**Перегрузка методов | часть 2**

[Java Core](https://javarush.ru/quests/QUEST_JAVA_CORE)

[Уровень 5](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=5), Лекция 3

— 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

class Cat

{

public static void print(int n)

{

System.out.println(n);

}

public static void print(short n)

{

System.out.println(n);

}

public static void print(Integer n)

{

System.out.println(n);

}

public static void print(String s)

{

System.out.println(s);

}

public static void main(String[] args)

{

Cat.print(1);

Cat.print((byte)1);

Cat.print("1");

Cat.print(null);

}

}

— 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 метод **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

class Cat

{

public static void print(short n)

{

System.out.println(n);

}

public static void print(Integer n)

{

System.out.println(n);

}

public static void main(String[] args)

{

Cat.print((byte)1);

Cat.print(1);

}

}

В первом случает, 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** |
| class Cat  {  public static void print(Object o)  {  System.out.println(o);  }  public static void print(String s)  {  System.out.println(s);  }  public static void main(String[] args)  {  Cat.print(1);  Cat.print(null);  }  } | 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.