Hoy vamos a continuar trabajando con el **practicoDeHerencia**:
1) Vamos a crear 5 o más clientes y 5 o más administradores y los agregaremos a nuestro arrayList. Recordas cómo se hace?

2) Cómo harías para mostrar la cantidad de elementos que agregaste a arrayList?
"La cantidad de elementos del arraylist es:"

3) Cómo harías para encontrar el cliente y el administrador que tienen más años?

4) Ahora necesitamos averiguar si el Cliente con el nombre "Homero" existe en nuestro arraylist. Si existe mostrar los datos del cliente/Administrador , indicar si es administrador o cliente. En caso de que no exista mostrar un mje de que no existe esa persona en nuestros registros.

Ayuda: podemos usar el método contains(X) del arraylist
Donde x es reemplazado por el objeto a buscar.

5) Sigamos aprendiendo de ArrayList:
Te animas a eliminar al segundo cliente que agregaste de la lista pero usando el método remove de arraylist? y volver a mostrar todos los nombres de los clientes /administradores que quedaron en el arraylist.

6) El Método indexOf, es útil para saber la posición de un elemento en particular, sabiendo esto cómo harías para saber en qué posición está el cliente de mayor edad? y mostrarla por pantalla.

RECORRER UN ARRAYLIST

Podemos recorrerlo de forma clásica con un **bucle for:**

```
for(int i = 0;i<array.size();i++){
        System.out.println(array.get(i));
}
```

Con un **bucle foreach:**
Si suponemos el array de enteros llamado *numeros:*

```
for(Integer i: numeros){
        System.out.println(i);
}
```

Si el array contiene objetos de tipos distintos o desconocemos el tipo:

```
for(Object o: nombreArray){
        System.out.println(o);
}
```

Utilizando un **objeto Iterator**.

http://docs.oracle.com/javase/9/docs/api/java/util/Iterator.html

La ventaja de utilizar un Iterador es que no necesitamos indicar el tipo de objetos que contiene el array.

Iterator tiene como métodos:

hasNext: devuelve true si hay más elementos en el array.

next: devuelve el siguiente objeto contenido en el array.
Ejemplo:

```
ArrayList<Integer> numeros = new ArrayList();
…..
//se insertan elementos
.....

Iterator it = numeros.iterator(); //se crea el iterador it para el array numeros
while(it.hasNext())                    //mientras queden elementos
      System.out.println(it.next());  //se obtienen y se muestran
```

7) Te animas a modificar o agregar otro método para buscar el menor de edad pero esta vez utilizando Iterator.???

8) Para terminar borra todos los elementos del arraylist usando el método clear y mostrar la cantidad de elementos que tiene ahora el arraylist. y si el resultado es cero mostrar un cartel indicando que el arraylist está vacío.El método isEmpty, indica si un arraylist esta vacio o no.

9) Algo muy común a la hora de programar, es como avisar al usuario de ciertas actividades, o como hacerlo escoger para recapacitar o darle la oportunidad de elegir que es lo que se desea hacer, para esto, en Java tenemos elementos muy sencillos pero funcionales, que nos permiten mostrar mensajes de dialogo, o cuadros de dialogo, como los prefieran llamar, estos elementos son sumamente sencillos de utilizar con un poco de lectura al API de Java
Los cuadros de dialogo que veremos estarán basados en el objeto JOptionPane, un objeto de java que nos permite precisamente trabajar con cuadros de dialog, los métodos de este objeto que veremos, serán los siguientes:
showMessageDialog
showInputDialog
showConfirmDialog
showOptionDialog

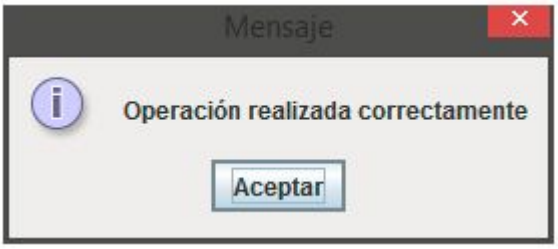
Para acceder a estos método simplemente tenemos que importar la clase JOptionPane:

```
import javax.swing.JOptionPane;
```

Una vez importado lo único que tenemos que hacer es teclear el nombre de la clase más el método que queremos utilizar, teniendo en cuenta una cosa, el método showMessageDialog tiene 3 sobrecargas del método, uno que recibe 2 parametros, otro de 4, y otro de 5

Primero veremos es que usa dos parámetros:
JOptionPane.showMessageDialog(null, "Operación realizada correctamente");

El primer parámetro, representa el componente padre sobre el cual el mensaje se mostrará, si nosotros no le enviamos ninguno, como en este caso lo estamos haciendo, simplemente mostrará una ventana similar a la siguiente:



Si nosotros le enviamos un componente, se colocará encima de el, sin ninguna variación en el funcionamiento, el segundo parámetro obviamente, es el mensaje que queremos observar, y listo, un mensaje fácil de hacer, bien, ahora veamos otra cosa, el anterior fue el método con dos parámetros, ahora bien, cuando utilizamos el método de cuatro parámetros tenemos un poco más de personalización, veamos:

JOptionPane.showMessageDialog(null, "Mensaje dentro de la ventana", "Mensaje en la barra de titulo", JOptionPane.WARNING_MESSAGE);

El funcionamiento de este método es un poco diferente, pero más útil:
Primer Parámetro: El componente padre sobre el cual se mostrará la ventana de dialogo.
Segundo Parámetro: El mensaje que se mostrará dentro de la ventana.
Tercer Parámetro: El mensaje que se mostrará en la barra de titulo.
Cuarto Parámetro: Una variable Int contenida por JOptionPane, que representa el icono que se mostrará en la ventana, algunos de los valores posibles son: INFORMATION_MESSAGE , WARNING_MESSAGE , QUESTION_MESSAGE , PLAIN_MESSAGE, ERROR_MESSAGE
En el caso anterior, el mensaje se vería de esta manera:

Vamos a practicar….
Modifica los diferentes System.out.print por showMessageDialog.