# Управление порядком выполнения операций

Как вы уже догадываетесь, команды (операции) выполняются друг за другом. Запись операций идет слева направо, сверху вниз. Как уже говорилось, каждая операция оканчивается знаком «;» (бывают исключения и мы их увидим).
Перед тем, как погрузиться в мир команд управления, давайте немного расширим наши возможности по выводу данных. Вот как будет выглядеть команда для вывода символов без перехода на другую строку. Например вывести символ «*».

- Расширенное описание классов
- Исключения
- Решения на основе классов
- Список контактов — начало
- Коллекции — базовые принципы
- Коллекции — продолжение
- Список контактов — GUI приложение
- Что такое JAR-файлы
- Многопоточность — первые шаги
- Многопоточность и синхронизация
- Работаем с XML
- Reflection — основы
- Установка СУБД PostgreSQL
- Базы данных на Java — первые шаги
- Возможности JDBC — второй этап
- JDBC — групповые операции
- Список контактов — работаем с БД
- Переезжаем на Maven
- Потоки ввода-вывода
- Сетевое взаимодействие
- С чего начинается Web

```
1 System.out.print("*");
```

Причем важно отметить, что курсор не будет переходить на следующую строку. А команда для перехода на следующую строку выглядит так:

```
1 System.out.println();
```

Для начала создадим простую программу для вывода квадрата со стороной в 4 символа. Выглядит она не очень красиво, но тем не менее она обладает необходимой функциональностью (я в какой-то степени нарочно сделал ее такой страшненькой).

```
1 public class Sixth
2 {
3     public static void main(String[] arg) {
4         // 4 символа * в строку
```

```
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        // И переход на следующую строку
        System.out.println();
        // Дальше все повторяется
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        System.out.println();
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        System.out.println();
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
        System.out.print("*");
    }
}
```

Существует иной вариант решения

```
public class Sixth
{
    public static void main(String[] arg) {
        System.out.println("****");
        System.out.println("****");
        System.out.println("****");
        System.out.println("****");
    }
}
```

Вы конечно заметили, что наш «квадрат» не совсем квадратный — он больше прямоугольник из-за того, что по вертикали строки больше. Но будем называть его квадратом, т.к. количество символов по вертикали и горизонтали одинаковое.
Как я только что говорил, наша программа в обоих вариантах не самая удачная. Первый вариант длинный и обладает следующим недостатком — если мы захотим сделать квадрат например 7×7, то нам придется добавлять много строк.
Второй вариант конечно короче. Но у него есть тот же недостаток — для рисования квадрата другого размера нам придется все равно менять программу. Да и код выглядит пока страшненько.

Думаю, достаточно очевидно, что требуется конструкция цикла. Для большинства языков программирования таких конструкций бывает несколько. Давайте их и рассмотрим.

## Операторы циклов

В Java существует три способа организации цикла.

### 1. Цикл for

Цикл for использует следующую форму записи

for(<инициализация>; <условие>; <последействие>) операция;

Давайте сразу рассмотрим в качестве примера использования цикла for нашу задачу — нарисовать квадрат.

```
1 public class LoopForOne
2 {
3     public static void main(String[] arg) {
4 
5         for(int i = 0; i < 10; i++)
6             System.out.println("**********");
7 
8     }
9 }
```

Внутри скобок у слова for мы видим три разделенных точкой с запятой части.
int i=0 — этот оператор не должен нас удивить — мы объявили переменную «i» целого типа и присвоили ей значение 0. Этот оператор будет выполняться ТОЛЬКО ОДИН раз в начале цикла.
i<10 — это условия будет проверяться ПЕРЕД КАЖДЫМ циклом. Если условие является истинным (возвращает true) — цикл выполняется. Если false — цикл заканчивается. Выражение может быть достаточно сложным, но мы этим воспользуемся позже.
i++ — этот оператор выполняется ПОСЛЕ КАЖДОГО цикла.

Все достаточно очевидно — мы выводим квадрат со стороной в 10 символов. Переменная «i» в данном случае исполняет роль счетчика — при каждом проходе цикла она увеличивается на 1, сравнивается со значением 10 и если меньше — выполняется операция печати строки из 10 символов.
Как видим, наша программа все-таки не так хороша. При желании вывести квадрат со стороной в 20 символов, нам придется менять значение 10 на 20 и еще изменить строку в печатью — заменить 10 символов «*» на 20.
Давайте сначала попробуем написать программу, которая выводит в ряд столько «звездочек», сколько мы захотим. Это несложно. Вот так:

```
public class LoopForTwo
{
    public static void main(String[] arg) {
        int count = 10;

        for(int i=0; i < count; i++)
            System.out.print("*");
        System.out.println();
    }
}
```

Обратите внимание — мы ввели новую переменную «count» и используем ее в цикле. Сейчас этот шаг может показаться не совсем понятным и нужным, но на самом деле чуть позже мы увидим, почему это удобно. Давайте разберем наш код по шагам. Итак:

int count = 10;

Этот оператор вводит переменную «count» и присваивает ей значение 10. Именно столько символов мы хотим напечатать. Если потребуется изменить количество символов, то будет достаточно просто поменять 10 на что-то другое.

```
for(int i=0; i < count; i++)
    System.out.print("*");
```

Этот цикл напечатает 10 символов «*» в одну строку. Есть отличие между командами System.out.print и System.out.println. Первая выводит символы и не переходит на следующую строку, вторая — напечатает символы и курсор перейдет на следующую строку.
В нашем случае мы напечатаем 10 символов и курсор останется на этой же строке. Если мы захотим напечатать что-то еще, то печать начнется на той же строке. И наконец:

System.out.println();

Эта команда просто переводит курсор на следующую строку

Если собрать наши программы LoopForOne и LoopForTwo вместе, то мы увидим, что в них есть все, что нам нужно — первая умеет печатать нужной количество строк, вторая — нужную нам строку.
Давайте попробуем провернуть это действие. Но сначала нам надо познакомиться еще с одним видом оператора — составной оператор. В другой литературе он может называться как-то иначе — я не могу точно сказать, как он должен называться по науке.
Его идея очень проста — заключив несколько операций в фигурные скобки, их можно рассматривать как один. Давайте опять посмотрим пример.

```
public class LoopForThree
{
    public static void main(String[] arg) {
        int count = 10;

        for(int i=0; i < count; i++) {
            System.out.print("*");
            System.out.println();
        }

    }
}
```

Если запустить это пример, то мы увидим, что он выводит 10 строк и в каждой строке только один символ. Давайте более внимательно посмотрим на наш цикл for. Теперь мы после него «открыли» фигурную скобку

```
for(int i=0; i < count; i++) {     // Последний символ - фигурная скобка
```

А после двух операций (печати символа и перехода на другую строку) — «закрыли». Т.е. теперь внутри цикла выполняется не один оператор, как в предыдущей программе, а два. Как вы уже наверно догадались, в цикле будут выполнятся ВСЕ операции, которые находятся внутри фигурных скобок.
Обратите внимание на два момента:
1. После фигурных скобок точка с запятой не ставится.
2. Достаточно часто даже одна операция в цикле заключается в фигурные скобки — это делает код однообразным и часто облегчает его чтение и понимание.
С учетом второго замечания наша программа LoopForTwo может выглядеть вот так.

```
public class LoopForTwo
{
    public static void main(String[] arg) {
        int count = 10;

        for(int i=0; i < count; i++) {
            System.out.print("*");
        }
        System.out.println();

    }
```

```
}
```

Думаю, вы обратили внимание, что я делаю отступы в программе. Они помогают легче читать код. Советую вам с самого начала привыкать форматировать текст программы — это быстро войдет в привычку и в будущем пригодится. В NetBeans существует команда форматирования текста открытого в редакторе файла — Alt+Shift+F или через меню Source->Format. Можно выделить кусок текста и отформатировать только его.
Теперь давайте выполним нашу главную задумку — напишем программу, которая рисует квадрат с заданной стороной. Это потребует от нас совместить два цикла — один цикл для строк, другой — для символов в строке. Итак:

```
public class LoopForFour
{
    public static void main(String[] arg) {
        int count = 10;

        // Внешний цикл
        for(int i=0; i < count; i++) {
            // Внутренний цикл для печати одной строки
            for(int k=0; k < count; k++) {
                System.out.print("*");
            }
            // Переход на следующую строку
            System.out.println();
        }

    }
}
```

Здесь мы видим более сложную конструкцию, состоящую из двух циклов. Причем один «вложен» в другой. Давайте разберем сначала внутренний цикл, который связан с переменной «k».

```
for(int k=0; k < count; k++) {
    System.out.print("*");
}
```

Его задача очень простая — он печатает строку из count символов. И если мы поместим его еще в один цикл, то он будет послушно печатать строку столько раз, сколько ему будет сказано.
Задача внешнего цикла (с переменной «i») заключается в том, чтобы сначала вызвать цикл для печати строки (с переменной «k») а после этого перейти на следующую строку (операция System.out.println(). Теперь еще раз самое время взглянуть на нашу программу еще раз.

```java
public class LoopForFour
{
    public static void main(String[] arg) {
        int count = 10;

        // Внешний цикл
        for(int i=0; i < count; i++) {
            // Внутренний цикл для печати одной строки
            for(int k=0; k < count; k++) {
                System.out.print("*");
            }
            // Переход на следующую строку
            System.out.println();
        }

    }
}
```

Надеюсь, что вы разобрались. Обратите внимание, что наша переменная count уже начинает нам помогать. Теперь для того, чтобы изменить размеры квадрата, нам надо поменять только ее значение. А циклы уже будут использовать ее значение для своей работы.
Давайте напоследок нарисуем более сложную фигуру — треугольник. Сначала я приведу код программы и вы можете попробовать понять, за счет чего рисуется именно треугольник. А потом можете прочитать мои подсказки.

```java
public class LoopForFive
{
    public static void main(String[] arg) {
        int count = 10;

        // Внешний цикл
        for(int i=0; i < count; i++) {
            // Внутренний цикл для печати одной строки
            // Обратите внимание на выражение k < i+1
            // В нем вся хитрость треугольника
            for(int k=0; k < i+1; k++) {
                System.out.print("*");
            }
            // Переход на следующую строку
            System.out.println();
        }

    }
}
```

Если вы внимательно смотрели код, то могли видеть подсказку — в комментарии по поводу k<i+1.
Именно в этом выражении и кроется решение — горизонтальный размер строки зависит от того, какая это строка по счету. У самой первой строки длина будет 1, у второй — 2 и так далее.

Цикл for совершенно необязательно должен иметь все три части внутри скобок. Вы можете написать вот такое выражение

```
int i=0;
for (; i < 10; i++) {
    System.out.print("#");
}
System.out.println();
```

Как видите, первая часть у нас отсутствует — мы ее сделали строчкой выше. А можете так:

```
for (int i = 0; i < 10; ) {
    System.out.print("#");
    i++;
}
System.out.println();
```

Здесь мы убрали третью часть и сделали увеличение переменной внутри цикла. Можно сделать и так:

```
int i=0;
for (; i < 10; ) {
    System.out.print("#");
    i++;
}
System.out.println();
```

Цикл for можно вообще сделать пустым — вот таким.

```
for (;;) {
    System.out.print("#");
```

```
}
```

Правда тогда этот цикл будет длиться бесконечно. И он не так уж редко встречается в программах.
Мы очень подробно рассмотрели цикл for, но кроме него есть еще два варианта циклов. Давайте обратимся к ним.

**2. Цикл с предусловием while**

Формат записи этого цикла

while(<условие>) операция;

Пока условие является истинным — цикл выполняется. Давайте сразу напишем простой пример для вывода строки символов произвольной длины.

```
public class LoopWhileFirst
{
    public static void main(String[] arg) {
        int count = 10;

        int i=0;
        while(i < count) {
            System.out.print("#");
            i++;
        }
        System.out.println();

    }
}
```

Как видите с одной цикл пишется проще — там всего лишь одно условие. С другой стороны — мы вынуждены сами менять нашу переменную «i» внутри цикла — кстати нередко об этом забывают и программа входит в бесконечный цикл. Хотя конструкция для бесконечного цикла не такая уж редкая вещь — его просто прерывают специальной конструкцией, о которой мы еще поговорим.
Хотя используя оператор ++ программу можно немного упростить (и даже избавиться от составного оператора в данном случае, но я вам не советую).

```
public class LoopWhileTwo
{
    public static void main(String[] arg) {
```

```
        int count = 10;

        int i = 0;
        // Именно так - i++. Если так: ++i, то символов будет меньше
        // Почему - догадайтесь сами
        while (i++ < count) {
            System.out.print(i);
        }
        System.out.println();

    }
}
```

Название «с предусловием» должно быть понятно — цикл будет выполняться после проверки условия — т.е. условие будет перед циклом.
Давайте с помощью этого цикла выполним нашу задачу — нарисуем квадрат. А для самостоятельной работы попробуйте нарисовать треугольник.

```
public class LoopWhileThree
{
    public static void main(String[] arg) {
        int count = 10;

        int i = 0;
        while (i < count) {
            int k = 0;
            while (k < count) {
                System.out.print("*");
                k++;
            }
            System.out.println();
            i++;
        }
    }
}
```

Если вам не очень нравятся отдельные операторы i++ и k++ — это дело вкуса — можно привести все к виду как в примере LoopWhileTwo.

### 3. Цикл с постусловием do .. while

Формат записи этого цикла

do операция; while(<условие>);

Т.е. делать операцию до тех пор, пока условие не станет ложным. И сразу (уже по традиции) пример рисования строки

```
public class LoopDoOne
{
    public static void main(String[] arg) {
        int count = 10;

        int i = 0;
        do {
            System.out.print("*");
            i++;
        } while(i < count);
        System.out.println();

    }
}
```

И снова мы можем упростить наш цикл используя оператор ++

```
public class LoopDoTwo
{
    public static void main(String[] arg) {
        int count = 10;

        int i = 0;
        do {
            System.out.print("*");
        } while (++i < count);    // Вот здесь наш оператор ++
        System.out.println();

    }
}
```

Думаю, что вам не составило труда понять, почему это оператор с постусловием — условие проверяется ПОСЛЕ цикла. В отличии от оператора while цикл do выполнится как минимум один раз. Даже если условие будет ложным. Можете установить значение count равным -10 и убедиться, что цикл while не выводит ни одного символа, а цикл do выводит один символ.

А теперь задачка для очень и очень внимательных. Попробуйте понять, что делает это код (подсказка — вспомните про пустой оператор)

```
public class CheckFirst {

    public static void main(String[] arg) {
        int count = 10;

        int i = 0;
        while (i++ < count) {
            System.out.print("*");
        } while (++i < count);

        System.out.println();
    }
}
```

Правда забавно выглядит ? Такое ощущение, что у нас цикл while совмещен с циклом do. Но на самом деле это не совсем так. Давайте немного отформатируем наш текст.

```
public class CheckFirst {

    public static void main(String[] arg) {
        int count = 10;

        int i = 0;
        // Обычный цикл while для вывода строки
        while (i++ < count) {
            System.out.print("*");
        }
        // Цикл while с пустым оператором
        while (++i < count) ;  // ; - и есть пустой оператор

        System.out.println();
    }
}
```

Т.е. мы сделали «ничего» во втором цикле while.

## 4. Операторы управления циклом — break и continue

При выполнении цикла вы можете использовать еще две очень удобные конструкции, которые позволят вам строить интересные конструкции. Оператор **break** позволяет прервать цикл, а оператор **continue** позволяет пропустить остаток операторов в цикле и сразу начать цикл заново.

#### Оператор break

Оператор break в простой форме позволяет прервать исполнение цикла и перейти к оператору, расположенному сразу после цикла.
Рассмотрим вариант использования оператора break. Напишем несложную программу, которая сосчитает количество чисел, сумма которых не должна первышать 300 и каждое следующее больше предыдущего на 10. Начальное число будет равно 1. У меня получилось 8 чисел (1, 11, 21, 31, 41, 51, 61, 71 — сумма 288).

```
public class BreakFirst {

    public static void main(String[] arg) {
        int number = 1;    // Начальное число
        int sum = 0;       // Переменная для хранения суммы
        int count = 0;     // Количество чисел - в начале ни одного
        int max = 300;     // Предел суммы

        for(;;) {
           sum += number;     // Увеличиваем сумму
           if(sum > 300) {    // Проверяем сумму - если больше
               break;             // То эта точка является точкой выхода
           } else {
               count++;           // А здесь просто увеличиваем количество чисел
           }
           number += 10;     // Увеличиваем число на 10
        }
        System.out.println("Count=" + count);
    }
}
```

Как видим, все достаточно просто — при сумме больше 300 мы просто вызываем команду break. И нас «перебрасывает» на оператор System.out.println(«Count=» + count);.
Кроме этого можно обратить внимание на конструкцию оператора for(;;). Если бы мы не использовали оператор break, то цикл стал бесконечным.
В этой части мы пока не будем рассматривать более сложную конструкцию оператора break, которая позволяет не просто выйти из оператора цикла (обращаю ваше внимание, что можно выйти не только из цикла for, но и других циклов), но и указать насколько «высоко» мы хотим выйти — если вы находитесь внутри нескольких циклов. Эту конструкцию предлагаю оставить на более позднее время. Во-первых это не часто используется. А во-вторых — не хочется прямо сейчас слишком усложнять.

**Оператор continue**

Оператор continue позволяет вам не прервать цикл, а перейти на начало цикла не исполняя операции после этого оператора. Например, давайте напечатаем только нечетные числа от 0 до 20. Конечно вы уже можете это сделать и без оператора continue, но попробуем использовать его.

```
public class ContinueFirst {

    public static void main(String[] arg) {

        for(int i=0; i < 20; i++) {
           if(i%2 == 0) {     // число четное -
               continue;        // переходим в начало цикла
           }
           System.out.println("Number=" + i);    // оператор исполняется только для нечетных чисел
        }

    }
}
```

Как видим, все достаточно просто — мы проверяем, является ли число четным и если это так, выполняем оператор continue.
Как и в случае с оператором break есть более сложные варианты использования, но по уже указанным соображениям мы пока не будем о нем говорить.

## Условный оператор

Появление условного оператора достаточно понятно — не всегда надо выполнять ту или иную операцию. Например, в одном случае надо печатать, а в другом — не надо. Иногда надо печатать один символ, а иногда — другой. Если опять начинать с примеров (я очень люблю примеры — там гораздо все понятнее, чем теоретические рассуждения), то давайте нарисуем наш квадрат, но теперь сделаем его сложнее — мы его сделаем пустым внутри. Вот таким:

```
**********
*        *
*        *
*        *
```

```
*          *
*          *
*          *
*          *
*          *
**********
```

Выглядит это конечно не квадратом, но как мы уже говорили — количество символов в строке и количество строк у нас совпадают — в этом заключается «квадратность» нашей фигуры.
Подход к рисованию такой фигуры будет такой же, как и при рисовании заполненного квадрата. но только теперь нам надо в одном случае рисовать символ «*», а в другом — пробел. Во такой » «. Итак, у нас явно возникает потребность в проверке некоторого условия, по которому мы будем рисовать или пробел или «звездочку».

Давайте по традиции приведем формат записи условного оператора.

if (<условие>) операция1; else операция2;

Т.е. если условие истинно, то выполняется операция1, иначе — операция2. Есть упрощенная форма:

if (<условие>) операция;

Если условие истинно (возвращает true), то операция выполнится. Если ложное — ничего не выполнится. А точнее будет выполняться следующая операция после условного оператора. Например

```
int a = 15;
int b = 25;

if (a > b)
    System.out.println("a > b");
else
    System.out.println("a <= b");
```

Подозреваю, что здесь все достаточно очевидно. Если переменная A больше B, то напечатаем строку «a > b». Иначе — «a <= b». Т.к. я очень люблю расставлять скобки — я бы записал это так

```
int a = 15;
```

```
int b = 25;

if (a > b) {
    System.out.println("a > b");
} else {
    System.out.println("a <= b");
}
```

Со скобками мы уже сталкивались — это дает нам возможность исполнять несколько операторов по условию. Давайте сразу приступим к делу — будем рисовать наш квадрат. По сути наша задача заключается в изменении алгоритма печати символа «*». Все остальное можно оставить как есть. Условием печати «*» будет совпадение значение переменных цикла i и k с крайними значениями. А это либо 0, либо count-1 (именно count-1, а не просто count)

```
public class LoopForWithIfOne
{
    public static void main(String[] arg) {
        int count = 10;

        // Внешний цикл
        for(int i=0; i < count; i++) {
            // Внутренний цикл для печати одной строки
            for(int k=0; k < count; k++) {
                // Вот наше сложное условие
                if(k==0 || k==count-1 || i==0 || i==count-1) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            // Переход на следующую строку
            System.out.println();
        }

    }
}
```

Вся идея заключается в нашем сложном условии

k==0 || k==count-1 || i==0 || i==count-1

Если пытаться его читать на человеческом языке, то получится следующее:
«k» равно нулю ИЛИ «k» равно count-1 ИЛИ «i» равное нулю ИЛИ «i» равно count-1

Вуаля. Мы сделали это. И ведь это было не так уж и трудно. Но у нас и фигура была не столь проблематичная. Теперь давайте закрепим наши знания путем создания программ. Итак, сделаем теперь треугольник, который мы сделали для демонстрации циклов for, но с усложнением — сделаем его тоже «полым». Приведу сразу код программы — читайте и пробуйте. Там ничего сложного нет.

```
public class LoopForWithIfTwo
{
    public static void main(String[] arg) {
        int count = 10;

        // Внешний цикл
        for (int i = 0; i < count; i++) {
            for (int k = 0; k < i + 1; k++) {
                // Здесь условие даже немного проще
                if (k == 0 || k == i || i == count - 1) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            // Переход на следующую строку
            System.out.println();
        }

    }
}
```

А теперь для самостоятельной работы я предлагаю вам нарисовать вот такие фигуры:

```
**********
**        **
* *      * *
*  *    *  *
*   * *   *
*    *    *
*   * *   *
*  *    *  *
* *      * *
**        **
**********

          *
         **
```

```
         * *
        *  *
       *   *
      *    *
     *     *
    *      *
   *       *
  *        *
 ***********

      *
     * *
    *   *
   *     *
  *       *
 *         *
***********

      *
     * *
    *   *
   *     *
  *       *
 *         *
*           *
 *         *
  *       *
   *     *
    *   *
     * *
      *
```

Я вам настоятельно советую сделать хотя бы одну фигуру. Самостоятельно. Это только сначала кажется, что все понятно и писать легко и просто — вам просто необходимо сделать самому хоть что-то — иначе можно считать наше знакомство с Java весьма поверхностным.
Как вы думаете, почему в компаниях спрашивают людей с опытом работы ? Ведь на самом деле они тоже не помнят все конструкции назубок и не всегда знают ответ на достаточно простые вопросы. Но эти люди писали код, отлаживали его, запускали, искали ошибки — накопили опыт. Так вот эти программы (пусть и простые) — это ваш опыт. Накапливайте его как можно больше.

## Множественный выбор

Давайте попробуем написать программу, которая делает несложную, но достаточно нудную операцию — в зависимости от числа (от 1 до 10) выводит его название.

```
```

```
public class SwitchFirst {

    public static void main(String[] arg) {
        int number= 5;

        if(number==1) {
           System.out.println("One");
        } else if(number==2) {
           System.out.println("Two");
        } else if(number==3) {
           System.out.println("Three");
        } else if(number==4) {
           System.out.println("Four");
        } else if(number==5) {
           System.out.println("Five");
        } else if(number==6) {
           System.out.println("Six");
        } else if(number==7) {
           System.out.println("Seven");
        } else if(number==8) {
           System.out.println("Eight");
        } else if(number==9) {
           System.out.println("Nine");
        } else if(number==10) {
           System.out.println("Ten");
        } else {
           System.out.println("Unknown");
        }

    }
}
```

На практике ситуация при которой необходимо осуществлять выбор из множества вариантов встречается достаточно часто и для этого был создан оператор множественного выбора. Его форма записи вот такая:

```
switch(<переменная>) {
    case <значение1> : <операция1>;
    case <значение2> : <операция2>;
    case <значение3> : <операция3>;
    case <значение4> : <операция4>;
    default: <операция>;
}
```

Все очень просто — если значение переменной совпадает со значением1, то выполняется операция1. Кажется все достаточно понятно. Если не подходит ни одно из перечисленных значений — выполняется операция под словом default. Не сложно и по первости все наталкиваются на весьма неприятную особенность. Давайте выполним нашу программу:

```
public class SwitchSecond {

    public static void main(String[] arg) {
        int number = 50;    // Проверяемое число

        switch(number) {
           case 1: System.out.println("One");
           case 2: System.out.println("Two");
           case 3: System.out.println("Three");
           case 4: System.out.println("Four");
           case 5: System.out.println("Five");
           case 6: System.out.println("Six");
           case 7: System.out.println("Seven");
           case 8: System.out.println("Eight");
           case 9: System.out.println("Nine");
           case 10: System.out.println("Ten");
           default: System.out.println("Unknown");
        }

    }
}
```

Запускаем нашу программу и видим слово Unknown — все правильно. Число 50 не входит в наши значения. Теперь подставим число 5. Наша программа выдаст несколько неожиданный результат:

```
Five
Six
Seven
Eight
Nine
Ten
Unknown
```

Мы добрались до строки, но дальше мы стали выполнять все операции подряд. Да, эта особенность оператора выбора. Для того, чтобы прервать операции, необходимо использовать оператор break. Вот так должна выглядеть наша программа:

```java
public class SwitchThird {

    public static void main(String[] arg) {
        int number= 5;

        switch(number) {
           case 1:
               System.out.println("One");
               break;
           case 2:
               System.out.println("Two");
               break;
           case 3:
               System.out.println("Three");
               break;
           case 4:
               System.out.println("Four");
               break;
           case 5:
               System.out.println("Five");
               break;
           case 6:
               System.out.println("Six");
               break;
           case 7:
               System.out.println("Seven");
               break;
           case 8:
               System.out.println("Eight");
               break;
           case 9:
               System.out.println("Nine");
               break;
           case 10:
               System.out.println("Ten");
               break;
           default:
               System.out.println("Unknown");
        }

    }
}
```

Оператор выбора позволяет выполнить одну и ту же операцию для нескольких значений. Давайте посмотрим пример:

```java
public class SwitchFourth {

    public static void main(String[] arg) {
        int number= 1;

        switch(number) {
           case 1:
           case 2:
               System.out.println("One or Two");
               break;
           case 3:
               System.out.println("Three");
               break;
           default:
               System.out.println("Unknown");
        }

    }
}
```

Здесь можно видеть, что мы можем выполнить операцию System.out.println(«One or Two»); как для значения 1, так и для значения 2.
Под словом switch может находится переменная целочисленных типов — byte, short, int. Тип long использовать нельзя.
Начиная с версии 1.7 можно использовать тип String (строки), но на это мы еще раз обратим внимание, когда дойдем до работы со строками.

Мы рассмотрели все основные конструкции для управления ходом программы. Нас ждут еще некоторые открытия на этом пути, но бОльшая часть пройдена. Так что постарайтесь решить те задачи, которые я предлагал.
Перед тем, как окунуться в изучение объектно-ориентированного программирования (ООП) я предлагаю вам установить IDE NetBeans — чтобы облегчить разработку наших программ. Встречайте Установка IDE NetBeans.

## 72 comments to *Порядок операций*

- 
  Май 18, 2015 at 01:09
  *VyaChesLav* says:

Отличный сайт. Просто супер, доступное обьяснение. эхх, не надо было мне прогуливать пары программинга, слава богу наткнулся на бескрайних просторах «О Великого» на этот замечательный сайт. админ, спасибо огрромное. всем советовать буду, в вк на страничке ссыль выложу. а если лавешка лишняя появится то переведу в помощь сайту. в общем класс!!

Reply

- 
Июль 6, 2015 at 09:14
*Grif* says:

Спасибо за Ваш труд! Посоветовали начать обучение с Вашего сайта, мне все нравится.

Есть небольшая недоработка в первом примере текущей темы —

«int max = 300; // Предел суммы

for(;;) {
sum += number; // Увеличиваем сумму
if(sum > 300) { // Проверяем сумму — если больше»

Вы случайно про переменную забыли.

Reply

- 
Июль 6, 2015 at 14:22
*Grif* says:

Вставлю и свои 5 копеек 🙂
Вот такой вот ромб 🙂

int count = 11;
int smh = 0;
for(int i=0; i < count; i++)
{
if(i<count/2)
{smh = count/2 + i;}

```
else
{smh = i-count/2;}
for(int k=0; k < count; k++)
{
if(k==smh || k==count-smh-1)
{System.out.print("*");}
else
{System.out.print(" ");}
}
System.out.println();
}
```

Reply

Июль 16, 2015 at 19:10
*Гарик* says:

в LoopForFive прога печатает треугольник

Reply

Июль 16, 2015 at 19:10
*Гарик* says:

извиняюсь не треугольник а строку

Reply

Август 4, 2015 at 18:28
*Art* says:

Добрый день!
В первую очередь хочу поблагодарить Вас за проделанную работу для огромного количества людей. Я решил обучаться программированию на Java с помощью вашего курса (ранее языки не изучал, решил поменять профессию).

Мне тяжеловато дается программирование, но я очень стараюсь.
Проходя очередной урок: «Порядок операций» (http://java-course.ru/begin/operations/), я сам себе дал задание, для того, чтобы закрепить урок: Написать программу, которая считает сумму нечетных чисел и если сумма больше или равна например max=25, то выводится фраза «Uknown number», если меньше, то выводится «Sum Odd Numbers = » В итоге получается, что выводит и то и другое. Уже 2 дня пытаюсь решить, не получается и спросить не у кого. Предполагаю,что ответ на поверхности) Подскажите, пожалуйста. Самая лучшая версия программы:

```
public class MyProgram {

public static void main(String[] args) {
int number=12; // Отсюда берем нечетные числа
int sum=0;
int max=26; // Предельная сумма

for(int i=1; i 0) {
sum +=i; {
if(sum >= max) {
System.out.println(«Unknown number»);
break;
}
}
} else {
System.out.print(«»);
}

} System.out.println(«Sum Odd Numbers = » +sum);
}
}
```

Reply

- 
  Август 4, 2015 at 18:33
  *Art* says:

строка for(int i=1; i 0)
Почему-то отобразилось в сообщение не верно(

Reply

Август 4, 2015 at 18:39
*Art* says:

опять(( Напишу текстом: for(int i=1; i меньше number; i плюс плюс) {
Если(i%2 больше 0)
дальше идет строка sum +=i;

P.S. Странно, почему не корректно копируется?

Reply

Август 5, 2015 at 23:46
*Grif* says:

Тут цензор настроен своеобразно — это минус сайта, но здесь толковые курсы — этот плюс перевешивает предыдущий минус. Я такой же пользователь как и Вы тоже почти с нуля обучаюсь. Если есть желание оставьте свой почтовый адрес будем переписываться.

Reply

Август 7, 2015 at 16:48
*Grif* says:

Ок, выслал Вам сообщение со своего ящика.

Reply

Февраль 25, 2018 at 14:33
*RomulFobos* says:

Привет. Не понял как вы почтой обменялись? Вроде почтовых адресов не видно o_O
Хотел спросить, как успехи спустя 2,5 года?
И как ты додумался до такого ромба, как ты прислал выше?
Вот мой (по сравнению с твоим, мой очень стремный)

```
//Ромб
if (count % 2 != 0){
count += 1;
}
System.out.println(«Ромб»);
System.out.println();
for (int i = 0; i <= count; i++){
for(int j = 0; j <= count; j++){
if ((i count/2) && ((j == i — count/2) || (j == i + count/2 — 2 * (i — count/2)))){
System.out.print(«*»);
}
else {
System.out.print(» «);
}
}
System.out.println();
}
```

Reply

Ноябрь 15, 2015 at 03:02
*_ALX_* says:

Недавно решился взяться за Java. Вот сижу читаю этот замечательный курс, и вижу эту задачку про таблицу умножения. Интересно стало попробовать, и вот что получилось:

```
public class MultiplicationTable
{
public static void main(String[] arg)
{
int a = 10;
int x = 0;
while (x++ < a)
{
int y = 0;
while (y++ < a)
{
int z = 0;
z = x*y;
if (z < 10)
{
System.out.print(" ");
}
System.out.print(z + " ");
}
System.out.println();
}
}
}
```

Reply

- 

  Ноябрь 16, 2015 at 03:14
  *messi* says:

  Друзья, подскажите, пож., как нарисовать прямоугольник из звездочек (внутри пробелы) только с одним циклом. Заранее спасибо.

  Reply

  - 

    Ноябрь 16, 2015 at 09:49

*admin* says:

Сделать цикл по одной переменной — только предел будет не N, а N*N и при достижении < счетчик>%N == 0 переходить на новую строку.

Reply

- 
Январь 14, 2016 at 21:26
*Игорь* says:

Ну как же так можно!? Ввел огромный комментарии. 20 минут его печатал. Нажал ДОБАВИТЬ и на следующей странице меня известили, что время ожидания капчи истекло и нужно перейти на предыдущую страницу. Перехожу и комментари как не бывало=C=C=C=C

Reply

  - 
Январь 15, 2016 at 11:06
*admin* says:

Могу только посопереживать. Если не делать капчу, то спам просто зашкаливает. По 200-300 сообщений в сутки, бывало и больше.

Reply

- 
Январь 14, 2016 at 21:43
*Игорь* says:

Превосходный ресурс и доступнейшим образом изложена информация по JAVA.
Но вот уже три дня я никак не могу вникнуть в формулы в «спейиальном условии» для создания фигур. Мозг уже взрывается. Прочитал все комментарии и ознакомился со всеми решениями, пялился в каждое по несколько часов и ве равно не понял=(
К примеру — public class hh {
public static void main(String[] arg) {
int x = 11;
int y = 11;

```
for (int i = 0; i < x; i++) {
for (int k = 0; k < y; k++) {
if (k==y/2-i || k==y/2+i || k==i-y/2 || k==x-i+x/2-1) {
System.out.print("*");
} else {
System.out.print(" ");}}
System.out.println();}}}
```

Возьмем, к примеру, первую формулу из "специального условия" — k==y/2-i
При условии, что условие истинно печатается "*".
НО ОБЪЯСНИТЕ МНЕ ПОЖАЛУЙСТА, КАК ИЛИ В КАКОЙ МОМЕНТ k==y/2-i МОЖЕТ БЫТЬ ИСТИННО, ЕСЛИ k==y/2-i ЭТО — 0(или 1,2,3 и т.д.)==11/2-0(или 1,2,3 и т.д.), А ТО ЕСТЬ 0==5.5 (или 1==4.5 или 2==3,5 или 3==3,5).
ПОМОГИТЕ ПОЖАЛУЙСТА ПОНЯТЬ И ОСОЗНАТЬ ДАННОЕ ЧУДО)

Reply

- 
  Январь 15, 2016 at 11:08
  *admin* says:

  Для начала хотелось бы понять, какую именно фигуру ты пытаешься нарисовать. От этого будет зависеть, какие условия должны быть.

  Reply

  - 
    Январь 16, 2016 at 16:19
    *Игорь* says:

    Например РОМБ. В комментарии я именно к РОМБУ дал решение, в свою очередь, позаимствоанное в комментариях выше.
    Мне понятно, как рисовать квадрат, и полый и нет. Также понятно, как рисовать квадрат с крестом внутри и треугольник с прямым углом. А вот на счет равнобедренного треугольника и ромба не ясно.
    Помогите пожалуйста понять формулы в специальном условии.
    То, как я их осознаю, не дает мне понять как же они вообще могут являться истинными.

    Reply

Январь 18, 2016 at 10:00
*admin* says:

Присылайте свои поделки на почту — здесь не так удобно. course@java-course.ru

Reply

Январь 16, 2016 at 21:14
*Алекс* says:

Спасибо за интересные задачки с ромбом оказалась самая интересная

Reply

Июль 6, 2016 at 17:22
*Максим* says:

Здравствуйте, сайт очень хороший, данный на сайте материал понять смог, но вот никак не могу понять как выходит ромб и как сделать другие фигуры, очень нужна помощь

Reply

Июль 6, 2016 at 17:57
*admin* says:

Может для начала сделать треугольник ?

Reply

Август 10, 2016 at 14:05
*Сергей* says:

Я сижу, смотрю на задания, и у меня нет даже тени мысли, как это можно было бы сделать.

Reply

Август 10, 2016 at 14:07
*Сергей* says:

буду думать пока не решу.

Reply

Август 25, 2016 at 05:16
*Евгений* says:

Здравствуйте. Для задания — нарисовать треугольник, у которого высота 6 символов, ширина 11 символов написал следующее:

```
public class ThreeAngle
{
public static void main(String[] arg){
int caunt = 11;
for(int i = 0; i < caunt; i++){
if(i < caunt/2)
continue;{
for(int k = 0; k < caunt; k++){
if(i == caunt — 1 || k == caunt — 1 — i || k == i )
System.out.print("*");
else
System.out.print(" ");
}
```

```
System.out.println();
}
}
}
}
```
Всё нарисовалось. Но кажется мне, что есть альтернативные варианты кода для этой фигуры на основе пройденного этапа обучения. Кому не трудно — поделитесь, пожалуйста. Очень любопытно.

Reply

Февраль 19, 2017 at 17:01
*Юрий* says:

```
public class If_Equ_Triang {
public static void main(String[] args) {
int count = 6;

for (int i = 0; i < count; i++) {

for (int k = 0; k < i + 6; k++) {
if (i == count — 1 || k == count — 1 — i || k == i + 5) {
System.out.print("*");

} else {
System.out.print(" ");
}

}

System.out.println();

}
}
}
```

Reply

Февраль 19, 2017 at 17:24
*Юрий* says:

```
public class If_Equ_Triang {
public static void main(String[] args) {
int count = 9;

for (int i = 0; i < count; i++) {

for (int k = 0; k < i+count; k++) {
if (i == count — 1 || k == count — 1 — i || k==i+count-1 ) {
System.out.print("*");

} else {
System.out.print(" ");
}

}

System.out.println();

}
}
}
```

надо было так написать, не было бы заметно, что подогнал))

Reply

Август 25, 2016 at 05:41
*Евгений* says:

Проверил, а мой метод при чётном caunt должным образом не работает. Если дописывать для чётного, то будет громоздким. Ищу другой путь.

Reply

Август 25, 2016 at 09:27
*admin* says:

Ищите — и удачи.

Reply

Февраль 19, 2017 at 17:04
*Юрий* says:

Уже много времени прошло, люди поломали голову, скиньте правильный ответ, как написать эти треугольник с ромбом)… я написал треугольник, но ромб по такой технологии не выйдет у меня)

Reply

Август 25, 2016 at 14:00
*Евгений* says:

Всё же мой метод правильный, просто для чётного caunt фигура немного иначе выглядит — с двумя символами в вершине — это меня и смутило.

Reply

Август 30, 2016 at 13:22
*RomulFobos* says:

Добрый день, Антон.
Я не понял вот эту часть:
// Именно так — i++. Если так: ++i, то символов будет меньше
// Почему — догадайтесь сами
while (i++ < count) {
System.out.print(i);
Почему получается разный результат? Во всех двух случаях, мы увеличиваем переменную i до сравнения с переменной count.

Reply

Август 30, 2016 at 14:07
*admin* says:

Вы делаете неверное предположение. В одном случае увеличение происходит до сравнения, в другом — после.

Reply

Август 30, 2016 at 14:25
*RomulFobos* says:

То есть, если ++ стоит после переменной, то вначале выполнятся все остальные операторы и только потому i увеличиться на единицу?

Reply

Август 30, 2016 at 15:12
*admin* says:

Да

Reply

Февраль 5, 2017 at 13:07
*Виктор* says:

первая фигура:
package javaapplication2;

public class JavaApplication2 {

public static void main(String[] args) {
int count = 10;
for (int i = 0; i < count; i++) {
for (int k = 0; k < count; k++) {
if (i == count — 1 || i == k || k == count — 1 || k == 0 || i == 0 || k == count — i — 1) {
System.out.print("*");
} else {
System.out.print(" ");
}
}
System.out.println();
}
}
}

Reply

Февраль 16, 2017 at 14:42
*Antofagasto* says:

Что то у меня 11 ошибок выдает)

Reply

Февраль 17, 2017 at 23:11
*Antofagasto* says:

Данный оператор k==count-1 значит: если k равно count минус 1?

Reply

Февраль 19, 2017 at 21:29
*admin* says:

Да

Reply

Февраль 18, 2017 at 00:26
*Antofagasto* says:

Зачем в данном случае использовать команду System.out.println(); если ее убрать, ничего не изменится)

```
public class LoopDoOne
{
public static void main(String[] arg) {
int count = 10;

int i = 0;
do {
System.out.print(«*»);
i++;
} while(i < count);
System.out.println();
```

}
}

Reply

- 
  Февраль 19, 2017 at 21:30
  *admin* says:

  Без System.out.println(); курсор не перейдет на другу строку — в данном случае это просто более эстетично. Для задачи в общем не требуется.

  Reply

- Февраль 19, 2017 at 20:55
  *Юрий* says:

```
public class If_Diamond {

public static void main(String[] args) {
int count = 11;
for (int i = count / 2; i < count; i++) {
for (int k = 0; k < count; k++) {
if (k == i || k == count — 1 — i) {
System.out.print("*");
} else {
System.out.print(" ");
}
}
System.out.println();
}

for (int i = 0; i < count / 2; i++) {

for (int k = 0; k < count; k++) {
if (k == i + 1 || k == count — 2 — i) {
```

```
System.out.print("*");
} else {
System.out.print(" ");
}
}

System.out.println();
}
}
}
```

Получилось только из двух циклов. Кто-нибудь смог из одного? Это, вообще, возможно?

Reply

- Февраль 28, 2017 at 19:52
*Марат* says:

Мои 5 копеек 🙂 Могу объяснить кому не понятно. Курс действительно стоящий!

```
----------------------------------------------------------
public class romb
{
public static void main(String[] arg) {
int count = 10;
// Внешний цикл
for(int i=0; i < count+1; ++i) {
// Внутренний цикл для печати одной строки
for(int k=0; k < count+1; ++k) {
// Сложное условие
if((k == count/2-i)||(k == (count/2-i)+count)||(k == count/2+i)||(k == (count/2+i)-count)) {
System.out.print("*");
} else {
System.out.print(" ");
}
}
// Переход на следующую строку
System.out.println();
// (i == count/2+k) ==== (k == (count/2+i)-count) — все можно выразить и через i
```

```
//(i,k) = (0,5); (1,4&6); (2,3&7); (3,2&8); (4,1&9); (5,0&10);
//(6,1&9); (7,2&8); (8,3&7); (9,4&6); (10,5)
}
}
}
```

Reply

- Апрель 28, 2017 at 13:53
*Виктория* says:

Условие для печати ромба :
count/2 — i==k || count/2 + i == k|| k+10==count/2+i || k-4==count-i

но значение couner необходимо изменить на 11

Reply

- Май 26, 2017 at 14:04
*BroXx* says:

Классный сайт для новичков. Тысячу лет не программил, аж с 90-х, когда в школе на QBasic пытались нас учить. Решил попробовать освоить java. Прошел туториал на SoloLearn, а потом набрел на этот сайт. Здесь и задачки поинтересней да и на русском, все же несколько легче вопринимается. Спасибо, тем кто поддерживает этот ресурс.
Ну и внесу пару копеек по заданию с фигурами — вот что у меня получилось (первые две выкладывать не стану — они очень просты, а вот равносторонний треугольник и ромб к вашему суду):
1. ТРЕУГОЛЬНИК

```
public class Treugolnik2 {

public static void main(String[] args) {
int count = 11;
for (int i = 0; i < count; i++) {
for (int k = 0; k < count; k++) {
// Условие для треугольника
```

```
// k == count/2 — i Это левое ребро
// k == count/2 + i Это правое ребро
// i == count/2 Это основание треугольника
if (k == count/2 + i || k == count/2 — i || i == count/2 ) {
System.out.print("*");} else{
System.out.print(" ");}
}
System.out.println();
}
}
}
```

2.РОМБ

```
public class Romb1 {
public static void main(String[] args) {
int count = 11;
for (int i = 0; i < count; i++){
for (int k = 0; k < count; k++) {
// Вот что получилось по условиям:
// count/2-i и count/2+1 Это верхняя часть
// i-count/2 Это нижняя левая часть
// (count-1)-(i-count/2) Это нижняя правая часть
if (k==count/2-i || k==count/2+i || k==i-count/2 || k==(count-1)-(i-count/2)) {
System.out.print("*");} else {
System.out.print(" ");}
}
System.out.println();
}
}
}
```

Reply

- Июнь 27, 2017 at 19:35
  *Gray* says:

BroXx, а почему ваш треугольник левитирует? Это, конечно, грех небольшой (задание-то выполнено), но выглядит непрофессионально… Я думаю, вам ясно, что причина множества пустых строк под фигурой — то, что внешний цикл выполняется 10 раз (i<count), тогда как строк с символами всего 5. Но что мешает вам сократить общее число строк? Измените упомянутое условие на i<count/2+1

Reply

- 

Июль 21, 2017 at 15:34
*Silver* says:

Мой вариант ромба:

```
public class Solution {
public static void main(String[] args) {
int count = 11;
for (int i = 0; i < count; i++) {
for (int k = 0; k < count; k++) {
if (i+k == count/2 || i-k == count/2 || k-i == count/2 || i+k == count+4) {
System.out.print("*");
} else {
System.out.print(" ");
}
}
// Переход на следующую строку
System.out.println();
}
}
}
```

Спасибо огромное за этот курс!

Reply

- 

Декабрь 5, 2017 at 16:38
*Ivan* says:

> Давайте попробуем провернуть это действие. Но сначала нам надо познакомиться еще с одним видом оператора — составной оператор. В другой литературе он может называться как-то иначе — я не могу точно сказать, как он должен называться по науке. Его идея очень проста — заключив несколько операций в фигурные скобки, их можно рассматривать как один. Давайте опять посмотрим пример.

По науке это вроде как называется область действия блока кода, который может быть в том числе и вложенным, как в вашем случае.

Оригинал учебника Герберта Шилдта:

> **Область действия и время жизни переменных**
> Все использовавшиеся до сих пор переменные объявлялись в начале метода main ().
> Но в Java можно объявлять переменные в любом блоке кода. Как пояснялось в главе l,
> блок начинается с открывающей фигурной скобки и оканчивается закрывающей фигурной
> скобкой. Блок определяет область действия (видимости) переменных. Начиная
> новый блок, вы всякий раз создаете новую область действия. По существу, область действия
> определяет доступность объектов из других частей программы и время их жизни
> (срок их действия).

Reply

- Декабрь 5, 2017 at 17:06
  *admin* says:

  Вам шашечки или ехать ? 🙂
  Вопрос терминологии конечно важен, но во-первых — я честно предупредил, что название может быть иным.
  Во-вторых — в свою защиту могу предложить посмотреть например вот это: https://msdn.microsoft.com/ru-ru/library/ce4b8s02.aspx.
  При желании в Интернете можно найти много указаний на название «составной оператор».

  Reply

- Декабрь 6, 2017 at 08:06
  *Ivan* says:

  Мне шашечки ни к чему:)

Мы с вами говорим про конкретный язык, вы же отправляете меня к C++

Если мои уточнения, вызывают у вас протест, я ничего подобного писать тут не буду:)

Reply

- 

  Декабрь 6, 2017 at 09:51
  *admin* says:

  Не вызывают протест — я ведь не сказал, что не согласен с Вашим определением. Термин «блок» тоже встречается.
  И уж тем более тут никаких обид — разумные замечания принимаются.
  В данном случае я просто указываю на свои аргументы в пользу использования термина «составной оператор». Вы указываете свои — мы обсуждаем.
  Ваш довод от Шилдта я принял, но в то же время первыми книгами Шилдта были как раз книги по C++. Отчасти поэтому я и привел эту ссылку.
  Давайте тогда отрицать, что в C++ нет названий «метод», «поле», «входные параметры», «условный оператор», «оператор цикла». Они там есть.
  И в Java тоже есть. Почему тогда нельзя использовать «составной оператор», который в C++ есть и означает он то, что я описываю ?
  В принципе можно найти и для Java упоминания «составной оператор»
  — http://netbeans.zlouatom.ru/if-else-switch/
  — http://cybern.ru/java-complex-operation.html
  — http://java-study.ru/49-bloki.html
  — http://darkraha.com/rus/java/lang/java10.php

  Reply

Декабрь 6, 2017 at 10:54
*Ivan* says:

Это: «Вам шашечки или ехать ?» было воспринято мной как ирония, именно поэтому я так и написал. Конечно в разных языках есть совпадающая терминология, но есть и исключения, но теперь определение «{ }» описано со всех сторон:)

Reply

Декабрь 6, 2017 at 12:06
*admin* says:

Ну вот и договорились. Спасибо за замечания. Удачи.

Reply

Декабрь 6, 2017 at 16:19
*Ivan* says:

Спасибо. Вам тоже, всего хорошего.
Ресурс действительно хорош.

Reply

Декабрь 30, 2017 at 11:59
*Олег* says:

ТРЕУГОЛЬНИК

```
public static void main(String[] args) {
int count = 11;
for (int i = 5; i < count; i++) {
for (int k = 0; k < count; k++) {
if (i ==count-1 || k==i || k == (count-1) — i) {
System.out.print("*");
} else {
System.out.print(" ");
}
}
System.out.println();
}
```

}
}

Reply

- 
  Декабрь 30, 2017 at 13:27
  *admin* says:

  Зачет.

  Reply

Декабрь 30, 2017 at 12:44
*Ivar* says:

public class Anime {
public static void main(String[] args) throws InterruptedException {
System.out.print(«8»);
Thread.sleep(1000);
for (int i=0; i<5; i++) {
System.out.print("=");
Thread.sleep(1000);
}
System.out.print("D");
}
}

Reply

- 
  Декабрь 30, 2017 at 13:28
  *admin* says:

1 секунда — слишком долго — но тоже зачет 🙂

Reply

- 
Декабрь 30, 2017 at 14:13
*Олег* says:

Спасибо за зачет и «живой» ресурс, что на сегодняшний день большая редкость. Есть вопрос, а какую инструменту применить, чтобы символ был не следующим в консоли, а вместо предыдущего? Задумка простая, выводить поочередно \ | / —. Этакая аллюзия крутящейся палки на одном месте. Умом понимаю, что это должен быть бесконечный цикл типа for(;;), но вот с позиционированием вывода на печать проблема.

Reply

  - 
Декабрь 30, 2017 at 14:42
*admin* says:

Если выводить текст в консоль, то когда-то в MS-DOS были так называемые управляющие символы ANSI — это невидимые символы, которые позволяли в том числе и возвращать курсор назад.
Как с этими символами обстоят дела сейчас — я не знаю. Не пользовался этим уже очень много лет. Также наверняка есть какие-то возможности работать с экраном.
Понимаю, что любопытно, но в данной ситуации не помогу.

Reply

    - 
Декабрь 30, 2017 at 16:46
*Олег* says:

Что то какая то недоговоренность просматривается )

Reply

Декабрь 30, 2017 at 21:25
*admin* says:

Ошибочка вышла с тэгами 🙂

Reply

Декабрь 31, 2017 at 10:17
*Олег* says:

Быть может можно как то удалить «старый» символ и на его месте новый написать? Нет такого метода?

Reply

Декабрь 31, 2017 at 13:08
*admin* says:

Можно попробовать символ «\b».

Reply

Март 3, 2018 at 14:29
*Arshak* says:

package com.Tahmazyan;

public class Main {

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

int count = 17;

for(int i = 0; i < count; i++)
{
for(int k = 0; k < count; k++)
{
if(i == 0 || k == 0 || k == count — 1 || i == count — 1 || i == k || k == count — i — 1){
System.out.print("*");
}else{
System.out.print(" ");
}
}
System.out.println();
}

System.out.println();

for (int i = 0; i < count; i++){
for (int k = 0; k < count; k++){
if(k == count — 1 || i == count — 1 || k == count — 1 — i){
System.out.print("*");
}else {
System.out.print(" ");
}
}
System.out.println();
}

System.out.println();

for (int i = 0; i < count / 2 + 1; i++){
for (int k = 0; k < count; k++){
if (i == count / 2 || k == count / 2 — i || k == count / 2 + i){
System.out.print("*");
}else {
System.out.print(" ");
}
}
```

```
System.out.println();
}

System.out.println();

for (int i = 0; i < count; i++){
for (int k = 0; k < count; k++) {
if (i = count / 2){
if(k == i — count / 2 || k == count — 1 — (i — count / 2)){
System.out.print(«*»);
}else {
System.out.print(» «);
}
}
}
}
System.out.println();
}

System.out.println();
}
}
```

Reply

- Май 4, 2018 at 15:57
*Chessare* says:

Уважаемый admin. Посмотрите плиз. Не слишком ли криворукий ромб получился (по факту получился).

```
public static void main (String[] args) throws java.lang.Exception
{
int count = 11;

// Внешний цикл
for(int i=0; i < count; i++) {
// Внутренний цикл для печати одной строки
```

```
for (int k = 0; k < count; k++) {
// Здесь условие даже немного проще
if (k == count-6-i || k == i+5 || i == k+5 || k == count — (i-4) ) {
System.out.print("*");
} else {
System.out.print(" ");
}
}
// Переход на следующую строку
System.out.println();
}

}
```

Reply

Май 4, 2018 at 17:08
*admin* says:

Вот это однозначно плохо — забиты константы и при увеличении count это перестанет работать. Я про строку

```
1 if (k == count-6-i || k == i+5 || i == k+5 || k == count — (i-4) ) {
```

Reply

Май 20, 2018 at 01:18
*Chessare* says:

Спасибо!
А вот так?: if(k == count/2-i||k ==i+5||i ==k+5||k==(count/2)-(i-10)) {

Reply

Май 21, 2018 at 09:06
*admin* says:

Так лучше

Reply

Июль 20, 2018 at 23:44
*SoulUran* says:

Dear @admin, a lot of thanks for your great work!
Please, assess my example of generating a diamond worked with two loops.

```
class Diamond {
public static void main (String[] args) {
int count = 11;
// external loop
for (int i = 0; i < count; i++) {
// internal loop
for (int k = 0; k < count; k++) {
// main condition
if ((k == (count/2)-i || k == (count/2)+i) || (k == (i-count/2) || k == (count — (i-count/2) — 1))) {
System.out.print("*"); // fill "*"
}
else {
System.out.print(" "); // fill space
}
}
System.out.println(); // line feed
}
}
}
```

Reply

Июль 21, 2018 at 05:14
*admin* says:

Dear @SoulUran, thank you for comment. Code looks good and condition is elegant, but comments within code do not make sense.
Everyone can see external and internal loop and see «main condition».
I guess, if you would describe what program does and algorithm for condition — it will be more better.

And next time please use «code» tags for source code. Good luck.

Reply

## Leave a reply

Comment

You may use these HTML tags and attributes: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code class="" title="" data-url=""> <del datetime=""> <em> <i> <q cite=""> <s> <strike> <strong> <pre class="" title="" data-url=""> <span class="" title="" data-url="">`

Имя *

E-mail *

Сайт

восемь × 3 =

Add comment

Copyright © 2018 Java Course
Designed by Blog templates, thanks to: Free WordPress themes for photographers, LizardThemes.com and Free WordPress real estate themes