Plan

2023-11-16

1. Стек как структура данных.
2. Очередь как структура данных.
3. Собственная реализация стека.
4. Собственная реализация очереди.

Theory

https://raw.githubusercontent.com/ait-tr/cohort34.2/main/basic*programming/lesson*49/resources/44.Data*structures*Queue,*Stack*.ArrayDeque.pptm

Homework

▶ **English**

▼ **На русском**

- Реализовать до конца методы очереди

Code

src/interfaces/MyCollection.java

```
package interfaces;

public interface MyCollection<T> {

    int size();

    boolean isEmpty();

    void push(T element);

    T pop();

    T peek();
```

```
    int search(T element);
}
```

src/queue/MyQueue.java

```
package queue;

import interfaces.MyCollection;

public class MyQueue<T> implements MyCollection<T> {

    private Object[] elements = new Object[7];
    private int size;
    private int head;
    private int tail;

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

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

    @Override
    public void push(T element) {

        if (tail == elements.length - 1) {
            System.out.println("Очередь заполнена");
            return;
        }

        if (element == null) {
            System.out.println("Добавляемый элемент не может быть null");
            return;
        }

        if (!isEmpty()) {
            tail++;
        }
        elements[tail] = element;
        size++;
    }

    @Override
```

```
    public T pop() {

        if (isEmpty()) {
            System.out.println("Очередь пуста");
            return null;
        }

        T element = (T) elements[head];
        elements[head++] = null;
        size--;

        if (isEmpty()) {
            head = 0;
            tail = 0;
        }
        return element;
    }

    @Override
    public T peek() {

        if (isEmpty()) {
            System.out.println("Очередь пуста");
            return null;
        }

        return (T) elements[head];
    }

    @Override
    public int search(T element) {

        if (isEmpty()) {
            System.out.println("Очередь пуста");
            return -1;
        }

        for (int i = head; i <= tail; i++) {
            if (element.equals(elements[i])) {
                return i - head;
            }
        }

//        int counter = 0;
//        for (int i = head; counter < size; i++) {
//            if (element.equals(elements[i])) {
```

```
//                return i - head;
//            }
//            counter++;
//        }

        return -1;
    }

    @Override
    public String toString() {
        if (isEmpty()) {
            return "[]";
        }

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

src/queue/Main.java

```
package queue;

import interfaces.MyCollection;

public class Main {

    public static void main(String[] args) {

        MyCollection<String> queue = new MyQueue<>();

        System.out.println("Пустая ли очередь? - " + queue.isEmpty());
        System.out.println("Размер очереди - " + queue.size());
        System.out.println(queue);

        queue.push("AAA");
        queue.push("BBB");
        queue.push("CCC");
        queue.push("DDD");

        System.out.println("Пустая ли очередь? - " + queue.isEmpty());
        System.out.println("Размер очереди - " + queue.size());
```

```
            System.out.println(queue);

            String removedElement = queue.pop();
            System.out.println("Удалённый элемент - " + removedElement);
            System.out.println("Размер очереди - " + queue.size());
            System.out.println(queue);

            removedElement = queue.pop();
            System.out.println("Удалённый элемент - " + removedElement);
            System.out.println("Размер очереди - " + queue.size());
            System.out.println(queue);

            queue.push("EEE");
            queue.push("FFF");

            System.out.println(queue);

            String receivedElement = queue.peek();
            System.out.println("Полученный элемент - " + receivedElement);
            System.out.println("Размер очереди - " + queue.size());
            System.out.println(queue);

            System.out.println("Индекс элемента CCC - " + queue.search("CCC"));
            System.out.println("Индекс элемента EEE - " + queue.search("EEE"));
            System.out.println("Индекс элемента FFF - " + queue.search("FFF"));
            System.out.println("Индекс элемента HHH - " + queue.search("HHH"));

            queue.pop();
            queue.pop();

            System.out.println(queue);
            System.out.println("Индекс элемента EEE - " + queue.search("EEE"));
            System.out.println("Индекс элемента FFF - " + queue.search("FFF"));
        }
    }
```

src/stack/MyStack.java

```
    package stack;

    import interfaces.MyCollection;

    public class MyStack<T> implements MyCollection<T> {

        private Object[] elements = new Object[7];
        private int size;
```

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

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

    @Override
    public void push(T element) {

        if (size == elements.length) {
            System.out.println("Стек переполнен");
            return;
        }

        if (element == null) {
            System.out.println("Добавляемый элемент не может быть null");
            return;
        }

        elements[size++] = element;
    }

    @Override
    public T pop() {

        if (isEmpty()) {
            System.out.println("Стек пуст");
            return null;
        }

        size--;
        T element = (T) elements[size];
        elements[size] = null;
        return element;
    }

    @Override
    public T peek() {

        if (isEmpty()) {
            System.out.println("Стек пуст");
```

```
            return null;
        }

        return (T) elements[size - 1];
    }

    @Override
    public int search(T element) {
        for (int i = 0; i < size; i++) {
            if (elements[i].equals(element)) {
                // По формуле size - i мы вычисляем
                // порядковый номер элемента
                // относительно вершины стека
                return size - i;
            }
        }
        return -1;
    }

    @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/stack/Main.java

```
package stack;

import interfaces.MyCollection;

public class Main {

    public static void main(String[] args) {

//        Stack<String> stack = new Stack<>();
//
```

```
//        for (String current : stack) {
//
//        }

        MyCollection<String> stack = new MyStack<>();

        System.out.println("Пустой ли стек? - " + stack.isEmpty());
        System.out.println("Размер стека - " + stack.size());
        System.out.println(stack);

        stack.push("AAA");
        stack.push("BBB");
        stack.push("CCC");

        System.out.println("Пустой ли стек? - " + stack.isEmpty());
        System.out.println("Размер стека - " + stack.size());
        System.out.println(stack);

        String removedElement = stack.pop();
        System.out.println("Удалённый элемент - " + removedElement);
        System.out.println("Размер стека - " + stack.size());
        System.out.println(stack);

        String receivedElement = stack.peek();
        System.out.println("Полученный элемент - " + receivedElement);
        System.out.println("Размер стека - " + stack.size());
        System.out.println(stack);

        stack.push("CCC");
        stack.push("DDD");
        stack.push("EEE");

        System.out.println(stack);

        System.out.println("Порядковый номер элемента CCC - " + stack.search("CCC'
        System.out.println("Порядковый номер элемента EEE - " + stack.search("EEE'
        System.out.println("Порядковый номер элемента BBB - " + stack.search("BBB'
        System.out.println("Порядковый номер элемента FFF - " + stack.search("FFF'
    }
}
```