MA-02: TAD Lista
Colecciones y Listas genéricas

TAD Lista

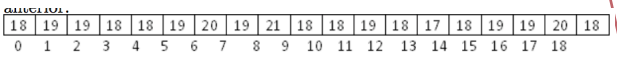
Lista

- Colección de elementos
- Existe un "primero"
- Existe un "último"
- Operaciones:
  - add(e)
  - get(i)
  - set(e,i)
  - contains(e)
  - indexOf(e)
  - remove(e)
  - remove(i)
  - isEmpty()

Listas

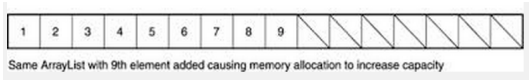
✓ **Implementación Estática:**

✓ Arreglo de elementos

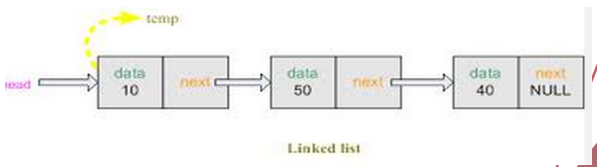


✓ **Implementación Dinámica**

✓ ArrayList de elementos



✓ LinkedList de elementos



Listas Genéricas: Arreglos

Implemente Lista Genérica utilizando arreglos genéricos:

```
public class Lista<T>
    private T []v;
    private int tamaño;
    private int size;
```

- ✔ Lista()
- ✔ Lista(tamaño)
- ✔ toString()
- ✔ add(e)
- ✔ get(i)
- ✔ set(e, i)
- ✔ contains(e)
- ✔ indexOf(e)
- ✔ isEmpty()
- ✔ remove(e)
- ✔ clear()

Clase Lista:

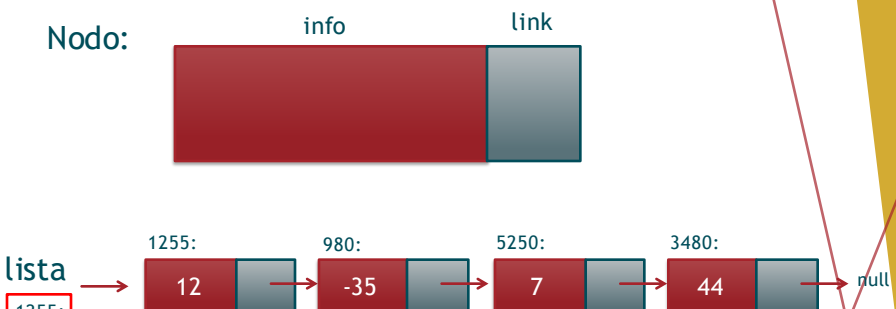
```
public Lista(){...5 lines }
public Lista(int t){...5 lines }
public int size(){...3 lines }
public boolean addLast(T  dato){...8 lines }
public boolean isEmpty(){...3 lines }
public void clear(){...3 lines }
public T get(int i){...6 lines }
public boolean remove(T dato){...19 lines }
public String toString(){...7 lines }
```

Implementación de Listas Dinámicas

Listas Enlazadas

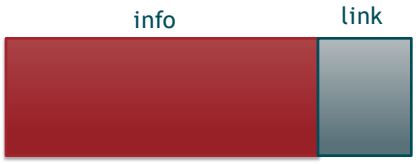
Clase: LinkedList

Listas Enlazadas: LLS

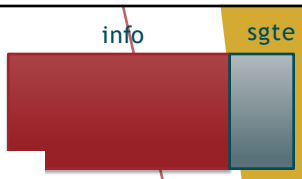


Declarativa Clase Nodo

```
public class Nodo{
    private int info;
    private Nodo link;
```



Clase Nodo

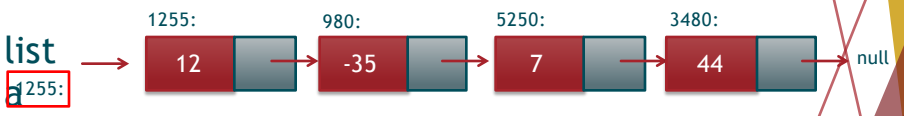



```
public class Nodo {
    private int info;
    private Nodo link;

    public Nodo(){...4 lines }

    public Nodo(int valor){...4 lines }

    public Nodo(Nodo p){...4 lines }

    public int getInfo() {...3 lines }

    public void setInfo(int info) {...3 lines }

    public Nodo getLink() {...3 lines }

    public void setLink(Nodo link) {...3 lines }

    public String toString(){...3 lines }


}
```

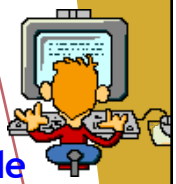
Clase Lista

✔Declarativa **Lista**:

```
public class Lista {
    private Nodo lista; //Puntero al "primer" nodo
```



Clase Lista



Ejercicio: Implementar en la clase: **Lista de enteros:**

- ✔ Lista()
- ✔ toString()
- ✔ add(e)
- ✔ get(i)
- ✔ set(e, i)
- ✔ contains(e)
- ✔ indexOf(e)
- ✔ isEmpty()
- ✔ remove(e)
- ✔ clear()

Listas Genéricas

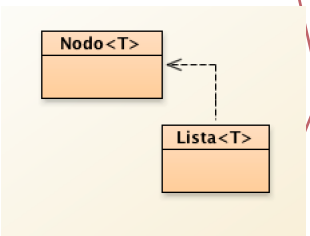
Lista Genérica: Lista<T>:

1. Nodo Genérico

```
public class Nodo<T> {
    private T info;
    private Nodo<T> link;
```

2. Lista Genérica

```
public class Lista <T>{
    private Nodo<T> lista;
```



Hacer una clase Iterable

1. **Clase Iterable:**
   - Clase implementa la "interface Iterable<T>"

```
public class Lista <T> implements Iterable<T>{
```

   - Sobre-escribe método: **iterator()**:

```
public Iterator<T> iterator()
```

2. **Crear clase interna: MiIterador():** Implementa interface: **Iterator<T>**

```
protected class MiIteradorLista implements Iterator<T>{
```

Iteradores

- **MiIterador():** Clase interna de una colección
  - **Atributos:**
    - Para tener la posición actual
    - Eliminar (Opcional)
  - **Métodos:**
    - hasNext()
    - next()
    - remove() (Opcional)

Ejemplos de clase: MiIterador

```
protected class MiIteradorLista implements Iterator<T>{
    private int posicion;
    private boolean eliminar;
    public MiIteradorLista(){...4 lines }
    public boolean hasNext(){...3 lines }
    public T next(){...6 lines }
    public void remove(){...7 lines }
}
```

Comparar objetos

Los elementos de una lista deben ser "comparables"

- La clases implementa la "Interface Comparable<T>"
  - *public int compareTo(T obj)*
- Sobre escribir método equals()
  - *public boolean equals(Object obj)*

Ejemplos de clase: MiIterador

```
protected class MiIteradorLista implements Iterator<T>{
    private Nodo<T> posicion;
    private Nodo<T> anterior;
    private boolean eliminar;
    public MiIteradorLista(){...4 lines }
    @Override
    public boolean hasNext(){...7 lines }
    @Override
    public T next(){...6 lines }
    @Override
    public void remove(){...9 lines }
}
```

MA05: Proyecto de Cátedra

PROYECTO CÁTEDRA MA02
Lista de Cuentas

Descripción y requerimientos funcionales

Generar una aplicación que permita representar listas genéricas para poder utilizarlas con diferentes objetos. Por ejemplo, para crear una nómina de postulantes a un trabajo (lista de personas: nombre, Rut y cargo al que postula)

Implementación