



Методы

Функция — часть программы, имеющая собственное имя. Это имя можно использовать в программе как команду (такая команда называется вызовом функции). При вызове функции выполняются команды, из которых она состоит. Вызов функции может возвращать значение (аналогично операции) и поэтому может использоваться в выражении наряду с операциями.

Функции используются в программировании, чтобы уменьшить его сложность:

- Вместо непрерывной последовательности команд, программу разбивают на подпрограммы, каждая из которых решает небольшую законченную задачу, а потом большая программа составляется из вызовов этих подпрограмм
- Уменьшается общее количество кода, потому что, как правило, одна функция используется в программе несколько раз.
- Написанная однажды функция, может быть включена в библиотеку функций и использоваться в других программах (например `System.out.println()` - входит в библиотеку `System`).

Методы

- 
- command1
 - command2
 - **command3**
 - **command4**
 - **command5**
 - command6
 - **command3**
 - **command4**
 - **command5**
 - command7
 - command8
 - **command3**
 - **command4**
 - **command5**

```
myFunction() {  
• command3  
• command4  
• command5  
}
```

- command1
- command2
- ***myFunction()***
command6
- ***myFunction()***
command7
- command8
- ***myFunction()***

Методы

Метод — это функция, являющаяся частью некоторого класса, которая может выполнять операции над данными этого класса. В языке Java вся программа состоит только из классов и функции могут описываться только внутри них. Поэтому все функции в языке Java являются методами.

Для того, чтобы использовать в программе собственный метод, его необходимо объявить.

При объявлении метода указывается:

- Модификаторы
- **Тип значения**, которое будет возвращено после выполнения метода в программу. Если значение возвращать не нужно, указывается ключевое слово `void`.
- **Имя метода** — модификаторы имени метода в круглых скобках указывается список параметров (может быть пустым).
- Тело метода — в фигурных скобках — команды, выполняющиеся при вызове метода

```
public static String myMethod (int par) {  
    ...  
}
```

тело метода
(команды)



Методы

Параметры — это данные, которые нужны методу для работы. Например, метод, рисующий круг, должен получить радиус и координаты центра круга.

Описание каждого параметра аналогично объявлению переменной (тип, а затем идентификатор — имя параметра). Параметры перечисляются через запятую.

В теле метода, возвращающего значение, должна быть команда **return**, после которой через пробел указывается выражение соответствующего типа. Эта команда заканчивает работу метода и передает указанное выражение в качестве возвращаемого значения основной программе — в то место, откуда метод был вызван.

```
int returnIntValueMethod() {
```

```
String returnStrinMethod(int par)
```

```
int massivMin(int[] mass, int n)
```

```
return 154;
```

```
return "the string with parameter " + par;
```

```
return mass[mini];
```

Методы

```
void noReturnMethod() {  
    System.out.println("Do something"); }  
  
int returnIntValueMethod() {return 154;}  
  
String returnStringValueParametersMethod (int par)  
    { return "the string with parameter " + par;}  
  
int massivMin(int[] mass) {  
    int min= mass[0], minin=0;  
    for (int i = 1; i < mass.length; i++) {  
        if (mass[i] < min) {  
            min = mass[i];  
            minin = i;        }}  
    return mass[minin];}  
  
String[] getAmassiv(int j) {  
    String[] mas = new String[j];  
    Scanner vvod = new Scanner(System.in);  
    for (int i = 0; i < j; i++) {  
        mas[i] = vvod.nextLine();}  
    return mas;}
```

Методы

Формальные и фактические параметры

- **Формальные параметры** – используются при описании алгоритма метода. У них есть только имена и типы. До вызова метода они не имеют значений – это только резервирование места для фактических параметров, определяя их число и тип данных
- **Фактические параметры** – те величины, для которых будет исполнен метод. На место формальных параметров в алгоритме метода подставляются реальные значения, переданные при вызове метода.

```
int methodOne(int[] mass, int n, String str)
```

```
...
```

```
methodOne(massiv, 7, "the string");
```

```
// initializing when calling method methodOne  
// mass=massiv; n=7; str="the string"
```

Методы

Вызов метода

Вызов метода осуществляется в теле другого метода, путём написания(обычно) **имени_объекта.имени_метода**.

Если метод объявлен в том же классе, то имени объекта писать не требуется. Если метод возвращает какое-то значение, то оно может быть присвоено переменной.

```
obj.hashCode();
```

```
noReturnMethod();
```

```
// допустимая, но довольно бессмысленная запись.  
// результат вызова метода, возвращающего значение,  
// хорошо бы кому-то присвоить  
methodOne(massiv, 7, "the string");
```

```
int value = methodOne(massiv, 7, "the string");
```


Методы

```
noReturnMethod();
```

```
int val = returnIntValueMethod();
```

```
System.out.println(val);
```

```
System.out.println(30 + returnIntValueMethod());
```

```
System.out.println(returnStringValueParametersMethod(48));
```

```
int[] massiv = { 23, 45, 67, 768, 6, 2, 12, -446, 475 };
```

```
System.out.println(massivMin(massiv));
```

```
String[] mas2 = getAmassiv(5);
```

```
for (String str : mas2)
```

```
    System.out.println(str);
```


Рекурсивные методы

Расчёт факториала числа N

$$N! = 1 * 2 * 3 * 4 * 5 * \dots * (N-2) * (N-1) * N$$

или

$$N! = N * (N-1) * (N-2) * \dots * 5 * 4 * 3 * 2 * 1$$

$$1! = 1;$$

$$2! = 2 * 1! = 2$$

$$3! = 3 * 2! = 6$$

```
int factorial(int i) {  
    int result;  
    if (i == 1)  
        return 1;  
    result = factorial(i - 1) * i;  
    return result;  
}
```

$\text{factorial}(5) = \text{factorial}(4) * 5 = \text{factorial}(3) * 4 * 5 = \text{factorial}(2) * 3 * 4 * 5 = \text{factorial}(1) * 2 * 3 * 4 * 5 = 1 * 2 * 3 * 4 * 5 = 120$

1. Вызов factorial(5)
2. Вызов factorial(4)
3. Вызов factorial(3)
4. Вызов factorial(2)
5. Вызов factorial(1)
6. Вернули 1
7. Вычислили $1 * 2 = 2$
8. Вернули 2
9. Вычислили $2 * 3 = 6$
10. Вернули 6
11. Вычислили $6 * 4 = 24$
12. Вернули 24
13. Вычислили $24 * 5 = 120$
14. Вернули 120



Методы с переменным числом аргументов

```
public void badWay()
{System.out.println("Кол-во арг.: "+0+"Содержимое пусто");}

public void badWay(int arg1)
{System.out.println("Кол-во арг:"+1+" Содержимое: "+arg1);}

public void badWay(int arg1, int arg2, int arg3) {
System.out.println("Количество аргументов: " + 3 + "
Содержимое: ["+ arg1 + "]" ["+ arg2 + "]" ["+ arg3 + "]");}

//-- до 1.5
public void oldWay(int old[]) {
System.out.print("Количество аргументов: " + old.length
+ " Содержимое: ");
for (int i : old)
    System.out.print "[" + i + "] ";
System.out.println();
}
```



Методы с переменным числом аргументов

```
public void varargsway(int ... varargs) {  
    System.out.print("Количество аргументов: " + varargs.length  
    + " Содержимое: ");  
    for (int i : varargs)  
        System.out.print("[ " + i + " ] ");  
    System.out.println();  
}
```

```
public void varargsway2(String arg, int... varargs) {  
    System.out.println(arg);  
    varargsway(varargs);  
}
```

```
varargsway();  
varargsway(1, 56, 90);  
varargsway(89, 34, 77, 44);
```

```
varargsway2("Превед");  
varargsway2("Превед", 1, 23, 45);
```



Методы с переменным числом аргументов

```
public void varargsWay3  
(String arg, int ... varargs , boolean arg2) {}
```

```
public void varargsWay3  
(String arg, int ... varargs , float ... arg2) {}
```



Практика

1. Создать метод вывода на консоль погодных условий. В метод передаётся температура (в градусах Цельсия) и скорость ветра. На консоль выдаётся строки типа: «Холодно, небольшой ветер», «Тепло, безветренно» и т.п.
2. Создать метод проверяющий, что у переданного числа первая цифра равна последней
3. Создать метод, вычисляющий факториал числа ($n! = 1 * 2 * 3 * 4 * \dots * n$)
4. Создать метод, подсчитывающий количество вхождения подстрок в строку
5. Создать метод вычисляющий количество дней в месяце определённого года (с учётом високосности года)
6. Путешественник проходит каждый день несколько километров. Создать метод, выводящий на экран его путь с начала путешествия(в виде "День №1 : 10км; День №2 : 7км; День №3 : 13км; "). Метод должен работать для любого количества дней путешествия.
7. Создать метод сравнивающий (лексиграфически) две строки

Практика

8. Создать метод, возвращающий true, если заданное число находится "повсюду" в целочисленном массиве. Под "повсюду" подразумевается, что при рассмотрении любой пары рядом стоящих элементов массива, одним из элементов будет искомое число, т.е.

```
isEverywhere({1, 2, 1, 3}, 1) → true
```

```
isEverywhere({1, 2, 1, 3}, 2) → false
```

```
isEverywhere({1, 2, 1, 3, 4}, 1) → false
```

9. Создать метод, проверяющий, может ли массив быть "сбалансированным", т.е. разделённым на две части в каком-то месте, таким образом, чтобы сумма элементов одной части равнялась сумме элементов второй. Т.е.

```
{ 1, 1, 1, 2, 1 }; // true
```

```
{ 1, 2, 3, 1, 0, 1, 3 }; // false
```

```
{ 1, 1, 1, 1, 4 }; // true
```

10. Реализовать метод быстрой сортировки одномерного массива с помощью рекурсивного метода.