

Java Basics

Java Fundamentals

Lesson 03

OLGA SMOLYAKOVA

План конспекта

1. Чтение из консоли

2. Циклы

3. Оператор switch

доп. Что такое структуры данных

4. Практическая работа

Для чтения из консоли в Java можно использовать класс `Scanner`. Так как `Scanner` ссылочный тип данных, то для его использования требуется создать объект и далее вызывать на объекте методы, заставляя работать (подробно этот материал разбирается в теме 'Классы и объекты').

1.1 Напишем приложение, читающее целое число из консоли.

```
import java.util.Scanner;

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

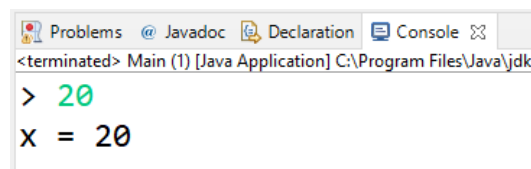
        Scanner sc;
        int x;

        sc = new Scanner(System.in);

        System.out.print("> ");
        x = sc.nextInt();

        System.out.println("x = " + x);
    }
}
```

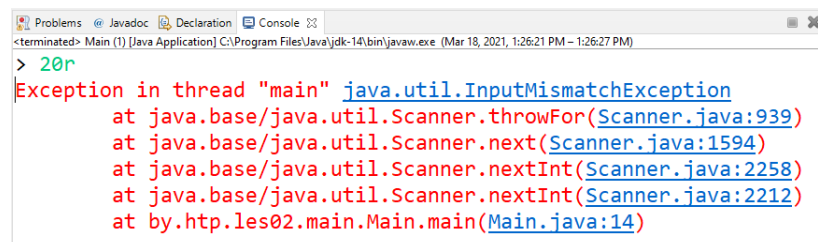
Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk
> 20
x = 20
```

1.2 Однако, если ввести некорректное значение, в результате получим исключение и выполнение программы прервется.

Результат:



```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar 18, 2021, 1:26:21 PM - 1:26:27 PM)
> 20r
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.nextInt(Scanner.java:2258)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
    at by.http.les02.main.Main.main(Main.java:14)
```

1.3 Прочитать же любую информацию из консоли можно в строку.

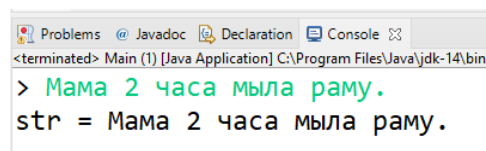
```
public static void main(String[] args) {
    Scanner sc;
    String str;

    sc = new Scanner(System.in);

    System.out.print("> ");
    str = sc.nextLine();

    System.out.println("str = " + str);
}
```

Результат:



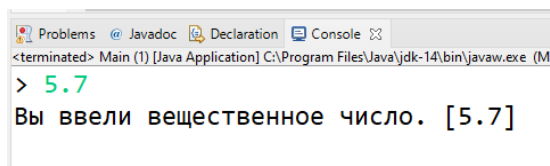
```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin
> Мама 2 часа мыла раму.
str = Мама 2 часа мыла раму.
```

1.4 Также класс Scanner может 'подглядывать' в поток ввода и сообщать, введено ли значение, приводимое к какому-либо примитивному типу (или введена просто строка).

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("> ");
    if (sc.hasNextInt()) {
        System.out.println("Вы ввели целое число. [" + sc.nextInt() + "]");
    } if (sc.hasNextDouble()) {
        System.out.println("Вы ввели вещественное число. [" +
sc.nextDouble() + "]");
    } else {
        System.out.println("Вы ввели что-то другое. [" + sc.nextLine() + "]");
    }
}
```

Результат:

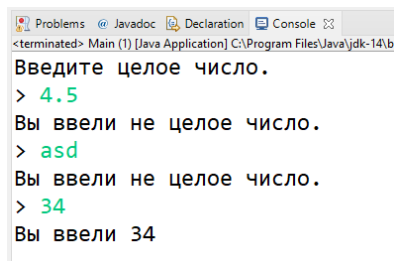


```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (M
> 5.7
Вы ввели вещественное число. [5.7]
```

- 1.5 А сейчас напишем приложение, которое будет просить пользователя ввести целое число до тех пор, пока он его не введет.

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int x;
    System.out.println("Введите целое число.");
    System.out.print("> ");
    while(!sc.hasNextInt()){
        sc.nextLine();//извлекли ошибочный ввод пользователя
        System.out.println("Вы ввели не целое число.");
        System.out.print("> ");
    }
    x = sc.nextInt();
    System.out.println("Вы ввели " + x);
}
```

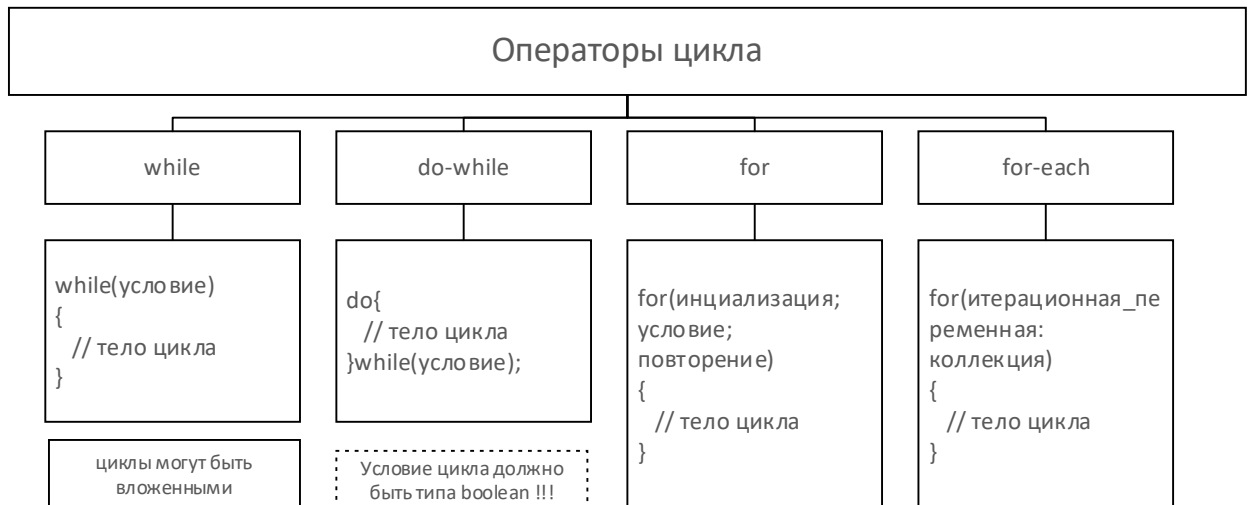
Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin
Введите целое число.
> 4.5
Вы ввели не целое число.
> asd
Вы ввели не целое число.
> 34
Вы ввели 34
```

В данном примере для реализации повторяющихся действий был использован цикл **while**.

В языке ява присутствуют 4 оператора цикла. Каждый цикл выполняется, пока условие цикла принимает истинное значение, и завершается, как только оно становится ложным.



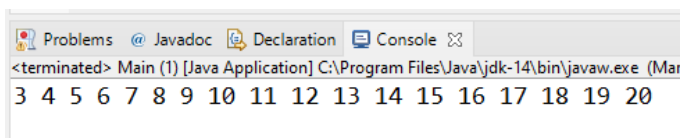
Решим ряд задач с использованием циклов.

2.1 Напишите приложение, которое выводит на консоль числа от 3 до 20 (используйте цикл while).

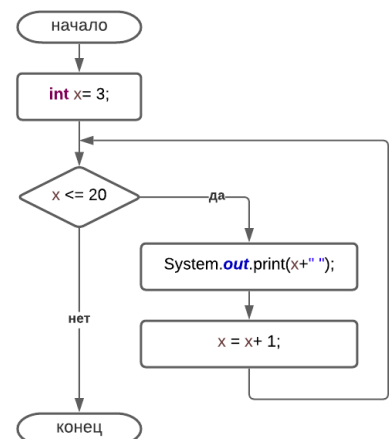
```
public static void main(String[] args) {
    int x = 3;

    while(x <= 20) {
        System.out.print(x + " ");
        x = x + 1;
    }
}
```

Результат:



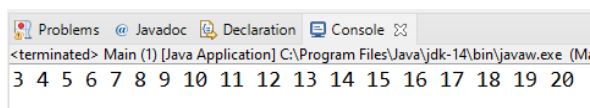
```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```



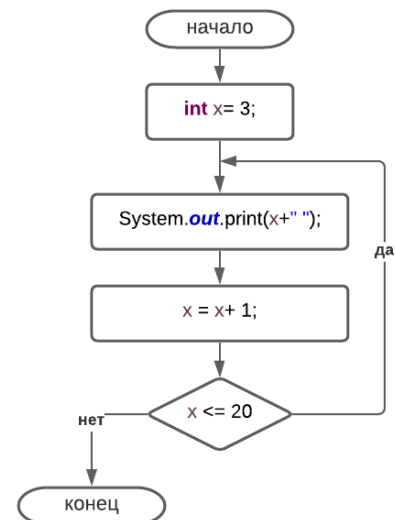
2.2 Напишите приложение, которое выводит на консоль числа от 3 до 20 (используйте цикл do-while).

```
public static void main(String[] args) {  
    int x = 3;  
  
    do{  
        System.out.print(x + " ");  
        x = x + 1;  
    }while(x <= 20);  
}
```

Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (M:  
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

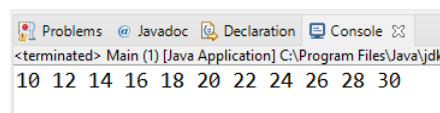


Отличие цикла **while** от **do-while** состоит в том, что цикл while может не выполниться ни разу, если при первом входе в цикл выражение сразу будет ложным. Цикл do-while в любом случае выполнится хотя бы один раз.

2.3 Напишите программу, которая печатает все четные числа от 10 до 30.

```
public static void main(String[] args) {  
  
    int x = 10;  
  
    while (x < 31) {  
        System.out.print(x + " ");  
        x = x + 2;  
    }  
}
```

Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk  
10 12 14 16 18 20 22 24 26 28 30
```

2.4 Введите два числа и распечатайте все числа от наибольшего до наименьшего.

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

    Scanner sc = new Scanner(System.in);
    int x;
    int y;

    System.out.print("> ");
    while(!sc.hasNextInt())
    {
        sc.nextLine();
        System.out.print("> ");
    }
    x = sc.nextInt();

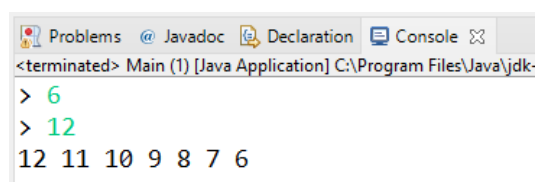
    System.out.print("> ");
    while(!sc.hasNextInt())
    {
        sc.nextLine();
        System.out.print("> ");
    }
    y = sc.nextInt();

    if(x < y) {
        int temp;
        temp = x;
        x = y;
        y = temp;
    }

    int i;
    i = x;

    while(i >= y) {
        System.out.print(i + " ");
        i--;
    }
}
```

Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-
> 6
> 12
12 11 10 9 8 7 6
```


2.5 Даны два числа. Посчитать сумму всех чисел между ними.

```
public static void main(String[] args) {  
    int x;  
    int y;  
    int sum;  
  
    x = 5;  
    y = 12;  
  
    sum = 0;  
  
    while(x <= y) {  
        sum = sum + x;  
        x++;  
    }  
  
    System.out.println("sum = " + sum);  
}
```

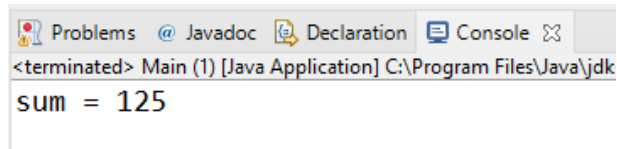
2.6 С помощью **while** и **do-while** можно написать 'вечный цикл'. Для этого в условии цикла достаточно написать литерал **true**. Выход из такого цикла возможен с помощью операторов **break**, **return** или сгенерированного исключения.

```
while(true) {  
  
}  
  
do {  
  
}while(true);
```

2.7 Найдите сумму всех чисел от a до b (используйте цикл for)

```
public static void main(String[] args) {  
    int a;  
    int b;  
    int sum;  
  
    a = 8;  
    b = 17;  
  
    sum = 0;  
    for(int i = a; i<=b; i++) {  
        sum = sum + i;  
    }  
  
    System.out.println("sum = " + sum);  
}
```

Результат:



Синтаксис цикла `for` отличается от `while` достаточно сильно:

```
for(выражение1; выражение2; выражение3){  
    // тело цикла  
}
```

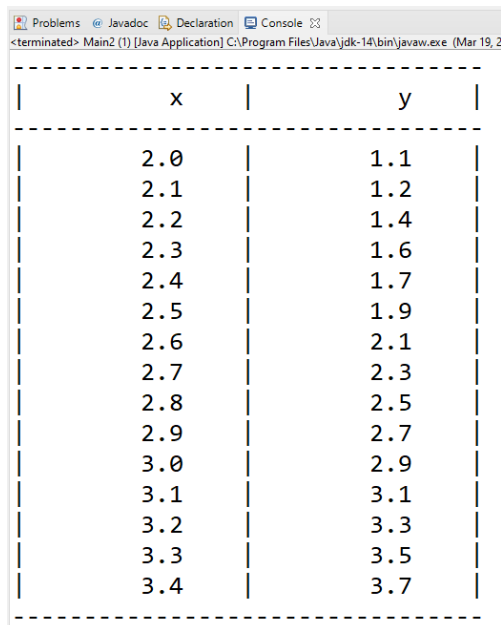
Выражение1 выполняется один раз при входе в цикл; выражение2 проверяется *перед* каждой итерацией цикла. Если оно вычислится в `false`, цикл остановится. Выражение3 выполняется *после* тела цикла на каждой итерации *перед* проверкой условия.

- 2.8 Задача. Написать программу для вычисления значений функции $F(x)$ на отрезке $[a, b]$ с шагом h . Результат представить в виде таблицы, первый столбец которой – значения аргумента, второй - соответствующие значения функции:

$$F(x) = x - \sin(x)$$

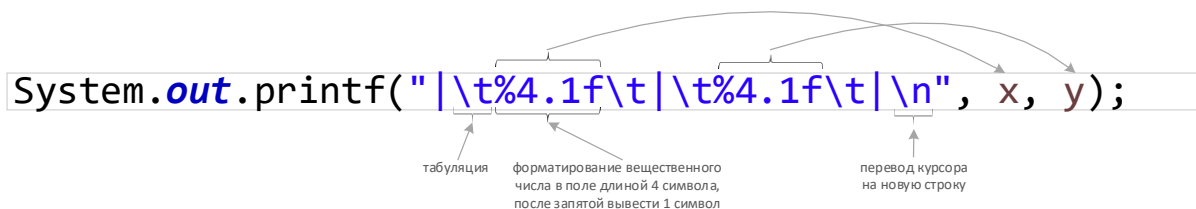
```
public static void main(String[] args) {  
  
    double a;  
    double b;  
    double h;  
    double y;  
  
    a = 2;  
    b = 3.5;  
    h = 0.1;  
  
    System.out.println("-----");  
    System.out.printf("|\\t%4s\\t|\\t%4s\\t|\\n", "x", "y");  
    System.out.println("-----");  
  
    for(double x = a; x <=b; x = x + h) {  
        y = x - Math.sin(x);  
        System.out.printf("|\\t%4.1f\\t|\\t%4.1f\\t|\\n", x, y);  
    }  
    System.out.println("-----");  
}
```

Результат:



x	y
2.0	1.1
2.1	1.2
2.2	1.4
2.3	1.6
2.4	1.7
2.5	1.9
2.6	2.1
2.7	2.3
2.8	2.5
2.9	2.7
3.0	2.9
3.1	3.1
3.2	3.3
3.3	3.5
3.4	3.7

Метод `printf`, который в решении использовался для вывода данных на консоль, позволяет делать вывод форматированным.



```
System.out.printf("\\t%4.1f\\t\\t%4.1f\\t\\n", x, y);
```

табуляция

форматирование вещественного числа в поле длиной 4 символа, после запятой вывести 1 символ

перевод курсора на новую строку

2.9 При необходимости цикл `for` позволяет себя достаточно сильно модифицировать.

Например переменную-счетчик цикла можно объявить и до цикла, оставив выражение1 пустым.

```
int i = 0;
for(; i < 10; i++) {

}
```

Счетчик цикла изменять можно и в теле цикла, оставив выражение3 пустым.

```
int i = 0;
for(; i < 10; ) {
    i = i + 2;
}
```

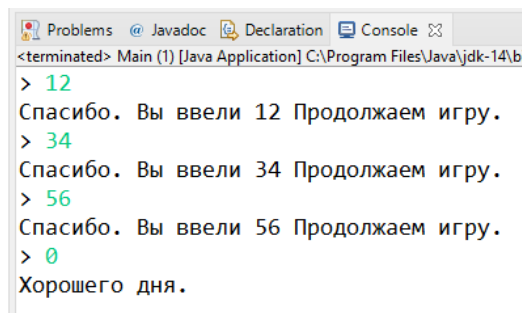
Цикл for можно сделать 'вечным' оставив выражение 2 пустым

```
for( ; ; ) {  
  
}
```

2.10 Напишите программу, запрашивающую у пользователя целые числа с клавиатуры до тех пор, пока пользователь не введет 0.

```
public static void main(String[] args) {  
  
    Scanner sc = new Scanner(System.in);  
    int x;  
    int y;  
  
    do {  
        System.out.print("> ");  
        while (!sc.hasNextInt()) {  
            sc.nextLine();  
            System.out.print("не-не > ");  
        }  
        x = sc.nextInt();  
  
        if (x == 0) {  
            break;  
        }  
  
        System.out.println("Спасибо. Вы ввели " + x + " Продолжаем игру.");  
    } while (true); // вечный цикл  
  
    System.out.println("Хорошего дня.");  
}
```

Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin  
> 12  
Спасибо. Вы ввели 12 Продолжаем игру.  
> 34  
Спасибо. Вы ввели 34 Продолжаем игру.  
> 56  
Спасибо. Вы ввели 56 Продолжаем игру.  
> 0  
Хорошего дня.
```

Операторы break и continue меняют поток выполнения программы и могут использоваться только в циклах; break также используется в операторе switch.

Оператор `break` в цикле прерывает выполнение цикла и передает управление оператору, следующему сразу же за телом цикла.

Стоит заметить, что циклы часто взаимозаменяемы. В том числе пример выше можно решить с и помощью цикла `while` без использования `break`.

2.11 Задача. Написать программу, которая генерирует случайные числа, на консоль при этом нужно вывести только нечетные числа.

```
import java.util.Random;

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

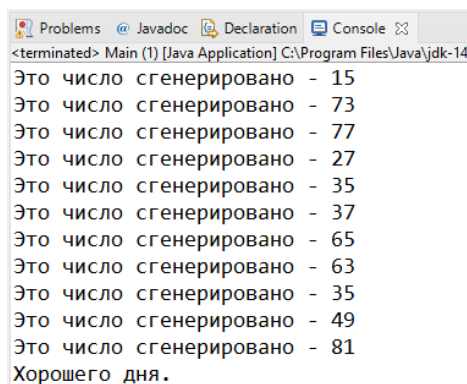
        Random rand = new Random();
        int x;

        while( (x = rand.nextInt(100)) != 0) {
            if (x % 2 != 1) {
                continue;
            }

            System.out.println("Это число сгенерировано - " + x);
        }

        System.out.println("Хорошего дня.");
    }
}
```

Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14
Это число сгенерировано - 15
Это число сгенерировано - 73
Это число сгенерировано - 77
Это число сгенерировано - 27
Это число сгенерировано - 35
Это число сгенерировано - 37
Это число сгенерировано - 65
Это число сгенерировано - 63
Это число сгенерировано - 35
Это число сгенерировано - 49
Это число сгенерировано - 81
Хорошего дня.
```

Оператор `continue` пропускает все операторы до конца текущей итерации и передает управление оператору проверки условия продолжения цикла, и если результат выражения будет истинным – цикл продолжит работу.

Оператор switch в Java- это условный оператор, который дает возможность сравнивать переменную со списком значений.

```
switch(выражение) {  
    case значение1:  
        // Блок кода 1  
        break;  
    case значение2:  
        // Блок кода 2  
        break;  
    ...  
    case значениеN:  
        // Блок кода N  
        break;  
    default :  
        // Блок кода для default  
}
```

В круглых скобках указывается выражение, значение которого будет сравниваться со списком значений, перечисленных после ключевого слова case. Если выражение совпадает со значением1, то выполняется блок кода, указанный после значение1. Если выражение не совпадает ни с одним из значений, то выполняется блок кода для default. Этот блок кода является необязательным - можно написать оператор switch и без него.

- 3.1 Задача. Напишите программу, позволяющую по последней цифре числа определить последнюю цифру его квадрата.

```
public static void main(String[] args) {  
  
    int number = 0;  
  
    int lastDigit;  
    int lastDigitSq;  
  
    number = 91;  
  
    lastDigit = number % 10;  
    switch (lastDigit) {  
    case 0:  
        lastDigitSq = 0;  
        break;
```

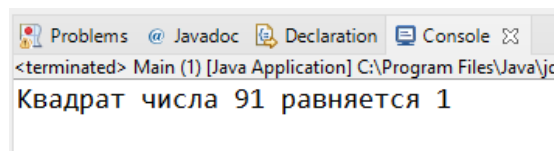
```

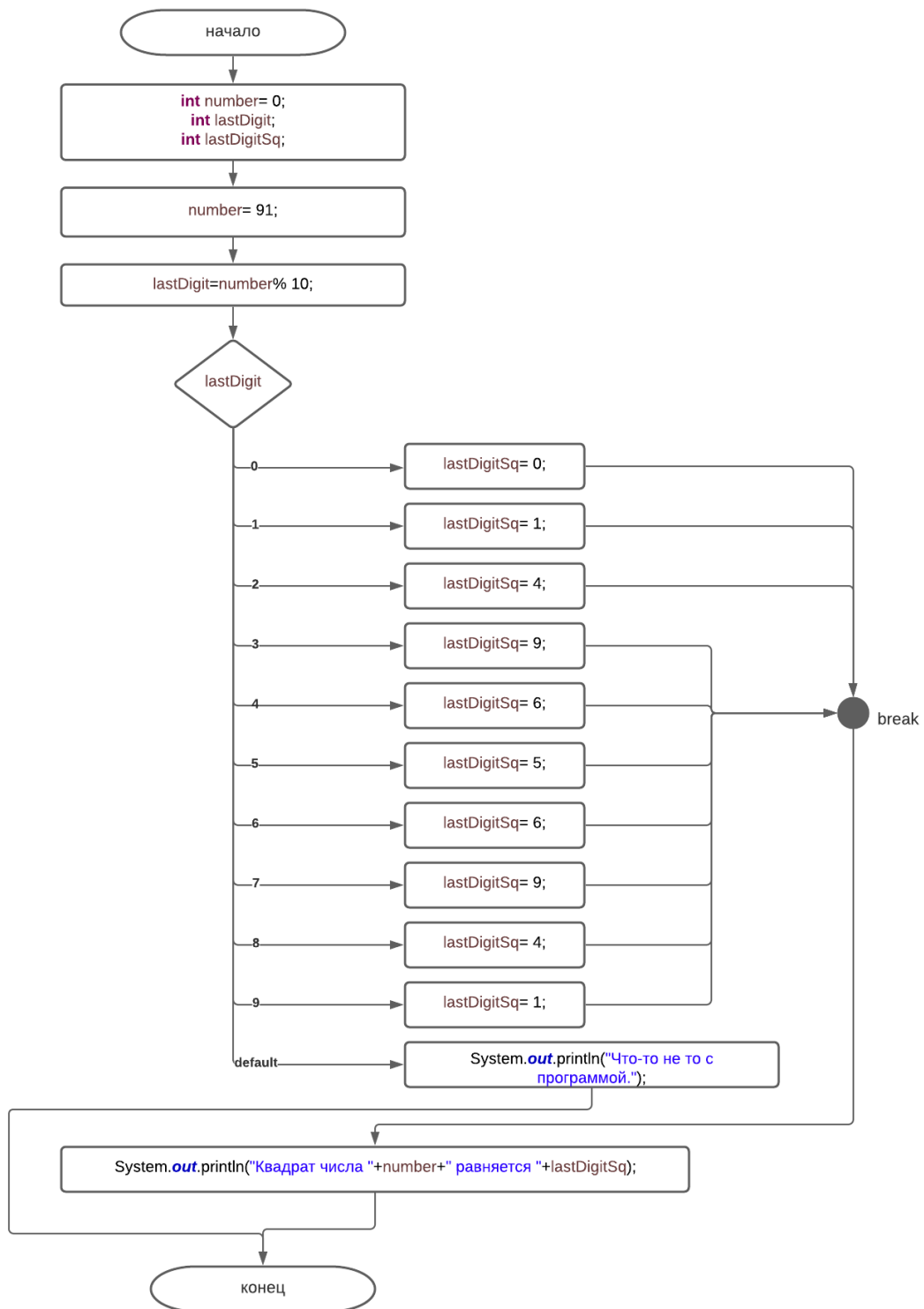
    case 1:
        lastDigitSq = 1;
        break;
    case 2:
        lastDigitSq = 4;
        break;
    case 3:
        lastDigitSq = 9;
        break;
    case 4:
        lastDigitSq = 6;
        break;
    case 5:
        lastDigitSq = 5;
        break;
    case 6:
        lastDigitSq = 6;
        break;
    case 7:
        lastDigitSq = 9;
        break;
    case 8:
        lastDigitSq = 4;
        break;
    case 9:
        lastDigitSq = 1;
        break;
    default:
        System.out.println("Что-то не то с программой.");
        return;
}

System.out.println("Квадрат числа " + number + " равняется " + lastDigitSq);
}

```

Результат выполнения:





При совпадении значений выражения в операторе switch и константы в case-ветке других сравнений с другими case-верками не происходит, а switch выполняет операторы последовательно либо до встречи с оператором break, либо до конца оператора switch.

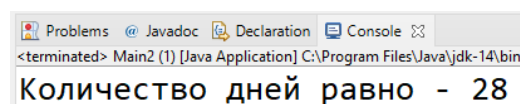
3.2 Задача. Составить программу, которая по заданным году и номеру месяца *m* определяет количество дней в этом месяце.

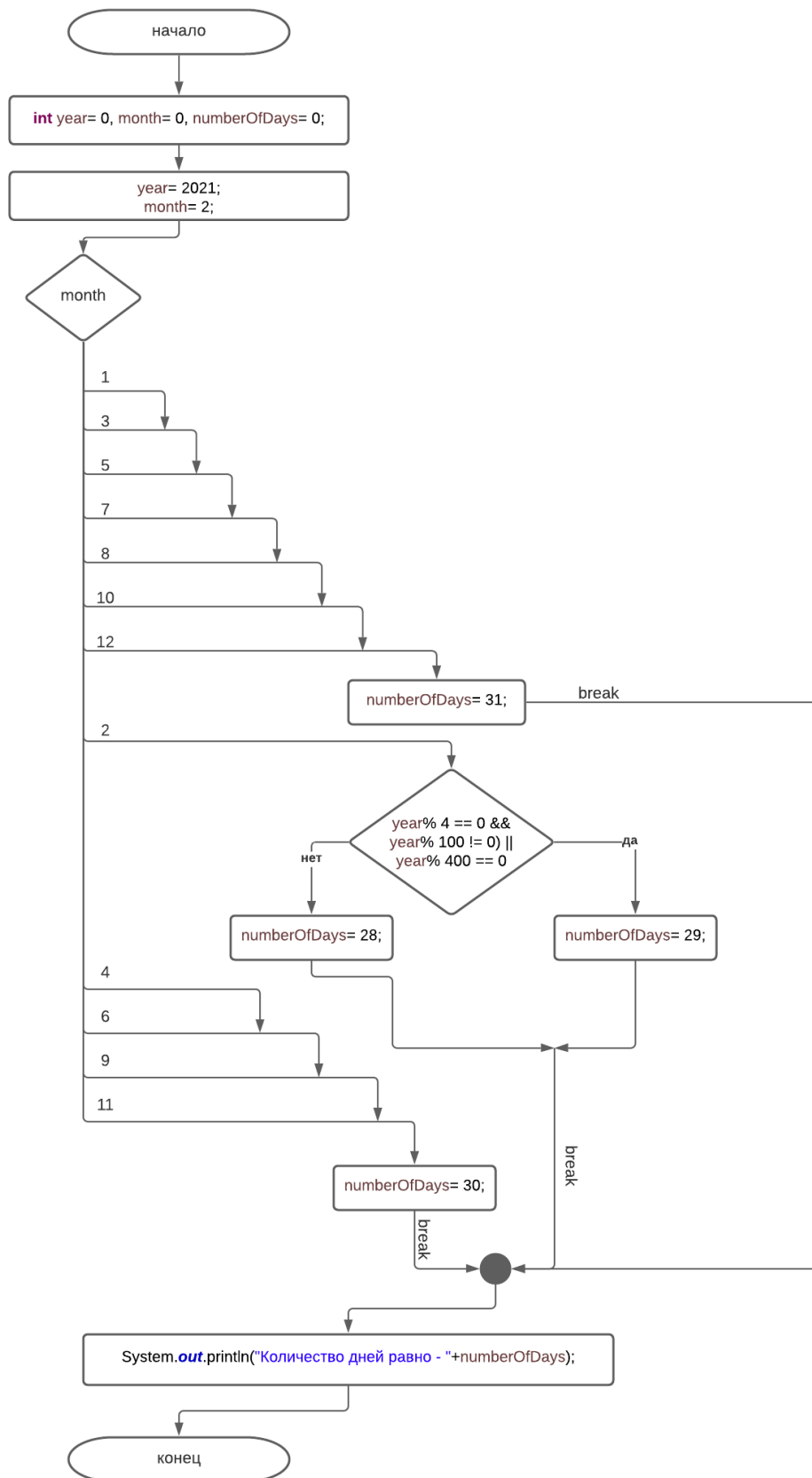
```
public static void main(String[] args) {
    int year = 0, month = 0, numberOfDays = 0;
    year = 2021;
    month = 2;

    switch (month) {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            numberOfDays = 31;
            break;
        case 2:
            if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
                numberOfDays = 29;
            } else {
                numberOfDays = 28;
            }
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            numberOfDays = 30;
            break;
    }

    System.out.println("Количество дней равно - " + numberOfDays);
}
```

Результат выполнения:





Выражение в switch должно иметь тип char, byte, short, int, enum (начиная с Java 6) или String (начиная с Java 7). Использование любого другого типа, например float, приведет к ошибке компиляции.

Оператор switch может только проверять на равенство. Такие операторы как >=, <= недопустимы.



Что такое структуры данных

- *.1 Решим задачу. Напишите приложение, которое вводит 5 чисел с клавиатуры и выводит их сумму в виде, например:

$$(2) + (-3) + (12) + (4) + (-1) = [14]$$

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a;
        int b;
        int c;
        int d;
        int e;
        int z;

        // опустим код, который контролирует корректность ввода
        System.out.print("> ");
        a = sc.nextInt();

        System.out.print("> ");
        b = sc.nextInt();

        System.out.print("> ");
        c = sc.nextInt();

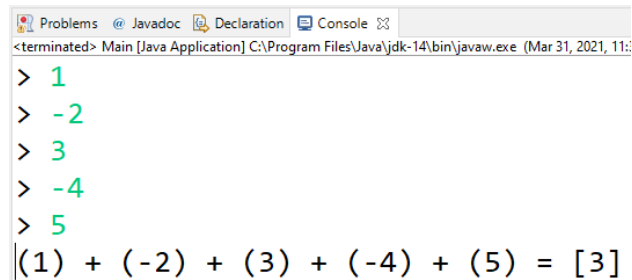
        System.out.print("> ");
        d = sc.nextInt();

        System.out.print("> ");
        e = sc.nextInt();

        z = a + b + c + d + e;

        System.out.println("(" + a + ") + (" + b + ") + (" + c
            + ") + (" + d + ") + (" + e + ") = " + "[" + z + "]");
    }
}
```

Результат:



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar 31, 2021, 11:
> 1
> -2
> 3
> -4
> 5
(1) + (-2) + (3) + (-4) + (5) = [3]
```

- *.2 Добавим новую возможность предыдущей задаче. Напишите приложение, которое вводит 5 чисел с клавиатуры и выводит их сумму в виде

$$(2) + (-3) + (12) + (4) + (-1) = [14]$$

или как

$$\begin{array}{r} 2 + \\ -3 + \\ 12 + \\ 4 + \\ -1 = \\ 14 \end{array}$$

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int a;
    int b;
    int c;
    int d;
    int e;
    int z;
    // опустим код, который контролирует корректность ввода
    System.out.print("> ");
    a = sc.nextInt();

    System.out.print("> ");
    b = sc.nextInt();

    System.out.print("> ");
    c = sc.nextInt();

    System.out.print("> ");
    d = sc.nextInt();

    System.out.print("> ");
    e = sc.nextInt();

    z = a + b + c + d + e;
```

```

String message = "Введите 1, если хотите вывести результат
строкой.\n" + "Введите любое другое число, если хотите вывести
результат лесенкой. \n> ";
System.out.print(message);
int menu;
while(!sc.hasNextInt()) {
    sc.nextLine();
    System.out.print(message);
}
menu = sc.nextInt();

if(menu == 1) {
    System.out.println("(" + a + ") + (" + b + ") + ("
        + c + ") + (" + d + ") + (" + e + ") = "
        + "[" + z + "]");
} else {
    System.out.println("(" + a + ") + \n\t (" + b
        + ") + \n\t\t (" + c + ") + \n\t\t\t ("
        + d + ") + \n\t\t\t\t ("
        + e + ") = \n\t\t\t\t\t " + "[" + z + "]");
}
}

```

Результат:

```

> 1
> -2
> 3
> -4
> 5
Введите 1, если хотите вывести результат строкой.
Введите любое другое число, если хотите вывести результат лесенкой.
> 2
(1) +
      (-2) +
        (3) +
          (-4) +
            (5) =
                        [3]

```

***.3** Модифицируем задачу дальше. В приложении по-прежнему необходимо иметь возможность выводить результат в двух разных видах, но приложение теперь должно спрашивать, сколько чисел хочет ввести пользователь.

Решить эту задачу объявив n переменных не получится – появилась необходимость централизованно хранить данные. Для таких случаев подходят структуры данных, которые называются массивами.

Итак, решим задачу.

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        int[] mas;
        // опустим код, который контролирует корректность ввода
        System.out.print("Сколько элементов хочет ввести > ");
        n = sc.nextInt();

        mas = new int[n];
        int sum = 0;

        for(int i=0; i<mas.length; i++) {
            System.out.print("mas[" + i + "]=");
            mas[i] = sc.nextInt();
            sum = sum + mas[i];
        }

        String message = "Введите 1, если хотите вывести результат
строкой.\n" + "Введите любое другое число, если хотите вывести
результат лесенкой. \n> ";
        System.out.print(message);
        int menu;
        while(!sc.hasNextInt()) {
            sc.nextLine();
            System.out.print(message);
        }
        menu = sc.nextInt();

        if(menu == 1) {
            for(int i = 0; i < mas.length; i++) {
                if(i == mas.length - 1) {
                    System.out.print("(" + mas[i] + ") ");
                }else {
                    System.out.print("(" + mas[i] + ") + ");
                }
            }
            System.out.println(" = " + "[" + sum + "]");
        }else {
            String delimiter = "\n\t";
            for(int i = 0; i < mas.length; i++) {
                System.out.print("(" + mas[i] + ") + "
                    + delimiter);
                delimiter = delimiter + '\t';
            }
        }
    }
}
```

```

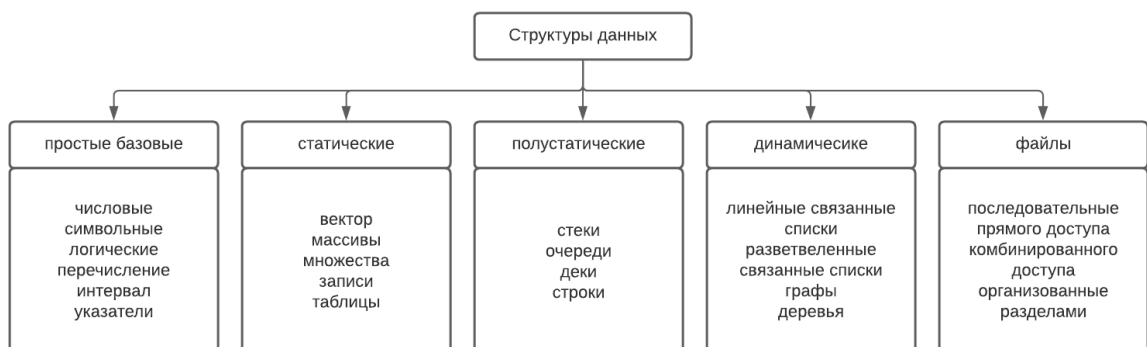
    }
    System.out.println(" = " + "[" + sum + "]");
}
}
}

```

*.4 Что же такое структуры данных?

Структурой данных называют множество элементов данных и связей между ними. Такое определение кажется непонятным при первом изучении. Попробуем сказать проще. Структура данных – это такой способ организации данных, который позволяет их обрабатывать единообразно и циклически в зависимости от способов, предусмотренных конкретной структурой. Группа студентов, например, тоже может быть примером структуры данных (если студентов рассматривать как данные, конечно 😊). Группе (всем студентам) можно назначить занятие, можно циклически перебирать студентов в группе (о которых будет известно только то, что они студенты, а на самом деле могут быть докторами или космонавтами) и задавать каждому вопрос и т.д.

Приведем общую классификацию структур данных.



Структуры данных и алгоритмы их обработки в отдельном курсе. В текущем курсе по Java понадобятся знания только некоторых из них.

Подготовка рабочего пространства для решения задач

1. Создайте проект с именем Unit03[Surname] (например, Unit02Ivanov).
2. Далее создайте класс Task01 и поместите его в пакет `by.epam.unit03.main`.
3. Аналогично создайте еще 7 классов Task02 ... Task08.
4. В каждом классе создайте метод `main` и пишите в нем код решения задачи из списка задач с таким же номером, как и у класса.

Список задач

Задача 1

Необходимо вывести на экран таблицу умножения на 3.

Задача 2

С помощью оператора `while` напишите программу определения суммы всех нечетных чисел в диапазоне от 1 до 99 включительно.

Задача 3

Вычислить: $(1+2) * (1+2+3) * \dots * (1+2+\dots+10)$.

Задача 4

Составить таблицу значений функции $y = 5 - x^2/2$ на отрезке $[-5; 5]$ с шагом 0.5.

Задача 5

Составить программу для вычисления значений функции $F(x)$ на отрезке $[a, b]$ с шагом h . Результат представить в виде таблицы, первый столбец которой – значения аргумента, второй - соответствующие значения функции:

$$F(x) = 2tg\left(\frac{x}{2}\right) + 1$$

Задача 6

Даны два числа. Определить цифры, входящие в запись как первого так и второго числа.

Задача 7

Написать программу, переводящую римские **цифры** в арабские.

Задача 8

Написать программу, в которой вводятся два операнда X и Y и знак операции (+, −, /, *). Вычислить результат Z в зависимости от знака. Предусмотреть реакции на возможный неверный знак операции, а также на ввод Y=0 при делении. Организовать возможность многократных вычислений без перезагрузки программа (т.е. построить цикл). В качестве символа прекращения вычислений принять '#’.