***Оператор выбора***

Оператор ветвления/выбора **switch** похож на оператор условного перехода **if-else**. Он позволяет организовать разветвление процесса исполнения программы. Но в отличие от оператора **if-else**, применим только к известному числу возможных ситуаций.

Какие существуют нюансы в работе оператора **switch**?

1. Сначала происходит вычисление значения "выражения".
2. Затем начинается последовательное сравнение значения выражения со значением ключей веток оператора **switch**.
3. Когда их значения совпадают, происходит замыкание исполнения программы на блок операторов соответствующей ветки.
4. Далее требуется выход из оператора **switch**. Для этого используется оператор **break**.
5. Если ни один ключ не совпал со значением выражения, то либо происходит выход из оператора, либо выполняется блок кода по ветке **default** (если ветка присутствует в теле оператора, другим словами, она является не обязательной).

Важно также **понимать**, что значения ключей должны:

* иметь тот же тип, что и "выражение"
* быть литералами или неизменяемыми переменными
* иметь тип int, byte, short, char; а начиная с версии Java 5 ключи могут иметь еще и тип enum и String.
* Обратите внимание на описание синтаксиса для оператора **switch**.

Как вы увидите далее, в некоторых случаях использование оператора **switch** дает более компактный программный код по сравнению с оператором **if-else**. Например, если у вас есть конструкция с цепочкой операторов **if-else**, предлагающая большое количество условных ветвлений, то они могут выглядеть очень громоздкими. Поэтому в тех случаях, когда цепочкой операторов **if-else-if**’s выполняется проверка значения одной и той же переменной, есть другое решение – оператор **switch**.

В следующем видео вы рассмотрите пример применения оператора, а также варианты разного выполнения в случае наличия/отсутствия оператора **break** в ветках case.

Обратите внимание, что в Java 14 добавлены новые формы оператора **switch**, которые:

* не требуют оператора **break**
* позволяют объединять повторяющиеся метки через запятую
* применяют специальный синтаксис выражений
* допускают применение в форме метода.
* Далее представлены полезные примеры работы с оператором **switch**.
* Для перехода на ветку оператора выполняется проверка только на равенство значения выражения и ключа ветки.
* Тип выражения и ключей может быть целочисленным, кроме **long**, перечислением **enum** или типа **String**.
* Для перехода на ветку выражение-условие может быть любой сложности и вырабатывать значение только логического типа.
* Тип данных, участвующих в вычислении выражения, может быть любым.