Операторы языка Java

Порядок выполнения программы определяется операторами. Операторы языка Java могут содержать в себе другие операторы или выражения. Операторы в блоке выполняются слева направо, сверху вниз. Если все операторы (выражения) в блоке выполняются нормально, то весь блок выполняется нормально. Если какой - либо оператор (выражение) завершается ненормально, то весь блок завершается ненормально. Следует помнить, что определение локальной переменной есть исполняемый оператор. Если задана инициализация переменой, то выражение исполняется слева направо и его результат присваивается локальной переменной. Использование не инициализированных локальных переменных запрещено и вызывает ошибку времени компиляции.

Пустой оператор

Оператор ; называется *пустым оператором* и записывается там, где необходим оператор, но не предполагается выполнение никаких действий. Преждевременное завершение пустого оператора невозможно.

Операторы прерывания, перехода и возврата

Последовательность выполнения операторов может быть как непрерывной, так и прерываться при возникновении определенных условий. Выполнение оператора может быть прервано, если в потоке вычислений будут обнаружены следующие операторы:

- continue – предназначен для передачи управления на начало содержащего его блока и может применяться только в циклах;
- brake – предназначен для прерывания выполнения цикла или последовательности операторов case;
- return – предназначен для возврата управления из вызываемого метода в вызывающий.

В этом случае управление будет передано в другое место в соответствии с правилами обработки этих операторов. Нормальное выполнение оператора также прерывает явное возбуждение исключительной ситуации с помощью оператора throw, а также все исключения, возбуждаемые виртуальной машиной Java. Выражения также могут завершаться нормально и преждевременно. Причиной последовательности выполнения выражения, отличной от нормальной, может быть только возникновение исключительной ситуации. Если в операторе содержится выражение, то в случае его аварийного завершения выполнение оператора тоже будет завершено преждевременно.

Любой оператор или блок может иметь *метку*. Метку можно указывать в качестве параметра для операторов break и continue. Область видимости метки ограничивается оператором или блоком, к которому она относится. В случае, если имеется несколько вложенных блоков и операторов, метки внешних блоков будут

видимы во внутренних блоках. Традиционно использование меток не рекомендуется, особенно в объектно-ориентированных языках, поскольку серьезно усложняет понимание порядка выполнения кода, и, следовательно, его тестирование и отладку. Для языка Java этот запрет можно считать не столь строгим, поскольку метод перехода к метке goto в нём отсутствует.

Оператор условного перехода

Наиболее часто встречающейся конструкцией в Java, как и в любом другом структурном языке программирования, является *оператор условного перехода*. В общем случае его конструкция выглядит так:

if (логическое выражение) выражение или блок 1 else выражение или блок 2;

Логическое выражение может быть любой языковой конструкцией, которая возвращает результат логического типа. В том случае, если логическое выражение принимает значение true, выполняется выражение или блок 1, в противном случае – выражение или блок 2. Условные операторы, имеющие такую структуру, называют *полными*. Вторая часть оператора (ключевое слово else и выражение или блок 2) не является обязательной и может отсутствовать – в этом случае условный оператор называют *неполным*. В обеих частях оператора могут быть любые другие операторы, в том числе и другие (*вложенные*) операторы условного перехода. В таких случаях следует помнить, что оператор else относится к ближайшему к нему предыдущему оператору if.

Оператор множественного выбора

В ряде случаев набор вложенных операторов if/else возможно заменить *оператором множественного выбора*, который позволяет обработать сразу несколько условий и имеет следующую структуру:

```
switch(value){
  case const1:
    выражение или блок 1
  case const2:
    выражение или блок 2
  case constN:
    выражение или блок N
  default:
    выражение или блок N+1
}
```

Оператор множественного выбора описывается и выполняется в соответствии со следующими правилами:

- типами переменных, которые используются в операторе switch/case, могут быть только типы byte, short, char, int, строки и перечисления;
- в рамках одного оператора switch/case можно описать любое количество case;

- за каждым case следует сравниваемое значение, а затем идёт двоеточие;
- значение case должен быть того же типа данных, как и переменная в switch, и она должна быть константой или литералом;
- в рамках одного switch/case не может быть двух case с одинаковыми значениями;
- когда значение переменной switch становится равным значению одного из case, операторы, следующие за case, будут выполняться до тех пор, пока не будет достигнут оператор break;
- при достижении оператора break, оператор switch/case завершается, и управление переходит к строке, следующей за оператором switch/case;
- не каждый case должен содержать break. Если break отсутствует, управление передаётся на следующий case, и так до тех пор, пока не будет достигнут оператор break или конец оператора switch/case;
- оператор switch/case может иметь дополнительный default case, который должен находиться в конце switch. Default case может быть использован для выполнения задачи, когда ни один из вариантов case не является правильным. В default case оператор break не требуется.

В ситуации, когда блок кода необходимо выполнить несколько раз, используются операторы цикла. В языке Java имеется три основных конструкции управления циклами: цикл с параметром, цикл с предусловием и цикл с постусловием.

Операторы цикла

Довольно часто необходимо изменять значение какой-либо переменной в заданном диапазоне, и выполнять повторяющуюся последовательность операторов с использованием этой переменной. Для выполнения этой последовательности действий как нельзя лучше подходит конструкция *цикла с параметром*. Основная форма цикла с параметром выглядит следующим образом:

```
for (выражение инициализации; условие; выражение обновления)
    повторяющееся выражение или блок;
```

Ключевыми элементами данной языковой конструкции являются выражения, заключенные в круглых скобках и разделенные точкой с запятой:

- выражение инициализации – выполняется до начала выполнения тела цикла. Чаще всего оно используется как некое стартовое условие (инициализация, или объявление переменной);
- условие – должно быть логическим выражением. Тело цикла будет выполняться до тех пор, пока логическое выражение будет принимать значение true. Если логическое выражение принимает значение false до начала выполнения цикла, тело цикла может не исполниться ни разу;
- выражение обновления – выполняется сразу после исполнения тела цикла, и до того, как проверено условие продолжения выполнения цикла. Обычно

здесь используется выражение инкрементации, но может быть применено и любое другое выражение.

Любая часть конструкции for () может быть пропущена. В случае, когда все три выражения в скобках отсутствуют, цикл будет выполняться бесконечно.

Основная форма *цикла с предусловием* выглядит следующим образом:

```
while (логическое выражение)
   повторяющееся выражение или блок;
```

В данной языковой конструкции повторяющееся выражение или блок будет исполняться до тех пор, пока логическое выражение будет иметь значение true. При этом сначала производится проверка значения логического выражения, затем – выполнение тела цикла.

Если выражение или блок, представляющий тело цикла, будет завершен ненормальным образом, то, в зависимости от причины:

- если встретился оператор continue, то часть тела цикла, следующая за ним, будет пропущена, и выполнение цикла продолжится с начала тела цикла;
- если встретился оператор break, то выполнение цикла будет прекращено;
- если выполнение блока будет прекращено по другим причинам (возникла исключительная ситуация), то выполнение цикла while будет прекращено по тем же причинам.

Следует помнить, что цикл с предусловием будет выполнен только в том случае, если на момент начала его выполнения логическое выражение будет иметь значение true. В противном случае тело цикла с предусловием не будет выполнено ни разу.

Основная форма *цикла с постусловием* имеет следующий вид:

```
do
   повторяющееся выражение или блок;
while (логическое выражение);
```

Цикл с постусловием также будет выполняться до тех пор, пока логическое выражение имеет значение true. Но, в отличие от цикла с предусловием, в данном операторе цикла сначала производится выполнение тела цикла, затем – проверка значения логического выражения. По этой причине цикл с постусловием будет выполнен как минимум один раз, даже если на момент начала его выполнения логическое выражение будет иметь значение false. В остальном выполнение цикла с постусловием аналогично выполнению цикла с предусловием.