

Arreglos y recorridos.

Semana 2 día 4

Qué vamos a aprender:

- Utilizar **estructuras contenedoras** de **tamaño fijo** para almacenar una secuencia de valores (simples u objetos)
- Utilizar **estructuras contenedoras** de **tamaño variable** para almacenar una secuencia de objetos
- Utilizar **instrucciones iterativas** para manipular estructuras contenedoras
- Crear una clase completa en java utilizando eclipse

Caso de estudio No. 1: Las notas de un curso

Las notas de un curso

- En el curso hay 12 estudiantes
- De cada estudiante se tiene la nota definitiva (un valor entre 0.0 y 5.0)
- Se quiere construir un programa que permita:
 1. Cambiar la nota de un estudiante
 2. Calcular el promedio del curso
 3. Establecer el número de estudiantes que están por encima de dicho promedio

Interfaz usuario



Requerimientos Funcionales

Nombre	R1 – Cambiar una nota
Resumen	Permite cambiar la nota definitiva de un estudiante del curso
Entradas	
<ol style="list-style-type: none">1. El estudiante a quien se quiere cambiar la nota2. La nueva nota del estudiante	
Resultado	
Se ha asignado al estudiante la nueva nota.	

Requerimientos Funcionales

Nombre	R2 – Calcular el promedio
Resumen	Permite calcular la nota promedio de los estudiantes del curso
Entradas	
Ninguna	
Resultado	
El promedio de las notas de los doce estudiantes del curso	

Requerimientos Funcionales

Nombre	R3 – Calcular el número de estudiantes por encima del promedio
Resumen	Permite saber cuántos estudiantes tienen una nota superior a la nota promedio del curso
Entradas	
Ninguna	
Resultado	
Número de estudiantes con nota mayor al promedio del curso	

Modelo de la Clase

Curso
<pre>double nota1; double nota2; double nota3; double nota4; double nota5; double nota6; double nota7; double nota8; double nota9; double nota10; double nota11; double nota12;</pre>

Modelo de la Clase

Curso
double nota1; double nota2; double nota3; double nota4; double nota5; double nota6; double nota7; double nota8; double nota9; double nota10; double nota11; double nota12;



Curso
double nota1; double nota2; double nota3; double nota4; double nota5; double nota6; double nota7; double nota8; double nota9; double nota10; double nota11; double nota12; double nota13; double nota14; double nota15;



Curso
double nota1; double nota2; double nota3; double nota4; double nota5; double nota6; double nota7; double nota8; double nota9; . . . double nota100;

Solución: Contenedora

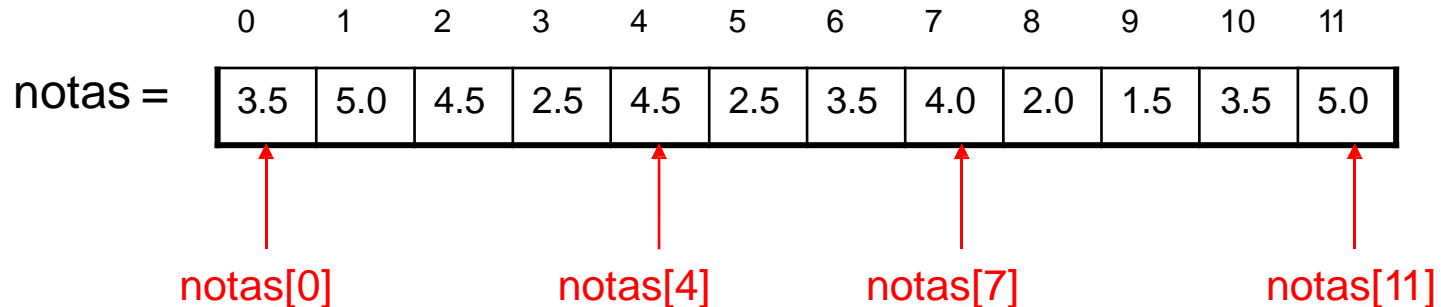
Curso
double nota1;
double nota2;
double nota3;
double nota4;
double nota5;
double nota6;
double nota7;
double nota8;
double nota9;
double nota10;
double nota11;
double nota12;



**UN SOLO
ATRIBUTO
LLAMADO
notas**

Curso	
double notas =	
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11

Arreglo = contenedora de tamaño fijo



- Cada posición del arreglo (casilla) se utiliza como una variable

```
notas[5] = 3.0;
```

```
i = 2;
```

```
if ( notas[ i ] > 4.0)
```

```
    notas[10] = notas[3];
```

```
if (notas[11] == notas[0])
```

```
    notas[0] = 2.0;
```

```
else
```

```
    notas[0] = notas[11];
```

Declaración de un arreglo

```
public class Curso
{
    //Constantes
    public final static int TOTAL_EST = 12;

    //Atributos
    private double[ ] notas;

}
```

Declaración de un arreglo

```
public class Curso
```

```
{
```

```
//Constantes
```

```
final public static int TOTAL_EST = 12; ←
```

Se declara una
constante para fijar el
tamaño del arreglo

```
//Atributos
```

```
private double[ ] notas;
```

```
}
```

↑
TODOS los
elementos del arreglo
son del MISMO TIPO

Inicialización de un arreglo

```
public Curso ( )  
{  
    notas = new double [TOTAL_EST];  
}
```

- El espacio en memoria (una cajita por posición del arreglo) queda reservada.
- El valor de los elementos del arreglo es indefinido al comienzo.
- Para consultar el número de elementos del arreglo: **length**
- Si se trata de acceder a una casilla con índice menor que 0 o mayor que el número máximo de casillas (en este caso 12)

java.lang.ArrayIndexOutOfBoundsException

Para dar valores a los elementos del arreglo

```
public void ponerNotasEnCero ( )  
{  
    notas[0] = 0;  
    notas[1] = 0;  
    notas[2] = 0;  
    notas[3] = 0;  
    notas[4] = 0;  
    notas[5] = 0;  
    notas[6] = 0;  
    notas[7] = 0;  
    notas[8] = 0;  
    notas[9] = 0;  
    notas[10] = 0;  
    notas[11] = 0;  
}
```


Para calcular el promedio de las notas del
curso

```
public double promedio ( )  
{  
    double suma;  
  
    suma = notas[0] + ;  
  
    return suma;  
}
```

Para calcular el promedio de las notas del
curso

```
public double promedio ( )  
{  
    double suma;  
  
    suma = notas[0] + notas[1] + notas[2] + notas[3] + notas[4] +  
           notas[5] + notas[7] + notas[8] + notas[9] + notas[10] + notas[11];  
  
    suma = suma / TOTAL_EST;  
  
    return suma;  
}
```

Para calcular el promedio de las notas del curso

```
public double promedio ( )
{
    double suma = 0.0;
    int indice = 0;

    suma += notas[ indice ];
    indice++;

    suma += notas[ indice ];
    indice++;

    suma += notas[ indice ];
    indice++;

    ...      (9 veces más)

    suma = suma / TOTAL_EST;

    return suma;
}
```

Para calcular el promedio de las

notas del curso

```
public double promedio ( )  
{  
    double suma = 0.0;  
    int indice = 0;  
  
    suma += notas[ indice ];  
    indice++;  
  
    suma += notas[ indice ];  
    indice++;  
  
    suma += notas[ indice ];  
    indice++;  
  
    ...  
  
    suma = suma / TOTAL_EST;  
  
    return suma;  
}
```

}

}

}

(9 veces más)

Paso que se repite

Para calcular el promedio de las

notas del curso

```
public double promedio ( )
{
    double suma = 0.0;
    int indice = 0;

    suma += notas[ indice ];
    indice++;

    suma += notas[ indice ];
    indice++;

    suma += notas[ indice ];
    indice++;

    ...      (9 veces más)

    suma = suma / TOTAL_EST;

    return suma;
}
```

```
public double promedio ( )
{
    double suma = 0.0;
    int indice = 0;

    while ( indice < TOTAL_EST )
    {
        suma += notas[ indice ];
        indice++;
    }

    suma = suma / TOTAL_EST;

    return suma;
}
```

Para calcular el promedio de las notas del curso

```
public double promedio ( )
{
    double suma = 0.0;
    int indice = 0;

    suma += notas[ indice ];
    indice++;

    suma += notas[ indice ];
    indice++;

    suma += notas[ indice ];
    indice++;

    ... (9 veces más)

    suma = suma / TOTAL_EST;

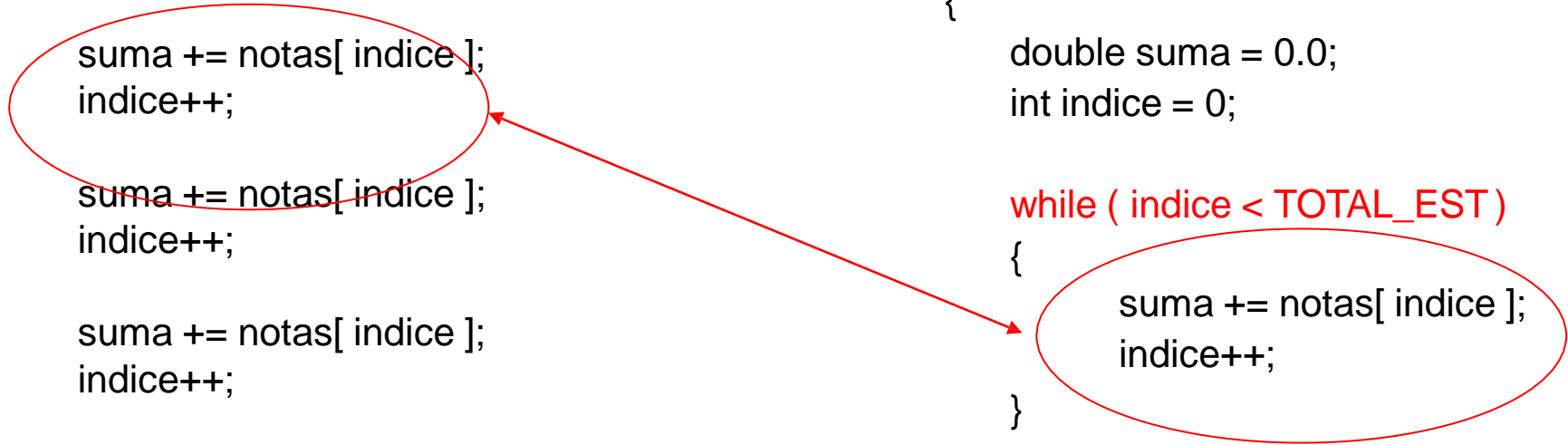
    return suma;
}
```

```
public double promedio ( )
{
    double suma = 0.0;
    int indice = 0;

    while ( indice < TOTAL_EST )
    {
        suma += notas[ indice ];
        indice++;
    }

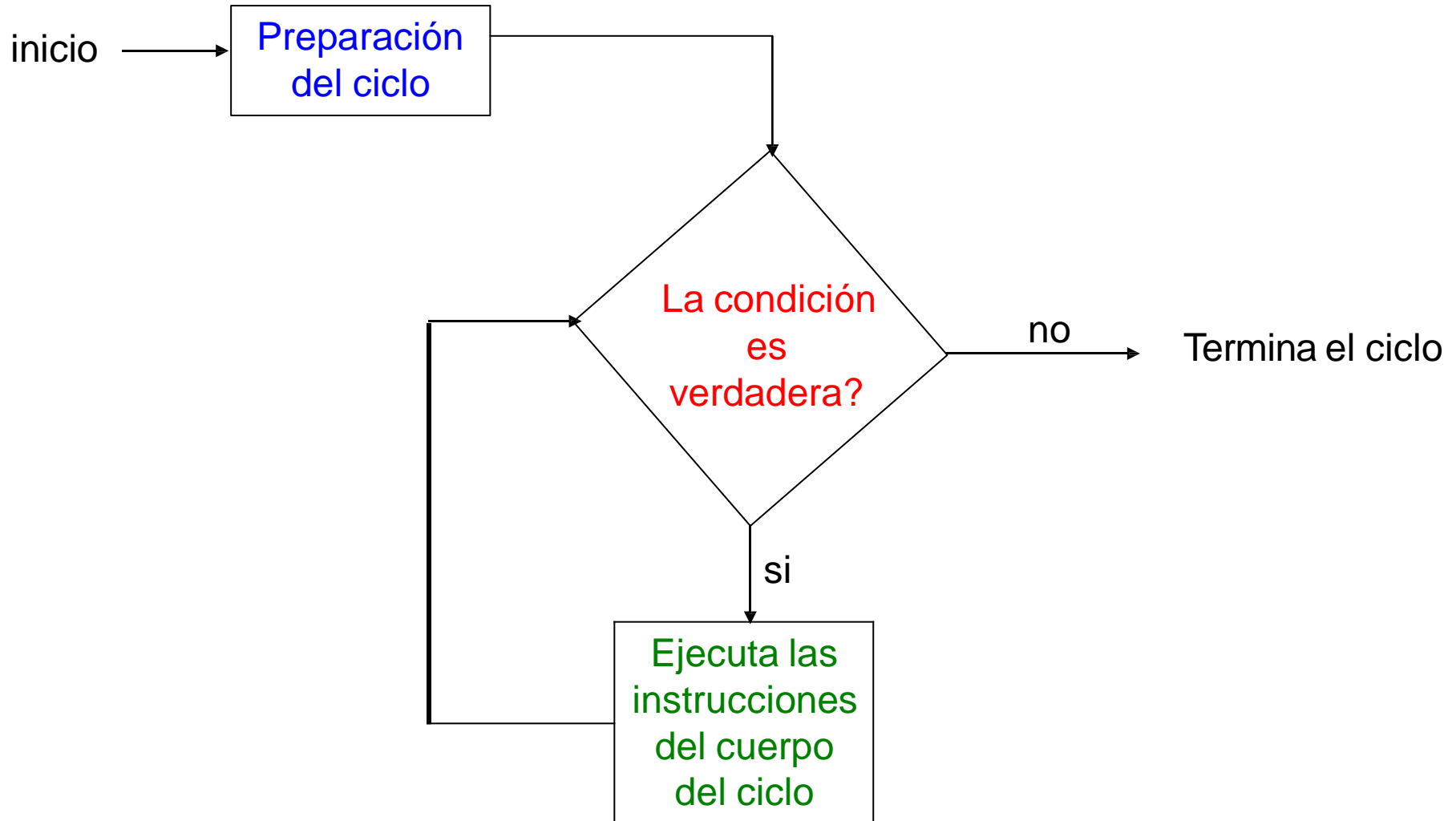
    suma = suma / TOTAL_EST;

    return suma;
}
```



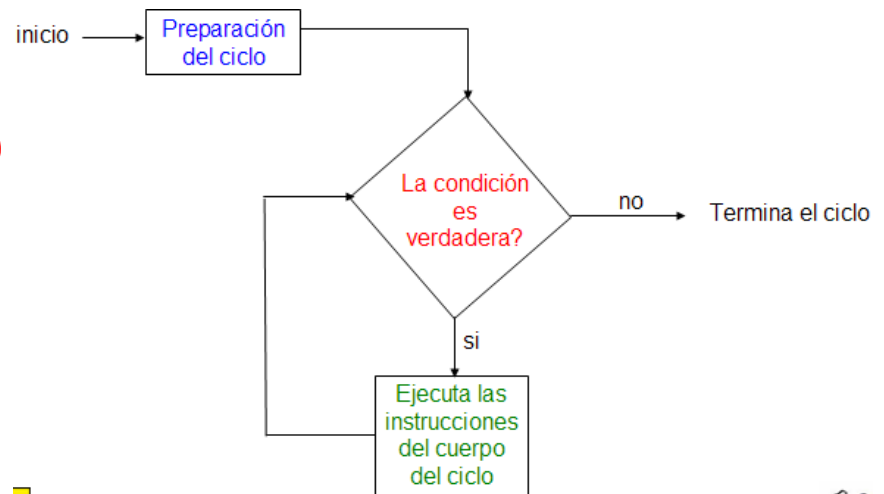
Instrucciones repetitivas

Ejecución de una instrucción repetitiva



Ejecución de una instrucción repetitiva

```
public double promedio ( )  
{  
    double suma = 0.0;  
    int indice = 0;  
  
    while ( indice < TOTAL_EST )  
    {  
        suma += notas[ indice ];  
        indice++;  
    }  
  
    suma = suma / TOTAL_EST;  
    return suma;  
}
```



while y for

<inicio>

while (<condición>)

{

 <cuerpo>

 <avance>

}

for (<inicio>; <condición>; <avance>)

{

 <cuerpo>

}

Para calcular el promedio de las notas del curso

```
public double promedio ( )
```

```
{
```

```
    double suma = 0.0;
```

```
    int indice = 0;
```

Inicio de las variables de trabajo:

- Indice para movernos en el arreglo
- Acumulado de la suma de las notas

```
    while ( indice < TOTAL_EST )
```

```
    {
```

```
        suma += notas[ indice ];
```

```
        indice++;
```

```
    }
```

```
    suma = suma / TOTAL_EST;
```

```
    return suma;
```

```
}
```

Para calcular el promedio de las notas del curso

```
public double promedio ( )
```

```
{
```

```
    double suma = 0.0;
```

```
    int indice = 0;
```

```
    while ( indice < TOTAL_EST )
```

```
    {
```

```
        suma += notas[ indice ];
```

```
        indice++;
```

```
    }
```

```
    suma = suma / TOTAL_EST;
```

```
    return suma;
```

```
}
```

Condición para continuar:

- Cualquier expresión lógica

Para calcular el promedio de las notas del curso

```
public double promedio ( )  
{  
    double suma = 0.0;  
    int indice = 0;  
  
    while ( indice < TOTAL_EST )  
    {  
        suma += notas[ indice ];  
        indice++;  
    }  
  
    suma = suma / TOTAL_EST;  
  
    return suma;  
}
```

Cuerpo del ciclo:

- Instrucciones que se van a repetir en cada iteración

Para calcular el promedio de las notas del curso

```
public double promedio ( )
{
    double suma = 0.0;
    int indice = 0;

    while ( indice < TOTAL_EST )
    {
        suma += notas[ indice ];
        indice++;
    }

    suma = suma / TOTAL_EST;

    return suma;
}
```

Avance del ciclo:

- Incremento del índice que indica la posición del arreglo en la que estamos en un momento dado

Para calcular el promedio de las notas del curso

```
public double promedio ( )  
{
```

```
    double suma = 0.0;
```

```
    for ( int indice = 0; indice < TOTAL_EST; indice ++ )  
    {  
        suma += notas[ indice ];  
    }
```

```
    suma = suma / TOTAL_EST;
```

```
    return suma;  
}
```

Inicio de las variables de trabajo:

- Indice para movernos en el arreglo
- Acumulado de la suma de las notas

Para calcular el promedio de las notas del curso

```
public double promedio ( )
```

```
{  
    double suma = 0.0;  
  
    for ( int indice = 0; indice < TOTAL_EST; indice ++ )  
    {  
        suma += notas[ indice ];  
    }  
  
    suma = suma / TOTAL_EST;  
  
    return suma;  
}
```

Condición para continuar:

- Cualquier expresión lógica

Para calcular el promedio de las notas del curso

```
• public double promedio ( )
• {
    • double suma = 0.0;

    • for ( int indice = 0; indice < TOTAL_EST; indice ++ )
    • {
        }    • suma += notas[ indice ];
    suma = suma / TOTAL_EST;

    return suma;
}
```

Cuerpo del ciclo:
• Instrucciones que se van a repetir en cada iteración

Para calcular el promedio de las notas del curso

```
public double promedio ( )  
{
```

```
    double suma = 0.0;
```

```
    for ( int indice = 0; indice < TOTAL_EST; indice ++ )
```

```
    {
```

```
        suma += notas[ indice ];
```

```
    }
```

```
    suma = suma / TOTAL_EST;
```

```
    return suma;
```

```
}
```

Avance del ciclo:

- Incremento del índice que indica la posición del arreglo en la que estamos en un momento dado

Tarea No. 2 - Calcular el número de estudiantes que sacaron una nota entre 3.0 y 5.0

```
public int cuantosPasaron ( )
```

```
{
```

```
}
```

Tarea No. 2 - Calcular la mayor nota del curso

```
public int mayorNota ( )
```

```
{
```

```
}
```

Tarea No. 2 - Contar el número de estudiantes que sacaron una nota inferior a la del estudiante que está en la posición del arreglo que llega como parámetro. Suponga que el parámetro posEst tiene un valor entre 0 y TOTAL_EST-1

```
public int cuantosPeoresQue ( int posEst )  
{
```

```
}
```

Tarea No. 2 - Aumentar en un 5% todas las notas del curso, sin que ninguna de ellas sobrepase el valor de 5.0

```
public void hacerCurva ( )
```

```
{
```

```
}
```

Tarea No. 4: Calcular el número mínimo de notas del curso necesarias para que la suma supere el valor 30, recorriéndolas desde la posición 0 en adelante. Si al sumar todas las notas no se llega a ese valor, el método debe retornar -1

```
public int sumadasDanTreinta ( )
{
    double sumaNotas = 0;
    int cuantasNotas = 0;
    boolean termino = false;

    for ( int i = 0; i < notas.length && !termino; i++)
    {
        sumaNotas += notas[ i ];
        cuantasNotas++;
        if (sumaNotas >= 30)
            termino = true;
    }
    if (i == notas.length)
        return -1;
    else
        return cuantasNotas;
}
```

Patrones de recorrido de arreglos

Patrón de recorrido total

- Se usa cuando se necesita recorrer **TODO** el arreglo
- Ejemplos:
 - Calcular la suma de TODAS las notas del curso
 - ...
 - ...
 - ...

Patrón de recorrido total


Indice para movernos
en el arreglo empieza
en CERO



```
for ( int indice = 0; indice < arreglo.length; indice++)  
{  
    <cuerpo del ciclo>  
}
```

Patrón de recorrido total


Condición para
continuar: índice menor
que la longitud del
arreglo



```
for ( int indice = 0; indice < arreglo.length; indice++)  
{  
    <cuerpo del ciclo>  
}
```

Patrón de recorrido total

Avance: incremento en
1 del índice



```
for ( int indice = 0; indice < arreglo.length; indice++)  
{  
    <cuerpo del ciclo>  
}
```

Ejemplo

Índice empieza en
CERO

Mientras índice menor
que la longitud del
arreglo

Incremento en 1 del
índice

```
public void hacerCurva ( )  
{  
    for ( int i = 0; i < notas.length; i++)  
    {  
        if ( notas[ i ] < 2.0 )  
            notas[ i ] = notas[ i ] * 1.1;  
    }  
}
```

Patrón de recorrido total con acumulado

- Se usa cuando se necesita recorrer **TODO** el arreglo y además **ACUMULAR** o **CALCULAR** alguna propiedad sobre **TODOS** los elementos.
- Ejemplos:
 - Contar cuántos estudiantes pasaron
 - Calcular el promedio
 - ...
 - ...

Patrón de recorrido total con acumulado

- Decisiones a tomar:
 - Cómo acumular?
 - Cómo inicializar el acumulado?
 - Condición para cambiar el acumulado?
 - Cómo cambiar el acumulado?

Ejemplo

```
public int cuantosPasaron ( )
```

```
{
```

```
    int cuantos = 0;
```



Cómo acumular?
R/ Con una variable

```
    for ( int i = 0; i < notas.length; i++)
```

```
    {
```

```
        if ( notas[ i ] >= 3.0 )
```

```
            cuantos++;
```

```
    }
```

```
    return cuantos;
```

```
}
```


Ejemplo

```
public int cuantosPasaron ( )  
{  
    int cuantos = 0;  
    for ( int i = 0; i < notas.length; i++)  
    {  
        if ( notas[ i ] >= 3.0 )  
            cuantos++;  
    }  
    return cuantos;  
}
```

Cómo inicializar el
acumulado?

R/ En cero o en un
valor adaptado al
problema

Ejemplo

```
public int cuantosPasaron ( )  
{  
    int cuantos = 0;  
    for ( int i = 0; i < notas.length; i++)  
    {  
        if ( notas[ i ] >= 3.0 )  
            cuantos++;  
    }  
    return cuantos;  
}
```

Condición para
cambiar el acumulado?

R/ En este ejemplo,
cuando el elemento
que se está revisando
sea mayor o
igual a 3

Ejemplo

```
public int cuantosPasaron ( )  
{  
    int cuantos = 0;  
    for ( int i = 0; i < notas.length; i++)  
    {  
        if ( notas[ i ] >= 3.0 )  
            cuantos++;  
    }  
    return cuantos;  
}
```

Cómo modificar el
acumulado?

R/ En este ejemplo,
incrementándolo en 1

Tarea No. 3: Modificar las notas de los estudiantes de la siguiente manera: a todos los que obtuvieron mas de 4 les quita 0.5. A todos los que obtuvieron menos de 2.0, les aumenta 0.5. A todos los demás, les deja la nota sin modificar

```
public void cambiarNotas ( )  
{
```

```
}
```

Tarea No. 3: Retornar la menor nota del curso

```
public double menorNota ( )  
{
```

```
}
```

Tarea No. 3: Indicar en qué rango hay mas notas en el curso: rango 1 de 0.0 a 1.99, rango 2 de 2.0 a 3.49, rango 3 de 3.5 a 5.0

```
public int rangoConMasNotas ( )  
{
```

```
}
```

Patrón de recorrido parcial

- Se usa cuando **NO** se necesita recorrer **TODO** el arreglo
- Existe una condición que debemos verificar en cada iteración para saber si debemos detener el ciclo o volver a repetirlo
- Ejemplos:
 - Decidir si al menos un estudiante sacó 5.0
 - Buscar el primer estudiante con nota igual a cero
 - ...
 - ...

Patrón de recorrido parcial

Indice para movernos
en el arreglo empieza
en CERO



```
for ( int indice = 0; indice < arreglo.length && !<condicion>; indice++)  
{  
    <cuerpo del ciclo>  
}
```


Patrón de recorrido parcial

Condición para continuar:

- índice menor que la longitud del arreglo
Y (&&)
- no se cumple una condición



```
for ( int indice = 0; indice < arreglo.length && !<condicion>; indice++)  
{  
    <cuerpo del ciclo>  
}
```

Patrón de recorrido parcial

Avance: incremento en
1 del índice



```
for ( int indice = 0; indice < arreglo.length && !<condicion>; indice++)  
{  
    <cuerpo del ciclo>  
}
```

Patrón de recorrido parcial

A veces la condición
puede ser difícil de
calcular

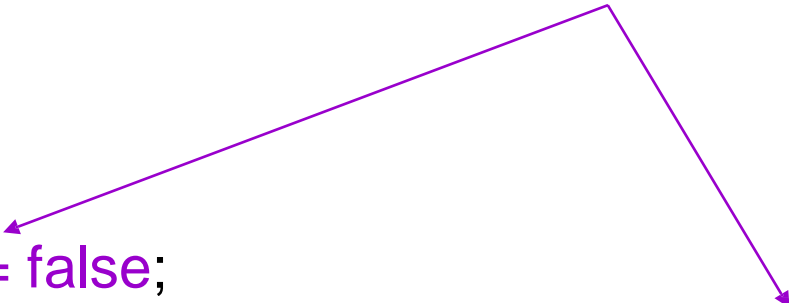


```
for ( int indice = 0; indice < arreglo.length && !<condicion>; indice++)  
{  
    <cuerpo del ciclo>  
}
```

SOLUCION ...

Patrón de recorrido parcial

... Calcular la condición dentro del ciclo
y utilizar una variable auxiliar (centinela)



```
boolean termino = false;  
for ( int indice = 0; indice < arreglo.length && !termino; indice++)  
{  
    <cuerpo del ciclo>  
    if ( <ya se cumplió el objetivo> )  
        termino = true;  
}
```

Ejemplo

```
public boolean alguienConCinco ( )  
{  
    boolean termino = false;  
    for ( int i = 0; i < notas.length && !termino; i++)  
    {  
        if ( notas[ i ] == 5.0 )  
            termino = true;  
    }  
    return termino;  
}
```

	0	1	2	3	4	5	6	7	8	9	10	11
notas =	3.5	4.5	3.5	5.0	4.5	2.5	3.5	4.0	2.0	1.5	3.5	5.0

Otra posibilidad: con while y sin centinela

```
public boolean alguienConCinco ( )
{
    int i = 0;
    while ( i < notas.length && notas[ i ] != 5.0 )
    {
        i++;
    }
    if ( i == notas.length)
        return false;
    else
        return true;
}
```

	0	1	2	3	4	5	6	7	8	9	10	11
notas =	3.5	4.5	3.5	5.0	4.5	2.5	3.5	4.0	2.0	1.5	3.5	5.0

Otra posibilidad: con while y sin centinela

```
public boolean alguienConCinco ( )
{
    int i = 0;
    while ( i < notas.length && notas[ i ] != 5.0 )
    {
        i++;
    }
    return ( i < notas.length );
}
```

	0	1	2	3	4	5	6	7	8	9	10	11
notas =	3.5	4.5	3.5	5.0	4.5	2.5	3.5	4.0	2.0	1.5	3.5	5.0

Patrón de recorrido parcial con acumulado

- Se usa cuando **NO** se necesita recorrer **TODO** el arreglo, sino hasta que se cumpla una condición y además
- Se necesita **ACUMULAR** o **CALCULAR** alguna propiedad sobre los elementos.

Ejemplo con acumulado

Encontrar las primeras TRES notas de 1.5 y asignarles 2.5.

```
public void subirTresNotas( )
```

```
{
```

```
    int numNotas = 0;
```

```
    boolean termino = false;
```

Una variable para
contar cuántas notas de
1.5 se han subido a 2.5

```
    for ( int i = 0; i < notas.length && !termino; i++)
```

```
    {
```

```
        if ( notas[ i ] == 1.5 )
```

```
        {
```

```
            numNotas++;
```

```
            notas[ i ] = 2.5;
```

```
        }
```

```
        if ( numNotas == 3 )
```

```
            termino = true;
```

```
    }
```

```
}
```

Ejemplo con acumulado

Encontrar las primeras TRES notas de 1.5 y asignarles 2.5.

```
public void subirTresNotas( )
```

```
{
```

```
    int numNotas = 0;
```

```
    boolean termino = false;
```

Una variable centinela
para parar el ciclo
cuando ya se hayan
subido 3 notas

```
    for ( int i = 0; i < notas.length && !termino; i++)
```

```
    {
```

```
        if ( notas[ i ] == 1.5 )
```

```
        {
```

```
            numNotas++;
```

```
            notas[ i ] = 2.5;
```

```
        }
```

```
        if ( numNotas == 3 )
```

```
            termino = true;
```

```
    }
```

```
}
```

Ejemplo con acumulado

Encontrar las primeras TRES notas de 1.5 y asignarles 2.5.

```
public void subirTresNotas( )
```

```
{
```

```
    int numNotas = 0;
```

```
    boolean termino = false;
```

```
    for ( int i = 0; i < notas.length && !termino; i++)
```

```
    {
```

```
        if ( notas[ i ] == 1.5 )
```

```
        {
```

```
            numNotas++;
```

```
            notas[ i ] = 2.5;
```

```
        }
```


```
        if ( numNotas == 3 )
```

```
            termino = true;
```

```
    }
```

```
}
```

Ciclo desde la primera posición del arreglo, mientras NO haya llegado al final y NO se haya cumplido la condición del centinela



Ejemplo con acumulado

Encontrar las primeras TRES notas de 1.5 y asignarles 2.5.

```
public void subirTresNotas( )
```

```
{
```

```
    int numNotas = 0;
```

```
    boolean termino = false;
```

```
    for ( int i = 0; i < notas.length && !termino; i++)
```

```
    {
```

```
        if ( notas[ i ] == 1.5 )
```

```
        {
```

```
            numNotas++;
```

```
            notas[ i ] = 2.5;
```

```
        }
```


```
        if ( numNotas == 3 )
```

```
            termino = true;
```

```
    }
```

```
}
```

Si la nota actual (de la posición i) es 1.5, subirla a 2.5 e incrementar el contador de notas




Ejemplo con acumulado

Encontrar las primeras TRES notas de 1.5 y asignarles 2.5.

```
public void subirTresNotas( )
{
    int numNotas = 0;
    boolean termino = false;

    for ( int i = 0; i < notas.length && !termino; i++)
    {
        if ( notas[ i ] == 1.5 )
        {
            numNotas++;
            notas[ i ] = 2.5;
        }
        if ( numNotas == 3 )
            termino = true;
    }
}
```

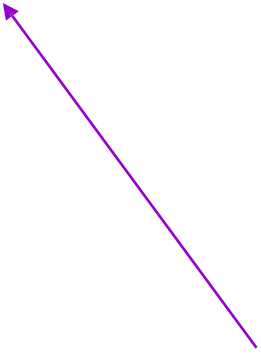
Si ya se han subido
TRES notas de 1.5,
entonces poner el
centinela en true para
acabar el ciclo



Otra posibilidad sin centinela

Encontrar las primeras TRES notas de 1.5 y asignarles 2.5.

```
public void subirTresNotas( )  
{  
    int numNotas = 0;  
  
    for ( int i = 0; i < notas.length && numNotas < 3; i++)  
    {  
        if ( notas[ i ] == 1.5 )  
        {  
            numNotas++;  
            notas[ i ] = 2.5;  
        }  
    }  
}
```



Se pone la condición de
continuación
directamente en el ciclo

Tarea No. 3: Reemplazar todas las notas del curso por 0.0 hasta que aparezca la primera nota superior a 3.0

```
public void notasACero ( )  
{
```

```
}
```

Ejemplo con acumulado

Método que retorna la posición en el arreglo de la tercera nota con valor 5.0

```
public int tercerCinco( )
```

```
{
```

```
• int cuantosCincos = 0; int posición = 0; boolean
```

Dos variables para
acumular la información

```
termino = false;
```

```
• for ( int i = 0; i < notas.length && !termino; i++)
```

```
• {
```

```
• if ( notas[ i ] == 5.0 )
```

```
• cuantosCincos++;
```

```
• if ( cuantosCincos == 3 )
```

```
• {
```

```
• termino = true; posicion = i;
```

```
• }
```

```
• }
```

```
} • return posicion;
```


Ejemplo con acumulado

Método que retorna la posición en el arreglo de la tercera nota con valor 5.0

```
public int tercerCinco( )
{
    int cuantosCincos = 0;
    • int posición = 0; boolean termino = false;
    • for ( int i = 0; i < notas.length && !termino; i++)
    • {
        • if ( notas[ i ] == 5.0 )
            • cuantosCincos++;
        • if ( cuantosCincos == 3 )
        • {
            • termino = true; posicion = i;
        • }
    • }
} • return posicion;
```

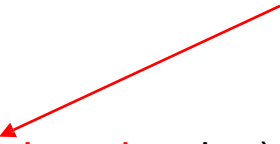
Variable centinela del ciclo

Ejemplo con acumulado

Método que retorna la posición en el arreglo de la tercera nota con valor 5.0

```
public int tercerCinco( )
{
    int cuantosCincos = 0;
    int posición = 0;
    boolean termino = false;
    for ( int i = 0; i < notas.length && !termino; i++)
    {
        if ( notas[ i ] == 5.0 )
            cuantosCincos++;
        if ( cuantosCincos == 3 )
        {
            termino = true;
            posicion = i;
        }
    }
    return posicion;
}
```

Condición con el centinela



Ejemplo con acumulado

Método que retorna la posición en el arreglo de la tercera nota con valor 5.0

```
public int tercerCinco( )
{
    int cuantosCincos = 0;
    int posición = 0;
    boolean termino = false;
    for ( int i = 0; i < notas.length && !termino; i++)
    {
        if ( notas[ i ] == 5.0 ) ← Condición para cambiar el acumulado
            cuantosCincos++;
        if ( cuantosCincos == 3 )
        {
            termino = true;
            posicion = i;
        }
    }
    return posicion;
}
```

Ejemplo con acumulado

Método que retorna la posición en el arreglo de la tercera nota con valor 5.0

```
public int tercerCinco( )
{
    int cuantosCincos = 0;
    int posición = 0;
    boolean termino = false;
    for ( int i = 0; i < notas.length && !termino; i++)
    {
        if ( notas[ i ] == 5.0 )
            cuantosCincos++;
        if ( cuantosCincos == 3 )
        {
            termino = true;
            posicion = i;
        }
    }
    return posicion;
}
```

Condición para cambiar el centinela y salir del ciclo

	0	1	2	3	4	5	6	7	8	9	10	11
notas =	5.0	4.5	3.5	5.0	4.5	5.0	3.5	4.0	2.0	1.5	3.5	5.0

Tarea No. 4: Reemplazar todas las notas del curso por 0.0 hasta que aparezca la primera nota superior a 3.0

```
public void notasACero ( )  
{
```

```
}
```

Tarea No. 4: Calcular el número mínimo de notas del curso necesarias para que la suma supere el valor 30, recorriéndolas desde la posición 0 en adelante. Si al sumar todas las notas no se llega a ese valor, el método debe retornar -1

```
public int sumadasDanTreinta ( )  
{
```

```
}
```

Sol No. 4: Calcular el número mínimo de notas del curso necesarias para que la suma supere el valor 30, recorriéndolas desde la posición 0 en adelante. Si al sumar todas las notas no se llega a ese valor, el método debe retornar -1

```
public int sumadasDanTreinta ( )
{
    double sumaNotas = 0;
    int cuantasNotas = 0;
    boolean termino = false;

    for ( int i = 0; i < notas.length && !termino; i++)
    {
        sumaNotas += notas[ i ];
        cuantasNotas++;
        if (sumaNotas >= 30)
            termino = true;
    }
    if (i == notas.length)
        return -1;
    else
        return cuantasNotas;
}
```