Apuntes Java - Listas

Profesor: Luis Javier López López

Índice

Listas en Java

Introducción

En Java, una lista es una interfaz que forma parte del framework de colecciones **(java.util**), y representa una secuencia ordenada de elementos. La interfaz **List** extiende la interfaz **Collection**, lo que significa que hereda métodos generales de manipulación de colecciones y también proporciona operaciones específicas para manipular listas de manera eficiente.

Algunas características clave de las listas en Java son:

1. **Orden**: Los elementos en una lista están ordenados por índice, lo que significa que puedes acceder a ellos en un orden específico utilizando un índice numérico.
2. **Permite duplicados**: A diferencia de algunos tipos de colecciones, las listas pueden contener elementos duplicados. Cada elemento en la lista tiene un índice único que lo identifica.
3. **Acceso por índice**: Puedes acceder a los elementos de una lista mediante su índice. El índice comienza desde 0 para el primer elemento y aumenta secuencialmente.
4. **Tamaño dinámico**: Las implementaciones de la interfaz **List** en Java, como **ArrayList** y **LinkedList**, permiten un tamaño dinámico, es decir, puedes agregar o eliminar elementos de la lista sin tener que especificar su tamaño previamente.

Implementaciones Comunes

Las implementaciones más comunes de la interfaz List en Java son:

- **ArrayList**: Implementada como un array dinámico, lo que significa que su tamaño puede cambiar dinámicamente al agregar o quitar elementos.

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

- **LinkedList**: Implementada como una lista doblemente enlazada, lo que permite una rápida inserción y eliminación de elementos, pero con un acceso más lento por índice en comparación con ArrayList.

```
List<String> linkedList = new LinkedList<>();
```

- **Vector**: Similar a ArrayList, pero es sincronizado, lo que significa que es seguro para operaciones en entornos concurrentes. Sin embargo, su uso se ha vuelto menos común debido a la sincronización, que puede afectar el rendimiento.

```
List<String> vector= new Vector<>();
```

Principales Métodos

Estos son algunos de los métodos más comunes de la interfaz **List** en Java:

- **add**(E elemento) y **add**(int índice, E elemento):
  - **add**(E elemento) agrega el elemento al final de la lista.
  - **add**(int índice, E elemento) inserta el elemento en la posición especificada por el índice.
- **addAll**(Collection<? extends E> colección):Añade todos los elementos de la colección dada al final de la lista.

- **remove**(Object objeto) y remove(int índice):
  - **remove**(Object objeto) elimina la primera ocurrencia del objeto dado en la lista.
  - **remove**(int índice) elimina el elemento en la posición especificada por el índice.
- **removeAll**(Collection<?> colección):Elimina de la lista todos los elementos que están presentes en la colección dada.

- **get**(int índice):Devuelve el elemento en la posición especificada por el índice.

- **set**(int índice, E elemento):Reemplaza el elemento en la posición especificada por el índice con el nuevo elemento.

- **indexOf**(Object objeto):Devuelve el índice de la primera ocurrencia del objeto dado en la lista, o -1 si no está presente.

- **size**():Devuelve el número de elementos en la lista.

- **isEmpty**():Devuelve true si la lista no contiene elementos.

- **clear**():Elimina todos los elementos de la lista.

- **contains**(Object objeto):Devuelve true si la lista contiene el objeto especificado.
- **containsAll**(Collection<?> colección):Retorna true si la lista contiene todos los elementos de la colección dada. En otras palabras, verifica si cada elemento de la colección está presente al menos una vez en la lista.

- **retainAll**(Collection<?> colección):Retiene sólo los elementos de la lista que están presentes en la colección dada, eliminando los demás.

- **subList**(int desde, int hasta):Devuelve una vista de la lista desde la posición desde (inclusive) hasta la posición hasta (exclusive).

- **sort**(Comparator<? super E> comparador):Ordena los elementos de la lista utilizando el comparador proporcionado.

Ejemplos

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.Collection;
import java.util.Collections;

public class EjemploListaCompleto {
    public static void main(String[] args) {
        // Crear una lista
        List<String> listaDeNombres = new ArrayList<>();

        // Añadir elementos
        listaDeNombres.add("Alice");
        listaDeNombres.add("Bob");
        listaDeNombres.add("Charlie");

        // Imprimir la lista original
        System.out.println("Lista original:");
        imprimirLista(listaDeNombres);

        // Acceder a un elemento por índice
        String primerNombre = listaDeNombres.get(0);
        System.out.println("Primer nombre: " + primerNombre);

        // Modificar un elemento
        listaDeNombres.set(1, "Bobby");
        System.out.println("Lista después de modificar 'Bob' a 'Bobby':");
        imprimirLista(listaDeNombres);

        // Verificar la presencia de un elemento
        boolean contieneAlice = listaDeNombres.contains("Alice");
        System.out.println("¿Contiene 'Alice'? " + contieneAlice);

        // Eliminar un elemento por índice
        listaDeNombres.remove(2);
        System.out.println("Lista después de eliminar el tercer nombre:");
        imprimirLista(listaDeNombres);

        // Obtener el tamaño de la lista
        int tamaño = listaDeNombres.size();
        System.out.println("Tamaño de la lista: " + tamaño);

        // Crear una sublista
        List<String> sublista = listaDeNombres.subList(0, 2);
        System.out.println("Sublista desde 0 hasta 2 (exclusive):");
        imprimirLista(sublista);
```

```
        // Crear una copia de la lista
        List<String> copia = new ArrayList<>(listaDeNombres);
        System.out.println("Copia de la lista original:");
        imprimirLista(copia);

        // Verificar si dos listas son iguales
        boolean sonIguales = listaDeNombres.equals(copia);
        System.out.println("¿Son iguales la lista original y su copia? " + sonIguales);

        // Ordenar la lista
        Collections.sort(listaDeNombres);
        System.out.println("Lista después de ordenar alfabéticamente:");
        imprimirLista(listaDeNombres);

        // Limpiar la lista
        listaDeNombres.clear();
        System.out.println("Lista después de limpiar:");
        imprimirLista(listaDeNombres);
    }

    // Método para imprimir una lista
    private static void imprimirLista(List<String> lista) {
        for (String elemento : lista) {
            System.out.println(elemento);
        }
        System.out.println();
    }
}
```