

Вложенные условия

Привет!

Сегодня обсудим, что условные операторы и соответствующие блоки программного кода могут быть вложенными и важно придерживаться наименьшего уровня вложенности.

Ранее мы обсуждали код, проверяющий сразу несколько условий. Теперь давайте усложним задачу: если кофемашина заблокирована, будем выводить отдельное сообщение об ошибке, а если не заблокирована — проверять, достаточно ли кофе и молока.

Перепишем код следующим образом:

```
if (isBlocked) {
    System.out.println("Кофе-машина заблокирована");
} else {
    if (coffeeIsEnough && milkIsEnough) {
        System.out.println("Готовим кофе");
    } else {
        if (!coffeeIsEnough) {
            System.out.println("Кофе недостаточно");
        }
        if (!milkIsEnough) {
            System.out.println("Молока недостаточно");
        }
    }
}
```

Обратите внимание, что блоки кода могут быть вложенными и может быть несколько уровней вложенности. Также обратите внимание на правильное форматирование такого кода. Вложенные блоки должны быть сдвинуты вправо на один отступ, открывающая фигурная скобка должна стоять на той же строке, где находится соответствующий оператор, а закрывающая — на новой строке на том же уровне, что и оператор, к которому относился закрываемый ею блок кода.

Если вы запутались с форматированием кода, можете выделить код и выбрать в меню Code пункт Reformat Code. Эта команда отформатирует код правильным образом. Несмотря на то, что код мы отформатировали, такой код

— с тремя уровнями вложенности блоков — достаточно сложен как для написания, так и для понимания.

Поэтому давайте перепишем его, уменьшим количество уровней вложенности и тем самым сделаем его проще. Для начала уберём `else`:

```
if (isBlocked) {
    System.out.println("Кофе-машина заблокирована");
} else {
    if (coffeeIsEnough && milkIsEnough) {
        System.out.println("Готовим кофе");
    }
    if (!coffeeIsEnough) {
        System.out.println("Кофе недостаточно");
    }
    if (!milkIsEnough) {
        System.out.println("Молока недостаточно");
    }
}
```

Поскольку проверка равенства переменной `true` или `false` — очень быстрая операция, пара лишних проверок этот код не сделают медленнее.

Остается два уровня вложенности. Это тоже много, лучше ограничиваться одним уровнем. Можно написать, например, так:

```
if (isBlocked) {
    System.out.println("Кофе-машина заблокирована");
} else if (!coffeeIsEnough) {
    System.out.println("Кофе недостаточно");
} else if (!milkIsEnough) {
    System.out.println("Молока недостаточно");
} else {
    System.out.println("Готовим кофе");
}
```


Такой код будет выводить только какую-то одну ошибку. Если необходимо получить список всех ошибок, можно ввести дополнительную переменную, в которой будет храниться информация о том, есть ли какие-то ошибки:

```
boolean hasErrors = false;
```

Затем проверять каждое отдельное условие и менять значение этой переменной на true, если произошла какая-то ошибка:

```
if (isBlocked) {  
    System.out.println("Кофе-машина заблокирована");  
    hasErrors = true;  
}  
  
if (!coffeeIsEnough) {  
    System.out.println("Кофе недостаточно");  
    hasErrors = true;  
}  
  
if (!milkIsEnough) {  
    System.out.println("Молока недостаточно");  
    hasErrors = true;  
}  
  
if (!hasErrors) {  
    System.out.println("Готовим кофе");  
}
```

Итоги

- Программный код может быть вложенным, уровней вложенности может быть много. Такой код должен быть правильно отформатирован, чтобы его можно было прочитать. Форматирование кода производится сочетаниями клавиш: в Windows и Linux — Ctrl + Alt + L, в Mac OS —  ⌘ L.
- Код с большим количеством вложенностей нужно упрощать. Это можно сделать путём преобразования в цепочку условий else-if либо в последовательность условий с использованием дополнительных переменных. Такой код будет значительно понятнее, и его будет легко изменять при необходимости.