

Guía de aprendizaje

Gestión de datos con listas en Java

Exp 2 – Semana 7

Fundamentos de Programación (PRY2201)

**Facilitador disciplinar:** Josué Oteíza

**Asesor par:** Luis Videla

# Índice

[Introducción a la semana 3](#_Toc156491519)

[Resultado de aprendizaje 4](#_Toc156491520)

[Conceptos relevantes 4](#_Toc156491521)

[Preguntas activadoras 5](#_Toc156491522)

[Actividad 5](#_Toc156491523)

[Listas en Java 6](#_Toc156491524)

[Principales características 6](#_Toc156491525)

[¿Qué son las listas en el contexto de Java? 7](#_Toc156491526)

[¿Cómo se comparan las listas con arreglos simples? 9](#_Toc156491527)

[Sintaxis de una lista en Java 10](#_Toc156491528)

[Operaciones para la manipulación de Listas en Java 12](#_Toc156491529)

[Declarar listas 12](#_Toc156491530)

[Importación de Paquetes para trabajar con listas 12](#_Toc156491531)

[Indexación y acceso a elementos para trabajar con Listas 14](#_Toc156491532)

[Agregar y eliminar elementos de una lista 16](#_Toc156491533)

[Arreglos en Java 19](#_Toc156491534)

[Ejemplo de listas en Java 23](#_Toc156491535)

[Video 29](#_Toc156491536)

[Cierre de la semana 30](#_Toc156491537)

[Lecturas de la semana 31](#_Toc156491538)

[Apuntes 32](#_Toc156491539)

# Introducción a la semana

La séptima semana de nuestra asignatura Fundamentos de Programación, marca un hito importante hacia la comprensión profunda de la programación. En esta fase, veremos la gestión de datos mediante el uso de listas en Java, centrándonos en la construcción de estructuras temporales que nos permitirán abordar problemáticas complejas que requieren un manejo eficiente de la memoria.

La capacidad de manipular datos de manera efectiva es esencial en el desarrollo de software, y en esta semana, nos centraremos en el uso de arreglos para almacenar información de manera dinámica. Utilizaremos NetBeans para aplicar nuestros conocimientos y explorar conceptos clave relacionados con la creación y manipulación de listas en Java.

# Resultado de aprendizaje

### El estudiante será capaz de:

**RA3.** Construye estructuras de datos temporales, utilizando arreglos para el almacenamiento de información y el trabajo con problemáticas que requieren la mantención en datos de memoria.

### Indicador de logro:

**IL7.** Define arreglos en Java a través de listas, brindando la capacidad de acceder, agregar y eliminar elementos con la sintaxis e indexación adecuadas.

# Conceptos relevantes

|  |  |  |
| --- | --- | --- |
| Listas en Java | Arreglos en Java | Declaración y creación de Listas y Arreglos |
| Iteración a través de Listas y Arreglos | Gestión de Memoria |  |

# Preguntas activadoras

* ¿Cuál es la diferencia fundamental entre una lista y un arreglo en Java?
* ¿Cómo se declaran y crean listas en Java?
* Si tienes una lista de elementos, ¿cómo eliminarías un elemento específico?

# Actividad

**Descripción de la actividad**

En esta séptima semana, realizarás una actividad formativa individual con encargo, llamada "Gestionando datos con listas en Java", donde definirás arreglos en Java usando listas para almacenar y acceder a datos primitivos, incluyendo acceder, agregar y eliminar elementos de manera adecuada.

# Listas en Java

## Principales características

La semana anterior, trabajamos en el entorno de desarrollo NetBeans en la escritura y depuración (*Debug*). Ahora, comenzaremos a trabajar con listas y arreglos en Java, lo que te permitirá construir estructuras de datos temporales eficientes y abordar problemas prácticos en el desarrollo de software. Entonces, ¿a qué nos referimos con listas en Java?

Las listas en Java representan una de las estructuras de datos más fundamentales y versátiles en la programación. Su importancia radica en su capacidad para almacenar secuencias de elementos de manera dinámica, a diferencia de los arreglos estáticos cuyo tamaño no puede modificarse una vez definido. Este dinamismo hace que las listas sean esenciales para situaciones donde el volumen de datos no se conoce de antemano o puede cambiar a lo largo del tiempo.

Java proporciona una rica biblioteca de clases e interfaces para trabajar con listas a través de su paquete `java.util`. Entre las implementaciones más comunes se encuentran `ArrayList`, `LinkedList`, y `Vector`, cada una con sus propias características y casos de uso específicos.

La manipulación de arreglos se facilita enormemente con las listas en Java por varias razones:

1. **Flexibilidad en tamaño:** las listas automáticamente ajustan su tamaño al agregar o eliminar elementos, eliminando la necesidad de crear nuevos arreglos y copiar elementos de uno viejo a uno nuevo.
2. **Métodos útiles:** las listas vienen con métodos incorporados para realizar operaciones comunes como agregar, eliminar, y acceder a elementos, así como útiles herramientas para ordenar y buscar.
3. **Mayor seguridad de tipo:** las listas genéricas en Java permiten definir un tipo específico para los elementos que contienen, reduciendo la posibilidad de errores en tiempo de ejecución y aumentando la legibilidad del código.

## ¿Qué son las listas en el contexto de Java?

En el contexto de Java, las listas son estructuras de datos dinámicas que forman parte del marco de colecciones Java (`java.util`), diseñadas para almacenar elementos en un orden secuencial. A diferencia de los arreglos estáticos, las listas pueden ajustar su tamaño automáticamente para acomodar la adición y eliminación de elementos, lo que las hace extremadamente flexibles para manejar conjuntos de datos que cambian dinámicamente.

Las listas en Java son implementadas principalmente a través de la interfaz `List`, que es una subinterfaz de `Collection`. Define un contrato que todas las clases concretas que implementan listas deben seguir, proporcionando métodos para operaciones básicas como agregar, eliminar, y acceder a elementos, así como obtener el tamaño de la lista, verificar si está vacía, y más.

Hay varias implementaciones de la interfaz `List` en Java, cada una con sus propias características y usos óptimos:

1. **ArrayList:** Es una de las implementaciones más populares, puesto que utiliza un arreglo dinámico para almacenar los elementos, además de ofrecer acceso rápido a los elementos a través de índices. Puede ser lento para operaciones que requieren modificar la lista, como insertar o eliminar elementos en medio de la lista.
2. **LinkedList:** Implementa tanto la interfaz ‘List’ como la interfaz `Deque`, y almacena sus elementos en nodos doblemente enlazados, lo que facilita la inserción y eliminación de elementos, especialmente en medio de la lista, pero el acceso a elementos específicos puede ser más lento que en un `ArrayList`.
3. **Vector:** Similar a `ArrayList`, pero con métodos sincronizados para operaciones seguras en entornos de múltiples hilos. Sin embargo, debido a esta sincronización, puede ser menos eficiente en términos de rendimiento en comparación con `ArrayList`.
4. **Stack:** Es una subclase de `Vector` que implementa una estructura de pila de último en entrar, primero en salir (LIFO).

Importante

El uso de genéricos en las listas permite que sean más seguras en cuanto a tipos, permitiendo especificar el tipo de objetos que la lista puede contener, lo que ayuda a prevenir errores en tiempo de ejecución y mejora la legibilidad del código.

## ¿Cómo se comparan las listas con arreglos simples?

Las listas y los arreglos simples son dos de las estructuras de datos más utilizadas en Java, y aunque pueden parecer similares porque ambos almacenan colecciones de elementos, tienen diferencias significativas en su funcionamiento, uso y eficiencia. Hagamos la comparación entre la naturaleza y uso de cada una y el tipo de datos que manejan:

* **Arreglos simples:** son colecciones de tamaño fijo que pueden almacenar elementos de un solo tipo. Cuando se declara un arreglo con un tamaño específico, no puede cambiar, lo que significa que debes saber la cantidad de elementos que almacenará de antemano.

**Listas:** son colecciones dinámicas que forman parte del marco de colecciones de Java. Pueden ajustar su tamaño automáticamente cuando se agregan o eliminan elementos. Las listas ofrecen una mayor flexibilidad y son más adecuadas para situaciones donde el número de elementos puede variar durante el tiempo de ejecución.

**Arreglos simples:** Pueden almacenar tanto tipos primitivos (como `int`, `char`, etc.) como objetos. Sin embargo, todos los elementos deben ser del mismo tipo.

**Listas:** Solo pueden almacenar objetos y no tipos primitivos. Sin embargo, con la introducción de clases envolventes (Wrapper classes) y autoboxing en versiones más recientes de Java, trabajar con tipos primitivos es más fácil y transparente.

## Sintaxis de una lista en Java

La sintaxis en Java para trabajar con listas es sencilla pero poderosa, permitiéndote manipular colecciones de objetos de forma dinámica y eficiente. A continuación, se explica la sintaxis básica para trabajar con listas en Java, incluyendo cómo declararlas, los paquetes necesarios para trabajar con ellas, y ejemplos de código para su creación e inicialización.

**Figura 1**

*Sintaxis básica de Java*

*Nota*. Muestra de sintaxis básica de Java. The Apache Software Foundation. (2023). Apache NetBeans (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

1. Todos los archivos pertenecen a un paquete.
2. Importa los paquetes para el proyecto.
3. Java usa clases para ejecutar el código.
4. Se debe indicar el tipo de dato.
5. Modificadores de acceso: private, public, protected o por defecto ninguno.
6. El método principal en Java es el método ‘main’.
7. La palabra reservada ‘new’ crea un objeto del tipo de dato especificado.
8. Se utilizan ‘;’ para cada sentencia.
9. Se usan llaves { } para identificar el bloque de código.

Ahora que ya conocemos lo básico, te detallaremos a continuación las acciones que puedes realizar para manejar listas en Java.

# Operaciones para la manipulación de Listas en Java

## Declarar listas

En Java, una lista se declara especificando el tipo de objetos que almacenará, seguido por el nombre de la variable. La declaración típica de una lista utiliza la interfaz ‘List’ del paquete ‘java.util’. Aquí hay un ejemplo básico:

List<String> myList;

En este caso, ‘List’ es la interfaz y ‘String’ es el tipo de los elementos que la lista almacenará. ‘myList’ es el nombre de la variable de lista. Puedes reemplazar ‘String’ con cualquier otro tipo de objeto.

## Importación de Paquetes para trabajar con listas

Para utilizar la interfaz ‘List’ y las clases relacionadas, debes importar el paquete adecuado de la biblioteca estándar de Java, que es ‘java.util’. Esto se hace con la declaración ‘import’ en la parte superior del archivo fuente de Java antes de la declaración de la clase. Por ejemplo, estas declaraciones:

* ‘java.util.List’: La interfaz ‘List’ que proporciona el contrato para las listas en Java.

import java.util.List;

import java.util.ArrayList;

import java.util.LinkedList;

* ‘java.util.ArrayList’: Una de las implementaciones más utilizadas de la interfaz ‘List’.
* ‘java.util.LinkedList’: Otra implementación común de ‘List’ que usa una estructura de datos de lista enlazada.

Aquí hay un ejemplo de cómo importar estas clases:

**Figura 2**

*Cómo importar clases*

*Nota*. Ejemplo de cómo importar clases. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, se crea una ‘ArrayList’ llamada ‘frutas’ que almacena objetos ‘String’. Luego, se añaden tres frutas a la lista y se imprime la lista completa en la consola.

Un ejemplo similar sería el siguiente:

**Figura 3**

*Importación de la interfaz ‘List’ y la clase ‘LinkedList’*

*Nota*. Ejemplo de importación de una ‘LinkedList’. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Este ejemplo es similar al anterior, pero utiliza ‘LinkedList’ para almacenar números enteros. Las operaciones básicas son las mismas: crear la lista, añadir elementos y mostrar la lista.

## Indexación y acceso a elementos para trabajar con Listas

La indexación y el acceso a elementos son aspectos fundamentales al trabajar con listas en Java, permitiendo interactuar y manipular los datos que estas contienen.

En Java, puedes acceder a elementos específicos de una lista utilizando índices. Un índice es un número entero que representa la posición de un elemento en la lista, comenzando desde 0. Por ejemplo, el primer elemento está en el índice 0, el segundo en el índice 1, y así sucesivamente.

Para acceder a un elemento, utilizas el método ‘get(index)’ de la lista, donde ‘index’ es la posición del elemento que deseas recuperar. Aquí hay un ejemplo de lo mencionado:

**Figura 4**

*Uso del método ‘get(index)’*

*Nota*. Ejemplo de uso del método ‘get(index)’. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

El código Java mostrado inicializa una ‘ArrayList’ de ‘String’ denominada nombres y agrega tres ‘nombres’ a la lista. Luego accede al segundo elemento de esta lista, que está en el índice 1, asignándolo a una variable llamada ‘nombre’. Finalmente, imprime ese nombre, que en este caso es "Luis", a la consola.

## Agregar y eliminar elementos de una lista

Para agregar elementos, utilizas el método ‘add()’. Puedes agregar un elemento al final de la lista simplemente pasando el elemento al método’ add()’:

**Figura 5**

*Método ‘add()’*

*Nota*. Ejemplo de uso del método ‘add()’. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

También puedes insertar un elemento en una posición específica de la lista indicando el índice y el elemento a insertar:

**Figura 6**

*Método ‘add(índice, elemento)’*

*Nota*. Ejemplo de uso del método *‘add(índice, elemento)’*. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Con el método ‘addAll(colección)’ puedes agregar todos los elementos de la colección especificada al final de la lista. Esto es útil cuando deseas combinar dos listas o agregar múltiples elementos de una vez.

**Figura 7**

*Método ‘addAll(colección)’*

*Nota*. Ejemplo de uso del método *‘addAll(colección)’*. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Por último, el método ‘addAll(indice, coleccion)’ Inserta todos los elementos de la colección especificada a partir del índice indicado, desplazando los elementos existentes hacia la derecha.

**Figura 8**

*Método ‘addAll(indice, coleccion)’*

*Nota*. Ejemplo de uso del método *‘addAll(indice, coleccion)’.* The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Si aplicáramos todos los métodos mencionados, nuestro código en Java quedaría de la siguiente manera:

**Figura 9**

*Utilización de métodos con una lista ‘String’*

*Nota*. Ejemplo de utilización de métodos en una lista. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

* Se agrega "Manzana" al final de la lista.
* Se inserta "Banana" al principio de la lista (índice 0).
* Se crea una nueva lista ‘masFrutas’ con "Naranja" y "Pera".
* Se agregan todos los elementos de ‘masFrutas’ al final de frutas.
* Se insertan todos los elementos de’ masFrutas’ en frutas empezando en el índice 1.

La salida final de este código cuando se imprime la lista ‘frutas’ será:

[Banana, Naranja, Pera, Manzana, Naranja, Pera]

Bien, ahora que conoces sobre las listas, es tiempo de aprender sobre los arreglos en Java

# Arreglos en Java

En Java, un arreglo es una estructura de datos que permite almacenar una colección de elementos del mismo tipo. Los arreglos se almacenan en ubicaciones de memoria contiguas, lo que facilita el acceso a sus elementos. Los arreglos en Java pueden ser de cualquier tipo primitivo, como enteros, números reales, caracteres o booleanos. También pueden ser de tipo objeto, como cadenas, objetos personalizados o clases.

También, pueden ser unidimensionales o multidimensionales. Los arreglos unidimensionales son los más comunes y se utilizan para almacenar una colección de elementos que se pueden organizar en una sola dimensión, como una lista de números o una lista de nombres. Los arreglos multidimensionales se utilizan para almacenar una colección de elementos que se pueden organizar en varias dimensiones, como una matriz de números o una tabla de nombres y edades.

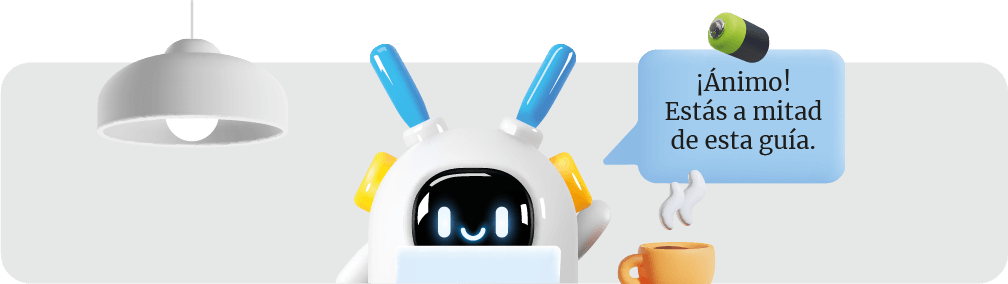
Los arreglos en Java se declaran utilizando el operador ‘[]‘. Por ejemplo, el siguiente código declara un arreglo de 10 enteros:

int[] arreglo = new int[10];

Una vez declarado, un arreglo se puede inicializar asignando valores a sus elementos. La inicialización puede realizarse en el momento de la declaración o posteriormente. Por ejemplo, el siguiente código inicializa el arreglo declarado anteriormente con los valores 1, 2, 3, ..., 10:

**Figura 10**  
*Inicializar y llenar un arreglo de enteros en Java*

*Nota*. Ejemplo de cómo inicializar y llenar un arreglo de enteros en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>



Los elementos de un arreglo se pueden acceder utilizando su índice. El índice de un elemento es un número entero que indica la posición del elemento en el arreglo. Por ejemplo, el siguiente código imprime el primer elemento del arreglo declarado anteriormente:

**Figura 11**

*Cómo imprimir el primer elemento del arreglo declarado*

*Nota*. Ejemplo de cómo imprimir el primer elemento del arreglo declarado. The Apache Software Foundation. (2023). *Apache NetBeans* (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Así como las listas, los arreglos en Java se pueden manipular utilizando una serie de métodos, como:

* ‘add()’: añade un elemento al arreglo.
* ‘remove()’: elimina un elemento del arreglo.
* ‘get()’: obtiene un elemento del arreglo.
* ‘length()’: devuelve el número de elementos del arreglo.

El siguiente ejemplo muestra cómo crear un arreglo de enteros, añadir elementos al arreglo e imprimir los elementos del arreglo:

**Figura 12**

*Cómo crear un arreglo de enteros*

*Nota*. Ejemplo de cómo crear un arreglo de enteros, añadir elementos al arreglo e imprimir los elementos del arreglo. The Apache Software Foundation. (2023*). Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Este código realiza lo siguiente:

* Declara e inicializa un ‘arreglo’ de enteros arreglo con espacio para 10 elementos.
* Llena el arreglo con números del 1 al 10 utilizando un bucle ‘for’ que recorre el arreglo por índices.
* Utiliza un bucle ‘for-each’ para recorrer el arreglo e imprimir cada uno de sus elementos.

El código recién visto producirá la siguiente salida:

**Figura 13**

*Salidas*

*Nota*. Salidas. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

# Ejemplo de listas en Java

Para finalizar, a continuación, te mostraremos un caso ficticio aplicado a un sistema de reservaciones en un hotel, donde se utilizan listas en Java para gestionar las operaciones:

El sistema de reservaciones de un hotel debe permitir realizar las siguientes operaciones:

1. **Realizar reservación:** el usuario debe ingresar los siguientes datos:

* Nombre del huésped
* Tipo de habitación (sencilla, doble, suite)
* Cantidad de noches
* Fechas de estancia

1. **Cancelar reservación:** el usuario debe ingresar el número de la reservación que desea cancelar.
2. **Buscar reservación:** el usuario debe ingresar el número de la reservación que desea buscar.

Al realizar la reservación, el sistema debe aplicar los siguientes descuentos:

* 5% para reservaciones de más de 5 noches
* 10% para reservaciones en temporada baja

Al finalizar la reservación, el sistema debe imprimir la confirmación indicando el siguiente mensaje:

*"Su reservación ha sido confirmada. ¡Esperamos que disfrute su estancia!".*

1. **Desarrollo**

El sistema se desarrolla utilizando listas en Java. Las listas se utilizan para almacenar las reservaciones realizadas.

La clase ‘Reservacion’ representa una reservación realizada. La clase tiene los siguientes atributos:

* ‘nombreHuesped’: el nombre del huésped
* ‘tipoHabitacion’: el tipo de habitación (sencilla, doble, suite)
* ‘cantidadNoches’: la cantidad de noches
* ‘fechasEstancia’: las fechas de la estancia
* ‘importe’: el importe de la reservación

La clase ‘Reservacion’ tiene los siguientes métodos:

* ‘constructor()’: el constructor de la clase
* ‘aplicarDescuento()’: aplica el descuento correspondiente basado en la duración de la estancia y la temporada
* ‘getImporte()’: devuelve el importe de la reservación

La clase ‘SistemaReservaciones’ representa el sistema de reservaciones del hotel. La clase tiene los siguientes atributos:

* ‘listaReservaciones’: la lista de reservaciones realizadas

La clase ‘SistemaReservaciones’ tiene los siguientes métodos:

* ‘constructor()’: el constructor de la clase
* ‘realizarReservacion()’: realiza una reservación
* ‘cancelarReservacion()’: cancela una reservación
* ‘buscarReservacion()’: busca una reservación

El menú del sistema tiene las siguientes opciones:

* Realizar reservación
* Cancelar reservación
* Buscar reservación
* Salir

Si reflejáramos este ejemplo completo con todas las funcionalidades detalladas sería bastante extenso, pero te mostraremos una versión simplificada que incluye las estructuras básicas de las clases y métodos mencionados para el sistema de reservaciones de un hotel en Java:

**Figura 14**

*Desarrollo del sistema de reservas de un hotel en Java*

*Nota*. Ejemplo de desarrollo de un sistema de reservas de un hotel en Java. The Apache Software Foundation. (2023*). Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

**Figura 15**

*Desarrollo del sistema de reservas de un hotel en Java*

*Nota*. Ejemplo de desarrollo de un sistema de reservas de un hotel en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

**Figura 16**

*Desarrollo del sistema de reservas de un hotel en Java*

*Nota*. Ejemplo de desarrollo de un sistema de reservas de un hotel en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

**Figura 17**

*Desarrollo del sistema de reservas de un hotel en Java*

*Nota*. Ejemplo de desarrollo de un sistema de reservas de un hotel en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

# Video

Por último, no olvides revisar los siguientes enlaces, donde podrás revisar el uso básico de listas en Java:

* <https://www.youtube.com/watch?v=5k6xDsSyVA8>
* <https://www.youtube.com/watch?v=yHFacwFar0A>

# Cierre de la semana

Las listas y los arreglos en Java son estructuras de datos fundamentales que almacenan colecciones de elementos. Los arreglos son estructuras de tamaño fijo, definidos con un tipo de dato específico y una longitud establecida en su creación, y son útiles cuando la cantidad de elementos a almacenar es conocida y no cambia durante la ejecución del programa. En cambio, las listas, como ‘ArrayList’ o ‘LinkedList’, son parte del Java Collections Framework y ofrecen una capacidad dinámica para expandirse y contraerse, permitiendo añadir, insertar, eliminar y acceder a elementos de manera flexible, lo cual los hace ideales para cuando el número de elementos puede variar a lo largo del tiempo. Mientras que los arreglos ofrecen un acceso rápido y eficiente a los elementos a través de índices, las listas aportan métodos más avanzados para el manejo de datos y operaciones como búsquedas, ordenación y filtrado.

# Lecturas de la semana

**Chapter 11:** Using Arrays to juggle values

Burd, B. (2022). Java for Dummies. New Jersey: John Wiley & Sons. <https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50163-Java_fordummies/302/> Páginas 295 a 321.

# Apuntes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.