Array vs ArrayList in Java

№	Параметр для сравнения	Array	ArrayList
1	**Implementation**	Native-компонент языка	Класс Java Collections framework. Реализован на основе массива.
2	**Performance**	Доступ по индексу: O(1)	Методы: get(), set() – O(1) add(), remove() – O(n) contains(), indexOf() – O(n)
3	**Type Safety**	Указание типа при объявлении массива.	Использование Generics
4	**Flexibility**	Менее гибкая структура.	Динамическое изменение размера, различные методы для работы с данными.
5	**Primitives**	Поддержка примитивных типов	Хранит только объекты (для примитивов – классы-обертки)
6	**Generics**	-	Поддерживаются
7	**Iteration**	Loops, For-Each	Loops, For-Each Iterator (возможна динамическая модификация)
8	**Supported Operations**	Хранение элементов, доступ по индексу	Тоже что и для массивов + удаление элементов, доступ по индексу и элементу, очистка, удаление всех элементов и т.д.
9	**Size() vs length**	Length	Size()
10	**Null values**	Массивы объектов – да Массивы примитивов – нет (default value)	Да
11	**Read only**	нет (вариант обхода в простых случаях: клонирование массива)	да (с использованием read-only-декоратора Collections.unmodifiableList(List))
12	**Transform one to another**	массив из списка: List.toArray() всегда возвращает новый массив объектов типа Object; List.toArray(T[]) позволяет указать приготовленный буфер для не-примитивов; преобразование в массив примитивов не поддерживается напрямую	список из массива: Arrays.asList(T...) без проблем, если массив является массивом объектов, а не примитивных типов; если массив является массивом примитивов, Arrays.asList() возвращает список с одного элемента, считая входной массив единственным объектом, хотя существуют варианты типа Ints.asList(int[]) из Guava
13	**.toString()**	нет (всегда имеет вид "[T@IDENTITY", где T - тип массива, IDENTITY - уникальный ID объекта; требуется использование Arrays.toString())	[n1, n2, n3...]