Вызываем методы, возвращаем значение

Java Syntax
2 уровень, 7 лекция

ОТКРЫТА

— 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

```
1  package com.javarush.lesson2;
2  public class MethodCall
3  {
4      public static void main(String[] args)
5      {
6           print4(“I like to move it, move it.”);
7      }
8
9      public static void print4(String s)
10     {
11         System.out.println(s);
12         System.out.println(s);
13         System.out.println(s);
14         System.out.println(s);
15     }
16 }
```

— 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.	Вот как это работает:

```java
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;
   }
}
```

```java
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);
   }
}
```

×2

Задача T Java Syntax, 2 уровень, 7 лекция

ДОСТУПНА

★★½☆☆

Набираем код

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.

Открыть

— 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.

Предыдущая

×2

+132

Комментарии (189)

популярные **новые** **старые**

Vitaly Morochenets

Введите текст комментария

Pervaya Lyubov 2 уровень, Москва
воскресенье, 21:58

Хах, прочитал сначала "с первого раза можно понять только js" (у меня просто на экране грязь какая-то была)

Ответить 0

tomce26 3 уровень
19 октября, 15:03

pervyi primer idea dazhe neispolniajet. min imejet neskolko variantov i v stroke "public static int min(int c, int d)" polno neuviazok

Ответить 0

Dragon777 10 уровень
5 октября, 21:20

почему бы просто вместо m не поставить m2 все равно работает и понятно

Ответить 0

iviole 4 уровень
9 октября, 15:44

Если в первом случае поставить m2 вместо m, то нужно будет делать на него ссылку в классе main + менять все переменные(в классе main) на c и d и делать на них ссылки в класс main перед основным кодом, по итогу код будет раза в два длиннее.
Во втором случае это сделано чтобы было легче сопоставить с первым примером.

Ответить 0

Alexander Brilliantov 5 уровень, Санкт-Петербург