

# Массивы

## Когда данных действительно много

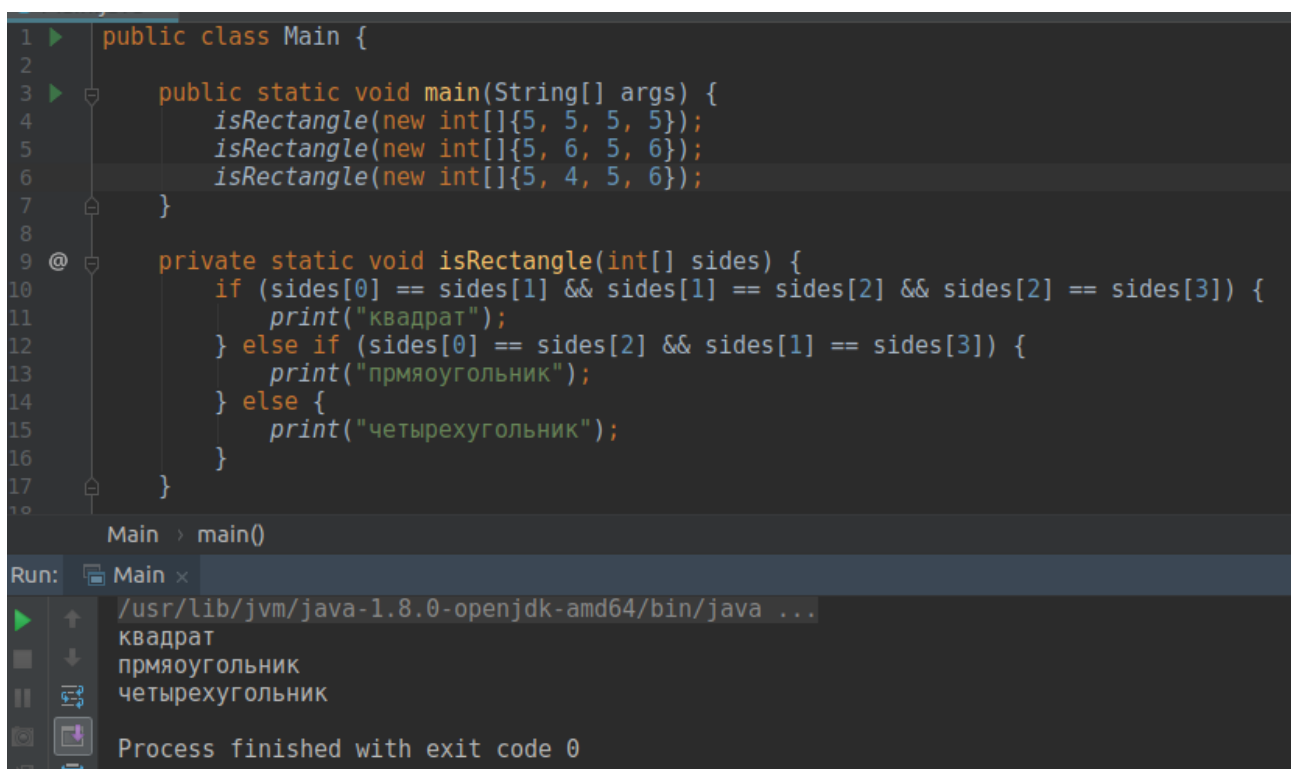
### Содержание

1. Представление однотипных данных посредством массива
2. Введение в циклы

### 1. Представление однотипных данных посредством массива

Если вы решали задачи из 11-ой лекции, то хорошо, если нет, то ничего страшного. Можете решать их постепенно, в своем ритме. Мы никуда не спешим. Возвращайтесь к решению задач когда почувствуете силы и решимость. А сейчас мы вспомним задачу 16.

Метод на вход получает 4 стороны четырехугольника и сравнивая их принимает решение какой это тип. И мы сейчас попробуем немного улучшить этот код так, чтобы не передавать 4 аргумента в метод, а передать 1, который бы в себе имел все эти 4 стороны. Нам понадобится новая штука под названием массив



```
1 public class Main {
2
3     public static void main(String[] args) {
4         isRectangle(new int[]{5, 5, 5, 5});
5         isRectangle(new int[]{5, 6, 5, 6});
6         isRectangle(new int[]{5, 4, 5, 6});
7     }
8
9     private static void isRectangle(int[] sides) {
10         if (sides[0] == sides[1] && sides[1] == sides[2] && sides[2] == sides[3]) {
11             print("квадрат");
12         } else if (sides[0] == sides[2] && sides[1] == sides[3]) {
13             print("прямоугольник");
14         } else {
15             print("четырёхугольник");
16         }
17     }
18 }
```

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

квадрат  
прямоугольник  
четырёхугольник

Process finished with exit code 0

Чтобы понимать как объявлять давайте вынесем переменную в мой метод. Выделим аргумент функции в первой линии(4) и нажмем на Ctrl+Alt+V. И мы видим такое же объявление переменной типа int, но за исключением квадратных скобок. Они указывают

именно на то, что у нас массив чисел, а не одно число. Далее мы видим как инициализация поменялась с `new int[] { }` на `{ }`. Дело в том, что мы изначально написали тип переменной – массив `int`. А когда мы передаем сразу массив аргументом функции то нужно указать тип `new int[]`, ведь это же мог быть не `int`, а `byte` или другой тип. Да, массивом может быть что угодно. Любой примитив или класс. Если вам нужен массив строк, то это будет `String[] texts`.

```
3 public static void main(String[] args) {  
4     int[] sides = {5, 5, 5, 5};  
5     isRectangle(sides);  
6     isRectangle(new int[]{5, 6, 5, 6});  
7     isRectangle(new int[]{5, 4, 5, 6});  
8 }
```

Ладно, вроде разобрались. Создаем массив типа `int` и передаем ему 4 числа в фигурных скобках или же через `new int[] { }` если без переменной. Ок. Как же дальше мы получим значения из этой одной переменной? Смотрим на линию 10.

`sides[0]`, `sides[1]`, `sides[2]`, `sides[3]` это соответственно значения массива по порядку добавления. В программировании принято считать с нуля и поэтому первое число будет находиться на нулевом месте. т.е. позиции. Если раньше у нас были аргументы `int AB`, `int BC`, `int CD`, `int DA`, то теперь у нас массив чисел и теперь соответственно

`AB - sides[0]` , `BC – sides[1]`, `CD-sides[2]`, `DA-sides[3]`.

Хорошо, вроде разобрались. Но есть одно но. Наша функция ожидает массив из 4 чисел. Что же будет, если передать туда не 4 числа, а скажем 3 или 5 или любое другое? Скорей всего мы получим ошибку (про ошибки поговорим попозже, ок?). Для этого давайте просто добавим 1 проверку входного параметра. Помните мы говорили, что в этой задаче нужно еще проверить входные данные, что эти стороны не ноль. Так вот, сейчас проверка будет выглядеть так.

Линия 11 – мы проверяем что нам пришел массив длиной 4. Т.е. как и у строки `text.length` у массива тоже есть длина. Это ее размер. Мы просто проверим что нам пришло 4 значения. После уже внутри мы проверим что они все больше нуля (линия 12). И только тогда, когда мы уверены в том, что метод получил корректные входные данные мы можем просто взять и проверить логику.

И кто-то можем спросить, ок, мы заменили 4 аргумента одним и как видим проверок стало на 1 больше, так в чем же тогда профит? В чем удобство? Ведь мы же поменяли 3 числа на массив не просто так, а для какого-то удобства, верно? Да, именно так.

И нам пора сейчас изучить еще одну ключевую штуку, такую же важную и нужную как условия.

И напоследок еще одну штуку про массивы. Вы можете сначала объявить массив и потом уже заполнять его. Когда объявляем массив нужно передать число между квадратными – длина массива. В данном случае это 2 – т.е. будет 2 значения в массиве.

```
int[] array = new int[2];
```

```
array[0] = 101;
```

```
array[1]= 202;
```

Но если вы можете сразу заполнить массив, то делайте это в 1 линию. Так проще.

```
3 public static void main(String[] args) {
4     int[] sides = {5, 5, 5, 5};
5     isRectangle(sides);
6     isRectangle(new int[]{5, 6, 5});
7     isRectangle(new int[]{5, 4, -5, 6});
8 }
9
10 private static void isRectangle(int[] sides) {
11     if (sides.length == 4) {
12         if (sides[0] > 0 && sides[1] > 0 && sides[2] > 0 && sides[3] > 0) {
13             if (sides[0] == sides[1] && sides[1] == sides[2] && sides[2] == sides[3]) {
14                 print("квадрат");
15             } else if (sides[0] == sides[2] && sides[1] == sides[3]) {
16                 print("прямоугольник");
17             } else {
18                 print("четырёхугольник");
19             }
20         } else {
21             print("все стороны должны быть больше нуля");
22         }
23     } else {
24         print("нужно 4 числа");
25     }
26 }
27
28
Main > main()
Run: Main x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
квадрат
нужно 4 числа
все стороны должны быть больше нуля
Process finished with exit code 0
```

## 2. Введение в циклы

Давайте вспомним для этого задачу 13 из 11-ой лекции.

Метод получает на вход 5 оценок и считает их арифметическое среднее. И наш метод получал 5 входных параметров. Но вопрос, а что если у нас не 5 оценок, а например 6? Тогда нам нужен еще один метод на 6 аргументов, так? Да. А если 50 оценок нужно обработать? А если 500? Было бы классно передать 1 массив. Но теперь перед нами стоит задача посчитать арифметическое среднее всех этих оценок. Для этого надо сложить их все и разделить на их количество. И здесь возникает резонный вопрос, как сложить все числа в массиве, если мы не знаем сколько их? Мы же не можем написать 100500 условий? Этого и не нужно. Специально для таких задач в Java есть циклы. Сначала обговорим это на словах. Вам нужно взять первое число в массиве и прибавить к нему следующее, потом к этой сумме добавить следующее и так до тех пор, пока элементы есть в массиве. Так? Значит нам нужна какая-то штука, которая пройдет по всем значениям массива и что-то сделает с каждым элементом. Итак, знакомьтесь, цикл for.

Вообще структура цикла for такая – for (итератор, условие, изменение итератора) и фигурные скобки. Что такое итератор? Это какое-нибудь число например, счетчик. Оно отвечает за начало и конец цикла. Например в нашем случае мы инициализируем его нулем, ставим условие что оно будет меньше длины массива и после каждой итерации (повтора цикла) мы будем увеличивать итератор на 1. Если помните i++ значит увеличить на 1 значение переменной и сохранить в ней.

```
3  ▶ public static void main(String[] args) {
4      checkMarks(new int[] {5,4,3,5});
5      checkMarks(new int[] {5,4,3,5,2,4,5});
6  }
7
8  @ private static void checkMarks(int[] marks) {
9      int average;
10     int sum = 0;
11     for (int i = 0; i < marks.length; i++) {
12         sum += marks[i];
13     }
14     average = sum / marks.length;
15
16     if (average == 5) {
17         print("Отличник");
18     } else {
19         print("не отличник");
20     }
21 }
```

Main > checkMarks()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

не отличник

не отличник

Process finished with exit code 0

Давайте конкретно на нашем примере. Мы создаем 2 переменные для среднего значения и суммы. Сумму инициализируем нулем. После создаем цикл, который пройдет по значениям массива. Для этого ставим условие что итератор будет ходить от нуля до длины массива, т.е. в случае с массивом из 3 элементов будет так

1.  $i=0$ ,  $i < 3$ ,  $i++$
2.  $i=1$ ,  $i < 3$ ,  $i++$
3.  $i=2$ ,  $i < 3$ ,  $i++$
4.  $i=3$ ,  $i < 3$  стоп. Нет, 3 не меньше 3 значит заканчиваем цикл и выходим из него.

Чтобы понять это хорошо, просто поставьте брейкпойнт на линии 12 и посмотрите на значения. Наш код для массива из 3 значений будет ходить по линиям так

9,10,11,12,11,12,11,12,11,14, 15...

Вот почему это называется цикл, что мы повторяем одно и то же действие до тех пор, пока выполняется условие в цикле. Когда мы доходим до конца цикла, то срабатывает код в третьем блоке –  $i++$ .

Чтобы было еще понятнее, давайте напишем простой цикл и выведем данные в консоль

```
Main.java
1 public class Main {
2
3     public static void main(String[] args) {
4         for (int i = 0; i < 3; i++) {
5             print(i);
6         }
7     }
8 }

Main > main()

Run: Main x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
0
1
2
Process finished with exit code 0
```

Мы написали цикл от 0 до 3 исключительно (ведь знак <, а не <=). И потому в консоли мы видим 0, 1 и 2. Надеюсь понятно, сейчас мы вернемся к нашей задаче

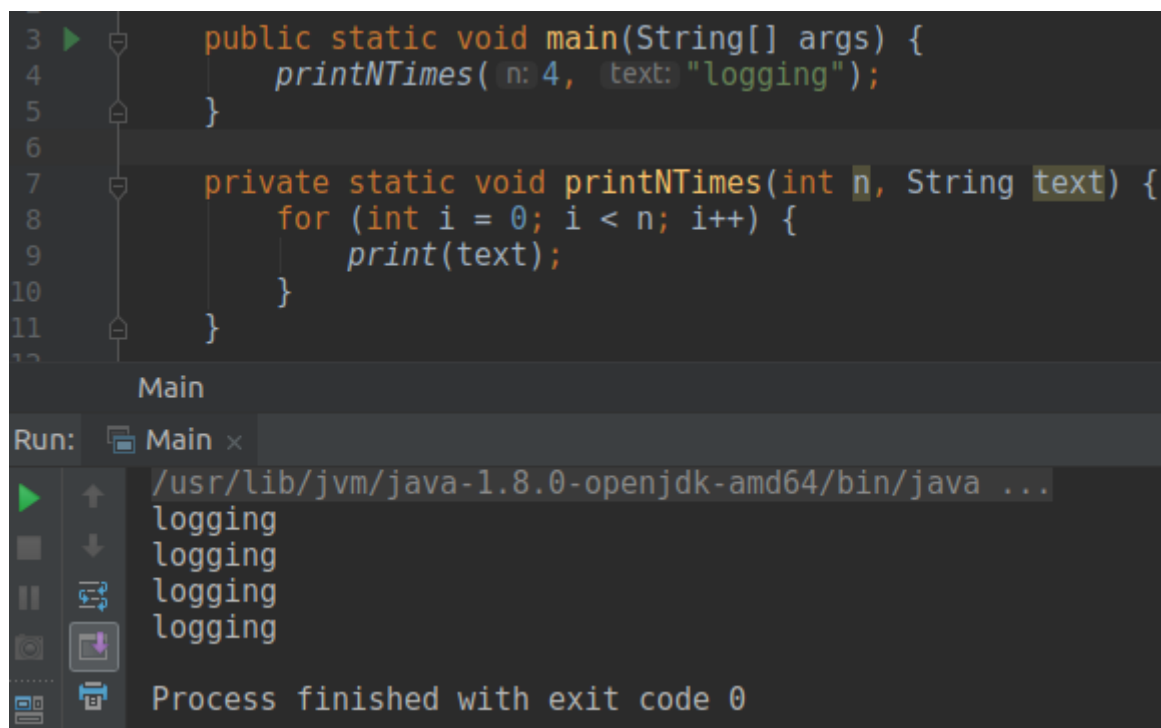
```
3 public static void main(String[] args) {
4     checkMarks(new int[]{5, 4, 3, 5});
5 }
6
7 @ private static void checkMarks(int[] marks) {
8     int average;
9     int sum = 0;
10    for (int i = 0; i < marks.length; i++) {
11        sum += marks[i];
12        print("i = " + i + " marks[i] = " + marks[i] + ", sum: " + sum);
13    }
14    average = sum / marks.length;
15
16    if (average == 5) {
17        print("Отличник");
18    } else {
19        print("не отличник");
20    }
21 }
22

Main > checkMarks()

Run: Main x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
i = 0 marks[i] = 5, sum: 5
i = 1 marks[i] = 4, sum: 9
i = 2 marks[i] = 3, sum: 12
i = 3 marks[i] = 5, sum: 17
не отличник
Process finished with exit code 0
```

Давайте выводить в консоль все что есть. Видим – первая итерация, i=0. Первый элемент массива 5, сумма 5. Логично, ноль плюс пять будет пять. Идем дальше. Вторая итерация: i=1, элемент массива равен 4, сумма равна 9. Ведь 5+4 будет 9. И так дальше.

Чтобы объяснить циклы еще проще, давайте подумаем над такой задачей. Предположим нам нужно вывести 5 раз одну и ту же строку. Как это сделать? Можно написать метод, который внутри вызывает 5 раз один и тот же метод print. Но как быть, если нам в разных случаях нужно вызывать разное количество раз этот метод? Давайте напишем метод для этого.



```
3 public static void main(String[] args) {  
4     printNTimes( n: 4, text: "logging");  
5 }  
6  
7 private static void printNTimes(int n, String text) {  
8     for (int i = 0; i < n; i++) {  
9         print(text);  
10    }  
11 }  
12
```

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...  
logging  
logging  
logging  
logging

Process finished with exit code 0

Но конечно же циклы были созданы именно для того, чтобы проще работать с такими переменными типа массив. Где ты не знаешь сколько раз нужно пройти по значениям, какой длины массив и сколько раз нужно прочитать его. Ведь в реальности никому не нужно выводить одну и ту же строку более 1 раза. Именно поэтому мы проходим циклы одновременно с массивом. Если вы что-то не поняли, то не переживайте, мы будем некоторое время проходить циклы и массивы и решать задачи. Так что вот вам задачи для закрепления

1. Написать метод, который найдет минимум из любого количества чисел (аргумент массив).
2. Написать метод, который перемножит все числа в массиве
3. Написать метод, который посчитает количество отрицательных чисел в массиве
4. Написать метод, который принимает на вход число и массив. Вывести, сколько раз в этом массиве встречается это число которое передали в аргументе.
5. Написать метод, который принимает массив строк и проверяет сколько в них пустых строк.
6. Написать метод, который проверит, что числа массива являются арифметической последовательностью

Более подробно про циклы мы пройдем в следующей лекции. По массивам основная информация у вас есть. Удачи!