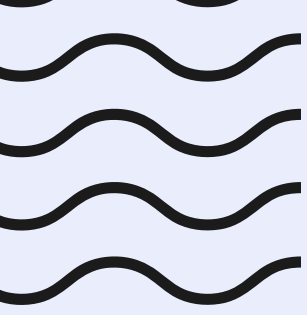


# Varargs



ПОДГОТОВИЛ: АЛИШЕР ХАМИДОВ



# План лекции

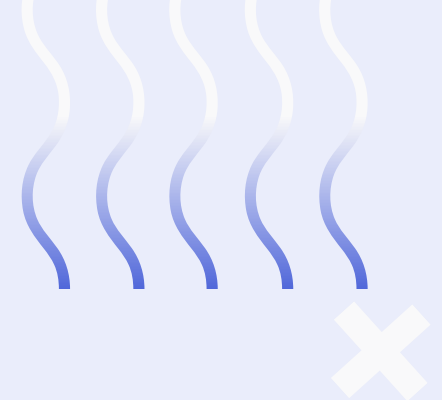

## Основные моменты:

1. Повторение понятия перегрузки методов
2. Varargs
3. Правила varargs



# Method overloading

В программе мы можем использовать методы с одним и тем же именем, но с разными типами и/или количеством параметров. Такой механизм называется перегрузкой методов (method overloading).



Стоит отметить, что на перегрузку методов влияют количество и типы параметров. Однако различие в типе возвращаемого значения для перегрузки **не имеют никакого значения.**

- (нельзя делать методы отличающиеся только типом возвр. значения).



**Допустим, что мы знаем, что в наш метод может прийти от одного до трех параметров.**

**Как нам сделать, чтобы программа не сломалась?**



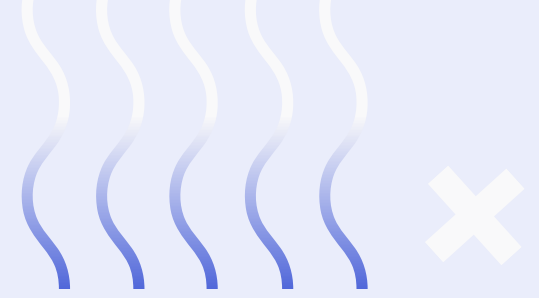


Допустим, что мы знаем, что в наш метод может прийти от одного до трех параметров.

**Как нам сделать, чтобы программа не ломалась?**

**Например, мы можем перегрузить наш метод:**

```
print(String a){...}  
○ print(String a, String b){...}  
print(String a, String b, String c){...}
```



А если в метод может быть передано совсем **произвольное** число параметров, например до 1000?

Неужели мы будем переопределять тысячу раз?

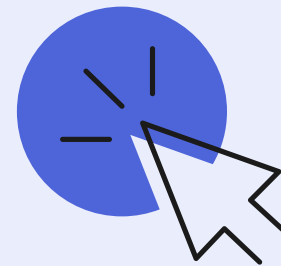
```
print(String a){...}
```

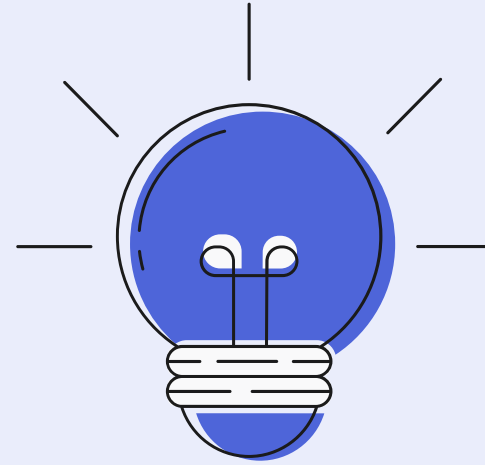
```
print(String a, String b){...}
```

```
print(String a, String b, String c){...}
```

```
...
```

```
// и так далее еще 997 раз!!!
```





К счастью, начиная с Java 5 можно использовать **Varargs**,  
которые обеспечивают синтаксис для методов с  
произвольным числом параметров одного типа







```
public void print(String... values) {  
    //  
}
```

- метод будет работать для произвольного числа аргументов указанного типа, в данном случае **String**
- с **values** мы можем работать как с обычным массивом.

```
public void print(String... values) {  
    for(String s: values){  
        System.out.println(s);  
    }  
}
```

Пример вызова этого метода

```
print("a", "b", "c"); // выведет в консоль: a b c
```

# Важные правила

## Только один

В одном методе может  
быть только один `varargs`  
параметр

## Стоит на последнем месте

Параметр `varargs` должен стоять  
после всех остальных  
параметров метода

