Оператори

- Вирази та оператори
- Блоки
- Керуючі оператори

Вирази (expressions)

- Вирази (expressions) **складаються з** констант і змінних, операцій над ними, викликів методів і дужок.
- Всі елементи вирази **повинні бути сумісні**, не можна написати, наприклад, 2 + true.

Правила обчислення виразів

При обчисленні виразу виконуються **чотири правила**:

- Операції **одного пріоритету** обчислюються **зліва направо**: x + y + z обчислюється як (x + y) + z.
  Виняток: операції присвоювання обчислюються справа наліво: x = y = z обчислюється як x = (y = z).
- **Лівий** операнд обчислюється **раніше правого**.
- Операнди повністю обчислюються **перед виконанням** операції.
- Перед виконанням **комбінованої операції** присвоювання значення лівої частини **зберігається для використання в правій частині**.

Оператори (statements)

Оператор — це **завершений модуль** виконання

- оператори опису змінних і інших об'єктів
- оператори-вирази
- оператори присвоювання

Блоки

- Блок містить в собі **нуль або декілька операторів** з метою використати їх **як один оператор** в тих місцях, де за правилами мови можна записати тільки один оператор.
  - Можна записати і **порожній блок**, це просто **пара фігурних дужок** {}.
- Блоки також використовуються для **обмеження області дії змінних** і просто для поліпшення читаності тексту програми.

Керуючі оператори

- **умовний** оператор if;
- три оператори **циклу** while, do-while, for;
- оператор **варіанту** switch;
- оператори **переходу** break, continue і return;

Оператор if - then

Оператор **if** забезпечує **виконання або пропуск інструкції** залежно від зазначеного логічного умови.

Якщо умова **істинна**, то інструкція **виконується**.

- if (умова)
  - інструкція;
- або
- if (умова)
  - інструкція1;
- else
  - інструкція2;

Оператор if-then-else

Надає новий вибір у випадку якщо **результат** логічного виразу оператора if обчислений як **false**

- if (умова)
  - інструкція1;
- **else if (умова)**
  - інструкція2;
- else
  - інструкція3;

Оператор множинного вибору

```
switch (перемикач) {
case значення 1:
інструкція 1;
break;
case значення2:
Інструкція 2;
break;
...
default:
інструкція_по_замовчанню;
}
```

Оператор працює з **примітивними типами** byte, short, char і int.

Він **також працює** з перерахуваннями, класом String і декількома спеціальними класами Character, Byte, Short, and Integer

Цикл типу «поки» (оператори while та do … while)

- while (умова) {
  - **// Тіло циклу**
- }
-
- do {
  - **// Тіло циклу**
- } While (умова)

Цикл типу «n-раз» (оператор for)

- for (ініціалізація; умова; ітерація) {
  - **// тіло циклу, тобто дії повторювані циклічно**
- }

- int sum = 0;
- for (int j = 2; j <= 100; j = j + 2) {
  - sum = sum + j;
- }
- System.out.println (sum);

Керування циклами break

Оператор **break**:

- **Завершує послідовність** операторів в операторі **switch**.
- Використовується для **виходу з циклу**.
- Використовується в якості "цивілізованої" форми оператора **безумовного переходу** ("goto").

Керування циклами continue

- Оператор continue може використовуватися тільки в циклах while, do, for.
- Якщо в потоці обчислень зустрічається оператор continue, то виконання поточної послідовності операторів повинно бути припинено і управління буде передано на початок блоку, що містить цей оператор:

- int x = (int) (Math.random () * 10);
- int arr [] = {....}
- for (int cnt = 0; cnt <10; cnt ++) {
  - if (arr [cnt] == x) continue; ...
- }

Оператор return

- Оператор призначений для повернення управління з методу що був викликаний в метод що викликав.
- **return;**
- Якщо метод повертає значення то воно вказується після оператора return:
- **return вираз;**