Массивы Java

Массив / элемент / индекс

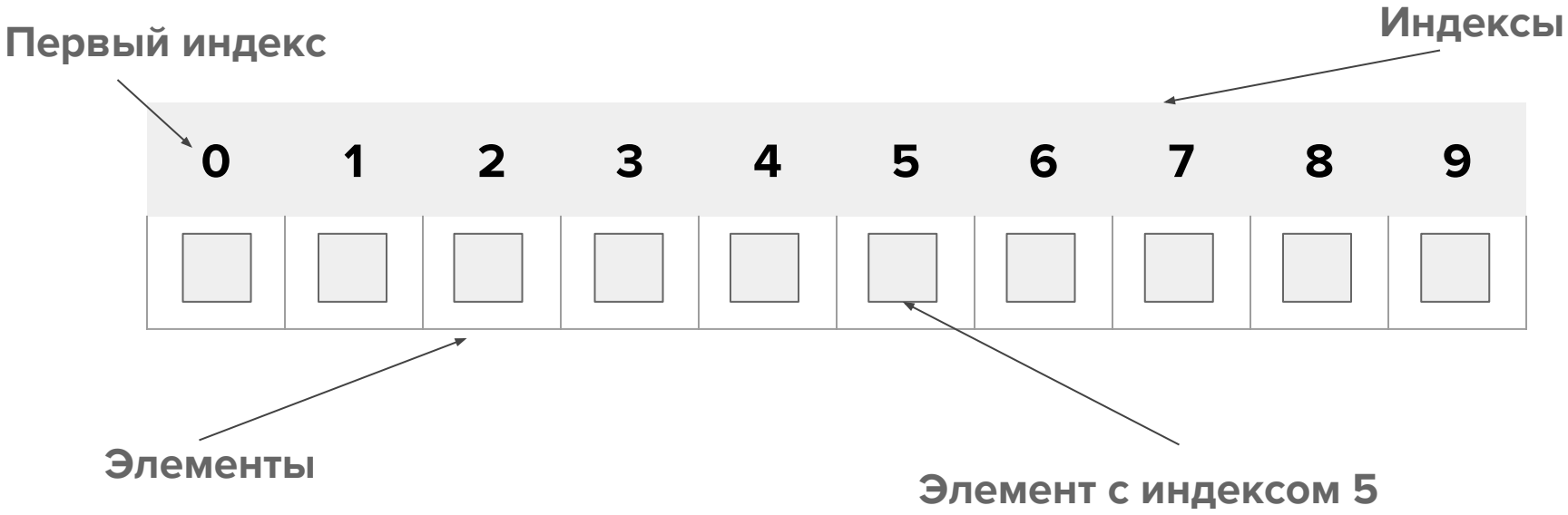
Массив - это объект, представляющий некоторую непрерывную область памяти определенного типа.

Элемент массива - каждая переменная в данном массиве.

Индекс (целое число) указывает на позицию конкретного элемента относительно начала массива.
Нумерация элементов массива начинается с 0.

Массив

Массив на 10 элементов. **Длина массива равна 10**



Класс Arrays

статический класс — Arrays (java.util.Arrays) позволяет решать типовые задачи при работе с массивами.

В этот класс собраны методы для решения самых распространенных задач, с которыми сталкиваются программисты в работе с массивами.

Объявление массива

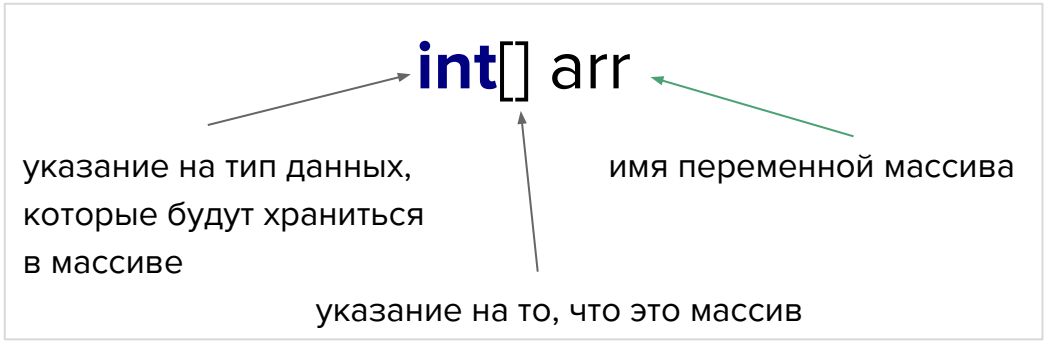
Объявление переменной массива:

Вариант 1 (предпочтительный способ):

```
int[] arr;
```

Вариант 2:

```
int arr[];
```



Создание массива

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

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

System.out.println(Arrays.toString(arr));       // [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Если массив создается таким образом, то всем элементам массива автоматически присваиваются **значения по умолчанию**:

- для числовых значений - 0;
- для массива типа boolean - false;
- для массива типа char - '\u0000;
- для массива типа класса (объекты) - null.

Размер массива

После создания массива **изменить его размер невозможно** (можно изменять отдельные его элементы).
Если в ходе выполнения программы необходимо часто изменять размер массива, лучше использовать другую структуру данных (array list).

Если необходимо узнать **количество элементов в массиве**, можно воспользоваться свойством **length**, которое можно прочитать (но не изменить).

```
int arrLength = arr.length;
```

Инициализация массива в момент создания

Можно инициализировать массив значениями, когда он объявляется, и определить количество элементов таким образом.

```
int[] arr2 = { 2, 4, 6, 8, 10, 12};
System.out.println(Arrays.toString(arr2));
```

Можно инициализировать безымянный массив:
new int[]{ элементы массива } и использовать для повторной инициализации массива без образования новой переменной

```
arr2 = new int[]{ 1, 3, 5, 7, 9, 11};
System.out.println(Arrays.toString(arr2));
```

Доступ к элементам массива

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, которые создают **положительное значение типа int** (при вычислении выражения с типом long, следует преобразовать результат в int. С типами short и byte проблем не будет - они полностью укладываются в диапазон int)

Заполнение массива значениями в цикле

После создания массив можно заполнять, например, с помощью цикла.

```
int[] arr = new int[10];
for (int i = 0; i < arr.length; i++){
      arr[i] = i * 2;
}
System.out.println(Arrays.toString(arr));
```

Копирование массива

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

Способы копирования массива:

1) **.clone()** метод **для полного копирования** массива

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

Копирование массива

Способы копирования массива:

2) **System.arraycopy()** — способ **сделать частичную копию массива**

System.arraycopy(**src**, int **srcPos**,
Object **dest**, int **destPos**,
int **length**)

Метод копирует **length** элементов массива **src**, начиная с позиции **srcPos**, в массив **dest**, начиная с позиции **destPos**.

Массив dest должен иметь достаточный размер, чтобы в нем поместились все копируемые элементы.

Копирование массива

Способы копирования массива:

3) **Arrays.copyOf((originalArr, int newLength)** — скопировать **несколько первых элементов** массива или **сделать полную копию** массива. Создает массив на **newLength** элементов и копирует в него элементы из **original** массива.

```
int[] arr = new int[] {1, 2 ,3};
int[] arrCopy = Arrays.copyOf(arr, 5); // из массива arr скопируются все
                                       элементы, элементы 4й и 5й заполнятся 0ми
```

Копирование массива

Способы копирования массива:
4) **Arrays.copyOfRange((originalArr, int from, int to)** — возвращает массив, в который копирует часть массива **originalArr**, начиная с элемента с индексом **from**. **to** индекс может быть больше чем длина исходного массива, тогда остальные элементы заполняются значениями по умолчанию.
Длина нового массива будет равна **to - from**.

```
int arr[] = { 12, 13, 14, 15, 16, 17, 18 };
int[] copy = Arrays.copyOfRange(arr, 2, 12);

System.out.println(Arrays.toString(copy)); // [14, 15, 16, 17, 18, 0, 0, 0, 0, 0]
```

Перебор значений массива

Способ 1 - **через цикл for**

```
int arr[] = { 12, 13, 14, 15, 16, 17, 18 };

for (int i = 0; i < arr.length; i++) {
        System.out.println(arr[i]);  // прочитали элемент
        arr[i] += 3; // изменили элемент
}
System.out.println(Arrays.toString(arr));    // [15, 16, 17, 18, 19, 20, 21]
```

Перебор значений массива

Способ 1 - **через цикл foreach**

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

```
for (int num: arr){
   System.out.println(num); // вывод значений элементов массива
   num += 3; // не можем изменить элемент,
                  тк num перезапишется на следующей итерации
}
System.out.println(Arrays.toString(arr)); // [12, 13, 14, 15, 16, 17, 18]
```

Сравнение массивов

Нельзя сравнивать массивы с помощью == или .equals()

```
int[] arr1 = new int[10];
int[] arr2 = new int[10];
System.out.println(arr1.equals(arr2)); // false
System.out.println(arr1 == arr2); // false
```

Массивы **нужно сравнивать по содержимому**

```
int[] arr1 = new int[10];
int[] arr2 = new int[10];
System.out.println(Arrays.equals(arr1, arr2)); // true
```

Другие операции над массивами с использованием класса Arrays

Наполнение массива данными

fill(arr, val) - наполняет массив **arr** значениями **val**

```
int[] arr1 = new int[10];
Arrays.fill(arr1, 1);
System.out.println(Arrays.toString(arr1)); // [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Другие операции над массивами с использованием класса Arrays

Сортировка массива

sort(arr, int fromIndex, int toIndex) - сортирует массив методом quick sort. Сортирует массив **arr** или часть массива **arr** от **fromIndex** до **toIndex**.

```
int[] arr1 = new int[]{4, 3, 7, 1, 9, 2, 0, 8, 6, 5};

Arrays.sort(arr1);

System.out.println(Arrays.toString(arr1)); // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Другие операции над массивами с использованием класса Arrays

Поиск в массиве

binarySearch(arr, key) – ищет элемент методом бинарного поиска
Он отыскивает элемент **key** в массиве **arr**.
Метод поиска **возвращает индекс найденного элемента** массива.
Если элемент не найден, то возвращается отрицательное число, означающее индекс, с которым элемент был бы вставлен в массив в заданном порядке, с обратным знаком.

```
int[] arr1 = new int[]{4, 3, 7, 1, 9, 2, 0, 8, 6, 5};
System.out.println(Arrays.binarySearch(arr1, 7)); // 2
```

Другие операции над массивами с использованием класса Arrays

Содержит ли массив указанный элемент

Используется прием с представлением массива в виде коллекции и вызовом метода **contains()**.

```
String[] stringArr = { "a", "c", "e" };
boolean contains = Arrays.asList(stringArr).contains("e");
System.out.println(contains); // true
```