Державний університет телекомунікацій
Навчально-науковий інститут телекомунікацій та інформатизації
Кафедра комутаційних систем

Методичне керівництво для виконання лабораторної роботи

Оператори мови програмування Java.

з дисципліни «Програмне забезпечення телекомунікаційних систем».
освітньо-кваліфікаційного рівня магістр

Київ - 2014

Укладачі: проф. каф. КС Кунах Н.І.
асистент. каф. КС Невдачина О.В.

Методичні вказівки обговорені і схвалені
на засіданні кафедри КС

Протокол № _____
від «___» _____ 2013р.

Тема: Оператори у мові програмування Java.

Мета заняття: Вивчення базових основ операторів у мові програмування Java. Отримання основних навичок написання найпростіших програм мовою Java.

Час заняття: 90 хвилин.

Список літератури.

1. Гребешков А.Ю. «Микропроцессорные системы и программное обеспечение в средствах связи», ПГУТИ Самара, 2009. (надається в електронному вигляді).

2. Голицина О.Л., Партыка Т.Л. «Програмное обеспечение», 2-ое издание, Москва, 2008. (надається в електронному вигляді).

Зміст заняття.

1. **Ознайомча частина.**

Перевірка присутності студентів, оголошення теми заняття та порядок його проведення.

2. Матріал для вивчення.

ОПЕРАТОРИ

Оператори в мові Java - це спеціальні символи, які повідомляють транслятор про те, що ви хочете виконати операцію з деякими операндами. Типи операцій вказуються за допомогою операторів, а операнди - це змінні, вирази або літерали. Деякі оператори вимагають одного операнда, їх називають унарними. Одні оператори ставляться перед операндами і називаються префіксними, інші - після, їх називають постфіксними операторами. Більшість же операторів ставлять між двома операндами, такі оператори називаються інфіксне бінарними операторами. Існує тернарний оператор, що працює з трьома операндами. У Java є 44 вбудованих оператора. Їх можна розбити на 4 класу - арифметичні, бітові, оператори порівняння та логічні.

Арифметичні оператори

Арифметичні оператори використовуються для обчислень так само як в алгебрі (див. таблицю із зведенням арифметичних операторів нижче). Допустимі операнди повинні мати числові типи. Наприклад, використовувати ці оператори для роботи з логічними типами не можна, а для роботи з типом char можна, оскільки в Java тип char - це підмножина типу int.

Таблиця 1. Таблиця арифметичних операторів

№	Оператор	Результат	Оператор	Результат
1	+	**Сложение**	+ =	**Сложение с присваиванием**
2	-	**Вычитание (также унарный минус)**	- =	**Вычитание с присваиванием**
3	*	**Умножение**	* =	**Умножение с присваиванием**
4	/	**Деление**	/ =	**Деление с присваиванием**
5	%	**Деление по модулю**	% =	**Деление по модулю с присваиванием**
6	++	**Инкремент**	--	**Декремент**

Нижче як приклад, наведена проста програма, що демонструє використання операторів. Зверніть увагу на те, що оператори працюють як з цілими літералами, так і зі змінними.

```
public class BasicMath {
        public static void main(String args[]) {
        int a = 1 + 1;
        int b = a* 3;
        int v = b / 4;
        int d = b - a;
        int e = -d;
        System.out.print("a = " + a);
        System.out.print("b = " + b);
        System.out.print("v = " + v);
        System.out.print("d = " + d);
        System.out.println("e = " + e);

        }

}
```

Виконавши цю програму, ви повинні отримати наведений нижче результат:

a = 2
b = 6
c = 1
d = 4
e = -4

Оператор ділення по модулю

Оператор ділення по модулю - оператор mod, позначається символом%. Цей оператор повертає залишок від ділення першого операнда на другий. Функція mod в Java працює не тільки з цілими, а й з речовими типами. Наведена нижче програма ілюструє роботу цього оператора.

```
class Modulus {
public static void main (String [] args) {
int x = 42;
double y = 42.3;
System.out.print("x mod 10 = " + x % 10);
System.out.println("y mod 10 = " + y % 10);
}
}
```

Виконавши цю програму, ви отримаєте наступний результат:

x mod 10 = 2 y mod 10 = 2.3

Арифметичні оператори присвоювання

Для кожного з арифметичних операторів є форма, в якій одночасно із заданою операцією виконується присвоювання. Якщо потрібно написати a + = 4, то з таким же успіхом можна використовувати a = a +4.

Такий спосіб придатний для всіх бінарних операторів, які використовуються у виразах виду:

переменная = переменная оператор выражение;

Будь-який такий оператор можна записати в короткій формі:

переменная оператор = выражение;

Інкремент та декремент

У Java існує два оператора, званих операторами інкремента і декремента (+ + і -) і є скороченим варіантом записи для додавання або віднімання з операнда одиниці. Ці оператори унікальні в тому плані, що можуть використовуватися як у префиксной, так і в постфіксной формі. При використанні префиксной форми операнд модифікується перед виконанням операції. У постфіксной формі спочатку використовується вміст операнда, а лише після цього операнд инкрементируется або декрементируется. Наступний приклад ілюструє використання операторів інкремента і декремента.

```
class IncDec {
public class IncDec {
        public static void main(String[] args) {
                int a = 1;
                int b = 2;
                int c = ++b;
                int d = a++;
                System.out.print("a = " + a);
                System.out.print("b = " + b);
                System.out.print("c = " + c);
                System.out.println("d = " + d);

        }

}
```

Результат виконання даної програми буде таким:

a=2b=3c=3d=1

Цілочисельні бітові оператори

Для цілих числових типів даних - long, int, short, char і byte - визначено додатковий набір операторів, за допомогою яких можна перевіряти і модифікувати стан окремих бітів відповідних значень. У таблиці наведена зведення таких операторів. Оператори бітової арифметики працюють з кожним бітом як з самостійною величиною.

Таблиця 2. Оператори бітової арифметики

№	Оператор	Результат	№	Оператор	Результат
1	**&**	**побитовое И (AND)**	**8**	&=	**Побитовое И (AND) с присваиванием**

2	**1**	**побитовое ИЛИ (OR)**	**9**	|=	**побитовое ИЛИ (OR) с присваиванием**
3	^	**побитовое исключающее ИЛИ (XOR)**	**10**	^=	**побитовое исключающее ИЛИ (XOR) с присваиванием**
4	>>	**сдвиг вправо**	**11**	>>=	**сдвиг вправо с присваиванием**
5	>>>	**сдвиг вправо с заполнением нулями**	**12**	>>>=	**сдвиг вправо с заполнением нулями с присваиванием**
6	<<	**сдвиг влево**	**13**	<<=	**сдвиг влево с присваиванием**
7	~	**побитовое унарное отрицание (NOT)**			

Зрушення вліво

Оператор << виконує зрушення вліво всіх бітів свого лівого операнда на число позицій, задане правим операндом. При цьому частина бітів в лівих розрядах виходить за межі і втрачається, а відповідні праві позиції заповнюються нулями.

Зрушення вправо

Оператор >> означає в мові Java зрушення вправо. Він переміщує всі біти свого лівого операнда вправо на число позицій, задане правим операндом. Коли біти лівого операнда висуваються за саму праву позицію слова, вони губляться. При зсуві вправо звільняються старші (ліві) розряди зрушуваної числа заповнюються попереднім вмістом знакового розряду. Зроблено це для того, щоб при зсуві вправо числа зберігали свій знак.

Беззнакове зрушення вправо

Часто потрібно, щоб при зсуві вправо розширення знакового розряду не відбувалося, а звільняються ліві розряди заповнювалися б нулями. З цією метою використовується оператор беззнакового зсуву вправо >>>.

Оператори відносин

Для того щоб можна було порівнювати два значення, в Java є набір операторів, що описують ставлення і рівність. Список таких операторів наведено в таблиці 3

Таблиця 3.

№	Оператор	Результат
1	= =	**равно**
2	!=	**не равно**
3	>	**больше**
4	<	**меньше**
5	>=	**больше или равно**
6	<=	**меньше или равно**

Значення будь-яких типів, включаючи цілі і речові числа, символи, логічні значення і посилання, можна порівнювати, використовуючи оператор перевірки на рівність == і нерівність! =. Зверніть увагу - у мові Java перевірка на рівність позначається послідовністю (==). Один знак (=) - це оператор присвоювання.

Оператори відносин можуть застосовуватися тільки до операндів числових типів. З їх допомогою можна працювати з цілими, речовими і символьними типами. Кожен з операторів відносини повертає результат типу boolean, тобто або true, або false.

Булевы логические операторы

Булеві логічні оператори, перелік яких наведено в таблиці 4, оперують тільки з операндами типу boolean. Всі бінарні логічні оператори сприймають в якості операндів два значення типу boolean і повертають результат того ж типу.

Таблиця 4. Таблиця булевих логічних операторів

№	Оператор	Результат	№	Оператор	Результат
1	**&**	**логическое И (AND)**	**7**	&=	**И (AND) с присваиванием**
2	|	**логическое ИЛИ (OR)**	**8**	=	**ИЛИ (OR) с присваиванием**
3	^	**логическое исключающее ИЛИ (XOR)**	**9**	^=	**исключающее ИЛИ (XOR) с присваиванием**

4	||	**оператор OR быстрой оценки выражений (short circuit OR)**	**10**	==	**равно**
5	&&	**оператор AND быстрой оценки выражений (short circuit AND)**	**11**	!=	**не равно**
6	!	**логическое унарное отрицание (NOT)**	**12**	? :	**тернарный оператор if-then-else**

Логічні булеві оператори AND (І), OR (АБО) і XOR (виключає АБО) виконують над логічними величинами ті ж операції, що і їх аналоги з сімейства бітової логіки. Унарний оператор NOT (НЕ) інвертує логічне значення. У таблиці 5 показані результати впливу логічних операторів на різні комбінації значень операндів.

Таблиця 5

A	**B**	**OR**	**AND**	**XOR**	**NOT A**
false	**false**	**false**	**false**	**false**	**true**
true	**false**	**true**	**false**	**true**	**false**
false	**true**	**true**	**false**	**true**	**true**
true	**true**	**true**	**true**	**false**	**false**

Існують два доповнення до набору логічних операторів. Це альтернативні версії операторів AND і OR, службовці для швидкої оцінки логічних виразів. Якщо перший операнд оператора OR має значення true, то незалежно від значення другого операнда результатом операції буде величина true. Аналогічно у випадку оператора AND, якщо перший операнд - false, то значення другого операнда на результат не впливає - він завжди буде дорівнює false. Якщо ви використовуєте оператори && і | | замість звичайних форм & і |, то Java не проводить оцінку правого операнда логічного виразу, якщо відповідь ясна із значення лівого операнда. Загальноприйнятою практикою є використання операторів && і | | практично у всіх випадках оцінки булевих логічних виразів. Версії цих операторів & і | застосовуються тільки в бітової арифметики.

Пріорітети операторів

У Java діє певний порядок, або пріоритет, операцій. В елементарній алгебрі множення і ділення мають вищий пріоритет, ніж додавання і віднімання. У програмуванні також доводиться стежити за пріоритетами операцій. У таблиці 8. Зазначені в порядку убування пріоритети всіх операцій мови Java.

Таблиця 6. Таблиця пріорітетів усіх операцій

№	**Высший**			
1	**( )**	[]	•	
2	~	!		
3	*****	/	%	
4	+	**-**		
5	>>	>>>	<<	
6	>	>=	<	<=
7	==	**!=**		
8	**&**			
9	^			
10	|			
11	&&			
12	||			
13	? :			
14	=	**op=**		
	Низший			

У першому рядку таблиці наведено три незвичайних оператора, про які ми поки не говорили. Круглі дужки () використовуються для явної установки пріоритету. Квадратні дужки [] використовуються для індексування змінної-масиву. Оператор. (точка) використовується для виділення елементів із заслання на об'єкт.

Оператор присвоювання

Після того, як змінна описана, з нею можна працювати в програмі. Зокрема, їй можна присвоїти значення відповідного типу. Тоді надалі при використанні цієї змінної в будь-якому вираженні замість неї буде автоматично підставлятися це поточне значення.

Значення зв'язується з змінної за допомогою оператора привласнення. У мові Java він записується простим знаком рівності:

змінна = вираз;

Зліва від оператора присвоювання завжди вказується змінна. Вираз праворуч має відповідати змінної по типу. Воно може являти собою просто літерал (наприклад, число або символ):

x = 7; / / Змінної x присвоюється значення 7 letter = 'Q'; / / Змінної letter присвоюється значення 'Q'

У загальному випадку вираз - це те, що може бути обчислено (наприклад, результат математичної операції або результат, що повертається деяким методом):

a = 7.5 + 2.4; // змінної a присвоюється 9.9 як результат обчислень

У виразі поряд з літералами можуть брати участь інші змінні. Замість них підставляється їх поточне значення. В результаті виконання команди:

b = a + 1;

змінна b прийме значення 10.9.

Отже, оператор присвоювання діє таким чином. Спочатку обчислюється значення виразу в правій частині, а потім отриманий результат присвоюється змінної, вказаної в лівій частині. Можлива навіть така ситуація:

x = x + 4;

Ця команда збільшує поточне значення цілочисельний змінної x на 4.

А наступні команди записані неправильно і працювати не будуть:

5 = x + 7; // ліворуч повинна стояти мінлива x + 3 = 14; / / Зліва повинна стояти просто одна змінна x = 4.5; / / Змінна x може приймати тільки цілочисельні значення

Eclipse спробує вказати на помилку в цих рядках ще до виконання програми, розставивши попереджувальні знаки на полях редактора коду. Ви можете подивитися, як він це робить.

Практично для кожної бінарної операції існує своя різновид-ність оператора присвоювання. Наприклад, для операції додавання + існує унарний оператор присвоювання + =, який збільшує значення операнда на задану величину:

x += 8; // те ж саме, що x = x + 8 (x збільшується на 8)

Аналогічно для інших операцій: оператори *=, -=, /=, %=, &= ^= і т.д:

x *= 3; // те ж саме, що x = x * 3 (x збільшується в 3 рази) b1 ^= b2; // те ж саме, що b1 = b1 ^ b2

Умовний оператор if

Найпростіша форма запису умовного оператора має вигляд:
if (умова) команда

Умова в дужках являє собою логічне вираження, тобто може бути істин ним чи хибним. Якщо умова виявиться істинним, команда буде виконана, в іншому випадку нічого не відбудеться. наприклад:

if (x < 17) x = 17; // если значення змінної x менше 17, x привласнити 17
Якщо ж необхідно, щоб у разі, коли умова помилково, була виконана якась інша команда, використовують розширену форму оператора if:

if (умова) команда1 else команда2

У прикладі, розглянутому вище, ми можемо захотіти привласнити змінної x значеня 5, якщо умова x <17 не виконується (навіщо воно нам, інше питання).
if (x < 17) x = 17; else x = 5;

Якщо необхідно використовувати кілька взаємовиключних умов, їх можна записати наступним чином:
if (умова1) команда1 else if (умова2) команда2 else if (умова3) команда3 ... else командаN

Оператор циклу while

Цикл while має наступну форму:

while (умова) команда

Якщо умова в дужках (яке являє собою логічне вираження типу boolean) істинно, буде виконана команда - тіло циклу (це може бути проста команда або послідовність команд у фігурних дужках), після чого програма знову повернеться до виконання цього оператора і буде повторювати це до тих пір , поки умова не виявиться хибним.

Отже, щоб програма не увійшла в нескінченний цикл і не зависла, в тілі циклу повинна передбачатися можливість виходу, тобто, наприклад, команди в тілі циклу повинні якось впливати на змінні, що входять в умову.

Наприклад, наступний фрагмент програми виводить парні числа від 2 до 10:

```
int x = 2; while (x <= 10){ System.out.println(x); x += 2; }
```

3. Завдання для самостійного виконання:

Оголосіть дві цілочисельних змінних, надайте їм будь-які значення. Виведіть їх суму і добуток.

Підказка: Ви можете скористатися вже створеним в Eclipse проектом, вставивши потрібні команди після команди виведення рядка "Hello, world!" або замість неї.