A Linked List Class

Modified version of author's class

The Linked List Node

```
class Node<AnyType>
  {
        public Node( AnyType d, Node<AnyType> p,
Node<AnyType> n )
        {
            data = d; prev = p; next = n;
        }

        public AnyType data;
        public Node<AnyType>   prev;
        public Node<AnyType>   next;
  }
```

```
public class MyLinkedList<AnyType> implements
Iterable<AnyType>
{


    Fields

    Methods



}
```

Fields

```
private int theSize;
public Node<AnyType> beginMarker;
public Node<AnyType> endMarker;
```

Constructor

```
public MyLinkedList( )
{
    clear( );
}
```

```
    public void clear( )
    {
        beginMarker = new Node<AnyType>( null, null, null );
        endMarker = new Node<AnyType>( null, beginMarker, null
);
        beginMarker.next = endMarker;

        theSize = 0;
    }
```

```
public int size( )
{
    return theSize;
}

public boolean isEmpty( )
{
    return size( ) == 0;
}
```

```
    private Node<AnyType> getNode( int idx, int lower, int upper
)
    {
        Node<AnyType> p;

        if( idx < lower || idx > upper )
            throw new IndexOutOfBoundsException( "getNode
index: " + idx + "; size: " + size( ) );
```

```
if( idx < size( ) / 2 )
{
    p = beginMarker.next;
    for( int i = 0; i < idx; i++ )
        p = p.next;
}
else
{
    p = endMarker;
    for( int i = size( ); i > idx; i-- )
        p = p.prev;
}
```

```
    return p;
}
```

```
private Node<AnyType> getNode( int idx )
{
    return getNode( idx, 0, size( ) - 1 );
}
```

```
public AnyType get( int idx )
{
    return getNode( idx ).data;
}
```

```
public AnyType set( int idx, AnyType newVal )
{
    Node<AnyType> p = getNode( idx );
    AnyType oldVal = p.data;

    p.data = newVal;
    return oldVal;
}
```

```
private void addBefore( Node<AnyType> p, AnyType x )
{
    Node<AnyType> newNode = new Node<AnyType>( x, p.
prev, p );
    newNode.prev.next = newNode;
    p.prev = newNode;
    theSize++;
}
```

```
public void add( int idx, AnyType x )
{
    addBefore( getNode( idx, 0, size( ) ), x );
}
```

```
public boolean add( AnyType x )
{
    add( size( ), x );
    return true;
}
```

```
public AnyType remove( Node<AnyType> p )
{
    p.next.prev = p.prev;
    p.prev.next = p.next;
    theSize--;

    return p.data;
}
```

```
public AnyType remove( int idx )
{
    return remove( getNode( idx ) );
}
```

Method that returns an iterator

```
public java.util.Iterator<AnyType> iterator( )
{
    return new LinkedListIterator<AnyType>(this);
}
```

```
class LinkedListIterator<AnyType> implements java.util.
Iterator<AnyType>
    {

 Fields

 Methods
}
```

Fields

```
private Node<AnyType> current;
private boolean okToRemove;
private MyLinkedList<AnyType> theList;
```

Constructor

```
public LinkedListIterator(MyLinkedList<AnyType> lst)
{
    theList = lst;
    current = theList.beginMarker.next;
    okToRemove = false;
}
```

```
public boolean hasNext( )
{
    return current != theList.endMarker;
}
```

```
public AnyType next( )
{
    if( !hasNext( ) )
        throw new java.util.NoSuchElementException( );

    AnyType nextItem = current.data;
    current = current.next;
    okToRemove = true;
    return nextItem;
}
```

```
public void remove( )
{
    if( !okToRemove )
        throw new IllegalStateException( );

    theList.remove( current.prev );
    okToRemove = false;
}
```

Testing the class within main

```
MyLinkedList<Integer> lst = new MyLinkedList<Integer>( );

for( int i = 0; i < 10; i++ )
    lst.add( i );
for( int i = 20; i < 30; i++ )
    lst.add( 0, i );

lst.remove( 0 );
lst.remove( lst.size( ) - 1 );
```

Testing Continued

```
java.util.Iterator<Integer> itr = lst.iterator( );
while( itr.hasNext( ) )
{
    itr.next( );
    itr.remove( );
    System.out.println( lst );
}
```