A continuación, te proporcionamos algunos ejemplos que ilustran cómo utilizar los métodos más comunes en un ArrayList en Java.

Agregar elementos (add()):

```
import java.util.ArrayList;

public class App {
    public static void main(String[] args) {
        // Crear un ArrayList de tipo String
        ArrayList<String> lista = new ArrayList<>();

        // Agregar elementos a la lista
        lista.add("Manzana");
        lista.add("Banana");
        lista.add("Naranja");

        // Imprimir la lista
        System.out.println("Lista después de agregar elementos: " + lista);
    }
}
```

Eliminar elementos (remove()):

```
import java.util.ArrayList;
public class App {
    public static void main(String[] args) {
        // Crear un ArrayList de tipo Integer
        ArrayList<Integer> numeros = new ArrayList<>();

        // Agregar elementos a la lista
        numeros.add(10);
        numeros.add(20);
        numeros.add(30);
        // Imprimir la lista original
        System.out.println("Lista después de agregar elementos: " +
numeros);

        // Eliminar un elemento por índice
        numeros.remove(1); // Elimina el elemento en el índice 1 (20)

        // Imprimir la lista después de eliminar un elemento
        System.out.println("Lista después de eliminar un elemento: " +
numeros); }
}
```

Obtener un elemento por índice (get()):

```
import java.util.ArrayList;

public class App {
    public static void main(String[] args) {

        // Crear un ArrayList de tipo Double
        ArrayList<Double> precios = new ArrayList<>();

        // Agregar elementos a la lista
        precios.add(10.5);
        precios.add(20.75);
        precios.add(30.0);

        // Obtener un elemento por índice
         double precio = precios.get(1); // Obtiene el elemento en el índice
1 (20.75)

        // Imprimir el precio obtenido
        System.out.println("Precio obtenido: " + precio);
    }
}
```

Comprobar si el ArrayList está vacío (is Empty()):

```
import java.util.ArrayList;
import java.util.Arrays;

public class App {
    public static void main(String[] args) {
        // CASO DE LISTA VACIA.
        ArrayList<String> lista = new ArrayList<>();

        // Verificar si la lista está vacía
        if (lista.isEmpty()) {
            System.out.println("La lista 1 está vacía.");
        } else {
            System.out.println("La lista 1 no está vacía.");
        }

          // // Crear e inicializar lista2 con elementos en una sola línea
utilizando Arrays.asList
        ArrayList<String> lista2 = new
ArrayList<>(Arrays.asList("elemento1", "elemento2", "elemento3"));



// Verificar si la lista no está vacía
        if (!lista2.isEmpty()) {
            System.out.println("La lista 2 no está vacía.");

            // Mostrar el primer elemento de la lista
            String primerElemento = lista2.get(0);
                  System.out.println("Primer elemento de la lista 2: " +
primerElemento);
```

```
        } else {
            System.out.println("La lista 2 está vacía.");
        }
    }

}
```

Conocer número de elementos (size()):

```
import java.util.ArrayList;

public class App {
    public static void main(String[] args) {
        ArrayList<String> lista = new ArrayList<>();

        // Agregar elementos a la lista
        lista.add("Manzana");
        lista.add("Banana");
        lista.add("Naranja");

         // Obtener el tamaño de la lista - // El método size() devuelve el
número total de elementos almacenados en la lista.

        int tamaño = lista.size();
        System.out.println("El tamaño de la lista es: " + tamaño);
    }
}
```

Ordenar elementos(sort():

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class App {
    public static void main(String[] args) {
        ArrayList<String> lista = new ArrayList<>();

        // Agregar elementos desordenados a la lista
        lista.add("Manzana");
        lista.add("Banana");
        lista.add("Naranja");

           // Ordenar la lista en orden natural (alfabético en este caso),
ascendente
        Collections.sort(lista);

        // Mostrar la lista ordenada
        System.out.println("Lista ordenada de String:");
        for (String elemento : lista) {
            System.out.print(elemento + " -");
        }
        System.out.println();
```

```
        // Crear un ArrayList de números
        ArrayList<Integer> numeros = new ArrayList<>();
       // Agregar números desordenados a la lista


        numeros.add(5);
        numeros.add(2);
        numeros.add(8);
        numeros.add(1);
        numeros.add(9);

        // Ordenar en orden ascendente
        Collections.sort(numeros);
        System.out.println("Orden ascendente lista de números: " + numeros);

        // Ordenar en orden descendente
        Collections.sort(numeros, Comparator.reverseOrder());
              System.out.println("Orden descendente lista de números: " +
numeros);
    }
}
```

Veamos un ejemplo con colecciones de objetos:

Te invito a revisar un ejemplo que incluye la clase Persona y una clase ejecutable en un proyecto. Si lo deseas, puedes copiar el código a tu editor de código para probar su funcionamiento.

```
public class Persona {
    private int documento;
    private String nombre;
    private String apellido;
    //CONSTRUCTOR CON PARAMETROS, GETTERS Y SETTERS
    public Persona(int documento, String nombre, String apellido) {
        this.documento = documento;
        this.nombre = nombre;
        this.apellido = apellido;
    }
    public int getDocumento() {
        return documento;
    }
    public String getNombre() {
        return nombre;
    }
    public String getApellido() {
        return apellido;
    }
    public void setDocumento(int documento) {
        this.documento = documento;
    }
    public void setNombre(String nombre) {
```

```
        this.nombre = nombre;
    }
    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
}
import java.util.ArrayList;

public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("********PRUEBA CON ARRAYLIST (Crear y
eliminar********");

        // Creamos 5 objetos del tipo persona, haciendo uso del constructor
con
        // parametros
        Persona p1 = new Persona(111111111, "Nombre 1", "Apellido 1");
        Persona p2 = new Persona(222222222, "Nombre 2", "Apellido 2");
        Persona p3 = new Persona(333333333, "Nombre 3", "Apellido 3");
        Persona p4 = new Persona(444444444, "Nombre 4", "Apellido 4");
        Persona p5 = new Persona(555555555, "Nombre 5", "Apellido 5");

        /************ CREO UN ARRAYLIST ****************/
        ArrayList<Persona> nuevoArreglo = new ArrayList<>();
        /*************
         * AGREGO ELEMENTOS, DEL TIPO "PERSONA" AL ARREGLO YA CREADO
         ****************/
        nuevoArreglo.add(p1); // Paso como parametro el objeto a almcenar
        nuevoArreglo.add(p2); // Paso como parametro el objeto a almcenar
        nuevoArreglo.add(p3); // Paso como parametro el objeto a almcenar
        nuevoArreglo.add(p4); // Paso como parametro el objeto a almcenar
        nuevoArreglo.add(p5); // Paso como parametro el objeto a almcenar
        nuevoArreglo.add(p1); // Como este tipo de coleccion acepta
duplicados, "paso nuevamente persona 1"
        /************ INVOCO AL METODO PARA CONOCER TAMAÑO ARREGLO
****************/

        System.out.println("En esta instancia, el arraylist tiene un tamaño
de: " + obtenerTamaño(nuevoArreglo));
        /************ INVOCO AL METODO PARA IMPRIMIR ARREGLO
****************/
        imprimirLista(nuevoArreglo);
        /************ ELIMINO UN OBJETO DE MI ARREGLO ****************/
        nuevoArreglo.remove(p3); // Opción 1: En este caso, pase el objeto a
eliminar
        nuevoArreglo.remove(0); // Opción 2:En este caso, pase el indice del
objeto a eliminar
        /************ INVOCO AL METODO PARA CONOCER TAMAÑO ARREGLO
****************/
        /*
         * Notamos que ahora se redujo en 2, ya que elimine 2 elementos de
mi colección
         */
        System.out.println("En esta instancia, el arraylist tiene un tamaño
de: " + obtenerTamaño(nuevoArreglo));
        /************ INVOCO AL METODO PARA IMPRIMIR ARREGLO
****************/
        imprimirLista(nuevoArreglo);
```

```
        System.out.println("********PRUEBA CON ARRAYLIST
(Concatenar)********");
        /************ CREO UN NUEVO ARRAYLIST ****************/
        System.out.println("Creando nuevo arreglo, y agregando elementos
....");
        ArrayList<Persona> otroArreglo = new ArrayList<>();
        Persona p6 = new Persona(666666666, "Nombre 6", "Apellido 6");
        Persona p7 = new Persona(777777777, "Nombre 7", "Apellido 7");
        /*************
         * AGREGO ELEMENTOS, DEL TIPO "PERSONA" AL ARREGLO YA CREADO
         *****************/
        otroArreglo.add(p6); // Paso como parametro el objeto a almcenar
        otroArreglo.add(p7); // Paso como parametro el objeto a almcenar
        /* Agrego a nuevoArreglo, la lista otroArreglo */
        nuevoArreglo.addAll(otroArreglo);// Envio como parametro la
colección
        /************ INVOCO AL METODO PARA CONOCER TAMAÑO ARREGLO
****************/
        System.out.println("En esta instancia, el arraylist tiene un tamaño
de: " + obtenerTamaño(nuevoArreglo));
        /************ INVOCO AL METODO PARA IMPRIMIR ARREGLO
****************/
        imprimirLista(nuevoArreglo);
        /************
         * Busco un Objeto puntual, para saber si existe. En este caso "p3"
que lo habia
         * eliminado
         *****************/
        System.out.println("¿El objeto P3 se encuentra en la lista:? " +
encontrarObjeto(nuevoArreglo, p3));
        /************
         * Busco un Objeto puntual, para conocer su indice. En este caso
"p2" que lo
         * habia eliminado
         *****************/
        /* Si retorna -1, significa que el elemento NO FUE ENCONTRADO */
        System.out.println(
                "¿El objeto P2 en que posicion de la lista se encuentra:? "
+ posicionObjeto(nuevoArreglo, p2));
        /************
         * INVOCO AL METODO PARA ver si EL ARREGLO ESTA VACIO
         *****************/
        System.out.println("¿El arreglo se encuentra vacio:? " +
estaVacia(nuevoArreglo));
        /* Vaciar la lista, */
        nuevoArreglo.clear();
        System.out.println("¿El arreglo se encuentra vacio:? " +
estaVacia(nuevoArreglo));

    }

    // Método para conocer tamaño del arreglo
    public static int obtenerTamaño(ArrayList<Persona> lista) {
        return lista.size();
    }

    // Método para imprimir el arreglo
```

```
    public static void imprimirLista(ArrayList<Persona> lista) {
        System.out.println("Elementos del ArrayList:");
        for (Persona elemento : lista) {
            System.out.println(elemento.getApellido() + " , " +
elemento.getNombre() + " , " + elemento.getDocumento());
        }

System.out.println("-----------------------------------------------");
    }

    // Método para conocer si un elemento esta en la lista
    public static boolean encontrarObjeto(ArrayList<Persona> lista, Persona
objeto) {
        return lista.contains(objeto);
    }

    // Método para conocer posicion de un objeto en la lista
    public static int posicionObjeto(ArrayList<Persona> lista, Persona
objeto) {
        return lista.indexOf(objeto);
    }

    // Método para conocer si una lista esta vacia.
    public static boolean estaVacia(ArrayList<Persona> lista) {
        return lista.isEmpty();
    }
}
```

💡Siempre es recomendable utilizar la **documentación oficial** para obtener más detalles sobre el material proporcionado, su uso y su implementación. Puedes acceder a la información sobre Collections Framework desde **aquí**.