보고서.

이번엔 Linkedlist 를 직접 구현해 보았다. 링크드 리스트는 자료구조중에 재미있는 것 같다. 찾아가려면 처음노드부터 따라가게 되고, 생각해볼 점이 많은 과제였다.
처음엔 자료구조가 약간 괴리감이 있었는데 자료를 표현하고 그 생각을 코드로 나타내보니까 점점 자료구조가 재밌어진다.

깃허브를 사용해서 테스트를 처음해봤는데 재밌다. 이런걸 CI테스트라고 하나..? 잘모르겠다. 깃을 활용해서 숙제하는게 더 재밌는 것 같다. 앞으로도 git classroom을 더 활용했으면 좋겠다.

```
class Node<T> {
    T data;
    Node<T> next;

    public Node() {
    }

    public Node(T data) { this.data = data; }

    public Node(T data, Node<T> next) {
        this.data = data;
        this.next = next;
    }
}
```

우선 노드 클래스를 봐보자. 노드 클래스는 데이터를 가지고 있고 다음 노드에대한주소를 참조하고 있다.

이점을 활용해서 Linkedlist 를 만들 수 있었다.
Size 메서드는 내가 리스트에 추가하거나 삭제할 때 ++ 해주거나 --를 이용해서 클래스 자체에서 조절해주었다.

isEmpty메서드는 사이즈가 0이면, true를 리턴하게 만들어주었다.

Add 메서드는 인덱스가 존재하는 것이 있고 인덱스가 존재하지 않는 것이있다. 인덱스가 존재하지 않으면 그냥 뒤에다가 넣으면 되므로 나는 add부터 구현했다.

```java
if (index ==0 && size == 0 ) {
    addFirst(element);
    return;
} else if (size >0 && index== 0) {
    Node<E> newNode = new Node<>(element);
    Node<E> preNode = head;
    Node<E> nextNode = preNode.next;
    preNode.next = newNode;
    newNode.next = nextNode;
    size++;
}

 else {
    Node<E> newNode = new Node<>(element);
    Node<E> preNode =  getNode( index: index-1);
    Node<E> nextNode = preNode.next;
    System.out.println("여기까진 실행");
    preNode.next = newNode;
    newNode.next = nextNode;
    size++;
}
```

위와 같이 새로운 노드를 만들면 주소를 참조시켜주어야하므로 위와같이 표현했다.

remove 메서드도 거의 비슷한데,

```java
if (index ==0 ) {
    Node<E> node = head.next;
    if (node.next instanceof Node) {
        head.next = node.next;
    }else {
        head.next = null;
    }
```

삭제할 때 다음 노드가 있을 수도 있고, 다음노드가 없을 수도 있다. 그래서 위와 같이 표현

해보았다. 이게 정확한 linkedlist인지는 모르겠다. 그래도 저렇게 표현해봤더니 동작했다.

```
    return node.data;
}else {

    Node<E> preNode =  getNode( index: index-1);

    Node<E> removeNode = preNode.next;
    if (removeNode.next instanceof Node) {
        Node<E> postNode = removeNode.next;
        preNode.next = postNode;
    }
```

아래도 저렇게 표현한 부분이다.

인덱스를 주지 않고 remove하는 부분은 조금 달랐다.

```java
@Override
public E remove() {
    Node<E> preNode;
    Node<E> tempNode;
    // head 노드가 null인 경우 모든 노드가 삭제되었으므로 return
    if (head == null) {
        return null;
    }
    if (head.next == null) {
        head = null;
        return null;
    } else {
        preNode = head;
        tempNode = head.next;
        while(tempNode.next != null) {

            preNode = tempNode;
            tempNode = tempNode.next;
        }
        preNode.next = null;
        size--;
    }
    return tempNode.data;
}
```

while문으로 끝까지 따라가서 마지막을 삭제했다.

메이븐으로 프로젝트를만들고, 테스팅 도구를 사용해서 테스트를하니까 퍼즐 맞추는 느낌이 들었다. 재밌는거같다.