

Методи

№ уроку: 7 Курс: Java Starter

Засоби навчання: Комп'ютер із встановленою IntelliJ IDEA

Огляд, мета та призначення уроку

Розгляд роботи методів

Вивчивши матеріал цього заняття, учень зможе:

- Розуміти роботу методів.
- Розуміти відмінність процедурі від функції.

Зміст уроку

1. Огляд методів.
2. Відмінності між процедурами та функціями.
3. Розгляд прикладів: Робота методів.
4. Розгляд керівної структури `return`.
5. Використання сторожових операторів.
6. Розгляд прикладу: Використання сторожового оператора для захисту номінального варіанту.

Резюме

- **Метод** – це іменована частина програми, яка може викликатися з інших частин програми стільки разів, скільки потрібно.
- **Метод** – це функція чи процедура, яка виконує одне завдання.
- **Про функції та процедури.** У деяких мовах програмування (наприклад, у Паскалі) функції та процедури (підпрограми, що не повертають значення) чітко розмежовані синтаксисом мови. У мові Java процедури є окремим випадком (підмножиною) функцій, що повертають значення типу `void` – порожнє значення.
- **Функція** – це метод, який повертає значення, **процедура** – це метод, який значення не повертає. Усі методи Java технічно є функціями, але логічно методи, які повертають `void`, є процедурами.
- У Java таких понять як **функція** та **процедура** взагалі немає, усе належить до **методів**.
- Визначення методу задає імена та типи будь-яких необхідних параметрів. Коли код виклику викликає метод, він передає конкретні значення (аргументи) для кожного параметра. Аргументи мають бути сумісні з типом параметра.
- Зavedено розрізняти сигнатуру виклику та сигнатуру реалізації методу. Сигнatura виклику зазвичай складається за синтаксичною конструкцією виклику методу з урахуванням імені цієї функції, послідовності фактичних типів аргументів у виклику та типі результата. У сигнатурі реалізації зазвичай беруть участь деякі елементи із синтаксичної конструкції оголошення функції: специфікатор області видимості функції, її ім'я та послідовність формальних типів аргументів.
- **Сигнатура методу** – частина загального оголошення методу, що дає змогу ідентифікувати функцію серед інших. У Java до сигнатурі методу входить ідентифікатор методу, тип (вже не входить до сигнатурі методу) і кількість формальних аргументів.
- Виклик методу об'єкта дуже нагадує звернення до поля. Після імені об'єкта ставиться крапка, потім ім'я методу та дужки. У дужках перераховуються аргументи, які розділені комами.
- Найважливіша причина створення методів – зниження складності програм.
- Одне з головних завдань, які вирішують методи, – уникнення дублювання коду. Або іншими словами, методи відкривають можливість повторного використання коду.
- Методи реалізують ідею приховування інформації. Створивши метод один раз, ви його використовуєте, не думаючи про його внутрішню роботу.
- Використання методів приводить до мінімізації коду, полегшення супроводу програм і зниження кількості помилок.

- Використання методів формує зрозумілу проміжну абстракцію.
- Виділення фрагмента коду в окремий, вдало названий метод – один зі способів документування цілей цього методу.
- Методи дають змогу оптимізувати код в одному місці. Це полегшує профілювання коду, яке спрямоване на визначення неефективних фрагментів.
- Оператор `return` – це керівна структура, яка дає змогу програмі в потрібний момент завершити **роботу** методу. У результаті метод завершується через нормальній канал виходу, повертаючи керування викликаному методу.
- Використовуйте `return`, якщо це підвищує читабельність методу.
- Використовуйте `return` як сторожовий оператор дострокового виходу.
- Методи можуть повертати значення об'єктів, які їх викликають. Якщо тип поверненого значення, яке вказується перед ім'ям методу, не дорівнює `void`, для повернення значення використовується ключове слово `return`.
- В результаті виконання інструкції з ключовим словом `return`, після якого вказано значення потрібного типу, викликаному методом об'єкта буде повернено це значення.
- Ключове слово `return` зупиняє виконання методу.
- Якщо тип поверненого значення `void`, інструкцію `return` без значення однаково можна використовувати для завершення виконання методу.
- Якщо ключове слово `return` відсутнє, виконання методу завершиться тоді, коли буде досягнуто кінець його блоку коду.
- Для повернення значень методами з типом поверненого значення, відмінним від `void`, необхідно обов'язково використовувати ключове слово `return`.
- Щоби використовувати повернене значення методом у викликаному методі, виклик методу можна помістити в будь-яке місце коду, де потрібне значення відповідного типу.
- Повернене значення методу можна присвоїти змінній.
- Мінімізуйте кількість повернень із кожного методу. Важко зрозуміти логіку методу, коли під час аналізу нижніх рядків доводиться пам'ятати про можливі виходи у верхніх рядках.
- Оскільки методи – це конструкції для виконання дій, рекомендується називати їх дієслівними фразами або дієсловами.
- Намагайтесь називати методи відповідно до завдань, які вони виконують, а не відповідно до деталей реалізації.
- Для найменування методів Java рекомендується використовувати угоду camelCasing. Щоби виділити слова в ідентифікаторі, зробіть перші літери кожного слова (крім першого) величими. Наприклад, `writeLine`, `getType`.
- Мова Java чутлива до регістру (case sensitivity). Наприклад, `GetType` та `getType` – це різні імена.
- Не використовуйте символи підкреслення, дефіси та інші неалфавітно-цифрові символи для розділення слів в ідентифікаторі.
- Описуйте все, що виконує метод.
- Уникайте невиразних і неоднозначних дієслів чи фраз.
- Для найменування методу-функції рекомендується використовувати опис поверненого значення. Наприклад: `currentColor()`

Закріплення матеріалу

- Що таке метод?
- Чим відрізняються функції та процедури?
- Що робить оператор `return`?
- Що таке сигнатура методу?
- Які правила іменування можна застосувати до методів?

Додаткове завдання

Використовуючи IntelliJ IDEA, створіть клас **Calculator**. Створіть метод з ім'ям `calculate`, який приймає як параметри три цілих аргументи та виводить на екран середнє арифметичне значень аргументів.

Самостійна діяльність учня

Завдання 1

Вивчіть основні конструкції та поняття, розглянуті на уроці.

Завдання 2

Використовуючи IntelliJ IDEA, створіть клас **Arithmetics**. Створіть чотири методи для виконання арифметичних операцій з іменами: add – додавання, sub – віднімання, mul – множення, div – ділення. Кожен метод має приймати два цілих аргументи та виводити на екран результат виконання арифметичної операції відповідної імені методу. Метод поділу div має виконувати перевірку спроби поділу на нуль.

Потрібно надати користувачеві можливість вводити з клавіатури значення операндів і знак арифметичної операції для виконання обчислень.

Завдання 3

Використовуючи IntelliJ IDEA, створіть клас **Conversion**. Напишіть програму, яка конвертуватиме валюти.

Користувач вводить:

- 1) суму грошей у певній валюті;
- 2) курс конвертації в іншу валюту.

Організуйте виведення результату операції конвертування валюти на екран.

Завдання 4

Використовуючи IntelliJ IDEA, створіть клас **NumbersCheck**. Напишіть метод, який визначатиме:

- 1) чи є введене число позитивним, чи негативним;
- 2) чи є воно простим (використовуйте техніку перебору значень).
Просте число – це натуральне число, яке ділиться на 1 й саме на себе. Щоби визначити просте число чи ні, варто знайти всі його цілі дільники. Якщо дільників більше 2-х, то воно не просте;
- 3) чи ділиться воно на 2, 5, 3, 6, 9 без залишку.

Завдання 5

Опишіть метод PowerA3(A, B), який обчислює третій ступінь числа A і повертає її до змінної B (A – вхідний, B – вихідний параметр; обидва параметри є дійсними). За допомогою цієї процедури знайдіть третій ступінь п'яти зазначених чисел.

Завдання 6

Опишіть метод PowerA234(A, B, C, D), який обчислює другий, третій і четвертий ступінь числа A і повертає ці ступені відповідно до змінних B, C і D (A – вхідний, B, C, D – вихідні параметри; усі параметри є дійсними). За допомогою цієї процедури знайдіть другий, третій і четвертий ступені п'яти зазначених чисел.

Рекомендовані ресурси

Методи в Java

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