Булевы операции

Привет!

В этом уроке мы рассмотрим, как можно объединять и комбинировать различные условия — переменные и выражения, результат выполнения которых является булевым — то есть равен “true” или “false”.

Представьте, что мы пишем систему управления кофемашиной и, в частности, реализуем код, который будет проверять, достаточно ли нужных ингредиентов для приготовления определённого напитка.

Создадим для этой системы новый проект, назовем его CoffeeMachine.

Для простоты напишем код, который будет проверять, достаточно ли в кофе-машине ингредиентов для приготовления капучино. Сначала создадим переменные, в которых будут количества ингредиентов. Для простоты это будут граммы:

```
int coffeeAmount = 2330;
int milkAmount = 3210;
```

Воду указывать не будем, сделаем допущение, что кофе-машина подключена к воде, и воды всегда достаточно.

Теперь укажем минимальные количества кофе и молока для приготовления капучино. Пусть это будет 60 грамм молока и, к примеру, 15 грамм кофе:

```
int cappuccinoMilkRequired = 60;
int cappuccinoCoffeeRequired = 15;
```

Теперь напишем код. Мы должны проверить два условия:

- достаточно кофе
- достаточно молока.

Проверять отдельно каждое условие мы уже умеем:

```
if (coffeeAmount >= cappuccinoCoffeeRequired) {
      System.out.println("Готовим капучино");
} else {
      System.out.println("Ингредиентов недостаточно :(")
}
```

Оператор “И” (&&)

02:40 - 04:28

Теперь нужно проверить второе условие. Чтобы это сделать, нам следует воспользоваться оператором объединения условий. По-русски он называется оператором “***И***”, а в коде пишется как двойной амперсанд:

```
if (coffeeAmount >= cappuccinoCoffeeRequired &&
milkAmount >= cappuccinoMilkRequired) {
      System.out.println("Готовим капучино");
} else {
      System.out.println("Ингредиентов недостаточно :(")
}
```

Условие выполнится только тогда, когда оба выражения будут равны “true”, то есть когда обоих ингредиентов будет достаточно. В данном случае их более чем достаточно, поэтому кофемашина может приготовить капучино.

Этот код можно прочитать следующим образом: если количество кофе больше, либо равно, чем требуемое количество для капучино, ***И*** количество молока больше, чем требуемое, то выводим в консоль сообщение “Готовим капучино”, в противном случае выводим сообщение о том, что ингредиентов недостаточно.

Только что мы познакомились с оператором “И”, который обозначается в коде двойным символом амперсанда: **&&** и значение выражения с этим оператором равно “true” только в том случае, если значения всех его компонентов, объединяемых этим оператором, тоже равны “true”. Если хотя бы один из них равен “false”, то результат будет “false”. А “true” будет только, если и первый, и второй компоненты равны “true”.

Выражение	Результат
true && true	true
true && false	false
false && true	false
false && false	false

Оператор “ИЛИ” (||)

04:28 - 06:46

Второй булевый оператор — это оператор “***ИЛИ***”. Представьте, что в кофе-машине есть два вида молока — обычное и обезжиренное, а для приготовления капучино достаточно любого из них.

Добавим количество обезжиренного молока:

```
int skimmedMilkAmount = 1290;
```

и пропишем это в условии. Напишем условие попроще, которое будет проверять, достаточно ли молока:

```
if (skimmedMilkAmount >= cappuccinoMilkRequired ||
milkAmount >= cappuccinoMilkRequired) {
      System.out.println("Молока достаточно");
} else {
      System.out.println("Молока недостаточно");
}
```

Оператор “**ИЛИ**” обозначается в Java двойной вертикальной чертой. Условие в этом коде выполняется в случае, если хотя бы какого-нибудь молока будет достаточно — то есть если хотя бы один из компонентов выражения вокруг оператора “**ИЛИ**” равен “true”.

Читается такой код следующим образом: если количество обезжиренного молока больше, чем требуемое количество молока ИЛИ количество обычного молока больше, чем требуемое количество для капучино, то выводим в консоль сообщение о том, что молока достаточно, иначе — о том, что недостаточно.

Результат выполнения выражения с этим оператором будет равен “true”, если либо первый, либо второй компонент, либо оба его компонента равны “true”.

Если же все компоненты выражения равны “false”, то итоговое значение будет тоже равно “false”.

Выражение	Результат
true || true	true
true || false	true
false || true	true
false || false	false

Оператор отрицания “НЕ” (!)

06:47 - 09:37

Третий булевый оператор, который важно знать, это оператор отрицания. Он, по сути, меняет значение выражения или переменной на противоположное. В коде он пишется как восклицательный знак перед именем соответствующей переменной. Если переменная или выражение, перед которым он стоит, равно “true”, то итоговое значение будет равно “false”, и наоборот, если значение было равно “false”, оно изменится на “true”.

К примеру, вы хотите проверить, что кофемашина не заблокирована, и только в этом случае готовить кофе. Вот у вас есть переменная, в которой хранится состояние кофемашины:

```
boolean isBlocked = true;
```

и вы проверяете, заблокирована она или нет:

```
if (!isBlocked) {
      System.out.println("Готовим кофе");
}
else {
     System.out.println("Кофемашина заблокирована");
}
```

Если читать этот код, то получится очень просто: если НЕ заблокирована, выводим в консоль сообщение “Готовим кофе”.

Выражение	Результат
!true	false
!false	true

Обратите внимание, что оператор отрицания в Java в отличие от операторов “И” и “ИЛИ” - одинарный. Если написать двойной восклицательный знак, то получится двойное отрицание:

```
if (!!isBlocked) {

}
```

значение в этом случае изменится на противоположное дважды и в итоге останется таким же, каким оно было изначально. То есть если эта переменная была равна “true”, первый восклицательный знак поменяет её значение на “false”, а второй — обратно на “true”.

В коде часто условия могут объединяться. К примеру, мы хотим приготовить кофе только в случае, если достаточно молока, кофе и машина не заблокирована. В этом случае мы можем написать:

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

В данном случае кофе приготовится при выполнении сразу трёх условий, поскольку все эти три условия объединены оператором “И”. Во-первых, машина не должна быть заблокирована. Здесь используется оператор отрицания.

Во-вторых, должно быть достаточно молока, а в-третьих, — кофе. И только в этом случае в консоль выведется сообщение “Готовим кофе”. Если хотя бы одно из этих условий не выполняется, приготовить кофе не получится.

Итоги урока:

09:37 - до конца урока

- В этом уроке вы познакомились с тремя основными булевыми операторами, с помощью которых можно создавать сложные условия, состоящие из нескольких компонентов. Это операторы “И”, “ИЛИ” и “НЕ”.
- Вы узнали, как можно комбинировать несколько операторов в одно условие.