Licenciatura en Ingeniería en Tecnologías de la Información e Innovación Digital

Desarrollo de Software Multiplataforma

Estructura de Datos

UNIDAD II

Estructuras de datos básicas

# Listas

Hernández Torrez Alondra Vianney -1224100684

Grupo: GTID 141

Docente:

Gabriel Barrón Rodríguez

**Dolores Hidalgo. C.I.N. Gto, Martes 21 de Octubre de 2025.**

**Actividad 01**

**MANIPULACIÓN DE LI□TA ENLAZADA** □˚. .˚□

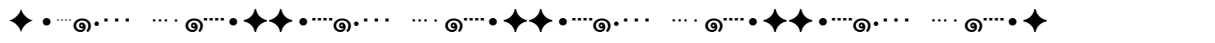
**Objetivo:** Desarrollar un programa que implemente operaciones básicas sobre una lista enlazada simple de números enteros positivos.

✦•···๑·····๑···•✦✦•···๑·····๑···•✦✦•···๑·····๑···•✦✦•···๑·····๑···•✦

```
package Listas;
import java.util.Random;
import java.util.Scanner;

/**
 *
 * @author Alondra Vianney Hernandez Torres
 */
// Actividad 01: Manipulación de Lista Enlazada
// Descripción: Programa que crea, recorre y elimina nodos
mayores a un valor dado en una lista enlazada simple.



// Clase Nodo representa un elemento de la lista
class Nodo {
    int dato;       // valor almacenado
    Nodo siguiente; // referencia al siguiente nodo

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

// Clase ListaEnlazada para gestionar los métodos solicitados
class ListaEnlazada {
    Nodo cabeza;

    // Crea la lista enlanzada con numeros aleatorios
    public void crearLista(int cantidad) {
        Random random = new Random();
        for (int i = 0; i < cantidad; i++) {
```

```
            int numero = random.nextInt(100) + 1; // genera
número entre 1 y 100
            insertarFinal(numero);
        }
    }

    // Inserta al final de la lista
    public void insertarFinal(int valor) {
        Nodo nuevo = new Nodo(valor);
        if (cabeza == null) {
            cabeza = nuevo;
        } else {
            Nodo temp = cabeza;
            while (temp.siguiente != null) {
                temp = temp.siguiente;
            }
            temp.siguiente = nuevo;
        }
    }

    // Recorre la lista y muestra los datos
    public void recorrerLista() {
        if (cabeza == null) {
            System.out.println("La lista está vacía.");
            return;
        }
        Nodo temp = cabeza;
        System.out.println("Contenido de la lista:");
        while (temp != null) {
            System.out.print(temp.dato + " -> ");
            temp = temp.siguiente;
        }
        System.out.println("null");
    }

    //Elimina nodos mayores dados que
    public void eliminarMayoresA(int limite) {
        // Eliminar nodos desde la cabeza si son mayores
        while (cabeza != null && cabeza.dato > limite) {
            cabeza = cabeza.siguiente;
        }

        // Recorrer el resto de la lista
        Nodo actual = cabeza;
```

```
        while (actual != null && actual.siguiente != null) {
            if (actual.siguiente.dato > limite) {
                actual.siguiente = actual.siguiente.siguiente;
            } else {
                actual = actual.siguiente;
            }
        }
    }
}

/**
 *
 * @author Alondra Vianney Hernandez Torres
 */

public class MainLista {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ListaEnlazada lista = new ListaEnlazada();

        System.out.println("ACTIVIDAD 01: MANIPULACIÓN DE
LISTA ENLAZADA");

        // Crear lista con números aleatorios
        System.out.print("¿Cuántos números desea generar? ");
        int n = sc.nextInt();
        lista.crearLista(n);
        System.out.println("\nLista generada:");
        lista.recorrerLista();

        // Eliminar nodos mayores a un valor
        System.out.print("\nIngrese un valor límite: ");
        int limite = sc.nextInt();
        lista.eliminarMayoresA(limite);

        System.out.println("\nLista después de eliminar nodos
mayores a " + limite + ":");
        lista.recorrerLista();
    }
}
```



**Actividad 02**

Lista Enlazada de Palabras desde Archivo □˚. .˚□

**Objetivo:** Desarrollar un programa que lea palabras desde un archivo de texto y las almacene en una lista enlazada, permitiendo su manipulación dinámica.

```
import java.io.*;
import java.util.*;
/**
 *
 * @author Alondra Vianney Hernandez Torres
 */
//Actividad 02: Lista Enlazada de Palabras desde Archivo
//Objetivo:
//Desarrollar un programa que lea palabras desde un archivo de
texto y las almacene en una
//lista enlazada, permitiendo su manipulación dinámica.


public class ListaArchivo {
    // Clase Nodo: cada nodo tiene una palabra y el enlace al
siguiente
    class Nodo {
        String palabra;
        Nodo sig;
        Nodo(String palabra) {
            this.palabra = palabra;
            this.sig = null;
        }
    }

    Nodo cabeza = null; // inicio de la lista

    // Inserta una palabra al final de la lista
    void insertar(String palabra) {
        Nodo nuevo = new Nodo(palabra);
        if (cabeza == null) cabeza = nuevo;
        else {
            Nodo temp = cabeza;
            while (temp.sig != null) temp = temp.sig;
            temp.sig = nuevo;
```

```
        }
    }

    // Elimina la primera aparición de la palabra
    boolean eliminar(String palabra) {
        if (cabeza == null) return false;
        if (cabeza.palabra.equals(palabra)) {
            cabeza = cabeza.sig;
            return true;
        }
        Nodo ant = cabeza, temp = cabeza.sig;
        while (temp != null) {
            if (temp.palabra.equals(palabra)) {
                ant.sig = temp.sig;
                return true;
            }
            ant = temp;
            temp = temp.sig;
        }
        return false;
    }

    // Muestra las palabras en consola
    void mostrar() {
        Nodo temp = cabeza;
        while (temp != null) {
            System.out.print(temp.palabra + " ");
            temp = temp.sig;
        }
        System.out.println();
    }

    // Lee palabras desde un archivo y las guarda en la lista
    void leerArchivo(String nombreArchivo) {
        try (Scanner sc = new Scanner(new
File(nombreArchivo))) {
            while (sc.hasNext()) insertar(sc.next());
        } catch (IOException e) {
            System.out.println("Error leyendo archivo: " +
e.getMessage());
        }
    }

    // Escribe todas las palabras de la lista al archivo
```

```
    void escribirArchivo(String nombreArchivo) {
        try (PrintWriter pw = new PrintWriter(new
FileWriter(nombreArchivo))) {
            Nodo temp = cabeza;
            while (temp != null) {
                pw.print(temp.palabra + " ");
                temp = temp.sig;
            }
        } catch (IOException e) {
            System.out.println("Error escribiendo archivo: " +
e.getMessage());
        }
    }

    //Main
    public static void main(String[] args) {
        ListaArchivo lista = new ListaArchivo();
        Scanner sc = new Scanner(System.in);
        String archivo = "palabras.txt";

        lista.leerArchivo(archivo);
        int opc;
        do {
            System.out.println("\n1. Mostrar\n2. Añadir\n3.
Eliminar\n4. Guardar y salir");
            System.out.print("Opción: ");
            opc = sc.nextInt();
            sc.nextLine(); // limpiar buffer

            switch (opc) {
                case 1 -> lista.mostrar();
                case 2 -> {
                    System.out.print("Palabra nueva: ");
                    lista.insertar(sc.nextLine());
                }
                case 3 -> {
                    System.out.print("Palabra a eliminar: ");
                    if (lista.eliminar(sc.nextLine()))
System.out.println("Eliminada.");
                    else System.out.println("No encontrada.");
                }
                case 4 -> {
                    lista.escribirArchivo(archivo);
                    System.out.println("Archivo guardado.");
```

```
                }
            }
        } while (opc != 4);
    }
}
```

✦•···ɑ·····ɑ···•✦✦•···ɑ·····ɑ···•✦✦•···ɑ·····ɑ···•✦✦•···ɑ·····ɑ···•✦✦•···ɑ·····ɑ···•✦

**Actividad 03**

Representación y Evaluación de Polinomios con Listas Enlazadas

**Objetivo:** Desarrollar un programa que permita representar polinomios mediante una lista enlazada simple y calcular sus valores para distintos puntos de evaluación.

```
import java.util.Scanner;
/**
 *
 * @author Alondra Vianney Hernandez Torres
 */




class Nodo {
    double coeficiente;
    int exponente;
    Nodo siguiente;

    public Nodo(double coeficiente, int exponente) {
        this.coeficiente = coeficiente;
        this.exponente = exponente;
        this.siguiente = null;
    }
}

class ListaPolinomio {
    private Nodo cabeza;

    public void agregarTermino(double coeficiente, int
exponente) {
        Nodo nuevo = new Nodo(coeficiente, exponente);
        if (cabeza == null) {
            cabeza = nuevo;
        } else {
```

```
            Nodo actual = cabeza;
            while (actual.siguiente != null) {
                actual = actual.siguiente;
            }
            actual.siguiente = nuevo;
        }
    }

    // Evalúa el polinomio en un valor x
    public double evaluar(double x) {
        double resultado = 0.0;
        Nodo actual = cabeza;
        while (actual != null) {
            resultado += actual.coeficiente * Math.pow(x,
actual.exponente);
            actual = actual.siguiente;
        }
        return resultado;
    }

    // Muestra la lista como polinomio
    public void mostrarPolinomio() {
        Nodo actual = cabeza;
        System.out.print("P(x) = ");
        while (actual != null) {
            System.out.print(actual.coeficiente + "x^" +
actual.exponente);
            if (actual.siguiente != null) System.out.print(" +
");
            actual = actual.siguiente;
        }
        System.out.println();
    }
}

public class Listasss{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ListaPolinomio polinomio = new ListaPolinomio();

        System.out.println("=== Representación y Evaluación de
Polinomios con Listas Enlazadas ===");
        System.out.print("¿Cuántos términos tendrá el
polinomio?: ");
```

```
        int n = sc.nextInt();

        // Entrada de términos
        for (int i = 0; i < n; i++) {
            System.out.println("\nTérmino " + (i + 1) + ":");
            System.out.print("Coeficiente: ");
            double coef = sc.nextDouble();
            System.out.print("Exponente: ");
            int exp = sc.nextInt();
            polinomio.agregarTermino(coef, exp);
        }

        // Mostrar polinomio
        System.out.println();
        polinomio.mostrarPolinomio();

        // Evaluación del polinomio
        System.out.println("\nTabla de valores:");
        System.out.println(" x\t|\tP(x)");
        System.out.println("------------------------");
        for (double x = 0.0; x <= 5.0; x += 0.5) {
            double px = polinomio.evaluar(x);
            System.out.printf(" %.1f\t|\t%.4f\n", x, px);
        }

        sc.close();
    }
}
```

✦•·····๑·····๑•✦✦•·····๑·····๑•✦✦•·····๑·····๑•✦✦•·····๑·····๑•✦✦•·····๑·····๑•✦

**Actividad 04**

Polinomio con Lista Enlazada Circular ˚. .˚

**Objetivo:** Modificar la representación de un polinomio mediante lista enlazada simple para que se convierta en una lista circular, optimizando el acceso y recorrido continuo.

```
import java.util.Scanner;
/**
 *
 * @author Alondra Vianney Hernandez Torres
 */
```

```
//Actividad 04
//Polinomio con Lista Enlazada Circular
//Objetivo: Modificar la representación de un polinomio
mediante lista enlazada simple para que se convierta en una
lista circular, optimizando el acceso y recorrido continuo.

class Nodo {
    double coeficiente;
    int exponente;
    Nodo siguiente;

    public Nodo(double coef, int exp) {
        this.coeficiente = coef;
        this.exponente = exp;
        this.siguiente = null;
    }
}

// Clase para manejar la lista circular del polinomio
class ListaCircularPolinomio {
    private Nodo ultimo; // referencia al último nodo

    // Método para agregar un término al polinomio
    public void agregarTermino(double coef, int exp) {
        Nodo nuevo = new Nodo(coef, exp);

        if (ultimo == null) {
            // Primer nodo: apunta a sí mismo
            ultimo = nuevo;
            ultimo.siguiente = ultimo;
        } else {
            // Inserta después del último y actualiza
referencia circular
            nuevo.siguiente = ultimo.siguiente; // el primero
            ultimo.siguiente = nuevo;
            ultimo = nuevo; // el nuevo se convierte en el
último
        }
    }

    // Método para mostrar el polinomio en forma legible
    public void mostrarPolinomio() {
        if (ultimo == null) {
            System.out.println("El polinomio está vacío.");
```

```
            return;
        }

        Nodo actual = ultimo.siguiente; // empieza en el
primero
        System.out.print("P(x) = ");
        do {
            System.out.print(actual.coeficiente + "x^" +
actual.exponente);
            actual = actual.siguiente;
            if (actual != ultimo.siguiente) System.out.print("
+ ");
        } while (actual != ultimo.siguiente);
        System.out.println();
    }

    // Método para evaluar el polinomio en un valor dado de x
    public double evaluar(double x) {
        if (ultimo == null) return 0.0;

        double resultado = 0.0;
        Nodo actual = ultimo.siguiente; // comienza desde el
primero

        do {
            resultado += actual.coeficiente * Math.pow(x,
actual.exponente);
            actual = actual.siguiente;
        } while (actual != ultimo.siguiente);

        return resultado;
    }

    // Muestra una tabla de valores de P(x)
    public void mostrarTablaValores() {
        System.out.println("\nTabla de valores:");
        System.out.println(" x\t|\tP(x)");
        System.out.println("-----------------------");

        for (double x = 0.0; x <= 5.0; x += 0.5) {
            double px = evaluar(x);
            System.out.printf(" %.1f\t|\t%.4f\n", x, px);
        }
    }
```

```
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ListaCircularPolinomio polinomio = new
ListaCircularPolinomio();

        System.out.println("=== Polinomio con Lista Enlazada
Circular ===");
        System.out.print("¿Cuántos términos tendrá el
polinomio?: ");
        int n = sc.nextInt();

        // Entrada de coeficientes y exponentes
        for (int i = 0; i < n; i++) {
            System.out.println("\nTérmino " + (i + 1) + ":");
            System.out.print("Coeficiente: ");
            double coef = sc.nextDouble();
            System.out.print("Exponente: ");
            int exp = sc.nextInt();
            polinomio.agregarTermino(coef, exp);
        }

        // Mostrar el polinomio completo
        System.out.println();
        polinomio.mostrarPolinomio();

        // Mostrar tabla de valores
        polinomio.mostrarTablaValores();

        sc.close();
    }
}
```

✦•···· ···•✦✦•···· ···•✦✦•···· ···•✦✦•···· ···•✦

**Actividad 05**

Lista Doblemente Enlazada de Caracteres □˚. .˚□

**Objetivo:** Desarrollar un programa que construya una lista doblemente enlazada a partir de los caracteres de una cadena ingresada por el usuario, y que luego ordene dicha lista alfabéticamente para mostrarla en pantalla.

```
import java.util.Scanner;
/**
 *
 * @author Alondra Vianney Hernandez Torres
 */


class Nodo {
    char dato;
    Nodo anterior;
    Nodo siguiente;

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

// Clase ListaDoble que maneja la estructura y el ordenamiento
class ListaDoble {
    private Nodo cabeza;
    private Nodo cola;

    // Agrega un carácter al final de la lista
    public void agregarCaracter(char c) {
        Nodo nuevo = new Nodo(c);

        if (cabeza == null) {
            cabeza = cola = nuevo;
        } else {
            cola.siguiente = nuevo;
            nuevo.anterior = cola;
            cola = nuevo;
        }
    }

    // Método para mostrar la lista (de inicio a fin)
    public void mostrarLista() {
        Nodo actual = cabeza;
        while (actual != null) {
            System.out.print(actual.dato + " ");
            actual = actual.siguiente;
```

```
        }
        System.out.println();
    }

    // Método para ordenar alfabéticamente usando burbuja
    public void ordenarLista() {
        if (cabeza == null) return;

        boolean huboIntercambio;
        do {
            huboIntercambio = false;
            Nodo actual = cabeza;

            while (actual.siguiente != null) {
                if (actual.dato > actual.siguiente.dato) {
                    // Intercambio de valores
                    char temp = actual.dato;
                    actual.dato = actual.siguiente.dato;
                    actual.siguiente.dato = temp;
                    huboIntercambio = true;
                }
                actual = actual.siguiente;
            }
        } while (huboIntercambio);
    }
}

// Clase principal con la lógica del programa
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ListaDoble lista = new ListaDoble();

        System.out.println(" Lista Doblemente Enlazada de
Caracteres ");
        System.out.print("Ingresa una cadena de texto: ");
        String cadena = sc.nextLine();

        // 1. Lectura y construcción de la lista
        for (int i = 0; i < cadena.length(); i++) {
            char c = cadena.charAt(i);
            lista.agregarCaracter(c);
        }
```

```
        System.out.println("\nLista original:");
        lista.mostrarLista();

        // 2. Ordenamiento alfabético
        lista.ordenarLista();

        // 3. Mostrar lista ordenada
        System.out.println("\nLista ordenada
alfabéticamente:");
        lista.mostrarLista();

        sc.close();
    }
}
```

✦•···๑·····๑···•✦✦•···๑·····๑···•✦✦•···๑·····๑···•✦✦•···๑·····๑···•✦