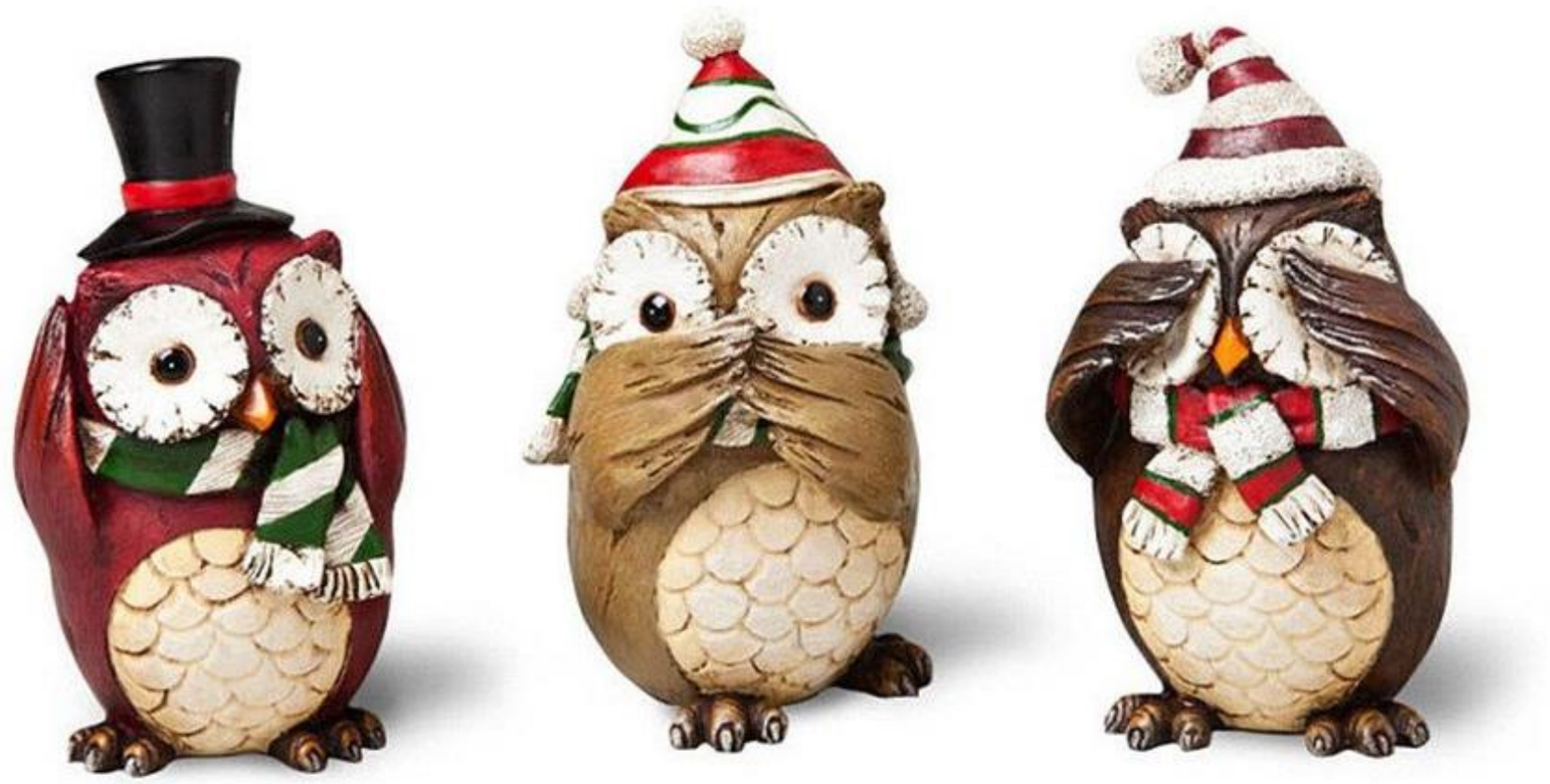
OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

12 – Operators and Decision Constructs (Часть 4)

Дмитрий Коган

Как меня слышно и видно?



Если нет – напишите, если слышите – смайлик в чат.

Цели :

- **Подключим логику**
- **Сообразим на троих (операндов)**
- **Взорвём мозг с Умой Турман**

Начинаем?

Темы экзамена

- ❑ Java Basics
- ❑ Working with Java Data Types
- ❑ Using Operators and Decision Constructs
- ❑ Creating and Using Arrays
- ❑ Using Loop Constructs
- ❑ Working with Methods and Encapsulation
- ❑ Working with Inheritance
- ❑ Handling Exceptions
- ❑ Working with Selected classes from the Java API

Подтемы экзамена

Using Operators and Decision Constructs

- **Use Java operators; use parentheses to override operator precedence**
- **Test equality between Strings and other objects using == and equals ()**
- **Create if and if/else and ternary constructs**
- **Use a switch statement**

Логические операторы

Логические операторы

Operator	Description
&	Logical AND is `true` only if both values are `true`.
|	Inclusive OR is `true` if at least one of the values is `true`.
^	Exclusive XOR is `true` only if one value is `true` and the other is `false`.

Основы

Boolean logical operators include the unary operator `!` (*logical complement*) and the binary operators `&` (*logical* AND), `|` (*logical inclusive* OR), and `^` (*logical exclusive* OR, also called *logical* XOR). These operators can be applied to `boolean` or `Boolean` operands, returning a `boolean` value. The operators `&`, `|`, and `^` can also be applied to integral operands to perform *bitwise* logical operations, but are not in the scope of this book.

Таблицы

x & y
(AND)

	y = true	y = false
x = true	true	false
x = false	false	false

x | y
(INCLUSIVE OR)

	y = true	y = false
x = true	true	true
x = false	true	false

x ^ y
(EXCLUSIVE OR)

	y = true	y = false
x = true	false	true
x = false	true	false

Применяем

```
boolean eyesClosed = true;
boolean breathingSlowly = true;

boolean resting = eyesClosed | breathingSlowly;
boolean asleep = eyesClosed & breathingSlowly;
boolean awake = eyesClosed ^ breathingSlowly;
System.out.println(resting);  // true
System.out.println(asleep);   // true
System.out.println(awake);    // false
```

Постфикс

```
i=0;
if (i == 0 & i++ == 0) System.out.println("postfix"); // true; print "postfix"
```

Порядок вычисления

```
boolean b1, b2, b3 = false, b4 = false;
Boolean b5 = true;
b1 = 4 == 2 & 1 < 4;              // false, evaluated as (b1 = ((4 == 2) & (1 <
4)))
b2 = b1 | !(2.5 >= 8);            // true
b3 = b3 ^ b5;                     // true, unboxing conversion on b5
b4 = b4 | b1 & b2;                // false
```

```
   (b4 = (b4 | (b1 & b2)))
⟹ (b4 = (false | (b1 & b2)))
⟹⟹ (b4 = (false | (false & b2)))
⟹⟹⟹ (b4 = (false | (false & true)))
⟹⟹⟹⟹ (b4 = (false | false))
⟹⟹⟹⟹⟹ (b4 = false)
⟹⟹⟹⟹⟹⟹ false
```

AND > XOR > OR

```
if  (true ^ true & false);    // true
if  ((true ^ true) & false); // false
if  (true ^ (true & false)); // true
```

Составной оператор

Expression	Given a and b are of type boolean or Boolean, the expression is evaluated as:
b &= a	b = (b & (a))
b ^= a	b = (b ^ (a))
b |= a	b = (b | (a))

Составной оператор

```
boolean b1 = false, b2 = true, b3 = false;
Boolean b4 = false;
b1 |= true;                 // true
b4 ^= b1;                   // (1) true, unboxing in (b4 ^ (b1)), boxing on
assignment
b3 &= b1 | b2;              // (2) false, b3 = (b3 & (b1 | b2))
b3 = b3 & b1 | b2;          // (3) true,  b3 = ((b3 & b1) | b2)
```

Короткое замыкание

Operator	Description
`&&`	Short-circuit AND is `true` only if both values are `true`. If the left side is `false`, then the right side will not be evaluated.
`||`	Short-circuit OR is `true` if at least one of the values is `true`. If the left side is `true`, then the right side will not be evaluated.

Ошибка компиляции

```
//if (5 || 6); // operator '||' cannot be applied to 'int', 'int'
```

Действуем аккуратно

```
int marks = 8;
int total = 10;
System.out.println(total < marks && ++marks > 5);
System.out.println(marks);
System.out.println(total == 10 || ++marks > 10);
System.out.println(marks);
```

1. Prints false
2. Prints 8
3. Prints true
4. Prints 8

Тренируемся

```
Boolean b1 = 4 == 2 && 1 < 4;     // false, short-circuit evaluated as
                                  // (b1 = ((4 == 2) && (1 < 4)))
boolean b2 = !b1 || 2.5 > 8;      // true, short-circuit evaluated as
                                  // (b2 = ((!b1) || (2.5 > 8)))
Boolean b3 = !(b1 && b2);         // true
boolean b4 = b1 || !b3 && b2;     // false, short-circuit evaluated as
                                  // (b4 = (b1 || ((!b3) && b2)))
```

```
  (b4 = (b1 || ((!b3) && b2)))
⇒ (b4 = (false || ((!b3) && b2)))
⇒⇒ (b4 = (false || ((!true) && b2)))
⇒⇒ (b4 = (false || ((false) && b2)))
⇒⇒ (b4 = (false || false))
⇒ (b4 = false)
```

Поступим методично

```
public class ShortCircuit {
  public static void main(String[] args) {
    // Boolean b1 = 4 == 2 && 1 < 4;
    Boolean b1 = operandEval(1, 4 == 2) && operandEval(2, 1 < 4);
    System.out.println();
    System.out.println("Value of b1: " + b1);

    // boolean b2 = !b1 || 2.5 > 8;
    boolean b2 = !operandEval(1, b1) || operandEval(2, 2.5 > 8);
    System.out.println();
    System.out.println("Value of b2: " + b2);

    // Boolean b3 = !(b1 && b2);
    Boolean b3 = !(operandEval(1, b1) && operandEval(2, b2));
    System.out.println();
    System.out.println("Value of b3: " + b3);

    // boolean b4 = b1 || !b3 && b2;
    boolean b4 = operandEval(1, b1) || !operandEval(2, b3) && operandEval(3,
b2);
    System.out.println();
    System.out.println("Value of b4: " + b4);

    // boolean b5 = b1 | !b3 & b2;    // Using boolean logical operators
    boolean b5 = operandEval(1, b1) | !operandEval(2, b3) & operandEval(3,
b2);
    System.out.println();
    System.out.println("Value of b5: " + b5);
  }

  static boolean operandEval(int opNum, boolean operand) {                //
(1)
    System.out.print(opNum);
    return operand;
  }
}
```

Output from the program:

```
1
Value of b1: false
1
Value of b2: true
1
Value of b3: true
12
Value of b4: false
123
Value of b5: false
```

Избегаем NPE

```
if (objRef != null && objRef.equals(other)) { /*…*/ }
```

```
if(duck!=null & duck.getAge()<5) { // Could throw a NullPointerException
   // Do something
}

if(duck!=null && duck.getAge()<5) {
   // Do something
}
```

```
String name = "hello";
if (name != null && name.length() > 0)
      System.out.println(name.toUpperCase());
```

Упражнение

```
public class Logic {
  public static void main(String[] args) {
    int i = 0;
    int j = 0;

    boolean t = true;
    boolean r;

    r = (t &  0 < (i+=1));
    r = (t && 0 < (i+=2));
    r = (t |  0 < (j+=1));
    r = (t || 0 < (j+=2));
    System.out.println(i + " " + j);
  }
}
```

Select the two correct answers.

(a) The first digit printed is 1.

(b) The first digit printed is 2.

(c) The first digit printed is 3.

(d) The second digit printed is 1.

(e) The second digit printed is 2.

(f) The second digit printed is 3.

Ответ: CD

Правила де Моргана

$\neg (P \wedge Q) \Leftrightarrow (\neg P) \vee (\neg Q)$ *в Джаве пишется так:* `!(A&&B) == !A || !B;  // true`

$\neg (P \vee Q) \Leftrightarrow (\neg P) \wedge (\neg Q)$ *в Джаве пишется так:* `!(A||B) == !A && !B;  // true`

Упражнение

```
int score = 80;
boolean passed = !((score < 80) || (score >= 100));
System.out.println(passed);
```

Which one is false?

A. The code prints true
B. The code prints false

Ответ: А

Тернарный оператор

```
booleanExpression ? expression1 : expression2
```

Сестра таланта

```
int owl = 5;
int food;
if(owl < 2) {
    food = 3;
} else {
    food = 4;
}
System.out.println(food);  // 4
```

```
int owl = 5;
int food = owl < 2 ? 3 : 4;
System.out.println(food); // 4
```

Возвращаемые типы

- Когда справа от ассопа стоит терноп, его (в смысле, тернопские) операнды должны 1) быть одинакового типа или хотя бы допускать численное продвижение при необходимости, и 2) быть совместимыми с ассопом.

```
int stripes = 7;

System.out.print((stripes > 5) ? 21 : "Zebra");

int animal = (stripes < 9) ? 3 : "Horse";  // DOES NOT COMPILE
```

Применяем

```
boolean leapYear = false;
byte v29 = 29;
byte v28 = 28;
int daysInFebruary = leapYear ? v29 : v28;   // (1)


int i = 3;
int j = 4;
int minValue1 = i < j ? i : j;                          // (2) int
int minValue2 = i < j ? i : Double.MIN_VALUE;           // (3) double. Not OK.
double minDoubleValue = i < j ? i : Double.MIN_VALUE; // (4) double


// Assume i and j are of type int and initialized correctly.
Object obj = i < j ? i : true;          // (5) value of i boxed in Integer or
                                        //     literal true boxed in Boolean
System.out.println(i < j ? i : "Hi"); // (6) value of i boxed in Integer or
                                        //     String object "Hi"
```

Одинокий оператор

```
(i < j) ? i : j;     // Compile-time error!
```

Пустая операция

- Пустая операция (empty statement) в виде одинокой точки с запятой может появиться в довольно неожиданных местах, заставляя думать, что код не скомпилируется, хотя на деле никаких правил не нарушено:

```
if (1 > 0);                                    // VALID
```

Возврат обязателен

- Оба тернопских операнда должны что-то возвращать или представлять собой некое – в том числе вычисляемое – значение. В примере ниже код компилируется, потому что **test()** возвращает `int`, вычисление `--res` тоже дает некое значение, да и `for` получает `boolean`:

```
class A {
    static int test(int a){return a*a;}
    public static void main(String[] args) {
        int res = 10, a = 0;
        if ((res > 10 ? test(a) : --res) < 10)
            System.out.println(res);                    // печатает 9
        for( ; Math.random()<.5? true : false ; ) { }   // VALID
    }
}
```

Простое следствие: Будет комперр, если любой из операндов представляет собой `void`-метод, например, **System.out.println()** и т.п.:

```
String str = (1>0) ? "!" : System.out.println("?"); // INVALID
```

Короткое замыкание

```
int sheep = 1;
int zzz = 1;
int sleep = zzz<10 ? sheep++ : zzz++;
System.out.print(sheep+","+zzz);  // 2,1
```

```
int sheep = 1;
int zzz = 1;
int sleep = sheep>=10 ? sheep++ : zzz++;
System.out.print(sheep+","+zzz);  // 1,2
```

Вставные тернопы

`a?b:c?d:e?f:g` evaluates as `(a?b:(c?d:(e?f:g)))`

```
int n = 3;
String msg = (n==0) ? “no cookies.” : (n==1) ? “one cookie.” : “many
cookies.”;
System.out.println(“You get ” + msg); // You get many cookies.
```

KISS

```
boolean a = false;
boolean b = a = true ? n > 2 ? true : false : false;
```

KISS

```
boolean a = false;
boolean b = a = true ? n > 2 ? true : false : false;

                // Сокращается в 'n > 2'
```

Упражнение

```
class Test {
    public static void main(String[] args) {
        int x = 10;
        int y = x--;
        int z = --y;
        int a = x++;
        int b = x < y ? x < z ? x : y < z ? y : z;
        System.out.println(b);
    }
}
```

What is the result?

A. 16
B. 17
C. 18
D. 23
E. Compilation fails

Ответ: E

Вопросы?

Порядок выполнения

Два сценария

```
System.out.println(true & false == false | true);
System.out.println(b1 & b2 == b3 | b4);
```

Scenario 1: (b1 & b2) == (b3 | b4)

Scenario 2: b1 & (b2 == b3) | b4

Самая удобная таблица

UMARELSA

Types of Operators	Symbols	Example Uses
Unary operators	-, !, ++, --	-7 * 4, !myBoolean
Multiplication, division, modulus	*, /, %	7 % 4
Addition, subtraction	+, -	7 + 4
Relational operators	<, >, <=, >=	y > x
Equality operators	==, !=	y != x
Logical operators (& beats |)	&, |	myBool & yourBool
Short-circuit (&& beats ||)	&&, ||	myBool || yourBool
Assignment operators	=, +=, -=	X += 5;

Тренируемся

```
System.out.println((-7 - 4) + " " + (-(7 - 4)));          // unary (-7), beats minus
                                                          // output: -11 -3

System.out.println((2 + 3 * 4) + " " + ((2 + 3) * 4));    // * beats +
                                                          // output: 14 20

System.out.println(7 > 5 && 2 > 3);                       // > beats &&
                                                          // output: false

System.out.print((true & false == false | true) + " ");  // == beats & System.out.
print(((true & false) == (false | true)));    // output: true
```

```
-11 -3
14 20
false
true false
```

Упражнение

```
10. boolean b1 = false;
11. boolean b2;
12. int x = 2, y = 5;
13. b1 = 2-12/4 > 5+-7 && b1 != y++>5 == 7%4 > ++x | b1 == true;
14. b2 = (2-12/4 > 5+-7) && (b1 != y++>5) == (7%4 > ++x) | (b1 == true);
15. System.out.println(b1 + " " + b2);
```

A. `true true`
B. `false true`
C. `true false`
D. `false false`
E. Compilation fails
F. An exception is thrown at runtime

Ответ: А

Вопросы?

Домашнее задание

Тест

Пожалуйста, пройдите опрос

https://otus.ru/polls/17818/

Спасибо за внимание!

Ломайте голову на здоровье!