**UNIVERSIDAD DE EL SALVADOR**
Facultad Multidisciplinaria De Occidente
Departamento de Ingeniería y Arquitectura
Ingeniería en Desarrollo de Software

**Asignatura:**

**Manejo de Estructura de Datos**

Ciclo III / Segundo año

**Tema:**

# Proyecto Final
# Manejo de Estructuras de Datos en Java

**Coordinador de Cátedra:**

Ing. Jonathan Wilfredo Rodríguez Ramos

**Tutor GT02:**

Ing. Jonathan Wilfredo Rodríguez Ramos

**Integrantes del Grupo 06:**

MO21016 Jason Alexander Molina Ortiz
RM24001 Cindy Ariana Reyes Molina RM24001

**Lugar y fecha de entrega:**

Ciudad Universitaria, domingo 22 de junio de 2025

# Introducción

Las listas enlazadas constituyen una de las estructuras de datos más esenciales en el ámbito de la programación, especialmente cuando se requiere gestionar colecciones de elementos cuyo tamaño puede variar en tiempo de ejecución. A diferencia de los arreglos convencionales, estas listas no necesitan una longitud fija ni almacenamiento contiguo en memoria, lo cual las hace especialmente flexibles y adaptables a diferentes escenarios.

Están compuestas por nodos interconectados mediante punteros o referencias, lo que permite insertar o eliminar elementos de manera eficiente sin tener que desplazar el contenido existente. Esta naturaleza dinámica permite optimizar tanto el uso de la memoria como el rendimiento en ciertas operaciones.

Existen diversas variantes de listas enlazadas, como las listas simplemente enlazadas, doblemente enlazadas, circulares y aquellas que incluyen un nodo de encabezado. Cada tipo presenta particularidades que lo hacen ideal para distintos contextos, dependiendo de la complejidad de las operaciones que se requiera realizar.

Además, se destacan por su utilidad en la implementación de estructuras más complejas como pilas, colas y tablas de dispersión. Este documento explorará en profundidad el concepto de listas enlazadas, su clasificación, las ventajas e inconvenientes que presentan, así como sus aplicaciones prácticas, ilustradas mediante ejemplos desarrollados en Java.

# Listas enlazadas

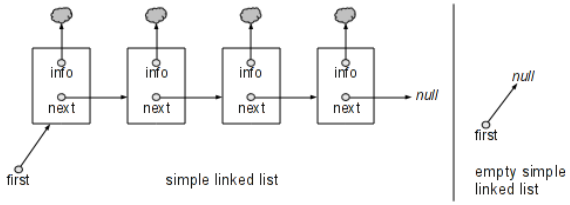
Una lista ligada o una lista enlazada es un tipo de estructura de datos que crece de manera dinámica. Está compuesta por nodos, cada uno con información y una referencia que apunta al siguiente en la secuencia. A diferencia de los arreglos tradicionales, no necesita un tamaño predeterminado, lo que facilita agregar o quitar elementos sin reorganizar toda la colección, ya que solo se modifican enlaces.

Cada nodo, al contener una referencia hacia otro del mismo tipo, convierte a las listas en estructuras autorreferenciadas. Esto aporta flexibilidad, especialmente útil cuando se deben realizar cambios constantes en distintas partes de la lista. Aunque no permiten acceder directamente a cualquier posición como lo hacen los arreglos, su forma de recorrer puede diferir del orden en que están almacenados físicamente, lo cual representa una ventaja en ciertos escenarios.

Hay varias clases de listas enlazadas. Las simples conectan un nodo con su siguiente, ideales para recorridos en una sola dirección. Las dobles enlazadas tienen referencias hacia el nodo anterior y el siguiente, lo que permite desplazarse en ambos sentidos. También existen las circulares simples, donde el último nodo apunta al primero, haciendo el recorrido cíclico. Por último, las circulares doblemente enlazadas integran enlaces en ambos sentidos y conexión entre el final y el inicio, lo cual permite navegar sin límites en cualquier dirección.

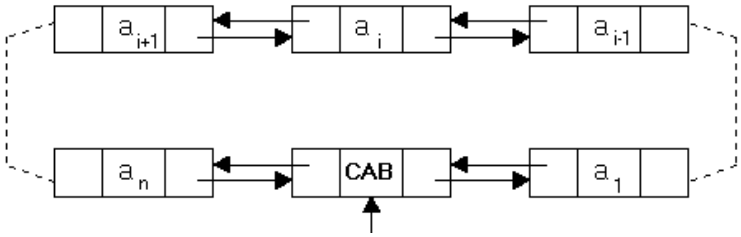
## Clasificación de listas enlazadas

- Listas enlazadas simples: es un tipo de estructura de datos que permite organizar información de manera lineal y dinámica. Está formada por una serie de nodos, donde cada nodo almacena un valor y una referencia que apunta al siguiente nodo en la secuencia. Gracias a este mecanismo de enlace, solo es necesario conocer el primer nodo para recorrer todos los elementos uno por uno, siguiendo los apuntadores hasta llegar al final de la lista.



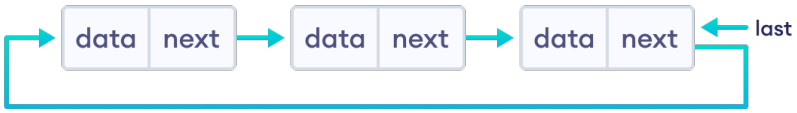

Ejemplo de aplicación:
La implementación de una cola de impresión en la que varias personas envían documentos a una impresora. Cada documento se coloca al final de la lista (como si hiciera fila), y la impresora va procesando uno por uno, comenzando desde el primero que llegó. La lista simplemente enlazada es perfecta aquí porque solo necesitas avanzar en una dirección: del primer documento al último. Además, agregar nuevos trabajos es tan sencillo como enlazarlos al final.

- Listas doblemente enlazadas: es una estructura dinámica que se forma mediante nodos conectados de forma secuencial. Cada nodo contiene un dato y dos enlaces: uno apunta al nodo siguiente y el otro al anterior. Esta estructura permite desplazarse por la lista en ambas direcciones desde cualquier nodo, sin necesidad de iniciar desde los extremos.
  El primer nodo usualmente enlaza hacia null o un nodo especial que indica el inicio, y lo mismo ocurre en sentido contrario con el último nodo. Aunque insertar o eliminar nodos implica modificar más punteros que en una lista simplemente enlazada, el procedimiento es más sencillo porque no se requiere guardar el nodo anterior: el propio nodo mantiene referencias tanto al anterior como al siguiente. Esto hace que sea posible modificar la lista en cualquier posición con eficiencia, aunque a costa de un consumo de memoria un poco mayor.




  Ejemplo de aplicación:
  La implementación de un editor de texto con función de deshacer y rehacer. Cada cambio que se hace (como escribir o borrar una letra) se guarda como un nodo en la lista. Al presionar “deshacer”, el programa se mueve al nodo anterior para restaurar el estado previo del texto. Si luego presionas “rehacer”, avanza al siguiente nodo.

  -Lista circular simplemente enlazada: es una estructura de datos lineal en la que los nodos están conectados de tal forma que el último elemento no apunta a null, sino que enlaza de vuelta al primero, formando un ciclo cerrado. Esta configuración elimina el concepto de un comienzo o un final absolutos, ya que se puede recorrer la lista indefinidamente en una sola dirección.
  Cada nodo almacena un dato y una referencia al siguiente nodo, manteniendo la simplicidad de una lista enlazada común, pero con la particularidad de que al llegar al final, el recorrido continúa automáticamente desde el principio. Esta estructura es útil en escenarios donde se requiere un comportamiento repetitivo o continuo, como en sistemas de asignación de turnos, juegos en ronda o estructuras de tipo buffer circular.




  Ejemplo de aplicación: el diseño de un sistema de turnos en juegos multijugador en el que un juego de mesa digital en el que participan varios jugadores por turnos. Cada jugador puede representarse como un nodo en la lista, y al terminar su turno,

el control pasa automáticamente al siguiente nodo (jugador). Como la lista es circular, cuando el último jugador termina, el turno vuelve al primero sin necesidad de reiniciar el recorrido manualmente. Esto permite mantener una secuencia infinita de turnos de forma natural y ordenada.

- Lista circular doblemente enlazada: es un tipo de estructura de datos en la que cada nodo está conectado tanto con el nodo siguiente como con el anterior, formando un ciclo continuo. En este diseño, el último nodo apunta al primero y, a su vez, el primero apunta al último, permitiendo recorrer la lista en cualquier dirección desde cualquier punto.
  Cada nodo contiene un valor y dos enlaces: uno hacia adelante y otro hacia atrás. Gracias a esta conexión circular, es posible moverse indefinidamente por la lista, lo que resulta útil en aplicaciones donde se requiere navegación fluida y repetitiva. A diferencia de las listas enlazadas tradicionales donde el recorrido termina en un nodo nulo, aquí se retorna eventualmente al mismo nodo desde el que se empezó, ya sea avanzando o retrocediendo.


Ejemplo de aplicación: los reproductores de música, especialmente cuando tienen activada la función de “repetir lista”.
En esta estructura, cada canción se representa como un nodo. Gracias a los enlaces hacia adelante y hacia atrás, puedes pasar fácilmente a la siguiente canción o volver a la anterior. Y como es circular, cuando terminas la última canción, automáticamente vuelve a sonar la primera sin necesidad de reiniciar manualmente. Lo mismo ocurre al retroceder desde la primera canción: se va directamente a la última.

-Lista enlazada de encabezado: es una estructura de datos que incluye un nodo adicional al principio, denominado nodo de encabezado. Este nodo no contiene datos relevantes del usuario, pero actúa como punto de referencia inicial. Su principal propósito es facilitar operaciones como la inserción y eliminación de nodos, especialmente al comienzo de la lista, evitando así tener que manejar casos especiales cuando la lista está vacía o se modifica el primer elemento.
- Ejemplo de aplicación: una aplicación que administre tus tareas diarias. Se puede usar una lista enlazada con nodo de encabezado para almacenar cada tarea como un nodo. El nodo de encabezado no contendría ninguna tarea real, pero serviría como punto de partida para recorrer, insertar o eliminar tareas sin preocuparte por si la lista está vacía o si modificas la primera tarea.

# Ejemplos de aplicación en JAVA

## Lista simplemente enlazada

```
class Nodo {
    int dato;
    Nodo sig;
    Nodo(int d) { dato = d; }
}

public class Lista {
    public static void main(String[] args) {
        Nodo n = new Nodo(1);
        n.sig = new Nodo(2);
        n.sig.sig = new Nodo(3);
        for (Nodo I = n; I != null; I = i.sig)
            System.out.print(i.dato + " -> ");
        System.out.println("null");
    }
}
```

## Lista doblemente enlazada

```
class Nodo {
    int dato;
    Nodo ant, sig;
    Nodo(int d) { dato = d; }
}
public class ListaDoble {
        public static void main(String[] args) {
        Nodo n1 = new Nodo(10);
        Nodo n2 = new Nodo(20);
        Nodo n3 = new Nodo(30);

        n1.sig = n2; n2.ant = n1;
        n2.sig = n3; n3.ant = n2;

        for (Nodo n = n1; n != null; n = n.sig)
            System.out.print(n.dato + " <-> ");
        System.out.println("null");
    }
}
```

## Lista circular simplemente enlazada

```
class Nodo {
    int dato; Nodo sig;
    Nodo(int d) { dato = d; }
}

public class Lista {
    public static void main(String[] args) {
        Nodo a = new Nodo(1), b = new Nodo(2), c =
new Nodo(3);
        a.sig = b; b.sig = c; c.sig = a;
        for (Nodo n = a, I = a; ; ) {
            System.out.print(n.dato + " -> ");
            n = n.sig;
            if (n == i) break;
        }
        System.out.println("(vuelve)");
    }
}
```

## Lista circular doblemente enlazada

```
class Nodo {
    int dato;
    Nodo ant, sig;
    Nodo(int d) { dato = d; }
}
public class ListaCircularDoble {
    public static void main(String[] args) {
        Nodo a = new Nodo(1), b = new Nodo(2), c =
new Nodo(3);
        a.sig = b; b.ant = a;
        b.sig = c; c.ant = b;
        c.sig = a; a.ant = c; // enlaces circulares
        Nodo n = a;
        do {
            System.out.print(n.dato + " <-> ");
            n = n.sig;
        } while (n != a);
        System.out.println("(vuelve)");
    }
}
```

## Ventajas

- Tamaño dinámico : las listas enlazadas pueden crecer o reducirse en tamaño durante el tiempo de ejecución sin necesidad de reasignar memoria para toda la estructura.
- Inserciones y eliminaciones eficientes : agregar o quitar elementos al principio o al final de la lista es muy rápido y, por lo general, O(1)complejo en tiempo.
- Sin desperdicio de memoria : las listas enlazadas utilizan solo la memoria necesaria para los datos reales, a diferencia de las matrices, que pueden tener espacio preasignado sin usar.
- Flexibilidad en la asignación de memoria : los nodos se pueden almacenar en cualquier lugar de la memoria, sin necesidad de bloques contiguos de memoria como las matrices.
- Fácil implementación de tipos de datos abstractos : las listas enlazadas se utilizan a menudo para implementar pilas, colas y otros tipos de datos abstractos.
- Flexibilidad en el reordenamiento : reordenar elementos en una lista vinculada es eficiente y no requiere mover grandes bloques de memoria.

## Desventajas

- El acceso aleatorio es ineficiente : a diferencia de los arrays, las listas enlazadas no permiten el acceso directo a los elementos por índice. Para llegar al elemento n, se debe recorrer desde el nodo principal, lo cual requiere O(n)tiempo.
- Uso adicional de memoria : Cada nodo de una lista enlazada requiere memoria adicional para almacenar los punteros a otros nodos. Esta sobrecarga puede ser significativa en listas grandes.
- El recorrido inverso es difícil : en listas simplemente enlazadas, es complicado recorrerlas hacia atrás. Las listas doblemente enlazadas solucionan este problema, pero requieren aún más memoria.
- Rendimiento de la caché : Las listas enlazadas tienen una ubicación de caché deficiente porque los elementos no se almacenan en ubicaciones de memoria contiguas. Esto puede provocar más fallos de caché de la CPU y un rendimiento más lento.
- Vulnerable a fugas de memoria : si no se gestionan adecuadamente, las listas enlazadas pueden provocar fugas de memoria, especialmente al eliminar nodos o destruir la lista.

## Conclusión

En definitiva, las listas enlazadas constituyen una estructura de datos sumamente útil y adaptable en el desarrollo de software, especialmente cuando se requiere gestionar colecciones de datos que cambian con frecuencia. Gracias a su diseño autorreferencial, permiten la expansión o reducción dinámica sin malgastar recursos de memoria, lo que las convierte en una alternativa eficiente frente a estructuras estáticas como los arreglos.

No obstante, su implementación también implica ciertos retos, como el acceso secuencial a los elementos y el uso adicional de memoria por los punteros. La elección del tipo de lista enlazada más adecuada dependerá de las necesidades particulares del problema: ya sea una lista simple para recorridos lineales, una doble para facilitar la navegación bidireccional, una circular para flujos repetitivos, o una con nodo de encabezado para optimizar tareas como inserción y eliminación.

Cuando se diseña e implementa correctamente, una lista enlazada puede ser una solución sofisticada y eficiente a diversas situaciones en el ámbito de la programación.

# Bibliografía


Youcademy. (s.f.). *Advantages & disadvantages of linked lists*. Youcademy. Recuperado el 21 de junio de 2025, de https://youcademy.org/advantages-disadvantages-of-linked-lists/

WsCube Tech. (s.f.). *Circular doubly linked list*. WsCube Tech. Recuperado el 21 de junio de 2025, de https://www.wscubetech.com/resources/dsa/circular-doubly-linked-list

GeeksforGeeks. (s.f.). *Advantages and disadvantages of linked list*. GeeksforGeeks. Recuperado el 21 de junio de 2025, de https://www.geeksforgeeks.org/advantages-and-disadvantages-of-linked-list/

GeeksforGeeks. (s.f.). *Types of linked list*. GeeksforGeeks. Recuperado el 21 de junio de 2025, de https://www.geeksforgeeks.org/types-of-linked-list/

Programiz. (s.f.). *Circular Linked List*. Programiz. Recuperado el 21 de junio de 2025, de https://www.programiz.com/dsa/circular-linked-list

Creately. (s.f.). *Lista doblemente enlazada* [Diagrama]. Creately. Recuperado el 21 de junio de 2025, de https://creately.com/diagram/example/VVDaWn7kLd7/lista-doblemente-enlazada

Universidad de Granada. (s.f.). *Listas doblemente enlazadas*. Recuperado el 21 de junio de 2025, de https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/ldoble.html

Study.com. (s.f.). *Singly linked lists in Java: creation & application*. Study.com. Recuperado el 21 de junio de 2025, de https://study.com/academy/lesson/singly-linked-lists-in-java-creation-application.html

W3Schools. (s.f.). *Types of linked lists*. W3Schools. Recuperado el 21 de junio de 2025, de https://www.w3schools.com/dsa/dsa_data_linkedlists_types.php