



UNED

# **Asignatura 824**

# **Programación Intermedia**

## **Tema I**


Cátedra Tecnología de Sistemas, 2023

# Introducción a las computadoras, a internet y a Java



## Java

Es uno de los lenguajes de programación de computadoras más utilizados, de acuerdo con el índice TIOBE.

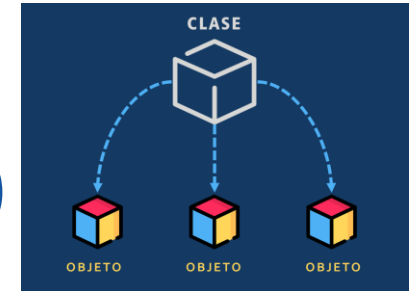
Dec 2022	Programming Language		Ratings
1		Python	16.66%
2		C	16.56%
<b>3</b>		<b>C++</b>	<b>11.94%</b>
4		Java	11.82%

# Introducción a las computadoras, a internet y a Java

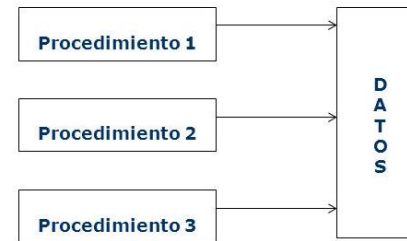


## Java

- Programación Orientada a Objetos (POO)



- Programación Procedimental



- Programación Genérica



## Programación Orientada a Objetos (POO)

- **Método**

*Alberga las declaraciones del programa que realizan sus tareas, además, oculta estas declaraciones al usuario.*

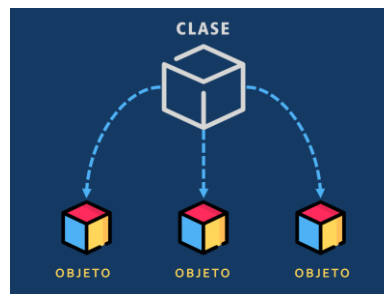
*Es una acción de una clase*

## Programación Orientada a Objetos (POO)

- Clase

*Conceptualmente, una clase es similar a los dibujos de ingeniería de un automóvil que albergan el diseño de un pedal del acelerador, de un volante y demás.*

*Es una plantilla para crear objetos.*

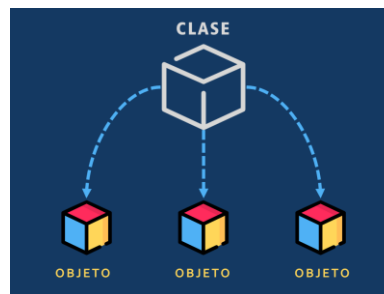


## Programación Orientada a Objetos (POO)

- Instanciación es el proceso de construir un objeto a partir de una clase.

- **Objeto**

*Es una instancia de una clase, contiene su estructura y métodos.*



## Clases

- *Se puede reutilizar “n” cantidad de veces para crear objetos.*
- *Permite la creación de otras clases, ahorra tiempo y esfuerzo de programación.*

## Atributos

- *Son características de las clases.*
- *Se especifican como parte de la clase del objeto. Por ejemplo, un objeto de cuenta bancaria tiene un atributo de saldo que representa la cantidad de dinero en la cuenta. **Cada objeto de cuenta bancaria conoce el saldo de la cuenta que representa, pero no los saldos de las otras cuentas del banco.***



## Encapsulamiento

- *Las clases encapsulan, es decir, ocultan sus atributos y métodos. Los atributos y métodos de una clase están íntimamente relacionados.*
- *Los objetos pueden comunicarse entre sí, pero normalmente no se les permite saber cómo se implementan otros objetos; los detalles de implementación se pueden ocultar dentro de los propios objetos.*

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## Comentarios en sus programas

- *Se utilizan para documentar el código*
- *Son ignorados por el compilador*
- *Comentarios de fin de línea*

```
// Fig. 2.1: Bienvenido1.java
```

- Comentarios tradicionales

```
/* Este es un comentario tradicional. Se  
puede dividir en varias líneas */
```

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## Puntos importantes en las clases

- *Declaración de clase:* `public class Bienvenido1`
- *Por convención, los nombres de clase comienzan con una letra mayúscula, así como la primera letra de cada palabra que incluyen:*  
*ClaseDeEjemplo*
- *Java es sensible a mayúsculas y minúsculas: las letras mayúsculas y minúsculas son distintas, por lo que “valor” y “Valor” son identificadores diferentes.*

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## System.out.println

- *Muestra un mensaje en la ventana de comandos.*

```
System.out.println("Bienvenido a la programacion en Java!");
```

- *Una sola instrucción puede mostrar varias líneas mediante el uso de caracteres de nueva línea (\n).*

```
System.out.println("Bienvenido\na la\nprogramacion\nen Java!");
```

# Introducción a las aplicaciones en Java: entrada/salida y operadores



## Secuencias de escape

- *Tienen un significado especial en los métodos `print` y `println`.*

Secuencia de escape	Descripción
<code>\n</code>	Nueva línea. Coloca el cursor de la pantalla al inicio de la <i>siguiente</i> línea.
<code>\t</code>	Tabulador horizontal. Mueve el cursor de la pantalla a la siguiente posición del tabulador.
<code>\r</code>	Retorno de carro. Coloca el cursor de la pantalla al inicio de la línea <i>actual</i> ; <i>no</i> avanza a la siguiente línea. Cualquier carácter que se imprima después del retorno de línea <i>sobrescribe</i> los caracteres previamente mostrados en esa línea.
<code>\\</code>	Diagonal inversa. Se usa para imprimir un carácter de diagonal inversa.
<code>\"</code>	Doble comilla. Se usa para imprimir un carácter de doble comilla. Por ejemplo, <code>System.out.println("\entre comillas");</code> muestra “entre comillas”.

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## System.out.printf

- *Muestra un mensaje con formato en la ventana de comandos.*

```
System.out.printf("%s\n%s\n", "Bienvenido a", "la programacion en Java!")
```

- *Paquetes: las clases en Java se agrupan en paquetes, se utiliza la palabra “import” para poder utilizarlos o las clases dentro de estos.*

```
import java.util.Scanner;
```

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## Lectura desde la ventana de consola

- *Se utiliza la clase “Scanner”*

```
import java.util.Scanner;  
  
Scanner entrada = new Scanner(System.in);  
  
int numero1 = entrada.nextInt();
```

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## Declaración de variables

- *Para declarar una variable se escribe primero su tipo y luego el nombre de esta.*

```
int numero1; // declara la variable int numero1  
numero1 = entrada.nextInt(); // asigna la entrada del usuario a numero1
```



# Introducción a las aplicaciones en Java: entrada/salida y operadores



## Aritmética

- Java utiliza los siguientes operadores aritméticos.*

Operación Java	Operador	Expresión algebraica	Expresión Java
Suma	+	$f + 7$	<code>f + 7</code>
Resta	-	$p - c$	<code>p - c</code>
Multiplicación	*	$bm$	<code>b * m</code>
División	/	$x / y$ o $\frac{x}{y}$ o $x \div y$	<code>x / y</code>
Residuo	%	$r \bmod s$	<code>r % s</code>

# Introducción a las aplicaciones en Java: entrada/salida y operadores



## Reglas de precedencia de operadores

- *Utiliza las mismas reglas que el álgebra.*

Operador(es)	Operación(es)	Orden de evaluación (precedencia)
* / %	Multiplicación División Residuo	Se evalúan primero. Si hay varios operadores de este tipo, se evalúan de izquierda a derecha.
+ -	Suma Resta	Se evalúan después. Si hay varios operadores de este tipo, se evalúan de izquierda a derecha.
=	Asignación	Se evalúa al último.

# Introducción a las aplicaciones en Java: entrada/salida y operadores

## Condición

- *Es una expresión que puede ser verdadera (true) o falsa (false).*
- *Utiliza instrucción “if”, esta permite que un programa tome una decisión con base en el valor de una condición.*

```
if (numero1 == numero2) {  
    System.out.printf("%d == %d%n", numero1, numero2);  
}
```

# Introducción a las aplicaciones en Java: entrada/salida y operadores



## Operadores de igualdad y relacionales

Operador algebraico	Operador de igualdad o relacional de Java	Ejemplo de condición en Java	Significado de la condición en Java
<i>Operadores de igualdad</i>			
=	==	x == y	x es igual a y
≠	!=	x != y	x no es igual a y
<i>Operadores relacionales</i>			
>	>	x > y	x es mayor que y
<	<	x < y	x es menor que y
≥	>=	x >= y	x es mayor o igual que y
≤	<=	x <= y	x es menor o igual que y

## Instrucciones de control: Parte I; operadores de asignación, ++ y —

### Operadores de asignación compuestos

*Abrevian las expresiones de asignación.*

```
c = c + 3; // suma 3 a c
```

*La expresión anterior se puede abreviar como:*

```
c += 3; // suma 3 a c de manera más concisa
```

Operador de asignación	Expresión de ejemplo	Explicación	Asigna
<i>Supongamos que: int c = 3, d = 5, e = 4, f = 6, g = 12;</i>			
+=	c += 7	c = c + 7	10 a c
-=	d -= 4	d = d - 4	1 a d
*=	e *= 5	e = e * 5	20 a e
/=	f /= 3	f = f / 3	2 a f
%=	g %= 9	g = g % 9	3 a g

# Instrucciones de control: Parte I; operadores de asignación, ++ y —



## Operadores de incremento y decremento

- Utilizados para aumentar o disminuir el valor de una variable.*

Operador	Expresión de ejemplo	Explicación
++ (preincremento)	++a	Incrementar a en 1, después utilizar el nuevo valor de a en la expresión en que esta variable reside.
++ (postincremento)	a++	Usar el valor actual de a en la expresión en la que esta variable reside, después incrementar a en 1.
-- (predecremento)	--b	Decrementar b en 1, después utilizar el nuevo valor de b en la expresión en que esta variable reside.
-- (postdecremento)	b--	Usar el valor actual de b en la expresión en la que esta variable reside, después decrementar b en 1.

## Instrucciones de control: Parte 2; operadores lógicos



### Instrucciones de iteración

- *for*
  - *while*
  - *do...while*
- 
- *Permiten a los programas realizar instrucciones repetidamente siempre que una condición (llamada condición de continuación del ciclo) siga siendo verdadera.*

## Instrucciones de control: Parte 2; operadores lógicos

### Operadores lógicos

- *And (&&)*
- *Or (||)*
- *Or exclusivo (^)*
- *Negación (!)*
  
- *Las instrucciones if, if...else, while, do...while y for requieren una condición para determinar cómo continuar el flujo de control de un programa, por medio de los operadores lógicos se pueden combinar las instrucciones anteriores.*



## Instrucciones de control: Parte 2; operadores lógicos



### Precedencia de operadores lógicos

Operadores	Asociatividad	Tipo
&	de izquierda a derecha	AND lógico booleano
^	de izquierda a derecha	OR exclusivo lógico booleano
	de izquierda a derecha	OR inclusivo lógico booleano
&&	de izquierda a derecha	AND condicional
	de izquierda a derecha	OR condicional

## Métodos “static”

- *Le pertenecen a la clase y no a las instancias.*
- *Son convenientes para tareas comunes.*
- *Un ejemplo es la clase Math (para cálculos matemáticos comunes)*

### Método “main”

- *Método de inicio de un programa en java.*
- *Debe ser “static”, ya que la JVM (Java Virtual Machine) no requiere crear una instancia.*

## Sobrecarga de métodos

- *Los métodos del mismo nombre se pueden declarar en la misma clase si tienen diferentes conjuntos de parámetros (determinados por el número, los tipos y el orden de los parámetros).*

```
// método cuadrado con argumento int
public static int cuadrado(int valorInt) {
    System.out.printf("%nSe llamo a cuadrado con argumento int: %d%n",
        valorInt);
    return valorInt * valorInt
}

// método cuadrado con argumento double
public static double cuadrado(double valorDouble) {
    System.out.printf("%nSe llamo a cuadrado con argumento double: %f%n",
        valorDouble);
    return valorDouble * valorDouble;
}
```

***“Si no estás dispuesto a aprender nadie te puede ayudar. Si estás dispuesto a aprender nadie te puede parar.” (Proverbio chino)***





**¡GRACIAS!**