

УМОВНІ КОНСТРУКЦІЇ

№ уроку: 4 Курс: Java Starter

Засоби навчання: Комп'ютер із встановленою IntelliJ IDEA

Огляд, мета та призначення уроку

Розгляд операторів розгалуження для побудови умовних конструкцій.

Вивчивши матеріал цього заняття, учень зможе:

- Розуміти роботу операторів розгалуження.
- Використовувати умовні конструкції: **if-else**.
- Використовувати тернарний оператор.
- Використовувати оператор багатозначного вибору **switch-case**.

Зміст уроку

1. Розгляд поняття розгалуження в програмуванні.
2. Огляд операторів розгалуження.
3. Розгляд випадків застосування умовних конструкцій.
4. Розгляд прикладу: Умовна конструкція – **if** (з однією гілкою).
5. Розгляд прикладів: Умовна конструкція – **if-else** (з двома гілками).
6. Розгляд прикладу: Умовна конструкція – **if-else** (з кількома гілками). Каскад умовних операторів.
7. Розгляд прикладів: Тернарна умовна операція (**? :**).
8. Розгляд прикладу: Обмеження, які пов'язані з типобезпекою.
9. Розгляд прикладу: Вкладені тернарні оператори.
10. Розгляд прикладів: Оператор багатозначного вибору – **switch-case** (перемикач).
11. Розгляд прикладів: Провалювання в перемикачах.

Резюме

- **Оператор розгалуження** (умовна конструкція, умовний оператор) – оператор, конструкція мови програмування, яка забезпечує виконання певної команди (набору команд) лише за умови істинності деякого логічного виразу або виконання однієї з кількох команд (наборів команд) залежно від значення деякого висловлювання.
- У Java є три основні форми умовної конструкції:
 - 1) умовний оператор (**if-else**);
 - 2) тернарний оператор (**? :**);
 - 3) оператор багатозначного вибору (перемикач, **switch-case**).
- **Умовний оператор if** реалізує виконання певних команд за умови, що логічний вираз, який використовується, в умові приймає значення **true**.
- Якщо використовувалася конструкція **if-else** і результатом умови було значення **true**, то виконується тільки тіло оператора **if**, а тіло блоку **else** залишиться не виконаним.
- Після виконання оператора **if** керування передається наступному оператору.
- Оператор, який виконується після перевірки умови, може бути будь-якого типу, зокрема й іншим оператором **if**, який вкладений в оригінальний оператор **if**. У вкладених операторах **if** пропозиція **else** належить до останнього оператора **if**, який не має відповідного **else**.
- Якщо тіло блоку **if** або **else** складається з одного виразу, то операторні дужки можна опустити.
- **Тернарна умовна операція** (записується **? :**) – операція, яка повертає свій другий чи третій операнд залежно від значення логічного висловлювання, заданого першим операндом.
- Тернарний оператор [**? :**] є скороченою формою конструкції **if... else**.
- Тернарний оператор складається з таких операндів:
(умова) **? (блок істинності або те): (блок інакше)**.
- Алгоритм роботи тернарного оператора:

(логічний вираз) ? вираз 1: вираз 2

1. Обчислюється логічний вираз (умова).
 2. Якщо логічний вираз істинний, то обчислюється значення виразу вираз 1 (блоку істинності), інакше – значення виразу вираз 2 (блоку інакше).
 3. Обчислене значення повертається.
- Тернарний оператор обов'язково має повертати значення, інакше буде помилка.
 - Або блок істинності та блок інакше мають бути однакового типу, або має бути неявне перетворення з одного типу на інший.
 - **Конструкція перемикача switch-case** має кілька (дві чи більше) гілок. Перемикач виконує одну задану гілку залежно від значення ключового виразу, що обчислюється. Принциповою відмінністю цієї конструкції від умовного оператора є те, що вираз, який визначає вибір гілки, що виконується, допускає використання не логічних значень.
 - Для порожніх операторів `case` дозволено «**провалювання**» від одного оператора до іншого.
 - У кожному операторі `case` вказується постійне значення. Виконується тіло того оператора `case`, постійне значення якого відповідає значенню виразу селектора оператора `switch`.
 - Якщо постійний вираз оператора `case` не містить відповідного значення, виконується блок `default`, якщо такий є. Якщо блок `default` відсутній, відбувається вихід за межі оператора `switch`.
 - Кожен блок `case`, як і блок `default`, у якому містяться оператори, має завершуватися оператором переходу `break`, `return` або `throw`.
 - Виконання порівняння значення виразу селектора з постійними значеннями операторів `case` з першого оператора і продовжується за списком, зазвичай до досягнення оператора переходу. У цій точці управління передається за межі оператора `switch` або переходить до іншого оператора `case`, якщо оператори переходу або тіло оператора `case` були відсутні. І так до того оператора `case`, у якого буде тіло й оператор переходу. Така техніка називається **провалюванням**.
 - Блок `default` може бути створений у будь-якому місці тіла перемикача `switch-case`. Винятком є тіло операторів `case`.

Закріплення матеріалу

- Чи обов'язково оператор `if` має використовуватися разом з оператором `else`?
- Чи обов'язково створювати блок `default` у перемикачі `switch`?
- Чи допустиме вкладення тернарних операторів?
- Значення якого типу можна передавати як параметр `if()`?
- Чи обов'язково в перемикачі `switch-case` використовувати оператор переходу `break`?
- Чи може `switch-case` мати лише блок `default`?
- Що таке техніка провалювання в операторі `switch-case`?

Додаткове завдання

Використовуючи IntelliJ IDEA, створіть клас **Translator**.

Напишіть програму «Українсько-англійський перекладач». Програма знає 10 слів про погоду. Потрібно, щоби користувач вводив слово українською мовою, а програма давала йому переклад англійською мовою. Якщо користувач ввів слово, для якого немає перекладу, варто вивести повідомлення, що такого слова немає.

Самостійна діяльність учня

Завдання 1

Вивчіть основні конструкції та поняття, розглянуті на уроці.

Завдання 2

Використовуючи IntelliJ IDEA, створіть клас **Calculator**.

Напишіть програму «Консольний калькулятор».

Створіть дві змінні з іменами `operand1` та `operand2`. Задайте змінним деякі довільні значення. Запропонуйте користувачу ввести знак арифметичної операції. Візьміть значення, введене користувачем, і помістіть його в рядкову змінну `sign`.

Для організації вибору алгоритму обчислювального процесу використовуйте перемикач switch. Виведіть на екран результат виконання арифметичної операції.

У разі використання операції розподілу організуйте перевірку спроби розподілу на нуль. І якщо така є, то скасуйте виконання арифметичної операції та повідомите користувача про помилку.

Завдання 3

Використовуючи IntelliJ IDEA, створіть клас **Interval**.

Напишіть програму визначення: чи потрапляє вказане користувачем число від 0 до 100 до числового проміжку [0 – 14] [15 – 35] [36 – 50][50 – 100]. Якщо так, то вкажіть, який саме проміжок. Якщо користувач вказує число, яке не входить до жодного з наявних числових проміжків, то виведіть відповідне повідомлення.

Завдання 4

Дано ціле число. Якщо воно є позитивним, додайте до нього 1; інакше не змінюйте його. Виведіть отримане число.

Завдання 5

Дано ціле число. Якщо воно є позитивним, додайте до нього 1; інакше відніміть із нього 2. Виведіть отримане число.

Завдання 6

Дано три цілих числа. Знайдіть максимальне, мінімальне та середнє.

Завдання 7

Дано номер місяця – ціле число в діапазоні 1–12 (1 – січень, 2 – лютий і так далі). Визначте кількість днів цього місяця для невисокосного року.

Рекомендовані ресурси

Оператор if-then-else

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html>

Оператор switch

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html>