Массивы

**Массив** — это структура данных, в которой хранятся элементы одного типа. Его можно представить, как набор пронумерованных ячеек, в каждую из которых можно поместить какие-то данные (один элемент данных в одну ячейку).

**Объявление массива:**

dataType[] arrayName; - Желательно объявлять массив именно таким способом, это Java-стиль.

1. **dataType arrayName[]; -** Унаследованный от С/С++ способ объявления массивов, который работает и в Java.

**Создание массива:**

**new typeOfArray [length]; -** Где typeOfArray — это тип массива, а length — его длина (то есть, количество ячеек), выраженная в целых числах (int). Однако здесь мы только выделили память под массив, но не связали созданный массив ни с какой объявленной ранее переменной.

**Длина массива в Java**

**длина массива** — это количество элементов, под которое рассчитан массив. Длину массива нельзя изменить после его создания.

получить доступ к длине массива можно с помощью переменной length/

Arrays

**Arrays –** это класс который содержит различные методы для манипулирования массивами (такие как сортировка и поиск). Этот класс также содержит статическую фабрику, которая позволяет просматривать массивы в виде списков.

**Методы:**

**asList(T... a)** - возвращает список фиксированного размера, поддерживаемый указанным массивом.

**binarySearch(T[] a, int fromIndex, int toIndex, T key, Comparator<? super T> c)** - выполняет поиск в диапазоне указанного массива для указанного объекта с использованием алгоритма двоичного поиска.

**binarySearch(T[] a, T key, Comparator<? super T> c)** - выполняет поиск в указанном массиве указанного объекта с использованием алгоритма двоичного поиска.

**compare(T[] a, int aFromIndex, int aToIndex, T[] b, int bFromIndex, int bToIndex)** - сравнивает два Object массива лексикографически по указанным диапазонам.

**compare(T[] a, int aFromIndex, int aToIndex, T[] b, int bFromIndex, int bToIndex, Comparator<? super T> cmp)** - сравнивает два Object массива лексикографически по указанным диапазонам.

**compare(T[] a, T[] b)** - сравнивает два Object массивы, внутри сопоставимых элементов, лексикографически.

**compare(T[] a, T[] b, Comparator<? super T> cmp)** - сравнивает два Object массивы лексикографически с использованием указанного компаратора.

**compareUnsigned(short[] a, int aFromIndex, int aToIndex, short[] b, int bFromIndex, int bToIndex)** - сравнивает два short массива лексикографически по указанным диапазонам, численно обрабатывая элементы как беззнаковые.

**compareUnsigned(short[] a, short[] b)** - сравнивает два short массивы лексикографически, численно обрабатывающие элементы как беззнаковые.

**copyOf(T[] original, int newLength)** - копирует указанный массив, усекая или дополняя нулями (при необходимости), чтобы копия имела указанную длину.

**copyOf(U[] original, int newLength, Class<? extends T[]> newType)** - копирует указанный массив, усекая или дополняя нулями (при необходимости), чтобы копия имела указанную длину.

**copyOfRange(T[] original, int from, int to)** - копирует указанный диапазон указанного массива в новый массив.

**copyOfRange(U[] original, int from, int to, Class<? extends T[]> newType)** - копирует указанный диапазон указанного массива в новый массив.

**deepEquals(Object[] a1, Object[] a2)** - возвращает true если два указанных массива являются глубоко равные друг к другу.

**deepHashCode(Object[] a)** - возвращает хэш-код, основанный на "глубинном содержимом" указанного массива.

**deepToString(Object[] a)** - возвращает строковое представление "глубокого содержимого" указанного массива.

**equals(T[] a, int aFromIndex, int aToIndex, T[] b, int bFromIndex, int bToIndex, Comparator<? super T> cmp)** - возвращает true, если два указанных массива объектов в указанных диапазонах являются равны друг к другу.

**equals(T[] a, T[] a2, Comparator<? super T> cmp)** - возвращает, true если два указанных массива объектов равны друг другу.

**fill(Object[] a, int fromIndex, int toIndex, Object val)** - присваивает указанную ссылку на объект каждому элементу указанного диапазона указанного массива объектов.

**fill(Object[] a, Object val)** - присваивает указанную ссылку на объект каждому элементу указанного массива объектов.

**hashCode(Object[] a)** - возвращает хэш-код, основанный на содержимом указанного массива.

**mismatch(T[] a, int aFromIndex, int aToIndex, T[] b, int bFromIndex, int bToIndex, Comparator<? super T> cmp)** - находит и возвращает относительный индекс первого несоответствия между двумя Object массивами в указанных диапазонах, в противном случае возвращает -1, если несоответствие не обнаружено.

**mismatch(T[] a, T[] b, Comparator<? super T> cmp)** - находит и возвращает индекс первого несоответствия между двумя Object массивы, в противном случае возвращают -1, если несоответствие не найдено.

**parallelPrefix(T[] array, int fromIndex, int toIndex, BinaryOperator<T> op)** - выполняет parallelPrefix(Object[], BinaryOperator) для данного поддиапазона массива.

**parallelPrefix(T[] array, BinaryOperator<T> op)** - накапливает, параллельно, каждый элемент данного массива на месте, используя предоставленную функцию.

**parallelSetAll(long[] array, IntToLongFunction generator)** - установите все элементы указанного массива параллельно, используя предоставленную функцию генератора для вычисления каждого элемента.

**parallelSetAll(T[] array, IntFunction<? extends T> generator)** - установите все элементы указанного массива параллельно, используя предоставленную функцию генератора для вычисления каждого элемента.

**parallelSort(T[] a)** - сортирует указанный массив объектов в порядке возрастания в соответствии с естественный порядок из его элементов.

**parallelSort(T[] a, int fromIndex, int toIndex)** - сортирует указанный диапазон указанного массива объектов в порядке возрастания в соответствии с естественный порядок из его элементов.

**parallelSort(T[] a, int fromIndex, int toIndex, Comparator<? super T> cmp)** - сортирует указанный диапазон указанного массива объектов в соответствии с порядком, заданным указанным средством сравнения.

**parallelSort(T[] a, Comparator<? super T> cmp**) - сортирует указанный массив объектов в соответствии с порядком, заданным указанным компаратором.

**setAll(T[] array, IntFunction<? extends T> generator)** - установите все элементы указанного массива, используя предоставленную функцию генератора для вычисления каждого элемента.

**sort(T[] a, int fromIndex, int toIndex, Comparator<? super T> c)** - сортирует указанный диапазон указанного массива объектов в соответствии с порядком, заданным указанным средством сравнения.

**sort(T[] a, Comparator<? super T> c)** - сортирует указанный массив объектов в соответствии с порядком, заданным указанным компаратором.

**spliterator(T[] array)** - возвращает Spliterator, охватывающий весь указанный массив.

**spliterator(T[] array, int startInclusive, int endExclusive)** - возвращает значение, Spliterator охватывающее указанный диапазон указанного массива.

**stream(T[] array)** - возвращает последовательность Stream с указанным массивом в качестве источника.

**stream(T[] array, int startInclusive, int endExclusive)** - возвращает последовательность Stream с указанным диапазоном указанного массива в качестве его источника.

**toString(byte[] a)** - возвращает строковое представление содержимого указанного массива.