

Область видимости переменной

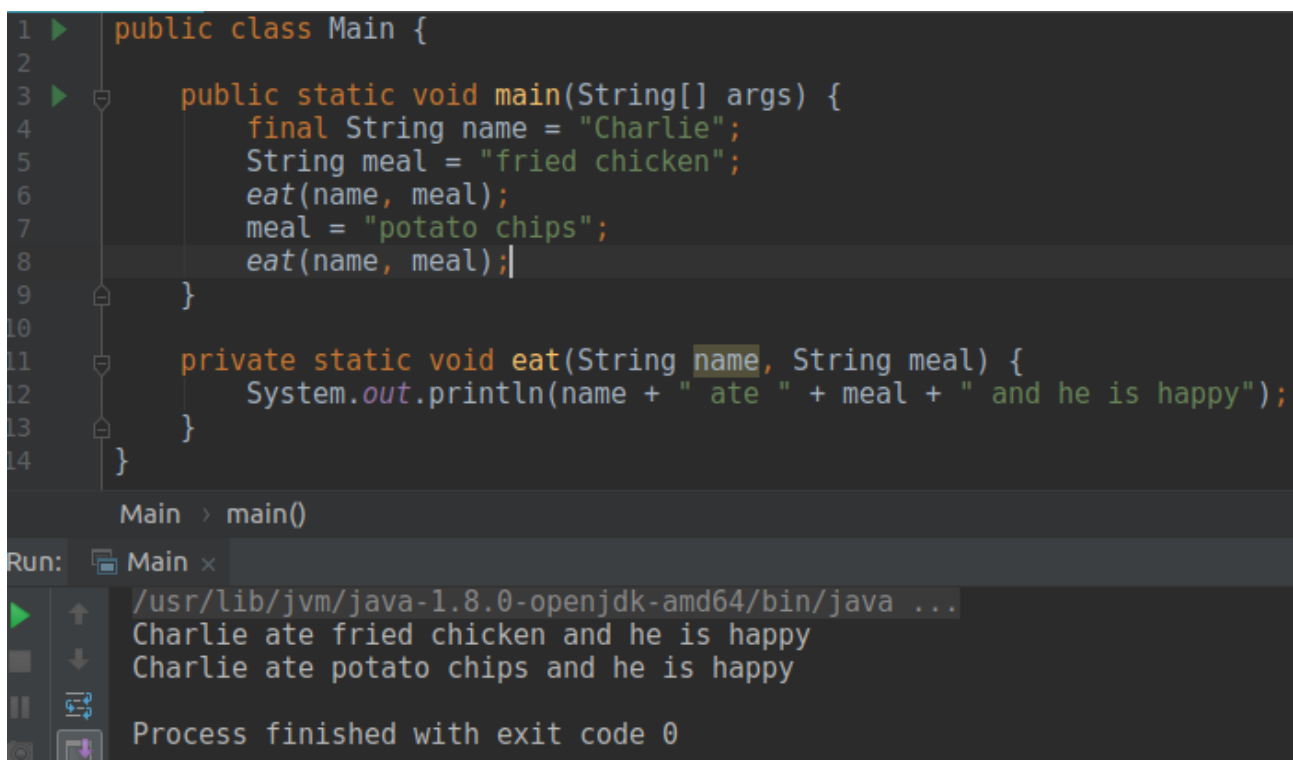
Где доступна переменная. Еще 2 типа

Содержание

1. Переменные внутри метода и вне
2. Еще 2 типа переменных

1. Переменные внутри метода и вне

Для начала давайте сделаем задание из предыдущей лекции



```
1 public class Main {
2
3     public static void main(String[] args) {
4         final String name = "Charlie";
5         String meal = "fried chicken";
6         eat(name, meal);
7         meal = "potato chips";
8         eat(name, meal);
9     }
10
11     private static void eat(String name, String meal) {
12         System.out.println(name + " ate " + meal + " and he is happy");
13     }
14 }
```

Main > main()

Run: Main x

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Charlie ate fried chicken and he is happy
Charlie ate potato chips and he is happy

Process finished with exit code 0
```

Нам нужны 2 переменные и 1 функция. Так как у нас имя персонажа не меняется, то оно должно быть `final`. А еда меняется каждый раз и ее надо переопределять. Но давайте теперь посмотрим на `String name` аргумент в функции. Почему-то оно высвечено. Наставим курсор на него. И мы увидим подсказку среды разработки: Actual value of parameter name is always 'Charlie'. Т.е. мы передаем в этот метод одно и то же значение. Ведь посмотрите на линии кода 6 и 8. Они же одинаковые. Было бы классно не передавать имя каждый раз при вызове, не правда ли? Да, можно захардкодить конечно вывод строки, типа :

“Charlie ate “ + meal ...

Но ведь например у нас может быть не 1 функция, а 2. И тогда, если нам понадобится поменять имя, то придется менять его в 2 функциях уже. И это неудобно. Вот бы мы могли иметь переменную для имени, которую видно из других методов. И это можно устроить. Давайте просто вынесем нашу константу за метод мейн. Да-да, это легально. Смотрим.

```
1 public class Main {
2
3     private static final String name = "Charlie";
4
5     public static void main(String[] args) {
6         String meal = "fried chicken";
7         eat(meal);
8         meal = "potato chips";
9         eat(meal);
10    }
11
12    private static void eat(String meal) {
13        System.out.println(name + " ate " + meal + " and he is happy");
14    }
15 }

Main > eat()

Run: Main x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Charlie ate fried chicken and he is happy
Charlie ate potato chips and he is happy

Process finished with exit code 0
```

Мы просто взяли и подняли объявление и инициализацию имени над методом мейн. И опять же, чтобы суметь использовать его в статик методах, нужно чтобы оно было статик. И посмотрите теперь на вызов метода eat и на него самого. Мы используем имя в методе не передавая его в аргументы. В этом и вся суть области видимости переменной. Если она объявлена внутри метода (например мейн), то она видна и доступна только внутри этого метода, т.е. до закрывающей фигурной скобки. А если она объявлена вне методов, то и доступа в любых других методах этого класса. То же самое и про аргументы, но это слишком ясно чтобы объяснять (meal доступен только внутри метода eat). Чтобы проиллюстрировать выгоду этого решения давайте напишем еще один метод.

Теперь у нас 2 метода, которые используют константу имени. И теперь мы можем сказать что не зря вынесли эту константу из метода мейн. Ведь мы используем переменную по назначению – она упрощает нам жизнь. Но если же у нас будет 2 персонажа, то тогда нам придется передавать имя внутрь функций eat, want аргументом.

Вообще постарайтесь запомнить private static final String (int) это константа и ее нужно писать капс локом – только заглавными буквами. Такой вот джава код-стайл. Но если у вас не final переменная, т.е. не константа, то и писать ее капсом не нужно.

```
1 public class Main {
2
3     private static final String NAME = "Charlie";
4
5     public static void main(String[] args) {
6         String meal = "fried chicken";
7         want(meal);
8         eat(meal);
9         meal = "potato chips";
10        want(meal);
11        eat(meal);
12    }
13
14    private static void eat(String meal) {
15        print(NAME + " ate " + meal + " and he is happy");
16    }
17
18    private static void want(String meal) {
19        print(NAME + " wanted to eat " + meal);
20    }
21
22    private static void print(String text) {
23        System.out.println(text);
24    }
25 }
```

Main > main()

Run: Main x

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Charlie wanted to eat fried chicken
Charlie ate fried chicken and he is happy
Charlie wanted to eat potato chips
Charlie ate potato chips and he is happy
Process finished with exit code 0
```

2. Еще 2 типа переменных

Помните у нас было задание написать еще 3 метода для простых математических операций? Сейчас я бы хотел остановиться на одном из них, а точнее на делении.

Давайте посмотрим на следующий код. Итак, мы берем 2 числа и делим первое на второе. Посмотрим на результат. 10 делим на 4 и получаем 2. 5 делим на 3 и получаем 1. Эм... ну, как бы. Не совсем то, что хотелось, не правда ли? Почему же так? А быть может потому, что у нас Integer (int) это целые числа? И когда ты делишь целое число на другое целое число, то получаешь... целое число (неожиданно, правда?). Так вот, если для ваших целей это подходит, то ок. Но что делать, если вам важно знать и дробную часть деления? Тогда нам понадобится наверно другой тип переменных? Интересно, что десятичные в английском называются числа с плавающей точкой. Т.е. если в целых у тебя точка всегда находится в одном положении, то у десятичной она типа двигается, а не всегда справа от цифр. Итак, смотрим. Float.

```
1 public class Main {
2
3     public static void main(String[] args) {
4         showDivide(10, 4);
5         showDivide(5, 3);
6     }
7
8     private static void showDivide(int number1, int number2) {
9         System.out.println(number1 / number2);
10    }
11 }

Main > main()

Run: Main x /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
2
1
Process finished with exit code 0
```

```
1 public class Main {
2
3     public static void main(String[] args) {
4         showDivide(10, 4);
5         showDivide(5, 3);
6     }
7
8     private static void showDivide(int number1, float number2) {
9         System.out.println(number1 / number2);
10    }
11 }

Main > main()

Run: Main x /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
2.5
1.6666666
Process finished with exit code 0
```

Вот так просто. Берем и меняем int на float. Что самое важное, можно поменять одно из двух чисел на десятичную и все. т.е. в джава есть такая штука как переход к большему. Если ты складываешь десятичное число и целое, то ты получаешь десятичное.

И давайте посмотрим на еще одну штуку, мы же хотим иметь дело с десятичными, так давайте поделим на десятичное число.

```
public static void main(String[] args) {
    showDivide(10, 4.25);
    showDivide(5, 3);
}
```

Что-то тут не так. Среда разработки подчеркнула наш 4.25. Если поставить курсор то там будет Wrong 2nd argument. Found double, required float. Что еще за double? Это как раз второй тип переменной, который мы сегодня рассмотрим. А как же пофиксить это? Суть в том, что нам нужно писать букву f в конце числа. Что? Писать букву в конце числа, чтобы это было десятичным числом? Seriously? Ну да, согласен, неудобно. Но мы это исправим, обещаю. Пробуем. И видим что все ок.

```
1 public class Main {
2
3     public static void main(String[] args) {
4         showDivide(10, 4.25f);
5         showDivide(5, 3);
6     }
7
8     private static void showDivide(int number1, float number2) {
9         System.out.println(number1 / number2);
10    }
11 }
```

Main > main()

Run: Main x

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
2.3529413
1.6666666
Process finished with exit code 0
```

Ладно, мы сказали про второй тип, который по сути легче использовать, так зачем же нам этот float? Если можно писать десятичные и без этой лишней буквы f? Кстати, давайте проверим на деле.

```
1 public class Main {
2
3     public static void main(String[] args) {
4         showDivide(10, 4);
5         showDivide(5, 3);
6     }
7
8     private static void showDivide(int number1, double number2) {
9         System.out.println(number1 / number2);
10    }
11 }
```

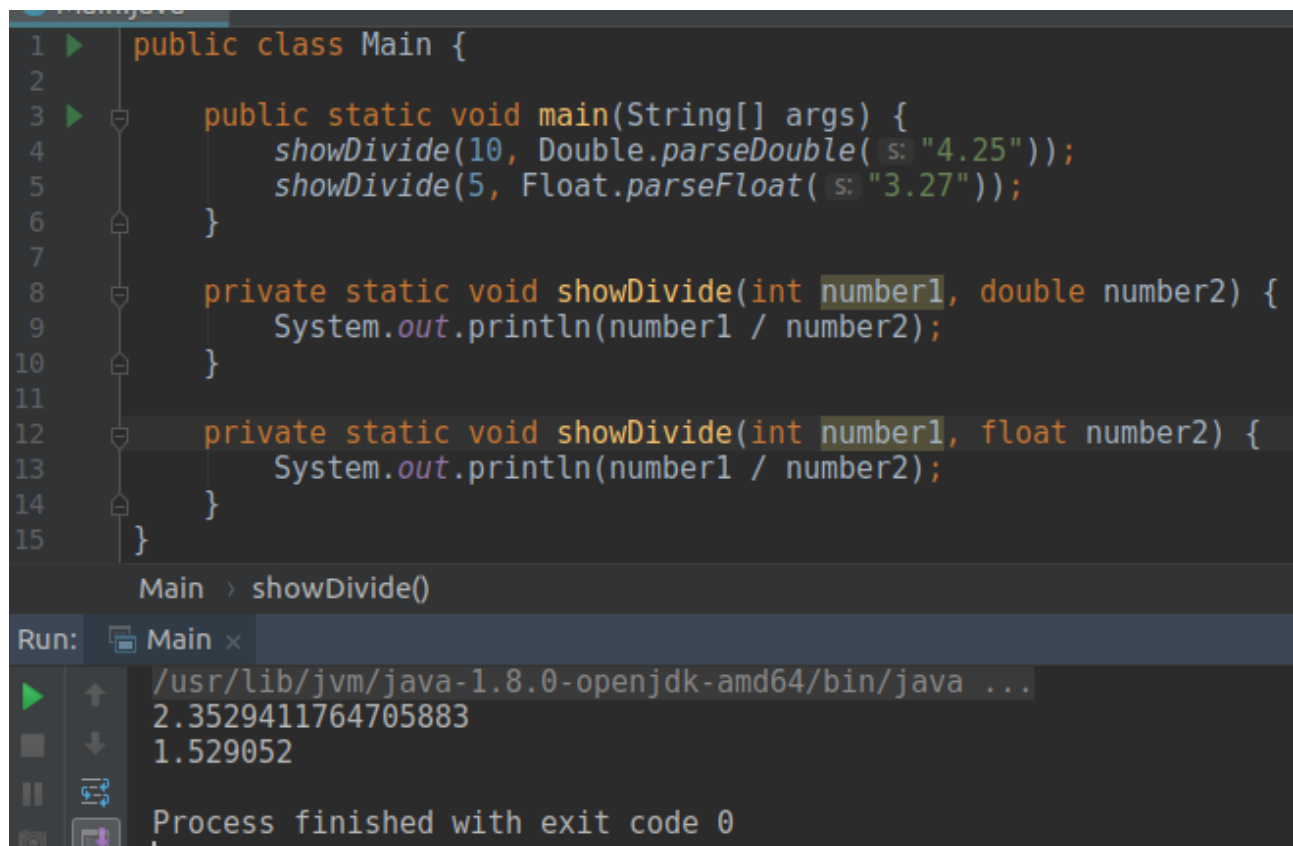
Main

Run: Main x

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
2.5
1.6666666666666667
Process finished with exit code 0
```

Ну вот, совсем другое дело: `double number2` и никаких букв не нужно. Красота же ну! А вы заметили результат деления 5 на 3 для первого случая и второго? В первом случае `float` после точки 7 цифр всего, а у `double` побольше точности. Но и цена этой точности конечно же есть. Если вы используете `double`, то вы запрашиваете в 2 раза больше места у вашего железа для этих целей. Так что в иной раз будьте внимательны к тому, какой тип переменной вы используете.

И у любого человека может возникнуть вопрос. А какие вообще значения у того или иного типа переменной? Какие значения может принимать `int`, `float`, `double`? Для этого и существуют такие классы как `Integer`, `Float`, `Double`. Помните метод, который превращал строку (`String`) в число? `Integer.parseInt(String text)`. Конечно же такие же методы есть и для преобразования десятичного числа.



```
1 public class Main {
2
3     public static void main(String[] args) {
4         showDivide(10, Double.parseDouble("4.25"));
5         showDivide(5, Float.parseFloat("3.27"));
6     }
7
8     private static void showDivide(int number1, double number2) {
9         System.out.println(number1 / number2);
10    }
11
12    private static void showDivide(int number1, float number2) {
13        System.out.println(number1 / number2);
14    }
15 }
```

Main > showDivide()

Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
2.3529411764705883
1.529052

Process finished with exit code 0

Если вы заметили, то у нас 2 метода `showDivide`, но у них разные типы второго аргумента, а значит они имеют право на существование (это называется перегрузка методов). Но вернемся к вопросу о лимитах. Какие же максимальные и минимальные значения у `int`, `float`, `double`. Посмотрим же на них.

```
1 public class Main {
2
3     public static void main(String[] args) {
4         print("Min of Int is: " + Integer.MIN_VALUE);
5         print("Max of Int is: " + Integer.MAX_VALUE);
6         print("Min of Float is: " + Float.MIN_VALUE);
7         print("Max of Float is: " + Float.MAX_VALUE);
8         print("Min of Double is: " + Double.MIN_VALUE);
9         print("Max of Double is: " + Double.MAX_VALUE);
10    }
11}
```

Main > main()

Run: Main x

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Min of Int is: -2147483648
Max of Int is: 2147483647
Min of Float is: 1.4E-45
Max of Float is: 3.4028235E38
Min of Double is: 4.9E-324
Max of Double is: 1.7976931348623157E308

Process finished with exit code 0
```

Если вы работаете с числами в пределах 2 миллиардов, то вам можно взять int. Если же у вас числа побольше, то берите float. Если уж совсем огромные, то double. Что делать в случае если титанические числа? Для этого есть свои классы, но оставляю это вам на самостоятельное гугление.

Теперь вернемся к нашей программе, которая производит действия с числами. Давайте напишем перегруженные методы для того, чтобы они работали как с int, так и для float и double. А еще удалите метод print(String) и напишите вместе него 3 других, для разных типов чисел. Ведь если ты не выводишь на экран строку, то тебе и не нужен этот метод, правильно?

Надеюсь у вас все получится, если же нет, то у нас есть возможность обсудить это в чате. В следующей лекции мы не будем разбирать это задание.

Удачи!