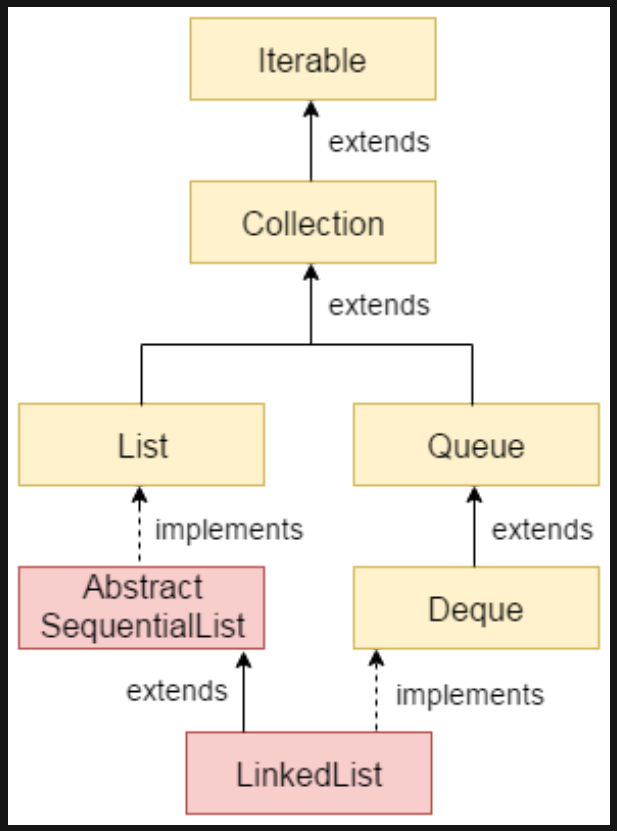
LINKEDLIST

LINKEDLIST

Son listas doblemente enlazadas para almacenar elementos

- Puede contener duplicados
- Lista ordenada
- Mantiene el orden de inserción
- Métodos no sincronizados
- La manipulación es rapida
- Pila, Cola o Lista

JERARQUÍA


Iterable
extends
Collection
extends
List
Queue
implements
extends
Abstract SequentialList
Deque
extends
implements
LinkedList

CONSTRUCTORES

```
LinkedList()

LinkedList(Collection c)
```

MÉTODOS

```
void add(int index, Object element)
void addFirst(Object o)
void addLast(Object o)
int size()
boolean add(Object o)
boolean contains(Object o)
boolean remove(Object o)
Object getFirst()
Object getLast()
int indexOf(Object o)
int lastIndexOf(Object o)
```

EJEMPLO

```
import java.util.*;

public class Test{
  public static void main(String args[]){

    List<String> list=new LinkedList<String>();
    list.add("Ravi");
    list.add("Vijay");
    list.add("Ravi");
    list.add("Ajay");

    Iterator<String> itr=al.iterator();
    while(itr.hasNext()){
      System.out.println(itr.next());
    }
  }
}
```

EJEMPLO

```
import java.util.*;

public class Test{
  public static void main(String args[]){

    LinkedList<String> list=new LinkedList<String>();
    list.add("Ravi");
    list.add("Vijay");
    list.add("Ravi");
    list.add("Ajay");

    Iterator<String> itr=al.iterator();
    while(itr.hasNext()){
      System.out.println(itr.next());
    }
  }
}
```

EJEMPLO

```
import java.util.*;

public class Test{
  public static void main(String args[]){

    Queue<String> queue=new LinkedList<String>();
    queue.add("Ravi");
    queue.add("Vijay");
    queue.add("Ravi");
    queue.add("Ajay");

  }
}
```

EJEMPLO

```
import java.util.*;

public class Test{
  public static void main(String args[]){

    Deque<String> deque=new LinkedList<String>();
    deque.add("Ravi");
    deque.add("Vijay");
    deque.add("Ravi");
    deque.add("Ajay");

  }
}
```

EJERCICIO

LinkedList La Cola de las Tortillas Tarea

INTERFAZ LIST

```
void add(int index,Object element)
boolean addAll(int index,Collection c)
Object get(int index)
Object remove(int index)
ListIterator listIterator()
ListIterator listIterator(int index)
```

EJEMPLO

```
import java.util.*;

public class Test{
  public static void main(String args[]){

    List<String> list=new LinkedList<String>();
    list.add("Ravi");
    list.add("Vijay");
    list.add("Ravi");
    list.add("Ajay");

  }
}
```

INTERFAZ LISTITERATOR

```
boolean hasNext()
Object next()
boolean hasPrevious()
Object previous()
```

EJEMPLO

```
import java.util.*;
public class TestCollection8{
  public static void main(String args[]){
    ArrayList<String> al=new ArrayList<String>();
    al.add("Amit");
    al.add("Vijay");
    al.add("Kumar");

    ListIterator<String> itr=al.listIterator();
    while(itr.hasNext()){
      System.out.println(itr.next());
    }

    while(itr.hasPrevious()){
       System.out.println(itr.previous());
    }

  }
}
```