Plan

2023-11-08

1. Theory of LinkedList

2. Custom LinkedList implementation

1. Теория LinkedList
2. Собственная реализация LinkedList

Theory

`LinkedList`

`LinkedList` is another implementation of the `List` interface. It allows storing any data, including `null`. The key feature of this collection's implementation is that it is based on a doubly-linked list (each element has references to the previous and next elements). Because of this, adding and removing from the middle, accessing by index or value, takes linear time O(n), while adding and removing from the beginning and end takes constant time O(1).

`LinkedList` — ещё одна реализация `List`. Позволяет хранить любые данные, включая `null`. Особенностью реализации данной коллекции является то, что в её основе лежит двунаправленный связный список (каждый элемент имеет ссылку на предыдущий и следующий). Благодаря этому, добавление и удаление из середины, доступ по индексу, значению происходит за линейное время O(n), а из начала и конца за константное O(1).

Learn more (Подробнее): [Структуры данных в картинках. `LinkedList`]

Homework

▶ English

▼ На русском

- дописать методы MyLinkedList

Code

src/linked_list/Main.java

```
package linked_list;

public class Main {

    public static void main(String[] args) {

        MyList<String> list = new MyLinkedList<>();

        System.out.println("Лист пустой? - " + list.isEmpty());
        System.out.println("Размер листа - " + list.size());
        System.out.println(list);

        list.add("AAA");
        list.add("BBB");
        list.add("CCC");

        System.out.println("Лист пустой? - " + list.isEmpty());
        System.out.println("Размер листа - " + list.size());
        System.out.println(list);

        list.add("DDD");
        list.add("EEE");

        System.out.println(list);

        System.out.println(list.get(3));

        list.set(3, "FFF");

        System.out.println(list);

        System.out.println("Есть ли в листе элемент EEE? - " + list.contain
        System.out.println("Есть ли в листе элемент GGG? - " + list.contain

        String deletedElement = list.remove(0);
        System.out.println("Удалённое значение - " + deletedElement);
        System.out.println(list);

        System.out.println("Размер листа - " + list.size());
    }
}
```

src/linked_list/MyLinkedList.java

```java
package linked_list;

import java.util.Objects;

public class MyLinkedList<T> implements MyList<T> {

    private int size;
    private Node<T> first;

    @Override
    public void add(T element) {
        Node<T> newNode = new Node<>(element);
        if (isEmpty()) {
            first = newNode;
        } else {
            Node<T> current = first;
            while (current.getNext() != null) {
                current = current.getNext();
            }
            current.setNext(newNode);
        }
        size++;
    }

    @Override
    public T get(int index) {
        if (index < 0 || index >= size) {
            System.out.println("Некорректный индекс!");
            return null;
        }
        if (isEmpty()) {
            return null;
        }
        Node<T> current = first;
        for (int i = 0; i < index; i++) {
            current = current.getNext();
        }
        return current.getValue();
    }

    @Override
```

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

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

    @Override
    public void set(int index, T element) {
        if (index < 0 || index >= size) {
            System.out.println("Некорректный индекс!");
            return;
        }

        Node<T> current = first;
        for (int i = 0; i < index; i++) {
            current = current.getNext();
        }

        current.setValue(element);
    }

    @Override
    public boolean contains(T element) {
        if (isEmpty()) {
            return false;
        }

        Node<T> current = first;
        while (current != null) {
            if (Objects.equals(element, current.getValue())) {
                return true;
            }
            current = current.getNext();
        }
        return false;
    }

    @Override
    public T remove(int index) {
        if (index < 0 || index >= size) {
```

```
            System.out.println("Некорректный индекс!");
            return null;
        }

        size--;

        if (index == 0) {
            T deletedValue = first.getValue();
            first = first.getNext();
            return deletedValue;
        }

        Node<T> current = first;
        for (int i = 0; i < index - 1; i++) {
            current = current.getNext();
        }
        T deletedValue = current.getNext().getValue();
        current.setNext(current.getNext().getNext());
        return deletedValue;
    }

    @Override
    public String toString() {
        if (isEmpty()) {
            return "[]";
        }
        StringBuilder builder = new StringBuilder("[");
        Node<T> current = first;
        while (current != null) {
            builder.append(current.getValue()).append(", ");
            current = current.getNext();
        }
        builder.setLength(builder.length() - 2);
        builder.append("]");
        return builder.toString();
    }
}
```

src/linked_list/MyList.java

```
package linked_list;

public interface MyList<T> {
```

```
        // Добавляем новый элемент в лист
        void add(T element);

        // Получаем элемент из листа по его индексу
        T get(int index);

        // Получаем размер листа, то есть количество элементов
        int size();

        // Позволяет узнать, пустой наш лист или нет
        boolean isEmpty();

        // Заменяем старое значение на новое по указанному индексу
        void set(int index, T element);

        // Проверяем, содержится ли в листе указанный элемент
        boolean contains(T element);

        // Удаляем элемент из листа по указанному индексу
        T remove(int index);
    }
```

src/linked_list/Node.java

```
    package linked_list;

    public class Node<T> {

        private T value;
        private Node<T> next;

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

        public T getValue() {
            return value;
        }

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

        public Node<T> getNext() {
```

```
            return next;
        }

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