Java Basic

lecture #5. Boolean. Logic expression

Mentor: <….>

lecture #5. Boolean. Logic expression

- Основные понятия
- Арифметические (и унарные) операторы
- Реляционные операторы Java
- Логические операторы
    - Оператор логического И (&&)
    - Оператор «логическое ИЛИ» (||):
    - Оператор логического НЕ(!):
- Операторы сдвига
- Побитовые операторы

Основные понятия - операторы

- **Операторы** составляют основной строительный блок любого языка программирования.
- **Java** предоставляет множество типов операторов, которые можно использовать в зависимости от необходимости выполнения различных вычислений и функций.
- **Операторы** классифицируются на основе предоставляемой ими функциональности.

1. Арифметические операторы
2. Унарные операторы
3. Оператор присваивания
4. Реляционные операторы
5. Логические операторы
6. Тернарный оператор
7. Побитовые операторы
8. Операторы сдвига

Арифметические (и унарные) операторы

- Арифметические операторы Java — операторы, которые используются в простейших математических операциях

Арифметические:

1. operator (+) -> var1 + var2
2. operator(-) -> var1 - var2
3. operator(*) -> var1 * var2
4. operator(/) -> var1 / var2
5. operator(%) -> var1 % var2

Унарные (++, --, +, -, ~):

1. инкремент (++) -> var1++ или ++var1
2. декремент (--) -> var1-- или --var1

Арифметические (и унарные) операторы | пример

```
System.out.println("a + b = " + (a + b));

System.out.println("a - b = " + (a - b));

System.out.println("a * b = " + (a * b));

System.out.println("b / a = " + (b / a));

System.out.println("b % a = " + (b % a));

System.out.println("c % a = " + (c % a));
```

```
System.out.println("a++ = " +  (a++));

System.out.println("b-- = " +  (a--));


System.out.println("d++ = " +  (d++));

System.out.println("++d = " +  (++d));
```

Реляционные операторы (операторы сравнения)

- Реляционные операторы Java — это набор бинарных операторов, используемых для проверки отношений между двумя операндами
- Возвращают логический результат после сравнения

Синтаксис:
var1 <relation operator> var2

Примеры:
1. operator (==) -> var1 == var2
2. operator(!=) -> var1 != var2
3. operator(>) -> var1 > var2
4. operator(<) -> var1 < var2
5. operator(>=) -> var1 >= var2
6. operator(<=) -> var1 <= var2

Реляционные операторы (операторы сравнения) | пример

```
System.out.println("a == b = " + (a == b) );

System.out.println("a != b = " + (a != b) );

System.out.println("a > b = " + (a > b) );

System.out.println("a < b = " + (a < b) );

System.out.println("b >= a = " + (b >= a) );

System.out.println("b <= a = " + (b <= a) );
```

Логические операторы

- Логические операторы используются для выполнения логических операций «И», «ИЛИ» и «НЕ»
- Используется для проверки условия или нескольких условий для принятия решения
- Второе условие не оценивается, если первое ложно

Синтаксис:
cond1 <logical operator> cond2

Примеры:

1. Оператор И (AND) (&&) –> если (cond1 && cond2) -> если true выполнить, иначе не делать

2. Оператор ИЛИ (OR) (||) –> если (cond1 || cond2) -> если один из них true, выполнить, иначе не выполнять

3. Оператор НЕ (NOT) (!) –> !(var1 < var2) -> false, если var1 меньше, чем var2

Логические операторы

1. Логический оператор «И» (&&)
   - cond1 && cond2 возвращает true, когда оба cond1 и cond2 истинны (т.е. ненулевые).

2. Логический оператор «ИЛИ» (||)
   - Если хотя бы один из двух дает истину, оператор возвращает истину.
   - Чтобы результат был ложным, оба условия должны возвращать false.

3. Логический оператор НЕ (!)
   - если условие ложно, операция возвращает истину, а когда условие истинно, операция возвращает ложь.

Логические операторы | пример

```
boolean a = true;

boolean b = false;


System.out.println("a && b = " + (a && b));

System.out.println("a || b = " + (a || b));

System.out.println("!(a && b) = " + !(a && b));
```

Побитовые операторы

работа с бинарным представлением чисел

& (побитовое И). Равно 1, если соответствующие биты в операндах также равны 1. Во всех остальных случаях значение результирующего бита равно 0

| (побитовое ИЛИ). Равно 1, если соответствующий бит в любом из операндов равен 1.

^ (побитовое исключающее ИЛИ). Равно 1, если соответствующий бит только в одном из операндов равен 1. Во всех других случаях результирующий бит равен 0.

~ (побитовый унарный оператор NOT). Меняет значение на “полярное” в бинарном представлении

```
int one = 17;
int two = 21;

System.out.println(Integer.toBinaryString(one));
System.out.println(Integer.toBinaryString(two));

System.out.println((one & two) + " " + Integer.toBinaryString((one & two)));
System.out.println((one | two) + " " + Integer.toBinaryString((one | two)));
System.out.println((one ^ two) + " " + Integer.toBinaryString((one ^ two)));
```

Побитовые сдвиговые операторы

работа с бинарным представлением чисел

<< (сдвиг влево). Смещает все биты влево на указанное количество позиций

>> (сдвиг вправо). Смещает все биты вправо на указанное количество позиций (без учета знака)

>>> (сдвиг вправо). Смещает все биты вправо на указанное количество позиций (с учетом знака)

```
int one = 64;
int res1 = one << 3;

System.out.println(Integer.toBinaryString(one) + " original " + one);
System.out.println(Integer.toBinaryString(res1) + " << " + res1);

int res2 = one >> 3;
System.out.println(Integer.toBinaryString(res2) + " >> " + res2);
```