Plan

2023-11-08

1. Theory of data structures

2. Custom ArrayList implementation

1. Теория структур данных
2. Собственная реализация ArrayList

Theory

▶ **English**

▼ **На русском**

- Java Collection Framework (JCF) - множество классов и интерфейсов которые реализуют наиболее часто используемые структуры данных. JCF состоит из двух больших подразделов: Map и Collection. Мы начинаем наше изучение с коллекций.
- Интерфейс Collection расширяет интерфейс Iterable, т. е. все коллекции итерируемые. Интерфейс Collection определяет некоторый основной набор методов для работы с коллекциями данных. Например добавление, удаление, поиск, получение колличества элементов в коллекции и т. д.
- Есть множество интерфейсов расширяющих интерфейс Collection. Мы рассмотрим интерфейсы Set и List. И начнем с интерфейса List. Интерфейс List определяет коллекции элементы которых имеют индексы, т. е. некий аналог массива, но не имеющий ограничения по размеру. Соответственно в интерфейсе List, помимо методов унаследованных от Iterable и Collection, определены методы работающие с индексами. Например вставка по индексу, удаление по индексу, получение элемента по индексу, поиск индекса заданного аргумента и т. п.
- Одной из имплементаций интерфейса List является класс ArrayList. Для реализации функциональности интерфейса List,
- ArrayList инкапсулирует в себе массив некоторого начального размера. Когда этот массив полностью заполняется, то его элементы копируются в новый массив, но уже большего размера. И теперь ждем когда заполнится новый массив. И т. д. Т. е. простым языком ArrayList представляет из себя "резиновый массив".

Реализация собственной версии ArrayList

Для практического занятия мы можем взять за основу простую структуру данных, аналогичную ArrayList в Java, которую назовем MyArrayList. Вот базовый контур класса для

реализации:

```
public interface MyList<E> {

    // Добавляем элемент и увеличиваем размер массива, если нужно
    void add(E o);

    // Получаем элемент по индексу
    E get (int index);

    // устанавливает объект по индексу, смещая объекты
    void set(E o, int index);

    // Возвращаем размер коллекции
    int size();
    boolean contains(E o);

    // Удаляем элемент по значению
    boolean remove(E o);

    // Удаляем элемент по индексу
    E removeByIndex(int index);
}
```

- Когда элементы добавляются в ArrayList и его текущая емкость заполняется, ArrayList должен увеличить свой размер, чтобы вместить больше элементов. Это происходит за счет создания нового массива большего размера и копирования элементов из старого массива в новый.
- Процесс увеличения размера называется "расширением" (resizing) или "перераспределением" (reallocating), и хотя он относительно эффективен, он может быть дорогостоящим с точки зрения производительности при добавлении большого количества элементов, так как при каждом расширении происходит копирование всех элементов. Поэтому рекомендуется, если известно количество элементов или примерный верхний предел, инициализировать ArrayList с этой начальной емкостью:

```
List<String> list = new ArrayList<>(начальная_емкость);
```

Это позволит избежать лишних расширений и увеличить производительность при добавлении большого количества элементов.

Код с урока в github:

- https://github.com/AR1988/Ait-342ArrayList
- https://github.com/AR1988/Game21_AIT

Homework

▶ English

▼ На русском

- дописать методы MyArrayList

Code

src/array_list/Main.java

```
package array_list;

public class Main {

    public static void main(String[] args) {

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

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

        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("Элемент листа по индексу 2 - " + list.get(2));

        list.add("FFF");
        list.add("GGG");
        list.add("HHH");
```

```
        list.add("III");
        list.add("JJJ");

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

        System.out.println(list);

        list.add("KKK");

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

        System.out.println(list);

        list.set(3, "LLL");

        System.out.println(list);

        System.out.println("Есть ли в листе FFF? - " + list.contains("FFF")
        System.out.println("Есть ли в листе MMM? - " + list.contains("MMM")
        System.out.println("Есть ли в листе null? - " + list.contains(null)

        list.set(4, null);

        System.out.println(list);

        System.out.println("Есть ли в листе FFF? - " + list.contains("FFF")
        System.out.println("Есть ли в листе MMM? - " + list.contains("MMM")
        System.out.println("Есть ли в листе null? - " + list.contains(null)

        System.out.println("Размер листа - " + list.size());
        String deletedElement = list.remove(2);

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

src/array_list/MyArrayList.java

```
package array_list;
```

```
import java.util.Objects;

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

    private Object[] elements;
    private int size;

    public MyArrayList() {
        elements = new Object[10];
    }

    @Override
    public void add(T element) {
        if (size == elements.length) {
            growSize();
        }
        elements[size++] = element;
    }

    private void growSize() {
        Object[] newElements = new Object[elements.length * 2];
        System.arraycopy(elements, 0, newElements, 0, elements.length);
        elements = newElements;
    }

    @Override
    public T get(int index) {
        if (index < 0 || index >= size) {
//            throw new IllegalArgumentException("Неверный индекс!");
            System.out.println("Неверный индекс!");
            return null;
        }
        return (T) elements[index];
    }

    @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) {
//            throw new IllegalArgumentException("Неверный индекс!");
            System.out.println("Неверный индекс!");
            return;
        }
        elements[index] = element;
    }

    @Override
    public boolean contains(T element) {
        for (int i = 0; i < size; i++) {
            if (Objects.equals(element, elements[i])) {
                return true;
            }
        }
        return false;
    }

    @Override
    public T remove(int index) {
        if (index < 0 || index >= size) {
//            throw new IllegalArgumentException("Неверный индекс!");
            System.out.println("Неверный индекс!");
            return null;
        }

        Object deletedElement = elements[index];

        System.arraycopy(elements, index + 1, elements, index, --size - ind

        return (T) deletedElement;
    }

    @Override
    public String toString() {
        if (isEmpty()) {
            return "[]";
        }
        StringBuilder builder = new StringBuilder("[");
```

```
            for (int i = 0; i < size; i++) {
                builder.append(elements[i]).append(", ");
            }
            builder.setLength(builder.length() - 2);
            builder.append("]");
            return builder.toString();
        }
    }
```

src/array_list/MyList.java

```
package array_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);
}
```