Guía Completa sobre Listas en Java

1. ¿Qué es una Lista en Java?

Una lista (List) en Java es una estructura de datos dinámica que permite almacenar una colección de elementos ordenados. A diferencia de los arrays, las listas pueden cambiar de tamaño dinámicamente (agregar o eliminar elementos sin definir una longitud fija).

Las listas forman parte del paquete:

```
import java.util.*;
```

2. Jerarquía de las Listas

Las listas son parte del framework de colecciones de Java (java.util.Collection):

```
Collection
   └── List
        ├── ArrayList
        ├── LinkedList
        ├── Vector
        └── Stack
```

3. Tipos de Listas más comunes

ArrayList

- Basada en un array dinámico.
- Permite acceso rápido a los elementos.
- Ideal para búsquedas y lecturas frecuentes.
- No es sincronizada (no segura para hilos).

```
import java.util.ArrayList;

ArrayList<String> nombres = new ArrayList<>();
nombres.add("Ana");
nombres.add("Carlos");
nombres.add("Pedro");
System.out.println(nombres);
```

LinkedList

- Basada en una lista doblemente enlazada.
- Ideal para inserciones y eliminaciones frecuentes.
- Acceso más lento que ArrayList.

```
import java.util.LinkedList;

LinkedList<Integer> numeros = new LinkedList<>();
numeros.add(10);
numeros.add(20);
numeros.addFirst(5); // inserta al inicio
System.out.println(numeros);
```

Vector

Similar a ArrayList, pero sincronizado (seguro para múltiples hilos).

- Es más lento en operaciones simples por su sincronización.

```
import java.util.Vector;

Vector<Double> precios = new Vector<>();
precios.add(12.5);
precios.add(7.99);
System.out.println(precios);
```

Stack

- Subclase de Vector.
- Sigue el principio LIFO (Last In, First Out).
- Usado como pila.

```
import java.util.Stack;

Stack<String> pila = new Stack<>();
pila.push("A");
pila.push("B");
pila.push("C");
System.out.println(pila.pop()); // Elimina y devuelve "C"
```

4. Métodos comunes de List

Método	Descripción	Ejemplo

Método	Descripción	Ejemplo
`add(elemento)`	Agrega un elemento	`lista.add("Juan");`
`add(index, elemento)`	Inserta en una posición específica	`lista.add(1, "Luis");`
`get(index)`	Obtiene un elemento por índice	`lista.get(0);`
`set(index, elemento)`	Modifica un elemento existente	`lista.set(1, "Maria");`
`remove(index)`	Elimina un elemento por posición	`lista.remove(0);`
`size()`	Devuelve el tamaño de la lista	`lista.size();`
`contains(obj)`	Verifica si contiene un elemento	`lista.contains("Ana");`
`clear()`	Vacía la lista	`lista.clear();`
`isEmpty()`	Verifica si está vacía	`lista.isEmpty();`

5. Recorrer una Lista

- Con un for clásico:

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

- Con un for-each:

```
for (String nombre : nombres) {
    System.out.println(nombre);
}
```

- Con Iterator:

```
import java.util.Iterator;

Iterator<String> it = nombres.iterator();
while (it.hasNext()) {
    System.out.println(it.next());
}
```

- Con forEach (Java 8+):

```
nombres.forEach(System.out::println);
```

6. Listas de Objetos

Puedes crear listas de tus propias clases:

```
class Persona {
    String nombre;
    int edad;

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public String toString() {
        return nombre + " (" + edad + ")";
    }
}

import java.util.ArrayList;

ArrayList<Persona> personas = new ArrayList<>();
personas.add(new Persona("Ana", 20));
personas.add(new Persona("Luis", 25));

for (Persona p : personas) {
    System.out.println(p);
}
```

7. Ordenar una Lista

- Con Collections.sort():

```
import java.util.*;

ArrayList<String> frutas = new ArrayList<>(List.of("Pera", "Manzana", "Banano"));
Collections.sort(frutas);
System.out.println(frutas);
```

- Ordenar objetos personalizados:

```
Collections.sort(personas, Comparator.comparing(p -> p.nombre));
```

8. Convertir entre Array y List

- De Array a List

```
String[] colores = {"Rojo", "Verde", "Azul"};
List<String> listaColores = Arrays.asList(colores);
```

- De List a Array

```
String[] nuevoArray = listaColores.toArray(new String[0]);
```

9. Ejemplo completo

```
import java.util.*;

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

        estudiantes.add("Carlos");
        estudiantes.add("Ana");
        estudiantes.add("Luis");

        System.out.println("Lista inicial: " + estudiantes);

        estudiantes.remove("Ana");
        System.out.println("Después de eliminar: " + estudiantes);

        estudiantes.add(1, "María");
        System.out.println("Después de insertar: " + estudiantes);

        Collections.sort(estudiantes);
        System.out.println("Lista ordenada: " + estudiantes);

        for (String e : estudiantes) {
            System.out.println("Estudiante: " + e);
        }
    }
}
```

10. Buenas Prácticas

- Usa ArrayList si realizas búsquedas frecuentes.
- Usa LinkedList si realizas muchas inserciones o eliminaciones.
- Declara las listas con la interfaz List:

```
List<String> lista = new ArrayList<>();
```

- Evita usar Vector y Stack en proyectos nuevos; usa ArrayList o Deque (ArrayDeque).