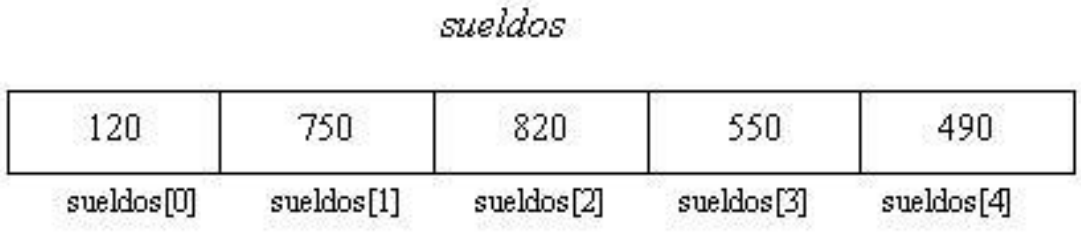
# MISCELÁNEA DE EJERCICIOS EJE TEMÁTICO 4 COLECCIONES EN JAVA

1. Se desea almacenar los sueldos de operarios. Cuando se ejecuta el programa se debe pedir la cantidad de sueldos a ingresar. Luego crear un vector con dicho tamaño.

2. Desarrollar un programa que permita ingresar un vector de n elementos, ingresar n por teclado. Luego imprimir la suma de todos sus elementos

Es una actividad común la búsqueda del mayor y menor elemento de un vector, lo mismo que su posición.

3. Confeccionar un programa que permita cargar los nombres de 5 operarios y sus sueldos respectivos. Mostrar el sueldo mayor y el nombre del operario.




El mayor elemento es el 820 y se encuentra en la posición nº 2.

4. Cargar un vector de n elementos. imprimir el menor y un mensaje si se repite dentro del vector.

Se puede ordenar tanto vectores con componentes de tipo int, float como String. En este último caso el ordenamiento es alfabético.

5. Se debe crear un vector donde almacenar 5 sueldos. Ordenar el vector sueldos de menor a mayor.

6. Definir un vector donde almacenar los nombres de 5 paises. Confeccionar el algoritmo de ordenamiento alfabético.

7. Cargar un vector de n elementos de tipo entero. Ordenar posteriormente el vector

## INTERFACE LIST

Vamos a continuar el estudio de la interface List del api de Java, pero esta vez usaremos la clase LinkedList como forma de instanciar la interface. También veremos las características más importantes de LinkedList, las diferencias que tiene con ArrayList y haremos un ejemplo a modo de ejercicio.


## LINKEDLIST

La clase LinkedList implementa la interface List. Eso quiere decir que tendrá una serie de métodos propios de esta interface y comunes a todas las implementaciones. Así utilizando siempre que se pueda declaración de objetos del tipo definido por la interface podemos cambiar relativamente fácil su implementación (por ejemplo pasar de ArrayList a LinkedList y viceversa) y conseguir mejoras en el rendimiento de nuestros programas con poco esfuerzo.

Ahora centrándonos en la clase LinkedList, ésta se basa en la implementación de listas doblemente enlazadas. Esto quiere decir que la estructura es un poco más compleja que la implementación con ArrayList, pero… ¿Qué beneficios nos aporta si la estructura es más compleja?.

¿Rapidez?, pues no mucha la verdad, de hecho ArrayList es la favorita para realizar búsquedas en una lista y podríamos decir que ArrayList es más rápida para búsquedas que LinkedList. Entonces, ¿qué interés tiene LinkedList?. Si tenemos una lista y lo que nos importa no es buscar la información lo más rápido posible, sino que la inserción o eliminación se hagan lo más rápidamente posible, LinkedList resulta una implementación muy eficiente y aquí radica uno de los motivos por los que es interesante y por los que esta clase se usa en la programación en java.

La clase Linkedlist no permite posicionarse de manera absoluta (acceder directamente a un elemento de la lista) y por tanto no es conveniente para búsquedas pero en cambio sí permite una rápida inserción al inicio/final de la lista y funciona mucho más rápido que Arraylist por ejemplo para la posición 0 (imaginemos que en un ArrayList deseamos insertar en la posición 0 y tenemos muchos elementos, no solo tendremos que realizar la operación de insertar este nuevo elemento en la posición 0 sino que tendremos que desplazar el elemento que teníamos de la posición 0 a la 1, el que estaba anteriormente en la posición 1 a la 2 y así sucesivamente… Si tenemos un gran número de elementos la operación de inserción es más lenta que en la implementación LinkedList, donde el elemento simplemente "se enlaza" al principio, sin necesidad de realizar desplazamientos en cadena de todos los demás elementos).

**NO SÉ QUE IMPLEMENTACION UTILIZAR ¿ARRAYLIST O LINKEDLIST?**

Es posible que en un programa tengamos que usar una lista pero no sepamos si es más conveniente usar ArrayList ó LinkedList. En este caso podemos hacer lo siguiente:

Si a priori pensamos que es mejor utilizar una implementación con ArrayList porque pensamos que las búsquedas van a ser la mayoría de las operaciones, entonces pondremos algo así: List listaObjetos = new ArrayList();

Por el contrario si pensamos a priori que la mayoría de las operaciones sobre esta lista de objetos van a ser inserciones o eliminaciones sobre listas grandes escribiremos: List listaObjetos = new LinkedList();

Aquí queda reflejada la utilidad que tiene el uso de interfaces, porque usando los mismos métodos, podemos tener varias implementaciones que nos permitirán un mejor rendimiento dependiendo del uso que demos a nuestra aplicación.

**EJERCICIO COMPARATIVO ARRAYLIST VS LINKEDLIST**

Vamos a realizar a continuación un ejercicio comparativo, donde vamos a observar y medir el tiempo que tarda en realizarse una inserción en una lista ArrayList y en otra lista LinkedList.

8. Se pide implementar una lista que gestione la lista de espera de un taller mecánico. El taller tendrá una lista de fichas de vehículos que serán dados de alta en el momento de su llegada al taller, donde se le asignará la hora de entrada. Una vez resuelto el problema del coche se le asignará una fecha de resolución de la avería y sólo cuando el técnico venga a buscarlo, se le dará una fecha de salida y el borrado de la lista de espera. Es recomendable crear tres listas, una Para los no arreglados, otra para los arreglados y están pendientes de retirar.

9. Vamos con el ejemplo de los ArrayList de java. Vamos a hacer un ejemplo sencillo con la baraja española.

10. Vamos a crear una agenda telefónica en la que vamos a vincular al nombre del contacto, el número de teléfono. La estructura es clara, el nombre será la clave, mientras que el número de teléfono será el valor. Se empleará map