Apunte: Arreglos

INTRODUCCION

Definición: colección de valores de datos, todos del mismo tipo.

Cada uno de los valores en el arreglo se puede identificar unívocamente mediante subíndices. Los elementos se diferencian entre sí por el número de índice. Cada celda de este tipo especial de variable es numerada del *0 a n-1* , donde *n* es el número de celdas del arreglo (su capacidad o tamaño).

Adecuados para trabajar con bucles, especialmente la estructura PARA. ..HASTA...HACER

ARREGLO UNIDIMENSIONAL

Cuando la colección de valores contenidos en un arreglo están enumerados en una secuencia de valores, se considera a esta colección de elementos como pertenecientes a un arreglo unidimensional, para distinguirlos de aquellos en dos dimensiones (por ej. una matriz). Para acceder a un elemento del arreglo se utiliza un índice que determina la posición requerida.

Ejemplo: un arreglo llamado nota de enteros de 4 elementos.

97	86	92	71

- `nota[0]` tiene el valor 97
- `nota[1]` tiene el valor 86
- `nota[2]` tiene el valor 92
- `nota[3]` tiene el valor 71

COMPONENTES DEL ARREGLO

Analicemos los elementos de un arreglo temperatura cuando accedemos a uno de sus componentes:

```
temperatura [n+2] = 32;
```

Nombre del arreglo Índice (O subíndice). Debe ser un int o una expresión que evalúe como int.

Note que elemento puede referirse a una sola variable indexada en el arreglo o al valor de una sola variable indexada

1. EL USO DE CORCHETES

Hay tres usos de los corchetes, para declarar, para crear y para acceder:

Ejemplo:

```
     int[ ] presion ; //Para declarar el tipo del arreglo.
     presion = new int [10] ; //Para crear el arreglo.

//de otra manera
      presion2[3]= TecladoIn.ReadLineInt(); //Para acceder a un elemento.
      System.out.println ("Usted ingreso:"+presion2[3] );


//de otra manera
      int[] otroArreglo = {3, 5, 7} ;//inicialización en la declaración
```

A continuación se muestra la declaración de un arreglo de números enteros (int):

Ejemplo:

```
int[ ] arrNumeros ; // Declaración de un arreglo de enteros
```

Java no deja indicar el tamaño de un arreglo vacío cuando se lo declara. La asignación de memoria se realiza de forma explícita en algún momento del programa cuando se crea el mismo. Para ello se utiliza el operador NEW:

Ejemplo:

```
int[] arrNumeros = new int[5]; //crea el arreglo para 5 números
```

O se asigna una lista de elementos en forma explícita lo cual crea el espacio de memoria y de en el tamaño del arreglo a la vez.

Ejemplo:

```
int[] arrNumeros ={2,5,8} // el arreglos se define con un tamaño de 3
```

Si se utiliza la forma de *new* se establecerá el valor 0 a cada uno de los elementos del arreglo. La longitud del arreglo se determina por los elementos.

Consideremos ahora la sintaxis para declarar y crear un arreglo:

```
tipoBase[ ] nombreArr = new tipoBase [longitud] ;
```

o por separado (primero la declaración y luego la creación):

```
tipoBase[ ] nombreArr ;
nombreArr =new tipoBase [longitud] ;
```

2. INDICES

- Los subíndices de un arreglo siempre comienzan en 0.
- El primer elemento tiene el subíndice 0.
- El segundo elemento tiene el subíndice 1.
- El enésimo elemento tiene el subíndice n-1
- El último elemento tiene el subíndice longitud del arreglo -1

Un arreglo int con 4 elementos

Subíndice	0	1	2	3
Valor	97	86	92	71

3. LONGITUD DE UN ARREGLO

El atributo length se establece en el momento de la creación.

```
int[ ] presion ; //todavía no se sabe la longitud
int longitud;

System.out.println ("De qué longitud desea el arreglo?");
longitud = TecladoIn.ReadLineInt();
presion = new int[longitud]; //se crea con esa longitud
```

No se puede cambiar a menos que el arreglo sea creado nuevamente.

Se puede consultar mediante la variable de instancia *length*. `//indica la longitud`

```
System.out.println ("La longitud es de ", presion.length);
```

4. INDICE FUERA DE RANGO

Si se utiliza un subíndice menor que 0 o mayor que *length-1* se produce un error en tiempo de ejecución: *ArrayOutOfBoundsException.* Se debe encontrar el problema, corregirlo y recompilar el código.

5. IDENTIFICADORES PARA ARREGLOS

Utilizar identificadores para arreglos en singular en lugar de plural mejora la legibilidad del código.

Aunque el arreglo contiene muchos elementos, el uso más común del mismo será por un subíndice que hace referencia a un valor único.

6. INICIALIZANDO UN ARREGLO EN TIEMPO DE EJECUCIÓN

Generalmente se utiliza un bucle for

```
int i ; // contador del bucle /índice del arreglo
int [ ] a = new int [10] ;

for (i=0; i<a.length ; i++){
            //en este caso no son necesarias las llaves
      a[i]=0;
      }
```

7. ARREGLOS COMO ARGUMENTOS DE UN MÉTODO

Un método puede retornar el valor de un elemento del arreglo o de un arreglo completo.

Cuando se pasa el arreglo completo como argumento, en realidad se pasa la referencia al arreglo, por lo cual toda modificación queda reflejada al finalizar el módulo.

Ejemplo: Módulo que inicializa un arreglo en -1. Si bien el arreglo se pasa por parámetro toda modificación de datos que se haga en el módulo se refleja al terminar.

```
public static void cargaArreglo (int [] presion )
{
      for ( i =0, i < presion.length ; i++) //se inicializa en 0,
           presion [i] = 0;
}
```

```
public static void cargaArregloLongitud (int[] presion, int longitud)
{
      for ( i =0, i < longitud ; i++) //se inicializa en 0,
           presion [i] = 0; //finaliza en longitud
}
```

La diferencia es que en el primer caso se utiliza la longitud del arreglo, en el segundo caso se inicializa hasta la variable *longitud*, esa variable debe ser menor o igual a la longitud real del arreglo para que no ocurrra una excepción por índice fuera de rango.
El método tiene acceso al arreglo original y puede cambiar el valor de los elementos.

El uso del atributo *length* (como control de la cantidad de elementos) dentro del método evita el error ***ArrayIndexOutOfBoundsException***
Ejemplo: Mostrar por pantalla todos los elementos de un arreglo del uso del atributo *length*

```
public static void muestraArreglo (char [] arr){
      int i;

      for ( i =0, i < arr.length; i++)
         System.out.println (arr[i]);
      }


char[] letras = new char[45];

                  //arreglo real letras es el formal arr

Miclase.muestraArreglo(letras);
```

En el código anterior, en la línea 1, char[] arr representa el argumento del método, este argumento es un arreglo de caracteres.
En la línea 4 del código anterior, el atributo length se utiliza para controlar el bucle.

Permite diferentes tamaños de arreglos y evita excepciones por índices fuera de rango.

Ejemplo: función para buscar el menor elemento almacenado en el arreglo y retornar su posición

.

```
public static int posMenor (int[] arr)
{
      int i, posi= 0, valorMenor = arr[0];
```

```
        for ( i =1, i < arr.length; i++){
                    //debe recorrer todo el arreglo
            if (arr[i] < valorMenor){
               valorMenor = arr[i];
               posi = i;
              }
            }
         return posi;
    }
```

Ejemplo: función para determinar si todos los elementos del arreglo son positivos o no. Observar que al encontrar el primer elemento no positivo, se termina el algoritmo.

```
    public static boolean todosPositivos (int[] arr){
        int i;
        boolean sigue = true;
        while ( i < arr.length && sigue){
               if (arr[i] >= 0){
                   i++;
                   }
                else
        //al encontrar un valor negativo cambia la variable sigue
                   sigue= false;
          }
        return sigue;
    }
```

Ejemplo: Multiplicar por 5 todos los elementos del arreglo que están en posición par.

```
public static void mult5 (int[] arr)
```

```
    {
    int i;
    for ( i =0, i < arr.length; i+=2) //i+=2 es lo mismo que i=i+2
         arr[i] *= 5 ; // arr[i] =arr[i]*5
    }
```

8. ARREGLO COMO RETORNO DE UN MÉTODO

Un método puede retornar un arreglo. En esos casos el arreglo se crea dentro del método y luego se retorna.

Ejemplo: Generar un arreglo con los elementos negativos.

```
public static int[] creaNegativos (int[] arr){
     int i, j=0;
     int[] aux = new int[arr.length]; //tiene la misma longitud que arr
     int[] retorno; //no se sabe todavía su longitud
     for ( i =0, i < arr.length; i++){ //solo traspasan los negativos
           if (arr[i] < 0){
                nuevo[j] = arr[i];
                j++;
              }
     retorno = new int[j]; //el arreglo de retorno tiene dimensión j
     for ( i =0, i < j; i++){ //copia los elementos del aux al retorno
           retorno [i] = aux[i];

     return retorno; //retorna el arreglo
  }
```

```
    public static void main (char[] argv)
    {
    int i, j=0;
```

```
int[] positivo = new int[100]; //recibe el arreglo creado en el método
int[] positivo = creaNegativos (arreglo);
….
}
```

9. ASIGNACIÓN DE ARREGLOS

```
int[] arr = new int[3];
int[] arr2 = new int[3];
for ( i =0, i < arr.length; i++)
arr[i] = i ;

arr2 = arr; //no copia el arreglo, sino que referencia al mismo
arr2[2] = 5

System.out.println (arr[1] + " " + arr2[1]); //muestra: 1 1
System.out.println (arr[2] + " " + arr2[2]); //muestra: 5 5
```

10. COMPARANDO ARREGLOS

Para comparar arreglos se debe comparar elemento a elemento

```
int i=0;
int[] arr1 = new int[3];
int[] arr2 = new int[3];

for ( i =0, i < arr1.length; i++) //inicializa con el subindice
        arr1[i] = i;
for ( i =0, i < arr2.length; i++) //inicializa con el subindice
        arr2[i] = i;
```

```
if (arr1 == arr2) //si la dirección de memoria donde está el arreglo
                  //es la misma para los dos
      System.out.println(" arr1 es arr2");
else //no compara por el contenido
      System.out.println(" arr1 no es arr2");

     //la salida será: arr1 no es arr2
```

Ejemplo: Método que compara elemento a elemento dos arreglos.

```
public static boolean comparaArreglos (int[] arr1, int[] arr2)
{
      int i=0;
      boolean iguales = true;
      if (arr1.length != arr1.length)
            iguales = false;
      while ( i < arr.length && iguales){
            if (arr[i] == arr2[i] )
            i++;
            else //corta al encontrar el primer valor distinto.
            iguales = false;
      }
      return iguales;
}
```

La siguiente tabla describe un resumen de varios conceptos analizados en este apunte:

	Tipo Primitivo	Arreglo Completo	Elemento del arreglo
Asignación (=)	Copia contenido	Copia dirección	Depende del tipo primitivo/Clase
Igualdad (==)	Compara contenido	Compara dirección	Depende del tipo primitivo/Clase
Pasaje de Parámetros	Pasa copia del valor	Pasa referencia de la dirección	Depende del tipo primitivo/Clase

ARREGLO BIDIMENSIONAL

Definición: es una colección de valores de datos tipo matriz, todos del mismo tipo.

Algunos datos vienen en dos dimensiones como los tableros, planillas de cálculo, matrices, etc.

Cada uno de los valores en la matriz se puede identificar unívocamente mediante dos subíndices. Uno indica la fila y otro la columna. Son adecuados para trabajar con dos bucles estructuras para

PARA. ..HASTA...HACER

PARA. ..HASTA...HACER

1. DECLARACION Y CREACION

```
int[][] tablero ; //Para declarar el tipo de la matriz.


tablero = new int [10][5] ; //Para crear la matriz.


int[][] tablero = new int [10][5]; //declara y crea .


int[][] mat = {{3,5,7}, {4,2,1}} //inicialización en la declaración
      //mat[0][0]=3 mat[0][1]=5 mat[0][2]=7
      //mat[1][0]=4 mat[1][1]=2 mat[1][2]=1
```

2. ACCESO A LOS ELEMENTOS DE LA MATRIZ

Al igual que en arreglos unidimensionales en un arreglo bidimensional podemos ver a cada elemento como una variable. Para recorrer la matriz se utilizan dos estructuras repetitivas, una va por las filas y la otra por las columnas. :

```
public void cargaMatriz (int[] [] mat, int cantFila, int cantCol) {
      int fila, col;
      for (fila = 0; fila< cantFila; fila++) { //carga por filas
         for (col =0; col < cantCol; col++) { //col dentro de cada fila
               System.out.println(ingres mat[fila][col);
            }
      TecladoIn.readLineInt();
      }
}
```

Nótese que cada módulo realiza una única acción. Ambos módulos llevan como parámetro un arreglo, en el *cargaMatriz* el mismo se modifica, en cambio en el *mostrarMatriz* sólo se muestra.

```
public void mostrarMatriz (int[] [] mat, int cantFila, int cantCol) {
            int fila, col;
            for (fila = 0; fila< cantFila; fila++) {
                 for (col =0; col < cantCol; col++)
                      System.out.println(mat[fila][col]);
             }
       }
```

2. CREACION E INICIALIZACION

Ejemplo: Creación de una matriz de *10 x 5* e inicialización del mismo.

```
int i=0;
int[][] mat = new int[10][5];

//mat.length es la cantidad de filas
for (i=0; i< mat.length ; i++) {

 //mat[i].length es la cantidad de columnas de la fila iésima
      for (j =0; j < mat[i].length; j++)
           mat[i][j])= 100;
      }
 }
```

2. LONGITUD DE UNA MATRIZ

matriz.length nos permite obtener el número de filas.
matriz[0].length nos permite obtener el número de columnas.

Ejemplo: Suma de dos matrices. Dadas dos matrices de igual dimensiones, sumar ambas.

```
static public int[][] sumaMatriz (int[][] matA, int[][] matB) {
      int i, j;     //mat es es de n x m y B también
      int[][] c = new int[matA.length][matA[0].length]
      for (i = 0; i< matA.length; i++) {
            for (j =0; j < matA[0].length; j++)
            c[i][j]= matA[i][j]+ matB[i][j];
      }
      return c;
   }
```

Nótese que cada ambas matrices tienen la misma dimensión, así como la matriz resultado.

```
int[][] a = {{3,5,7} ,{4,2,1}} ; //inicialización en la
declaración

int[][] b = {{3,5,7}, {4,2,1}} ;

int[][] c = sumaMatriz (a, b) ;
```

Ejemplo: Multiplicación de dos matrices **matA,** de *n x m* y **matB** de *m x l*, generando la matriz resultado matC de *n x l* , producto de la multiplicación de ambas.

```
static public int[][] multiMatriz (int[][] matA, int[][] matB) {
int i, j, k, suma=0;
int[][] c = new int[matA.length][matB[0].length]
      for (i = 0; i< matA.length; i++) {
            for (j =0; j < matB[0].length; j++){
                  for (k =0; k < matA[0].length; k++)
                        suma = matA[i][k]* matB[k][j];
                  c[i][j]= suma;
                  }
      suma = 0;}
      return c;
}
//inicialización en la declaración
int[][] a = {{3,5,7}, {4,2,1}} ;
int[][] b = {{3,5,7, 9} ,{4,2,1,1} ,{3,5,1,-1} } ;
int[][] c = multiMatriz (a, b) ;
```

3. MATRICES IRREGULARES

Cuando cada fila tiene dimensiones diferentes. Se crea fila por fila

```
int[][] mat = new int[2][]; //no se coloca la segunda dimensión

mat[0] = new int[5]; //a la fila 0 se define la dimensión de 5

mat[1] = new int[2]; //a la fila 1 se define la dimensión de 2

mat[2] = new int[3]; //a la fila 1 se define la dimensión de 3
```

Ejemplo: Mostrar los datos de una matriz irregular.

```
public void mostrarMatriz (int[][] mat) {
            int fila, col;
            for (fila = 0; fila< mat.length; fila++) {
                  for (col =0; col < mat[fila].length; col++)
                        System.out.println(mat[fila][col]);
            }
```