**Вызываем методы, возвращаем значение**

[Java Syntax](https://javarush.ru/quests/QUEST_JAVA_SYNTAX)

[Уровень 2](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_SYNTAX&level=2), Лекция 7

— Ладно, давай зайдем с другой стороны. Я тебе сейчас расскажу как работает вызов методов, а ты потом еще раз попробуешь пробежаться по предыдущей лекции, ок?

— Идет.

— Отлично, тогда я расскажу тебе о вызове функций/методов и возвращаемых ими значениях.

— Команды группируют в функции, чтобы потом можно было исполнять их единым блоком – как одну сложную команду. Для этого надо написать имя функции(метода) и в скобках после него перечислить значения-параметры.

Пример

package com.javarush.lesson2;

public class MethodCall

{

public static void main(String[] args)

{

print4(“I like to move it, move it.”);

}

public static void print4(String s)

{

System.out.println(s);

System.out.println(s);

System.out.println(s);

System.out.println(s);

}

}

— В примере выше мы написали функцию, которая выводит на экран переданную строку 4 раза. Затем мы вызвали функцию print4 в строке номер 6.

— Когда программа дойдет до выполнения строчки 6, она перескачет на строчку 9 – переменной s будет присвоено значение “I like to move it, move it.”

— Затем будут выполнены строки 11-14, и, наконец, функция завершится и программа продолжит работу со строчки номер 7.

— Ясно.

— В функцию можно не только передавать аргументы (параметры), функция еще может возвращать результат(значение) своей работы. Это делается с помощью ключевого слова return. Вот как это выглядит:

|  |  |
| --- | --- |
| Пример 1. Вычисление минимума из двух чисел.  public class MethodCall  {  public static void main(String[] args)  {  int a = 5, b = 7;  int m = min(a, b);  System.out.println("Minimum is "+ m);  }  public static int min(int c, int d)  {  int m2;  if (c < d)  m2 = c;  else  m2 = d;  **return m2;**  }  } | Вот как это работает:  public class MethodCall  {  public static void main(String[] args)  {  int a = 5, b = 7;  int c = a, d = b;  int m2;  if (c < d)  m2 = c;  else  m2 = d;  int m = m2;  System.out.println("Minimum is "+ m);  }  } |

1

Задача

Java Syntax,  2 уровень,  7 лекция

Набираем код Ӏ Java Syntax: 2 уровень, 7 лекция

Java Syntax: 2 уровень, 7 лекция. Иногда думать не надо, строчить надо! Как ни парадоксально звучит, порой пальцы «запоминают» лучше, чем сознание. Вот почему во время обучения в секретном центре JavaRush вы иногда встречаете задания на набор кода. Набирая код, вы привыкаете к синтаксису и зарабатываете немного материи. А ещё — боретесь с ленью.

— Похоже, начинаю понимать. Слева и справа написан один и тот же код. Просто слева он вынесен в отдельную функцию.

— Функция вычисляет какое-то значение и отдает его тем, кто ее вызвал с помощью команды return. По крайней мере, мне так кажется.

— В принципе верно.

— А что еще за тип void такой?

— Некоторые функции просто что-то делают, но никаких значений не вычисляют и не возвращают, как наш метод **main**(), например. Для них придуман специальный тип результата – **void** – пустой тип.

— А почему нельзя было просто ничего не указывать, раз функция ничего не возвращает?

— Вспомни, как объявляется любая переменная – «тип и имя». А функция – «тип, имя и круглые скобки». А имя функции и затем круглые скобки – это вызов функции!

— Т.е. было проще придумать «пустой тип», чем разделять функции на две категории – возвращающие значение и не возвращающие значение?

— Именно! Ты отлично соображаешь, мой мальчик.

— А как возвращать пустой тип?

— Никак. Вот как все это работает: когда Java-машина выполняет команду return, она вычисляет значение выражения, стоящего справа от слова return, сохраняет это значение в специальной части памяти и **тут же завершает работу функции.** А сохранённое значение использует как результат вызова функции в том месте, где её вызвали. Ты можешь увидеть это на примере вверху.

— Ты говоришь про то место, где int m = min(a,b) трансформировалось в m=m2?

— Да. После вызова функции всё продолжает работать так, как будто вместо неё на этом же месте был написан её результат. Прочитай эту фразу еще раз и посмотри на код последнего примера.

— По-моему это только кажется лёгким, а на самом деле – сложно. Я только чуть-чуть чего-то понял и всё.

— Ничего. С первого раза можно понять только то, что уже знаешь. Чем больше не понятно, тем сильнее ты влез в новую для тебя область. И тем круче будет результат. Со временем всё прояснится.

— Ну, если ты так говоришь, тогда поехали дальше.