```
+LinkedList
  -Node
    +data
    +next

  -head
  -tail
  -size

  +iterator
    new LLIterator

  -LLIterator
    -nextNode
    -removeOK
    -posToRemove

      nextNode ← head
      removeOK ← false
      posToRemove ← -1

    +hasNext
      nextNode ≠ null

    +next
      assert hasNext
      result ← nextNode.data
      nextNode ← nextNode.next
      removeOK ← true
      posToRemove++
      result

    +remove
      assert removeOK
      removeOK ← false
      LinkedList.this.remove(posToRemove)
      posToRemove--

  +makeEmpty
    head ← tail ← null
    size ← 0

  +remove(pos)
    assert 0 ≤ pos < size
    Pos=0
      result ← head.data
      head ← head.next
      size = 1 ? tail ← NULL
    Pos ≠ 0
      temp ← head
      1 ≤ i < pos
        temp ← temp.next
      result ← temp.next.data
      temp.next ← temp.next.next
      pos = size - 1 ?
        tail ← temp
    size--
    result

  +get(pos)
    assert 0 ≤ pos < size
    Pos = size - 1 ?
      result ← tail.data
    else
      temp ← head
      1 ≤ i < pos
        temp ← temp.next
      result ← temp.data
    result

  +insert(pos,obj)
    assert 0 ≤ pos < size
    Pos = 0 ?
      addFirst(obj)
    else
      Pos = size ?
        add(obj)
      else
        temp ← head
        1 ≤ i < pos
          temp ← temp.next
        newNode ← new Node(obj, temp.next)
        temp.Next ← newNode
        size++

  +add(obj)
    newNode = new Node(obj, NULL)
    size = 0 ?
      head ← newNode
    else
      Tail.Next ← newNode
    result

  +addFirst(obj)
    assert 0 ≤ pos < size
    size = 0 ?
      add(obj)
    else
      newNode ← new Node(obj, head)
      head ← newNode
      size++

  + toString
    Join data with space from head by next
```