Урок 2

Циклы

Понятие и разновидности циклов в Java.

Понятие цикла

Цикл — разновидность управляющей конструкции в высокоуровневых языках программирования, организует многократное исполнение набора инструкций. Цикл заставляет группу действий повторяться до момента выполнения нужного условия. К примеру:

- Опрашивать клиентов, пока не наберем 100 ответов;
- Читать строчки из файла, пока не дойдем до его окончания.

Набор инструкций — это тело цикла. Каждое выполнение тела цикла — итерация. Условие цикла определяет, делать следующую итерацию или завершить цикл. Удобно знать, сколько итераций уже сделано — то есть хранить их число в переменной, которая называется счетчиком итераций цикла (не обязателен для всех циклов).

Каждая итерация состоит из следующих шагов:

1. Инициализация переменных цикла.
2. Проверка условия выхода.
3. Исполнение тела цикла.
4. Обновление счетчика итераций.

В большинстве языков программирования есть инструменты досрочного прерывания всего цикла или выполнения текущей итерации.

Кодовые блоки и области видимости

Мы уже сталкивались с практикой выделения кода в отдельные смысловые составляющие. Рассмотрим подробнее, что они из себя представляют, чтобы перейти к изучению циклов, где они широко применяются.

Кодовый блок — это группа операторов. Для оформления в виде блока они помещаются между открывающей и закрывающей фигурными скобками и становятся единым логическим блоком. В частности, его можно использовать при работе с **if** и **for**. Пример:

```
public static void main(String args[]) { // <- Начало кодового блока main
    int w = 1, h = 2, v = 0;
    if (w < h) {                          // <- Начало кодового блока if
        v = w * h;
        w = 0;
    }                                     // <- Конец кодового блока if
}                                         // <- Конец кодового блока main
```

В данном примере оба оператора в блоке выполняются в случае, если значение переменной **w** меньше значения переменной **h**. Эти операторы составляют единый логический блок, и один не может быть выполнен без другого. Кодовые блоки позволяют оформлять многие алгоритмы в удобном для

восприятия виде. Рассмотрим пример программы, в которой кодовый блок предотвращает деление на ноль:

```
public static void main(String args[]) {
    double a = 5.0, b = 10.0, c = 0.0;
    if (b != 0) {
        System.out.println("b не равно нулю");
        c = b / a;
        System.out.print("b / a равно " + c) ;
    }
}
Результат:
b не равно 0
b / a равно 2.0
```

Области видимости переменных в кодовых блоках:

```
public static void main(String args[]) { // Кодовый блок метода main()
    int x = 10;                           // Эта переменная доступна для всего кода
в методе main
    if (x == 10) {                        // Кодовый блок тела if
        int y = 20;                       // Эта переменная доступна только в
данном кодовом блоке
// Обе переменные, x и y, доступны в данном кодовом блоке
        System.out.println("x & y: " + x + " " + y);
        x = y * 2;
    }
// y = 100; // Ошибка! Переменная y недоступна за пределами тела if
    System.out.println("x = " + x);      // Переменная x по-прежнему доступна
}
```

Циклы while

Цикл while работает до тех пор, пока указанное условие истинно. Как только условие становится ложным, управление программой передается строке кода, следующей непосредственно после цикла. Если заранее указано условие, которое не выполняется, программа даже не попадет в тело цикла.

```
while (условие) {
   набор_операторов;
}
```

Для управления циклом обычно требуется одна или несколько переменных. К примеру, значение **boolean**, которое обращается в **false** при достижении граничного условия; или целочисленное значение, которое каждый раз увеличивается на единицу, пока не достигнет определенного значения.

Цикл **do-while** похож **while**, в котором условие проверяется в начале (предусловие), но здесь условие выполнения проверяется в конце (постусловие). Следовательно, цикл **do-while** всегда выполняется хотя бы один раз.

```
do {
   набор_операторов;
while (условие);
```

Цикл **do...while** используют редко: он громоздкий и плохо читается.

На практике циклы часто применяются для выполнения **N** повторяющихся операций. Реализация **N** повторений через copy–paste — не самая лучшая идея, как и **N** вызовов одного метода с явным повторением в коде. Здесь и применяются циклы. Зачастую в задачах с ними число итераций заранее неизвестно и определяется непосредственно перед инициированием самого цикла.

В следующем видео научимся применять циклы для работы с массивами. А сначала напишем функционал, который будет выводить все четные числа от 0 до числа N, вводимого пользователем:

```
import java.util.Scanner;

public class WhileCycle {
    public static void main(String[] args){
        Scanner userInput = new Scanner(System.in);
        System.out.print("Введите ограничение: ");
        int limit = userInput.nextInt();

        int counter = 0;
        System.out.println("Список четных числел из заданного промежутка:");
        while(counter <= limit){
            if(counter % 2 == 0){
                System.out.println(counter);
            }

            counter++;
        }
    }
}
```

Чтобы цикл функционировал:

- Вводим переменную-счетчик **counter**, которая перед каждой итерацией цикла будет проверять условие продолжения;
- В теле цикла проверяем на четность текущее число, которое будет равно текущему значению счетчика. Это делается простой проверкой: ищем остаток от деления текущего значения счетчика на 2. Если он равен нулю, число четное;
- Не забываем инкрементировать счетчик в конце каждого шага, иначе рискуем никогда не выйти из цикла!