9.G. Algoritmos sobre listas y arrays.

3. Algoritmos (III).

¿Qué más ofrece las `clases java.util.Collections` y `java.util.Arrays` de Java? Una vez vista la ordenación, quizás lo más complicado, veamos algunas operaciones adicionales. En los ejemplos, la variable "`array`" es un array y la variable "lista" es una lista de cualquier tipo de elemento:

Operaciones adicionales sobre listas y arrays.		
Operación	**Descripción**	**Ejemplos**
Desordenar una lista.	Desordena una lista, este método no está disponible para arrays.	`Collections.shuffle (lista);`
Rellenar una lista o array.	Rellena una lista o array copiando el mismo valor en todos los elementos del array o lista. Útil para reiniciar una lista o array.	`Collections.fill (lista,elemento);` `Arrays.fill (array,elemento);`
Búsqueda binaria.	Permite realizar búsquedas rápidas en un una lista o array ordenados. Es necesario que la lista o array estén ordenados, si no lo están, la búsqueda no tendrá éxito.	`Collections.binarySearch(lista,elemento);` `Arrays.binarySearch(array, elemento);`

Convertir un array a lista.	Permite rápidamente convertir un array a una lista de elementos, extremadamente útil. No se especifica el tipo de lista retornado (no es `ArrayList` ni `LinkedList`), solo se especifica que retorna una lista que implementa la interfaz `java.util.List`.	`List lista=Arrays.asList(array);`
		Si el tipo de dato almacenado en el array es conocido (`Integer` por ejemplo), es conveniente especificar el tipo de objeto de la lista:
		`List<Integer>lista = Arrays.asList(array);`

Convertir una lista a array.	Permite convertir una lista a array. Esto se puede realizar en todas las colecciones, y no es un método de la clase `Collections`, sino propio de la interfaz `Collection`. Es conveniente que sepas de su existencia.	Para este ejemplo, supondremos que los elementos de la lista son números, dado que hay que crear un array del tipo almacenado en la lista, y del tamaño de la lista:
		`Integer[] array=new Integer[lista.size()];`
		`lista.toArray(array)`

Dar la vuelta.	Da la vuelta a una lista, poniéndola en orden inverso al que tiene.	`Collections.reverse(lista);`

Otra operación que no se ha visto hasta ahora es la dividir una cadena en partes. Cuando una cadena está formada internamente por trozos de texto claramente delimitados por un separador (una coma, un punto y coma o cualquier otro), es posible dividir la cadena y obtener cada uno de los trozos de texto por separado en un array de cadenas. Es una operación sencilla, pero dado que es necesario conocer el funcionamiento de los arrays y de las expresiones regulares para su uso, no se ha podido ver hasta ahora. Para poder realizar esta operación, usaremos el método `split` de la clase `String`. El delimitador o separador es una expresión regular, único argumento del método `split`, y puede ser obviamente todo lo complejo que sea necesario:

```
String texto="Z,B,A,X,M,O,P,U";
```

```
String []partes=texto.split(",");
```

```
Arrays.sort(partes);
```

En el ejemplo anterior la cadena `texto` contiene una serie de letras separadas por comas. La cadena se ha dividido con el método `split`, y se ha guardado cada carácter por separado en un array. Después se ha ordenado el array. ¡Increíble lo que se puede llegar a hacer con solo tres líneas de código!