Clase Arrays del api Java.

Métodos equals (comparar), copyOf (copiar), fill (rellenar),Sort (ordenar).

Class Arrays

java.util.Arrays

Esta clase contiene varios métodos para manipular arrays (por ejemplo para ordenar un array o buscar un valor u objeto dentro de él) y para comparar arrays.

Dado que pertenece al **package util**, para poder usar esta clase habremos de incluir en cabecera import java.util.Arrays; o bien **import java.util.***;.

Al igual que los arrays son unos objetos que hemos dicho son especiales (al carecer de métodos), podemos decir que la clase Arrays es una clase un tanto especial por cuanto carece de constructor. Digamos que directamente al cargar la clase con la sentencia import correspondiente automáticamente se crea un objeto denominado Arrays que nos permite realizar manipulaciones con uno o varios arrays (p. ej. ordenar un array, comparar dos arrays, etc.).

Dicho objeto podemos utilizarlo directamente: no es necesario declararlo ni crearlo, eso es automático en Java, y por eso decimos que esta clase es una clase especial.

La clase Arrays tiene muchos métodos, entre ellos varios métodos equals (sobrecarga del método) que hacen que equals sea aplicable tanto a arrays de los distintos tipos primitivos como a arrays de objetos.

En concreto el método aplicable a arrays de enteros primitivos es:

> static boolean **equals** (int[ ] a,int[ ] a2)
>
> Devuelve true si los dos arrays especificados tienen relación de igualdad entre sí.

Vamos a aplicar este método para comparar el contenido de dos arrays de enteros (relación de igualdad). La aplicación al resto de tipos primitivos y objetos es análoga.

La sintaxis en general es: Arrays.equals (nombreArray1, nombreArray2).

EJERCICIO EJEMPLO 1. COMPARAR ARRAYS (RELACIÓN DE IGUALDAD) USANDO LA CLASE ARRAYS

```
import java.util.Arrays;
//Test comparar arrays relación de igualdad
public class TestCompararArrays {
   public static void main (String [ ] Args) {
   int [ ] miArray1 = {2, -4, 3, -7};
   for (int i=0; i<miArray1.length; i++) {
      System.out.print ("miArray1[" + i +"]= " + miArray1[i]+"; ");}
   System.out.println ("");
   int [ ] otroArray = {2, -4, 3, -7};
   for (int i=0; i<otroArray.length; i++) {
      System.out.print ("otroArray[" + i +"]= " + otroArray[i]+"; ");}
   System.out.println ("¿Son el mismo objeto? ... " + (miArray1==otroArray));
      System.out.println ("¿Tienen el mismo contenido (relación de igualdad)? ... " +
Arrays.equals(miArray1, otroArray) );
   otroArray = miArray1;//otroArray pasa a ser el mismo objeto que miArray1
   for (int i=0; i<otroArray.length; i++) {
   System.out.print ("otroArray[" + i +"]= " + otroArray[i]+"; ");}
   System.out.println ("¿Son el mismo objeto? ... " + (miArray1==otroArray) );
      System.out.println ("¿Tienen el mismo contenido (relación de igualdad)? ... " +
Arrays.equals(miArray1, otroArray) );
   }//Cierre del main
}//Cierre de la clase
```

Trata de predecir tú mismo el resultado y compáralo con el que ofrecemos. **Resultado**:

```
miArray1[0]= 2; miArray1[1]= -4; miArray1[2]= 3; miArray1[3]= -7;
otroArray[0]= 2; otroArray[1]= -4; otroArray[2]= 3; otroArray[3]= -7; ¿Son el mismo objeto? ... false
¿Tienen el mismo contenido (relación de igualdad)? ... true
otroArray[0]= 2; otroArray[1]= -4; otroArray[2]= 3; otroArray[3]= -7; ¿Son el mismo objeto? ... true
¿Tienen el mismo contenido (relación de igualdad)? ... true
```

El resultado ahora sí es correcto, porque hemos usado correctamente la clase Arrays para realizar la comparación entre dos arrays.

EJERCICIO EJEMPLO 2. COPIAR CONTENIDOS ENTRE ARRAYS SIN ESTABLECER RELACIÓN DE IDENTIDAD ("manual", aplicable a tipos primitivos y a objetos).

Antes vimos cómo asignar el contenido de un array a otro haciendo que la variable apunte al mismo objeto. Vamos a ver ahora cómo copiar el contenido entre dos arrays pero manteniendo que cada variable denominadora del array apunte a un objeto diferente:

```
import java.util.Arrays;
//Test copia arrays con igualdad sin identidad
public class TestCopiaConIgualdadSinIdentidad {
 public static void main (String [ ] Args) {
 int [] miArray1 = {2,-4,3,-7};
 for (int i=0; i<miArray1.length; i++) {
    System.out.print ("miArray1[" + i +"]= " + miArray1[i]+"; ");}
 System.out.println();
 int [] otroArray = {1,2,4,8};
 for (int i=0; i<otroArray.length; i++) {
     System.out.print("otroArray[" + i +"]= " + otroArray[i]+"; ");}
 System.out.println("¿Son el mismo objeto? ... " + (miArray1==otroArray) );
    System.out.println("¿Tienen el mismo contenido (relación de igualdad)? ... " +
Arrays.equals(miArray1, otroArray) );
//Realizamos una asignación elemento a elemento
 for (int i=0; i < otroArray.length; i++) {
    otroArray[i] = miArray1[i];}
 for (int i=0; i < otroArray.length; i++) {
    System.out.print ("otroArray[" + i +"]= " + otroArray[i]+"; ");}
 System.out.println("¿Son el mismo objeto? ... " + (miArray1==otroArray) );
    System.out.println("¿Tienen el mismo contenido (relación de igualdad)? ... " +
Arrays.equals(miArray1, otroArray) );
}//Cierre del main
}//Cierre de la clase
```

Trata de predecir tú mismo el resultado y compáralo con el que ofrecemos. **Resultado**:

```
miArray1[0]= 2; miArray1[1]= -4; miArray1[2]= 3; miArray1[3]= -7;
otroArray[0]= 1; otroArray[1]= 2; otroArray[2]= 4; otroArray[3]= 8; ¿Son el mismo objeto? ... false
¿Tienen el mismo contenido (relación de igualdad)? ... false
otroArray[0]= 2; otroArray[1]= -4; otroArray[2]= 3; otroArray[3]= -7; ¿Son el mismo objeto? ... false
¿Tienen el mismo contenido (relación de igualdad)? ... true
```

EJERCICIO 3. COPIAR CONTENIDOS ENTRE ARRAYS SIN ESTABLECER RELACIÓN DE IDENTIDAD (Usando el método copyOf de la clase Arrays, aplicable a tipos primitivos y a objetos).

El método copyOf de la clase Arrays nos permite:

a) Copiar un array manteniendo el número de elementos.

b) Copiar un array agrandando el número de elementos que tiene, quedando los nuevos elementos rellenos con valores cero o nulos.

c) Copiar un array empequeñeciendo el número de elementos que tiene; los elementos que no caben en el

nuevo array, dado que tiene menor capacidad, se pierden (el array queda truncado).

copyOf es un método sobrecargado. En el caso de arrays de enteros su signatura es la siguiente:

> static int[ ] **copyOf** (int[ ] original, int newLength)
>
> Copia el array especificado, truncando o rellenando con ceros (si fuera necesario) de manera que la copia tenga el tamaño especificado.

Para el resto de tipos primitivos su sintaxis es análoga: Arrays.copyOf (nombreDelArray, n) siendo n un entero que define la nueva longitud del array (n puede ser mayor, menor o igual que la longitud del array original). El código de ejemplo sería este (usamos el copyOf sin variar la longitud del array):

```
import java.util.Arrays;
//Test uso de copyOf método clase Arrays
public class TestUso_copyOf_1 {
 public static void main (String [ ] Args) {
 int [ ] miArray1 = { 2, -4, 3, -7 };
 for (int i=0; i<miArray1.length; i++) {
    System.out.print ("miArray1[" + i +"]= " + miArray1[i]+"; "); }
 System.out.println ("");
 int [ ] otroArray = { 1, 2, 4, 8 };
 for (int i=0; i<otroArray.length; i++) {
    System.out.print ("otroArray[" + i +"]= " + otroArray[i]+"; "); }
 System.out.println ("¿Son el mismo objeto? ... " + (miArray1==otroArray) );
   System.out.println ("¿Tienen el mismo contenido (relación de igualdad)? ... " +
Arrays.equals(miArray1, otroArray) );
//Copiamos el array utilizando el método copyOf de la clase Arrays
 otroArray = Arrays.copyOf(miArray1, miArray1.length);
 for (int i=0; i<otroArray.length; i++) {
    System.out.print ("otroArray[" + i +"]= " + otroArray[i]+"; "); }
 System.out.println ("¿Son el mismo objeto? ... " + (miArray1==otroArray) );
   System.out.println ("¿Tienen el mismo contenido (relación de igualdad)? ... " +
Arrays.equals(miArray1, otroArray) );
 }//Cierre del main
}//Cierre de la clase
```

Trata de predecir tú mismo el resultado y compáralo con el que ofrecemos. Resultado:

> miArray1[0]= 2; miArray1[1]= -4; miArray1[2]= 3; miArray1[3]= -7;
> otroArray[0]= 1; otroArray[1]= 2; otroArray[2]= 4; otroArray[3]= 8; ¿Son el mismo objeto? ... false
> ¿Tienen el mismo contenido (relación de igualdad)? ... false
> otroArray[0]= 2; otroArray[1]= -4; otroArray[2]= 3; otroArray[3]= -7; ¿Son el mismo

objeto? ... false
¿Tienen el mismo contenido (relación de igualdad)? ... true

Hemos comprobado que el método copyOf de la clase Arrays realiza una copia elemento a elemento entre los contenidos de dos arrays pero no hace que los punteros apunten al mismo objeto.

Prueba a variar la longitud que se le pasa como parámetro al método copyOf, por ejemplo:

otroArray = Arrays.copyOf(miArray1, miArray1.length+2); //Resto del código igual

otroArray = Arrays.copyOf(miArray1, miArray1.length-2); //Resto del código igual

Comprueba que los resultados son el alargamiento del array y su relleno con ceros, o el acortamiento con pérdida de los datos (truncamiento) que no caben debido al recorte de la longitud.

En el caso de alargamiento o expansión del array cuando se trata de un array que no sea de enteros, si son tipos numéricos se rellenan los excedentes con ceros, si son booleanos se rellenan los excedentes con false, si son char se rellenan de caracteres vacío, y si son objeto se rellenan los excedentes con null.

RELLENAR UN ARRAY CON UN VALOR U OBJETO. MÉTODO FILL DE LA CLASE ARRAYS

La clase Arrays tiene un método, denominado fill, sobrecargado, que permite rellenar un array con un determinado valor u objeto. En el caso de arrays de enteros la sintaxis es:

static void **fill** (int[ ] a, int val)
Asigna el valor entero especificado a cada elemento del array de enteros indicado.

En general la sintaxis será: Arrays.fill (nombreDelArray, valor con el que se rellena). El valor con el que se rellena depende del tipo del array. Por ejemplo, si es un array de tipo booleano, tendremos que rellenarlo bien con true o bien con false, no podremos rellenarlo con un tipo que no sea coherente. Ejemplos de uso:

Arrays.fill (resultado, '9'); Como rellenamos con un carácter, resultado habrá de ser un array de caracteres, ya que en caso contrario no habría coincidencia de tipos.

Arrays.fill (permitido, true); Como rellenamos con un true, resultado será un array de booleanos. De otra manera, no habría coincidencia de tipos. Ejemplo de código:

```
import java.util.Arrays;
public class TestMetodoFillArrays {
   public static void main (String [ ] Args) {//main cuerpo del programa
      int [ ] miArray = new int[10];
      Arrays.fill(miArray, 33);
      for (int tmp: miArray) {
      System.out.print (tmp + ",");
      }//Recorrido del array con un for each
   }//Cierre del main
}//Cierre de la clase
```

Ejecuta el código y comprueba que el resultado es: 33,33,33,33,33,33,33,33,33,33,. Es decir, el array queda relleno en todos sus elementos con 33.

En caso de que el array tenga contenidos previos al aplicarle el fill, todos sus elementos quedarán reemplazados por el elemento de relleno.

No obstante, hay otro método que permite especificar los índices de relleno de modo que se pueda preservar parte del contenido previo del array:

> static void **fill**(int[ ] a, int fromIndex, int toIndex, int val)
>
> Asigna el valor entero especificado a cada elemento del rango indicado para el array especificado.

Comprobar funcionamiento extraño.

```
import java.util.Arrays;
public class TestMetodoFillArrays1Limitado {
   public static void main (String [ ] Args) {//main cuerpo del programa
      int [ ] miArray = new int[10];
      Arrays.fill(miArray, 33);
      for (int tmp: miArray) {
      System.out.print (tmp + ",");
     }//Recorrido del array con un for each
  Arrays.fill(miArray,2,6,4);
   for (int tmp: miArray) {
      System.out.print (tmp + ",");
     }
   }//Cierre del main
}//Cierre de la clase
```

Escribe un fragmento de código utilizando esta signatura del método fill y comprueba sus resultados.

Para ordenar un array se utiliza: **Arrays.sort(a);**

```
import java.util.Arrays;
public class OrdenarVector {
   public static void main(String[] args) {//main cuerpo del programa
   int[] a={9,2,4,3,1,8,7,5, 6};
   System.out.println("Arreglo desordenado");
      for (int i = 0; i < a.length; i++)
      System.out.print(a[i]+", ");
   Arrays.sort(a);
   System.out.println("\nArreglo ordenado");
      for (int i = 0; i < a.length; i++)
      System.out.print(a[i]+", ");
}//Cierre del main
}//Cierre de la clase
```