6.4 La clase Arrays

1. Introducción

En el paquete *java.utils* se encuentra la clase *Arrays*. Esta clase contiene métodos estáticos para manipular arrays, como por ejemplo, búsquedas, comparaciones, copias y ordenaciones.

2. Contenido de un array

- Método **toString**: devuelve una representación en cadena del contenido del array especificado. Dicha cadena está formada por los elementos del array entre corchetes y separados por comas. Es muy útil para sacar por consola la información sobre el array. Se utiliza con arrays unidimensionales.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class ToString {

    public void show() {

        int[] array = { 3, 4, 5, 6, 7, 8 };
        System.out.printf("La información del array es: %s",
Arrays.toString(array));

    }

    public static void main(String[] args) {

        new ToString().show();

    }

}
```

La salida por consola es:

```
La información del array es: [3, 4, 5, 6, 7, 8]
```

- Método ***deepToString***: igual que el anterior pero para arrays multidimensionales.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class DeepToString {

    public void show() {

        int[][] array = { { 0, 1, 2, 3, 4 }, { 5, 6, 7, 8, 9 }, { 10, 11, 12, 13,
14 } };

        System.out.printf("La información del array es: %s",
Arrays.deepToString(array));

    }

    public static void main(String[] args) {

        new DeepToString().show();

    }

}
```

La salida por consola es:

```
La información del array es: [[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13,
14]]
```

3. Rellenar un array

Método **fill**: permite rellenar todo un array unidimensional con un determinado valor. También tiene una versión sobrecargada para especificar un rango.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class Fill {

    public void show() {

        int[] array = new int[6];
        Arrays.fill(array, 1);
        System.out.println(Arrays.toString(array));//[1, 1, 1, 1, 1, 1]
        Arrays.fill(array, 2, 5, 0);
        System.out.println(Arrays.toString(array));//[1, 1, 0, 0, 0, 1]

    }
```

```
    public static void main(String[] args) {

        new Fill().show();

    }

}
```

4. Búsquedas

Método **binarySearch**:

- permite buscar un elemento de forma ultrarrápida en un array ordenado (en un array desordenado, sus resultados son impredecibles).
- Devuelve el índice en el que está colocado el elemento.
- Si el elemento no se encuentra, devuelve un número negativo.
- Si el array contiene múltiples elementos con el valor especificado, no hay garantía de cuál será encontrado.
- Está sobrecargado para buscar en un determinado rango del array. Se proporcionan dos parámetros más para especificar el rango:
    - *fromIndex*: el índice (incluido) de la parte inferior del rango.
    - *toIndex*: el índice (excluido) de la parte superior del rango.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class BinarySearch {

    public void show() {

        int lowerRange, higherRange;
        int[] array = { 3, 4, 5, 6, 7, 8 };
        int number = 9;
        int index = Arrays.binarySearch(array, number);
        if (index >= 0) {
            System.out.printf("El número %d se encuentra en el índice %d\n",
number, index);
        } else {
            System.out.printf("El número %d no se encuentra en el array\n",
number);
        }
        lowerRange = 3;
        higherRange = 6;//Como está excluido, se busca hasta el índice 5
        index = Arrays.binarySearch(array, lowerRange, higherRange, number);
        if (index >= 0) {
            System.out.printf("El número %d se encuentra en el índice %d",
number, index);
        } else {
            System.out.printf("El número %d no se encuentra en el array en el
rango %d-%d", number, lowerRange,
```

```
                    higherRange - 1);
        }

    }

    public static void main(String[] args) {

        new BinarySearch().show();

    }

}
```

Si ejecutamos dicho código con *number=5*, tendremos la siguiente salida por consola:

```
El número 5 se encuentra en el índice 2
El número 5 no se encuentra en el array en el rango 3-5
```

Si ejecutamos dicho código con *number=9*, entonces obtendremos:

```
El número 9 no se encuentra en el array
El número 9 no se encuentra en el array en el rango 3-5
```

5. Comparaciones

- Método **equals**: compara dos arrays y devuelve true si son iguales. Se consideran iguales si son del mismo tipo, tamaño y contienen los mismos valores. También tiene una versión sobrecargada para especificar un rango.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class Equals {

    public void show() {

        int[] array1 = { 3, 4, 5, 6, 7, 8 };
        int[] array2 = { 3, 4, 5, 6, 7, 8 };
        int[] array3 = { 3, 4, 5, 6, 7, 8, 9 };

        System.out.printf("%s y %s %s iguales\n", Arrays.toString(array1),
Arrays.toString(array2),
                Arrays.equals(array1, array2) ? "son" : "no son");
        System.out.printf("%s y %s %s iguales\n", Arrays.toString(array1),
Arrays.toString(array3),
                Arrays.equals(array1, array3) ? "son" : "no son");
        System.out.printf("%s y %s %s iguales en el rango 1-5",
Arrays.toString(array1), Arrays.toString(array3),
                Arrays.equals(array1, 1, 6, array3, 1, 6) ? "son" : "no son");

    }
```

```
    public static void main(String[] args) {

        new Equals().show();

    }

}
```

La salida por consola es la siguiente:

```
[3, 4, 5, 6, 7, 8] y [3, 4, 5, 6, 7, 8] son iguales
[3, 4, 5, 6, 7, 8] y [3, 4, 5, 6, 7, 8, 9] no son iguales
[3, 4, 5, 6, 7, 8] y [3, 4, 5, 6, 7, 8, 9] son iguales en el rango 1-5
```

- Método **deepEquals**: compara arrays multidimensionales y devuelve true si son iguales.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class DeepEquals {

    public void show() {

        int[][] array1 = { { 0, 1, 2, 3, 4 }, { 5, 6, 7, 8, 9 }, { 10, 11, 12,
13, 14 } };

        int[][] array2 = { { 0, 1, 2, 3, 4 }, { 5, 6, 7, 8, 9 }, { 10, 11, 12,
13, 14 } };

        int[][] array3 = { { 0, 1, 2, 3, 4 }, { 5, 6, 7, 8, 9 }, { 10, 11, 12,
13, 15 } };

        System.out.printf("%s\n y\n%s\n%s iguales\n\n",
Arrays.deepToString(array1), Arrays.deepToString(array2),
                Arrays.deepEquals(array1, array2) ? "son" : "no son");
        System.out.printf("%s\n y\n%s\n%s iguales\n",
Arrays.deepToString(array1), Arrays.deepToString(array3),
                Arrays.deepEquals(array1, array3) ? "son" : "no son");

    }

    public static void main(String[] args) {

        new DeepEquals().show();

    }

}
```

La salida por consola es la siguiente:

```
[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14]]
 y
[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14]]
son iguales

[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14]]
 y
[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 15]]
no son iguales
```

- Método **compare**: compara dos arrays lexicográficamente. También tiene una versión sobrecargada para especificar un rango. Devuelve:
  - el valor 0: si los dos arrays son iguales y contienen los mismos elementos en el mismo orden.
  - un número negativo: si el primer array es lexicográficamente inferior al segundo.
  - un número positivo: si el primer array es lexicográficamente superior al segundo.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class Compare {

    public void show() {

        int result;
        boolean[] arrayBoolean1 = { true, true, false };
        boolean[] arrayBoolean2 = { true, true, true };
        char[] arrayChar1 = { 'm', 'n', 't' };
        char[] arrayChar2 = { 'm', 'n', 'a' };
        char[] arrayChar3 = { 'M', 'n', 'a' };
        int[] arrayInt1 = { 3, 4, 5 };
        int[] arrayInt2 = { 3, 4, 5 };
        int[] arrayInt3 = { 3, 4, 5, 6 };

        result = Arrays.compare(arrayBoolean1, arrayBoolean2);
        System.out.printf("El resultado de comparar %s y %s es: ",
Arrays.toString(arrayBoolean1),
                Arrays.toString(arrayBoolean2));
        comparison(result);

        result = Arrays.compare(arrayChar1, arrayChar2);
        System.out.printf("El resultado de comparar %s y %s es: ",
Arrays.toString(arrayChar1),
                Arrays.toString(arrayChar2));
        comparison(result);

        result = Arrays.compare(arrayChar3, arrayChar2);
        System.out.printf("El resultado de comparar %s y %s es: ",
Arrays.toString(arrayChar3),
                Arrays.toString(arrayChar2));
        comparison(result);
```

```
        result = Arrays.compare(arrayInt1, arrayInt2);
        System.out.printf("El resultado de comparar %s y %s es: ",
Arrays.toString(arrayInt1),
                Arrays.toString(arrayInt2));
        comparison(result);

        result = Arrays.compare(arrayInt1, arrayInt3);
        System.out.printf("El resultado de comparar %s y %s es: ",
Arrays.toString(arrayInt1),
                Arrays.toString(arrayInt3));
        comparison(result);

        result = Arrays.compare(arrayInt1, 0, 3, arrayInt3, 0, 3);
        System.out.printf("El resultado de comparar %s y %s en el rango 0-2 es:
", Arrays.toString(arrayInt1),
                Arrays.toString(arrayInt3));
        comparison(result);

    }

    public void comparison(int result) {
        if (result == 0) {
            System.out.println("los arrays son iguales");
        } else if (result < 0) {
            System.out.println("el primer array es menor");
        } else {
            System.out.println("el primer array es mayor");
        }
    }

    public static void main(String[] args) {

        new Compare().show();

    }

}
```

La salida por consola es la siguiente:

```
El resultado de comparar [true, true, false] y [true, true, true] es: el primer
array es menor
El resultado de comparar [m, n, t] y [m, n, a] es: el primer array es mayor
El resultado de comparar [M, n, a] y [m, n, a] es: el primer array es menor
El resultado de comparar [3, 4, 5] y [3, 4, 5] es: los arrays son iguales
El resultado de comparar [3, 4, 5] y [3, 4, 5, 6] es: el primer array es menor
El resultado de comparar [3, 4, 5] y [3, 4, 5, 6] en el rango 0-2 es: los arrays
son iguales
```

- Método **mismatch**: encuentra y devuelve el índice de la primera diferencia entre dos arrays. Si no encuentra ninguna diferencia, devuelve -1. También tiene una versión sobrecargada para especificar un rango.

```
package tema6_4_ArraysClass;
```

```
import java.util.Arrays;

public class Mismatch {

    public void show() {

        int[] array1 = { 3, 4, 5, 6, 7, 8 };
        int[] array2 = { 3, 4, 5, 6, 7, 8 };
        int[] array3 = { 3, 4, 5, 6, 7, 8, 9, 10, 11 };
        int indexMismatch;

        indexMismatch = Arrays.mismatch(array1, array2);

        if (indexMismatch == -1) {
            System.out.printf("%s y %s son iguales\n", Arrays.toString(array1),
Arrays.toString(array2));
        } else {
            System.out.printf("%s y %s tienen su primera diferencia en el
elemento %d\n", Arrays.toString(array1),
                    Arrays.toString(array2), indexMismatch);
        }

        indexMismatch = Arrays.mismatch(array1, array3);

        if (indexMismatch == -1) {
            System.out.printf("%s y %s son iguales\n", Arrays.toString(array1),
Arrays.toString(array3));
        } else {
            System.out.printf("%s y %s tienen su primera diferencia en el
elemento %d\n", Arrays.toString(array1),
                    Arrays.toString(array3), indexMismatch);
        }

        indexMismatch = Arrays.mismatch(array1, 2, 6, array3, 2, 6);

        if (indexMismatch == -1) {
            System.out.printf("%s y %s son iguales en el rango 2-5\n",
Arrays.toString(array1),
                    Arrays.toString(array3));
        } else {
            System.out.printf("%s y %s tienen su primera diferencia en el
elemento %d en el rango 2-5\n",
                    Arrays.toString(array1), Arrays.toString(array3),
indexMismatch);
        }

    }

    public static void main(String[] args) {

        new Mismatch().show();

    }

}
```

La salida por consola es la siguiente:

```
[3, 4, 5, 6, 7, 8] y [3, 4, 5, 6, 7, 8] son iguales
[3, 4, 5, 6, 7, 8] y [3, 4, 5, 6, 7, 8, 9, 10, 11] tienen su primera diferencia
en el elemento 6
[3, 4, 5, 6, 7, 8] y [3, 4, 5, 6, 7, 8, 9, 10, 11] son iguales en el rango 2-5
```

6. Copias

- Método **copyOf**: obtiene una copia de un array. Recibe dos parámetros, el array a copiar y el tamaño del array resultante. De modo que:
    - si el tamaño del array resultante es ***menor*** que el tamaño del array original: sólo obtiene copia de los primeros elementos, tantos como indique el tamaño.
    - si el tamaño del array resultante es ***mayor*** que el tamaño del array original: devuelve un array en el que los elementos que superan al original se rellenan con el valor por defecto según el tipo de datos del array.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class CopyOf {

    public void show() {

        int[] array1 = { 3, 4, 5, 6, 7, 8, 9, 10 };
        int[] array2 = Arrays.copyOf(array1, 5);
        int[] array3 = Arrays.copyOf(array1, 10);

        System.out.println(Arrays.toString(array2));//[3, 4, 5, 6, 7]
        System.out.println(Arrays.toString(array3));//[3, 4, 5, 6, 7, 8, 9, 10,
0, 0]

    }

    public static void main(String[] args) {

        new CopyOf().show();

    }

}
```

- Método **copyOfRange**: obtiene una copia de un array especificando el rango, es decir, de qué elemento a qué elemento se hace la copia.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class CopyOfRange {
```

```
    public void show() {

        int[] array1 = { 3, 4, 5, 6, 7, 8, 9, 10 };
        int[] array2 = Arrays.copyOfRange(array1, 2, 6);
        System.out.println(Arrays.toString(array2));//[5, 6, 7, 8]

    }

    public static void main(String[] args) {

        new CopyOfRange().show();

    }

}
```

7. Ordenaciones

Método **sort**: ordena un array en orden ascendente. También tiene una versión sobrecargada para especificar un rango.

```
package tema6_4_ArraysClass;

import java.util.Arrays;

public class Sort {

    public void show() {

        int[] array1 = { 8, 4, 3, 7, 5, 6 };
        int[] array2 = { 8, 4, 3, 7, 5, 6 };

        Arrays.sort(array1);
        System.out.println(Arrays.toString(array1));//[3, 4, 5, 6, 7, 8]

        Arrays.sort(array2, 1, 5);
        System.out.println(Arrays.toString(array2));//[8, 3, 4, 5, 7, 6]

    }

    public static void main(String[] args) {

        new Sort().show();

    }

}
```