Universidad de Cundinamarca

Estructura de Datos

R3-A1-S12 Notebook

Javier Mateo Barrero Vanegas

Jim Alejandro Quiñones Martínez

Luis Ángel Martínez Cuenca

Harick Yesid Villarraga Rincon

Cristian Esteban Ruiz Parra

Docente: Dilia Ines Molina Cubillos

30 octubre de 2025

Introducción.

En el presente informe se desarrolla una investigación y un trabajo práctico enfocado en el estudio y aplicación de las listas circulares dinámicas, una de las estructuras de datos más utilizadas en la programación por su eficiencia y flexibilidad en el manejo de información. Estas listas se caracterizan por mantener una conexión continua entre los nodos, de modo que el último elemento enlaza nuevamente con el primero, formando un ciclo cerrado que permite recorrer la estructura de manera indefinida. El objetivo de esta investigación es comprender el funcionamiento, las ventajas y los usos principales de las listas circulares, además de implementarlas en un programa práctico que demuestre cómo se pueden insertar, eliminar y recorrer datos de forma dinámica. A través de este trabajo se busca fortalecer el conocimiento sobre estructuras enlazadas y su importancia en el diseño de algoritmos eficientes y sistemas que requieren manejo continuo de datos, como los buffers, colas circulares o procesos cíclicos en sistemas operativos.

Listas Circulares Dinámicas.

Una lista circular dinámica es una estructura de datos enlazada en la que los nodos se conectan de manera que el último nodo apunta nuevamente al primero formando un “circuito cerrado” y se denomina dinámica porque su tamaño no es fijo ya que los nodos pueden agregarse o eliminarse en tiempo de ejecución ya que se almacenan en memoria dinámica mediante punteros o referencias.

Cada nodo de la lista circular contiene al menos dos partes, un dato que es la información que almacena (un número, una palabra o un objeto) y un enlace que es una referencia o puntero que conecta con otro nodo.

- Algunos tipos de listas circulares dinámicas son…

Lista circular simplemente enlazada: en esta, cada nodo tiene un enlace que apunta al siguiente, y el último apunta al primero.

Lista circular doblemente enlazada: en la que cada nodo tiene dos enlaces: uno hacia el nodo siguiente y otro hacia el anterior, lo que permite recorrer la lista en ambas direcciones.

- Ejemplo de uso cotidiano…

La situación en que existen tres canciones en una lista de reproducción y cuando se termina la última canción, la reproducción continúa desde la primera sin detenerse.

En los juegos se usa para controlar turnos de jugadores de forma cíclica.

- Ventajas del modelo…

Permite recorridos infinitos sin necesidad de reiniciar manualmente la posición, es más eficientes que en estructuras estáticas como los arreglos ya que no requiere definir un tamaño fijo al inicio y es ideal para procesos que se repiten en ciclo continuo.

- Y algunas desventajas son:

El manejo de punteros puede ser más complejo especialmente en listas doblemente enlazadas.

Requiere más memoria por los enlaces adicionales.

Y si no se controla bien el recorrido, puede causar bucles infinitos en el programa.

CODIGO.

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;

public class ListaCc extends JFrame {
    // --- Clase interna Nodo ---
    class Nodo {
        String cedula, nombre;
        Nodo siguiente ;
        Nodo(String c, String n) { cedula = c; nombre = n; }
    }

    // --- Lista simple circular ---
    Nodo inicio =null;
void insertar(String c, String n) {
    Nodo Nueva = new Nodo(c, n) ;
     if (inicio == null) { // lista vacia
        inicio = Nueva;
        Nueva.siguiente = Nueva;
    }else{
         Nodo temp = inicio;
         while(temp.siguiente !=inicio)
             temp=temp.siguiente;
         temp.siguiente=Nueva;
```

```
        Nueva.siguiente = inicio.siguiente;
        inicio = Nueva;     // el inicio queda en el ultimo nodo
    }
}
    java.util.List<String> mostrarDerecha() {
        java.util.List<String> datos = new ArrayList<>();
        if (inicio == null) return datos;
        Nodo temp = inicio.siguiente;   // Primer nodo (después del último)
        do {
            datos.add(temp.cedula + " - " + temp.nombre);
            temp = temp.siguiente;
        } while (temp != inicio.siguiente);
        return datos;
    }

    // --- Interfaz gráfica ---
    JTextField txtCed = new JTextField(8);
    JTextField txtNom = new JTextField(10);
    JTextArea area = new JTextArea(10, 25);



    public ListaCc() {
        super("◈ Lista Simple Circular - Estilo Moderno ◈");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(450, 480);
        setLocationRelativeTo(null);
```

```
        // 🎨 Fondo con degradado
JPanel fondo = new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        int w = getWidth();
        int h = getHeight();

        // Degradado principal
        Color color1 = new Color(58, 123, 213);
        Color color2 = new Color(91, 192, 222);
        GradientPaint gp = new GradientPaint(0, 0, color1, 0, h, color2);
        g2d.setPaint(gp);
        g2d.fillRect(0, 0, w, h);

        // Figuras difuminadas
        g2d.setColor(new Color(255, 255, 255, 40));
        g2d.fillOval(30, 60, 140, 140);
        g2d.fillOval(w - 180, h - 180, 200, 200);
    }
};
fondo.setLayout(new FlowLayout(FlowLayout.CENTER, 15, 15));
setContentPane(fondo);
```

```
        // 🎨 Estilo de campos
        txtCed.setBackground(new Color(255, 255, 255, 220));
        txtNom.setBackground(new Color(255, 255, 255, 220));
        txtCed.setFont(new Font("SansSerif", Font.PLAIN, 14));
        txtNom.setFont(new Font("SansSerif", Font.PLAIN, 14));
        txtCed.setBorder(BorderFactory.createLineBorder(new Color(91, 192, 222), 2,
true));
        txtNom.setBorder(BorderFactory.createLineBorder(new Color(91, 192, 222), 2,
true));


        // 🧩 Restricción: cédula solo números
        txtCed.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent e) {
                char c = e.getKeyChar();
                if (!Character.isDigit(c) && c != '\b' && c != 127) {
                    e.consume();
                    Toolkit.getDefaultToolkit().beep();
                }
            }
        });


        // 🎨 Área de texto
        area.setBackground(new Color(255, 255, 255, 230));
        area.setFont(new Font("Monospaced", Font.PLAIN, 13));
        area.setBorder(BorderFactory.createTitledBorder(
```

```
        BorderFactory.createLineBorder(new Color(0, 153, 204), 2, true),
        "Clientes registrados",
        0, 0,
        new Font("SansSerif", Font.BOLD, 12),
        new Color(0, 102, 153)
));

// 🎨 Botones modernos
JButton bIns = new JButton("➕ Insertar cliente");
JButton bDer = new JButton("📜 Listar clientes");
JButton bSalir = new JButton("❌ Salir");

Color base = new Color(0, 153, 204);
Color hover = new Color(0, 204, 255);

JButton[] botones = {bIns, bDer, bSalir};
for (JButton b : botones) {
    b.setBackground(base);
    b.setForeground(Color.WHITE);
    b.setFocusPainted(false);
    b.setFont(new Font("SansSerif", Font.BOLD, 13));
    b.setBorder(BorderFactory.createEmptyBorder(6, 15, 6, 15));
    b.setCursor(new Cursor(Cursor.HAND_CURSOR));

    b.addMouseListener(new java.awt.event.MouseAdapter() {
```

```
            public void mouseEntered(java.awt.event.MouseEvent evt) {
b.setBackground(hover); }

            public void mouseExited(java.awt.event.MouseEvent evt) {
b.setBackground(base); }

        });

    }


    // 🧩 Componentes

    JLabel lblCed = new JLabel("Cédula:");

    JLabel lblNom = new JLabel("Nombre:");

    lblCed.setForeground(Color.WHITE);

    lblNom.setForeground(Color.WHITE);

    lblCed.setFont(new Font("SansSerif", Font.BOLD, 14));

    lblNom.setFont(new Font("SansSerif", Font.BOLD, 14));


    fondo.add(lblCed); fondo.add(txtCed);

    fondo.add(lblNom); fondo.add(txtNom);

    fondo.add(bIns); fondo.add(bDer); fondo.add(bSalir);

    fondo.add(new JScrollPane(area));


    // ⚙ Acciones

  bIns.addActionListener(e -> {

    if (txtCed.getText().isEmpty() || txtNom.getText().isEmpty()) {

        JOptionPane.showMessageDialog(this, "Complete ambos campos");

    } else {

        insertar(txtCed.getText(), txtNom.getText());

        txtCed.setText("");
```

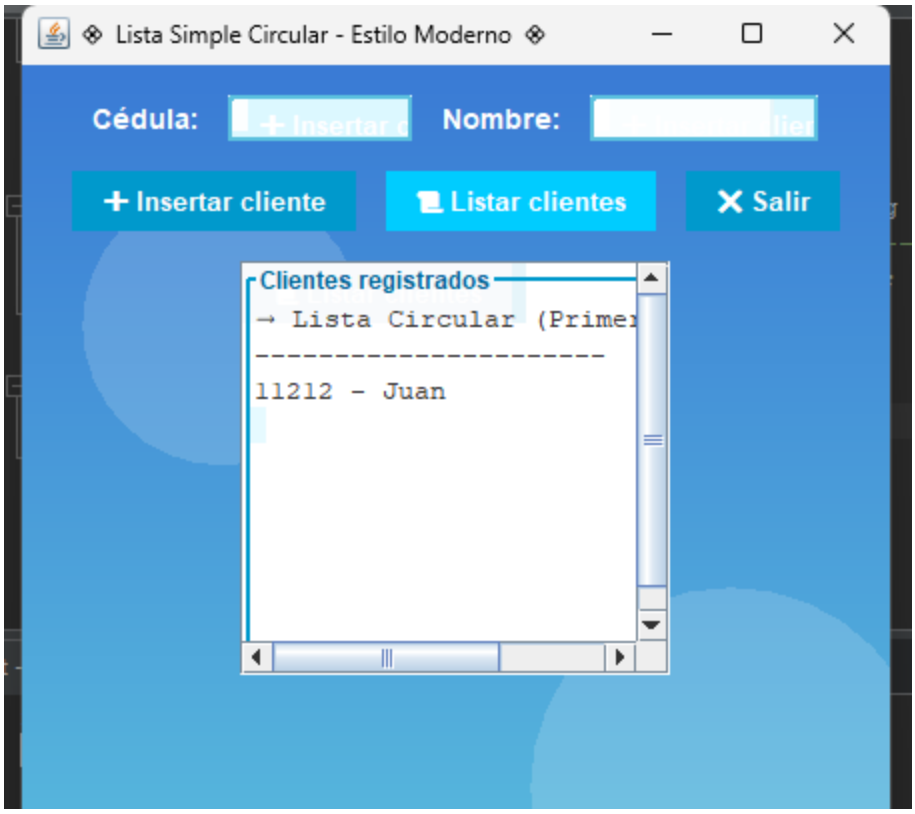
```
            txtNom.setText("");
            // 👇 Nueva línea agregada: muestra automáticamente todos los datos
            mostrar(mostrarDerecha(), "→ Lista Circular (Primer a Último)");
        }
    });

    bDer.addActionListener(e -> mostrar(mostrarDerecha(), "→ Lista Circular (Primer a Último)"));
    bSalir.addActionListener(e -> System.exit(0));

    setVisible(true);
}

    void mostrar(java.util.List<String> datos, String titulo) {
        area.setText(titulo + "\n----------\n");
        for (String s : datos) area.append(s + "\n");
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ListaCc::new);
    }
}
```


Lista Simple Circular - Estilo Moderno
Cédula:
Nombre:
+ Insertar cliente
Listar clientes
Salir
Clientes registrados
→ Lista Circular (Prime
11212 - Juan

Conclusión

Después del estudio y la implementación de las listas circulares dinámicas en el lenguaje de Java pudimos comprender cómo las estructuras de datos pueden adaptarse a diferentes necesidades dentro de la programación. Este tipo de lista ofrece una forma eficiente de almacenar, recorrer y manipular información de manera continua, gracias a su estructura en la que el último nodo se enlaza nuevamente con el primero, formando un ciclo sin fin. Con el trabajo práctico evidenciamos que las listas circulares son especialmente útiles en procesos repetitivos o cíclicos, como el control de turnos. Además, su carácter dinámico permite añadir o eliminar elementos sin necesidad de redimensionar toda la estructura, optimizando el uso de la memoria.

En conclusión, las listas circulares dinámicas son una herramienta fundamental dentro de la programación estructurada y orientada a objetos, ya que combinan eficiencia, flexibilidad y reutilización, aspectos esenciales para el desarrollo de algoritmos y sistemas más robustos y funcionales.