

Методы. Часть 2

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Получить представление о методах, возвращающих значение

ПЛАН ЗАНЯТИЯ

■ Методы, возвращающие значения

■ Сигнатура метода

■ Вызов метода

■ Перегрузка методов

■ `varargs`

■ Передача аргументов в формальные параметры (для ссылочных и примитивных типов)

МЕТОДЫ, ВОЗВРАЩАЮЩИЕ ЗНАЧЕНИЯ

Методы, способные **возвращать значения** могут иметь **единственный и явный результат** своей работы - например число, строка, массив и т.д.

```
int[] a = {7, 2, -5, 11, 1};  
int sumA = 0;  
  
for (int i = 0; i < a.length; i++) {  
    sumA = sumA + a[i];  
}  
  
int[] b = {5, 1, 2, 13, 4};  
int sumB = 0;  
  
for (int i = 0; i < b.length; i++) {  
    sumB = sumB + b[i];  
}
```

```
public static int getSum(int x[]) {  
    int result = 0;  
    for (int i = 0; i < x.length; i++) {  
        result = result + x[i];  
    }  
  
    return result;  
}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Метод

Обоснования для создания методов

СИГНАТУРА МЕТОДА

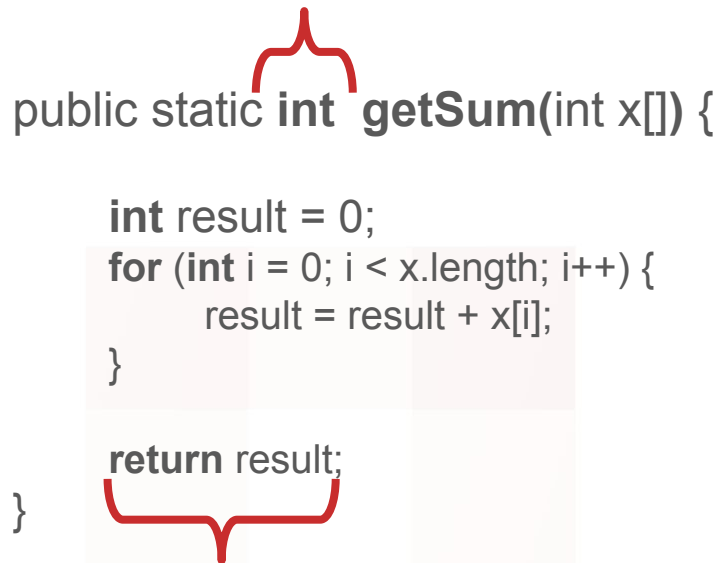
Методы, возвращающие значения, имеют сигнатуру, аналогичную void-методам. Исключением является наличие **типа возвращаемого значения** и обязательного оператора **return** в теле

Тип возвращаемого значения, указанный в сигнатуре метода, **обязательно** должен совпадать с типом значения, указанным после оператора **return**

Не допускаются ситуации, когда какая-либо из веток кода метода не приводит к выполнению оператора **return**

Тип возвращаемого значения важен для строгой проверки типов во время компиляции, предотвращая ошибки, связанные с неправильным вызовом метода и присваивания его результата переменной другого типа

тип возвращаемого
значения



```
public static int getSum(int x[]) {  
  
    int result = 0;  
    for (int i = 0; i < x.length; i++) {  
        result = result + x[i];  
    }  
  
    return result;  
}
```

возврат результата

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

■
Название метода

■
Параметры

■
Тело

ВЫЗОВ МЕТОДА

Вызов метода, возвращающего значение, аналогичен void-методу. Но главным отличием при этом является возможность присвоить результат работы метода переменной или подставить вызов метода в конструкцию, ожидающую какого-либо значения

```
int[] a = {7, 2, -5, 11, 1};
```

```
int sumA = getSum(a);
```

```
int[] b = {5, 1, 2, 13, 4};
```

```
int sumB = getSum(b);
```

Таким образом, такие методы возвращают результат вычислений в то место кода, где этот метод был вызван.

```
public static int getSum(int[] x) {  
    // ...  
    return result;  
}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Вызов метода



Аргументы и параметры

ПЕРЕГРУЗКА МЕТОДОВ

В Java имеется возможность создания методов, имеющих одинаковое название, но разный список формальных параметров (отличающихся типами и/или количеством). Такие методы называются **перегруженными**

Это делает код более интуитивно понятным, так как одно и то же имя метода используется для схожих операций.

Нельзя создавать методы с одинаковым названием и параметрами, но разными возвращаемыми значениями. Также нельзя создавать методы с одинаковыми именами, одинаковым порядком и типом параметров, но имеющих разные названия параметров. Во всех этих случаях Java не сможет правильно выбрать нужный метод при вызове

// сумма элементов массива

```
public static int getSum(int x[]) { ... }
```

// сумма элементов массива в промежутке

```
public static int getSum(int x[], int from, int to) { ... }
```

// сумма чисел в промежутке

```
public static int getSum(int from, int to) { ... }
```

// ошибка компиляции

```
public static int getSum(int from, int to) { ... }  
public static double getSum(int from, int to) { ... }
```

// ошибка компиляции

```
public static int getSum(int from, int to) { ... }  
public static int getSum(int a, int b) { ... }
```

varargs

Механизм **variable number of arguments** позволяет методу принимать переменное количество аргументов. Это удобно, когда количество аргументов заранее неизвестно.

Внутри метода `varargs` ведет себя как массив.

Метод с `varargs` можно вызывать с нулевым, одним или несколькими аргументами

Только один `vararg` параметр может быть указан в объявлении метода, и он должен находиться в конце списка параметров.

// сумма чисел

```
public static int getSum(int ... numbers) {  
    int result = 0;  
    for (int i = 0; i < numbers.length; i++) {  
        result = result + numbers[i];  
    }  
  
    return result;  
}
```

// ВЫЗОВЫ

```
int a = getSum(); // a == 0  
int b = getSum(5); // b == 5  
int c = getSum(1, 2); // c == 3
```

ПЕРЕДАЧА АРГУМЕНТОВ В ПАРАМЕТРЫ

Важно понимать разницу между передачей аргументов примитивных типов и аргументов ссылочных типов при вызове метода

В обоих случаях происходит копирование **значения** аргумента (**передача по значению**). В случае примитивного типа копируется непосредственно значение аргумента, в случае ссылочного типа этим значением аргумента является **ссылка** на объект в памяти.

```
... void change(int x) { // прием примитива  
    x = 7;  
}
```

```
... void change(int[] y) { // прием ссылки  
    y[0] = 7;  
}
```

```
int a = 5;  
int[] b = {4, 5, 6};
```

```
change(a); // значение a не будет изменено  
change(b); // b[0] будет иметь значение 7
```

ПЕРЕДАЧА АРГУМЕНТОВ В ПАРАМЕТРЫ

```
... void change(int x) { // прием примитива  
    x = 7;  
}
```

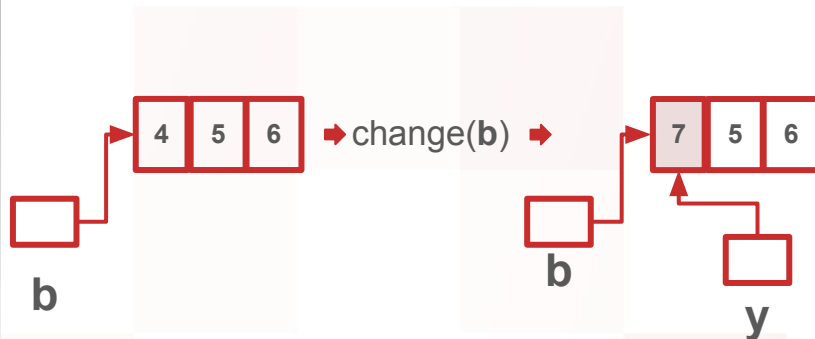
```
... void change(int[] y) { // прием ссылки  
    y[0] = 7;  
}
```

```
int a = 5;  
int[] b = {4, 5, 6};
```

```
change(a); // значение a не будет изменено  
change(b); // b[0] будет иметь значение 7
```



a и **x** - независимые переменные



b и **y** ссылаются на один и тот же массив

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Вызов метода



Аргументы и параметры

ПОИГРАЕМ ;)

Метод

Вызов метода

Сигнатура метода

Аргументы и параметры

ДОМАШНЕЕ ЗАДАНИЕ

1. Доделайте прошлые домашние задания
2. В CurrencyConverter вынесите switch в котором определяете нужное значение курса выбранной валюты в отдельный метод и получайте значение курса из него, передавая в метод в качестве аргумента текущую валюту, массив значений курсов. Сигнатура должна быть такой:
`public static double getRateValue(String currency, double[] rates)`
3. Порешайте задачи на странице <https://w3schools.com/java/exercise.asp> раздел Java Methods (задания 4, 5)

По желанию: полностью декомпозировать main метод, выделив повторяющиеся или логически выделяемые блоки кода в отдельные методы

По желанию: читать курсы валют из файла currency-rates.json, структуру содержимого можете задать сами, но она должна представлять из себя корректный json формат



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH