9.G. Algoritmos sobre listas y arrays.

1. Algoritmos (I).

La palabra algoritmo seguro que te suena, pero, ¿a qué se refiere en el contexto de las colecciones y de otras estructuras de datos? Las colecciones, los arrays e incluso las cadenas, tienen un conjunto de operaciones típicas asociadas que son habituales. Algunas de estas operaciones ya las hemos visto antes, pero otras no. Veamos para qué nos pueden servir estas operaciones:

- Ordenar listas y arrays.
- Desordenar listas y arrays.
- Búsqueda binaria en listas y arrays.
- Conversión de arrays a listas y de listas a array.
- Partir cadenas y almacenar el resultado en un array.

Estos algoritmos están en su mayoría recogidos como métodos estáticos de las `clases java.util.Collections` y `java.util.Arrays`, salvo los referentes a cadenas obviamente.

Los algoritmos de ordenación ordenan los elementos en orden natural, siempre que Java sepa como ordenarlos. Como se explico en el apartado de conjuntos, cuando se desea que la ordenación siga un orden diferente, o simplemente los elementos no son ordenables de forma natural, hay que facilitar un mecanismo para que se pueda producir la ordenación. Los tipos "ordenables" de forma natural son los enteros, las cadenas (orden alfabético) y las fechas, y por defecto su orden es ascendente.

La clase `Collections` y la clase `Arrays` facilitan el método `sort`, que permiten ordenar respectivamente listas y arrays. Los siguientes ejemplos ordenarían los números de forma ascendente (de menor a mayor):

Ordenación natural en listas y arrays.	
Ejemplo de ordenación de un array de números	**Ejemplo de ordenación de una lista con números**
	`ArrayList<Integer> lista=new ArrayList<Integer>();`
`Integer[] array={10,9,99,3,5};`	`lista.add(10); lista.add(9);lista.add(99);`
`Arrays.sort(array);`	`lista.add(3); lista.add(5);`
	`Collections.sort(lista);`