Циклы

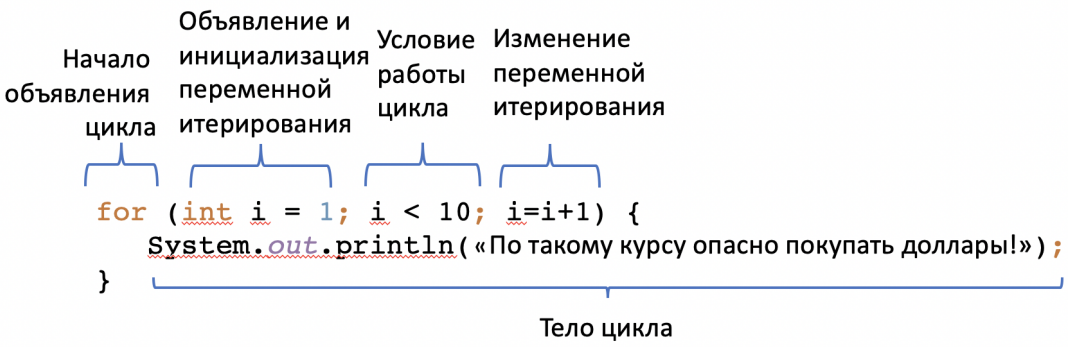
В этой теме вы познакомились с понятием цикл. Научились применять циклы `for` и `while`. Изучили бесконечные циклы и как с ними работать с помощью break. И немного узнали про исключения, которые могут появиться при ошибках ввода пользователем. Вот что нужно запомнить:

Что есть что в цикле

Циклы позволяют выполнять одно и то же действие несколько раз. Существует несколько типов циклов — они начинаются с разных служебных слов и применяются для разных задач.

`for` используется тогда, когда число повторений (итераций) известно заранее.

Синтаксис цикла с `for`:



От трёх параметров в условии цикла зависит откуда он начнет своё движение и сколько раз выполнится тело цикла.

```
public class Praktikum {
    public static void main(String[] args) {

          for (int i = 1; i <= 5; i = i + 1) {
              System.out.println("Байт съел " + i + "-ю морковку.");
          }
```

```
    }
}
```

Во время выполнения этого цикла происходит следующее: при каждой итерации значение `i` увеличивается на единицу `i= i+1`, после чего программа проверяет заданное условие (`i <= 5`) на истинность. Цикл повторяется до тех пор, пока условие истинно. Если требуется увеличить или уменьшить количество итераций, нужно изменить условие цикла.

Выражения `i <= 5` и `i < 6` полностью равнозначны. Учитывайте это при создании циклов.

Переменная итерирования традиционно называется `i`. Она может принимать не только положительные значения или равняться нулю, но и быть отрицательной.

Вариации цикла for

Для того, чтобы запрограммировать цикл `for` в обратном порядке необходимо внести изменения в его условие.

```
for (int i = 9; i > 0 ; i-- ) {
    System.out.println("Этаж " + i);
}
```

Теперь номера этажей отображаются в порядке убывания.

`i` может меняться не только на единицу. Число, на которое после каждой итерации изменяется переменная итерирования называется **шаг цикла**. Оно может быть положительным (цикл по возрастанию) `i = i + 45` и отрицательным (цикл по убыванию)`i = i - 350`.

Операции `i = i - 1` и `i--` равнозначны. Для увеличения переменной `i` на единицу можно также использовать сокращённое выражение `i++`.

Цикл используются не только для вывода строки нужное число раз, но и для расчётов.

```
public class Praktikum {
    public static void main(String[] args) {
        int carrotCount = 0; // Объявляем переменную для общего числа морковок
        int carrotPerDay = 5; // В этой переменной фиксируем ежедневный рацион

        // Число итераций совпадает с количеством дней в неделе
        for(int day = 1; day <= 7; day = day + 1) { // Переменная итерирования - day
            carrotCount = carrotCount + carrotPerDay; // При каждой итерации плюс 5 морковок
        // Сколько морковок Байт съедает за разное количество дней
```

```
            System.out.println(day + "-й день, Байт съест " + carrotCount + " морковок.");
        }
        // Сколько овощей требуется покупать на неделю
        System.out.println("Рацион на неделю: " + carrotCount + " морковок.");
    }
}
```

`carrotCount` — переменная для общего количества моркови.

`carrotPerDay` — переменную для ежедневного рациона хомяка.

`for(int day = 1; day <= 7; day = day + 1)` — цикл установлен на семь итераций (неделя), в ходе которого считаются съеденные хомяком овощи.

`carrotCount = carrotCount + carrotPerDay;` — каждый день прибавляется по пять морковок.

Таким образом можно отследить сколько корнеплодов съедает грызун за разные промежутки времени.

Циклы с условием

Циклы не существуют в программе сами по себе. Их выполнение может зависеть от внешних обстоятельств.Условные выражения и циклы часто используются вместе. Причем, как цикл может находиться внутри условного выражения, так и условное выражение может быть внутри цикла.

```
public class Praktikum {
    public static void main(String[] args) {

        int weight = 750; // Объявили переменную веса Байта и присвоили ей значение
        if (weight < 800) { // Цикл сработает, только если вес Байта меньше 800 грамм
            for (int i = 0; i < 5; i = i + 1) {
                System.out.println("Байт съел " + i + "-ю морковку");
                if (i == 5) {
                    System.out.println("Это была последняя морковка на сегодня");
              }
            }
        } else {
            System.out.println("Разгрузочный день. Пьем водичку, крутим колесо!");
        }
    }
}
```

Внутри цикла помимо условия можно поместить другой цикл — он называется **вложенным**. Вложенные циклы — удобный инструмент. Границ воображению здесь нет — можно «вкладывать» один цикл в другой сколько угодно раз.

```
public class Praktikum {
    public static void main(String[] args) {

        int days = 3; // Количество дней, когда Байт будет тренироваться
        int run = 2; // Число пробежек в день
```

```
        int carrots = 5; // Число морковок после пробежки

        for (int i = 1; i <= days; i++) { // Внешний цикл для смены дней
            System.out.println("День " + i);

            for (int j = 1; j <= run; j++) { // Вложенный цикл для пробежек
                System.out.println(" Пробежка " + j);

                for (int k = 1; k <= carrots; k++) { // Ещё один вложенный цикл для морковок
                    System.out.println("  Морковка " + k);
                }
            }
        }
         System.out.println("Самое время спеть: все идет по планууу!");
    }
}
```

Логика этого кода следующая: сначала запускается внешний цикл, считающий дни. Программа печатает, что наступил первый день тренировки. После этого срабатывает вложенный цикл с пробежками. В каждой итерации этого цикла запускается новый цикл с морковками совершает пять итерации и завершается. После завершения всех повторений цикла с пробежкой запускается внешний цикл — наступил второй день тренировки. И так далее.

Обратите внимание, переменную итерирования мы чаще всего называем `i`. Это не строгое правило, вы можете дать ей любое имя, но `i` — самое распространённое. Для вложенного цикла, в свою очередь, самое популярное имя — `j`.

При объявлении нескольких вложенных циклов переменная итерирования каждого из них должна называться по-своему. Если назвать их одинаково, то программа выдаст ошибку. Также важно помнить об областях видимости таких переменных: они видны в блоке своего цикла и вложенных в него, а за пределами этих циклов — нет.

Цикл while

`for` — подходит для тех случаев, когда заранее известно количество итераций. Но так бывает не всегда. В этом случае работа цикла будет определяться не числом повторений, а булевым выражением. Такой цикл описывается с помощью служебного слова `while` (от англ. *while* — до тех пор, пока).

```
int foodWeight = 500; // Количество корма

while (foodWeight > 10) { // Условие работы цикла - пока осталась хотя бы одна порция
    foodWeight = foodWeight - 10; // Байт съедает 10 грамм корма за раз
}

System.out.println("Корм закончился! Пора идти в магазин!");
```

Цикл `while` используется для расчётов, когда работа программы зависит от внешних данных, которые изначально неизвестны — например, их вводит пользователь.

```
import java.util.Scanner;
public class Praktikum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int dayCount = 0; // Для учёта дней накоплений
        int moneyTotal = 0; // Суммарное количество накоплений
        int moneyToday; // Сколько откладываем сегодня
        int goal = 5000; // Финансовая цель

        while (moneyTotal <= goal) {
            moneyToday = scanner.nextInt(); // Сумма, которую сегодня отложил пользователь
            moneyTotal = moneyTotal + moneyToday; // Добавили эти деньги в копилку
            dayCount = dayCount + 1; // Засчитали день
        }

        System.out.println("Ура! Вы смогли накопить " + goal + " за " + dayCount + " дней.");
    }
}
```

`scanner` — метод для считывания информации из консоли.

Находится в пакете java.util.

`import java.util.Scanner;`

Объявляется переменной `scanner` с типом `Scanner` в методе `main()`.

`Scanner scanner = new Scanner(System.in);`

```
String command = scanner.nextLine(); // Создали переменную command и занесли в нее информацию с клавиатуры
```

`Random` (x) — генератор случайных чисел. Подбирает случайное число в диапазоне от 0 до x.

Находится в пакете java.util.

`import java.util.Random;`

Объявляется переменной `random` с типом `Random` в методе `main()`.

`Random random = new Random();`

```
pilotInput = random.nextInt(100);//Создали переменную pilotInput и занесли в нее случайное значение от 0 до 100
```

Бесконечный цикл

`while` выполняется до тех пор, пока условие является истинным. Следовательно, если в условии написать `true` , то цикл будет выполняться снова и снова. Такой цикл называется **бесконечным.**

> 💡 Если написать в условии цикла `false` и попробовать запустить код, то вообще ничего не получится. Программа выдаст ошибку: `unreachable statement` (англ. «недостижимое утверждение»).

Выйти из бесконечного цикла можно двумя способами, первый — остановить программу. Второй — использовать служебное слово `break` (англ. «перерыв»). На `break` цикл сразу завершится, а программа перейдет к выполнению следующего за блоком цикла кода.

```
while (true) {
    System.out.println("Бегу, бегу, бегу!");
    break; // Цикл завершился
}
 System.out.println("Отлично побегал, пойду поем!"); // Теперь эта строка будет напечатана
```

Бесконечный цикл с прерыванием `break` нужен, когда требуется выполнять условно-неограниченное количество итераций до достижения определенного результата. Это довольно ограниченный спектр задач вроде цифрового меню или передача данных при нестабильном соединении.

Исключения

Пользователи бывают невнимательны и могут ошибиться. Если ввести не верное значение программа выдаст ошибку:

```
Exception in thread "main" java.util.InputMismatchException
  at java.base/java.util.Scanner.throwFor(Scanner.java:939)
  at java.base/java.util.Scanner.next(Scanner.java:1594)
  at java.base/java.util.Scanner.nextDouble(Scanner.java:2564)
  at Praktikum.main(Praktikum.java:13)
```

`Exception` — маркер исключения (англ. «исключение»)

`InputMismatchException` — имя исключения (англ. «несоответствие ввода»).

`Praktikum.main(Praktikum.java:13)` — расположение строки, где возникло исключение строка 13.

Исключения возникают, когда программа уже запустилась, начала работать и вдруг что-то пошло не так. Например, произошло деление на ноль, пользователь вместо числа ввёл строку или попытался открыть файл, которого нет на компьютере.

На этом всё! Вы уже на середине пути, отдохните и подготовьтесь к следующей теме "Массивы".


STOP USING
"i" IN
FOR LOOPS