

Java Basics. Java Fundamentals. Lesson 02.

OLGA SMOLYAKOVA

План конспекта

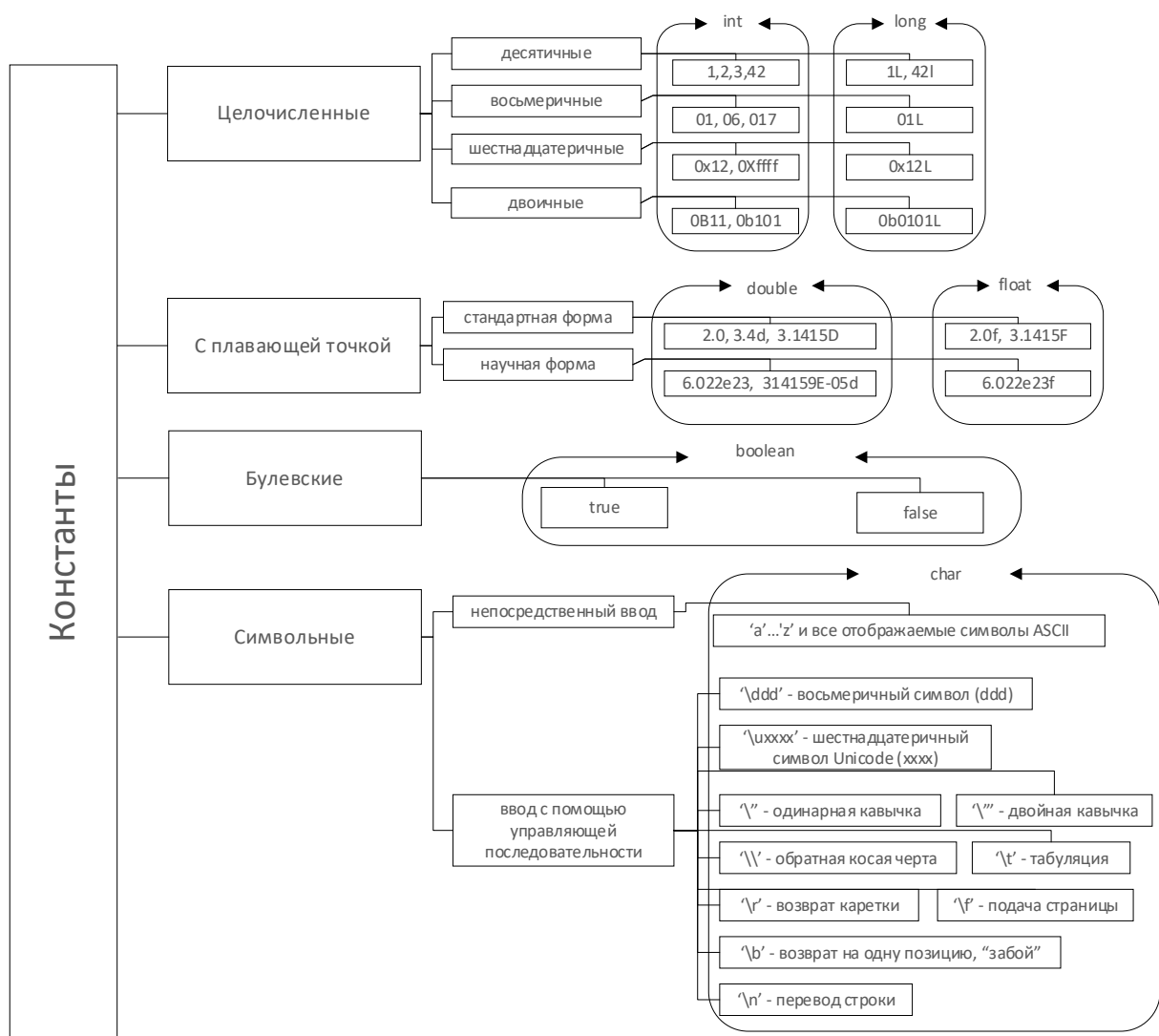
1. Литералы	
2. Блок-схемы	
3. Приведение примитивных типов	
4. Приоритет операции	
5. Оператор if	
6. Составные логические выражения	
7. Практическая работа	

- 1.1 Решим задачу. Идет n-я секунда суток, определить, сколько полных часов, полных минут и секунд прошло к этому моменту.

```
public static void main(String[] args) {  
  
    int numberOfSeconds = 0;  
    int hour, min, sec;  
  
    int saveNumberOfSeconds;  
  
    numberOfSeconds = 3405;  
    saveNumberOfSeconds = numberOfSeconds;  
  
    if ((numberOfSeconds > 86400) || (numberOfSeconds < 0)) {  
        System.out.println("Неверное значение.");  
        return;  
    }  
  
    hour = numberOfSeconds / 3600;  
    numberOfSeconds = numberOfSeconds - hour * 3600;  
    min = numberOfSeconds / 60;  
    numberOfSeconds = numberOfSeconds - min * 60;  
    sec = numberOfSeconds;  
  
    System.out.println("В " + saveNumberOfSeconds + " секундах: " +  
hour + " ч. " + min + " мин. " + sec + " сек.");  
}
```

Для решения задач в выражениях часто применяют не только переменные, но и конкретные значения/константы/литералы. Литералом называется значение, изменить которое можно только в исходном коде, после чего приложение следует перекомпилировать.

Литералы, как и типы, делятся на две категории: примитивные и литерал ссылочного типа String (строка).



2

Блок-схемы

При решении задач разработчикам постоянно приходится создавать алгоритмы, а также изучать стандартные или разбираться с реализованными модификациями алгоритмов. **Алгоритм** – это четкое описание последовательности действий, которые необходимо выполнить для получения результата.

Отобразить действия алгоритма проще всего в виде рисунка – блок-схемы. **Блок-схемой** называется наглядное графическое изображение алгоритма, когда отдельные его этапы изображаются при помощи различных геометрических фигур – блоков, а связи между этапами (последовательность

выполнения этапов) указываются при помощи стрелок, соединяющих эти фигуры. Блоки сопровождаются надписями. Типичные действия алгоритма изображаются следующими геометрическими фигурами:

Блок начала-конца алгоритма. Надпись в блоке: «начало» («конец»).



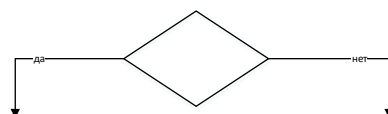
Блок ввода-вывода данных. Надпись в блоке: слово «ввод» («вывод» или «печать») и список вводимых (выводимых) переменных.



Блок решения или **действия**. Надпись в блоке: конкретная операция или группа операций.

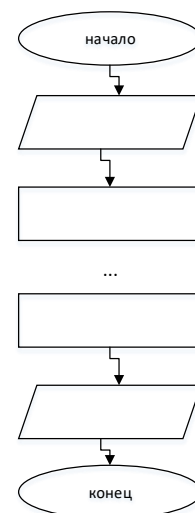
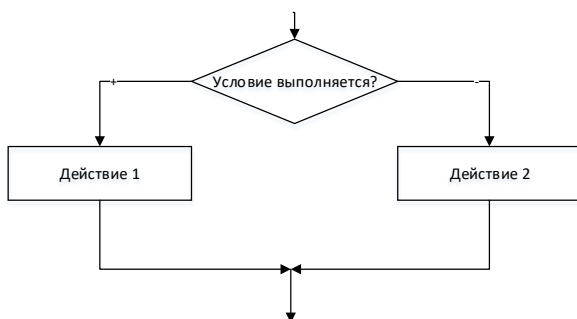


Условный блок. Надпись в блоке: проверяемое условие. В результате проверки условия осуществляется выбор одного из возможных путей (ветвей) вычислительного процесса. Если условие выполняется, то следующим выполняется этап по ветви «да», если условие не выполняется, то выполняется этап по ветви «нет».

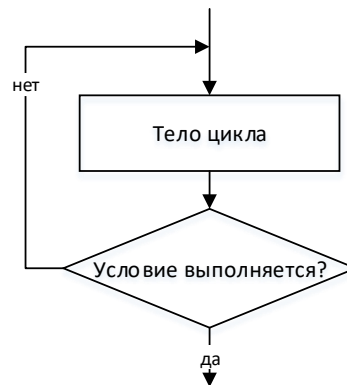
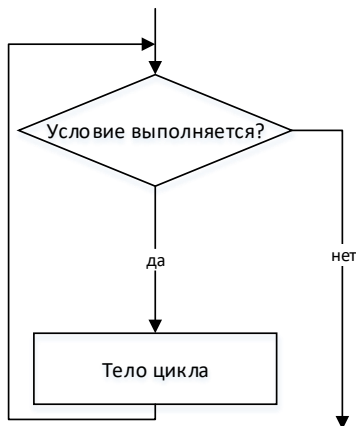


Линейный алгоритм. В котором все операции выполняются последовательно одна за другой.

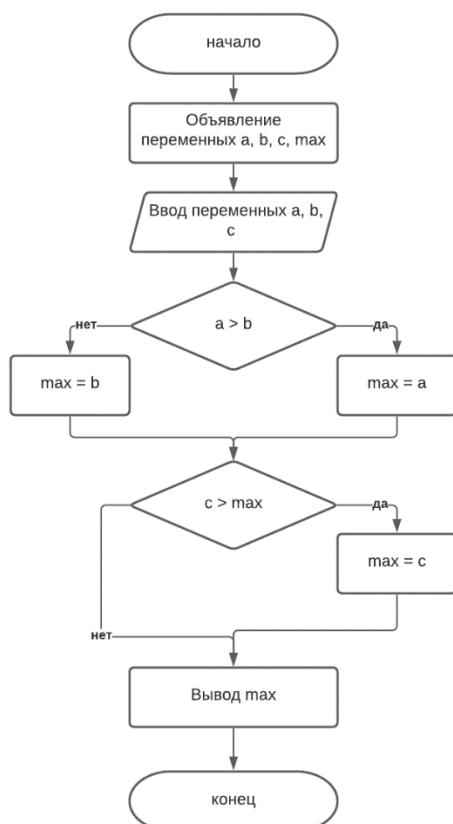
Разветвленный алгоритм. Алгоритмы разветвленной структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие.



Циклический алгоритм. Циклом называют повторение одних и тех же действий (шагов). Последовательность действий, которые повторяются в цикле, называют телом цикла.



2.1 Приведем пример блок-схемы программы нахождения наибольшего из трех чисел и соответствующий ей программный код.



```

public static void main(String[] args) {
    int a;
    int b;
    int c;

    int max;

    a = 67;
    b = 14;
    c = -2;

    if(a > b) {
        max = a;
    }else {
        max = b;
    }

    if(c > max) {
        max = c;
    }

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

3

Приведение примитивных типов

3.1 Решим задачу. Напишите программу, вычисляющую значение функции $F(x)$:

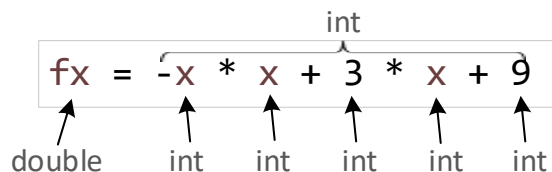
$$F(x) = \begin{cases} -x^2 + 3x + 9, & \text{если } x \geq 3; \\ \frac{1}{x^3 - 6}, & \text{если } x < 3 \end{cases}$$

```
public static void main(String[] args) {
    int x;
    double fx;

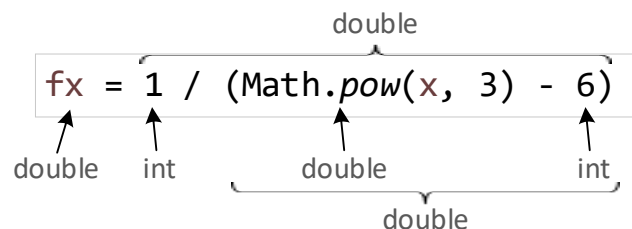
    x = 6;

    if (x >= 3) {
        fx = -x * x + 3 * x + 9;
    } else {
        fx = 1 / (Math.pow(x, 3) - 6);
    }
    System.out.println("F(x)=" + fx);
}
```

Проанализировав реализацию формул можно заметить, что в одном случае, выражение будет посчитано в типе `int`, а потом будет присвоено типу `double`.



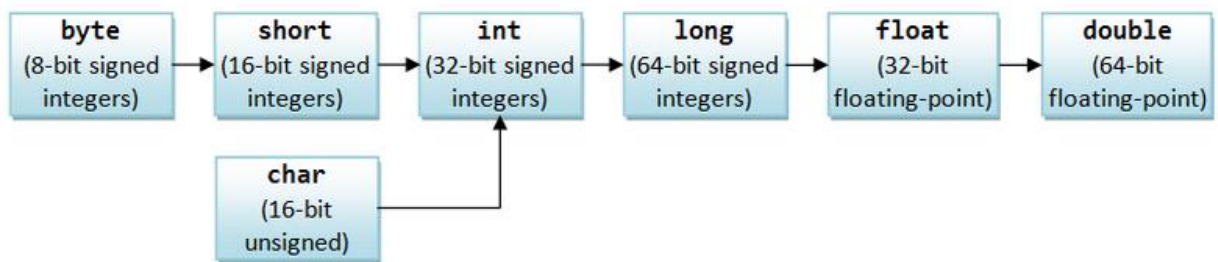
В другом случае выражение считается в типе `double` и результат тоже присваивается типу `double`.



Чтобы присвоить значение одного типа другому необходимо провести приведение типов. Различают:

- повышающее (разрешенное, неявное) преобразование;

- понижающее (явное) приведение типа.



При преобразовании типов, следуя по стрелкам схемы, производится неявное приведение. Если же преобразование необходимо выполнить против стрелок схемы – такое преобразование будет явным.

Расширяющее (повышающее) преобразование – результирующий тип имеет больший диапазон значений, чем исходный тип.

```
int x = 200;
long y = x;
long value1 = 200;
```

Сужающее (понижающее) преобразование – результирующий тип имеет меньший диапазон значений, чем исходный тип. В этом случае программист вынужден указывать явно в круглых скобках тип, к которому следует преобразовать значение; этим же действием разработчик подтверждает компилятору, что понимает последствия такого приведения.

```
long value2 = 1000L;
int value3 = (int)value2;
```

Напишем несколько примеров, демонстрирующих возможные последствия явного приведения типов.

```
3.2 public static void main(String[] args) {
    long x = 9876543212345L;
    int y;
    y = (int)x;
    System.out.println("x = " + x);
    System.out.println("y = " + y);
}
Результат:
```



```

<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-1...
x = 9876543212345
y = -1881568455

```

При приведении значения типа long к int, если это значение не попадает в диапазон значений типа, результат будет содержать 'мусор'.

3.3 **public static void** main(String[] args) {

```

    double x = 16.89;
    int y;

    y = (int)x;

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

```

Результат:

```

<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk...
x = 16.89
y = 16

```

При приведении типа double к типу int будет 'потерян хвостик' у вещественного числа.

3.4 **public static void** main(String[] args) {

```

    double x1 =
96968678655748879789678685969696858585879766.575886583424546462345463758796773;
    double x2 =
5424324234242423424323424234242342342.23234242342424234234242424242;

    int y1;
    int y2;

    y1 = (int)x1;
    y2 = (int)x2;

    System.out.println("x1 = " + x1 + " :\t y1 = " + y1);
    System.out.println("x2 = " + x2 + " :\t y2 = " + y2);
}

```

Результат:

```

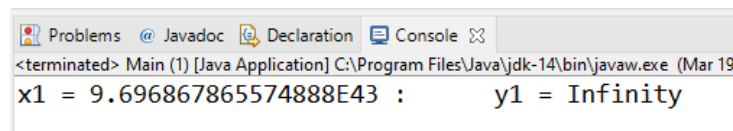
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar 19, 2021)
x1 = 9.696867865574888E43 : y1 = 2147483647
x2 = 5.424324234242424E36 : y2 = 2147483647

```

Однако, согласно стандарту IEEE754, при приведении очень большого (очень маленького) дробного числа **double** к целочисленному типу **int** результат преобразуется в максимальное (минимальное) число диапазона значений.

```
3.5 public static void main(String[] args) {  
    double x1 =  
        969686786557488797896786859696858585879766.575886583424546462345463758796773;  
  
    float y1;  
  
    y1 = (float)x1;  
  
    System.out.println("x1 = " + x1 + " :\t y1 = " + y1);  
  
}
```

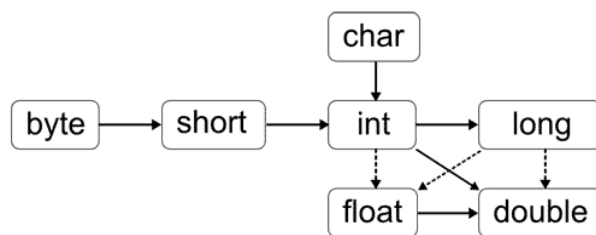
Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar 19  
x1 = 9.696867865574888E43 : y1 = Infinity
```

При приведении очень большого (очень маленького) дробного числа **double** к типу **float** результат преобразуется в положительную (отрицательную) бесконечность.

3.6 Существует три неявных приведения типа (на рисунке они обозначены пунктиром), при которых возможна потеря точности.



Например,

```
public static void main(String[] args) {  
    int x1 = 123456789;  
    int x2 = 999999999;  
    float f1 = x1;  
    float f2 = x2;  
    System.out.println("x1 = " + x1 + ", f1 = " + f1);  
    System.out.println("x2 = " + x2 + ", f2 = " + f2);  
}
```

Результат:

```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk
x1 = 123456789, f1 - 1.23456792E8
x2 = 99999999, f2 - 1.0E8
```

4 Приоритет операции

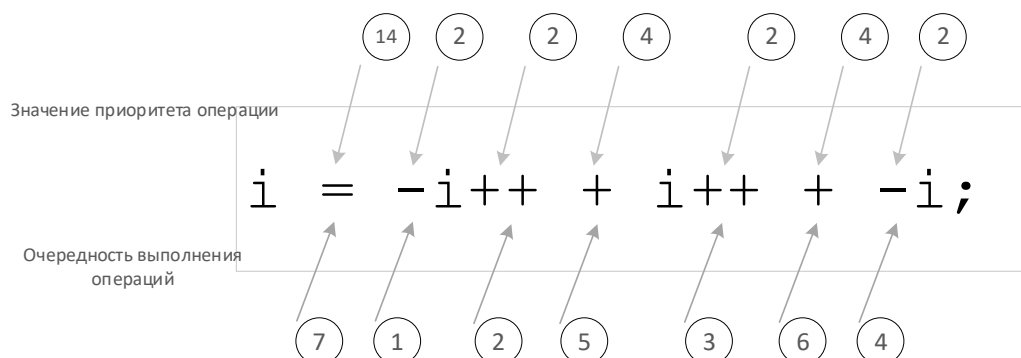
- 4.1 Пару лет назад интернет-сообщество разделилось, отвечая на вопрос, чему будет равно значение выражения

$$6/2*(2+1) - \begin{matrix} 9? \\ 1? \end{matrix}$$

Для Java последовательность решения очевидна и основана на приоритете операций. Чем выше приоритет, тем раньше будет выполнена операция. Если в выражении сочетаются операции одинакового приоритета их выполнение определяется порядком 'справа-налево' или 'слева-направо'.

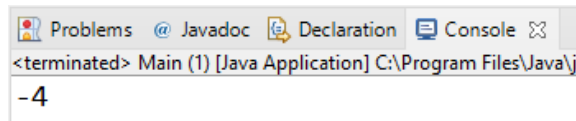
Увеличение приоритета →													
14	13	12	11	10	9	8	7	6	5	4	3	2	1
= or=	?: ?:	 	&& &&	 	^ ^	& &	== !=	> >= < <=	>> >>= << <<=	+ - * /	++ -- ~ ! new	([.)] .
справа-налево	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо	слева-направо

- 4.2 Выражения в коде следует писать так, чтобы долго не раздумывать над приоритетом. Например:



```
public static void main(String[] args) {
    int i=3;
    i = -i++ + i++ + -i;
    System.out.println(i);
}
```

Результат:



Операции ++ и -- называются операциями инкремента и декремента. Различают операции постфиксного инкремента (x++), постфиксного декремента (x--), префиксного инкремента (++x), префиксного декремента (--x).

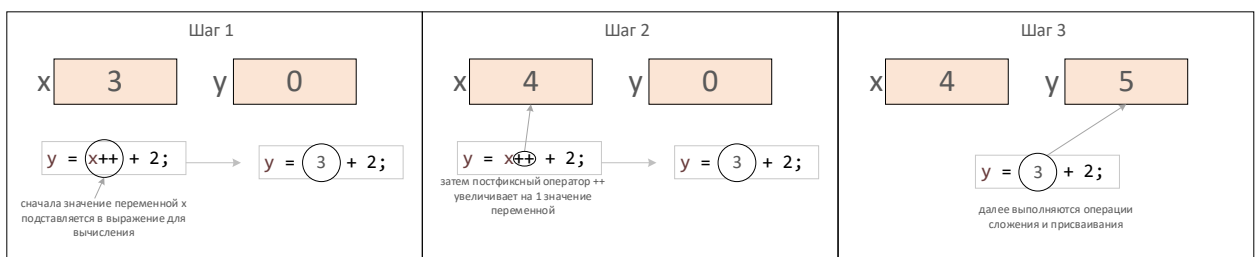
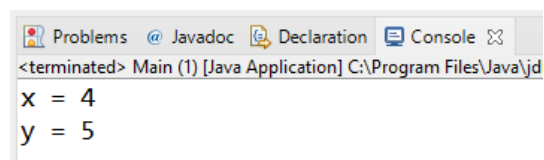
- 4.3 Если переменная, к которой применяется **постфиксный** оператор инкремента (декремента) участвует в выражении, то ее значение сначала будет подставлено в выражение, а только потом сама переменная увеличится (уменьшится) на единицу. Например:

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

    y = x++ + 2;

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

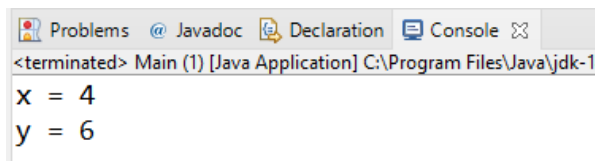
Результат:



- 4.4 Если переменная, к которой применяется **префиксный** оператор инкремента (декремента) участвует в выражении, то ее значение сначала будет увеличено (уменьшено) на единицу, а затем подставлено в выражение для вычисления. Например:

```
public static void main(String[] args) {  
    int x = 3;  
    int y = 0;  
  
    y = ++x + 2;  
  
    System.out.println("x = " + x);  
    System.out.println("y = " + y);  
}
```

Результат:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-1  
x = 4  
y = 6
```



5 Оператор if

- 5.1 Решим задачу. Напечатайте 'YES', если среди заданных целых чисел a , b , c , d есть хотя более двух четных, и 'NO', если их нет.

Оператор % - вычисляет остаток от деления.

```
public static void main(String[] args) {  
  
    int a = 16, b = 18, c = 11, d = 3;  
  
    int count = 0;
```

```

if (a % 2 == 0) {
    count++;
}

if (b % 2 == 0) {
    count++;
}

if (c % 2 == 0) {
    count++;
}

if (d % 2 == 0) {
    count++;
}

if (count > 2) {

    System.out.println("YES. В последовательности есть
более двух четных чисел.");

} else {

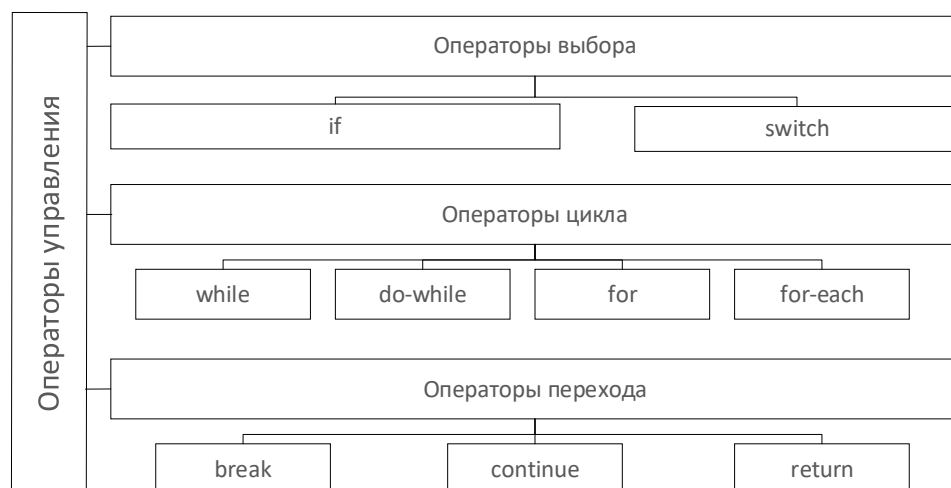
    System.out.println("NO. В последовательности нет более
двух четных чисел.");

}

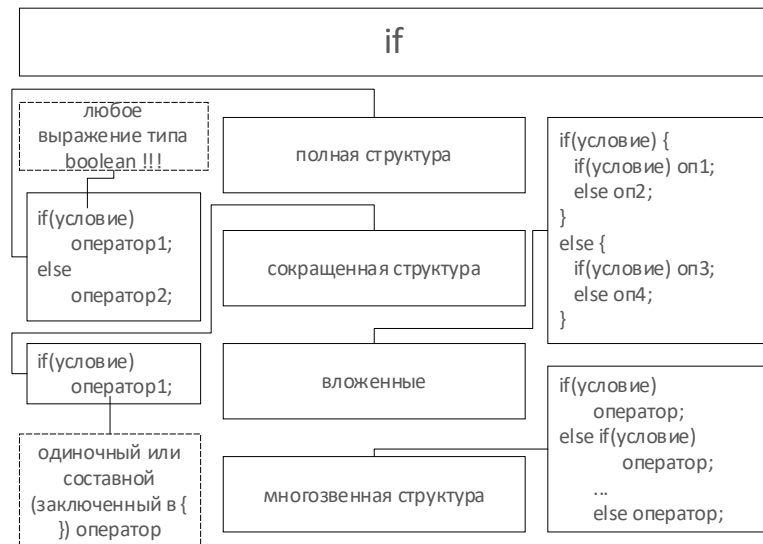
}

```

Для решения задач часто нужно использовать операторы управления.



Оператор управления if позволяет пустить поток выполнения программы по одному из двух вариантов выполнения.

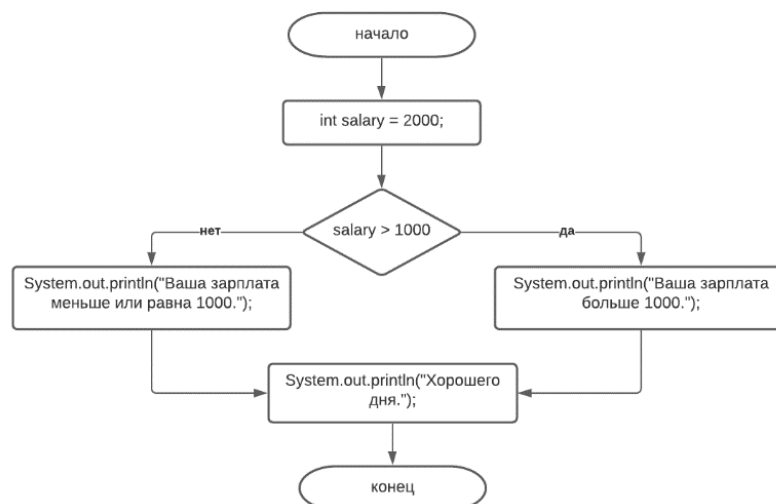


5.2 Пример применения оператора if полной структуры.

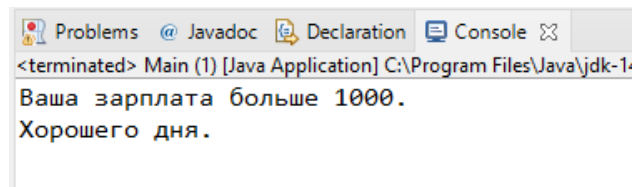
```
public static void main(String[] args) {
    int salary = 2000;

    if(salary > 1000) {
        System.out.println("Ваша зарплата больше 1000.");
    }else {
        System.out.println("Ваша зарплата меньше или равна 1000.");
    }
    System.out.println("Хорошего дня.");
}
```

Блок-схема примера.



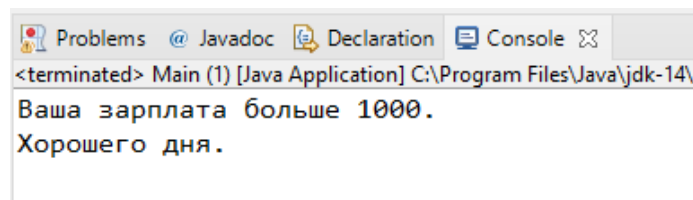
Результат выполнения:



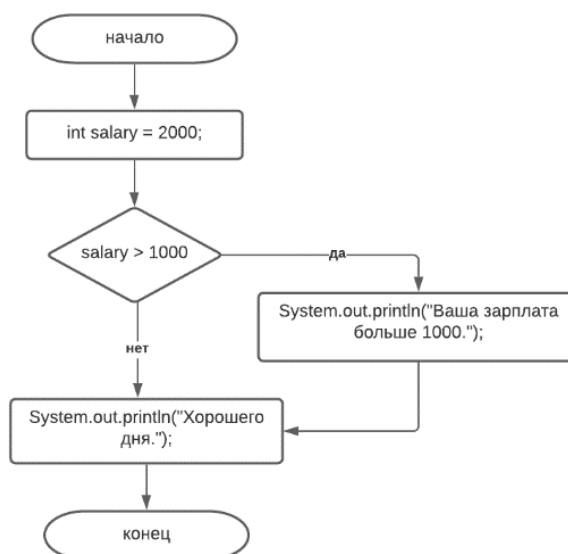
5.3 Пример применения оператора if сокращенной структуры.

```
public static void main(String[] args) {  
    int salary = 2000;  
  
    if(salary > 1000) {  
        System.out.println("Ваша зарплата больше 1000.");  
    }  
  
    System.out.println("Хорошего дня.");  
}
```

Результат выполнения:



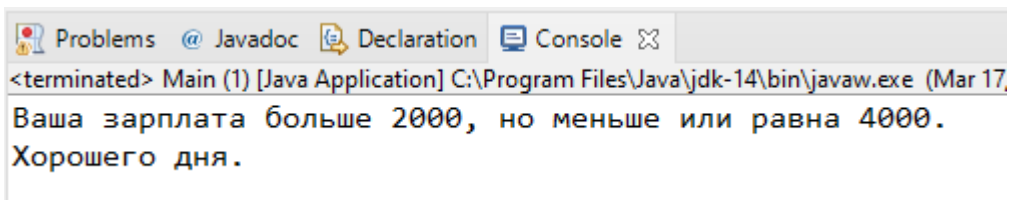
Блок-схема примера.



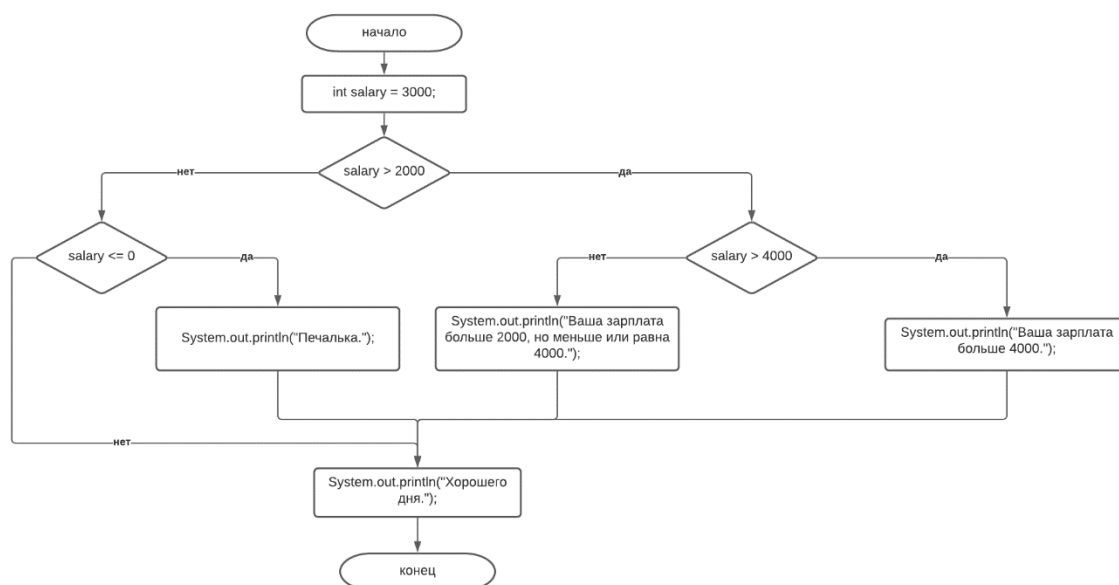
5.4 Пример применения вложенных операторов if.

```
public static void main(String[] args) {  
    int salary = 3000;  
  
    if(salary > 2000) {  
        if(salary > 4000) {  
            System.out.println("Ваша зарплата больше 4000.");  
        }else {  
            System.out.println("Ваша зарплата больше 2000, но  
меньше или равна 4000.");  
        }  
    }else {  
        if(salary <= 0) {  
            System.out.println("Печалька.");  
        }  
    }  
  
    System.out.println("Хорошего дня.");  
}
```

Результат выполнения:



Блок-схема примера.

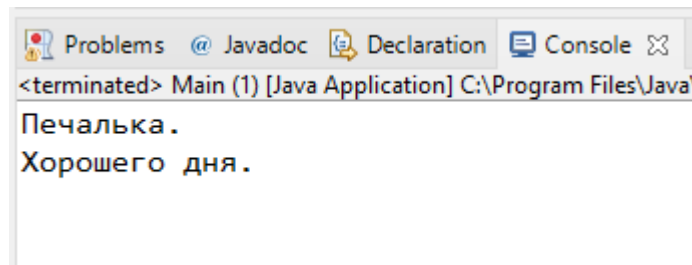


5.5

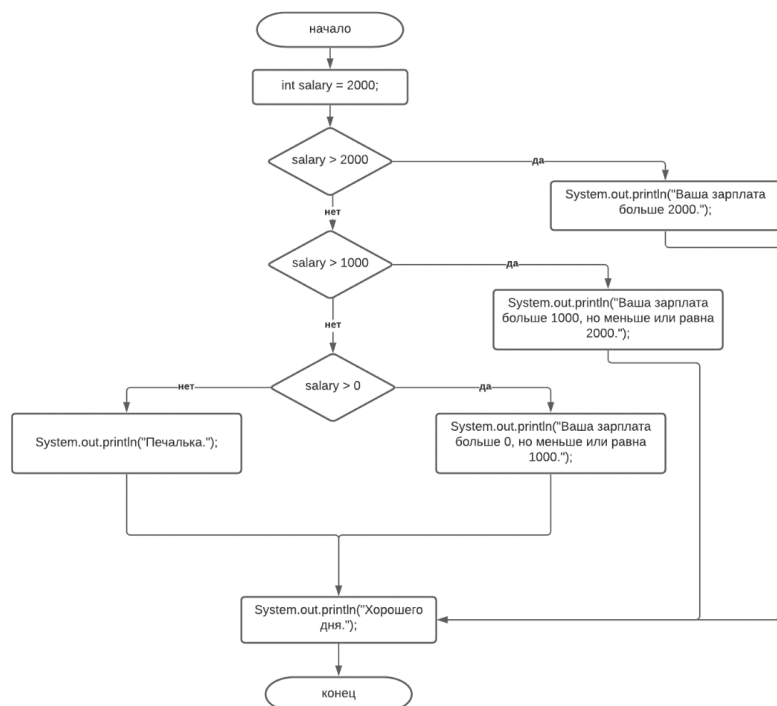
Пример применения многозвенного оператора if.

```
public static void main(String[] args) {  
    int salary = 0;  
  
    if(salary > 2000) {  
        System.out.println("Ваша зарплата больше 2000.");  
    }else if (salary > 1000) {  
        System.out.println("Ваша зарплата больше 1000, но  
меньше или равна 2000.");  
    }else if (salary > 0) {  
        System.out.println("Ваша зарплата больше 0, но меньше  
или равна 1000.");  
    }else {  
        System.out.println("Печалька.");  
    }  
  
    System.out.println("Хорошего дня.");  
}
```

Результат выполнения:

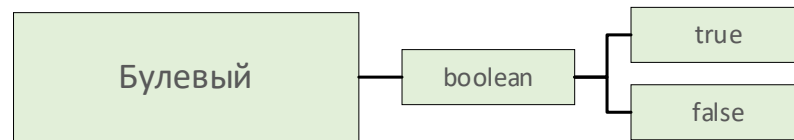


Блок-схема примера.



В качестве выражения оператора `if` нужно использовать выражение, результатом которого является логическое значение `true` или `false`, например, операторы отношения.

Логическое значение в языке `java` хранится в переменных типа `boolean`, размер которого в спецификации не определен и зависит от реализации платформы.



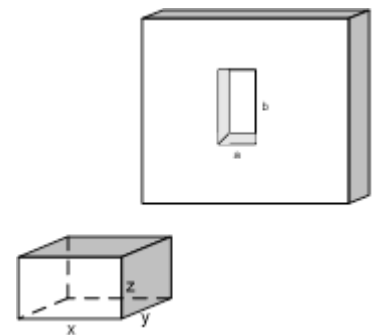
```
public static void main(String[] args) {  
    double d1 = 5.6, d2 = 3.4;  
  
    boolean b1, b2, b3, b4, b5, b6;  
  
    b1 = d1 > d2;    // строго больше  
    b2 = d1 >= d2;   // больше или равно  
    b3 = d1 < d2;    // строго меньше  
    b4 = d1 <= d2;   // меньше или равно  
    b5 = d1 == d2;   // строго равно  
    b6 = d1 != d2;   // строго не равно  
}
```

6

Составные логические выражения

6.1 Решим задачу. Заданы размеры a , b прямоугольного отверстия и размеры x , y , z кирпича. Определить, пройдет ли кирпич через отверстие.

```
public static void main(String[] args) {  
  
    double a = 10, b = 8;  
    double x = 9, y = 9, z = 20;  
  
    if ((a > x) & (b > y)) {  
        System.out.println("Проидет.");  
    } else if ((a > y) & (b > x)) {  
        System.out.println("Проидет.");  
    } else if ((a > x) & (b > z)) {  
        System.out.println("Проидет.");  
    } else if ((a > z) & (b > x)) {  
        System.out.println("Проидет.");  
    } else if ((a > y) & (b > z)) {  
        System.out.println("Проидет.");  
    }  
}
```



```

    } else if ((a > z) & (b > y)) {
        System.out.println("Проидет.");
    } else {
        System.out.println("Не проидет.");
    }
}

```

В решении этой задачи использовался логический оператор И(&&, &, AND). Логические операции выполняются над значениями типа **boolean** логическими операторами 😊.

if ((a > x) && (b > y)) {

Diagram illustrating the logical AND operator (&&) in the condition (a > x) && (b > y). Brackets above each sub-expression (a > x) and (b > y) are labeled 'true/false', indicating they are boolean expressions. A bracket above the entire condition is also labeled 'true/false'.

6.2 Результатом выполнения логической операции И(&&) является истина (true), когда два операнда также истинны.

```

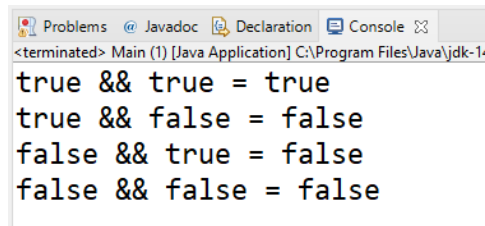
public static void main(String[] args) {

    boolean TRUE = true;
    boolean FALSE = false;

    System.out.println(TRUE + " && " + TRUE + " = " + (TRUE && TRUE));
    System.out.println(TRUE + " && " + FALSE + " = " + (TRUE && FALSE));
    System.out.println(FALSE + " && " + TRUE + " = " + (FALSE && TRUE));
    System.out.println(FALSE + " && " + FALSE + " = " + (FALSE && FALSE));
}

```

Результат:



```

<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-1.
true && true = true
true && false = false
false && true = false
false && false = false

```

6.3 Результатом выполнения логической операции ИЛИ(||) является истина, когда хотя бы один операнд равен истине.

```

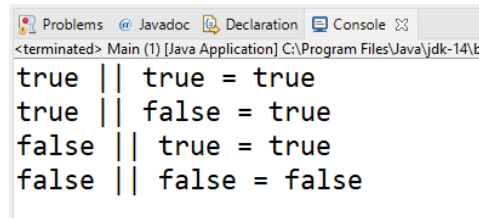
public static void main(String[] args) {

    boolean TRUE = true;
    boolean FALSE = false;

    System.out.println(TRUE + " || " + TRUE + " = " + (TRUE || TRUE));
    System.out.println(TRUE + " || " + FALSE + " = " + (TRUE || FALSE));
    System.out.println(FALSE + " || " + TRUE + " = " + (FALSE || TRUE));
    System.out.println(FALSE + " || " + FALSE + " = " + (FALSE || FALSE));
}

```

Результат:



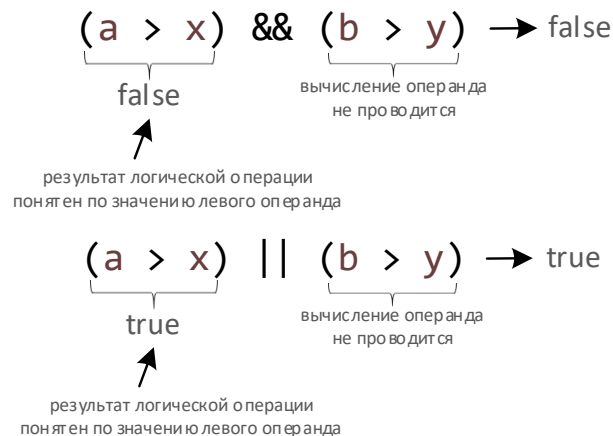
```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-14\bin
true || true = true
true || false = true
false || true = true
false || false = false
```

6.4 В Java есть две формы логического оператора И(&&) и ИЛИ(||): вычисляющих по полной и по сокращенной форме:

&& и **||** - вычисление по сокращенной схеме;

& и **|** - вычисление по полной схеме.

При вычислении с помощью операторов полной схемы результат определяется только после выполнения и левого, и правого операндов. При использовании операторов сокращенной схемы результат может быть вычислен на основании выполнения только левого операнда, а правый в этом случае не выполняется.



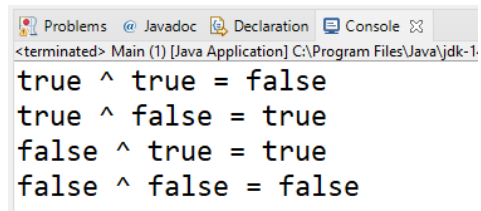
6.5 Результатом выполнения логической операции XOR (^, исключающее или) является истина, когда только один из операндов равен истине.

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

    boolean TRUE = true;
    boolean FALSE = false;

    System.out.println(TRUE + " ^ " + TRUE + " = " + (TRUE ^ TRUE));
    System.out.println(TRUE + " ^ " + FALSE + " = " + (TRUE ^ FALSE));
    System.out.println(FALSE + " ^ " + TRUE + " = " + (FALSE ^ TRUE));
    System.out.println(FALSE + " ^ " + FALSE + " = " + (FALSE ^ FALSE));
}
```

Результат:



```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-1.
true ^ true = false
true ^ false = true
false ^ true = true
false ^ false = false
```

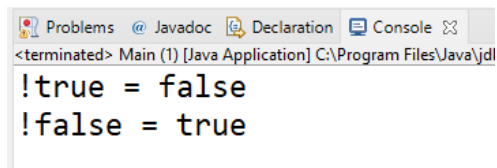
6.6 Логическая операция НЕ(!) инвертирует значение, к которому применяется.

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

    boolean TRUE = true;
    boolean FALSE = false;

    System.out.println("!" + TRUE + " = " + (!TRUE));
    System.out.println("!" + FALSE + " = " + (!FALSE));
}
```

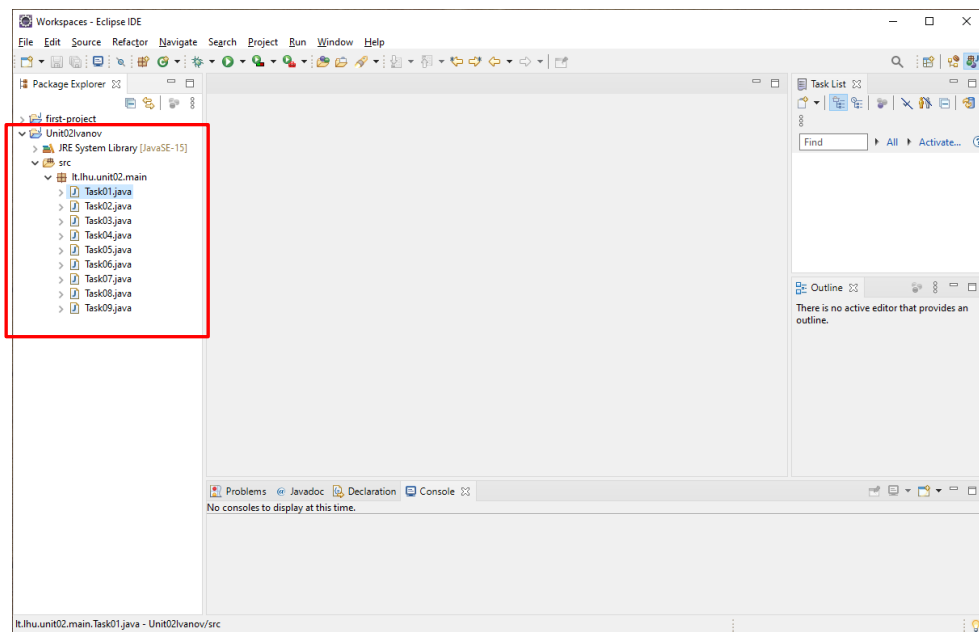
Результат:



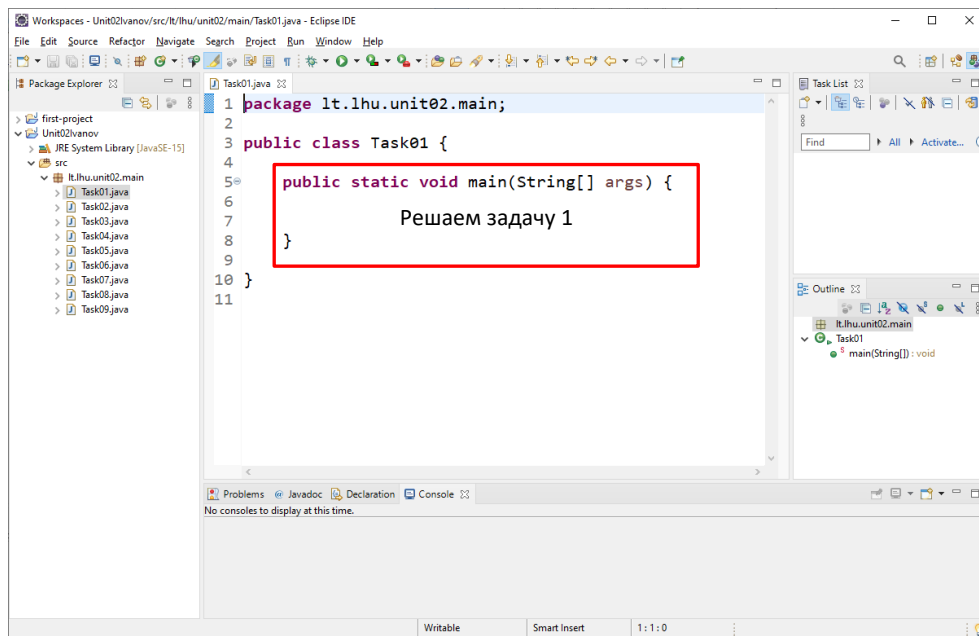
```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-1.
!true = false
!false = true
```

Подготовка рабочего пространства для решения задач

1. Создайте проект с именем Unit02[Surname] (например, Unit02Ivanov).
2. Далее создайте класс Task01 и поместите его в пакет `by.epam.unit02.main`.
3. Аналогично создайте еще 8 классов Task02 ... Task09.



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

Найдите значение функции: $z = (a - 3) * b / 2 + c$.

Задача 2

Перераспределить значения переменных x и y так, чтобы в x оказалось большее из этих значений, а в y - меньшее.

Задача 3

Дано натуральное число T , которое представляет длительность прошедшего времени в секундах. Вывести данное значение длительности в часах, минутах и секундах в следующей форме:

HHч MMмин SSс.

Задача 4

Вычислить расстояние между двумя точками с данными координатами (x_1, y_1) и (x_2, y_2) .

Задача 5

Вычислить значение выражения по формуле (все переменные принимают действительные значения):

$$\frac{\sin x + \cos y}{\cos x - \sin y} * \operatorname{tg} xy$$

Задача 6

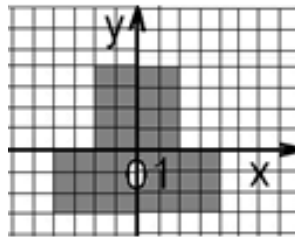
Даны натуральные числа M и N . Вывести старшую цифру дробной части и младшую цифру целой части числа M/N .

Задача 7

Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень — отрицательные.

Задача 8

Для данных областей составить линейную программу, которая печатает true, если точка с координатами (x, y) принадлежит закрашенной области, и false — в противном случае:



Задача 9

Вычислить значение функции:

$$F(x) = \begin{cases} 9, & \text{если } x \leq -3 \\ \frac{1}{x^2 + 1}, & \text{если } x > 3 \end{cases}$$