ARRAYLIST EN JAVA (Listas Dinámicas)

¡Claro que sí! Siguiendo fielmente la estructura, el diseño de tablas y el estilo pedagógico de tu PDF sobre "Arreglos Multidimensionales", he transformado tu código de `ArrayList` en una guía de estudio organizada.

Aquí tienes la guía adaptada:

ARRAYLIST EN JAVA (Listas Dinámicas)

1. Conceptos Fundamentales

A diferencia de los arreglos tradicionales ([ ]) que tienen un tamaño fijo una vez creados, el `ArrayList` es una estructura dinámica. Imagínalo como un acordeón: puede estirarse para alojar más elementos o encogerse si los eliminas.

Sus características esenciales son:

- **Tamaño Dinámico: No necesitas definir cuántos espacios tendrá al principio. Crece automáticamente.**
- **Solo Objetos: No almacena tipos primitivos (`int`, `boolean`, `double`) directamente. Usa sus clases "envoltorio" o *Wrappers* (`Integer`, `Boolean`, `Double`) o Strings y Objetos propios.**
- **Librería: Requiere importar `java.util.ArrayList`.**

El siguiente código ilustra la creación, modificación, eliminación y recorrido de listas dinámicas.

```
import java.util.ArrayList;

public class ArrayListEjemplo {

    public static void main(String[] args) {
        // CREACION DEL ARRAYLIST
        // solo trabajan con objetos, no con primitivos
        ArrayList<String> nombres = new ArrayList<>();
        // agregar elementos
        nombres.add("PEPE");
        nombres.add("MARIA");
        nombres.add("JUAN");
        System.out.println(nombres);
        // MODIFICAR
```

```
        nombres.set(0, "PEPE PEREZ");
        System.out.println(nombres);
        // eliminar
        nombres.remove(2);
        System.out.println(nombres);
        // leer
        System.out.println("NOMBRE: " + nombres.get(1));

        // como creo un arrylist que me deje agregar elementos de
        // diferente tipo
        ArrayList<Object> mixto = new ArrayList();
        // agregamos un entero
        mixto.add(34);
        // agregamos una cadena
        mixto.add("ADSO");
        System.out.println(mixto);
        // recorrido con for
        for (int i = 0; i < mixto.size(); i++) {
            System.out.println("@@@@@@@@@@@@");
            System.out.println(mixto.get(i));
        }
        // recorrido con for each
        for (Object elemento : mixto) {
            System.out.println("$$$$$$$$$");
            System.out.println(elemento);
        }

    }

}
```

2. Declaración e Inicialización

En los `ArrayList`, usamos los símbolos < > (diamante) para indicar qué tipo de datos guardaremos.

Concepto	Código de Ejemplo	Descripción

Declaración Típica	**ArrayList<String> lista;**	**Declara la variable. Aún no está inicializada (es null).**
Inicialización	**lista = new ArrayList<>();**	**Crea la instancia en memoria. La lista está vacía [] pero lista para usarse.**
Lista de Objetos	**ArrayList<Object> mixto;**	**Permite guardar cualquier tipo de dato (enteros, strings, booleanos) mezclados.**

3. Métodos Principales (CRUD)

A diferencia de los arreglos normales donde usamos corchetes [ ], aquí usamos métodos.

Acción	Arreglo Normal []	ArrayList (Método)	Descripción
Agregar	**No se puede (tamaño fijo)**	`lista.add("Dato");`	**Añade el elemento al final de la lista.**
Leer	`valor = arr[0];`	`valor = lista.get(0);`	**Obtiene el valor en la posición indicada.**
Modificar	`arr[0] = "Nuevo";`	`lista.set(0, "Nuevo");`	**Sobrescribe el valor en esa posición.**
Eliminar	**No se puede**	`lista.remove(0);`	**Borra el elemento y recorre los demás a la izquierda.**

Tamaño	arr.length	lista.size()	Devuelve la cantidad actual de elementos.

¡Claro que sí! Siguiendo fielmente la estructura, el diseño de tablas y el estilo pedagógico de tu PDF sobre "Arreglos Multidimensionales", he transformado tu código de ArrayList en una guía de estudio organizada.

Aquí tienes la guía adaptada:

ARRAYLIST EN JAVA (Listas Dinámicas)

1. Conceptos Fundamentales

A diferencia de los arreglos tradicionales ([]) que tienen un tamaño fijo una vez creados, el ArrayList es una estructura dinámica. Imagínalo como un acordeón: puede estirarse para alojar más elementos o encogerse si los eliminas.

Sus características esenciales son:

- Tamaño Dinámico: No necesitas definir cuántos espacios tendrá al principio. Crece automáticamente.
- Solo Objetos: No almacena tipos primitivos (int, boolean, double) directamente. Usa sus clases "envoltorio" o *Wrappers* (Integer, Boolean, Double) o Strings y Objetos propios.
- Librería: Requiere importar java.util.ArrayList.

El siguiente código ilustra la creación, modificación, eliminación y recorrido de listas dinámicas.

Java

```java
import java.util.ArrayList;

public class ArrayListEjemplo {
    public static void main(String[] args) {
        // 1. CREACIÓN DEL ARRAYLIST
        // <String> define que esta lista SOLO aceptará texto.
        ArrayList<String> nombres = new ArrayList<>();

        // 2. AGREGAR ELEMENTOS (Create)
        nombres.add("PEPE");
```

```
        nombres.add("MARIA");
        nombres.add("JUAN");

        // Impresión directa (ArrayList ya trae un toString() bonito)
        System.out.println(nombres);

        // 3. MODIFICAR (Update)
        // En el índice 0, cambiamos "PEPE" por "PEPE PEREZ"
        nombres.set(0, "PEPE PEREZ");
        System.out.println(nombres);

        // 4. ELIMINAR (Delete)
        // Borramos el elemento en el índice 2
        nombres.remove(2);
        System.out.println(nombres);

        // 5. LEER (Read)
        // Accedemos a un elemento específico
        System.out.println("NOMBRE: " + nombres.get(1));
    }
}
```

2. Declaración e Inicialización

En los ArrayList, usamos los símbolos < > (diamante) para indicar qué tipo de datos guardaremos.

Concepto	Código de Ejemplo	Descripción
Declaración Típica	ArrayList<String> lista;	Declara la variable. Aún no está inicializada (es null).
Inicialización	lista = new ArrayList<>();	Crea la instancia en memoria. La lista está vacía [] pero lista para usarse.
Lista de Objetos	ArrayList<Object> mixto;	Permite guardar cualquier tipo de dato (enteros, strings, booleanos) mezclados.

3. Métodos Principales (CRUD)

A diferencia de los arreglos normales donde usamos corchetes [], aquí usamos métodos.

Acción	Arreglo Normal []	ArrayList (Método)	Descripción
Agregar	No se puede (tamaño fijo)	lista.add("Dato");	Añade el elemento al final de la lista.
Leer	valor = arr[0];	valor = lista.get(0);	Obtiene el valor en la posición indicada.
Modificar	arr[0] = "Nuevo";	lista.set(0, "Nuevo");	Sobrescribe el valor en esa posición.
Eliminar	No se puede	lista.remove(0);	Borra el elemento y recorre los demás a la izquierda.
Tamaño	arr.length	lista.size()	Devuelve la cantidad actual de elementos.

4. Listas Mixtas (Clase Object)

Como se vio en tu código, es posible romper la regla de "un solo tipo" usando la clase padre de todo en Java: Object.

```
// Lista que acepta TODO
ArrayList<Object> mixto = new ArrayList<>();

mixto.add(34);      // Agregamos un entero (Integer)
mixto.add("ADSO");  // Agregamos una cadena (String)
mixto.add(true);    // Agregamos un booleano

System.out.println(mixto); // Salida: [34, ADSO, true]
```

Nota: Aunque es flexible, se recomienda usar tipos específicos (`<String>`, `<Integer>`) para evitar errores de conversión más adelante.

5. Recorridos (Bucles)

Para visitar cada elemento, podemos usar un `for` clásico (usando índices) o un `for-each` (más limpio).

Tipo de Recorrido	Código de Ejemplo	Descripción

For Clásico	`for (int i = 0; i < mixto.size(); i++) {`
	`System.out.println(mixto.get(i));`
	`}`
		Útil si necesitas saber el índice (posición) de cada elemento mientras recorres.

For-Each	`for (Object elemento : mixto) {`
	`System.out.println(elemento);`
	`}`
		Sintaxis más sencilla. "Para cada elemento dentro de mixto, haz esto...".

Ejemplos Adicionales

Ejemplo A: Lista de Compras (Gestión Dinámica)

Simulación de un carrito de compras donde el usuario añade y se arrepiente de productos.

```
ArrayList<String> carrito = new ArrayList<>();
carrito.add("Leche");
carrito.add("Huevos");
carrito.add("Pan");

// Se arrepiente de los Huevos (estaban en indice 1)
carrito.remove(1);

// ¿Cuántas cosas llevo?
System.out.println("Llevo " + carrito.size() + " productos: " +
carrito);
```

Ejemplo B: Números (Wrappers)

Recordemos que `ArrayList` no usa `int`, usa `Integer`.

```
ArrayList<Integer> numeros = new ArrayList<>();
numeros.add(10);
numeros.add(25);
numeros.add(100);

int suma = 0;
for(Integer n : numeros) {
    suma += n;
}
System.out.println("La suma total es: " + suma);
```

Ejercicios:

1. Practicar las operaciones básicas (CRUD).

- **Enunciado: Crea un `ArrayList` de `String` llamado `colores`.**
    1. **Agrega "Rojo", "Verde" y "Azul".**
    2. **Usa `.set()` para cambiar "Verde" por "Amarillo".**
    3. **Usa `.remove()` para eliminar el "Rojo".**
    4. **Imprime la lista final y su tamaño `.size()`.**

2. Entender el uso de Tipos Mixtos y Objetos.

- **Enunciado: Crea un perfil de usuario usando `ArrayList<Object>`.**
    1. **Agrega en este orden: Nombre (String), Edad (int), ¿Es Activo? (boolean).**
    2. **Recorre la lista con un bucle `for` clásico.**
    3. **Imprime cada dato precedido por el texto "Dato de usuario: ".**

3. Filtrado de datos numéricos.

- **Enunciado: Tienes una lista de calificaciones.**
    1. **Crea `ArrayList<Integer>` y agrega los valores: 3, 8, 1, 10, 5.**
    2. **Usa un bucle `for-each` para recorrer la lista.**
    3. **Dentro del bucle, usa un `if`: Si el número es mayor o igual a 6, imprime "Aprobado: [numero]". Si no, imprime "Reprobado: [numero]".**

Nivel Básico

Suma de Elementos: Se le proporciona un ArrayList de tipo Integer. El programa debe calcular la suma de todos los números presentes en la lista y mostrar el resultado total.

Búsqueda de un Elemento: Dado un ArrayList de String y un String específico, el programa debe determinar si el elemento buscado existe dentro de la lista. Deberá imprimir un mensaje indicando si fue encontrado o no.

Conteo de Ocurrencias: Se tiene un ArrayList de Character y un carácter objetivo. El programa debe contar cuántas veces aparece el carácter objetivo en la lista e imprimir dicho conteo.

Nivel Intermedio

Encontrar el Número Mayor: A partir de un ArrayList de Integer no vacío, el programa debe encontrar y mostrar el número más grande contenido en la lista sin utilizar métodos de ordenamiento.

Creación de una Sublista de Pares: Dado un ArrayList de Integer, el programa debe generar un nuevo ArrayList que contenga únicamente los números pares del ArrayList original. Al finalizar, debe imprimir la nueva lista de pares.

Concatenación de Cadenas: Se proporciona un ArrayList de String. El programa debe concatenar todas las cadenas de la lista en una sola, separadas por un espacio, y mostrar la cadena resultante.

Nivel Avanzado

Inversión de una Lista: Dado un ArrayList, el programa debe invertir el orden de sus elementos. Por ejemplo, si la lista original es [1, 2, 3, 4], la lista invertida debe ser [4, 3, 2, 1]. Esta operación debe realizarse modificando la lista original o creando una nueva.

Eliminación de Elementos Duplicados: Se tiene un ArrayList con elementos que pueden estar duplicados. El programa debe generar un nuevo ArrayList que contenga los elementos del original pero sin duplicados, conservando el orden de la primera aparición de cada elemento.

Reemplazo de Elementos: Dado un ArrayList de String, un elemento a buscar y un elemento de reemplazo, el programa debe sustituir todas las ocurrencias del elemento a buscar por el elemento de reemplazo.

Verificación de Palíndromo: Se le entrega un ArrayList de Character que representa una secuencia de letras. El programa debe determinar si la secuencia forma un palíndromo (se lee igual de izquierda a derecha que de derecha a izquierda), devolviendo un valor booleano o un mensaje que lo indique.