

Licenciatura en Ingeniería en Tecnologías de la Información e Innovación  
Digital

Desarrollo de Software Multiplataforma

Estructura de Datos

UNIDAD II

Listas Enlazadas simples, dobles, circulares

Estructuras de datos básicas

Hernández Torrez Alondra Vianney -1224100684

Grupo: GTID 141

Docente:

Gabriel Barrón Rodríguez

Dolores Hidalgo. C.I.N. Gto, Martes 14 de Octubre de 2025.

## Estructura de Datos

```

Inicio
temp ← cabeza
while temp ≠ nulo
    temp ← temp.derecha
Fin While
temp.derecha ← nulo (data)
temp.derecha ← izquierda ← temp
Fin
    
```

```

// Temp ← cabeza
// Si temp.siguiente ≠ igual a Nulo
temp.anterior ← Temp.
    
```

While temp.siguiente ≠ distinto a nulo

## Leer y Escribir

```

Inicio
temp ← cabeza,
imprimir temp,
while temp ≠ null
    temp ← temp.siguiente
    imprimir temp,
Fin While
Fin
    
```

listas

// Listas enlazadas dobles

Nodo Sara

Sara.siguiente = null

Sara.anterior = null

Si (Cabeza == null) {  
    Cabeza = Nodo(Sara);

Si No { Insertar

temp = Cabeza;

while (temp.siguiente != null)

temp = temp.siguiente;

Fin while

temp.siguiente = Nodo

Sara, anterior = temp

Fin

1 Cabeza

2 Cabeza / / / / Sara / /

3 Cabeza / /  
↑  
temp / /  
/ / Sara / /

4 / / Cabeza / / Sara / /  
↑  
temp / /  
/ / Sara / /

5 Cabeza / / Sara / /

// Lista Circular

Inicio

Nodo nuevo

nuevo.dato ← "Sara"

nuevo.siguiente ← null

Si (Cabeza == null) { Enlace

Cabeza ← Nuevo

nuevo.siguiente ← Cabeza

// Se apunta a si mismo

Si No

temp ← Cabeza

Mientras (temp.siguiente != Cabeza)

Hacer

temp ← temp.siguiente

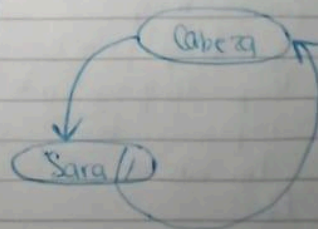
Fin While

temp.siguiente ← Nuevo

nuevo.siguiente ← Cabeza

Fin Si

Fin



## Implementación a Java

### SIMPLE

//Alondra Vianney Hernandez Torres

// Clase que representa un nodo de una lista enlazada simple

```
class NodoSimple {
```

```
    String dato;          // Almacena el valor del nodo
```

```
    NodoSimple siguiente; // Referencia al siguiente nodo en la lista
```

```
    // Constructor que inicializa el nodo con un valor
```

```
    public NodoSimple(String dato) {
```

```
        this.dato = dato;    // Asigna el valor al nodo
```

```
        this.siguiente = null; // Inicialmente no apunta a ningún nodo
```

```
    }
```

```
}
```

// Clase que representa la lista enlazada simple

```
class ListaSimple {
```

```
    NodoSimple cabeza; // Primer nodo de la lista
```

```
    // Método para insertar un nuevo nodo al final de la lista
```

```
    public void insertar(String valor) {
```

```
        NodoSimple nuevo = new NodoSimple(valor); // Crear un nuevo nodo con el valor  
        dado
```

```
        if (cabeza == null) {    // Si la lista está vacía
```

```
            cabeza = nuevo;      // El nuevo nodo se convierte en la cabeza
```

```
        } else {                // Si la lista no está vacía
```

```
            NodoSimple temp = cabeza;    // Comenzar desde la cabeza
```

```
            while (temp.siguiente != null) { // Recorrer hasta el último nodo
```

```
                temp = temp.siguiente;    // Moverse al siguiente nodo
```

```
            }
```

```
            temp.siguiente = nuevo;      // Enlazar el nuevo nodo al final
```

```
        }
```

```
    }
```

```
}
```

## DOBLE

//Alondra Vianney Hernandez Torres

```
class NodoDoble {
    String dato;
    NodoDoble siguiente; // Referencia al siguiente nodo en la lista
    NodoDoble anterior; // Referencia al nodo anterior en la lista

    // Constructor que inicializa el nodo con un valor
    public NodoDoble(String dato) {
        this.dato = dato; // Asigna el valor al nodo
        this.siguiente = null; // Inicialmente no apunta a ningún nodo siguiente
        this.anterior = null; // Inicialmente no apunta a ningún nodo anterior
    }
}

// Clase que representa la lista doblemente enlazada
class ListaDoble {
    NodoDoble cabeza; // Primer nodo de la lista

    // Método para insertar un nuevo nodo al final de la lista
    public void insertar(String valor) {
        NodoDoble nuevo = new NodoDoble(valor); // Crear un nuevo nodo con el valor dado

        if (cabeza == null) { // Si la lista está vacía
            cabeza = nuevo; // El nuevo nodo se convierte en la cabeza
        } else { // Si la lista no está vacía
            NodoDoble temp = cabeza; // Comenzar desde la cabeza
            while (temp.siguiente != null) { // Recorrer hasta el último nodo
                temp = temp.siguiente; // Moverse al siguiente nodo
            }
            temp.siguiente = nuevo; // Enlazar el nuevo nodo al final
            nuevo.anterior = temp; // Establecer el enlace del nuevo nodo hacia atrás
        }
    }
}
```

## CIRCULAR

//Alondra Vianney Hernandez Torres

```
class Nodo {
    String dato;
    Nodo siguiente;

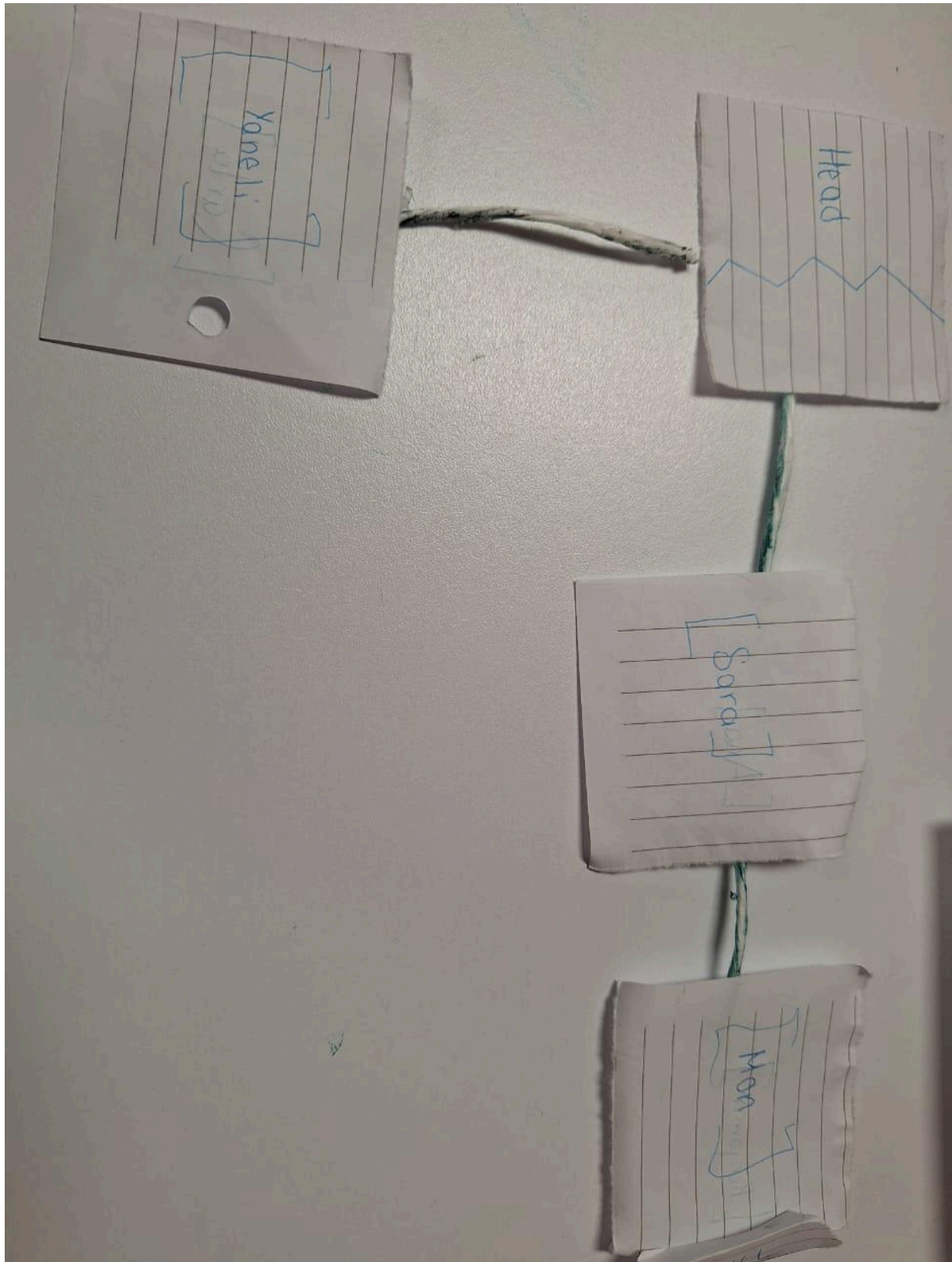
    // Constructor que inicializa el nodo con un valor
    public Nodo(String dato) {
        this.dato = dato;
        this.siguiente = null;
    }
}

class ListaCircular {
    Nodo cabeza; // Primer nodo de la lista

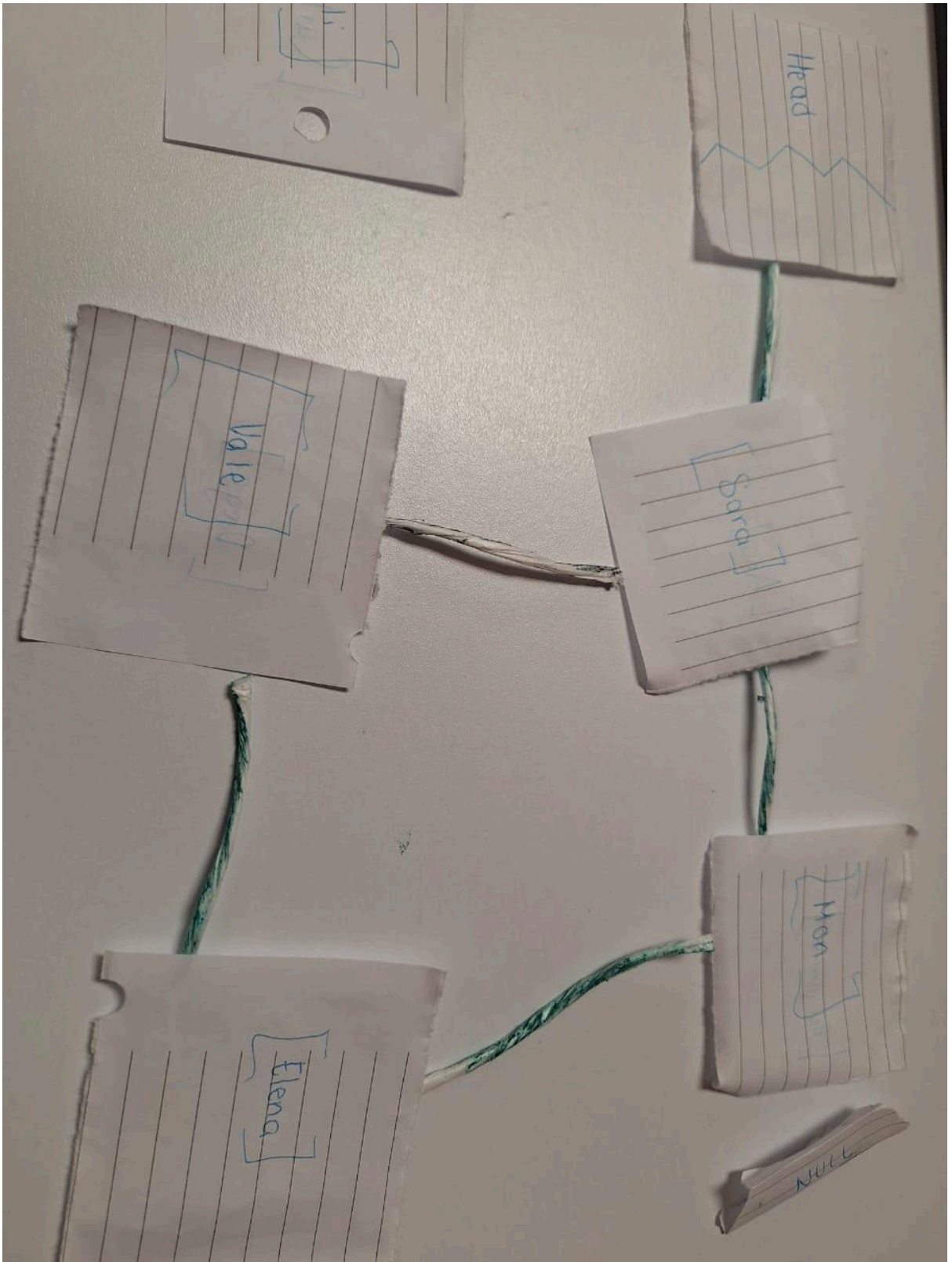
    // Método para insertar un nuevo nodo en la lista circular
    public void insertar(String valor) {
        Nodo nuevo = new Nodo(valor); // Crear un nuevo nodo con el valor dado

        if (cabeza == null) {          // Si la lista está vacía
            cabeza = nuevo;            // El nuevo nodo se convierte en la cabeza
            nuevo.siguiente = cabeza;   // Apunta a sí mismo para formar la circularidad
        } else {                      // Si la lista no está vacía
            Nodo temp = cabeza;        // Comenzar desde la cabeza
            while (temp.siguiente != cabeza) { // Recorrer hasta el último nodo (que apunta a la
cabeza)
                temp = temp.siguiente; // Moverse al siguiente nodo
            }
            temp.siguiente = nuevo;    // El último nodo apunta al nuevo nodo
            nuevo.siguiente = cabeza;  // El nuevo nodo apunta a la cabeza, cerrando el círculo
        }
    }
}
```

## EVIDENCIAS ACTIVIDAD EN CLASE

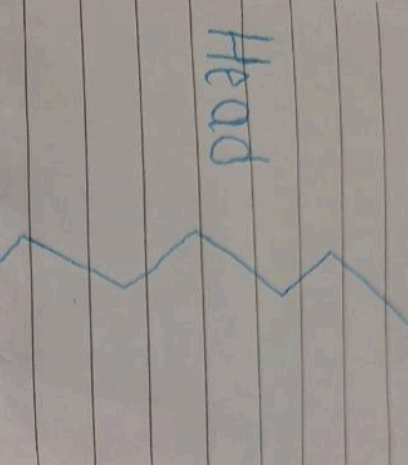






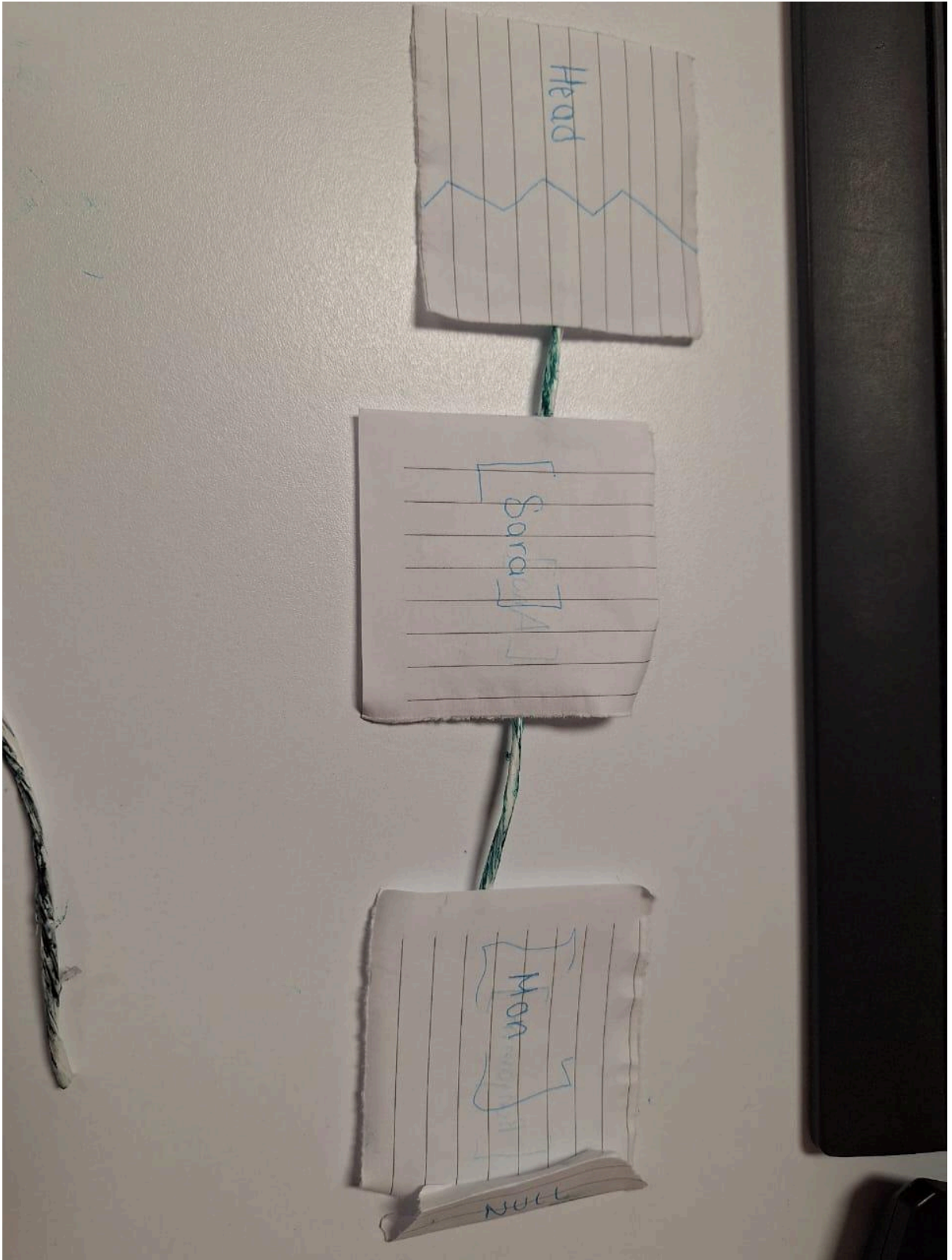


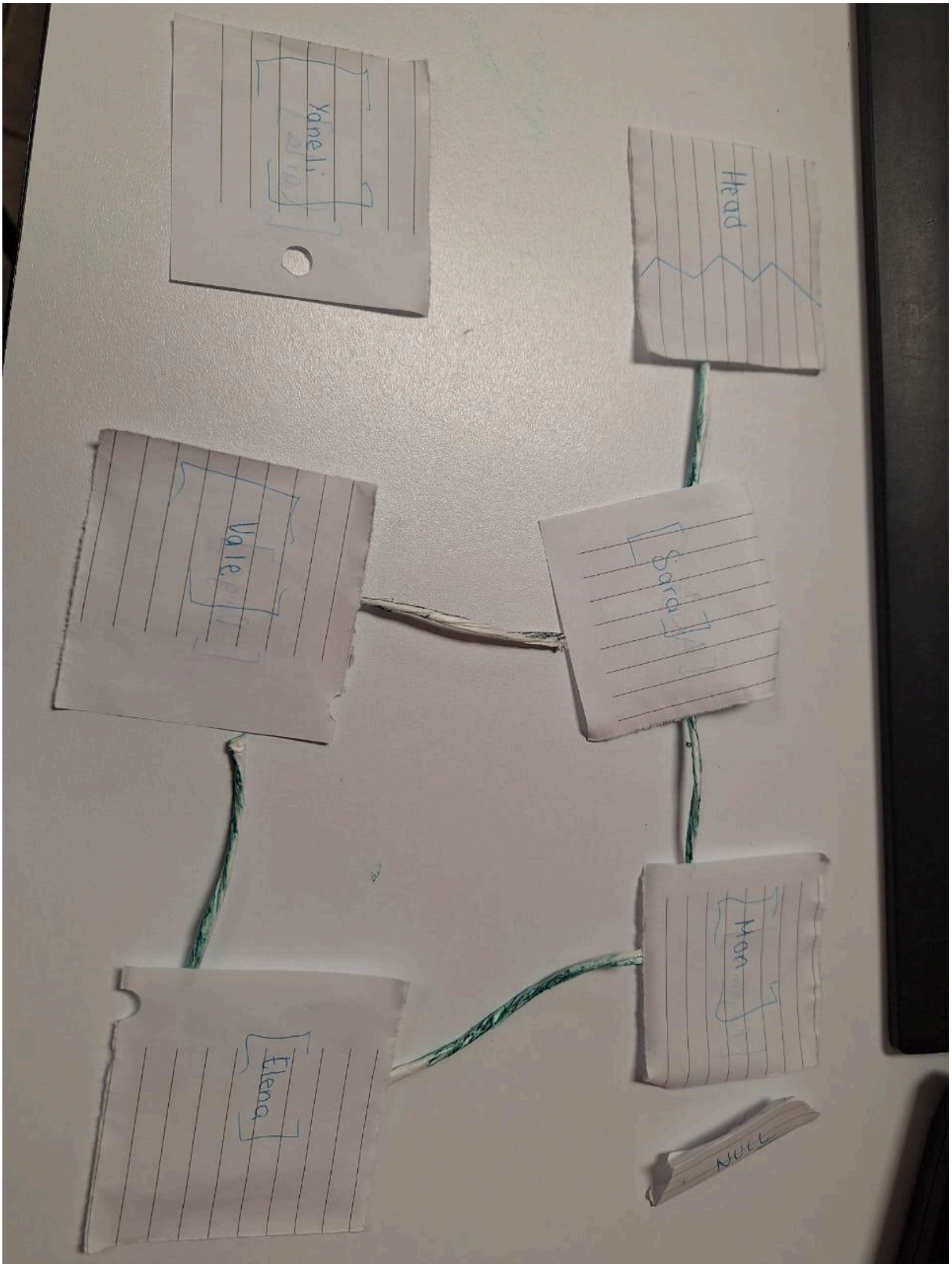
Head



Sara

Sara





## Reflexión final

**1. Describe con tus propias palabras los conceptos de lista simple, doble y circular.**

La lista simple conecta los nodos en una sola dirección, la doble lo hace hacia adelante y hacia atrás, y la circular une el último nodo con el primero.

**2. ¿Qué tipo de lista es más eficiente para insertar y eliminar en cualquier posición?**

La lista doble, porque permite moverse en ambos sentidos y facilita las operaciones.

**3. ¿Qué ventaja tiene la lista circular frente a la simple?**

Que no tiene final, se puede recorrer continuamente sin llegar a un null.

**4. ¿Qué sucede si se rompe un enlace en una lista doble?**

Se pierde la conexión entre los nodos y la lista puede quedar incompleta o dañada.

**5. ¿Cómo se representa el “NULL” en una lista circular?**

En una lista circular no existe el null, porque el último nodo apunta al primero.