**Методы в JAVA**

Если Вы видите в программе какое-то слово, а затем круглые скобки - значит это **метод**:

**слово()**

Например - это названия 4 методов:

**println()**

**hasNextInt()**

**getNumber()**

**main()**

Зачем же они нужны?

Давайте поясним на примере. Представьте, что мы пишем программу-калькулятор.

В упрощенном виде программа-калькулятор должна выполнять 4 базовые операции: сложение, вычитание, умножение и деление. Поэтому, если бы мы писали программу-калькулятор, мы бы попросили пользователя:

1. Ввести 2 числа

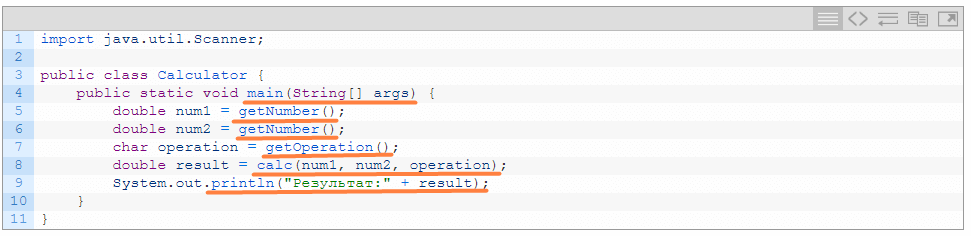
2. Ввести операцию: "+"  для сложения,   "-"  для вычитания, "\*" для умножение и "/" для деления

Поэтому мы бы написали следующий код:

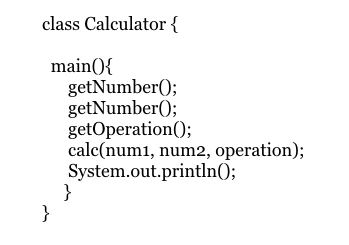
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import java.util.Scanner;    public class Calculator {      public static void main(String[] args) {          double num1 = getNumber();          double num2 = getNumber();          char operation = getOperation();          double result = calc(num1, num2, operation);          System.out.println("Результат:" + result);      }  } |

*Обратите внимание: этот только часть кода программы, а именно главный метод****main****. Если Вы попытаетесь запустить данный код у себя на компьютере, получите ошибку. Полную версию кода мы покажем ниже.*

Что мы тут видим? На самом деле, в каждой строчке выполняется метод:

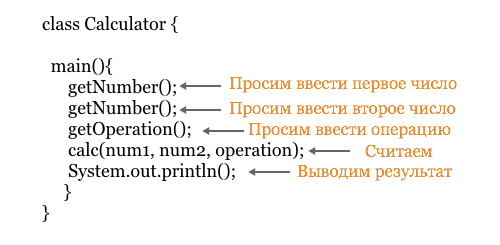


Все, что подчеркнуто - методы. И да-да, main - это тоже метод! Если записать схематически, то получим:



Что делают все эти методы? Какая логика этого кода?

Давайте объясним это так. Каждый метод отвечает за какое-то **действие**:



**Два типа методов**

В коде Вы можете увидеть:

* **методы, которые Вы уже знаете:**

main()

println()

* **методы, которые видите впервые - эти методы мы написали сами:**

getNumber()

getOperation()

calc()

Таким образом, все виды методов в Java можно поделить на две категории:

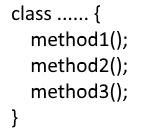
1. **Стандартные**, то есть написанные в стандартных библиотеках Java. Просто берешь нужный метод и используешь.

2. **Пользовательские**, то есть методы, которые Вы сами написали.

**Как методы располагаются в программе**

Мы уже говорили, что метод - это "строительный блок" программы. Как же они располагаются в программе?

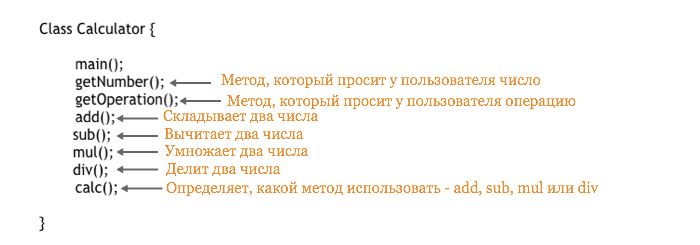
Схематически это можно показать так:



Но выше, когда мы показывали код программы калькулятор, у нас был только один метод - **main()**. Как мы писали, это была только часть кода. Давайте теперь покажем весь - только не пугайтесь

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | import java.util.Scanner;    public class Calculator {      public static void main(String[] args) {          double num1 = getNumber();          double num2 = getNumber();          char operation = getOperation();          double result = calc(num1, num2, operation);          System.out.println("Результат:" + result);      }        public static double getNumber() {          Scanner sc = new Scanner(System.in);          System.out.println("Введите число:");          if(sc.hasNextDouble()) {              return sc.nextDouble();          } else {              System.out.println("Ошибка при вводе. Повторите ввод");              return getNumber();          }      }        public static char getOperation() {          Scanner sc = new Scanner(System.in);          System.out.println("Выберите номер операции:\n1 - прибавить\n2 - отнять\n3 - умножить\n4 - разделить");          int operationNumber = 0;          if(sc.hasNextInt()) {              operationNumber = sc.nextInt();          } else {              System.out.println("Вы ввели не число! Повторите ввод!");              return getOperation();          }          switch (operationNumber) {              case 1:                  return '+';              case 2:                  return '-';              case 3:                  return '\*';              case 4:                  return '/';              default:                  System.out.println("Неправильная операция! Повторите ввод!");                  return getOperation();          }      }        public static double add(double num1, double num2) {          return num1+num2;      }        public static double sub(double num1, double num2) {          return num1-num2;      }        public static double mul(double num1, double num2) {          return num1\*num2;      }        public static double div(double num1, double num2) {          if(num2 != 0.0) {              return num1/num2;          } else {              System.out.println("На 0 делить нельзя!");              return Double.NaN;          }      }        public static double calc(double num1, double num2, char operation) {          switch (operation) {              case '+':                  return add(num1, num2);              case '-':                  return sub(num1, num2);              case '\*':                  return mul(num1, num2);              case '/':                  return div(num1, num2);              default:                  return Double.NaN;          }      }  } |

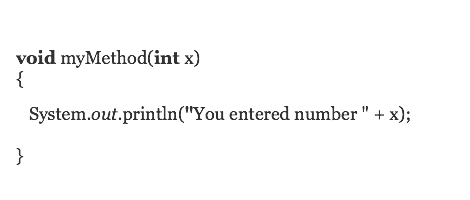
Много? Давайте представим этот код схематически:



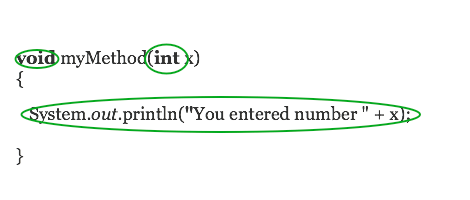
А теперь на пару секунд отойдем от нашего примера и посмотрим, как строится каждый конкретный метод.

**Как же строится метод?**

Давайте посмотрим на самый простой метод:

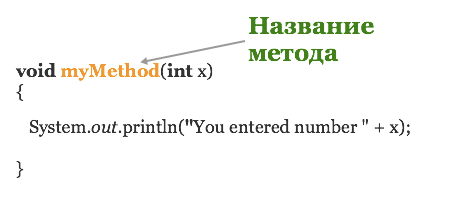


Тааак, что мы тут имеем? Если присмотреться, мы видим знакомые слова - **int**, **метод println().**Из незнакомого только слово void, с ним мы разберемся чуть позже:



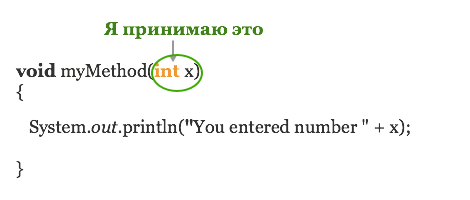
Отлично! Половина есть  Так что же все это значит?

Как Вы могли догадаться, myMethod - это название метода.



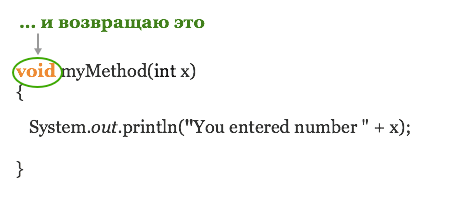
Название метода должно быть уникальным. Если название состоит из нескольких слов, напишите его, используя CamelCase (что такое "CamelCase" можно прочитать[в данной статье](https://vertex-academy.com/tutorials/ru/chto-takoe-camelstyle/)).

То, что написано справа от метода в круглых скобочках - это то, что метод принимает. Иными словами - это значения, которые мы можем использовать внутри метода.

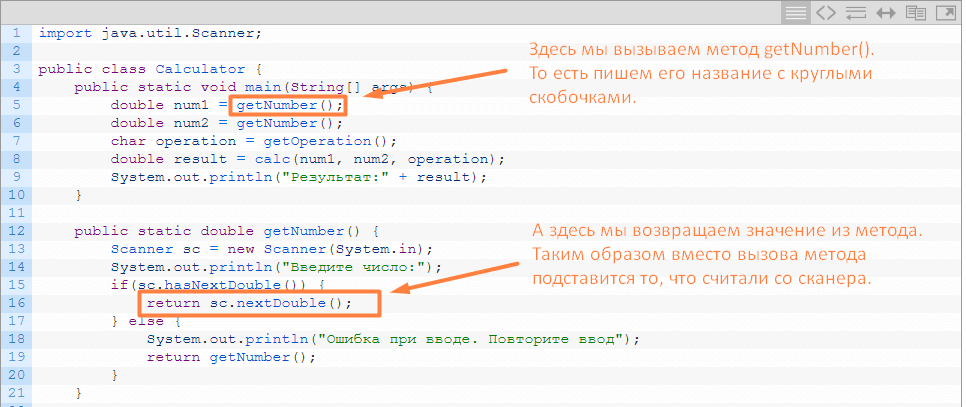


Таким образом для того, чтобы метод выполнился, ему нужно в скобочках записать число (в терминологии методов это называется "передать значение в метод") типа int. Таким образом оно автоматически подставится в переменную х и будет доступно внутри метода.  Кстати, то, что метод принимает, в данном случае **int x**, называется **параметром метода**.

То, что записано перед названием метода, то есть перед myMethod , это то, что метод возвращает.



Что значит возвращает? "Возвращает" - это значит, что после того, как метод выполнился, мы можем получить определенное число, строчку или другое значение, которое нужно использовать в том месте, **где метод вызван**.

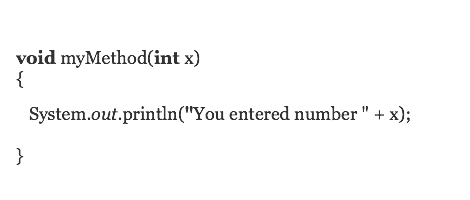


Таким образом, как видите, **перед значением, которое нужно вернуть, пишется специальное слово return**. Обратите внимание, что значение, которое возвращает метод, должно быть того же типа, который записан перед названием метода. В нашем случае double. Собственно говоря, поэтому мы считываем с консоли именно nextDouble().

А что же такое за слово void, которое записано в предыдущем примере перед названием метода? Перед названием метода записывается **слово void** (с англ. "пустота"), **если метод не должен ничего возвращать**. То есть метод выполнил код, а в место, где он вызван, он ничего не возвращает. Примером такого метода может быть метод println(), который мы используем в System.out.println(). Все, что нужно от метода, - это вывести в консоль значение. И все. Мы же ничему не присваиваем System.out.println() ? Нет. Таким образом можно отличать методы, которые что-то возвращают и "войдовские" методы:

* если вызов метода мы чему-то присваиваем - метод возвращает определенное значение
* если же просто вызываем метод в отдельной строчке - этот метод "войдовский"

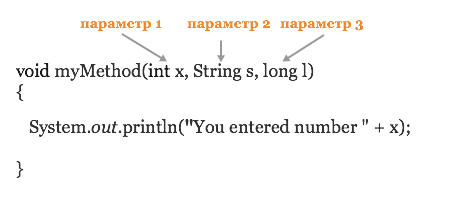
Ок, это уяснили. Вернемся к нашему простому примеру.



То есть я - метод с названием myMethod(). Я принимаю на вход какое-то число x, которое обязательно должно быть integer.

Принимать метод может и несколько переменных или другими словами параметров, но возвращать -  только одну:

Если Вы хотите передать несколько параметров методу - пишите их через запятую:



Как видите, мы передали методу myMethod() 3 параметра:

* **int x** - то есть какое-то число x, которое обязательно должно быть integer
* **String s** - какую-то строку (слово или фразу)
* **long l**  - какое-то число l, которое должно быть long типа

Помимо параметров, есть еще такое понятие как "тело метода". С этим все просто: все, что написано в фигурных  скобках {} - это **тело метода**. То есть это код, который должен выполнять метод. В данном примере в теле метода только 1 строчка кода:

* System.out.println("You enterеd number " + x);

А значит метод myMehod() всего-навсего выводит в консоль введенное пользователем число.

**На практике**

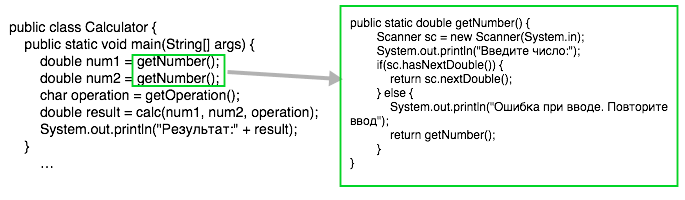
Теперь давайте вернемся к нашему примеру  - к огромному устрашающему куску кода, который мы показали чуть выше в этой статье  Сейчас мы с Вами разберем по частям из чего состоит этот "устрашающий код" и тогда он покажется совсем не страшным.

Что делает каждый метод?

Начнем с метода**main()**, который задает общую логику нашей программы:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class Calculator {      public static void main(String[] args) {          double num1 = getNumber();          double num2 = getNumber();          char operation = getOperation();          double result = calc(num1, num2, operation);          System.out.println("Результат:" + result);      } |

* Ниже справа в зеленой рамочке мы показали из чего состоит сам метод getNumber() - вся его логика. А уже в главном методе **main()** мы дважды вызываем метод **getNumber()**, чтоб получить от пользователя 2 введенных числа. Первое введенное пользователем число мы записываем в переменную **num1**, второе введенное пользователем число мы записываем в переменную **num2.**Метод **getNumber()** возвращает число типа **double**, потому что перед названием метода getNumber стоит double. Именно поэтому переменные num1 и num2 тоже имеют тип double.

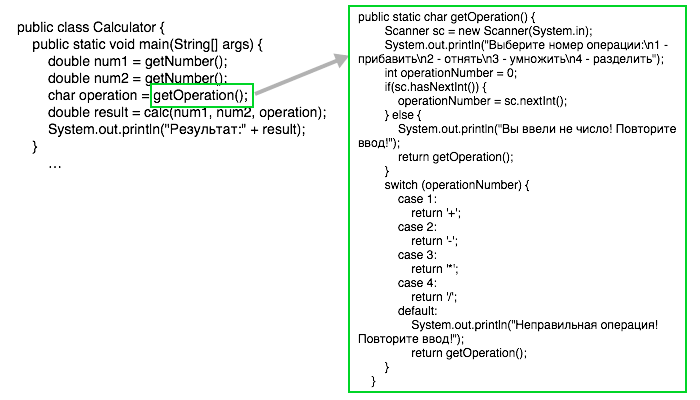


Что мы видим в этом методе?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public static double getNumber() {          Scanner sc = new Scanner(System.in);          System.out.println("Введите число:");          if(sc.hasNextDouble()) {              return sc.nextDouble();          } else {              System.out.println("Ошибка при вводе. Повторите ввод");              return getNumber();          }  } |

Вы все тут знаете: мы создаем **Scanner**, и считываем с консоли число. Если пользователь вводит не число, мы выводим сообщение "Ошибка при вводе. Повторите ввод", и заново вызываем этот же метод (т.е. заново просим пользователя ввести число).

* Дальше вызываем метод **getOperation()**. Он ничего не принимает, но возвращает **char**, соответствующий операции, которую надо провести. Тоже всю логику метода **getOperation()** мы прописали отдельно, а уже в главном методе **main()** мы вызываем метод **getOperation()**:



Как видите, мы опять вызываем **Scanner** и просим пользователя ввести номер операции (1, 2, 3 или 4).

Тут у нас два "уровня защиты"  Во-первых, мы проверяем, ввел ли пользователь число:

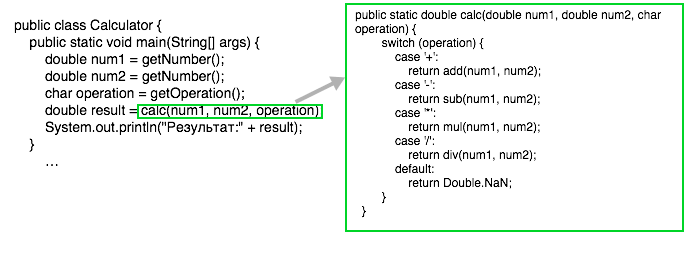
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if(sc.hasNextInt()) {      operationNumber = sc.nextInt();  } else {      System.out.println("Вы ввели не число! Повторите ввод!");      return getOperation();  } |

Тем не менее, даже если пользователь ввел число, оно не обязательно будет от 1 до 4. Поэтому, мы используем **switch case**:

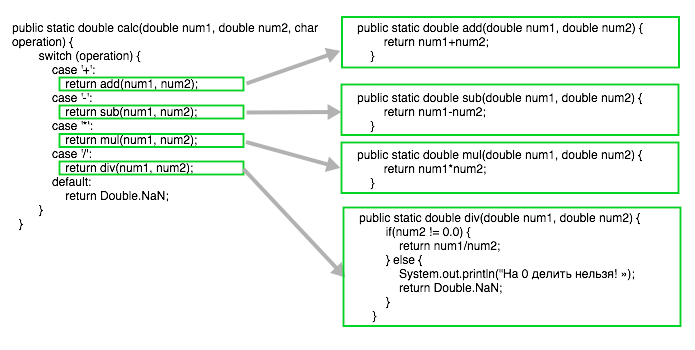
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | switch (operationNumber) {              case 1:                  return '+';              case 2:                  return '-';              case 3:                  return '\*';              case 4:                  return '/';              default:                  System.out.println("Неправильная операция! Повторите ввод!");                  return getOperation();  } |

Если мы ввели правильный код операции, метод вернет **char**. Если нет, мы заново вызываем метод **getOperation()**.

* И наконец мы вызываем метод **calc()**. Он принимает три параметра (два числа типа **double** и операцию типа **char**), и  возвращает результат типа **double**:



По сути, метод **calc()** очень простой. Он состоит только из оператора **switch case**, который "перенаправляет" на другие методы:



Фух, надеюсь теперь все понятно! Теперь попробуйте запустить наш "огромный устрашающий код" на своем компьютере, чтоб увидеть, как работают методы на практике.

**Подытожим:**

1. Вы можете **писать свои методы**. Методы могут быть **без параметров и с параметрами:**

* getNumber() - метод без параметров
* getOperation() - метод без параметров
* calc( double num1, double num2, char operation) - метод с параметрами

2. Сначала прописывается вся логика каждого метода отдельно. А потом **каждый** **метод вызывается в главном методе main()**