

# ¿Cómo escribimos en JAVA una clase que implementa una LISTA?

Una **Lista** o **Secuencia** es una estructura de datos que contiene elementos ubicados en forma secuencial.

Las **Listas** pueden ser ordenadas, desordenadas, pueden contener elementos duplicados, se puede acceder a sus elementos mediante un índice ó posición, etc.

## En JAVA las implementamos:

- Usando arreglos de longitud fija
- Usando nodos enlazados

# Repasamos.....

Una LISTA sabe/puede:

Devolver el elemento ubicado en una posición: **elemento(int pos)**

Conocer si un elemento pertenece a la lista y devolver verdadero ó falso: **incluye(Integer elem)**

Insertar un elemento al comienzo de la lista: **agregarInicio(Integer elem)**

Insertar un elemento al final de la lista: **agregarFinal(Integer elem)**

Insertar un elemento en una posición dada: **agregarEn(Integer elem, int pos)**

Eliminar el elemento de la posición dada: **eliminarEn(int pos)**

Eliminar un elemento dado: **eliminar(Integer elem)**

Conocer si la lista está vacía o no: **esVacía()**

Devolver la cantidad de elementos: **tamaño()**

Inicializar el iterador : **comenzar()**

Devolver un elemento y avanzar en el iterador : **proximo()**

Determinar si terminó de iterar en la lista: **fin()**

BUSCAR

AGREGAR Y  
ELIMINAR  
ELEMENTOS

CONSULTAR

RECORRER LA LISTA

```
public class ListaDeEnterosConArreglos {  
    private int tamaño;  
    private Integer[] datos = new Integer[100];  
    private int actual = 0;
```

Estos datos forman la **IMPLEMENTACIÓN**

DEBEN ser **PRIVADOS: SOLAMENTE SE USAN EN LA CLASE**

```
    public void comenzar() {  
        actual = 0;  
    }  
  
    public Integer proximo() {  
        return datos[actual++];  
    }  
  
    public boolean fin() {  
        return actual == tamaño;  
    }  
  
    public Integer elemento(int pos) {  
        return datos[pos - 1];  
    }  
  
    public boolean agregarEn(Integer elem, int pos) {  
        if (pos < 1 || pos > tamaño + 1 || pos > datos.length  
            || tamaño == datos.length) return false;  
        tamaño++;  
        for (int i = tamaño - 1; i >= pos; i--)  
            datos[i] = datos[i - 1];  
        datos[pos - 1] = elem;  
        return true;  
    } //SIGUEN LOS MÉTODOS
```

Estos métodos forman la **INTERFACE PÚBLICA**

DEBEN ser **PÚBLICOS: TODOS LOS QUE USEN  
ListaDeEnterosConArreglos PUEDEN USAR LOS  
MÉTODOS**

# ¿Cómo usamos la Lista de enteros implementada con arreglos?

ESTOS FRAGMENTOS DE CÓDIGO ESTÁN EN UN MÉTODO DE UNA CLASE CUALQUIERA

```
ListaDeEnterosConArreglos l=new ListaDeEnterosConArreglos();  
  
l.agregarInicio(10);  
l.agregarInicio(20);  
l.agregarFinal(25);  
l.agregarFinal(30);  
l.agregarEn(15,3);
```

## ¿Podemos hacer eso?

```
ListaDeEnterosConArreglos l=new ListaDeEnterosConArreglos();  
  
l.tamano;  
l.datos;  
l.actual;
```

**NO!!!**

Solamente se pueden usar los métodos públicos desde afuera de la clase **ListaDeEnterosConArreglos**.

Los datos privados son **INACCESIBLES**

**OCULTAMIENTO DE LA INFORMACIÓN**

```
public class ListaDeEnterosEnlazada {
```

```
private NodoEntero inicio;  
private NodoEntero actual;  
private NodoEntero fin;  
private int tamanio;
```

```
public void comenzar() {
```

```
    actual = inicio;
```

```
}
```

```
public Integer proximo() {
```

```
    Integer elto = actual.getDato();
```

```
    actual = actual.getSiguiente();
```

```
    return elto;
```

```
}
```

```
public boolean fin() {
```

```
    return (actual == null);
```

```
}
```

```
public Integer elemento(int pos) {
```

```
    if (pos < 1 || pos > this.tamanio())
```

```
        return null;
```

```
    NodoEntero n = this.inicio;
```

```
    while (pos-- > 1)
```

```
        n = n.getSiguiente();
```

```
    return n.getDato();
```

```
}
```

```
//SIGUEN LOS MÉTODOS
```

Estos datos forman la **IMPLEMENTACIÓN**

DEBEN ser **PRIVADOS: SOLAMENTE SE USAN EN LA CLASE**

Estos métodos forman la **INTERFACE PÚBLICA**

DEBEN ser **PÚBLICOS: TODOS LOS QUE USEN  
ListaDeEnterosEnlazada PUEDEN USAR LOS MÉTODOS**

```
public class NodoEntero {
```

```
private Integer dato;
```

```
private NodoEntero siguiente;
```

```
public Integer getDato() {
```

```
    return dato;
```

```
}
```

```
public void setDato(Integer dato) {
```

```
    this.dato = dato;
```

```
}
```

```
public NodoEntero getSiguiente() {
```

```
    return siguiente;
```

```
}
```

```
public void setSiguiente(NodoEntero siguiente) {
```

```
    this.siguiente = siguiente;
```

```
}
```

```
}
```

# ¿Cómo usamos la Lista de enteros implementada con nodos encadenados?

ESTOS FRAGMENTOS DE CÓDIGO ESTÁN EN UN MÉTODO DE UNA CLASE CUALQUIERA

```
ListaDeEnterosEnlazada l=new ListaDeEnterosEnlazada();  
  
l.agregarInicio(10);  
l.agregarInicio(20);  
l.agregarFinal(25);  
l.agregarFinal(30);  
l.agregarEn(15,3);
```

## ¿Podemos hacer eso?

```
ListaDeEnterosEnlazada l=new ListaDeEnterosEnlazada();  
  
l.inicio;  
l.actual;  
l.fin;  
l.tamano;
```

**NO!!!**

**Solamente se pueden usar los métodos públicos desde afuera de la clase ListaDeEnterosEnlazada.**

**Los datos privados son INACCESIBLES**

**OCULTAMIENTO DE LA INFORMACIÓN**

# Usamos las implementaciones de LISTAS

```
package tp03.ejercicio2;
import tp03.ejercicio1.ListaDeEnterosConArreglos;

public class PilaDeEnteros {
    private ListaDeEnterosConArreglos datos = new
        ListaDeEnterosConArreglos();

    public void apilar(int dato) {
        datos.agregarEn(dato, 1);
    }

    public int desapilar() {
        int x = datos.elemento(1);
        datos.eliminarEn(1);
        return x;
    }

    public int tope() {
        return datos.elemento(1);
    }

    public boolean esVacia() {
        return datos.tamano()==0;
    }
}
```

¿Cuál es la IMPLEMENTACIÓN?

¿Cuál es la INTERFACE PÚBLICA?

¿Qué podemos usar de  
PilaDeEnteros desde cualquier  
clase?

**Usamos la PILA:**

**PilaDeEnteros p=new PilaDeEnteros();**

**p.apilar(10),  
p.apilar(15);  
p.apilar(1);**

# Usamos las implementaciones de LISTAS

```
package tp03.ejercicio2;
import tp03.ejercicio1.ListaDeEnterosEnlazada;

public class PilaDeEnteros {
    private ListaDeEnterosEnlazada datos = new
        ListaDeEnterosEnlazada();

    public void apilar(int dato) {
        datos.agregarEn(dato, 1);
    }

    public int desapilar() {
        int x = datos.elemento(1);
        datos.eliminarEn(1);
        return x;
    }

    public int tope() {
        return datos.elemento(1);
    }

    public boolean esVacia() {
        return datos.tamano()==0;
    }
}
```

**Cambie la implementación de la PilaDeEnteros, ¿afecta a las clases que la usan?**

**Usamos la PILA:**

**PilaDeEnteros p=new PilaDeEnteros();**

**p.apilar(10),  
p.apilar(15);  
p.apilar(1);**



# Una clase LISTA genérica que admite diferentes implementaciones

```
public class TestLista {  
  
public void ordenar(ListaDeEnterosConArreglos  
I){  
    int elem;  
    I.comenzar();  
    while(! I.fin()) {  
        elem=I.proximo();  
        //métodos de la ListaDeEnterosConArreglos  
    }  
}  
  
public void ordenar(ListaDeEnterosEnlazada I){  
  
    int elem;  
    I.comenzar();  
    while(! I.fin()) {  
        elem=I.proximo();  
        //métodos de la ListaDeEnterosEnlazada  
    }  
}
```

Los 2 ordenar() son similares.

Las 2 implementaciones tienen una firma similar

Y...si definimos una clase **ListaDeEnteros** que nos permita abstraer diferentes implementaciones y así definir un solo método **ordenar(ListaDeEnteros)** que nos “oculte” la implementación.

```
public void ordenar(ListaDeEnteros I){  
    int elem;  
    I.comenzar();  
    while(! I.fin()) {  
        elem=I.proximo();  
        //métodos de la Lista  
    }  
}
```

# Una clase LISTA genérica que admite diferentes implementaciones

```
public class TestLista {  
    public void ordenar(ListaDeEnteros l){  
        int elem;  
        l.comenzar();  
        while(! l.fin()) {  
            elem=l.proximo();  
            //métodos de la Lista  
        }  
    }  
}
```

Usamos el ordenar(ListaDeEnteros):

```
TestLista t=new TestLista();  
ListaDeEnterosConArreglos l=new  
ListaDeEnterosConArreglos();
```

```
l.agregarInicio(10);  
l.agregarInicio(20);  
l.agregarFinal(25);  
l.agregarFinal(30);  
l.agregarEn(15,3);  
t.ordenar(l);
```

```
TestLista t=new TestLista();  
ListaDeEnterosEnlazada l=new  
ListaDeEnterosEnlazada();
```

```
l.agregarInicio(1);  
l.agregarInicio(2);  
l.agregarFinal(20);  
l.agregarEn(15,3);  
t.ordenar(l);
```

# Definimos una ListaDeEnteros genérica

```
package tp03.ejercicio2;
public abstract class ListaDeEnteros {
    public abstract void comenzar();
    public abstract Integer proximo();
    public abstract boolean fin();

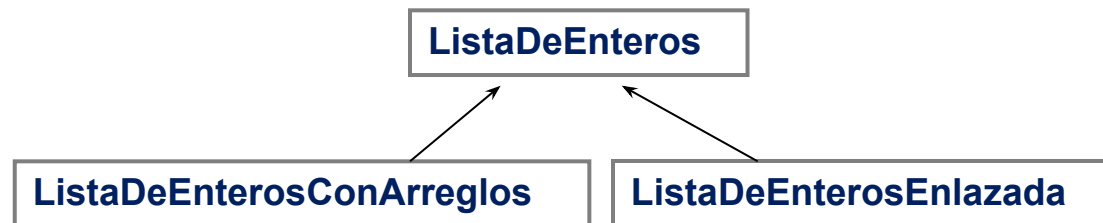
    public abstract Integer elemento(int pos);

    public abstract boolean agregarEn(Integer elem, int pos);
    public abstract boolean agregarInicio(Integer elem);
    public abstract boolean agregarFinal(Integer elem);

    public abstract boolean eliminar(Integer elem);
    public abstract boolean eliminarEn(int pos);

    public abstract boolean incluye(Integer elem);
    public abstract boolean esVacia();
    public abstract int tamano();
}
```

INTERFACE  
PÚBLICA



# La clase ListaDeEnteros

- Es una clase abstracta, no hay implementación, solamente se declaran las firmas de los métodos, es decir declara **qué sabe hacer una lista**.
- En JAVA las clases abstractas se usan para **agrupar el comportamiento común de diferentes implementaciones**.
- En nuestro caso tenemos 2 implementaciones diferentes de Listas de números enteros. La clase abstracta nos sirve para usar cualquier implementación y guardarla en una variable de tipo ListaDeEnteros.

```
ListaDeEnteros lista = new ListaDeEnterosConArreglos();
```

```
lista.agregarInicio(2);  
lista.agregarInicio(4);  
lista.agregarFinal(6);  
lista.agregarEn(15,2);
```

```
TestLista t=new TestLista();  
t.ordenar(lista);
```

```
new ListaDeEnterosEnlazada();
```

# La clase ListaDeEnteros

```
public abstract class ListaDeEnteros {  
    //Código JAVA  
}
```

```
public class ListaDeEnterosConArreglos extends ListaDeEnteros {  
    //Código JAVA  
}
```

```
public class ListaDeEnterosEnlazada extends ListaDeEnteros {  
    //Código JAVA  
}
```

De esta manera en JAVA indicamos que **ListaDeEnteros** es una **superclase** y que **ListaEnlazadaConArreglos** y **ListaDeEnterosEnlazada** son **subclases**.

**Todas las subclases de las clases abstractas DEBEN implementar los métodos definidos en la superclase: YA LO TENEMOS HECHO!!!**

# Y si necesitamos una Lista de nombres o de alumnos o de películas, etc...

Usamos Tipos Genéricos.

En JAVA podemos decir: en esta lista se van a guardar elementos de diferentes tipos de datos (clases), T, y en el momento que la creamos indicamos de qué tipo de datos son sus elementos.

```
package tp03.ejercicio6;

public abstract class ListaGenerica<T> {
    public abstract void comenzar();
    public abstract T proximo();
    public abstract boolean fin();

    public abstract T elemento(int pos);
    public abstract boolean agregarEn( T elem, int pos);
    public abstract boolean agregarInicio( T elem);
    public abstract boolean agregarFinal( T elem);

    public abstract boolean eliminar( T elem);
    public abstract boolean eliminarEn(int pos);

    public abstract boolean incluye( T elem);
    public abstract boolean esVacia();
    public abstract int tamano();
}
```

# Diferentes implementaciones de Listas con Tipos Genéricos

```
package tp03.ejercicio6;

public class ListaEnlazadaGenerica<T> extends ListaGenerica<T> {
    private NodoGenerico<T> inicio;
    private NodoGenerico<T> actual;
    private NodoGenerico<T> fin;
    private int tamaño;

    public void comenzar() {
        actual = inicio;
    }

    public T proximo() {
        T elto = actual.getDato();
        actual = actual.getSiguiente();
        return elto;
    }

    //MÁS MÉTODOS
}
```

```
public class NodoGenerico<T> {
    private T dato;
    private NodoGenerico<T> siguiente;

    public T getDato() {
        return dato;
    }

    public void setDato(T dato) {
        this.dato = dato;
    }

    public NodoGenerico<T> getSiguiente() {
        return siguiente;
    }

    public void setSiguiente(NodoGenerico<T> siguiente) {
        this.siguiente = siguiente;
    }
}
```

```
ListaEnlazadaGenerica<Integer> lista = new ListaEnlazadaGenerica<Integer>();
lista.agregarFinal(new Integer(50));
```

```
ListaEnlazadaGenerica<String> series = new ListaEnlazadaGenerica<String>();
lista.agregarInicio("FLASH");
```

# Diferentes implementaciones de Listas con Tipos Genéricos

```
package tp03.ejercicio6;
public class ListaConArreglosGenerica<T> extends ListaGenerica<T> {
    private int tamaño;
    private T[] datos;
    private int actual = 0;

    public void comenzar() {
        actual = 0;
    }

    public T proximo() {
        return datos[actual++];
    }

    public T elemento(int pos) {
        return datos[pos - 1];
    }
    //MÁS MÉTODOS
}
```

```
ListaConArreglosGenerica <Integer> lista = new ListaConArreglosGenerica<Integer>();
lista.agregarFinal(new Integer(50));
```

```
ListaConArreglosGenerica<String> series = new ListaConArreglosGenerica<String>();
lista.agregarInicio("FLASH");
```



# Diferentes implementaciones de Listas con Tipos Genéricos

```
ListaGenerica<Integer> lista = new ListaEnlazadaGenerica<Integer>();  
lista.agregarFinal(new Integer(50));
```

```
ListaGenerica<String> series = new ListaEnlazadaGenerica<String>();  
lista.agregarInicio("FLASH");
```

```
ListaGenerica<Alumno> series = new ListaDeArreglosGenerica<Alumno>();  
lista.agregarInicio(new Alumno("1234/6", "Perez", "Analia", "Lic en Informática");
```