LISTAS Y LISTAS SIMPLES

ESTRUCTURAS DE DATOS

Cristian Camilo González Carmona
Francisco Andrés Taborda Amaya
David Rueda Zuleta

ITM

DEFINICIONES

¿Qué es una lista?

- Una lista es una estructura de datos secuencial que permite almacenar elementos manera ordenada.
- Las listas pueden contener datos duplicados y el orden de inserción se mantiene.

DEFINICIONES

ArrayList:

- Implementa la interfaz List.
- Ofrece acceso rápido mediante índices.
- Crece dinámicamente (no es necesario declarar su tamaño inicial).

DEFINICIONES

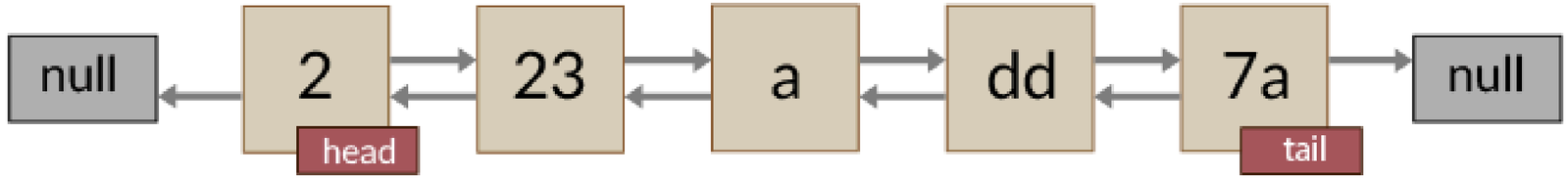
LinkedList:

- Implementa tanto List como Deque.
- Se basa en nodos enlazados que facilitan inserciones y eliminaciones eficientes en cualquier parte de la lista, aunque el acceso aleatorio puede ser más lento.

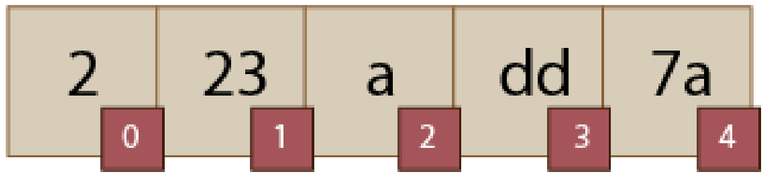
DEFINICIONES

ArrayList vs. LinkedList

LinkedList



Array and ArrayList



ARRAY VS LINKED

	ArrayList	**LinkedList**
Acceso por índice	Rápido	Lento
Inserción al comienzo	Lento	Rápido
Eliminación en medio	Lento	Eficiente
Estructura interna	Array	Nodos enlazados
Objetivo	Lecturas frecuentes	Inserciones y eliminaciones

MÉTODOS

add()

Añade elementos
a la lista

```java
import java.util.ArrayList;

public class Main {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```

MÉTODOS

get()

Consulta un elemento

```
cars.get(0);
```

remove()

Elimina un elemento

```
cars.remove(0);
```

size()

Muestra el tamaño de la lista

```
cars.size();
```

set()

Modifica un elemento

```
cars.set(0, "Opel");
```

clear()

Elimina todos los elementos de la lista

```
cars.clear();
```

sort()

Ordena los elementos

```
Collections.sort(cars);  // Sort cars
for (String i : cars) {
  System.out.println(i);
}
```

MÉTODOS

addfirst()

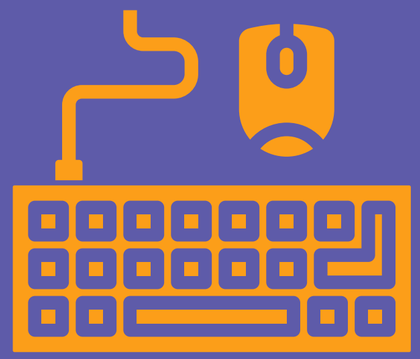
Añade un
elemento al inicio

addlast()

Añade un
elemento al final

removefirst()

Elimina un
elemento al inicio



removelast()

Elimina un
elemento al final

getfirst()

Consulta un
elemento al inicio

getlast()

Consulta un
elemento al final

PYTHON VS JAVA

```python
mi_lista = ["Hola", 23, True]
print(mi_lista[0])  # Hola
```

```python
import array

numeros = array.array('i', [1, 2, 3])
print(numeros[1])  # 2
```

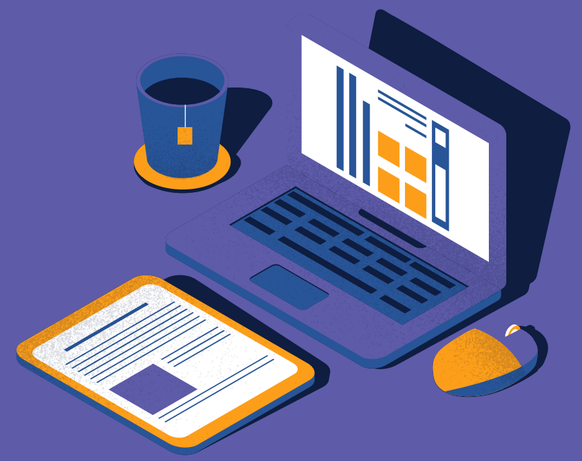
```java
import java.util.ArrayList;

ArrayList<String> nombres = new ArrayList<>();

nombres.add("Ana");
nombres.add("Luis");
System.out.println(nombres.get(0));  // Imprime "Ana"
```

```java
ArrayList<Persona> listaPersonas = new ArrayList<>();

listaPersonas.add(new Persona("Ana", 25, true));
listaPersonas.add(new Persona("Luis", 30, false));

// Mostrar datos
for (Persona p : listaPersonas) {
    System.out.println(p.nombre + " - " + p.edad + " años - Vacunado: " + p.vacunado);
}
```

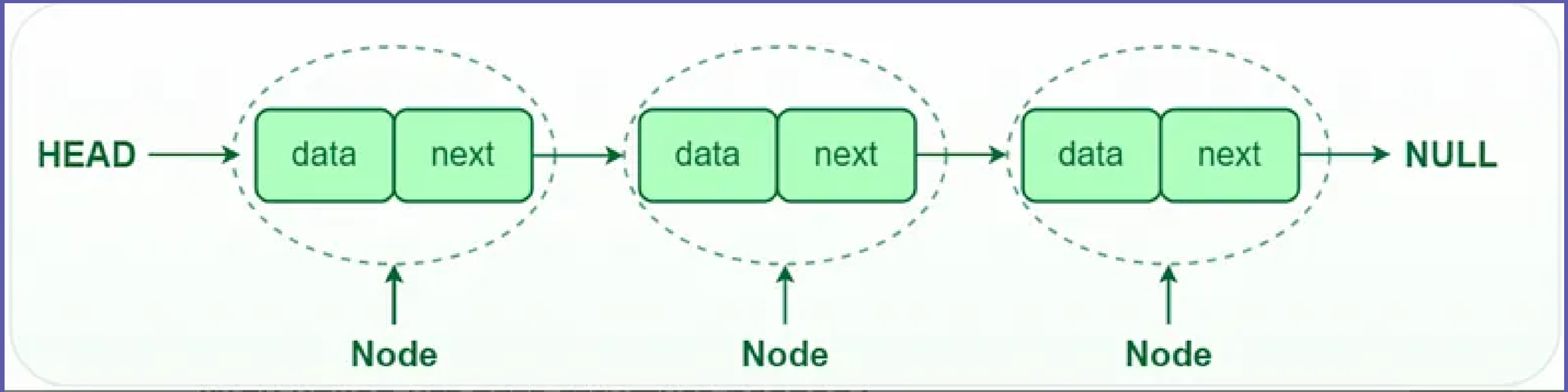
LISTAS SIMPLES ENLAZADAS

Una lista simple enlazada es una estructura de datos lineal compuesta por una secuencia de elementos llamados nodos. A diferencia de los arrays, las listas enlazadas no almacenan sus elementos en posiciones contiguas de memoria. En cambio, cada nodo contiene dos partes:

1. Dato: la información que se desea almacenar (por ejemplo, un número, una cadena, un objeto).
2. Apuntador (referencia): una referencia al siguiente nodo de la lista.



LISTAS SIMPLES ENLAZADAS



CLASE
Nodo{}

```
class Nodo {
    int dato;                // El dato almacenado en el nodo
    Nodo siguiente;          // Referencia al siguiente nodo

    public Nodo(int dato) {
        this.dato = dato;
        this.siguiente = null; // Al crear un nodo, no apunta a nadie
    }
}
```

CLASE

ListaEnlazada{}

```java
class ListaEnlazada {
    private Nodo inicio, fin;

    public ListaEnlazada() {
        inicio = fin = null;
    }

    // Verifica si la lista está vacía
    Windsurf: Refactor | Explain | ×
    public boolean estaVacia() {
        return inicio == null;
    }

    // Inserta un nodo al principio
    Windsurf: Refactor | Explain | ×
    public void insertarEnFrente(int dato) {
        Nodo nuevo = new Nodo(dato);
        if (estaVacia()) {
            inicio = fin = nuevo;
        } else {
            nuevo.siguiente = inicio;
            inicio = nuevo;
        }
    }
```

CLASE

ListaEnlazada{}

```java
// Inserta un nodo al final
Windsurf: Refactor | Explain | ×
public void insertarAlFinal(int dato) {
    Nodo nuevo = new Nodo(dato);
    if (estaVacia()) {
        inicio = fin = nuevo;
    } else {
        fin.siguiente = nuevo;
        fin = nuevo;
    }
}

// Elimina el primer nodo
Windsurf: Refactor | Explain | ×
public void eliminarEnFrente() {
    if (!estaVacia()) {
        System.out.println("Eliminado: " + inicio.dato);
        inicio = inicio.siguiente;
        if (inicio == null) { // Si la lista queda vacía
            fin = null;
        }
    }
}
```

CLASE

ListaEnlazada{}

```java
// Elimina el ultimo nodo
public void eliminarFinal() {
    if (inicio == fin) {
        System.out.println("Eliminado: " + inicio.dato);
        inicio = final = fin;
    }
    else {
        Nodo actual = inicio;
        while (actual.siguiente != null) {
            actual = actual.siguiente;
        }
        System.out.println("Eliminado: " + actual.dato);
        actual.siguiente = null;
        fin = actual;
    }
}

// Buscar un dato
public Nodo buscarNodo(int valor) {
    Nodo actual = inicio;
    while (actual != null) {
        if (actual.dato == valor) {
            return actual;
        }
        actual = actual.siguiente;
    }
    System.out.println("Nodo con valor " + valor + " no encontrado.");
    return null;
}
```

CLASE
ListaEnlazada{}

```java
// Actualizar un dato
public void actualizarNodo(int valorAntiguo) {
    Nodo actual = inicio;
    while (actual != null) {
        if (actual.dato == valorAntiguo) {
            System.out.print("Ingresa el nuevo valor: ");
            int nuevoValor = scanner.nextInt();
            actual.dato = valorNuevo;
            return;
        }
        actual = actual.siguiente;
    }
    System.out.println("Nodo con valor " + valorAntiguo + " no encontrado.");
}
// Imprimir lista
public void imprimirLista() {
    if (inicio == null) {
        System.out.println("La lista está vacía.");
        return;
    }
    Nodo actual = inicio;
    while (actual != null) {
        System.out.print(actual.dato + " -> ");
        actual = actual.siguiente;
    }
    System.out.println("null");
}
```

CLASE

Principal{}

```java
public class Principal {
    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        ListaEnlazada lista = new ListaEnlazada();
        lista.insertarAlFinal(dato:10);
        lista.insertarEnFrente(dato:5);
        lista.insertarAlFinal(dato:20);
        lista.imprimirLista();
        lista.eliminarEnFrente();
        lista.imprimirLista();
    }
}
```

Salida por consola

```
5 -> 10 -> 20 -> null
Eliminado: 5
10 -> 20 -> null
```

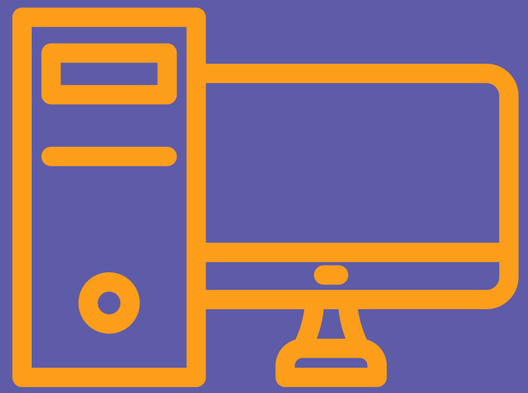
PYTHON VS JAVA

	Java	Python
¿Existe una lista enlazada propia?	Sí, con LinkedList en java.util	No directamente, debes crearla tú o usar collections.deque
¿Tipo de dato?	Estricto (int, String, etc.)	Dinámico (puede mezclar tipos)
Implementación desde cero	Necesita crear clases (Nodo, Lista)	Más simple y directa con clases
Métodos incorporados	Muchos: add(), remove(), get(), etc.	Si se implementa desde cero, todo se hace manual

```java
class Nodo {
    int dato;
    Nodo siguiente;

    public Nodo(int dato) {
        this.dato = dato;
        this.siguiente = null;
    }
}

class ListaEnlazada {
    Nodo cabeza;

    public void agregar(int dato) {
        Nodo nuevo = new Nodo(dato);
        if (cabeza == null) {
            cabeza = nuevo;
        } else {
            Nodo actual = cabeza;
            while (actual.siguiente != null) {
                actual = actual.siguiente;
            }
            actual.siguiente = nuevo;
        }
    }

    public void mostrar() {
        Nodo actual = cabeza;
        while (actual != null) {
            System.out.print(actual.dato + " → ");
            actual = actual.siguiente;
        }
        System.out.println("null");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        ListaEnlazada lista = new ListaEnlazada();
        lista.agregar(10);
        lista.agregar(20);
        lista.agregar(30);
        lista.mostrar();  // 10 → 20 → 30 → null
    }
}
```

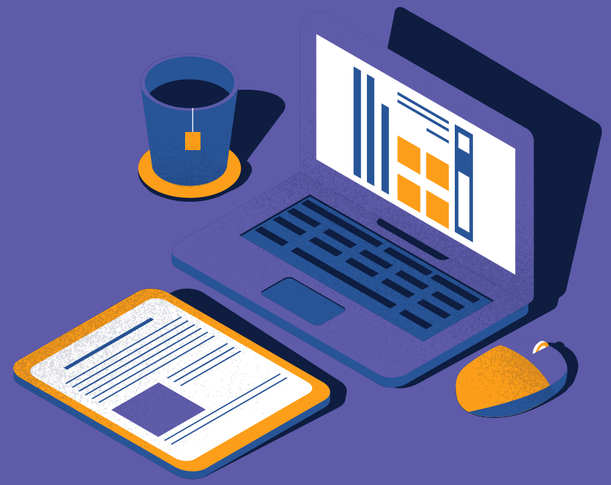
```python
class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.siguiente = None

class ListaEnlazada:
    def __init__(self):
        self.cabeza = None

    def agregar(self, dato):
        nuevo = Nodo(dato)
        if self.cabeza is None:
            self.cabeza = nuevo
        else:
            actual = self.cabeza
            while actual.siguiente:
                actual = actual.siguiente
            actual.siguiente = nuevo

    def mostrar(self):
        actual = self.cabeza
        while actual:
            print(f"{actual.dato} → ", end="")
            actual = actual.siguiente
        print("None")

# Ejemplo de uso
lista = ListaEnlazada()
lista.agregar(10)
lista.agregar(20)
lista.agregar(30)
lista.mostrar()  # 10 → 20 → 30 → None
```

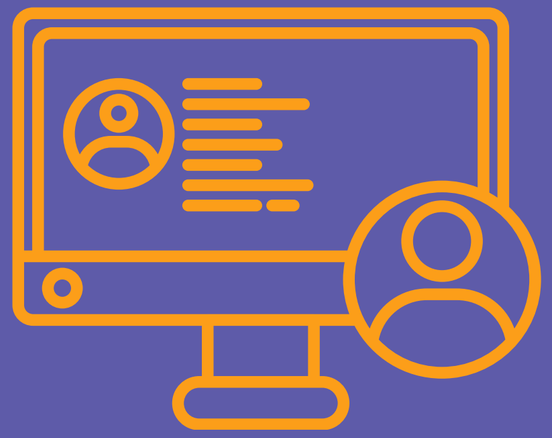


CUANDO USAR ARRAY LIST

- Necesitas acceder rápidamente a los elementos por su índice.
- La cantidad de datos no cambia demasiado (o cambia principalmente al final).
- Te importa el rendimiento en lectura más que en inserción o eliminación.
- No te importa que el array tenga que crecer (redimensionarse) ocasionalmente.

Ejemplo típico: mostrar un catálogo de productos donde se accede mucho por índice.





CUANDO USAR LINKED LIST

- Inserciones y eliminaciones son frecuentes, especialmente al principio.
- No necesitas acceso rápido por índice.
- Quieres evitar el costo de redimensionar arrays.
- Estás trabajando con una estructura dinámica, que crece y se reduce mucho.

Ejemplo típico: una cola de impresión o historial de operaciones, donde agregas o quitas elementos constantemente.

GRACIAS

GIT CLONE : HTTPS://GITHUB.COM/DAVIDRUEDA-Z/LISTAS.GIT