Змінні та типи даних

№ уроку: 3 **Курс:** Java Starter

Засоби навчання: Комп'ютер з встановленим IntelliJ IDEA

Огляд, ціль та призначення уроку

Огляд поняття змінної, константи та типу даних.
Огляд арифметичних операторів та операторів порівняння.

Вивчивши матеріал даного заняття, учень зможе:

- Застосовувати змінні та константи.
- Розуміти, коли та які типи використовувати під час створення змінної.
- Виконувати арифметичні операції над значеннями змінних.
- Порівнювати значення змінних.
- Виконувати форматування рядків.

Зміст уроку

1. Огляд прикладу: Константи.
2. Огляд прикладу: Перетворення типів (Casting).
3. Огляд прикладу: Арифметичні оператори.
4. Огляд прикладу: Математичні функції.
5. Огляд прикладу: Інкремент та Декремент.
6. Огляд прикладу: Операції порівняння.
7. Огляд прикладу: Присвоєння з дією.
8. Огляд прикладу: Локальні області видимості.
9. Огляд прикладу: Ключові слова у якості ідентифікаторів.
10. Огляд прикладу: Конкатенація
11. Огляд прикладу: Форматування рядків
12. Огляд прикладу: Порівняння значень різних типів.

Резюме

- **Змінна (Variable)** – це іменована область пам'яті, яка зберігає у собі певне значення, яке можна змінити.
- **Ініціалізація змінної** – це перше присвоєння їй значення. Всі наступні присвоєння нових значень цій змінній не вважаються ініціалізацією.
- Технічно, імена змінних можуть починатися зі знаку «_» – нижнє підкреслення, знаку
- «$» і будь-якого алфавітного символу (імена не можуть починатися з цифр та інших символів).
- Для іменування локальних змінних Java, рекомендується використовувати угоду camel Casing. Щоб виділити слова в ідентифікаторі, зробіть перші літери кожного слова (крім першого) великими. Наприклад, `myAge, myName`.
- Мова Java чутлива до реєстру (casesensitivity). Наприклад, `MyName та myName` – це різні імена.
- Не використовуйте символи підкреслення, дефіси та інші неалфавітно-цифрові символи для розділення слів в ідентифікаторі.
- Не використовуйте угорську нотацію. Суть угорської нотації зводиться до того, що імена ідентифікаторів передуються заздалегідь обумовленими префіксами, які складаються з одного чи навіть кількох символів. Наприклад, `String sClientName; int iSize;`
- Імена змінних мають бути зрозумілі та передавати сенс кожного елементу.
- В окремих випадках, якщо ідентифікатор не має точного семантичного значення, використовуйте загальні назви. Наприклад, `value, item`.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

t. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- При створенні змінної, використовуйте назву-псевдонім, коли це можливо, а не повне ім'я типу.
- **Константа (Constant)** – це область пам'яті, яка містить у собі певне значення, яке не можна змінити.
- Правила використання констант:
  1. Константам необхідно присвоювати значення безпосередньо у місці створення;
  2. Спроба присвоєння константі нового значення призводить до помилки рівня компіляції;
- Перетворення типу (Casting або Type conversion) – це перетворення значення змінної одного типу на значення іншого типу (перетворення не слід плутати з приведенням типів – Cast). Виділяють явне (explicit) та неявне (implicit) перетворення типів.
- Неявне перетворення типу (безпечне) – перетворення меншого типу на більший або цілого типу на дійсний. Є безпечним, тому що не відбувається втрати точності.
- Явне перетворення типу (небезпечне) – перетворення більшого типу на менший чи дійсного типу на цілий. Є небезпечним, оскільки відбувається втрата точності результату без заокруглення.
- Можливе неявне перетворення значення константи більшого типу в менший, при ініціалізації змінної значенням константи, якщо значення константи не перевищує максимально допустимого значення змінної.
- Можливе явне перетворення значення константи дійсного типу на цілий тип при ініціалізації змінної значенням константи, якщо значення константи не перевищує максимально допустимого значення змінної.
- Якщо значення константи перевищує максимально допустимий діапазон значення змінної, таке перетворення можливе зі втратою результату (всі старші біти будуть відкинуті).
- Оператор присвоєння (=) зберігає значення свого правого операнду у місці зберігання (змінної), позначеної у лівому операнді. Операнди повинні бути одного типу (або правий операнд повинен допускати явне перетворення на тип лівого операнда).
- Якщо **після знаку присвоєння** йде вираз із обчисленням чи передачею будь-яких значень, то дана операція **виконується справа наліво**. Для підвищення пріоритету операції можна використовувати круглі дужки ( ).
- **Тільки чотири операції гарантують порядок обчислень зліва направо: ,, ?:, && і ||**
- Мова Java надає великий набір операторів, які являють собою символи, що визначають операції, які необхідно виконати з виразом. До операторів, які виконують арифметичні операції, можна віднести оператори:

  +(додавання),
  − (віднімання),
  *(множення),
  / (ділення),
  % (отримання залишку від ділення)
- Мова Java надає великий набір математичних функцій виконання різних обчислень.
- Math.sqrt() – математична функція, яка обчислює квадратний корінь. В аргументних дужках вказуємо значення числа, із якого хочемо витягти квадратний корінь.
- Math.pow() – зведення числа в степінь. В аргументних дужках через кому вказуємо два аргументи (перший – число, яке хочемо звести у степінь, другий – степінь, в яку хочемо звести число).
- **Операції множення, ділення, отримання залишку від ділення мають більший пріоритет, ніж додавання та віднімання**, тому виконуються у першу чергу.
- При отриманні результату залишку від ділення – знак результату не скорочується і відповідає значенню першого операнда (діленого).
- Якщо в правій частині виразу виконувалися операції ділення між цілими числами, то результат буде приведений компілятором до цілого типу, навіть якщо результат записати в змінну дійсного типу або привести всі вирази до дійсного типу.
- Оператор інкременту (++) збільшує свій операнд на 1. Оператор інкременту може бути як перед операндом, і після нього: ++variable або variable++.

CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

t. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **Префіксна операція збільшення** – результатом виконання цієї операції є використання значення операнду після його збільшення.
- **Постфіксна операція збільшення** – результат виконання цієї операції є використання значення операнду перед його збільшенням.
- Оператор **декременту** (--) зменшує свій операнд на 1. Оператор декременту може бути як перед операндом, так і після нього: --variable або variable--.
- Префіксна операція декременту – результатом виконання цієї операції є використання значення операнду після його декременту.
- Постфіксна операція декременту – результатом цієї операції є використання значення операнду до його декременту.
- До операцій порівняння можна віднести операції:
  > більше,
  >= більше або дорівнює,
  < менше,
  <= менше або дорівнює.
- До операцій перевірки на рівність можна віднести операції:
  == дорівнює,
  != не дорівнює.
- Результатом виконання операцій порівняння та перевірки на рівність-нерівність завжди буде або `false`, або `true`.
- Для передвизначених типів значень оператор рівності (==) повертає значення `true`, якщо значення його операндів збігаються, інакше – значення `false`. Для типу `string` оператор == порівнює значення рядків.
- Оператор нерівності (!=) повертає значення `false`, якщо його операнди рівні, інакше – значення `true`.
- Оператор порівняння "менше або дорівнює" **(<=)** повертає значення `true`, якщо перший операнд менше або дорівнює другому, інакше повертається значення `false`.
- Оператор порівняння "менше" **(<)** повертає значення `true`, якщо перший операнд менше другого, інакше повертається значення `false`.
- Оператор порівняння "більше" **(>)** повертає значення `true`, якщо перший операнд більший за другий, в іншому випадку повертається значення `false`.
- Оператор порівняння "більше або дорівнює" **(>=)** повертає значення `true`, якщо перший операнд більше або дорівнює другому, інакше повертається значення `false`.
- Усі арифметичні операції, що виконуються над двома значеннями типу (`byte, short`), як результат повертають значення типу `int`.
- Для типів `int`, `long` не відбувається перетворення типу результату арифметичних операцій.
- **Локальна область** – ділянка коду всередині класу або блок, обмежений фігурними дужками.
- **Область видимості змінної** – частина тексту програми, у якій ім'я можна явно використовувати. Найчастіше область видимості збігається із областю дії.
- Змінна, створена всередині локальної області, називається **локальною змінною**, область її дії — від відкриваючої дужки локальної області до її закінчення (закриваючої дужки блоку), включаючи всі вкладені локальні області.
- Змінна рівня класу називається **глобальною змінною чи полем**.
- У коді можна створювати локальні області та у двох різних локальних областях зберігати однойменні змінні.
- Якщо код містить локальні області, то забороняється зберігати однойменні змінні за межами локальних областей. І навпаки, якщо за межами локальних областей вже створено змінні з якимось ім'ям, то в локальних областях цього рівня забороняється створювати однойменні змінні.
- **Конкатенація** – **зчеплення рядків** або значень змінних типу `string` для отримання рядків більшого розміру за допомогою операції +.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

t. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Для форматування числових результатів та виведення їх на екран можна використовувати метод `System.out.print()` або `System.out.println()`, який викликає метод `String.format()`. Також можна скористатися методом `System.out.printf()`, який виводить рядок із заданим форматуванням.
- Інструкція форматування виглядає так:
  %[argument_index$][flags][width][.precision]conversion, де
  - **%** – спеціальний символ, що позначає початок конструкції форматування.
  - **[argument_index$]** – ціле десяткове число, що вказує позицію аргументу в списку аргументів (1$ – перший аргумент зі списку, 4$ – четвертий). Не є обов'язковою частиною конструкції; якщо позиція не задана, то аргументи братимуться у порядку черговості.
  - **[flags]** – спеціальні прапори форматування. Не є обов'язковою частиною конструкції.
  - **[width]** – позитивне ціле десяткове число, що визначає мінімальну кількість символів, які будуть виведені. Не є обов'язковою частиною конструкції.
  - **[.precision]** – позитивне ціле десяткове число з точкою перед ним. Використовується для обмеження кількості символів. Не є обов'язковою частиною конструкції.
  - **Conversion** – символ, що вказує, як аргумент має бути відформатований. Не є обов'язковою частиною конструкції.

Закріплення матеріалу

- Що таке змінна?
- Де і навіщо використовуються змінні?
- Назвіть основні типи даних.
- Які типи даних підходять для зберігання значень чисел із плаваючою комою?
- У якому форматі повинні задаватися значення для рядкових змінних?
- Що таке константа?
- У яких випадках використовують константи?
- Що таке перетворення значень типів (Casting)?
- Які існують правила використання перетворення значень під час роботи з константами?
- У чому різниця явного та неявного перетворення значення типу?
- Що таке конкатенація?
- Що таке інкремент та декремент?
- Які обмеження застосовуються до неініціалізованих локальних змінних?
- Чи можна використовувати в операціях порівняння два значення різних типів даних?

Додаткове завдання

Завдання
Використовуючи IntelliJ IDEA, створіть проект із класом main.
Створіть дві цілочисельні змінні та виведіть на екран результати всіх арифметичних операцій над цими двома змінними.

Самостійна діяльність учня

Завдання 1
Є 3 змінні типу `int x = 10, y = 12, и z = 3;`
Виконайте та розрахуйте результат наступних операцій для цих змінних:

- `x += y - x++ * z;`
- `z = --x – y * 5;`
- `y /= x + 5 % z;`
- `z = x++ + y * 5;`

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
t. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- `x = y - x++ * z;`

Завдання 2
Використовуючи IntelliJ IDEA, створіть клас **Arithmetic Average**.
Обчисліть середнє арифметичне трьох цілочисельних значень та виведіть його на екран.
З якою проблемою ви зіткнулися? Який тип змінних краще використовувати для коректного відображення результату?

Завдання 3
Використовуючи IntelliJ IDEA, створіть клас **Circle**.
Створіть константу під назвою PI (число π «пі»), створіть змінну радіус з назвою – r. Використовуючи формулу πR^2, обчисліть площу кола та виведіть результат на екран.

Завдання 4
Використовуючи IntelliJ IDEA, створіть клас **Volume**.
Напишіть програму розрахунку об'єму - V та площі поверхні - S циліндра. Об'єм V циліндра радіусом – R та висотою – h, обчислюється за формулою: $V = \pi R^2 h$
Площа поверхні циліндра обчислюється за формулою: $S = 2\pi R^2 + 2\pi Rh = 2\pi R(R+h)$ Результати розрахунків виведіть на екран.

Завдання 5
Використовуючи IntelliJ IDEA, створіть клас **Main**.
Перевірте, чи можна створити змінні з такими іменами:
`uberflu?, _Identifier, \u006fIdentifier, &myVar, myVariab1le`

Рекомендовані ресурси

Final методи та класи в Java
https://docs.oracle.com/javase/tutorial/java/IandI/final.html

Арифметичні операції
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op1.html

Приведення типів
https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

Клас Math
http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

Оператори порівняння
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op2.html

Форматування строк (клас Formatter)
http://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html

Введення даних з консолі. Клас Scanner
http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
t. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com