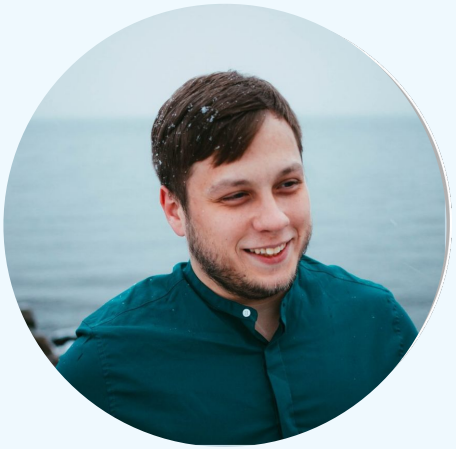
нетология

Массивы одномерные

Филипп
Воронов



Филипп Воронов

Teamlead, VK Group

@Филипп Воронов

План занятия

1. Массив, одномерные массивы, структура массива
2. Инициализация массива
3. Обращение к элементу массива
4. Сортировка массива
5. Метод `clone()`
6. Класс `Arrays`

Массив, одномерные массивы, структура массива

Что такое массив? Одномерные массивы

Массив – это набор элементов (переменных) одного типа.

Индекс – это позиция элемента в массиве (начинается индекс с нуля).

В одномерных массивах для определения местоположения элемента нужно указать значение одного индекса.

Структура массива

Для объявление массива надо указать тип, размер и имя. Массив можно создать двумя способами:

```
тип имя_переменной[] = new тип[размер];

или

тип[] имя_переменной = new тип[размер];
```

тип – базовый тип элементов массива;

размер – число элементов массива (выделение памяти для элементов).
Элементы массива могут быть любого типа, допустимого в языке Java.

Структура массива

После определения размера, выделяется память для массива. Конечно же выделенная память не может быть пустой, для разных типов данных память заполняется разными значениями.

При выделении памяти элементы массива заполняются:

- для числовых типов — нулевыми значениями;
- для логического типа `boolean` — значениями `false`;
- для ссылочных типов — значениями `null`.

Инициализация массива

Предположим у нас есть список цен на товары. Нам нужно этот список сохранить, для этого определим массив типа `int` и дадим размер массиву (например, у нас список из 5 цен).

```
int[] prices = new int [5];
```

И так у нас есть массив с определенным размеров и нулевыми значениями, осталось заполнить его значениями.

Для того чтобы заполнить значениями, нужно взять ячейки массива и присвоить им значения. К примеру заполним первую ячейку:

```
prices[0] = 100;
```

Инициализация массива

Заполним остальные ячейки:

```
prices[1] = 200;
prices[2] = 300;
prices[3] = 400;
prices[4] = 500;
```

Суммируя вышесказанное, инициализация одномерного массива из 5 чисел типа `int`, и заполнение ячеек будет выглядеть так:

```
int[] prices = new int [5];

for (int i=0; i<prices.length; i++){
    prices[i] = i;
}
```

Инициализация массива

Также можно заполнить ячейки значениями вместе с инициализацией массива.

```
double prices[] = {100, 200, 300, 400, 500};
```

Инициализация одномерного массива классов

Типом для одномерного массива может быть и объект, ссылочный тип.

Предположим, нам нужно сохранить список продуктов и цен в одном массиве.

Цены и продукты должны быть связаны между собой и быть частью одного объекта.

Создадим класс `Product`, объединяющий их.

Инициализация одномерного массива классов

```
public class Product {
    private int price;
    private String name;

    public Product(int price, String name) {
        this.price = price;
        this.name = name;
    }

    public Product() {
    }

    public int getPrice() {
        return price;
    }

    public String getName() {
        return name;
    }
}
```

Инициализация массива классов и заполнение ячеек

У нас уже есть класс `Product`, теперь осталось создать массив, который будет в себе хранить данные этого типа. Создать массив можно двумя способами:

Способ N1

```
Product[] products = new Product[2];
```

После создадим объекты типа `Product` и присвоим их в качестве элементов массива:

```
products[0] = new Product(20, "Хлеб");
products[1] = new Product(10, "Соль");
```

Инициализация массива классов и заполнение ячеек

Второй способ: создать объект класса со значениями во время инициализации массива:

```
Product[] products = {new Product(20, "Хлеб"), new Product(10, "Соль")};
```

Вопрос

Как вы думаете, скомпилируется ли такой код:

```
System.out.println({new Product(20, "Хлеб"), new Product(10, "Соль")});
```

Ответ

Нет, так как не указан тип массива.

Правильная инициализация массива в этом случае должна быть следующей:

```
System.out.println(new Product[]{new Product(20, "Хлеб"), new Product(10, "Соль")});
```

Обращение к элементу массива

Предположим, у нас есть массив с названиями продуктов.

```
String[] products = new String[5];
```

Запишем в каждый элемент массива по продукту:

- products [1] = "Хлеб"; в 2-й элемент массива записать продукт "Хлеб"
- products [0] = "Соль"; в 1-й элемент массива записать продукт "Соль"
- products [3] = "Сыр"; в 4-й элемент массива записать продукт "Сыр"
- products [5] = "Масло"; ошибка — ArrayIndexOutOfBoundsException: 5

При попытке записи продукта "Масло" в элемент массива с индексом 5 у нас вылетела ошибка `ArrayIndexOutOfBoundsException`, это означает, что указанный индекс не соответствует диапазону размера массива и программа не может найти элемент массива с данным индексом.

Сортировка массива

У нас уже есть список продуктов с названиями и ценами. Очень часто в онлайн магазинах вы можете заметить свойства сортировки товаров (например, по цене). Рассмотрим сортировку подробнее.

Пример сортировки одномерного массива методом «пузырька».

Объявление массива из 10 элементов

```
float prices[] = new float[10];
```

вспомогательная переменная

```
prices tempPrice;
```

Сортировка массива

Заполнение случайными значениями массива `prices` (случайные значения в диапазоне от 0 до 10 можно получить, использовав метод `new Random().nextInt(11)`).

Полная формула выглядит так – `new Random().nextInt((max – min) + 1) + min`

```
for (int i = 0; i < prices.length; i++){
    prices[i] = new Random().nextInt(11);
}
```

Пример текста на слайде

Ниже представлен цикл сортировки методом «пузырька»:

```
for (int i=0; i<prices.length-1; i++){
    for (int j=i; j>=0; j--){
        if (prices [j]> prices [j+1])
        {
            tempPrice = prices[j];
            prices [j] = prices [j+1];
            prices [j+1] = tempPrice;
        }
    }
}
```

Кроме метода пузырька существуют и другие методы сортировки, самые распространенные из них это: QuickSort и MergeSort.

Метод `clone()`

Метод `clone()`

Часто возникает необходимость создать копию уже существующего массива и произвести работу над ним. Давайте рассмотрим, как правильно создать копию массива. Предположим, у нас есть массив с ценами продуктов:

```
int[] prices = {10, 20, 30, 40, 50};
```

Нам нужно создать еще один массив с теми же данными, которые мы будем обрабатывать. Как же правильно это сделать?

Пример текста на слайде

Создадим массив с новыми ценами:

```
int[] newPrices = new int[5];
```

присвоим первый массив второму:

```
newPrices = prices
```

В данном случае, если изменить значения в первом массиве, то изменения также коснутся второго массива, так как `prices` и newPrices указывают на один и тот же участок памяти. Они копируют не значения, а ссылки на них.

Такое копирование также называется «неглубокое копирование».

Пример текста на слайде

Для того чтобы заполнить второй массив значениями из первого массива, нужно использовать метод `clone()`:

```
newPrices = prices.clone();
```

На этот раз массивы `newPrices` и `prices` — разные массивы, так как были скопированы не ссылки, а значения, то есть были созданы два различных объекта.

Такое копирование также называется «глубокое копирование».

Вопрос

Как вы думаете, что напишет код ниже:

```
int array1 = {1, 2, 3};
int array2 = {1, 2, 3};

System.out.println(array1.equals(array2));
```

Вопрос

```
int array1 = {1, 2, 3};
```

А что выведет метод `array1.toString()`?

Ответ

Метод `equals` и `toString` наследуются от класса `Object`, а как мы знаем, `toString` возвращает адрес в памяти в 16-ричном виде, а метод equals — сравнивает не значения полей или элементов внутри массива, а адреса в памяти.

Ответы на предыдущие вопросы: false и [I@2f7c7260 (адрес в памяти)

Для решения таких задач еще в JDK 1.2 был добавлен специальный утильный класс `Arrays`, давайте рассмотрим его подробнее.

Класс `Arrays`

Класс `Arrays`

Как вы уже поняли, Java позаботилась о нас и внесла метод `clone()`, который помогает легко копировать класс. Кроме него для работы с массивами были созданы другие вспомогательные инструменты — например, класс `Arrays`.

- `copyOf()` – копирует массив;
- `copyOfRange()` – копирует часть массива;
- `toString()` – возвращает все элементы в виде одной строки;
- `sort()` – сортирует массив;
- `binarySearch()` – ищет элемент методом бинарного поиска;
- `fill()` – заполняет массив переданным значением;
- `equals()` – проверяет на идентичность массивы;
- `asList()` – возвращает массив как коллекцию.

Сортировка массивов примитивов (Arrays)

Рассмотрим пример использования `Arrays.sort()`. Данный метод получает на вход как примитивные, так и ссылочные типы данных:

```
int[] numbers = {4, 9, 1, 3, 2, 8, 7, 0, 6, 5};
java.util.Arrays.sort(numbers);
```

на выходе:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Метод `toString` (Arrays)

Метод `toString` в массивах наследуется от класса Object и не несет кроме адреса в памяти какую-то еще пользу.

Для вывода элементов массива рекомендуется использовать метод утильного класса `Arrays`:

```
int[] numbers = {4, 9, 1, 3, 2, 8, 7, 0, 6, 5};
java.util.Arrays.toString(numbers);
```

на выходе:

```
[4, 9, 1, 3, 2, 8, 7, 0, 6, 5]
```

Лекционная задача

Давайте разберем пример.

Чему мы научились

- Создавать одномерные массивы;
- Работать с элементами одномерных массивов;
- Сортировка одномерного массива;
- Использовать методы Arrays для одномерных массивов.

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Филипп Воронов

@Филипп Воронов