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

Методичне керівництво для виконання лабораторної роботи №2

Змінні та масиви у мові програмування Java.

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

Київ - 2013

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: Змінні та масиви у мові програмування Java.

Мета заняття: Вивчення базових основ створення змінних та масивів 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. Завдання для виконання.

ЗМІННІ

Змінна - це основний елемент зберігання інформації в Java-програмі. Мінлива характеризується комбінацією ідентифікатора, типу і області дії. Залежно від того, де оголошена змінна, вона може бути локальною, наприклад, для коду всередині методу, або це може бути змінна екземпляра класу, доступна всім методам даного класу. Локальні області дії оголошуються за допомогою фігурних дужок.

Оголошення змінних

Всі змінні повинні бути оголошені до першого їх використання в програмі. Основна форма оголошення змінної така:

тип идентификатор [= значение] [, идентификатор [= значение]...];

Тип - це або один з вбудованих типів, або ім'я класу або інтерфейсу. Нижче наведено кілька прикладів оголошення змінних різних типів. Зверніть увагу на те, що деякі приклади включають в себе ініціалізацію початкового значення. Змінні, для яких початкові значення не вказані, автоматично инициализируются нулем.

У Java є вісім простих типів: byte, short, int, long, char, float, double і boolean. Їх можна розділити на чотири групи:

1. Цілі. До них відносяться типи byte, short, int і long. Ці типи призначені для цілих чисел зі знаком.

2. Типи з плаваючою точкою - float і double. Вони служать для представлення чисел, що мають дробову частину.

3. Символьний тип char. Цей тип призначений для представлення елементів з таблиці символів, наприклад, букв або цифр.

4. Логічний тип boolean. Це спеціальний тип, використовуваний для представлення логічних величин.

У Java на відміну від деяких інших мов відсутня автоматичне приведення типів. Розбіжність типів призводить до повідомлення про помилку. Для кожного типу строго визначені набори допустимих значень і дозволених операцій.

Таким чином, в Java визначено вісім простих типів, особливості яких представлені в таблиці:

Тип	Размер (бит)	Минимальное значение	Максимальное значение	Описание	Значение по умолчанию
boolean	—	—	—	логический тип	false
char	16	Unicode 0	Unicode 2^{16}-1	символьный тип	'\u0000'
byte	8	-128	+127	целое со знаком	0
short	16	-2^{15}	$+2^{15}$-1	целое со знаком	0
int	32	-2^{31}	$+2^{31}$-1	целое со знаком	0
long	64	-2^{63}	$+2^{63}$-1	целое со знаком	0L
float	32	3.4e-038	3.4e+038	вещественное	0.0f
double	64	1.7e-308	1.7e+308	вещественное	0.0d

Ідентифікатор - це найменування змінної. Як ідентифікатор може використовуватися будь-яка послідовність малих і великих літер, цифр і символів _ (підкреслення) і $ (долар). Ідентифікатори не повинні починатися з цифри.

Значення - це будь-який літерал або вираз, результатом якого є значення того ж (або сумісного з зазначеним в оголошенні змінної) типу. У наведеному нижче прикладі створюються три змінні, відповідні сторонам прямокутного трикутника, а потім за допомогою теореми Піфагора обчислюється довжина гіпотенузи, в даному випадку числа 5, величини гіпотенузи класичного прямокутного трикутника зі сторонами 3-4-5.

```
class Variables {
public static void main (String args []){
double a = 3;
double b = 4;
double  c = Math.sqrt (a* a + b* b);
                    System.out.println ("c = "+ c);
}
```

```
class Variables {
```

ПОЯСНЕННЯ ДО ПРОГРАМИ

Цей рядок оголошує клас з імені Variables. При створенні класу використовується ключове слово class разом з ім'ям класу / ім'ям файлу.

Зверніть увагу: прийнято, щоб ім'я класу починалося з великої літери.

Ключове слово class використовується для оголошення нового класу. Variables-ідентифікатор, що відображає назва класу. Повний опис класу робиться в межах відкритої і закритої вигнутих фігурних дужках. Фігурні дужки вказують компілятору, де починається і закінчується опис класу. Відкриття та закриття зігнутої дужки формують блок цього класу.

public static void main(String args[ ])

Ключове слово main () - основний метод. Це - рядок, з якої починається виконання програми. Всі додатки Java повинні мати один метод main (). Давайте розшифруємо кожне слово в коді.

Ключове слово public - це специфікатор доступу. Коли члену класу передує public, то до цього члену можливий доступ з коду, зовнішнього по відношенню до класу, в якому описаний даний метод. В даному випадку, main метод оголошений як public так, щоб JVM міг звернутися до цього методу.

Ключове слово static дозволяє методу main () викликатися без потреби створювати зразок класу. До об'єкту класу не можна звернутися, не створивши це. Але в цьому випадку, є копія цього методу, доступного в пам'яті після того, як клас розташований, навіть якщо не був створений зразок цього класу. Це важливо, тому що JVM викликає цей метод в першу чергу. Отже цей метод повинен бути як static і не повинен залежати від примірників будь-якого створюваного класу.

Ключове слово void каже компілятору, що метод не повертає ніякого значення.

main () - метод, який виконує специфічну задачу. Це місце з якого починається виконання всіх програм Java. Клас, який не має основного методу, може бути успішно откомпилирован, але не може бути виконаний, оскільки він не має відправної точки виконання, якої є main () метод.

String args [ ] - Один з параметрів, який передається основному методу. Будь-яка інформація, яку ми передаємо методу, отримана змінними, які згадані в межах круглої дужки методу. Ці змінні - параметри цього методу. Навіть якщо ми не повинні передавати ніякої інформації методом, назву методу має супроводжуватися порожніми круглими дужками, args [] (змінна) - масив типу String. Параметри, які передають в командному рядку, збережені в цьому масиві. Відкриття та закриття зігнутої дужки для main методу складають блок методу. Функції, які будуть виконані від основного методу повинні бути визначені в цьому блоці.

```
System.out.println( Welcome to the world of Java ) ;
```

Цей запис відображає рядок на екрані. Вивод рядка здійснюється за допомогою методу println (). println () відображає тільки рядок, яка передається з довідкою System.out.

System - клас, який є наперед визначеним і забезпечує доступ до системи.

out - вихідний потік і пов'язаний з консоллю.

Всі інструкції в Java закінчуються крапкою з комою (;).

МАСИВИ

Масив - це група змінних одного типу, доступ до яких здійснюється за допомогою загального імені. Для оголошення типу масиву використовуються квадратні дужки. У наведеній нижче рядку оголошується змінна month_days, тип якої - «масив цілих чисел типу int».

int month_days [];

Для того щоб зарезервувати пам'ять під масив, використовується спеціальний оператор new. При використанні цього оператора необхідно вказати необхідний тип елементів і невід'ємне число елементів, які потрібно мати на масиві. У наведеній нижче рядку коду за допомогою оператора new масиву month_days виділяється пам'ять для зберігання дванадцяти цілих чисел.

month_days = new int [12];

Отже, тепер month_days - це посилання на дванадцять цілих чисел. Нижче наведено приклад, в якому створюється масив, елементи якого містять число днів в місяцях року (невисокосного).

```
class Array {
public static void main (String args []) {
int month_days[];
month_days = new int[12];
month_days[0] = 31;
month_days[l] = 28;
month_days[2] =31;
month_days[3] = 30;
month_days[4] =31;
month_days[5] = 30;
month_days[6] = 31;
month_days[7] = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[ 11 ] = 31;
System.out.println("Апрель содержит” + month_days[3] +” дней.");} }
```

При запуску ця програма друкує кількість днів у квітні. Нумерація елементів масиву в Java починається з нуля, так що число днів у квітні - це month_days [3].

Є можливість автоматично ініціалізувати масиви способом, багато в чому нагадує ініціалізацію змінних простих типів. Ініціалізатор масиву являє

собою список розділених комами виразів, укладений у фігурні дужки. Коми відокремлюють один від одного значення елементів масиву. При такому способі створення масив буде містити рівно стільки елементів, скільки потрібно для зберігання значень, зазначених у списку ініціалізації.

```
class AutoArray {
public static void main(String args[]) {
int month_days[] = { 31,28,31,30,31,30, 31, 31,30,31, 30, 31 };
System.out.println("Апрель содержит " + month_days[3] + " дней."); }
}
```

Java строго стежить за тим, щоб ви випадково не записали або спробували отримати значення, вийшовши за межі масиву. При виконанні програми Java перевіряє, чи всі індекси потрапляють в допустимий діапазон. Якщо ж ви спробуєте використовувати як індексів значення, що виходять за межі масиву - негативні числа або числа, які більше або дорівнюють кількості елементів у масиві, то отримаєте повідомлення про помилку часу виконання.

Багатовимірні масиви

Насправді справжніх багатовимірних масивів в Java не існує. Зате є масиви масивів, які поводяться подібно багатовимірним масивів за винятком кількох незначних відмінностей. Наведений нижче код створює традиційну матрицю з десяти елементів типу double, кожен з яких инициализируется нулем. Внутрішня реалізація цієї матриці - масив масивів double.

```
double matrix [][] = new double [5][2];
```

Наступний фрагмент коду инициализирует така ж кількість пам'яті, але пам'ять під другу розмірність відводиться вручну. Це зроблено для того, щоб наочно показати, що матриця насправді являє собою вкладені масиви.

```
double matrix [][] = new double [5][];
matrix [0] = new double[2];
matrix[l] =new double[2];
matrix[2] = new double[2];
matrix[3] = { 0,1};
matrix[4] = { 2,3};
```

У наступному прикладі створюється матриця розміром 4 на 4 з елементами типу double, причому її діагональні елементи (ті, для яких x == y) заповнюються одиницями, а всі інші елементи залишаються рівними нулю.

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

```
double m[][]; m = new double[4][4];
m[0][0]=l;
m[l][l] = l;
m[2][2] = l;
m[3][3] = l;
System.out.println(m[0][0] +" "+ m[0][l] +" "+ m[0][2] +" "+ m[0][3]);
System.out.println(m[1][0] +" "+ m[l][l] +" "+ m[l][2] +" "+ m[l][3]);
System.out.println(m[2][0] +" "+ m[2][l] +" "+ m[2][2] +" "+ m[2][3]);
System.out.println(m[3][0] +" "+ m[3][l] +" "+ m[3][2] +" "+ m[3][3]);
}
}
```

Запустивши цю програму, ви отримаєте наступний результат:

```
1000
0100
0010
0001
```

Зверніть увагу - якщо ви хочете, щоб значення елемента було нульовим, вам не потрібно його ініціалізувати, це робиться автоматично. Для завдання початкових значень масивів існує спеціальна форма ініціалізатора, придатна і в багатовимірному випадку. У про1рамме, наведеної нижче, створюється матриця, кожен елемент якої містить твір номера рядка на номер стовпця. Зверніть увагу на той факт, що всередині ініціалізатора масиву можна використовувати не тільки літерали, а й вираження.

```
class AutoMatrix {
public static void main(String args[]) {
double m[][]={
{ 0*0,1*0,2*0,3*0 }, { 0*1,1*1,2*1,3*1 }, { 0*2,1*2,2*2,3*2 },
{0*3,1*3,2*3,3*3 } };
System.out.println(m[0][0] +" "+ m[0][l] +" "+ m[0][2] +" "+ m[0][3]);
System.out.println(m[ 1 ][0] +" "+m[l][l] +" "+m[l][2] +" "+m[l][3]);
System.out.println(m[2][0] +" "+m[2][l] +" "+m[2][2] +" "+ m[2][3]);
System.out.println(m[3][0] +" "+m[3][l] +" "+ m[3][2] +" "+ m[3][3]);
}
}
```

Запустивши цю програму, ви отримаєте наступний результат:

```
0000
0123
0246
0369
```