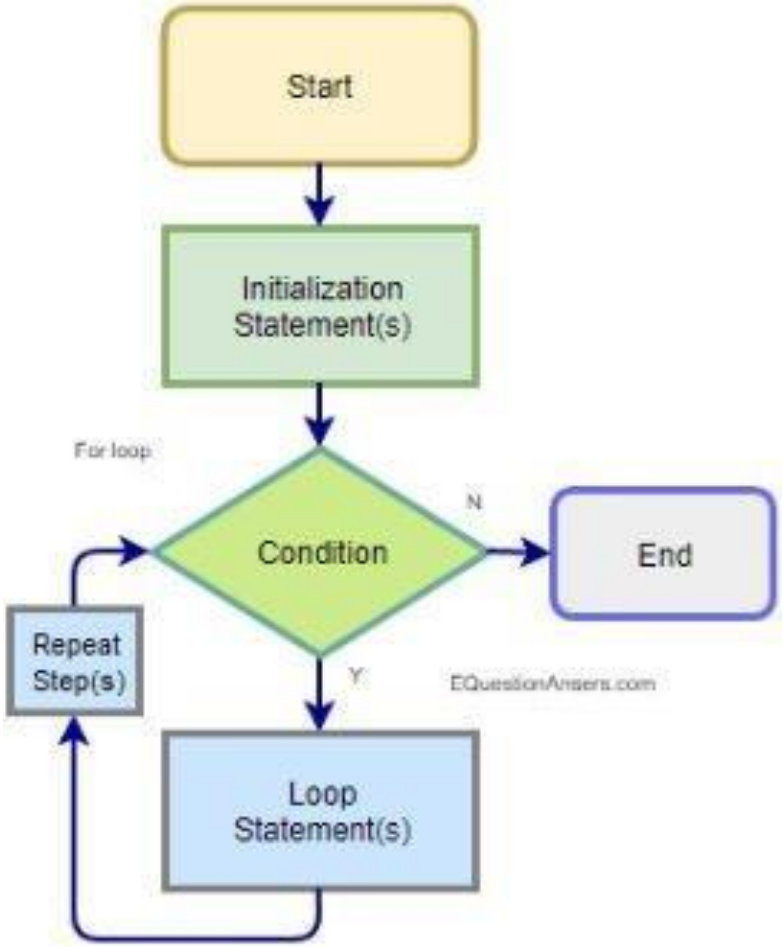
INTRODUCCIÓN A JAVA:

4.VERIFICACIÓN: Se da la verificación de que el código de bytes sea válido y no contenga violaciones de seguridad de seguridad, enviando el código a un intérprete que tendrá por misión que ese código sea entendible por la computadora.

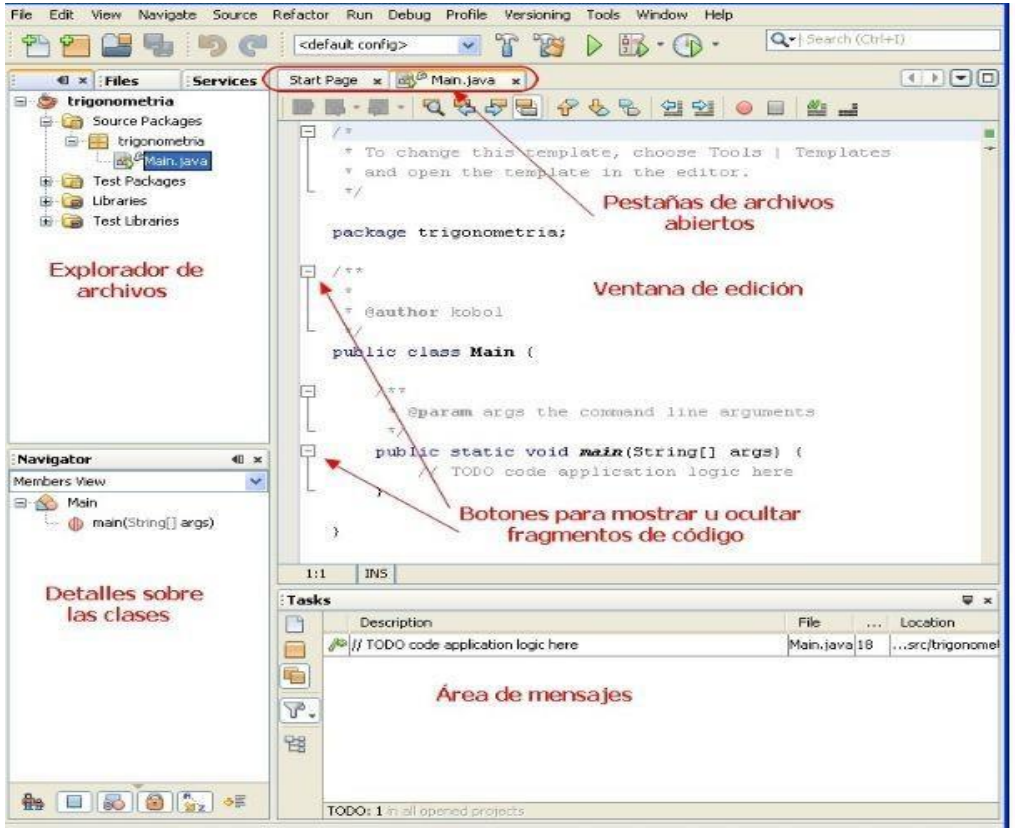
5.EJECUCIÓN: El código de bytes depurado es enviado a la Máquina Virtual de Java (JVM) para su ejecución y visualización en el entorno que el usuario crea conveniente.

ESTRUCTURAS DE CONTROL BÁSICA:

- Secuenciales
- Condicionales
  - Simples
  - Sobles
  - Anidadas
  - Múltiples
- Repetitivas



ENTORNO DE DESARROLLO APACHE NETBEANS



EJEMPLO

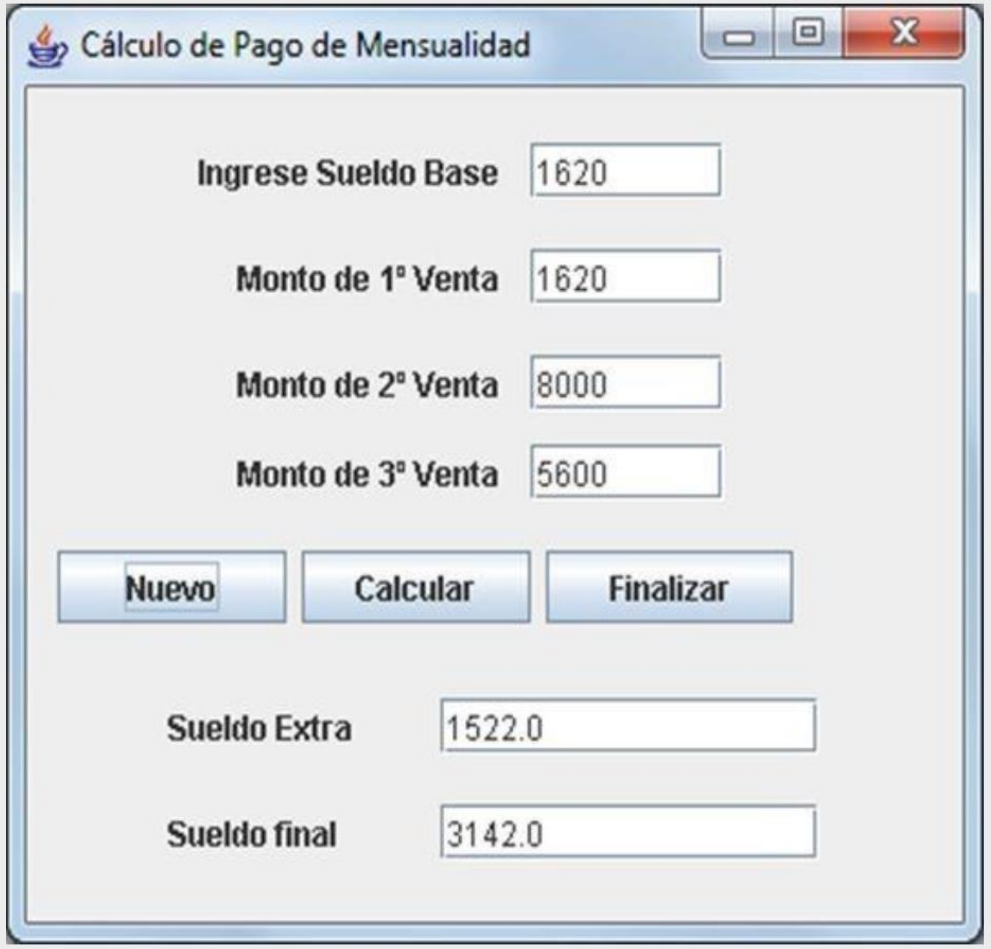
Un vendedor recibe mensualmente un sueldo base mas un extra, que es el 10% de cada venta que realice, el vendedor desea saber cuanto dinero extra obtendrá por las 3 ventas que realizó en el mes, además del total que recibirá en el mes tomando en cuenta su sueldo base y las comisiones.

DISEÑO DEL FORMULARIO

Lectura de los Datos de Entrada

Botones que contendrán las instrucción que permitirán obtener los datos de salida

Datos de salida



CODIFICACIÓN EN EL BOTÓN CALCULAR:

```
    private void jbtnCalcularActionPerformed(java.awt.event.ActionEvent evt) {

// Declaración de Variables
      double sb,mv1,mv2,mv3,extra1,extra2,extra3,totextra,sueldo;

// Datos de Entrada
       sb=Double.parseDouble(jtxtSbase.getText());
       mv1=Double.parseDouble(jtxtVenta1.getText());
       mv2=Double.parseDouble(jtxtVenta2.getText());
       mv3=Double.parseDouble(jtxtVenta3.getText());

 // Cálculos de extras y Pagos
       extra1=mv1*0.10;
       extra2=mv2*0.10;
       extra3=mv3*0.10;
       totextra=extra1+extra2+extra3;
       sueldo=sb+totextra;

 // Datos de Salida
      jtxtExtra.setText(String.valueOf(totextra));
      jtxtSFinal.setText(String.valueOf(sueldo));

 // Enfocar btnNuevo
      jbtnNuevo.requestFocus();
```

Las instrucciones de cómo se leían los datos de entrada y como se mostraban los datos de salida cambian

CODIFICACIÓN EN EL BOTÓN CALCULAR:

```
    private void jbtnNuevoActionPerformed(java.awt.event.ActionEvent evt) {

    // Reinicializar los objetos en blanco
    jtxtSbase.setText("");
    jtxtVenta1.setText("");
    jtxtVenta2.setText("");
    jtxtVenta3.setText("");
    jtxtExtra.setText("");
    jtxtSFinal.setText("");
    // Colocar el cursor en el jtxtSbase
    jtxtSbase.requestFocus();

    }
```

CODIFICACIÓN EN EL BOTÓN CALCULAR:

```
    private void jbtnFinalizarActionPerformed(java.awt.event.ActionEvent evt) {

        System.exit(0);

    }
```

ABSTRACCIÓN DE DATOS - POO

Persona 1
Nombre: José
Profesión: Granjero

Persona 4
Nombre: Ricardo
Profesión: Cartero

1 CLASE
12 OBJETOS

1 CLASE: Persona
12 OBJETOS: Instancias de la clase

Persona 12
Nombre: Geraldine
Profesión: Peluquera

ABSTRACCIÓN DE DATOS
¿CÓMO SE RELACIONA A LA PRÁCTICA PROFESIONAL?

- Está muy ligada a la comprensión y modelamiento de las reglas del negocio.
- Por ejemplo, el término “cuenta” podría tener diferentes acepciones de acuerdo al entorno/industria en el que estemos trabajando.

OCULTAMIENTO Y ENCAPSULAMIENTO
POO

- Ocultamiento es la propiedad que permite asegurar que los atributos (y sus estados) de un objeto estén ocultos al exterior.
- Encapsulamiento consiste en declarar dentro de una clase a los atributos con un acceso privado y los métodos con un acceso público.
- Esto conlleva a que para acceder, ya sea para consultar o modificar a los miembros de un objeto, esto deberá realizarse a través de sus métodos públicos.

DEMO EN NETBEANS

NECESIDAD DE ENCAPSULAR DATOS POO

- El encapsulamiento de datos permite que los atributos y estados de un objeto sean accesibles a través de métodos públicos.
- Cuando esto no sucede, es decir cuando se establece que los atributos sean públicos, éstos podrían tomar valores inconsistentes, lo cual sería fatal para cualquier sistema.

CONCEPTOS POO

- Instancia / objeto
- Clase
- Atributos (private)
- Métodos (public)
- Métodos set y get
- Constructor
- Abstracción
- Encapsulamiento

CONCLUSIONES
TRABAJO FUTURO

- La POO nos abre un nuevo panorama respecto a la forma cómo programamos y aumenta nuestras posibilidades de programar de manera ordenada y profesional.
- La POO habla de modelar la realidad que no es más que el entendimiento y control de las reglas de negocio.
- Existen funcionalidades más potentes y sofisticadas que veremos en detalle en próximas clases tales como herencia y polimorfismo que nos llevarán a ampliar aún más nuestra capacidad como analistas y programadores.

CONCLUSIONES
TRABAJO FUTURO

- Estas buenas prácticas no solo nos ayudarán en el desarrollo de las actividades de campo y en el avance del trabajo final a presentar al final del curso, sino en nuestra práctica profesional.

ACTIVIDAD DE CLASE

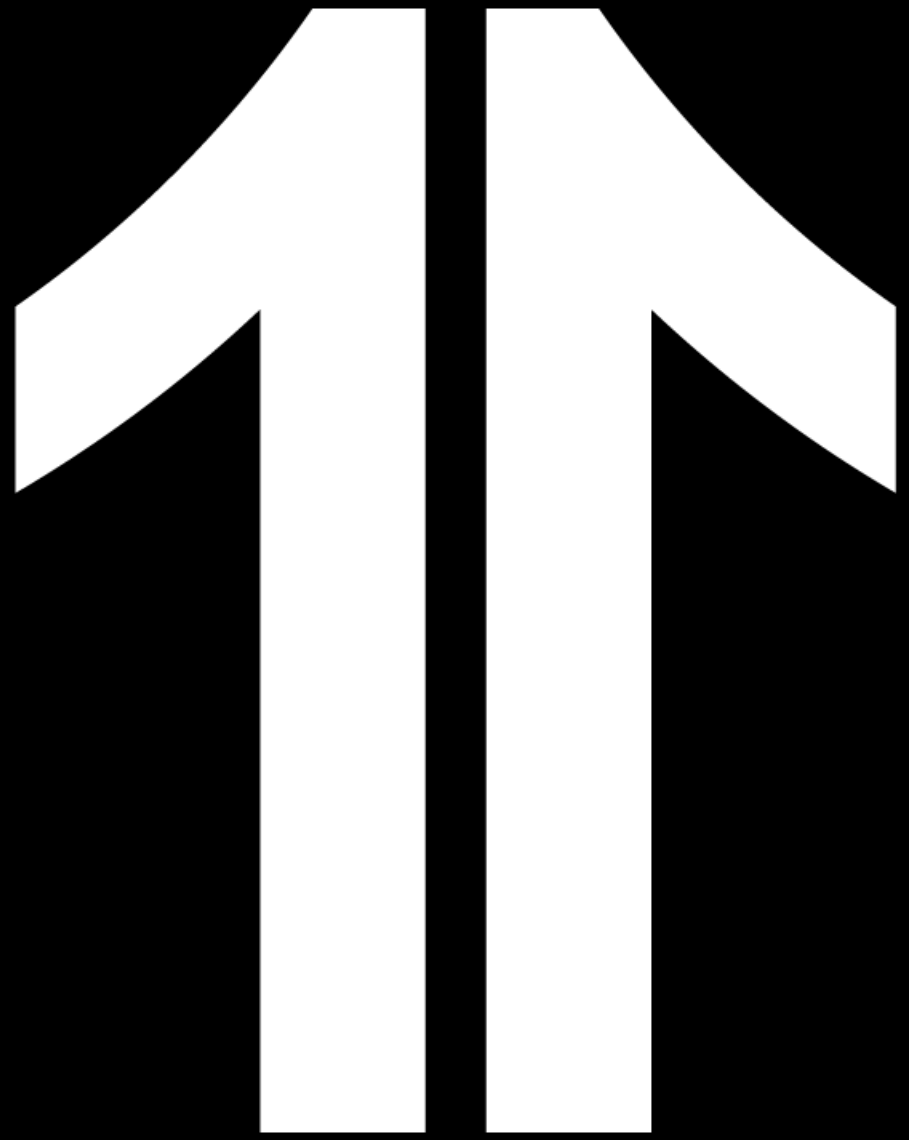
Ingresa a Recursos / Materiales para la sesión de clase: : Allí se encuentra las indicaciones y el instrumento con que será evaluado tu producto/evidencia de aprendizaje:

El docente establecerá la fecha y hora de entrega.

REFERENCIAS BIBLIOGRÁFICAS

Academia Oracle:

- https://myacademy.oracle.com/lmt/clmscoursecalendar.pr Main?site=oa&in_language_ logged_out=es



GRACIAS


UPN
UNIVERSIDAD
PRIVADA
DEL NORTE