

# Практическая работа. “Операторы, типы данных и структуры данных в Java”

Продолжая работать над проектом коллекции задач необходимо описать класс списка задач, который будет хранить и работать со списком объектов, описанных в предыдущей практической работе(предполагается, что вы скопируете файлы предыдущей практики в текущую, изменив номер практики).

## Информация о структуре ArrayList

**ArrayList, динамический массив** - это массив, размер которого может меняться во время исполнения программы. Динамические массивы дают возможность более гибкой работы с данными, так как позволяют не прогнозировать хранимые объёмы данных, а регулировать размер массива в соответствии с реально необходимыми объёмами.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	null	null	null	null	null	null

## Обязательное задание

Описать абстрактный класс `com.nc.edu.ta.yourname.prN.AbstractTaskList`, который должен содержать следующие методы:

- **abstract void** `add(Task task)` – метод для добавления не уникальных задач. Возможность добавления пустых задач должна быть запрещена реализацией.
- **abstract void** `remove(Task task)` – метод для удаления всех задач равных входной. Возможность удаления пустой задачи должна быть запрещена реализацией.
- **abstract int** `size()` — количество задач в текущем списке.
- **abstract Task** `getTask(int index)` — получить задачу под заданным номером (с нуля). Должна быть реализована проверка корректности входного значения – допустимыми значениями является не отрицательные числа не превышающие `size()`.

Описать класс `com.nc.edu.ta.yourname.prN.ArrayTaskList`, унаследовав его от `AbstractTaskList`, который будет хранить список задач в массиве. Список реализовать своими силами, без использования классов, реализующих `java.util.List`.

Размер массива задаётся с запасом. Когда массив полностью заполняется, его размер нужно увеличить (создать массив большего размера и скопировать в него задачи из предыдущего). Для этого объявить переменную, являющаяся константой для каждого из списков задач и указывающая количество элементов, на которое следует расширить массив.

Список задач должен содержать следующие методы (определенные абстрактными в родительском классе):

- Методы добавления/удаления задачи:
  - `void add(Task task)`
  - `void remove(Task task)`
- `Task getTask(int index)` — получить задачу под заданным номером (с нуля).

Объявить переменную, в которой будет храниться актуальное количество созданных списков задач (значение должно быть одинаковым, независимо от списка задач, в котором вызывается). Она будет использоваться конструктором класса `ArrayTaskList`.

Объявить текстовую константу (значение должно быть одинаковым, независимо от списка задач, в котором вызывается). Заголовок каждой задачи из списка должен начинаться со значения, которое хранит в себе эта переменная. Значение этой переменной – `[EDUCTR][TA]`.

Изменить в классе `AbstractTaskList` метод `size()` убрав абстрактность и добавив реализацию, так как его реализация не зависит от способа хранения данных в списке.

- `int size()` — количество задач в текущем списке.

Обратите внимание, что в классе `System` есть метод копирования массивов [System.arraycopy](#). Одна из основных задач практики – разобраться с массивами и циклами. Реализуйте копирование массивов самостоятельно, без применения стандартного метода.

## Тестирование

Необходимо добавить в проект тесты, которые находятся в архиве `unit.zip` (будут доступны на занятии), а так же использовать тесты из предыдущей практики.

## Дополнительное задание

- `Task[] incoming(int from, int to)` — массив задач из списка, время оповещения которых находится между *from* (исключительно) и *to* (включительно).
- Размер массива не должен быть *значительно* больше количества хранимых элементов. Например, если мы храним 5 событий, то массив из 100 элементов будет явно избыточен. Сделайте так, чтобы размер массива уменьшался при удалении элементов.