DATOS ESTRUCTURADOS

Arrays

Matrices

Operaciones con arrays

DATOS ESTRUCTURADOS

- Una estructura de datos es un **conjunto finito y ordenado de elementos homogéneos** que se caracteriza por su organización y por las operaciones que se definen en ella.
- Los datos de tipo estándar (entero, real, cadena, lógico, carácter) se pueden organizar en diferentes estructuras de datos estáticas y dinámicas.
- Las **estructuras de datos estáticas** son aquellas en las que el espacio ocupado en memoria se define en tiempo de compilación y no puede ser modificado durante la ejecución del programa (array, o vectores, matrices..)

ESTRUCTURAS DE DATOS

Array.- Conjunto de datos del mismo tipo que **se almacena en posiciones contiguas de memoria y reciben un nombre común**.

- El número de elementos de un array se denomina ***rango*** y viene determinado por el valor del índice superior.
- Para hacer **referencia a un elemento** del array se utiliza un **índice** que indica su posición relativa dentro de la estructura.
- Los índices de un array pueden ser constantes, variables o expresiones enteras.

ESTRUCTURAS DE DATOS

Ejemplo.

i =3

T[i] representa el elemento que ocupa la posición i .

T[0] es el primer elemento.

T[3] representa el elemento que ocupa la posición 4

T[i + 1] representa el elemento que ocupa la posición 5

ESTRUCTURAS DE DATOS

La dimensión de un array viene definida por el número de índices que utiliza para hacer referencia a sus elementos.

Ejemplo.

matriz [i][j] hace referencia al elemento que ocupa la fila i, y la columna j.

Pueden ser:

Unidimensionales.- Arrays / Vectores

Bidimensionales.- Matrices Multidimensionales.

Declaración. Ejemplo.

double lluvia [ ] ; // lluvia es un array de valores de tipo double

Aún **no hemos definido el número de elementos** del array.

Igual que para otro tipo de variables, el valor por omisión es null, que representa un array sin ningún valor.

Para indicar el número de elementos utilizaremos el operador new que asigna espacio de memoria al array.

Lluvia = new double [31];

Declaración. Ejemplo.

tipoBase lluvia [ ] = new tipoBase [num] ;

// lluvia es el nombre de un array de num elementos de tipoBase

Ejemplo.

int numDias =31;

// La variable numDias contiene el número de días del mes.

double lluvia [ ] = new double [numDias];

Declaración. Ejemplo.

En este momento disponemos de **un array de 31 elementos** numerados del 0 al 30 de tipo double, accesible mediante la notación

lluvia [0], lluvia[1] , ... lluvia[30]

Con cada una de las componentes del array se pueden hacer las mismas operaciones que haríamos con una variable simple de tipo doble. Ejem. lluvia[3]=30.5;

lluvia[5]=lluvia[3] *2 ;

int i = 6;

lluvia[i+1]=lluvia [i-1];

Inicialización.

Existe la posibilidad de inicializar todas las componentes de un array de forma simultánea, o hacerlo individualmente componente a componente. Ejemplo.

double ejem[]={1.3, 2.678, 246.8/2+1, 25.7, 17.98, 12.64 }

// individualmente

double ejem =new double[6];

ejem[0]=1.3 ; ejem[1]=2.678; ejem[2]=246.8/2+1

ejem[3]=25.7; ejem[4]=17.98; ejem[5]=12.64 ;

Ejemplo-1.Creación de un array y cálculo de la media

```
import java.util.*;
public class crear_array {
static Scanner sc = new  Scanner (System.in);
public static void main(String[] args) {
double lluvia[ ] = new double[31];
double suma=0 ;
System.out.println("Dame valores");
for (int i=0; i<31; i++) {
      lluvia[ i ] = sc.nextDouble();
      suma += lluvia[ i ];  } //Cierra el for
System.out.println("La media es : "+(s/31));
 } // cierra main()
} // cierra la clase
```

El atributo length.

Asociado a cualquier variable de tipo array existe un valor (atributo) que determina la longitud real, o número de componentes del array.

Dicho atributo se denomina **length** y se utiliza posponiendo al nombre de la variable la palabra **length**.

Así, mediante la siguiente instrucción se escribiría el número de componentes del array lluvia.

```
System.out.println(lluvia.length);
```

Paso de arrays a métodos.

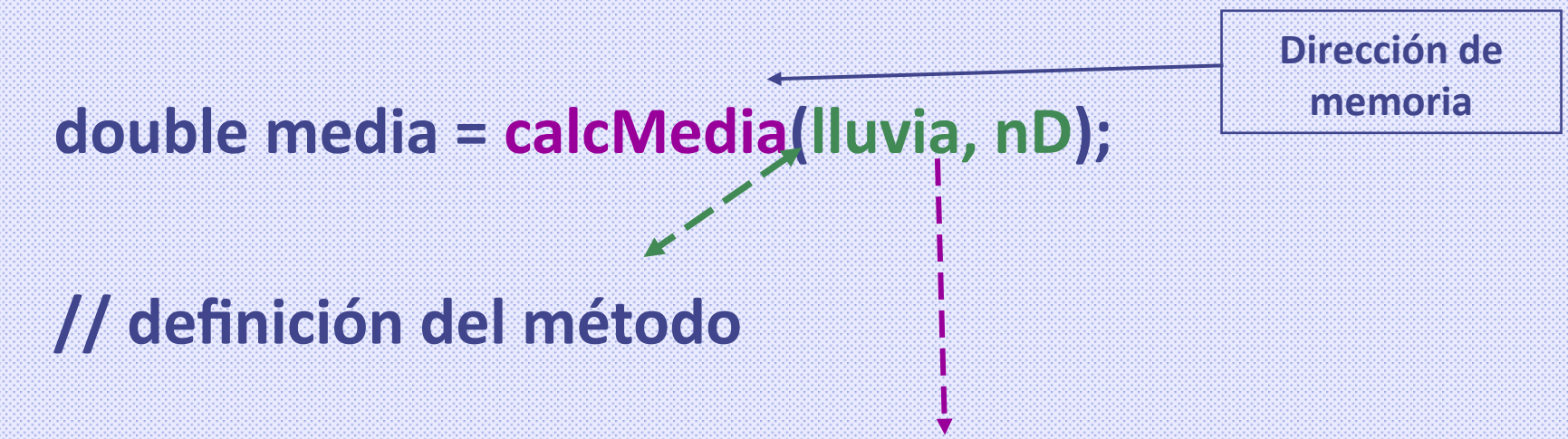
Los parámetros de tipo array y en general **cualquier objeto**, se pasa **siempre por referencia.**

En el paso por referencia lo que realmente **pasamos es la dirección de la variable** u objeto, por lo que el papel del parámetro formal es el de ser una referencia al parámetro real.

Las modificaciones que el método pueda hacer sobre los parámetros formales, tienen **efecto directo** sobre los parámetros reales, ya que **ambas variables actúan sobre la misma dirección de memoria.**

Arrays como parámetros.

Para utilizar un array como argumento real en la llamada a un método, se utiliza el nombre de la variable, sin embargo en la definición formal del parámetro es necesario indicar que es un array, mediante los corchetes.



Ejemplo-2. Crea un array y cálculo de la media, con método.

```
import java.util.*;
public class crear_array {
  static Scanner sc = new Scanner(System.in);
  public static void main(String[] args) {
        double lluvia[] = new double[31];
        double m;
        System.out.println("Dame valores");
        for(int i=0; i<31;i++)
        lluvia[ i ]= sc.nextDouble(); // cierra el for
         m=calcMedia(lluvia);
         System.out.println("La media es : "+m);
  } // cierra main()
```

Ejemplo-2. - Continuación -

```
static double calcMedia (double lluv[] ){
  int j=lluv.length ;
 double s=0;
   for (int c=0; c<j; c++)
          s += lluv[c]; //Cierra el for
   return (s/j);
     }  // cierra el método calcMedia()
} // cierra clase
```

Ejemplo 3.- Cambia el menor a la primera posición

```
public static void main(String arg[]) {
        int j = 0, aux;
      int v[] = new int[5]; //Vector de 5 enteros
      Scanner sc = new Scanner(System.in);
      for (int i = 0; i < 5; i++) {
         System.out.println("Elemento "+i+" del vector");
         v[i] = sc.nextInt(); } // Cierra el for y  leer elementos
        for (int i = 1; i < 5; i++) {
         if (v[i] < v[j])
            j = i; } //Cierra el for y buscamos el menor
   aux = v[0];        v[0] = v[j];       v[j] = aux;  //Se intercambian posiciones
     System.out.println("La solución es:");
     for (int i = 0; i < 5; i++) {
         System.out.println(v[i]);        }
  } //Cierra el main
```

Ejemplo 4. Crea un array de 4 nombres sin repetidos.

```
public static void main(String[] ar) {
Scanner sc = new Scanner(System.in);
 String n;
 String v[] =new String[5];
 int j, i=0;
 boolean sw = false;
 System.out.println("Nombre :"+i);
 v[i]=sc.nextLine();
```

Ejemplo 4. Crea un array de 4 nombres sin repetidos.

```
while (i< 4) {
  j=0; sw=false;
 System.out.println("Nombre :"+ (i+1));
 n=sc.nextLine();
  do {
    if (n.equals(v[j])) {
       sw  =true ;
       System.out.println("Repetido ");
          }
       else j++;
    } while (j<=i && sw== false); // cierra el do while
```

Ejemplo 4. Crea un array de 4 nombres sin repetidos.

```
 if (sw==false) {
 i++;  v[i]=n;
 }
 } // cierra while
System.out.println("\n  Nombres");
   for( i=0;i<4;i++){
        System.out.println(i+"  "+v[i]);
   }
 } //Cierra el main
} //Cierra la clase
```

ESTRUCTURAS DE DATOS

BIDIMENSIONALES

Una matriz es un vector de vectores / array de arrays.

Declaración:

```
double matriz [] [] = new double [4] [4];
double matriz [] [] = new double [4] [];
matriz[0]= new double[4];
matriz[1]= new double[4];
matriz[2]= new double[4];
matriz[3]= new double[4];
```

// Es posible inicializar cada uno de los subarrays con un tamaño diferente.

Ejemplo-5. Crea un array bidimensional.

```
import java.util.*;
public class array_bi {
    public static void main(String[] ar) {
   Scanner sc=new Scanner(System.in);
    int matriz[][] = new int[4][4];
int f, c;
// leemos los elementos de la matriz
for(int f=0; f<4;f++) {
for (int c=0; c<4; c++) {
        System.out.println("Elemento "+c+" fila "+f);
        matriz[f][c]=sc.nextInt();   } // Cierre for
}  // Cierre for
for(int f=0; f<4;f++) {   // Visualizamos los elementos
     System.out.println();
     for (int c=0; c<4; c++)
   System.out.print(matriz[f][c]+" ");
}  } }  // Cierre primer for main y clase.
```

Ejemplo-6. Multiplica dos arrays bidimensionales.

```
public static void main(String[] args) {
int a[][]={{1,2},{3,-2}};
int b[][]= {{1,1},{2,0}};
int p[][]=new int [a.length][b[0].length];
for (int i=0;i<a.length;i++) {
  for(int h=0; h<b[i].length;h++) {
   int  s=0;
    for(int j=0; j<b.length;j++)
        s+=a[i][j]*b[j][h]; // Cierre for
    p[i][h]=s;
  } // Cierre for
 } // Cierre for
for(int j=0;j<p.length;j++){
System.out.println();
for(int h=0;h<p[j].length;h++)
 System.out.print(p[j][h]+"\t"); // Muestra los resultados
 } // Cierre for
} // Cierre main
```

https://www.youtube.com/watch?v=WAYpQ4bS84o

Ejemplo 7. Crea la matriz traspuesta.

```
import java.util.*;
public class transpuesta {
 public static void main(String[] args)
{
  int m[][]= {{4,6,8},{1,2,4},{0,1,2}};
  int t[][]= tras(m);
  for (int i = 0; i < t.length; i++) {
    System.out.println();
    for (int j=0;j<t[i].length;j++)
   System.out.print((t[i][j]));
   }   System.out.println();  }
```

```
 static int[][] tras (int v[][]) {
int m[][]= new int[v[0].length]
[v.length];
 for (int i = 0; i < v.length; i++)
  for (int j=0;j<v[i].length;j++)
          m[j][i]=v[i][j];
 return m;
 }
}
```

https://www.youtube.com/watch?v=gH9EcHLmvG8

Operaciones con arrays

- Recorrido
- Actualización
- Ordenación
- Búsqueda

Operaciones con arrays : Recorrido

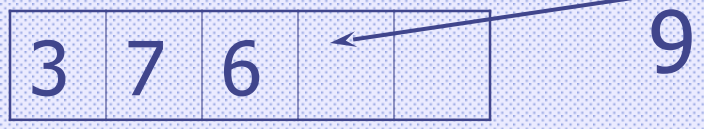
```
// Recorremos todo el array para visualizar sus elementos
public static void main(String[] ar) {
        int v[] = new int[10];
        for (int i = 0; i < v.length; i++) {
            System.out.println(v[i] + "");
        }
    }
```

Operaciones con arrays : *Actualización*

Consta de **3** operaciones elementales

- **Añadir.**
- **Insertar.**
- **Borrar.**

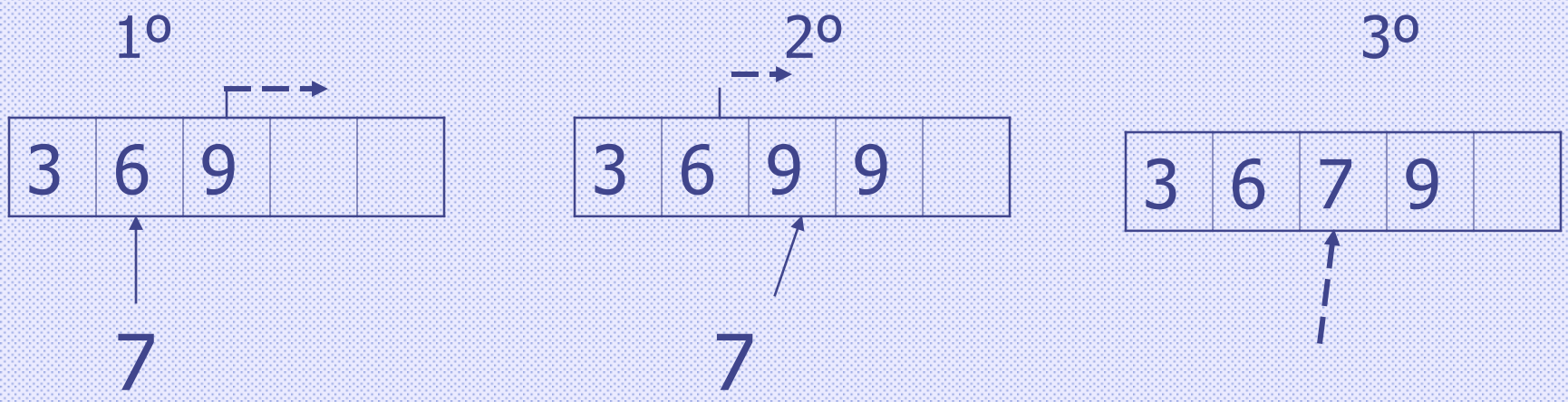
Añadir. Es la operación de añadir un nuevo elemento al final del vector, (comprobando que haya espacio de memoria suficiente para este nuevo elemento)



Operaciones con arrays : Actualización

Insertar.- Consiste en introducir un nuevo elemento en el interior del vector.

Es necesario realizar un desplazamiento previo, para colocar el nuevo elemento en su posición relativa .

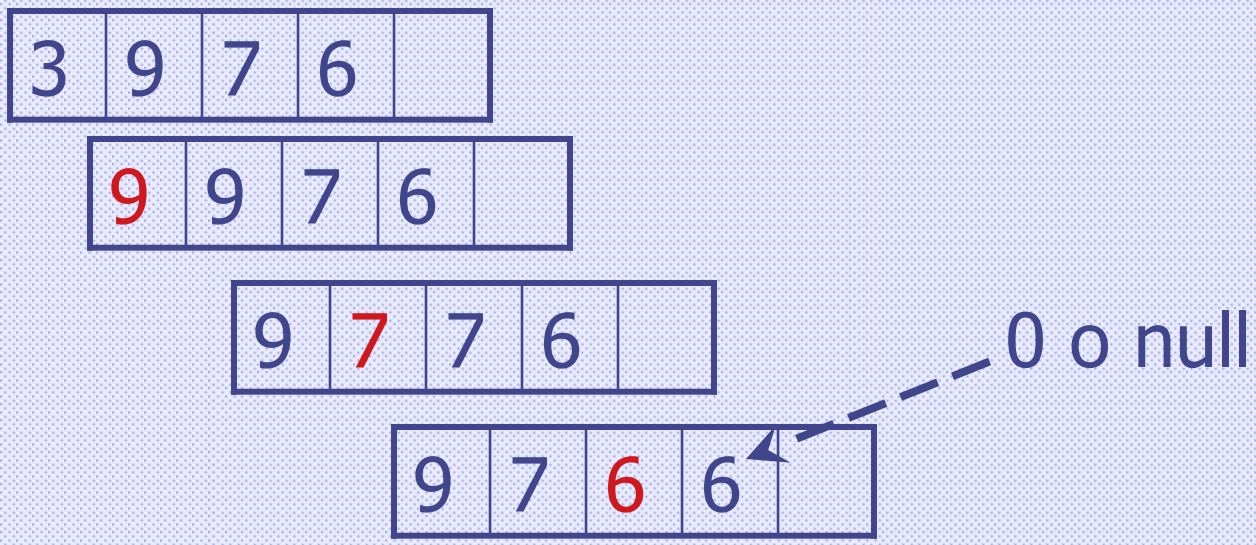


Operaciones con arrays : Actualización

Borrar.- Consiste en eliminar un elemento del vector.

Realizando un desplazamiento *hacia izquierda,* de los elementos inferiores a él , para reorganizar el vector.

Borramos el 3



Operaciones con arrays : Ordenación

Métodos de ordenación:

- Intercambio o burbuja
- Inserción o baraja
- Selección directa

Operaciones con arrays : Ordenación

Método de intercambio o burbuja.

Consiste en comparar **pares de elementos adyacentes** e intercambiarlos entre sí, hasta que estén todos ordenados.

1º Se comparan a[0] y a[1] si a[1]>a[0] se mantienen, sino se intercambian.

2º Se comparan los elementos a[1] y a[2] de nuevo, y si es necesario se intercambian.

3º El proceso continua hasta que cada elemento del vector ha sido comparado con el adyacente y se han realizado los cambios necesarios.

Ejemplo. Método de la burbuja

```
class burbuja {
 public static void main(String[] args) {
 int v[ ]={1,34,18,12,69,96,22,61,17,30};
System.out.println("Array inicial ");
for(int i=0;i<v.length;i++)
System.out.print(v[i]+" ");//Muestra vector
for(int i=0; i<v.length; i++)
 for(int j=i+1; j<v.length; j++)
  if (v[ j-1] > v[j] ) {
  int may = v[j-1] ;
  v[j-1] = v[j] ;
  v[j]=may;  }  System.out.println("\
nOrdenado");
for(int i=0;i<v.length;i++)
   System.out.print(v[i]+" ");  }
} //fin clase
```

- Tras realizar el primer recorrido el mayor irá a la última posición.
- En el segundo el siguiente mayor irá a la penúltima y así sucesivamente.
- **La operación para un array de n elementos se repetirá n-1 veces.**

Ejemplo. Método de la burbuja mejorado

```
class burbuja2 {
   public static void main(String[] args) {
     int v[]={1,34,18,12,69,96,22,61,17,30};
     boolean sw=false;
     System.out.println("Array inicial "); //Se muestra
     for(int i=0;i<v.length;i++) System.out.print(v[i]+" "); //End for
 while (!sw) {
   sw=true;
   for(int j=0; j<v.length-1; j++)
        if (v[j] > v[j+1] ){   int may = v[j] ;
                               v[j] = v[j+1] ;
                               v[j+1]=may;
                               sw=false;   }
 }
System.out.println("\nOrdenado");
  for(int i=0;i<v.length;i++) System.out.print(v[i]+" ");
     }
 }
```

2.- Ordenación por inserción (método de la baraja)

```
Algoritmo cla_insercion
//declaracion variables
Inicio
// leer vector
Para I ← 2 hasta n hacer
Aux ← A[I]
J ← I -1
Sw ← falso
Mientras no sw y j >= 1 hacer
 Si aux < A[j] entonces A[j+1] ←
A[j]
                    J ← j -1
       Si_no  sw ← verdad
 Fin_si
Fin_mientras
 A[j+1] ← Aux
Fin_para
Fin.
```

```
Inserción mejorada
Inicio
// leer vector
Para I ← 2 hasta n hacer
Aux ← A[i]
P ← 1 // primero
U ← I -1  // último
Mientras P < =  U hacer
C ← (P + U ) div 2
Si aux < A[C] entonces U ← C -1
Si_no  P ← C + 1
Fin_si
Fin_mientras
Para  k ← I - 1 hasta P decremento
-1 hacer
A[k + 1 ] ←A[k]
Fin_para
A[p] ← aux
Fin_para
Fin.
5 4 24 39 43 68 84 45
```

Operaciones con arrays : Ordenación

- **Método de ordenación por selección.**

Consiste en buscar el elemento menor del vector y colocarlo en la primera posición

```
Algoritmo selección
 Inicio
 Para I =1 hasta n hacer
 Leer a[I]
 Fin_para
 // clasificación
 Para i = 1 hasta n - 1 hacer
 Aux ← a[i]
 K ← i
Para j = i +1 hasta  n hacer
    Si a[ j ] < aux entonces Aux ←a[ j ]
                             K ← j
         Fin_si
 Fin_para
 a[ k] ← a[i]
 a[I]← aux
 Fin_para

 / /visualizar la lista ordenada
 Para j = 1 hasta N hacer
          Escribir ( a[j] )
 Fin_para
 Fin.
```

Operaciones con arrays : Búsqueda

```
import java.util.Arrays;
class buscaDicotomica {
  public static void main(String[] args) {
  int v[]={1, 34 ,18, 12 ,6 ,9 ,22 ,61 ,17 ,30};
  Arrays.sort(v); //Array ordenado
  int resp, busca=12;
  boolean sw=false;
  System.out.println("Buscamos el: "+busca);
  resp = busca(v,  busca);
  if (resp==-1) System.out.print("\nNo existe el "+ busca );
    else System.out.print("\nEstá en la posición "+(resp+1));
   }  // cierra main
```

Operaciones con arrays : Búsqueda continuación

```
static int busca(int b[], int m){
int alto=b.length, bajo=0, central=0;
boolean sw=false;
while (! sw && bajo <= alto) {
    central=(alto+bajo)/2;
     if (b[central]==m) sw=true;
          else if (b[central]>m) alto=--central;
                  else bajo=++central;  }
  if (sw) return central;
       else return -1;
  } // cierra el método
} // cierra la clase
```

La Clase Arrays

- Para poder utilizarla tendremos que importar el package **java.util.Arrays**;
- No es necesario crear un objeto de la clase para utilizar los métodos, simplemente utilizamos el nombre de clase.
- La clase contiene varios métodos para manipular Arrays.

Métodos de la Clase Arrays

Los métodos mas frecuentes son:

static int [] copyOf(int [] original, int newLength)

static int [] copyOfRange(int [] original, int from, int to)

static boolean equals (int[ ] a, int[ ] a2)

static void sort(Object [] , a)

static int binarySearch(Object [], Object Key)

Métodos de la Clase Arrays

- **static int [] copyOf**(int [] original, int newLength)

Ejemplo:

```
int [] array2 = {2, -4, 3, -7};
int nueva[]=  Arrays.copyOf(array2, 6);
```

array2= 2, -4, 3, -7

nueva= 2, -4, 3, -7, 0, 0

Métodos de la Clase Arrays

static int [] copyOfRange(int [] original, int from, int to)

- Copia los elementos de un arrays de cualquier tipo simple: char, byte, boolean, double, long indicando desde y hasta donde copia. (Rango)

Métodos de la Clase Arrays

static boolean equals (int[ ] a, int[ ] a2)

- Devuelve true si los dos arrays especificados tienen relación de igualdad entre sí, es decir si tienen el mismo numero de elementos y los correspondientes pares de elementos son iguales.

Métodos de la Clase Arrays

- **static int binarySearch(Object [], Object Key)**

Realiza la búsqueda binaria de un elemento dentro del array, que debe estar previamente ordenado.

Métodos de la Clase Arrays

- **static void sort(Object [] , a)**

Ordena el array de tipo Object o tipos simples en modo ascendente.

Métodos de la Clase Arrays

public static void fill(int[] a, int val)

- Llena el array especificado con el valor indicado por val.
- public static String toString(short[] a)

Métodos de la Clase Arrays

- **public static String toString(int[] a)**
- Devuelve una cadena que representa el contenido del array especificado.
- La cadena es una lista de elementos, entre corchetes ("[]“ separados por una coma seguida de un espacio).
- Los elementos se convierten en cadenas como lo haria el método String.valueOf (int) .