МАССИВЫ

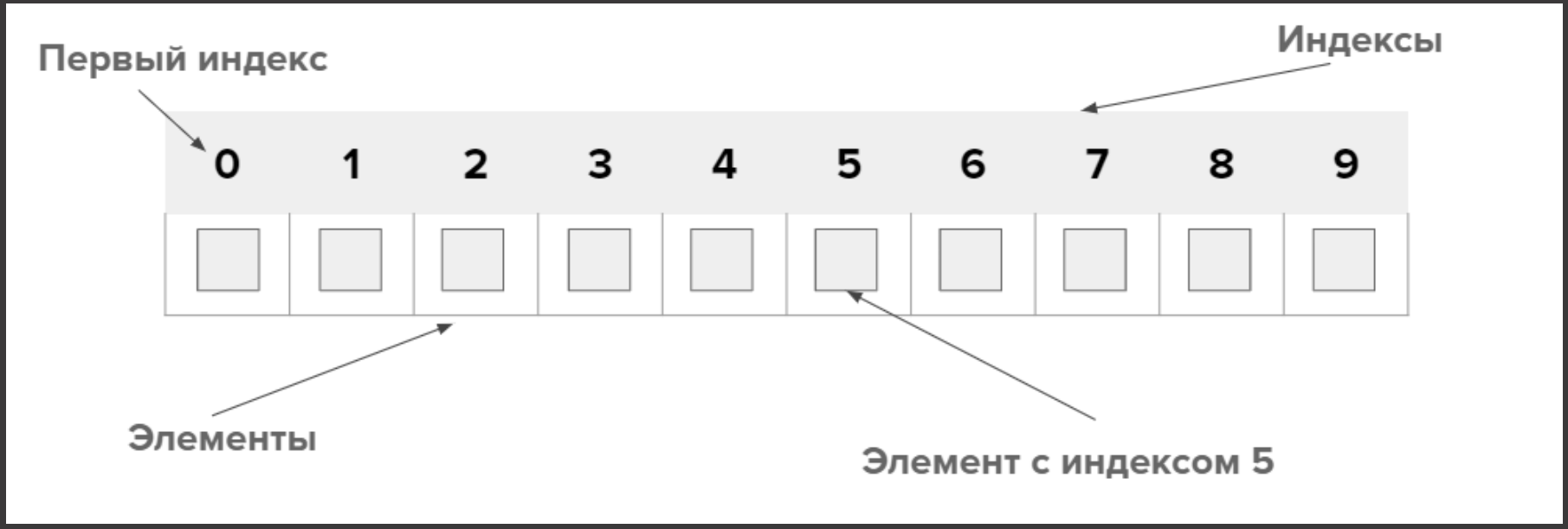
Массив / Элемент / Индекс

Массив - это объект (ссылочный тип), представляющий некоторую непрерывную область памяти определенного типа. Массивы позволяют хранить наборы данных одного типа.

Элемент массива - каждое значение в данном массиве.

Индекс (тип int) - указывает на позицию конкретного элемента относительно начала массива. Нумерация элементов массива начинается с 0.

Массив на 10 элементов
Длина массива равна 10



Класс Arrays

```
import java.util.Arrays
```

Перейти к описанию класса Arrays в документации

Класс с набором статических методов, позволяющий решать типовые задачи при работе с массивами.

В этом классе собраны методы для решения самых распространенных задач, с которыми сталкиваются программисты при работе с массивами.

Объявление переменной массива

```
1 int[] arr; // первый вариант (предпочтительный способ)
2 int arr[]; // второй вариант
```

int - указание на тип данных, которые будут храниться в массиве

[ ] - указание на то, что это массив

arr - имя переменной массива

Создание массива, заполненного значениями по умолчанию

происходит с помощью ключевого слова new
с указанием типа и размера.

```
1 int[] arr;
2 arr = new int[10]; // создали массив типа int на 10 элементов
```

- массивы типа byte/short/int/long заполняются 0
- массивы типа double/float заполняются 0.0
- массивы типа boolean заполняются false
- массивы типа char заполняются '\u0000'
- массивы ссылочного типа заполняются null

Инициализация массива с указанием значений

```
1 int[] arr2 = {2, 4, 6, 8, 10, 12}; // массив типа int на 6 элементов
```

Можно инициализировать безымянный массив и ссылку на него присвоить объявленной ранее переменной.

```
1 arr2 = new int[]{1, 3, 5, 7, 9}; // массив типа int на 5 элементов
2 //предыдущий массив остается в памяти, пока не будет удален сборщиком
```

Размер массива (тип int)

После создания массива изменить его размер невозможно (можно изменять отдельные его элементы).

Если необходимо часто изменять размер массива, лучше использовать другую структуру данных, например ArrayList.

Узнать количество элементов в массиве можно через его свойство length (нельзя изменить).

```
1 int arrLength = arr.length; // 10
```

После создания массива можно обратиться к любому его элементу, прочитать или изменить его.

Обращение к элементу массива происходит по имени массива, за которым следует значение индекса элемента, заключенного в квадратные скобки.

В качестве индекса можно использовать числа или выражения, создающие значение типа int

```
1 arr[3] = 3102; // присвоили значение элементу с индексом 3
2 System.out.println(arr[1]); // вывели значение элемента с индексом 1
3 // доступ к несуществующему элементу массива
4 // приводит к java.lang.ArrayIndexOutOfBoundsException
5 System.out.println(arr[100]);
```

Перебор массива через цикл for позволяет изменять значения массива

```
1 int nums[] = {12, 13, 14, 15, 16, 17};
2 for (int i = 0; i < nums.length; i++) {
3         System.out.println(nums[i]);  // прочитали элемент
4         nums[i] += 3; // изменили элемент
5 }
6 System.out.println(Arrays.toString(nums)); // [15, 16, 17, 18, 19, 20]
```

Перебор массива через цикл foreach не позволяет изменять значения массива

Цикл foreach используется для перебора элементов массива или коллекции. Позволяет пройти весь массив без использования индекса элемента, в этом случае мы вообще не имеем доступа к индексам.

```
1 int nums[] = {12, 13, 14, 15, 16, 17};
2 for (int num: nums){
3     System.out.println(num); // вывод значений элементов массива
4     num += 3; // не можем изменить элемент,
5               // т.к. num перезапишется на следующей итерации
6 }
7 System.out.println(Arrays.toString(nums)); // [12, 13, 14, 15, 16, 17]
```

Заполнение массива значениями в цикле

```
1 int[] arr = new int[47];
2 for (int i = 0; i < arr.length; i++){
3       arr[i] = i * 2;
4 }
5 System.out.println(Arrays.toString(arr));
```

```
1 int[] arr =  { 2, 4, 6, 8, 10, 12};
2 // не копирование, а ссылка на один массив, на одну область памяти
3 int[] arr2 = arr;
```

Метод массива clone()

Копирование массива

```
1 int[] arr = {2, 4, 6, 8};
2 int[] arr2 = arr.clone(); // arr2 - полная копия массива arr
```

Метод System.arraycopy()

Копирование массива

```
System.arraycopy(Object src, int srcPos, Object dst, int dstPos, int len)
```

Метод копирует len элементов массива src, начиная с позиции srcPos, в массив dst, начиная с позиции dstPos. Массив dst должен иметь достаточный размер, чтобы в нем поместились все копируемые элементы.

Метод Arrays.copyOf()

Копирование массива

```
Arrays.copyOf(Object originalArr, int newLength)
```

Позволяет скопировать несколько первых элементов массива или сделать полную копию массива.

Создает и возвращает массив на newLength элементов и копирует в него элементы из originalArr массива.

Метод Arrays.copyOfRange()

Копирование массива

```
Arrays.copyOfRange(Object originalArr, int from, int to)
```

Создает и возвращает массив, в который копирует часть массива originalArr, начиная с элемента с индексом from до элемента с индексом to. to индекс может быть больше чем длина исходного массива, тогда остальные элементы заполняются значениями по умолчанию.
Длина нового массива будет равна to - from.

```java
// Нельзя сравнивать массивы с помощью == или метода equals(),
// т.к происходит сравнение ссылок
int[] arr1 = new int[10];
int[] arr2 = new int[10];
System.out.println(arr1.equals(arr2)); // false
System.out.println(arr1 == arr2); // false
```

Метод Arrays.equals()

Сравнение массивов

```java
Arrays.equals(Object arr1, Object arr2)
```

Сравнивает массивы arr1 и arr2 сначала по размеру, если он одинаковый - по содержимому.

Метод Arrays.sort()

Сортировка массива

```
Arrays.sort(Object arr)
Arrays.sort(Object arr, int fromIndex, int toIndex)
```

Сортирует в порядке возрастания массив arr или часть arr от индекса fromIndex до индекса toIndex. Метод использует алгоритм быстрой сортировки .

Метод Arrays.binarySearch()

Поиск в массиве

```
binarySearch(Object arr, Object key)
```

Метод ищет элемент key в отсортированном массиве arr. Метод поиска возвращает индекс найденного элемента массива. Если элемент не найден, то возвращается точка ввода - 1. Точка ввода - индекс, на который key мог бы быть добавлен в массив.
Метод использует алгоритм бинарного поиска.