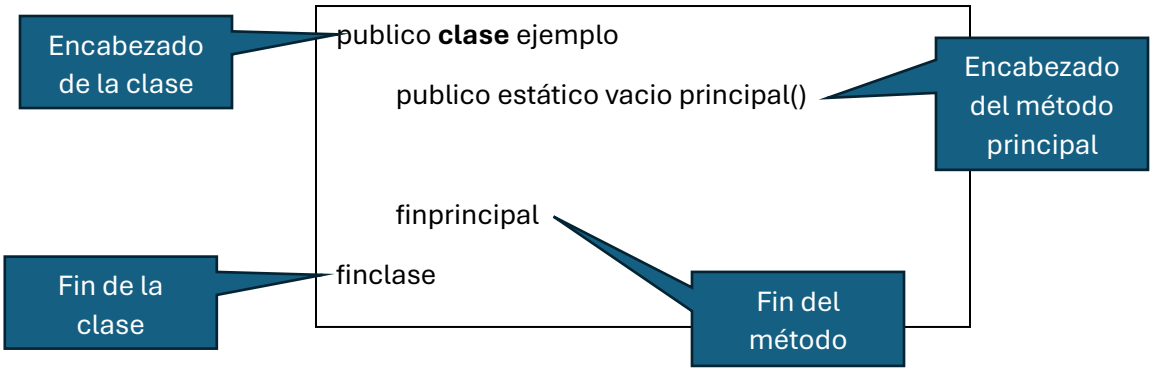
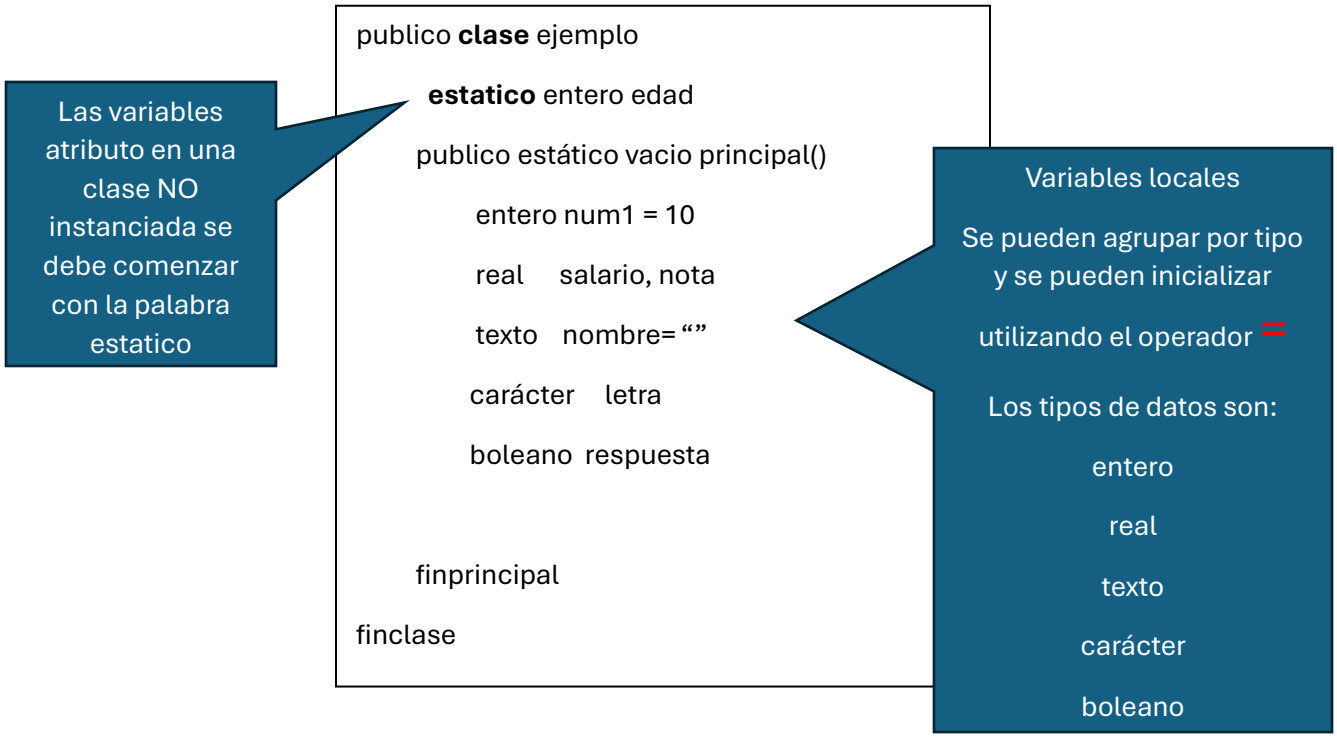
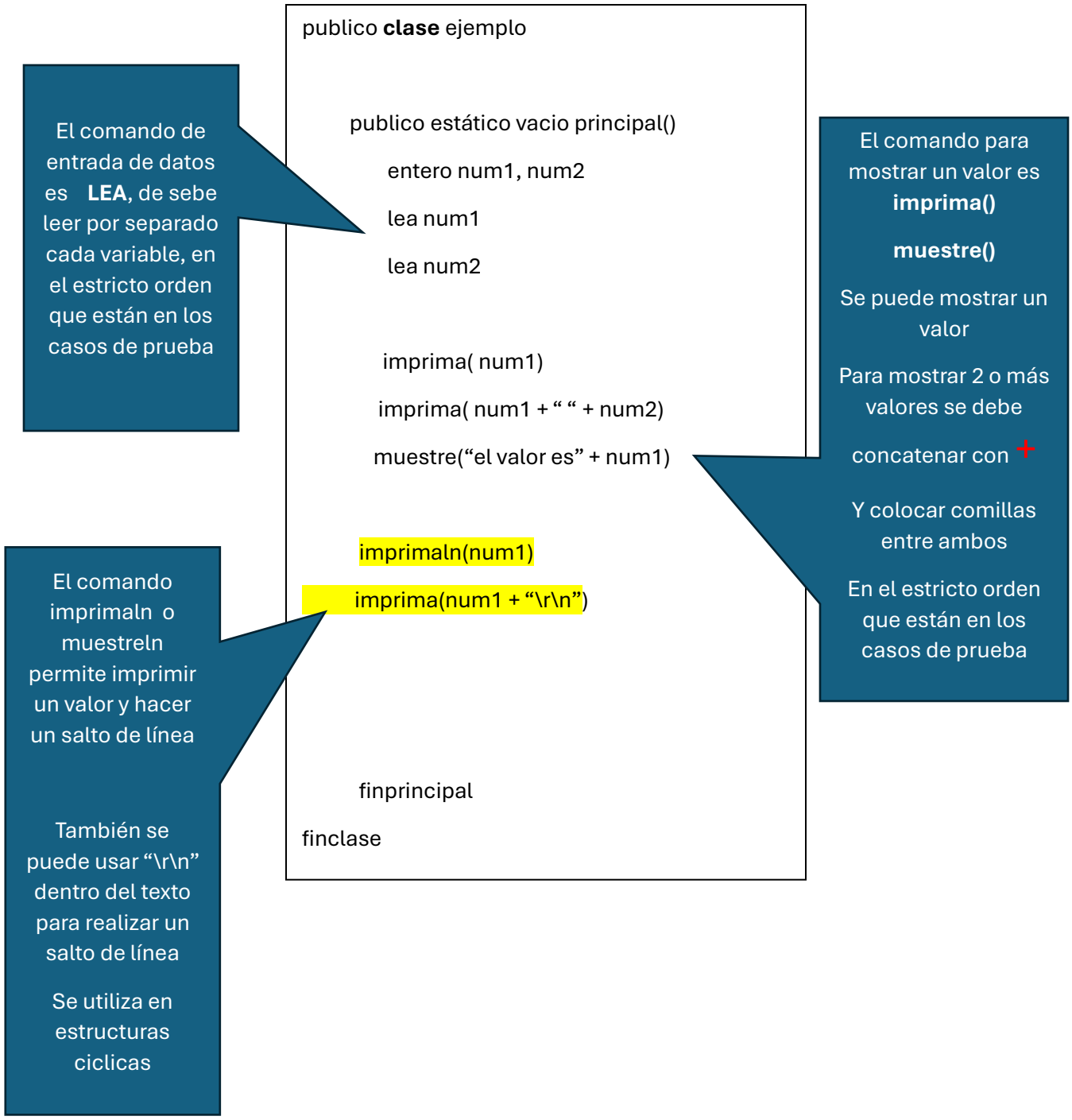
ESTRUCTURA GENERAL DE UNA CLASE



CREACION DE VARIABLES ATRIBUTO, LOCALES Y TIPOS DE DATOS EN UNA CLASE



COMANDO DE LECTURA Y ESCRITURA



FORMATEAR UN VALOR

```
publico clase ejemplo
        estatico entero edad
       publico estático vacio principal()
            entero num1, num2
            lea num1
            lea num2

1.        imprima(formatear( num,2))
2.        muestre(formatear(num,3)  + “ “ +formatear( num,1))
3.       imprima(formatear(num,2) + “ “)
          Imprima(formatear(num,0))
4.       muestre(“el valor es”+formatear(num,2))
         finprincipal
finclase
```

El comando para indicar la cantidad de decimales que debe tener un numero es FORMATEAR, cada valor se debe formatear por separado como ocurre en la opción 2

Esa misma línea se pude dividir en 2 como ocurre en la opción 3, el resultado es el mismo

2, 3 indica la cantidad de decimales a mostrar, sino desea mostrar decimales se coloca 0

OPERADORES ARITMÉTICOS

```
publico clase ejemplo


    publico estático vacio principal()
        entero num1, num2, resultado


        lea num1
        lea num2
        resultado=  num1 +num2
        resultado = num1 – num2
        resultado = num1 * num2
        resultado = num1 / num2
        resultado = num1 mod num2


       //funciones
        resultado = potencia(num1, 2)     // potencia(base, exp)
        resultado = raiz(num1)            //raíz cuadra
        resultado  =seno(num1)            //seno de un numero
        resultado= coseno(num1)           // coseno de un numero
        resultado = tangente(num1)        //tangente de un numero


       texto frase=”bienvenido”
       entero tamaño= longitud(frase)     //devuelve  10
       carácter letra= frase.obtenercaracter(2)   //devuelve  (e)
       entero  índice= frase.obtenerindice(“d”)  //devuelve  8


    imprima(formatear(resultado,2))


     finprincipal
finclase
```

Para las operaciones matemáticas se utilizan los siguientes operadores aritméticos

- \+ (suma)
- \- (resta)
- * (multiplicación)
- / (división exacta)
- mod (residuo de una división)

Funciones

- potencia (base, exponente)
- raíz(numero)
- seno(numero)
- coseno(numero)
- tangente(numero)
- longitud(cadena)
- longitud(vector)
- cadena.obtenercaracter(indice)
- cadena.obtenerindice(caracter)

CASTEAR UN NUMERO

```
publico clase ejemplo


      publico estático vacio principal()
            entero num1, num2
            real resultado
            lea num1
            lea num2
            resultado= (real) num1 / num2


       imprima(formatear(resultado,2))


        finprincipal
finclase
```

Castear es convertir un valor de un tipo a otro tipo de dato ejemplo de real a entero o de entero a real

Es muy común realizar operaciones enteras el resultado es un entero, la operación que puede dar más confusión es la división, dividir dos números enteros el resultado es entero

Ejemplo 1/2 es resultado es 0

Si deseamos que sea 0.5

Seria (real) 1/2 el resultado es 0.5

Si deseamos de real a entero

Seria (entero) 2.5 / 2.0 el resultado es 1

ESTRUCTURA CONDICIONAL

CONDICIONAL SIMPLE

```
publico clase ejemplo

    publico estático vacio principal()
        entero num1, num2, resultado=0
        lea num1
        lea num2
        // SIMPLE
        si(num1 > num2)
            resultado = num1+ num2
        finsi

        si(num1 > num2 ^ num2 <= 10)
            resultado = num1* num2
        finsi

        si(num1== num2  v num2>= 20)
            resultado = num1 / num2
        finsi

        imprima(formatear(resultado,2))

        finprincipal
finclase
```

Estructura condicional simple

si (condición (s))

Instrucción(s)

finsi

Operadores relacionales

< (menor que)

<= (menor o igual que)

> (mayor que)

>= (mayor o igual que)

== (igualdad)

<> (diferente)

Operadores lógicos

^ (y)

v (O)

Se ejecuta la instrucción si la comparación es verdadero

Para evitar posibles errores colocar los operadores relacionales y los logicos entre espacios, ejemplo

A < B

A >= B v B > C

A <> B

Evitar esto

A<BvB<=C

A<>B

ESTRUCTURA CONDICIOANL COMPUESTA

```
publico clase ejemplo

    publico estático vacio principal()
        entero num1, num2, resultado=0
        lea num1
        lea num2
        // COMPUESTA
        si(num1 > num2)
            resultado = num1+ num2
        sino
            resultado= num1 – num2
        finsi

    imprima(formatear(resultado,2))

    finprincipal
finclase
```

Se ejecuta esta instrucción si la comparación es verdadera

Se ejecuta esta instrucción si la comparación es falsa

Estructura condicional COMPUESTA

si (condición (s))

Instrucción(s)

sino

Instrucción(s)

finsi

ANIDADA

```
publico clase ejemplo


    publico estático vacio principal()
        entero num1, num2, resultado=0
        lea num1
        lea num2
        // ANIDADA
        si(num1 > num2)
            si(num == 20)
                resultado = num1+ num2
            sino
                resultado= num1 – num2
            finsi
        sino
            si(num1 <> 50)
                resultado = num1* num2
            sino
                resultado= num1/ num2
            finsi
        finsi


    imprima(formatear(resultado,2))


    finprincipal
finclase
```

Estructura condicional ANIDADA

si (condición (s))

Instrucción(s)

sino

si(condición(es)

Instrucción(s)

finsi

finsi

Cada instrucción SI debe tener su respectivo finsi

ESTRUCTURA MULTICONDICIONAL (SEGÚN)

```
publico clase ejemplo

    publico estático vacio principal()
        entero num1, num2, resultado=0
        lea num1
        lea num2
        según(num1)
            caso 1:
                resultado=num1+num2
                salto
            caso 2:
                resultado= num1-num2
                romper
            caso 3:
                resultado= num1/ num2
                salto
            caso 4:
                resultado= num1 * num2
                salto
            en otro caso:
                imprima("error")
        finsegun
        imprima(formatear(resultado,2))

    finprincipal
finclase
```

La variable en el según no puede ser real

Estructura multicondicional (según)

Según(variable o expresión)

- Caso valor1:
  - Instrucción(s)
  - romper
- Caso valor2:
  - Instrucción(s)
  - Salto
- Caso valor3:
  - Instrucción(s)
  - Salto
- En otro caso:
  - Instrucción(s)
  - Salto

Finsegun

Se puede usar la palabra salto o romper

OPERADOR TERNARIO

```
publico clase ejemplo


    publico estático vacio principal()
        entero edad
        lea edad
        texto mensaje = (edad >= 18) ? “Eres mayor de edad” : “Eres menor de edad”
        Imprima(mensaje)


     finprincipal
finclase
```

La sintaxis básica del operador ternario es la siguiente:

(condición) ? expresión_si_verdadero : expresión_si_falso

TRABANADO CON 2 O MAS METODOS

1. **No Envía Parámetros, No Recibe Parámetros**

```
publico clase Operaciones
    publico estatico vacio principal()
        calcular()
    finmetodo

    publico estatico vacio metodo calcular()
        real nota1,nota2,nota3,nd
        lea nota1
        lea nota2
        lea nota3
        nd=(nota1 + nota2 + nota3)/3
        Imprima(nd)
    finmetodo
finclase
```

De esta forma se invoca un método sin enviar parámetros

Se debe tener este orden

Publico (o el especificador de acceso)

estático(si no ha sido instanciada)

vacío (o el tipo devuelto)

Método (palabra reservada)

Nombre del método y luego paréntesis

2. Envía parámetros pero no recibe Parámetros

```
publico clase Operaciones
publico estatico vacio principal()
real nota1,nota2,nota3
lea nota1
lea nota2
lea nota3
calcular(nota1,nota2,nota3)
Finmetodo
publico estatico vacio metodo calcular(real n1,real n2,real n3)
real nd
nd=(n1 + n2 + n3)/3
imprima(nd)
finmetodo
finclase
```

Se invoca un método y se envían parámetros

Se reciben los parámetros indicando por cada variable el tipo de dato, se reciben en su estricto orden

3. No Envía Parametros y Recibe parámetro

```
publico clase Operaciones
    publico estatico vacio principal()
        real notadef
        notadef=calcular()
        imprima(notadef)
    finmetodo
    publico estatico real metodo calcular()
        real nota1,nota2,nota3,nd
        lea nota1
        lea nota2
        lea nota3
        nd=(nota1 + nota2 + nota3)/3
        retorne nd
    finmetodo
finclase
```

Se invoca un método y se recibe el resultado

Se debe cambiar por el tipo devuelto

La palabra RETORNE indica devolver un valor al método que lo invoco

- **4. Envía Parametros y Recibe parámetros**

```
publico clase Operaciones
    publico estatico vacio principal()
        real nota1,nota2,nota3,notadef
        lea nota1
        lea nota2
        lea nota3
        notadef=calcular(nota1,nota2,nota3)
        imprimia(notadef)
    finmetodo
    publico estatico real metodo calcular(real n1, real n2, real n3)
        real nd
        nd=(n1 + n2 + n3)/3
        retorne nd
    finmetodo
finclase
```

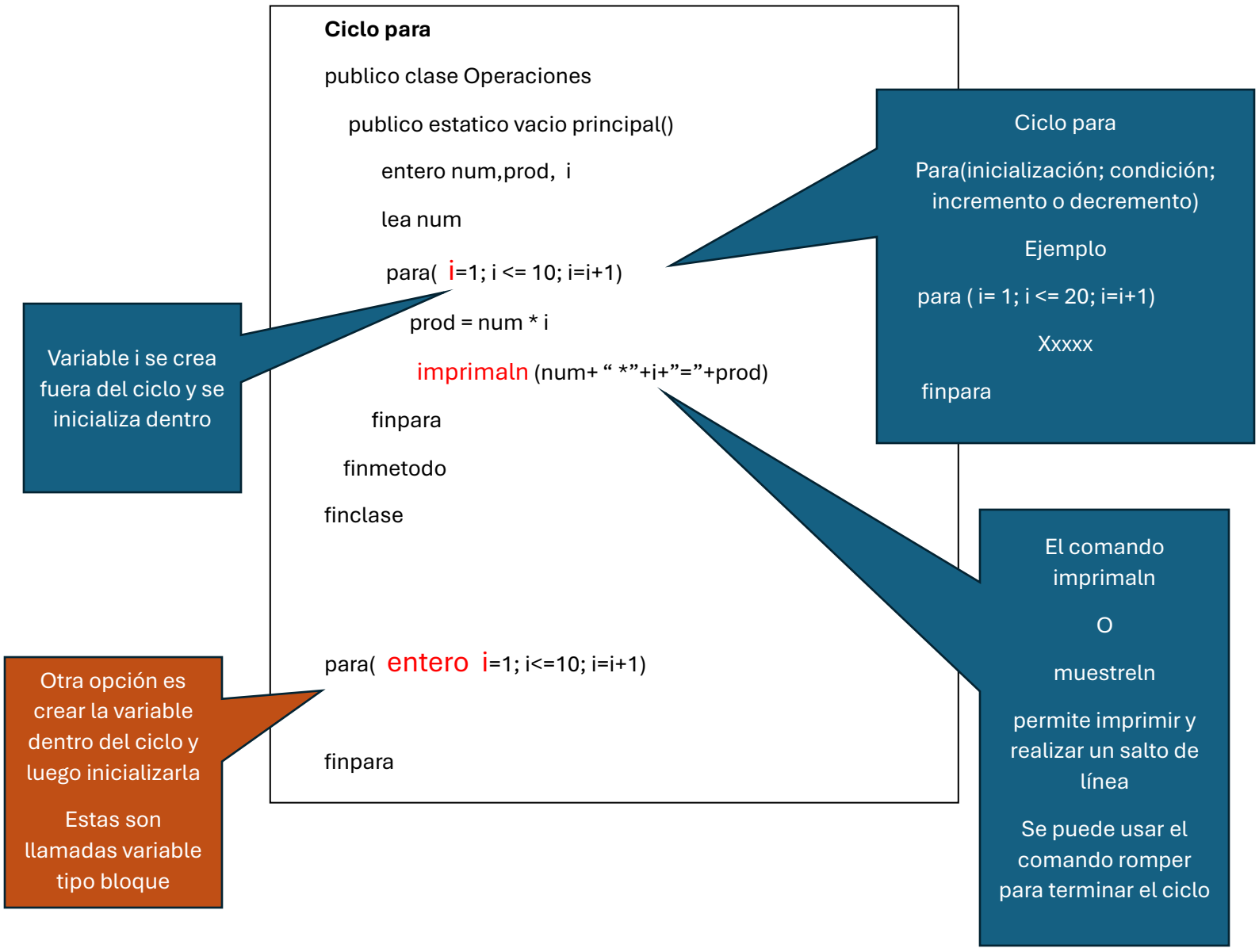
Se invoca el método, se envían parametros y se recibe el resultado

Se debe cambiar por el

tipo devuelto

La palabra RETORNE indica devolver un valor al método que lo invoco

ESTRUCTURAS CICLICAS



Ciclo mientras

```
publico clase Operaciones
    publico estatico vacio principal()
        entero num,prod,  i
        lea num
        i= 1
        mientras(i <= 10)
            prod = num * i
            imprimaln (num+ " *"+i+"="+prod)
            i= i+1
        finmientras
    finmetodo
finclase
```

Variable i se crea y se inicializa fuera del ciclo, en el mientras se coloca la condición o condiciones

Ciclo mientras

Inicialización de variable

Mientras(condición)

Se altera la variable

finmientras

Ejemplo

```
Entero i
I= 1
   mientras ( i <= 20)
                Xxxxx
                I= i+1
    finmientras
```

Ciclo HAGA MIENTASQUE

Ciclo Haga-Mientrasque

```
publico clase Operaciones
    publico estatico vacio principal()
        entero edad
        haga
                lea edad
        mientrasque(edad < 18)
        imprima (“tienes “  + edad + “eres mayor de edad”)
finmetodo
finclase
```

Si la condición es verdadera vuelve al ciclo en la instrucción haga

Ciclo haga- mientrasque

Haga

Xxxxxx

Xxxxxxx

Mientrasque(condición(s))

Mecanismos de control de flujo (romper y continuar)

ROMPER O SALTO

```
publico clase Operaciones
    publico estatico vacio principal()
        entero num,prod,  i
        lea num
        para(  i=1; i <= 10; i=i+1)
            si ( i ==6)
                romper
            finsi
            imprimaln (num+ " *"+i+"="+prod)
        finpara
    finmetodo
finclase
```

El comando romper o salto

Este comando se utiliza para terminar la ejecución de un ciclo de manera inmediata, en este caso cuando la variable i sea igual a 6 se termina el ciclo, sin importar que la condición del PARA sea hasta 10

CONTINUAR

```
publico clase Operaciones
    publico estatico vacio principal()
        entero num,prod,  i
        lea num
         para(  i=1; i <= 10; i=i+1)
            si ( i  mod 2 ==0)
               continuar
            finsi
             imprimaln (num+ " *"+i+"="+prod)
        finpara
    finmetodo
finclase
```

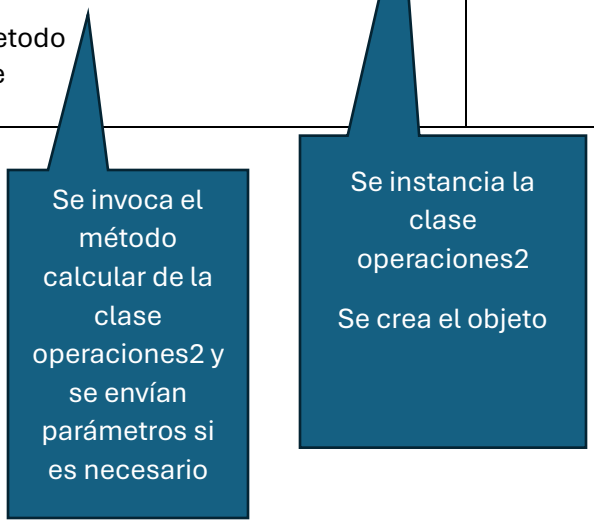
El comando continuar

el comando continuar no sale del ciclo por completo. En su lugar, cuando se ejecuta continuar dentro de un bucle, salta la iteración actual y pasa a la siguiente iteración del mismo ciclo

en este caso solo imprime cuando la variable i es impar

TRABAJANDO CON 2 CLASES

CLASE1	CLASE2
publico clase Operaciones publico estatico vacio principal() real nota1,nota2,nota3 lea nota1 lea nota2 lea nota3 operaciones2 op = nuevo operaciones() op.calcular(nota1,nota2,nota3) finmetodo finclase	Publico clase operaciones2 publico vacio método calcular(real n1,real n2,real n3) real nd nd=(n1 + n2 + n3)/3 imprima(nd) finmetodo finclase



USANDO METODO CONSTRUCTOR

CLASE1	CLASE2
publico clase Operaciones publico estatico vacio principal() real nota1, nota2,nota3 lea nota1 lea nota2 lea nota3 operaciones2 op = nuevo operaciones(n1,n2,n3) op.calcular() finmetodo finclase	Publico clase operaciones2 real n1,n2,n3 Publico método operaciones2(real n1, real n2, real n3) esteobjeto.n1=n1 esteobjeto.n2=n2 esteobjeto.n3=n3 finmetodo publico vacio método calcular() real nd nd=(n1 + n2 + n3)/3 imprima(nd) finmetodo finclase

Se instancia la clase operaciones2

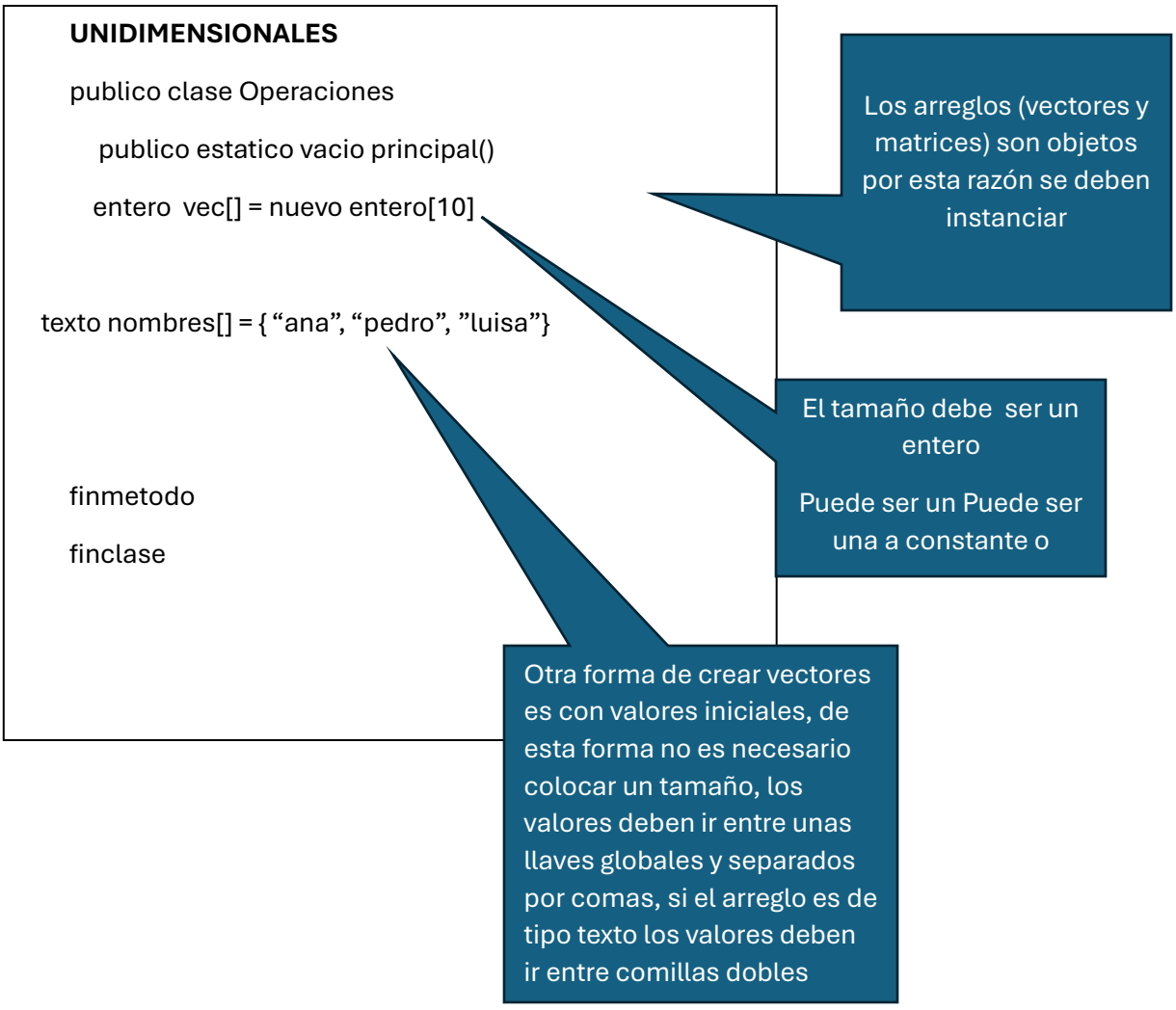
Se crea el objeto, se envían parámetros al método constructor

Método constructor, debe tener el mismo nombre de la clase, No tiene tipo de dato devuelto

Se utiliza la palabra esteobjeto para hacer referencia a la variable atributo

ARREGLOS

1.UNIDIMENSIONALES (VECTORES)



RECORRER UN VECTOR

UNIDIMENSIONALES

```
publico clase Operaciones
    publico estatico vacio principal()
    entero  edades[] = nuevo entero[10]

texto nombres[] = { “ana”, “pedro”, ”luisa”}

Para(i = 0; i < 10;i=i+1)
   lea edades[i]
finpara

Para(i = 0; i < longitud(edades) ;i=i+1)
   lea edades[i]
finpara

    finmetodo
    finclase
```

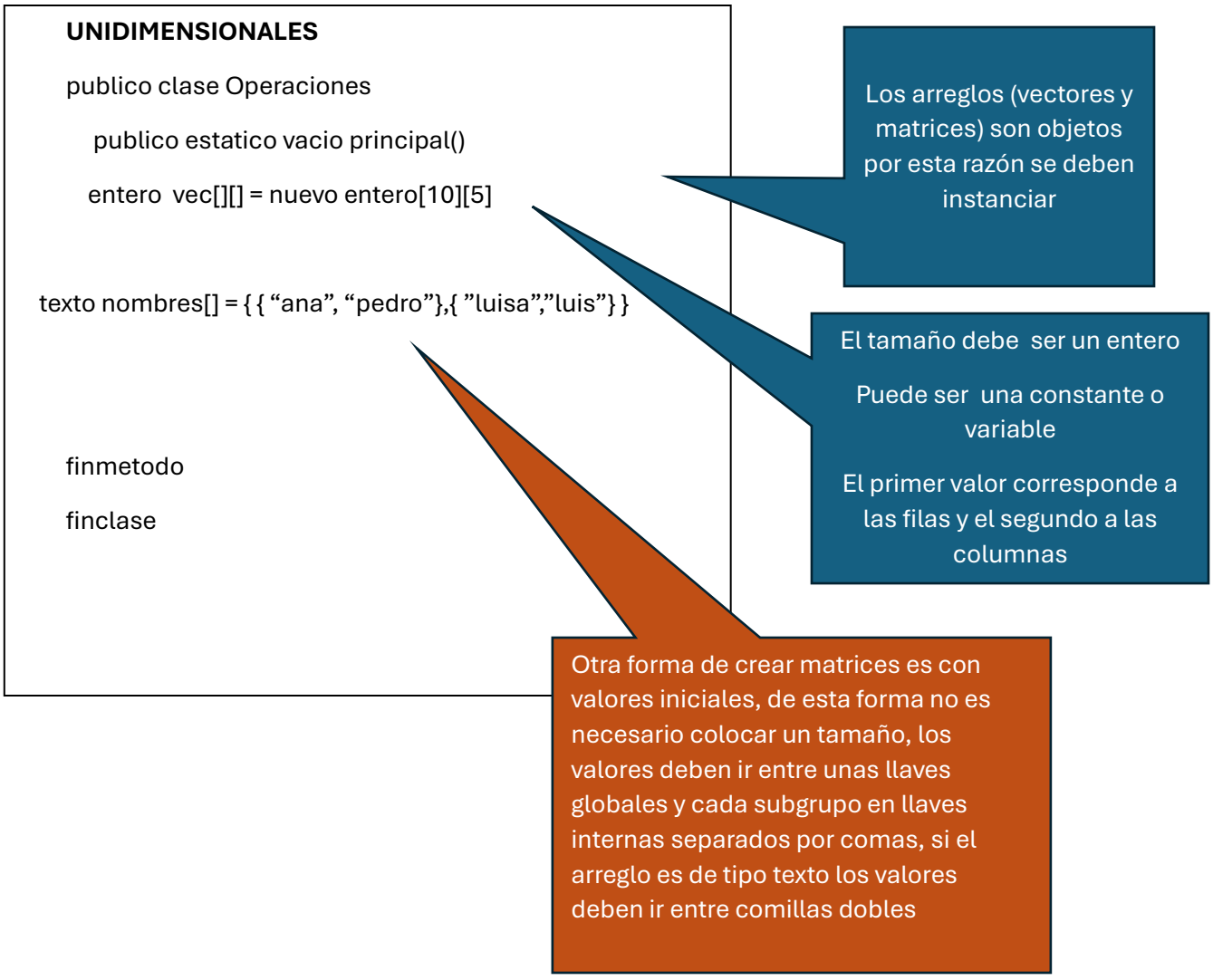
Para recorrer un vector se debe utilizar un ciclo que comience en la posición 0, hasta el tamaño del vector menos 1

Otra forma es utilizando el comando longitud, este nos devuelve el tamaño del vector

La sintaxis es longitud(vector)

ARREGLOS

1.BIDIMENSIONALES (MATRICES)



RECORRER UNA MATRIZ

```
publico clase Operaciones
    publico estatico vacio principal()
    entero  edades[][] = nuevo entero[10][5]

texto nombres[] = { { “ana”, “pedro”},{ ”luisa”,”luis”} }

Para(i = 0; i < 10;i=i+1)
   Para(j= 0; j < 5; j=j+1)
      lea edades[i][j]
   finpara
finpara

Para(i = 0; i < longitud(edades) ;i=i+1)
  Para(j= 0; j < longitud(edades[i]); j=j+1
   lea edades[i][j]
finpara
```

Para recorrer un vector se debe utilizar DOS ciclo que comience en la posición 0, hasta el tamaño de las filas y columnas menos 1

Otra forma es utilizando el comando longitud, este nos devuelve el tamaño de la fila de la matriz

La sintaxis es longitud(matriz)

Para recorrer las columnas

Sintaxis longitud(vector[fila])