Приоритеты и скобки в условиях

Привет!

Разберём приоритетность булевых операторов в сложных выражениях и более простое написание таких выражений с использованием круглых скобок.

Ранее вы видели, что выражения, где проверяются условия, могут содержать несколько операторов сразу. Если это операторы И, то понять такое выражение очень легко:

```
if (!isBlocked &&
milkAmount >= cappucinoMilkRequired &&
coffeeAmount >= cappucinoCoffeeRequired) {
      System.out.println("Готовим кофе");
} else {
      System.out.println("Что-то пошло не так :(");
}
```

Оно выполняется тогда, когда все компоненты будут равны true. Если хотя бы один из них не равен true, оно не выполнится.

С оператором ИЛИ всё аналогично. Есть два вида молока в кофемашине, обычное и обезжиренное; добавим третий вид, например безлактозное:

```
int lowLactoseMilkAmount = 1000;
```

Напишем условие проверки того, что нам хватает молока:

```
if (skimmedMilkAmount >= cappucinoMilkRequired ||
milkAmount >= cappucinoMilkRequired ||
lowLactoseMilkAmount >= cappucinoMilkRequired) {
       System.out.println("Молока достаточно");
}
```

Здесь тоже всё легко: если хотя бы один из компонентов выражения равен true, то всё выражение будет равно true.

Но как будет работать выражение, содержащее сразу два оператора — оператор ИЛИ и оператор И?

Напишем такой код и объединим сразу все условия, которые писали до этого, но уберём безлактозное молоко для упрощения кода:

```
if (!isBlocked &&
coffeeAmount >= cappucinoCoffeeRequired &&
skimmedMilkAmount >= cappucinoMilkRequired ||
milkAmount >= cappucinoMilkRequired) {
System.out.println("Готовим кофе");
}
```

Здесь возникает вопрос: в какой последовательности будут работать операторы? Сначала выполнится оператор ИЛИ — то есть будет понятно, достаточно ли молока — а уже потом И? Или наоборот, сначала проверяются первые три условия, а потом условие ИЛИ?

Сначала посмотрим, как выражение будет выполняться, и дойдём до момента, когда надо будет понять, какой оператор выполнится первым.

Заменим первый компонент:

```
if (true &&
coffeeAmount >= cappucinoCoffeeRequired &&
skimmedMilkAmount >= cappucinoMilkRequired ||
milkAmount >= cappucinoMilkRequired) {
System.out.println("Готовим кофе");
}
```

Затем второй:

```
if (true &&
true &&
skimmedMilkAmount >= cappucinoMilkRequired ||
milkAmount >= cappucinoMilkRequired) {
System.out.println("Готовим кофе");
}
```

Затем третий и так далее.

Если мы заменим каждый компонент выражения на значение true или false, будет легче понять, каким будет конечный результат.

Вернёмся к вопросу приоритетности операторов: какой оператор выполнится первым, оператор И или оператор ИЛИ?

Сначала выполнится оператор И, то есть проверятся первые три условия, а уже потом оператор ИЛИ. Если кофемашина заблокирована, кофе и обезжиренного молока недостаточно, но достаточно обычного молока, то условие выполнится. Очевидно, что этот результат нас не устраивает и мы хотим обратной приоритетности операторов: сначала определить, достаточно ли нам хоть какого-нибудь молока, а потом все компоненты объединить с помощью оператора И.

Это можно сделать при помощи скобок:

```
if (!isBlocked &&
coffeeAmount >= cappucinoCoffeeRequired &&
(skimmedMilkAmount >= cappucinoMilkRequired ||
milkAmount >= cappucinoMilkRequired)) {
System.out.println("Готовим кофе");
}
```

Так сначала будет вычислено значение выражения в скобках, а затем всё остальное.

Запомнить приоритетность операторов бывает сложно, а код с большим количеством операторов, тем более разных, трудно писать, читать и поддерживать. Поэтому рекомендуется использовать скобки, чтобы явно устанавливать приоритетность выполнения операторов.

Использовать скобки нужно только тогда, когда может возникнуть путаница в приоритетности операторов. Не нужно писать такой код:

```
if (!isBlocked &&
(coffeeAmount >= cappucinoCoffeeRequired) &&
((skimmedMilkAmount >= cappucinoMilkRequired) ||
(milkAmount >= cappucinoMilkRequired))) {
System.out.println("Готовим кофе");
}
```

Также не стоит объединять скобками всё подряд — так легко сделать ошибку и такой код сложно будет читать.

Этот кусок кода можно упростить и сделать более поддерживаемым. Очевидно, что в этом коде используются однородные условия, например проверка количества молока. Такие условия можно выделить в отдельные переменные, получится сильно проще:

```
boolean milkIsEnough = skimmedMilkAmount >=
cappucinoMilkRequired ||
milkAmount >= cappucinoMilkRequired;
```

Условие с кофе тоже можно выделить, чтобы после оператора if было меньше нагромождений:

```
boolean coffeeIsEnough = coffeeAmount >=
cappucinoCoffeeRequired;
```

В итоге код будет таким:

```
if (!isBlocked && coffeeIsEnough && milkIsEnough) {
      System.out.println("Готовим кофе");
}
```

Итоги

с 09:33 до конца

- Оператор И приоритетнее оператора ИЛИ.
- В сложных случаях лучше приоритизировать операторы при помощи скобок — не так важно, знаете ли вы приоритетность операторов или нет. Так код становится более читабельным и понятным. Но стоит избегать излишних скобок: если их слишком много, в коде становится сложнее разобраться.
- Для чтения сложных выражений можно мысленно заменять их компоненты на значения true или false, это позволяет быстро оценить результат без запуска кода.
- Для упрощения сложных выражений их фрагменты выделяют в отдельные переменные и подставляют эти переменные в конечное выражение. Это существенно повышает читабельность, понятность и, как следствие, поддерживаемость программного кода.