LISTAS MUTABLES E INMUTABLES

Característica	`List` (Inmutable)	`MutableList`	`ArrayList`	`LinkedList`
Declaración	`val lista: List<T> = listOf(...)`	`val lista: MutableList<T> = mutableListOf(...)`	`val arrayList: ArrayList<T> = arrayListOf(...)`	`val linkedList: LinkedList<T> = LinkedList()`
Inicialización Vacía	`listOf<T>()`	`mutableListOf<T>()`	`arrayListOf<T>()`	`LinkedList<T>()`
Inicialización con Datos	`listOf("a", "b", "c")`	`mutableListOf("a", "b", "c")`	`arrayListOf("a", "b", "c")`	`LinkedList(listOf("a", "b", "c"))`
Mutabilidad	Inmutable	Mutable	Mutable	Mutable
Agregar/ Eliminar	No	`add`, `remove`	`add`, `remove`	`add`, `remove`
Acceso a Índice	Sí	Sí	Sí	Sí
Acceso Rápido (`O(1)`)	Sí	Sí	Sí	No
Inserción Rápida en Extremos	No	No	No	Sí
Ideal para	Listas constantes	Datos cambiantes	Acceso rápido por índice	Inserción/eliminación en extremos
Estilo Kotlin Idiomático	Muy idiomático	Muy idiomático	Idiomático	Menos idiomático (requiere import)
Conversión	`toMutableList()`	`toList()`	Compatible con `MutableList`	`toMutableList()` / `toList()`

Ejemplo de Uso de Cada Tipo

1. **`List` (Inmutable)**: Ideal para listas de solo lectura.

   ```
   kotlin
   Copiar código
   val lista: List<String> = listOf("a", "b", "c")
   println(lista[0]) // Acceso a elementos permitido
   ```

2. **`MutableList`**: Permite modificar la lista (agregar y eliminar elementos).

   ```
   kotlin
   Copiar código
   val listaMutable: MutableList<String> = mutableListOf("a", "b", "c")
   listaMutable.add("d")   // Agrega un elemento
   listaMutable.remove("a") // Elimina un elemento
   ```

3. **`ArrayList`**: Útil para listas de tamaño variable con acceso rápido a través de índices.

```
kotlin
Copiar código
val arrayList: ArrayList<String> = arrayListOf("a", "b", "c")
arrayList.add("d")
arrayList[1] = "nuevo valor" // Modificación mediante índice
```

4. **`LinkedList`**: Buena opción para listas con inserciones y eliminaciones frecuentes en los extremos.

```
kotlin
Copiar código
import java.util.LinkedList
val linkedList: LinkedList<String> = LinkedList(listOf("a", "b", "c"))
linkedList.addFirst("inicio") // Agregar al inicio
linkedList.addLast("fin")     // Agregar al final
linkedList.remove("a")
```

Resumen de Cuándo Usar Cada Uno

- **`List` (Inmutable)**: Cuando los datos no deben cambiar después de su inicialización.
- **`MutableList`**: Cuando necesitas una lista modificable sin una implementación específica.
- **`ArrayList`**: Si necesitas acceso rápido a través de índices y una lista mutable.
- **`LinkedList`**: Cuando planeas realizar muchas inserciones o eliminaciones en los extremos de la lista.