

# Методы. Часть 2.

Методы в Java, возвращающие значения, обеспечивают возможность получения данных после выполнения определенных операций. Эти методы могут возвращать различные типы данных, такие как числа, строки, массивы и объекты.

```
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;  
}
```

В отличие от void методов, методы, возвращающие значения, должны указывать конкретный тип данных, который они возвращают. Этот тип данных указывается перед именем метода в его сигнатуре. Например, метод `int getNumber()` возвращает целое число (`int`).

В теле такого метода обязательно должен быть использован оператор **return**, за которым следует возвращаемое значение. Это значение должно соответствовать объявленному типу возвращаемого значения. Например, если метод обещает вернуть `int`, то после `return` должно следовать целочисленное значение.

```
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
```

Тип данных, возвращаемый оператором `return`, должен строго соответствовать типу, указанному в сигнатуре метода. Строгая проверка типов во время компиляции обеспечивает безопасность типов, предотвращая ошибки, которые могли бы возникнуть из-за несоответствия типов данных при вызове метода и присваивании его результата переменной.

Важно убедиться, что каждая возможная ветвь исполнения в методе в конечном итоге достигает оператора `return`. Если существует путь выполнения кода, который не приводит к `return`, это приведет к ошибке компиляции. Например, в условных операторах или циклах следует учитывать все возможные пути выполнения.

Методы, возвращающие значения, облегчают структурирование программы. Они позволяют разделить сложные вычисления на более мелкие части и использовать результаты этих вычислений в других частях программы. Это также улучшает читаемость и поддержку кода.

В Java **перегрузка методов (method overloading)** предоставляет возможность определять несколько методов с одинаковым названием в одном классе, при условии, что их список формальных параметров различается по типу или количеству. Это позволяет методам с одним и тем же именем выполнять разные задачи, что улучшает читаемость и логику кода, так как одноименные методы обычно выполняют схожие или связанные функции, но с разными данными.

**// 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) { ... }
```

Однако существуют ограничения при перегрузке методов в Java. Методы не могут быть перегружены, если они отличаются только типом возвращаемого значения или названиями параметров. Ключевым фактором для перегрузки является различие в типах и/или количестве параметров. Если два метода имеют одинаковое название, одинаковый порядок и типы параметров, но различаются только типами возвращаемых значений или названиями параметров, компилятор Java не сможет различить, какой метод вызывается, что приведет к ошибке компиляции.

**// 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) { ... }
```