

Методы/функции и строки

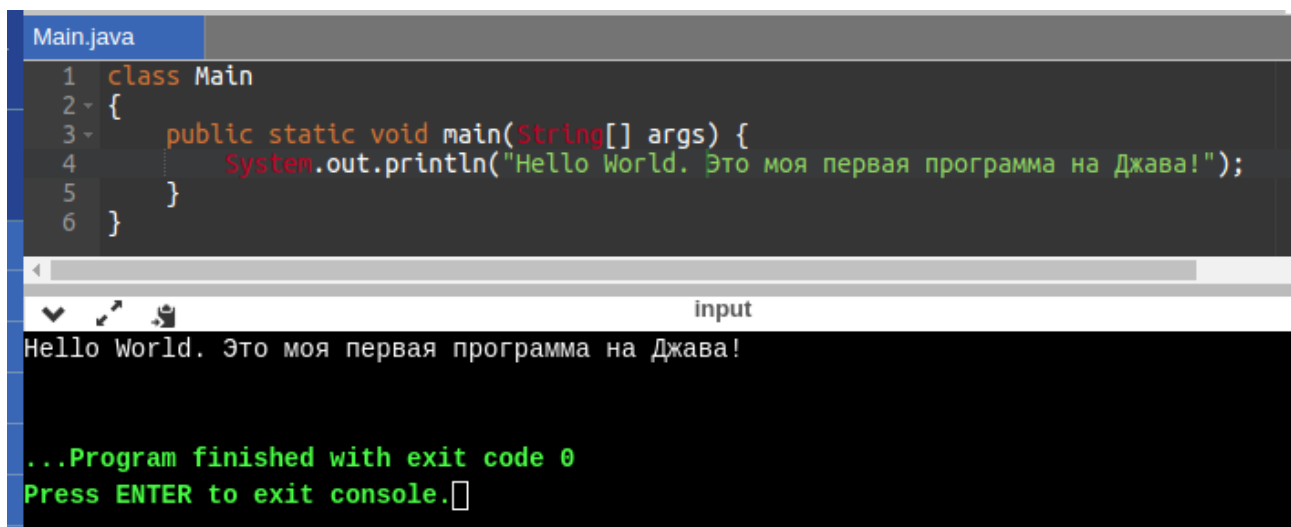
Код, который что-то делает

Содержание

1. Исполняемая функция и исполнение функций
2. Истинное предназначение функций, аргументы

1. Исполняемая функция и исполнение функций

В предыдущей лекции мы рассмотрели структуру программы – файл, имя файла, имя класса и исполняемая функция. Мы уже поняли что в файле должен быть класс с тем же именем и давайте пока оставим тему классов на потом и рассмотрим функции (они же методы).



```
Main.java
1 class Main
2 {
3     public static void main(String[] args) {
4         System.out.println("Hello World. Это моя первая программа на Джава!");
5     }
6 }
```

input

Hello World. Это моя первая программа на Джава!

...Program finished with exit code 0
Press ENTER to exit console.

Мы уже выяснили, что менять в исполняемой функции ничего нельзя (мы про `public static void main(String[] args) {}`). И мы знаем, что класс начинается с имени и фигурной скобки и заканчивается закрывающейся фигурной скобкой. В данном случае между линиями 2 и 6 может быть что угодно. Например... исполняемая функция. А еще? А что если мы напишем похожую функцию? Давайте попробуем.

Я решил сразу убрать ключевые слова `public static` и аргументы функции как в мейне. После запуска мы видим что ошибок нет и на экране все тот же Hello World. Почему же так произошло? Мы уже говорили, что код будет выполняться только в методе мейн, потому что только он исполняется. т.е. все остальное будет просто лежать там мертвым грузом. Но как мы можем сделать так, чтобы другие функции тоже выполнялись? Давайте подумаем логически. Если мы точно знаем, что выполняется мейн метод, т.е. все что там будет написано будет исполняться, то и следовательно нужно как-то вызвать методы там, по аналогии с `System.out.println()`. Ведь по сути чем отличается эта функция от нашей? Ничем.

```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6
7     void show() {
8         System.out.println("Это моя первая программа на Джава");
9     }
10 }
```

input

Hello World

...Program finished with exit code 0
Press ENTER to exit console.

Кстати, я не убрал ключевое слово `void`, но вы можете это сделать и увидите ошибку о том, что функция должна иметь возвращаемый тип. Что это такое мы поговорим позже, когда пройдем типы. А для старта просто считайте что `void` означает функция, которая что-то делает. Теперь, давайте по аналогии с `System.out.println` вызовем нашу функцию.

```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5         show();
6     }
7
8     void show() {
9         System.out.println("Это моя первая программа на Джава");
10    }
11 }
```

input stderr

Compilation failed due to following error(s).

Main.java:5: error: non-static method show() cannot be referenced from a static context
 show();
 ^
1 error

И конечно же получаем ошибку. В ней говорится, что мы не можем вызывать (т.е. использовать) нестатичную функцию внутри статичной. Т.е. если говорить простыми словами, так как наша мейн функция имеет в себе ключевое слово `static`, то и все другие функции должны быть так же `static` (это очень простое объяснение которое не должно вас сейчас грузить, а дать минимальное понимание для того чтобы уметь использовать его по назначению, позже мы подробно остановимся на этом слове). Следовательно, нам нужно добавить в объявление нашего метода слово `static` по аналогии с мейн. Попробуем.

```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5         show();
6     }
7
8     static void show() {
9         System.out.println("Это моя первая программа на Джава");
10    }
11 }
```

input

Hello World
Это моя первая программа на Джава

...Program finished with exit code 0
Press ENTER to exit console.

Давайте подробно остановимся на этом результате. Мы говорили что наш компилятор читает код сверху вниз, так? Да, это так. Сначала он ищет наш мейн метод. Когда находит линию 3, то он идет внутрь, то есть читает линию 4. Там у нас вывод Hello World. Мы видим в консоли это. А что же дальше? Дальше он идет на линию 5 и там видит вызов функции show(). После этого компилятор ищет наш метод show и находит его на линии 8 и так как компилятор выполняет все, что написано внутри метода мейн, то он и должен выполнить вызов функции show(). Следовательно читает код на линии 9 и выводит вторую строку. Теперь мы можем добавить еще код в метод show и он тоже будет исполняться до тех пор пока мы вызываем его в методе мейн. Вроде несложно, правда? Ведь мы могли бы не создавать метод show и просто выводить на экран 2 строки друг за другом, например так.

```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5         System.out.println("Это моя первая программа на Джава");
6     }
7 }
```

input

Hello World
Это моя первая программа на Джава

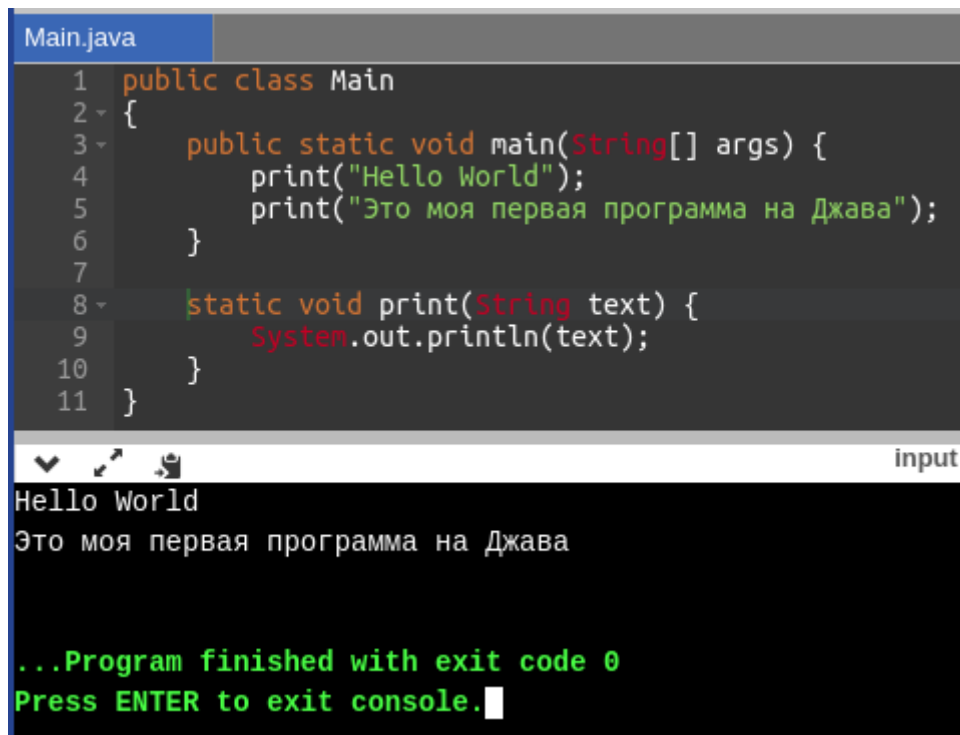
...Program finished with exit code 0
Press ENTER to exit console.

Согласитесь, что нет никакой разницы между этим кодом и кодом, где мы написали новую функцию show. Ведь результат не поменялся, мы имеем 2 строки в консоли. Тогда зачем мы создали наш метод show? Согласен, особого толку в нем нет, кроме того, что мы показали как можно создавать новые функции и вызывать их в мейн методе.

2. Истинное предназначение функций, аргументы

Итак, мы поняли что функции/методы это код, который что-то делает. А теперь внимательно посмотрите на последний скриншот. Что мы видим? Мы видим что 2 раза вызывается длинная функция `System.out.println()`. Т.е. если бы нам нужно было вывести не 2 строки на экран, а 20, то мы бы писали 20 раз `System.out.println`? Это было бы в тягость, согласны? Так в чем же истинное предназначение функций? А в том, чтобы не только бездумно выполнять код, а упрощать жизнь разработчику/программисту. И давайте зададим себе такой вопрос – а мы не могли написать такой метод, который бы помог нам не писать это длинное `System.out.println()`? И здесь мы познакомимся подробнее с такой вещью как аргумент функции. Если посмотреть внимательно, то в мейне есть аргумент функции – `String[] args`. `String` это класс строки. Квадратные скобки означают массив, но о них мы поговорим чуть позже. И `args` просто имя аргумента мейн функции, просто сокращение от аргументы. Ранее мы написали метод `show()` и внутри круглых скобок ничего не было. т.е. мы вызывали наш метод `show` и не передавали в него никакие аргументы. Потому что наш метод делал лишь одно действие – выводил на экран строку “Это моя первая программа на Джава”.

Теперь, можем ли мы написать такой метод, который бы выводил на экран строку, которую мы передадим в аргументе? Давайте попробуем.



```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         print("Hello World");
5         print("Это моя первая программа на Джава");
6     }
7
8     static void print(String text) {
9         System.out.println(text);
10    }
11 }
```

input

Hello World
Это моя первая программа на Джава

...Program finished with exit code 0
Press ENTER to exit console.

Давайте подробно остановимся на линии 8. Мы создали метод `print(String text)` который принимает аргумент некий текст строку и что же он с ней делает? Выводит на экран. Т.е. когда мы будем вызывать наш метод `print` в мейн методе, мы должны будем передать

аргументом некий текст/строку. А давайте для интереса вызовем наш метод print без аргумента.

```
1 public class Main
2 {
3     public static void main(String[] args) {
4         print("Hello World");
5         print("Это моя первая программа на Джава");
6         print();
7     }
8
9     static void print(String text) {
10        System.out.println(text);
11    }
12 }
```

input

Compilation failed due to following error(s).

```
Main.java:6: error: method print in class Main cannot be applied to given types;
    print();
    ^
required: String
found: no arguments
reason: actual and formal argument lists differ in length
1 error
```

И конечно же мы получили ошибку компиляции. Ведь в объявлении метода print было указано что он ожидает аргумент строку, но по факту он ничего не получил – found: no arguments. Т.е. мы не можем вызвать наш метод так, как нам хочется. Ведь тогда если мы не передадим никакую строку в метод print, то он не будет знать что делать в этом случае. Так как его единственная задача вывести на экран эту строку. Кстати, не путайте пожалуйста System.out.println и print. Первое это системный метод вывода на экран, а второе наш вспомогательный метод, который упрощает нам жизнь. Теперь, мы можем починить наш код (пофиксить баг/ошибку) просто передав в третий вызов функции print какую-нибудь строку. И здесь у нас встанет вопрос о том, что мы называем строкой. Строка (String) в языке Java это все, что обрамлено с 2 сторон двойными кавычками. Но что если между ними ничего не будет? Давайте посмотрим.

```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         print("Hello World");
5         print("Это моя первая программа на Джава");
6         print("");
7     }
8
9     static void print(String text) {
10        System.out.println(text);
11    }
12 }
```

input

Hello World
Это моя первая программа на Джава

...Program finished with exit code 0
Press ENTER to exit console.□

Это называется пустая строка. Когда между двойными кавычками просто ничего нет. Попробуйте переместить вызов функции `print` на линии 6 между 4 и 5 и вы увидите на экране пустую строку между первой и второй. Кстати, так как пустая строка имеет право на существование, то мы часто будем проверять на пустоту любую строку перед тем как ее использовать. Зачем? А чтобы не возникало таких ситуаций, когда мы в консоль пишем пустоту. Ведь зачем лишний раз выполнять то, что не имеет значения? Но об этом мы поговорим в следующих лекциях. А пока постарайтесь разобраться в методах и аргументах.

И еще раз хотелось бы упомянуть нашу методику. Не ждите пока вам дадут информацию на блюдечке. Пробуйте сами понять то или иное. Можете гуглить, но вместо гугла вам многое скажет компилятор. Пробуйте вносить изменения. Ставьте вопросы например такие – а я не могу написать метод не после мейна, а до? А что будет если я напишу один и тот же код 2 раза. Просто 2 одинаковых метода `print` с теми же аргументами. Пробуйте и смотрите что вам ответит компилятор. Так вы многое узнаете и запомните, так как это будет закреплено личным опытом.