3.3

Циклы while и do while

00:00—07:03

С помощью оператора while можно создать бесконечный цикл. Для этого в качестве условия ему нужно передать значение true, и затем в фигурных скобках мы можем написать код, который будет повторяться в нашем цикле. Пока в скобках будет оставаться true, программа не остановится. Остановить её можно только с помощью отключения программы.

```
while (true) {
System.out.println("Текст будет выводится постоянно");
}
```

Оператор while, как и оператор for, можно использовать для создания бесконечных циклов, но его также удобно использовать, когда вы не знаете, сколько итераций вам нужно осуществить.
Примеров из практики может быть много. Например, вы получаете данные с какого-нибудь внешнего источника, не знаете сколько их и выполняете какую-то операцию до тех пор, пока эти данные отдаются. Как только они прекращают отдаваться, вы прекращаете выполнять свой код.
Поскольку с базами данных мы ещё не знакомы и познакомимся с ними в одном из следующих модулей, воспользуемся примером, в котором используются просто числа.
Давайте напишем чуть более сложный код, например, программу, которая будет считать сумму всех переданных ей пользователем чисел до тех пор, пока пользователь не передаст ей число 0.
Для начала создадим переменную, в которой мы будем суммировать все полученные от пользователя числа, назовем её int sum и изначально она будет равна нулю. Затем напишем цикл — пусть в качестве условия в нём пока будет значение true, но чуть

позже мы его заменим. Теперь внутри цикла будем получать число, введённое в консоли, напишем:

```
int value = new Scanner (System.in).nextInt();
```

Далее будем прибавлять это число к числу sum. Теперь нам надо написать условие в цикле таким образом, чтобы проверялось, чему равно введённое пользователем число и цикл выполнялся тогда, когда это число не равно нулю.

По-хорошему, нам надо написать `while (value ! = 0)`, но такой код не будет работать, поскольку переменной value он создаётся внутри цикла и, соответственно, доступен только внутри. Для того, чтобы этот код заработал, вынесем переменную value за пределы цикла и зададим её значение (-1):

int sum=04

```
int value = -1;
while(value!=0){
      value = new Scanner (System.in).nextInt();
      sum= sum + value;
}
```

Мы задаём её значение (-1), т. к. если бы она изначально была равна нулю, то цикл бы даже не начался, поскольку сначала происходит проверка выполнения условия `while (value ! = 0)` и только потом выполняется код внутри фигурных скобок.

Проверяем работу кода и убеждаемся, что всё работает верно и программа прекращает суммирование, как только мы вводим 0.

Аналогичный код можно написать и с помощью оператора for. Для этого:

- вместо while вводим for;
- вводим первое условие в виде int value =-1;
- второе условие — value !=0.

Третий параметр нам не нужен.

Получаем:

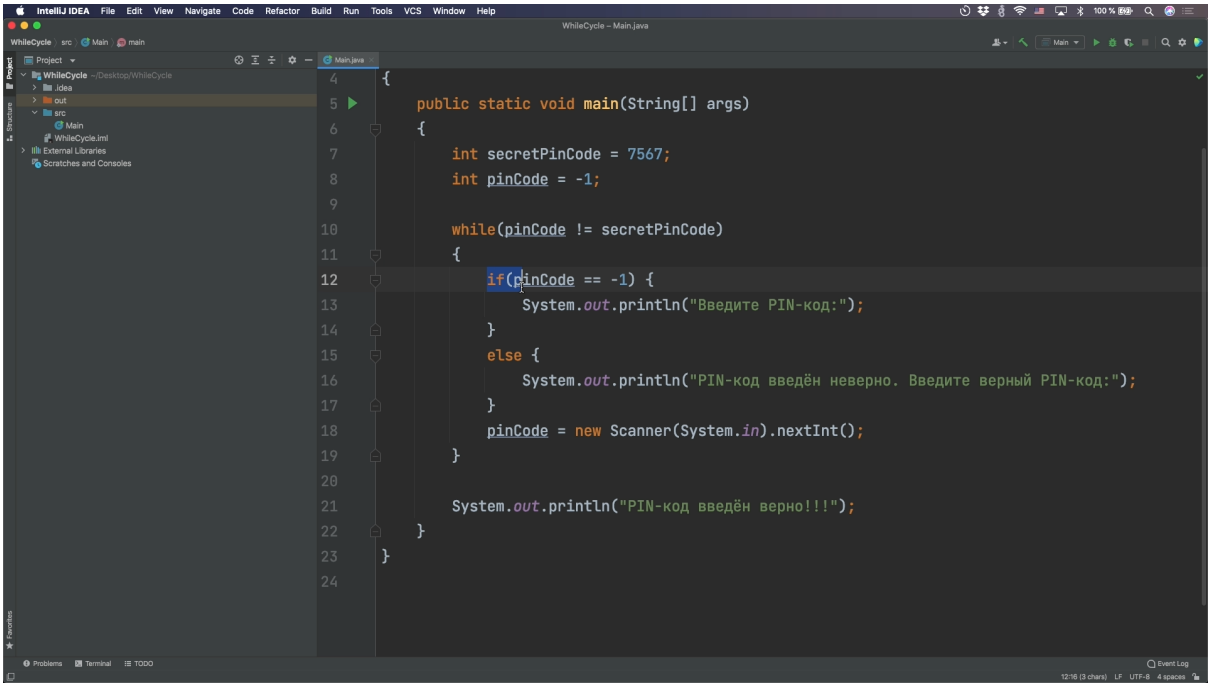
```
for (int value = -1; value != 0; ) {
      value = new Scanner (System.in).nextInt();
      sum= sum + value;
}
```

Такой код работает, но он не очень красивый. Цикл for, в котором есть только некоторые параметры, применять не принято и читать такой код немного тяжелее.

Поэтому в таких случаях удобнее всё же использовать цикл while.

Вот пример другой программы, которая проверяет в консоли некий пин-код, и проверяет его бесконечно, пока он не совпадаёт с верным.

У нас созданы две переменные — с правильным пин-кодом и с пин-кодом, который будет равен тому пин-коду, который ввёл пользователь. Далее выполняется цикл while, который проверяет на каждой итерации равенство двух переменных и выполняется до тех пор, пока они не будут равны.



07:03—09:25

Оператор do while

Теперь давайте рассмотрим ещё один оператор циклов — оператор do while. В кодах он встречается гораздо реже, чем операторы for и while. Оператор while работает так: сначала он проверяет условия, которые у него указаны в круглых скобках, а потом выполняет или не выполняет код, который написан в фигурных скобках.

На практике бывают случаи, когда нужна обратная последовательность — сначала выполнить какой-то код, а потом уже проверять условия.

Синтаксис этого оператора следующий:

- сначала пишется ключевое слово do;
- затем в фигурных скобках пишется тот код, который будет выполняться в цикле;
- затем пишется ключевое слово while и условия в круглых скобках, которые будут проверяться после выполнения этого цикла, и далее ставится «;».

Проверка условия в круглых скобках будет выполняться только после выполнения кода в фигурных скобках. То есть, даже если в круглых скобках вы напишите false, то код в фигурных скобках выполнится один раз. После этого произойдёт проверка условия и, поскольку оно равно false, цикл на этом прервётся, но один раз этот код всё же выполнится.
Когда такой код может пригодиться на практике? Это может пригодиться только в том случае, когда вам нужно сначала выполнить какое-то действие, а потом узнать результат этого действия, а узнать результат этого действия вы можете только после его выполнения.

Возьмём тот же аппарат по изготовлению бургеров. Представим, что по рецепту на булочку должно наноситься 10 грамм соуса, и аппарат измеряет количество соуса по увеличению веса булочки, которая лежит на подставке. При этом часть соуса оседает на стенках трубок, по которым он поступает и, более того, эти трубки периодически очищаются для того, чтобы соус там не застывал. То есть, в итоге, на определённое количество бургеров соуса тратится немного больше, чем если просто умножить количество бургеров на 10. В данном случае правильно будет проверять количество соуса,

которое осталось после изготовления очередного бургера. Если это количество стало меньше, чем 10, — например, 15 грамм, — цикл останавливается.

09:25—09:43
Итоги

Итак, в этом видеоматериале вы познакомились с операторами циклов while и do while и узнали, что они, в отличие от оператора for, применяются, когда вы точно не знаете, какое количество итераций должен выполнять тот или иной код.