



Перегрузка методов | часть 2

Java Core (/quests/QUEST_JAVA_CORE)
5 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=5), 3 лекция (/quests/lectures/questcore.level05.lecture03)

ОТКРЫТА

— Привет, Амиго! Пару дней назад я тебе рассказывал о перегрузке методов. Ты все понял?

— Да. Я помню. Каждый метод класса должен быть уникальным. Метод класса уникальный, если в этом классе нет метода с таким же именем и типом параметров, где порядок параметров имеет значение.

— Отлично! Я вижу, что ты хорошо выучил тот урок. Сегодня я хочу лишь немного расширить твои познания в этом деле. Как ты думаешь, какой метод будет вызван в каждом случае?

Код

```
1 class Cat
2 {
3     public static void print(int n)
4     {
5         System.out.println(n);
6     }
7     public static void print(short n)
8     {
9         System.out.println(n);
10    }
11    public static void print(Integer n)
12    {
13        System.out.println(n);
14    }
15    public static void print(String s)
16    {
17        System.out.println(s);
18    }
19    public static void main(String[] args)
20    {
21        Cat.print(1);
22        Cat.print((byte)1);
23        Cat.print("1");
24        Cat.print(null);
25    }
26 }
```

— Затрудняюсь ответить.

— В первом случае 1 имеет тип int, у нас есть 100% совпадение метода, который принимает int. Будет вызван первый void print(int n).

Во втором случае, у нас нет метода, который принимает byte. Но есть два метода, которые принимают short и int. По стандарту расширения типов, byte сначала будет расширен до short, а уж затем расширен до int. Вердикт – будет вызван метода void print(short n).

В третьем случае у нас есть 100% совпадение метода, который принимает String. Будет вызван метод void print(String s).

В четвертом случае у нас неопределенность. null не имеет определенного типа, компилятор откажется компилировать этот код. В таком случае нужно написать Cat.print((Integer)null), чтобы вызвать третий метод и Cat.print((String)null), чтобы вызвать четвертый.

— Очень познавательно, спасибо.

— Обращаю твое внимание, что в процессе определения метода, который нужно вызвать, типы могут только расширяться, но не сужаться. Пример:

Код

```
1 class Cat
2 {
3     public static void print(short n)
4     {
5         System.out.println(n);
6     }
7     public static void print(Integer n)
8     {
9         System.out.println(n);
10    }
11
12    public static void main(String[] args)
13    {
14        Cat.print((byte)1);
15        Cat.print(1);
16    }
17 }
```

В первом случае, тип byte будет расширен до short и произойдет вызов первого метода: void print(short n).

Во втором случае неявно будет выполнено разрешенное преобразование от int к Integer, и произойдет вызов второго метода void print(Integer n).

— Неожиданно.

— Нет, неожиданно – это тут:

Код на Java

Описание

```
1 class Cat
2 {
3     public static void print(Object o)
4     {
5         System.out.println(o);
6     }
7     public static void print(String s)
8     {
9         System.out.println(s);
10    }
11
12    public static void main(String[] args)
13    {
14        Cat.print(1);
15        Cat.print(null);
16    }
17 }
```

В первом случае int будет расширен до Integer, а так как нет метода для Integer, то вызовется наиболее подходящий метод, т.е. метод void print(Object o)

Во втором случае, ошибки компиляции не будет и вызовется метод void print(String s), что несколько не очевидно.

Надеюсь, Амиго ты понял, что лучше всего в таких случаях указать оператор преобразования типа, как в случае с (byte), чтобы точно знать, какой метод вызовется.

— Уж от чего, от чего, а от перегрузки методов я никаких проблем не ожидал. И тут – на тебе. Спасибо, Риша, буду держать ухо востро и не расслабляться.

[< \(/quests/lectures/questcore.level05.lecture02\)](/quests/lectures/questcore.level05.lecture02)[×15 > \(/quests/lectures/questcore.level05.lecture04\)](/quests/lectures/questcore.level05.lecture04)

 0  0  0  0  0 +18



Комментарии (55)

популярные

новые

старые

Никита

[javarush.ru/](#), **G+** [_\(/https://plus.google.com/114772402300089087607\)](https://plus.google.com/114772402300089087607),  [_\(/https://twitter.com/javarush_ru\)](https://twitter.com/javarush_ru), 



Программистами не рождаются
© 2018