OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

09 – Operators and Decision Constructs (Часть 1)

Дмитрий Коган

Как меня слышно и видно?

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

Цели :

- Рассчитаемся по порядку
- Изучим унарные операторы
- Разберём по косточкам инкремент с декрементом

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**

Операторы, операнды и операции

Операнды

```
goldCoins = goldCoins ++;
            (operand1)
```

One Operand

```
totalCoins = silverCoins + GoldCoins;
             (operand1)    (operand2)
```

Two Operands

```
Int pIrateShares = (IsCaptain == true)? TEN_SHARES : FIVE_SHARES;
                   (operand1)           (operand2)    (operand3)
```

Three Operands

Операторы

Operator type	Operators	Purpose
Assignment	=, +=, -=, *=, /=	Assign value to a variable
Arithmetic	+, -, *, /, %, ++, --	Add, subtract, multiply, divide, and modulus primitives
Relational	<, <=, >, >=, ==, !=	Compare primitives
Logical	!, &&, ||	Apply NOT, AND, and OR logic to primitives

Вопросы?

Порядок выполнения

По порядку

```
int cookies = 4;
double reward = 3 + 2 * --cookies;
System.out.print("Zoo animal receives: "+reward+" reward points");
```

```
Zoo animal receives:     reward points
```

По порядку

```
int cookies = 4;
double reward = 3 + 2 * --cookies;
System.out.print("Zoo animal receives: "+reward+" reward points");
```

```
Zoo animal receives: 9.0 reward points
```

Приоритет операторов

Приоритет	Оператор	Операция	Ассоциативность
1	`[ ]`	индекс массива	**слева направо**
	`( )`	вызов метода	
	`.`	обращение к члену класса	
2	`++`	постфиксный / префиксный инкремент	**справа налево**
	`--`	постфиксный / префиксный декремент	
	`+ -`	унарный плюс, унарный минус	
	`~`	поразрядное дополнение (поразрядное **НЕ**)	
	`!`	логическое **НЕ** (инвертер)	
	`(type)`	приведение типов (каст, кастинг)	
	`new`	инстанциация класса (создание объекта)	
3	`* / %`	умножение, деление, взятие 'остатка'	**слева направо**
4	`+ -`	сложение, вычитание	**слева направо**
	`+`	конкатенация строк	
5	`<<`	сдвиг влево	**слева направо**
	`>>`	сдвиг вправо	
	`>>>`	беззнаковый сдвиг вправо	
6	`< <=`	меньше, меньше или равно	**слева направо**
	`> >=`	больше, больше или равно	
	`instanceof`	проверка принадлежности к типу	
7	`==`	проверка равенства значений / ссылок	**слева направо**
	`!=`	проверка неравенства значений / ссылок	
8	`&`	поразрядное **И**	**слева направо**
	`&`	булево **И**	
	`^`	поразрядное **исключающее ИЛИ**	
	`^`	булево **исключающее ИЛИ**	
	`|`	поразрядное **ИЛИ**	
	`|`	логическое **ИЛИ**	
9	`&&`	логическое **И** (оно же условное **И**)	**слева направо**
	`||`	логическое **ИЛИ** (условное **ИЛИ**)	
10	`? :`	тернарный условный оператор	**справа налево**
11	`=`	простое присваивание	**справа налево**
	`*= /= += -= %= <<= >>= >>>= &= ^= |=`	составное присваивание (как оператор, так и операция)[47]	
12	`->`	лямбда-токен	**слева направо**

Приоритет операторов

Приоритет	Оператор	Операция
↑	`[ ] ( ) .`	индекс массива – вызов метода – обращение к члену
↑	`++ -- ! (type) new`	постфикс / префикс инкр/декр – **NOT** – каст – инстанциация
↑	`* / %`	умножение, деление, 'остаток'
↑	`+ -`	сложение, вычитание – конкатенация строк
↑	`< <= > >= instanceof`	«больше/меньше» – проверка принадлежности к типу
↑	`== !=`	проверка равенства значений / ссылок
↑	`& ^ |`	булевый **AND – XOR – OR**
↑	`&& ||`	логический («короткозамкнутый») **AND – OR**
↑	`? :`	тернарное условие
↑	`= *= /= += -= %= ...`	присваивание (в любой форме)

Приоритет операторов

Помимо правил школьной арифметики (*/ против +–), важнее всего для экзамена помнить следующее:

- операторы постфиксного / префиксного инкремента или декремента имеют высший приоритет;
- все арифметические операции, включая сравнения, выполняются до проверки равенства или неравенства значений / ссылок;
- присваивание, в т.ч. в составных формах, выполняется в последнюю очередь.

Ассоциативность

Задает порядок исполнения равноправных операторов при отсутствии круглых скобок. Взять хотя бы `int a=3-2-1;` может выйти либо 0 (если двигаться слева направо), либо 2 (справа налево). Правильный ответ 0.

Left associativity implies grouping from left to right: The expression `7 - 4 + 2` is interpreted as `((7 - 4) + 2)`, since the binary operators + and − both have same precedence and left associativity.

Right associativity implies grouping from right to left: The expression `- - 4` is interpreted as `(- (- 4))` (with the result `4`), since the unary operator − has right associativity.

Вычисление операндов

```
int b = 10;
System.out.println((b=3) + b);
```

Вычисление операндов

```
int b = 10;
System.out.println((b=3) + b);
```

```
(b=3)  + b
   3    + b        b is assigned the value 3
   3    + 3
```

Вычисление операндов

```
import static java.lang.System.out;

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

    int j = 2;
    out.println("Evaluation order of operands:");
    out.println(eval(j++, " + ") + eval(j++, " * ") * eval(j, "\n"));    //
(1)

    int i = 1;
    out.println("Evaluation order of arguments:");
    add3(eval(i++, ", "), eval(i++, ", "), eval(i, "\n")); // (2) Three
arguments.
  }

  public static int eval(int operand, String str) {        // (3)
    out.print(operand + str);       // Print int operand and String str.
    return operand;                 // Return int operand.
  }

  public static void add3(int operand1, int operand2, int operand3) {    //
(4)
    out.print(operand1 + operand2 + operand3);
  }
}
```

```
Evaluation order of operands:
2 + 3 * 4
14
Evaluation order of arguments:
1, 2, 3
6
```

Продолжение следует

...когда подробнее разберёмся с операторами.

Вопросы?

Унарные операторы

Унарные операторы

Operator	Description
!	Inverts a `boolean`'s logical value
+	Indicates a number is positive, although numbers are assumed to be positive in Java unless accompanied by a negative unary operator
-	Indicates a literal number is negative or negates an expression
++	Increments a value by 1
--	Decrements a value by 1
(*type*)	Casts a value to a specific type.

Инвертер

```
boolean isAnimalAsleep = false;
System.out.println(isAnimalAsleep);  // false
isAnimalAsleep = !isAnimalAsleep;
System.out.println(isAnimalAsleep);  // true
```

Унарные минус и плюс

```
double zooTemperature = 1.21;
System.out.println(zooTemperature);  // 1.21
zooTemperature = -zooTemperature;
System.out.println(zooTemperature);  // -1.21
zooTemperature = -(-zooTemperature);
System.out.println(zooTemperature);  // -1.21
```

Нужен пробел

```
int value = - -10;                  // (-(-10)) is 10
```

Не перепутайте

```
int pelican = !5;           // DOES NOT COMPILE
boolean penguin = -true;    // DOES NOT COMPILE
boolean peacock = !0;       // DOES NOT COMPILE
```

Unary numeric promotion

Unary numeric promotion proceeds as follows:

- If the single operand is of type `Byte`, `Short`, `Character`, or `Integer`, it is unboxed. If the resulting value is narrower than `int`, it is promoted to a value of type `int` by a widening conversion.
- Otherwise, if the single operand is of type `Long`, `Float`, or `Double`, it is unboxed.
- Otherwise, if the single operand is of a type narrower than `int`, its value is promoted to a value of type `int` by a widening conversion.
- Otherwise, the operand remains unchanged.

In other words, unary numeric promotion results in an operand value that is either `int` or wider.

Unary numeric promotion

Unary numeric promotion is applied in the following expressions:

- Operand of the unary arithmetic operators + and −
- Array creation expression; for example, `new int[20]`, where the dimension expression (in this case `20`) must evaluate to an `int` value
- Indexing array elements; for example, `objArray['a']`, where the index expression (in this case `'a'`) must evaluate to an `int` value

Возвышение

```
byte b = 3;             // int literal in range. Narrowing conversion.
b = (byte) -b;          // Cast required on assignment.
```

Упражнение

Which of the following expressions are valid?

Select the three correct answers.

(a) (- 1 -)

(b) (+ + 1)

(c) (+-+-+-1)

(d) (—1)

(e) (1 * * 1)

(f) (- -1)

Ответ: BCF

Упражнение

What is the value of evaluating the following expression: (`- -1-3 * 10 / 5-1`)?

Select the one correct answer.

(a) –8

(b) –6

(c) 7

(d) 8

(e) 10

(f) None of the above

Ответ: В

Инкремент и декремент

- ++ и -- представляют собой унарные операторы инкремента и декремента, соответственно, которые к тому же можно использовать в префиксной или постфиксной нотации.
- В префиксной нотации значение переменной увеличивается / уменьшается на единицу непосредственно перед тем, как произойдет обращение к данной переменной.
- Экзамен просто обожает тестировать нас на постфиксной нотации, в рамках которой значение переменной увеличивается / уменьшается на единицу сразу после того, как эта переменная была использована.

Область применения

- Операторы инкремента и декремента допускается применять только к переменным, но не к константам или выражениям:

```
int x=0, a=1, b=2;
x++;
x = 5++;                    // INVALID
x = (a + b)++;              // INVALID
```

final значит final

```
final int x = 5;
int y = x++;
```

```
Test.java:4: cannot assign a value to final variable x
int y = x++;
        ^
```

Ассоциативность

We cannot associate increment and decrement operators. Given that `a` is a variable, we cannot write `(++(++a))`. The reason is that any operand to `++` must evaluate to a variable, but the evaluation of `(++a)` results in a value.

Как это работает

```
++i
```

```
i += 1;
result = i;
return result;
```

```
j++
```

```
result = j;
j += 1;
return result;
```

```
--i
```

```
i -= 1;
result = i;
return result;
```

```
j--
```

```
result = j;
j -= 1;
return result;
```

Пробуем

```
int parkAttendance = 0;
System.out.println(parkAttendance);     // 0
System.out.println(++parkAttendance);   // 1
System.out.println(parkAttendance);     // 1
System.out.println(parkAttendance--);   // 1
System.out.println(parkAttendance);     // 0
```

Усложняем

```
int lion = 3;
int tiger = ++lion * 5 / lion--;
System.out.println("lion is " + lion);
System.out.println("tiger is " + tiger);
```

Усложняем

```
int lion = 3;
int tiger = ++lion * 5 / lion--;
System.out.println("lion is " + lion);
System.out.println("tiger is " + tiger);
```

```
int tiger = 4 * 5 / lion--;  // lion assigned value of 4
int tiger = 4 * 5 / 4;  // lion assigned value of 3

lion is 3
tiger is 5
```

Набиваем руку

```
int a = 20;                    <-- Assign 20 to a
int b = 10;          <-- Assign 10 to b
++a;
b++;
System.out.println(a);         <-- Prints 21
System.out.println(b);         <-- Prints 11
```

Инкремент

```
int a = 20;
int b = 10;
int c = a - ++b;
System.out.println(c);
System.out.println(b);
```

Assign 20 to a

Assign 10 to b

Assign 20 – (++10), that is, 20–11, or 9, to c

Prints 9

Prints 11

```
int a = 50;
int b = 10;
int c = a - b++;
System.out.println(c);
System.out.println(b);
```

Assign 50 to a

Assign 10 to b

Assign 50 – (10++), that is, 50–10, or 40, to c

Prints 40

Prints 11

Декремент

```
double d = 20.0;
double e = 10.0;
double f = d * --e;
System.out.println(f);
System.out.println(e);
```

Assign 20.0 to d

Assign 10.0 to e

Assign 20.0 * (--10.0), that is, 20.0 * 9.0, or 180.0, to f

Prints 180.0

Prints 9.0

```
double d = 20.0;
double e = 10.0;
double f = d * e--;
System.out.println(f);
System.out.println(e);
```

Assign 20.0 to d

Assign 10.0 to e

Assign 20.0 * (10.0--), that is, 20.0 * 10.0, or 200.0, to f

Prints 200.0

Prints 9.0

Постфикс

```
1. class PostFix {
2.     static int incr1(int x){ return x++; }
3.     static int incr2(int x){ x++; return x; }
4.     public static void main(String[] args) {
5.         int a, b, c;
6.         a = b = c = 0;
7.         System.out.println(a++);
8.         System.out.println(a);
9.         System.out.println(incr1(b));
10.        System.out.println(incr2(c));
11.    }
12. }
```

Постфикс

```
1. class PostFix {
2.     static int incr1(int x){ return x++; }
3.     static int incr2(int x){ x++; return x; }
4.     public static void main(String[] args) {
5.         int a, b, c;
6.         a = b = c = 0;
7.         System.out.println(a++);                    // печатает 0
8.         System.out.println(a);                      // 1
9.         System.out.println(incr1(b));               // 0
10.        System.out.println(incr2(c));               // 1
11.    }
12. }
```

Упражнение

```
public static void main(String[] args) {
    int a = 0;
    System.out.println(a++ - a--);
    // System.out.println(a);  }
```

Каков результат?

A. -1
B. 0
C. 1

Ответ: А

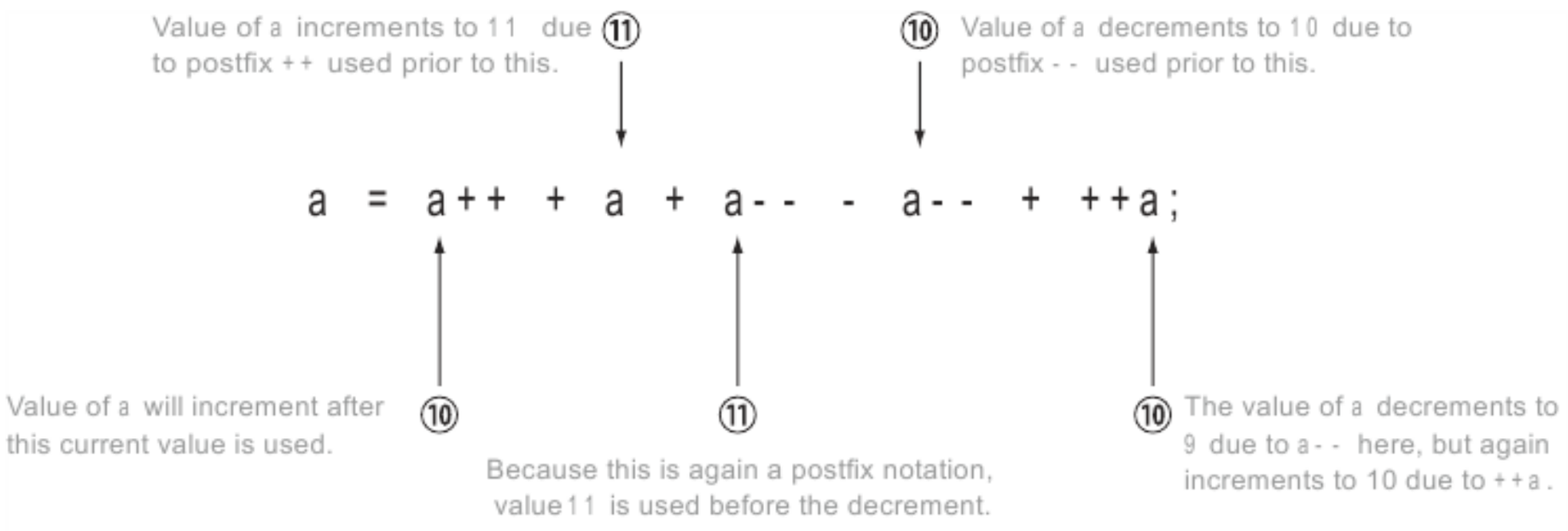
Всё вместе

```
int a = 10;
a = a++ + a + a-- - a-- + ++a;
System.out.println(a);
```

Всё вместе

```
int a = 10;
a = a++ + a + a-- - a-- + ++a;
System.out.println(a);
```

```
a = 10 + 11 + 11 - 10 + 10;
```



На дом

```
int a = 10;
a = ++a + a + --a - --a + a++;
System.out.println (a);
```

Продолжаем

```
int x = 3;
int y = ++x * 5 / x-- + --x;
System.out.println("x is " + x);
System.out.println("y is " + y);
```

Продолжаем

```
int x = 3;
int y = ++x * 5 / x-- + --x;
System.out.println("x is " + x);
System.out.println("y is " + y);
```

```
int y = 4 * 5 / x-- + --x;  // x assigned value of 4

int y = 4 * 5 / 4 + --x;  // x assigned value of 3

int y = 4 * 5 / 4 + 2;  // x assigned value of 2
```

```
x is 2
y is 7
```

Добавим оболочку

```
// (1) Prefix order: increment/decrement operand before use.
int i = 10;
int k = ++i + --i;  // ((++i) + (--i)). k gets the value 21 and i becomes 10.
--i;                // Only side effect utilized. i is 9. (expression
statement)

Integer iRef = 11;  // Boxing on assignment
--iRef;             // Only side effect utilized. iRef refers to an Integer
                    // object with the value 10. (expression statement)
k = ++iRef + --iRef;// ((++iRef) + (--iRef)). k gets the value 21 and
                    // iRef refers to an Integer object with the value 10.

// (2) Postfix order: increment/decrement operand after use.
long j = 10;
long n = j++ + j--; // ((j++) + (j--)). n gets the value 21L and j becomes 10L.
j++;                // Only side effect utilized. j is 11L. (expression
statement)
```

Возвышения нет

```
/* unary numeric promotion */
byte bstart = -10, bfinish;
bfinish = (byte) + bstart;
bfinish = (byte) - bstart;

/* no promotion */
bfinish = ++bstart;
```

Перезапись переменной

```
double x = 4.5;
x = x + ++x;          // x gets the value 10.0.
```

Следим за руками

```
int i = 0;
i = i++;
System.out.println(i);
```

```
int ii = 0;
ii++;
System.out.println(ii)
```

Следим за руками

```
int i = 0;
i = i++;
System.out.println(i); // вернет 0
```

```
int ii = 0;
ii++;
System.out.println(ii); // вернет 1
```

Упражнение

```
public class Prog1 {
  public static void main(String[] args) {
    int k = 1;
    int i = ++k + k++ + + k;      // (1)
    System.out.println(i);
  }
}
```

Select the one correct answer.

(a) The program will not compile, because of errors in the expression at (1).

(b) The program will compile and print the value `3` at runtime.

(c) The program will compile and print the value `4` at runtime.

(d) The program will compile and print the value `7` at runtime.

(e) The program will compile and print the value `8` at runtime.

Ответ: D

Упражнение

```
int[] arr = new int[1];
int index = 2;
arr[--index] = 1 / --index;
```

A. ArrayIndexOutOfBoundsException
B. ArithmeticException
C. Завершится без ошибок
D. Compile error

Ответ: В

Вопросы?

Домашнее задание

Тест

Пожалуйста, пройдите опрос

https://otus.ru/polls/17815/

Спасибо за внимание!

Инкрементируйте!