Evgen959 / Advanced_Backend Public

Code Issues Pull requests Actions Projects Security Insights

Advanced_Backend / Lesen 037 / code / d06_3_2 / src / **MyLikedList.java**

Evgen959 newLes37 078446a · 7 minutes ago

185 lines (159 loc) · 4.07 KB

Code Blame Raw

```java
import java.util.Iterator;

public class MyLikedList<E> implements MyList<E>{

    private Node<E> head = null;
    private Node<E> tail = null;
    private int size = 0;

    @Override
    public boolean add(E element) {
        Node<E> node = new Node<>(tail, null, element);
        size++;
        if (tail!=null){
            tail.setNext(node);
        }
        if (head==null){
            head = node;
        }
        tail = node;
        return true;
    }

    public boolean isEmpty(){
        return  head==null;
    }

    //  0....size-1
    @Override
    public boolean add(int index, E element) {
        if (index>=size){
            return add(element);
        }
        Node<E> node = new Node<>(null,null,element);
        Node<E> next = getNode(index);
        if(next == null||index<=0){ //добавляем ноду в 0 индекс
            next=head;
            head=node;
        }
        Node<E> prev = next.getPrev();
        next.setPrev(node);
        node.setNext(next);
        node.setPrev(prev);
        if (prev!=null){
            prev.setNext(node);
        }
        size++;
        return false;
    }

    @Override
    public E get(int index) {
        Node<E> node = getNode(index);
        return (node!=null)?node.getValue():null;
    }

    private Node<E> getNode(int index){
        if (index>=size || index<0 || head==null){
            return null;
        }
        int counter = 0;
        Node<E> aktiveNode = head;
        while (aktiveNode!=null && counter<index){
            aktiveNode = aktiveNode.getNext();
            counter++;
        }
        return aktiveNode;
    }

    @Override
    public int size() {
        return size;
    }

    private E remove(Node<E> node){// удоляем ноду
        if (node==null){
            return null;
        }
        Node<E> prev = node.getPrev();
        Node<E> next = node.getNext();

        if (prev!=null){
            prev.setNext(next);
        } else {
            head = next;
        }
        if (next!=null){
            next.setPrev(prev);
        } else {
            tail = prev;
        }
```

```java
            size--;
            node.setPrev(null);
            node.setNext(null);
            E removedValue = node.getValue();
            return removedValue;
        }

        @Override
        public E remove(int index) {
            Node<E> node = getNode(index); // ищит ноду
            return remove(node);
        }

        @Override
        public E remove() {
            return remove(tail);
        }

        @Override
        public E set(int index, E element) {
            return null;
        }

        @Override
        public String toString() {
            if (head==null){
                return "[]";
            }
            StringBuilder sb = new StringBuilder();
            Node<E> currenNode = head;
            while (currenNode!=null){
                sb.append(currenNode.getValue()).append(";");
                currenNode=currenNode.getNext();
            }
            sb.setLength(sb.length()-1);
            return "[" + sb.toString() + ']';
        }

        @Override
        public Iterator<E> iterator(){
            return this.new MyListIteraror();
        }

        private class MyListIteraror implements Iterator<E>{
            private Node<E> current = head;

            @Override
            public boolean hasNext() {
                return current!=null;
            }

            @Override
            public E next() {
                E value = current.value;
                current=current.next;
                return value;
            }
        }

        public class Node<E> {
            Node<E> prev;
            Node<E> next;
            E value;

            public Node(Node<E> prev, Node next, E value) {
                this.prev = prev;
                this.next = next;
                this.value = value;
            }

            public Node<E> getPrev() {
                return prev;
            }

            public void setPrev(Node<E> prev) {
                this.prev = prev;
            }

            public Node<E> getNext() {
                return next;
            }

            public void setNext(Node<E> next) {
                this.next = next;
            }

            public E getValue() {
                return value;
            }

            public void setValue(E value) {
                this.value = value;
            }
        }
    }
```