

Методы

№ урока: 7 Курс: Java Starter

Средства обучения: Компьютер с установленной IntelliJ IDEA

Обзор, цель и назначение урока

Рассмотрение работы методов.

Изучив материал данного занятия, учащийся сможет:

- Понимать работу методов.
- Понимать отличие процедуры от функции.

Содержание урока

1. Обзор методов.
2. Различия между процедурами и функциями.
3. Рассмотрение примеров: Работа методов.
4. Рассмотрение управляющей структуры `return`.
5. Использование сторожевых операторов.
6. Рассмотрение примера: Использование сторожевого оператора, для защиты номинального варианта.

Резюме

- **Метод** – это именованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо.
- **Метод – это функция или процедура, выполняющая одну задачу.**
- О функциях и процедурах. В некоторых языках программирования (например, в Паскале) функции и процедуры (подпрограммы, не возвращающие значения) четко разграничены синтаксисом языка. В языке Java, – процедуры являются частным случаем (подмножеством) функций, возвращающими значение типа `void` – пустое значение.
- **Функция** – это метод, возвращающий значение, **процедура** – это метод который значения не возвращает. Все методы в Java технически являются функциями, но логически методы, которые возвращают `void`, являются процедурами.
- В Java таких понятий как **функция и процедура** вообще не существуют, все относится к **методам**.
- Определение метода задает имена и типы любых необходимых параметров. Когда код вызова вызывает метод, он передает в него конкретные значения, называемые аргументами, для каждого параметра. Аргументы должны быть совместимыми с типом параметра.
- Принято различать сигнатуру вызова и сигнатуру реализации метода. Сигнатурой вызова обычно является по синтаксической конструкции вызова метода с учётом имени данной функции, последовательности фактических типов аргументов в вызове и типе результата. В сигнатуре реализации обычно участвуют некоторые элементы из синтаксической конструкции объявления функции: спецификатор области видимости функции, её имя и последовательность формальных типов аргументов.
- **Сигнатура метода** – часть общего объявления метода, позволяющая идентифицировать функцию среди других. В Java, в сигнатуре метода входит идентификатор метода, тип (уже не включается в сигнатуре метода) и количество формальных аргументов.
- Вызов метода объекта очень похож на обращение к полю. После имени объекта ставится точка, затем имя метода и скобки. В скобках перечисляются аргументы, разделенные запятыми.
- Самая важная причина создания методов – это снижение сложности программ.
- Одна из главных задач, которые решают методы, – избежать дублирования кода. Или другими словами, методы открывают возможность повторного использования кода.

- Методы реализуют идею сокрытия информации. Один раз, создав метод, вы его используете, не думая о его внутренней работе.
- Использование методов приводит к минимизации кода, облегчению сопровождения программ и снижению числа ошибок.
- Использование методов, формирует понятную промежуточную абстракцию.
- Выделение фрагмента кода в отдельный, удачно названный метод, является одним из способов документирования целей данного метода.
- Методы позволяют выполнять оптимизацию кода в одном месте. Это облегчает профилирование кода, направленное на определение неэффективных фрагментов.
- Оператор `return` – это управляющая структура, которая позволяет программе в нужный момент завершить работу метода. В результате метод завершается через нормальный канал выхода, возвращая управление вызывающему методу.
- Используйте `return`, если это повышает читабельность метода.
- Используйте `return`, как сторожевой оператор досрочного выхода.
- Методы могут возвращать значения вызывающим их объектам. Если тип возвращаемого значения, указываемый перед именем метода, не равен `void`, для возвращения значения используется ключевое слово `return`.
- В результате выполнения инструкции с ключевым словом `return`, после которого указано значение нужного типа, вызвавшему метод объекту будет возвращено это значение.
- Ключевое слово `return` останавливает выполнение метода.
- Если тип возвращаемого значения `void`, инструкцию `return` без значения все равно можно использовать для завершения выполнения метода.
- Если ключевое слово `return` отсутствует, выполнение метода завершится, когда будет достигнут конец его блока кода.
- Для возврата значений методами с типом возвращаемого значения отличным от `void` необходимо обязательно использовать ключевое слово `return`.
- Чтобы использовать возвращаемое методом значение в вызываемом методе, вызов метода можно поместить в любое место кода, где требуется значение соответствующего типа.
- Возвращаемое значение метода можно присвоить переменной.
- Минимизируйте число возвратов из каждого метода. Тяжело понять логику метода, когда при анализе нижних строк приходится помнить о возможных выходах в верхних строках.
- Так как, методы – это конструкции для выполнения действий, рекомендуется их называть глагольными фразами или глаголами.
- Страйтесь именовать методы в соответствии с задачами, которые они выполняют, а не в соответствии с деталями реализации.
- Для именования методов в Java, рекомендуется использовать соглашение camelCasing. Чтобы выделить слова в идентификаторе, первые буквы каждого слова (кроме первого) сделайте заглавными. Например, `writeLine`, `getType`.
- Язык Java чувствительный к регистру (case sensitivity). Например, `GetType` и `getType` – это разные имена.
- Не используйте символы подчеркивания, дефисы и любые другие неалфавитно-цифровые символы для разделения слов в идентификаторе.
- Описывайте все, что метод выполняет.
- Избегайте невыразительных и неоднозначных глаголов или фраз.
- Для именования метода-функции рекомендуется использовать описание возвращаемого значения. Например: `currentColor()`

Закрепление материала

- Что такое метод?
- Чем отличаются функции и процедуры?
- Что делает оператор `return`?
- Что такое сигнатура метода?
- Какие правила именования применимы к методам?

Дополнительное задание

Задание

Используя IntelliJ IDEA, создайте класс **Calculator**.

Создайте метод с именем `calculate`, который принимает в качестве параметров три целочисленных аргумента и выводит на экран среднее арифметическое значений аргументов.

Самостоятельная деятельность учащегося

Задание 1

Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2

Используя IntelliJ IDEA, создайте класс **Arithmetics**.

Создайте четыре метода для выполнения арифметических операций, с именами: `add` – сложение, `sub` – вычитание, `mul` – умножение, `div` – деление. Каждый метод должен принимать два целочисленных аргумента и выводить на экран результат выполнения арифметической операции соответствующей имени метода. Метод деления `div`, должен выполнять проверку попытки деления на ноль.

Требуется предоставить пользователю возможность вводить с клавиатуры значения операндов и знак арифметической операции, для выполнения вычислений.

Задание 3

Используя IntelliJ IDEA, создайте класс **Conversion**.

Напишите программу, которая будет выполнять конвертирование валют.

Пользователь вводит:

сумму денег в определенной валюте,

курс для конвертации в другую валюту.

Организуйте вывод результата операции конвертирования валюты на экран.

Задание 4

Используя IntelliJ IDEA, создайте класс **NumbersCheck**.

Напишите метод, который будет определять:

1) является ли введенное число положительным или отрицательным.

2) Является ли оно простым (используйте технику перебора значений).

Простое число – это натуральное число, которое делится на 1 и само на себя. Чтобы определить простое число или нет, следует найти все его целые делители. Если делителей больше 2-х, значит оно не простое.

3) Делится ли на 2, 5, 3, 6, 9 без остатка.

Рекомендуемые ресурсы

Методы в Java

<https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>