2\_9 Логический тип Boolean

**1. Тип boolean**

Как мы уже убедились, в Java есть очень полезный оператор — if-else. Он выполняет один блок команд, если условие в скобках истинно, и второй блок команд, если условие ложно.

Чтобы было удобно работать с выражениями, которые могут быть истинными или ложными, в Java добавили специальный тип — boolean. Его главная особенность заключается в том, что переменные этого типа могут принимать всего два значения: true (истина) и false (ложь).

Никакие другие значения присвоить переменным типа boolean невозможно. Компилятор не позволит.

И зачем же нужен такой примитивный тип?

Все дело в том, что в него можно сохранять значения логических выражений. Пример:

|  |  |
| --- | --- |
| **Код** | **Пояснение** |
| boolean isOK = true; | Логическая переменная isOK содержит значение true (истина) |
| boolean hasError = false; | Логическая переменная hasError содержит значение false (ложь) |
| int age = 70;  boolean isSenior = (age > 65); | Логическая переменная isSenior содержит значение true (истина) |
| int record = 612;  int value = 615;  boolean hasNewRecord = (value > record); | Логическая переменная hasNewRecord содержит значение true (истина) |
| int min = 0;  int max = 100;  int temperature = -20;  boolean isIce = (temperature < min);  boolean isSteam = (temperature > max); | Логическая переменная isIce содержит значение true (истина)  Логическая переменная isSteam содержит значение false (ложь) |

**2. Использование булевых переменных**

От булевых переменных было бы мало толку, если бы в них можно было только сохранять результаты выражений. Все дело в том, что их можно еще и использовать. Где? Да везде, где можно написать логическое выражение.

Например, логическую переменную можно подставить в условие if-а:

|  |  |
| --- | --- |
| **Код** | **Эквивалент** |
| int age = 70;  boolean isSenior = (age > 65);  if (isSenior)  System.out.println("Пора на пенсию"); | int age = 70;  if (age > 65)  System.out.println("Пора на пенсию"); |

В данном примере выгоды от такой замены немного, но чем больше программа, тем сложнее у нее условия. В ближайшее время вы в этом убедитесь.

**3. Операторы сравнения**

В Java, как и в других языках программирования, часто приходится сравнивать переменные между собой. И именно для сравнения в Java есть такие операторы:

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Пояснение** | **Пример** |
| < | Меньше | a < 10 |
| > | Больше | b > a |
| <= | Меньше либо равно | a <= 10 |
| >= | Больше либо равно | speed >= max |
| == | Равно | age == 18 |
| != | Не равно | time != 0 |

Результатом действия каждого из приведенных операторов будет логическое выражение. Его можно сохранить в переменную типа boolean, ну или использовать в качестве условия в if.

Важный момент 1:

Операторы, состоящие из двух символов, разрывать нельзя.

Т.е. такой код компилироваться не будет:

a < = 10

speed > = max

age = = 18

time ! = 0

Важный момент 2:

Обратите внимание, что операторов => и =< нет: есть только <= и >=. Если вы напишете a =< 3, ваш код просто не скомпилируется.

Важный момент 3:

В Java вы не можете записать выражение вида 18 < age < 65. Ведь у выражения 18 < age будет значение true или false. А сравнивать true < 65 (разные типы) нельзя. По крайней мере, в языке Java.

И что же делать? Ответ на этот вопрос вы найдете в следующей лекции.