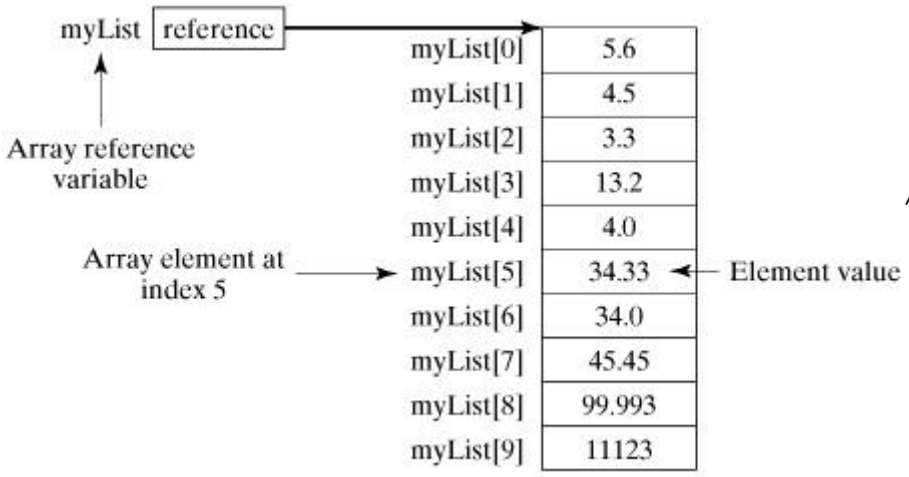
UF4.NF1.p5.Arrays

Un array es un medio de guardar un conjunto de objetos de la misma clase. Se accede a cada elemento individual del array mediante un número entero denominado índice. 0 es el índice del primer elemento y n-1 es el índice del último elemento, siendo n, la dimensión del array.



Array de 10 elements

Los arrays son objetos en Java y como tales vamos a ver los pasos que hemos de seguir para usarlos convenientemente

- -Declarar el array
- -Crear el array
- -Inicializar los elementos del array
- -Usar el array

Declarar y crear un array.(declarar la variable y crear el array)

1.-Para declarar un array se escribe

```
tipo_de_dato[] nombre_del_array;
```

Para declarar un array de enteros escribimos

```
int[] numeros;
```

Para crear un array de 4 número enteros escribimos

```
numeros=new int[4];
```

La declaración y la creación del array se puede hacer en una misma línea.

```
int[] numeros =new int[4];
```

Inicializar y usar los elementos del array

Para inicializar el array de 4 enteros escribimos:

```
numeros[0]=2;
numeros[1]=-4;
numeros[2]=15;
numeros[3]=-25;
```

También se puede construir mediante asignación, **sin necesidad de new**

```
int datos={2,-3,0,7}

String[] nombres={"Juan", "José", "Miguel", "Antonio"};
```

Sólo en la declaración se pueden utilizar los {}. Por ejemplo

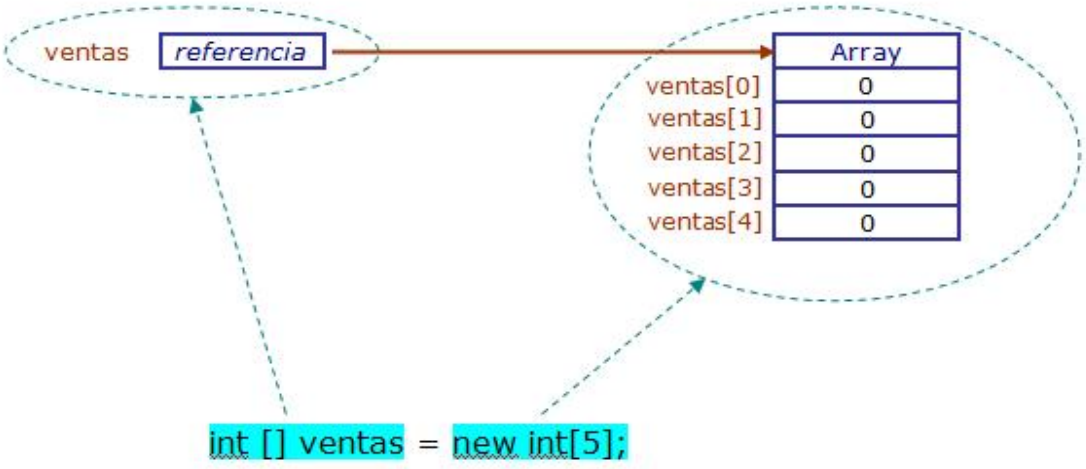
```
int datos[];

datos={2,-3,0,7};     ->>>>>>>>>>>>>>>>>> ERROR
```

Se pueden inicializar en un bucle **for** como resultado de alguna operación

```
for(int i=0; i<4; i++){
     numeros[i]=i*i+4;
}
```

Los **int** se inicializan a **0**, los booleanos a **false.** El resto de los elementos se inicializan a **null.**



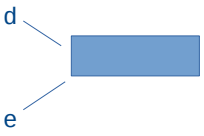
Si hiciéramos:

```
System.out.println(ventas);// mostraría la referencia que guarda la variable ventas, que apunta a la posición de memoria donde está el array. Algo similar a I@65efd
```

Sería posible acceder a los elementos de una tabla mediante más de una variable.

Por ejemplo:

```
int d[],e[];
d=new int[6];              //Tanto d como e apunta a la misma tabla
e=d;
```



Recolector de basura-NULL

Para que no queden espacios de memorias sin utilizar por arrays que ya no son útiles, periódicamente Java inicia un proceso llamado **recolector de basura** que comprueba todas las tablas construidas. Si encuentra alguna inaccesible -sin variable de referencia- la destruye, dejando libre el espacio que estaba ocupando en memoria.

Es importante que cuando una tabla ya no la utilicemos la igualamos a **null**.

Cantidad de elementos de un array

No se puede confundir el valor *0* dentro de un array con el *0* con el que se inicializa un array al denifirlo como **int**, para ello los elementos del array se deben añadir al principio y no dejar huecos.

Cuando la tabla está incompleta es necesario tener una variable que nos indique la cantidad de elementos de que dispone el array, no la longitud total sino los elementos de que dispone.

Esta variable aumentará cuando añadimos un elemento y disminuirá cuando eliminemos un valor del array.

Clase Arrays

La API de Java proporciona esta herramienta al programador para trabajar con arrays.

Para utilizarla debemos importarla, se encuentra en el paquete java.util.

```
import java.util.Arrays
```

Operaciones con tablas:

1.-Obtención del tamaño

```
nombreVariable.length
```

ejemplo:

```
int notas[]=new int[10];
system.out.println("tamaño notas"+notas.length);
```

2.-Recorrido

Ejemplo:Para recorrer los elementos del array **nombres** se escribe

```
for(int i=0; i<nombres.length; i++){
     System.out.println(nombres[i]);
}
```

Ejemplo: queremos incrementar un 10% todos los sueldos de los empleados del array **sueldos**:

```
for(int i=0; i<sueldos.length; i++){
     sueldos[i] = sueldos[i] + 0,1*sueldos[i];
}
```

En C/C++, cuando se intenta acceder y almacenar datos en un elemento del array que está fuera de límites, siempre se consigue, aunque los datos se almacenen en una zona de memoria que no forme parte del array. En Java, si se intenta hacer esto, el Sistema generará una excepción, tal como se muestra en el ejemplo. En él, la excepción simplemente hace que el programa termine, pero se podría recoger esa excepción e implementar un controlador de excepciones (*exception*

handler) para corregir automáticamente el error, sin necesidad de abortar la ejecución del programa.

Ejercicio 1: Hacer un programa donde definas un array de 4 enteros y al insertar un elemento fuera de rango dé un error del tipo: java.lang.ArrayIndexOutOfBoundsException

Recorrido for-each

ejemplo:

```
double sueldos[]=new double[5]
double total=0;
for (double s:sueldos){
      total+=s;
}
```

3.-Mostrar una tabla

Para poder mostrar todos los elementos de una tabla, podemos utilizar la función estática toString() de la clase Arrays.

ejemplo:

```
int t[]={8,41,37,22,19}
System.out.println(Arrays.toString(t));
```

Mostraría los valores de la tabla entre []->[8,41,37,22,19]

4.-Inicialización(método fill)

Ya hemos visto al principio de la actividad cómo se inicializan por defecto los arrays:

0 -int

false-boolean

null -el resto

Si queremos inicializarlos de otra manera habría que recorrer la tabla elemento a elemento.

La clase Arrays dispone de un método fill.

Ejemplo:

```
Arrays.fill(sueldos, 1234,56)//inicializa cada elemento del array sueldos a 1234,56
```

Si sólo queremos inicializar un rango de valores:

```
Arrays.fill(sueldos, 3,7,1234,56)//sólo inicializa los elementos entre rango 3..7 sueldos a 1234,56
```

5.-Comparación de dos tablas(método equals)

Comparar 2 tablas con el operador == sólo compararía las sus referencias.

Para comparar dos tablas utilizamos el método **equals**

Dos tablas son iguales si contienen los mismos elementos y en el mismo orden.

Por ejemplo:

```
int a[]={8,41,37,22,19}
int b[]={8,41,37,22,19}

System.out.println(Arrays.equals(a,b)); //muestra true
```

6.-Búsqueda(método)

Consiste en averiguar si un elemento se encuentra en una tabla y en qué posición.

Búsqueda en una tabla desordenada

Habría que recorrer el array y comparar el elemento buscado con los elementos del array

Ejercicio 2: Hacer un programa en el definas un array de enteros desordenados y un dato a buscar. Recorre el array buscando el dato y determina su posición o que imprima -1 en caso de que no se encuentre.

Búsqueda en una tabla ordenada(método binarySearch)

La clase Arrays dispone de un método **binarySearch** que realiza la búsqueda dicotómica de un elemento, siempre que la tabla esté ordenada. Devuelve el índice donde se ha encontrado la primera ocurrencia del elemento o un valor negativo en caso de que no se encuentre.

Ejemplo: deseamos buscar en la tabla ordenada precios, si existe y en qué posición está, algún producto de 19,95 euros.

```
int pos=Arrays.binarySearch(precios, 19,95);

if (pos>=0){

      System.out.println("encontrado en el indice"+ pos);

} else{

      System.out.println("no se ha encontrado")

}
```

Cuando el elemento a buscar no se encuentra, el valor negativo devuelto tiene un significado especial:informa de la posición donde tendría que colocarse ele elemento buscado para que la tabla continúe ordenada.

```
Indiceinsercion=-(pos -1)
```

ejemplo:

```
int a[]={2,4,5,8,9}

int pos=Arrays.binarySearch(a,7);// pos vale -4

int indiceInsercion=-(pos +1) //se insertaría en la valor de la
                                posición 3
```

Si se quiere sólo buscar un dato entre dos valores:

```
Arrays.binarySearch(tipo t[], int desde, int hasta, tipo aBuscar)

Arrays.binarySearch(a, 2,4,5)// busca el dato 5, desde el elemento 2 hasta el 4
```

6.-Inserción.

Añadir un elemento en un array dependerá si está ordenado o no

Inserción en array no ordenado

Si no está ordenado el array se añadirá al final de los elementos, siempre y cuando no esté llena el array y aumentará en 1 el valor de num_elem(número de elementos)

Ejercicio 3: Hacer un programa en el insertes un elemento en un array, aumentando el num_elem cada vez que se añada un elemento y controle si ha llegado al final, si es así se ha de indicar.

Inserción en array ordenado

Primero hay que encontrar la posición del elemento a buscar. Se han de desplazar los valores una posición comenzando por el final, desde el último índice utilizado. De esta forma, la tabla continua ordenada.

Ejercicio 4: Realizar un programa el que se quiera insertar un elemento en un array ordenado de manera que una vez insertado, mantenga el orden.

7.-Borrado.

El paso previo a eliminar un elemento de una tabla, es buscarlo. Una vez localizado, dependerá si la tabla está ordenada o no.

Borrado en array no ordenada

Una vez localizado lo sustituimos por el último elemento, y disminuimos en 1 el num_elem.

Ejercicio 5: Realizar un programa el que se quiera borrar un elemento en un array desordenado.

Borrado en array ordenado

Primero hay que buscarlo y una vez localizado en la posición **pos**, tenemos que desplazar los valores que le siguen a **pos** hacia la derecha y disminuimos en 1 el **num_elem.**

Ejercicio 6: Realizar un programa el que se quiera borrar un elemento en un array ordenado, manteniendo su orden.

8.-Ordenación (método sort)

El método sort ordena el array de forma creciente.

Ejemplo:

```
int edad={81,19,23,3,7}

Arrays.sort(edad);//Ahora edad={3,7,19,23,81}
```

Ordenar una tabla tiene sentido cuando vayamos a realizar muchas búsqueda ya que resulta muy costoso en tiempo y ralentizará el programa.

9.-Copia (método copy/copyOf)

Arrays dispone del método **copy.**

Ejemplo:

```
int a={8,19,3,3,7}
int b[], c[];
b=Arrays.copy(a, 3);//Ahora b=[8,19,3]
c=Arrays.copy(a, 10);//Ahora c=[8,19,3,3,7,0,0,0,0,0]
```

Si se quiere sólo copiar un array entre un rango de valores, se utiliza el método **copyOf()**:

Ejemplo:

```
int a={8,19,3,3,7}
int b[];
b=Arrays.copyOf(a, 1,3);//Ahora b=[19,3,3]
```

Ejercicio 7: Realizar un programa el que se solicite al usuario cuántos número desea introducir . A continuación, se introducirán esos número y después los mostrará en orden inverso.

Ejercicio 8: Realizar un programa para gestionar un campeonato de programación, donde participan un máximos de 10 programadores. Se introducen la puntación (enteros) obtenidos por los participantes, conforme van terminando su prueba. La aplicación debe mostrar las puntuaciones ordenadas de los participantes, que van acabando la prueba. La forma de especificar que no han acabado la prueba más programadores es introducir como puntuación un -1. La aplicación debe mostrar, finalmente, los puntos ordenado de todos los participantes.