Програмне забезпечення ОС

Програмування мовою JAVA

Лекція 1

доц. кафедри Інформатики
Сінельнікова Т.Ф.

ХНУРЕ, Кафедра Інформатики

informatika@kture.kharkov.ua

- Мета та завдання дисципліни
- Лексичні основи
- Зарезервовані слова Java
- Зарезервовані імена методів Java
- Ідентифікатори
- Цілі числа
- Літерали з плаваючою точкою
- Оператори
- Розділювачі
- Змінна
- Оголошення змінної
- Пріоритети операторів

Мета та завдання дисципліни

Метою навчальної дисципліни є формування у студентів знань та практичних навичок по алгоритмізації та програмуванню задач платформо-незалежною мовою JAVA, Internet-програмуванню .

Після вивчення дисципліни студенти мають:

знати: основи та можливості об'єктно-орієнтованого програмування мовою Java, для вирішення задач інформатики;

уміти: застосовувати на практиці, у задачах інформатики Internet технологію і засоби об'єктно-орієнтованого програмування мовою Java

У 1990 році розробник ПО компанії Sun Microsystems Патрік Нотон (Patrick Naughton). Звернення викликало схвалення і у вищого керівництва компанії, а саме, у Білла Джоя (Bill Joy), засновника Sun Microsystems, і Джеймса Гослінга (James Gosling), начальника Нотона. Команда з шести осіб приступила до розробки нового об'єктно-орієнтованої мови програмування, який був названий Oak (дуб). Нова компанія мала найцікавішу концепцією, але не могла знайти їй відповідного застосування. Після низки невдач несподівано ситуація для компанії різко змінилася: був анонсований броузер Mosaic - так народився World Wide Web, з якого почався бурхливий розвиток Internet. Нотон запропонував використовувати Oak у створенні Internet-додатків. Так Oak став самостійним продуктом, незабаром був написаний Oak-компілятор і Oak-браузер "WebRunner". У 1995 році компанія Sun Microsystems прийняла рішення оголосити про новий продукт, перейменувавши його в Java. Коли Java виявилася в руках Internet, стало необхідним запускати Java-аплети - невеликі програми, що завантажуються через Internet. WebRunner був перейменований в HotJava.

Якості:
- простота і міць,
- безпека,
- об'єктну орієнтованість,
- надійність,
- інтерактивність,
- архітектурну незалежність,
- можливість інтерпретації,
- високу продуктивність,
- легкість у вивченні.

назва пакета	вміст
java.applet	Класи для реалізації аплетів
java.awt	Класи для роботи з графікою, текстом, вікнами і GUI
java.awt.datatransfer	Класи для забезпечення передачі інформації (Copy / Paste)
java.awt.event	Класи і інтерфейси для обробки подій
java.awt.image	Класи для обробки зображеньй
java.awt.peer	GUI для забезпечення незалежності від платформи
java.beans	API для моделі компонентів JavaBeans
java.io	Класи для різних типів вводу-виводу
java.lang	Класи ядра мови (типи, робота з рядками, тригонометричні функції, обробка виключень, легковагі процеси)
java.lang.reflect	Класи Reflection API
java.math	Класи для арифметичних операцій довільної точності
java.net	Класи для роботи в мережі Інтернет (сокети, протоколи, URL)

java.rmi	Класи, пов'язані з RMI (вилучений виклик процедур)
java.rmi.dgc	Класи, пов'язані з RMI
java.rmi.registry	Класи, пов'язані з RMI
java.rmi.server	Класи, пов'язані з RMI
java.security	Класи для забезпечення безпеки
java.security.acl	Класи для забезпечення безпеки
java.security.interfaces	Класи для забезпечення безпеки
java.sql	
java.text	Класи для забезпечення багатомовної підтримки
java.text.resources	Класи для забезпечення багатомовної підтримки
java.util	Різні корисні типи даних (стеки, сдоварі, хеш-таблиці, дати, генератор випадкових чисел)
java.util.zip	Класи для забезпечення архівації

Вихідний файл на мові Java - це текстовий файл, що містить в собі одне або декілька описів класів. Транслятор Java припускає, що вихідний текст програм зберігається у файлах з розширеннями java. Отримується в процесі трансляції код для кожного класу записується в окремому файлі вихідному, з ім'ям збігається з ім'ям класу, і розширенням

```
class HelloWorld
{
  public static void main (String args [])
  {
    System. out. println ("Hello World");
  }
}
```

Мова Java вимагає, щоб весь програмний код був укладений всередині пойменованих класів. Наведений вище текст прикладу треба записати в файл HelloWorld.java.

```
C:\>javac HelloWorld.java
C:> java HelloWorld
```

Результат:

Hello World

Лексичні основи

Програми на Java - це набір прогалин, коментарів, ключових слів, ідентифікаторів, літеральних констант, операторів і роздільників, який допускає довільне форматування тексту програм. за умови, що між окремими лексемами (між якими немає операторів або роздільників) є принаймні по одному пробілу, символу табуляції або символу перекладу рядка.

Коментарі:

a = 42; // якщо 42 - відповідь, то яким повинне бути питання?

```
/*
* ….
*/
```

Javadoc

```
class CoolApplet extends Applet
 {
 /**
* У цього метода два параметра:
* @param key – це ім’я параметра.
* @param value - це значення параметра з іменем key.
*/
void put (String key, Object value)
 {
```

Зарезервовані слова Java

abstract	boolean	break	byte	byvalue
case	cast	catch	char	class
const	continue	default	do	double
else	extends	false	final	finally
float	for	future	generic	goto
if	implements	import	inner	instanceof
int	interface	long	native	new
null	operator	outer	package	private
protected	public	rest	return	short
static	super	switch	syncronized	this
throw	throws	transient	true	try
var	void	volatile	while	

Зарезервовані імена методів Java

clone	equals	finalize	getClass	hashCode
notify	notifyAll	toString	wait	

Ідентифікатори

Ідентифікатори використовуються для іменування класів, методів і змінних. В якості ідентифікатора може використовуватися будь-яка послідовність малих і великих літер, цифр і символів _ (підкреслення) і $ (долар). Ідентифікатори не повинні починатися з цифри. Java - мова, чутливий до регістру букв.

Цілі числа - це цілий літерал. як цілі літерали можуть використовуватися також числа з основою 8 і 16 - вісімкові і шістнадцяткові. Java розпізнає вісімкові числа по стоїть попереду нулю.. Діапазон значень шістнадцятковій цифри - 0 .. 15, причому в якості цифр для значень 10 .. 15 використовується літери від A до F (або від a до f)
(Oxffff замість 65535).
Ox7fffffffffffffL або 92233720368547758O7L - це значення, найбільше для числа типу long.

Літерали з плаваючою точкою

Числа з плаваючою точкою представляють десяткові значення, у яких є дрібна частина. Їх можна записувати або в звичайному, або експоненційному форматах 2.0, 3.14159 і .6667. В експоненційному форматі після перерахованих елементів додатково вказується десятковий порядок: 6.022e23, 314159E-05, 2e +100. Якщо вам потрібно константа типу float, праворуч до літерали треба приписати символ F або f. до літералів типу double символ D або d. Значення типу double зберігаються в 64-бітовому слові, типу float - в 32-бітових.

Логічні літерали
У логічної змінної може бути лише два значення - true (істина) і false (неправда). Логічні значення true і false не перетворюються ні в яке числове подання ..
Рядкові літерали в Java виглядають точно також, як і в багатьох інших мовах - це довільний текст, укладений в пару подвійних лапок ("").

Символьні літерали

Символи в Java - це індекси в таблиці символів UNICODE. Вони являють собою 16-бітові значення. Символьні літерали поміщаються усередині пари апострофів ('). Всі видимі символи таблиці ASCII можна прямо вставляти всередину пари апострофів: - 'a', 'z', '@'.

Керуючі послідовності символів

\ ddd	Вісімковий символ (ddd)
\ uxxxx	Шістнадцяткові символ UNICODE (xxxx)
\ '	Апостроф
\ "	Лапки
\ \	Зворотній слеш
\ r	Повернення каретки (carriage return)
\ n	Переклад рядка (line feed, new line)
\ f	Переклад сторінки (form feed)
\ t	Горизонтальна табуляція (tab)
\ b	Повернення на крок (backspace)

Оператори

Оператор - це щось, що виконує певну дію над одним або двома аргументами і видає результат.

Оператори иови Java

	+=	-	-=
*	*=	/	/=
|	|=	^	^=
&	&=	%	%=
>	>=	<	<=
!	!=	++	--
>>	>>=	<<	<<=
>>>	>>>=	&&	||
==	=	~	?:
	instanceof	[ ]	

Розділювачі

Це - прості роздільники, які впливають на зовнішній вигляд і функціональність програмного коду.

Символи

() круглі дужки

Виділяють списки параметрів в оголошенні і виклику методу, також використовуються для завдання пріоритету операцій у виразах, виділення виразів в операторах управління виконанням програми, і в операторах приведення типів.

{} фігурні дужки

Містять значення автоматично ініціалізіруемих масивів, також використовуються для обмеження блоку коду в класах, методах і локальних областях видимості.

[] квадратні дужки

Використовуються в оголошеннях масивів та при доступі до окремих елементів масиву.

; крапка з комою

Розділяє оператори.

, кома

Розділяє ідентифікатори в оголошеннях змінних, також використовується для зв'язку операторів в заголовку циклу for.

. точка

Відокремлює імена пакетів від імен подпакетах і класів, також використовується для відділення імені змінної або методу від імені змінної.

Змінна - це основний елемент зберігання інформації в Java-програмі,яка характеризується комбінацією ідентифікатора, типу і області дії.

Оголошення змінної
тип ідентифікатор [= значення] [, ідентифікатор [= значення 7 ...];
Тип - це або один з вбудованих типів, тобто, byte, short, int, long, char, float, double, boolean, або ім'я класу або інтерфейсу. Змінні, для яких початкові значення не вказані, автоматично ініціалізувалися нулем.

int a, b, c; Оголошує три цілих змінних a, b, c.
int d = 3, e, f = 5; Оголошує ще три цілих змінних, ініціалізує d і f.
byte z = 22; Ініціалізує z.
double pi = 3. 14159; Оголошує число пі (не дуже точне, але все таки пі).
char x = 'x'; змінна x отримує значення 'x'.

```
class Variables {

public static void main (String args []) {

double a = 3;

double b = 4;

double c;

3 = Math.sqrt (a * a + b * b);

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

}}
```

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

Виконавши цю програму, ви отримаєте такий результат:

```
C: \> Modulus

x mod 10 = 2
y mod 10 = 2.3
```

```
class OpEquals
{
   public static void main (String args []) {
   int a = 1;
   int b = 2;
   int c = 3;
   a + = 5;
   b *= 4;
   c + = a * b;
   з% = 6;
  System.out.println ("a =" + a);
  System.out.println ("b =" + b);
  System.out.println ("c =" + c);
}}
```

результат, отриманий при запуску цієї програми:

```
C:> Java OpEquals
a = 6
b = 8
c = 3
```

```
class Bitlogic
{
   public static void main (String args []) {
   String binary [] = {"OOOO", "0001", "0010", "0011", "0100", "0101", "0110",
"0111", "1000", "1001", "1010", "1011", "1100", "1101", "1110", "1111"};
   int a = 3; / / 0 +2 +1 або двійкове 0011
   int b = 6; / / 4 +2 +0 або двійкове 0110
   int c = a | b;
   int d = a & b;
   int e = a ^ b;
   int f = (~ a & b) | (a & ~ b);
   int g = ~ a & 0x0f;
   System.out.println ("a =" + binary [a]);
  System.out.println ("b =" + binary [b]);
  System.out.println ("ab =" + binary [c]);
  System.out.println ("a & b =" + binary [d]);
  System.out.println ("a ^ b =" + binary [e]);
  System.out.println ("~ a & b | a ^ ~ Ь =" + binary [f]);
  System.out.println ("~ a =" + binary [g]);
 }}
```

Нижче наведено результат, отриманий при виконанні цієї програми:

```
C: \> Java BitLogic
a = 0011
b = 0110
a | b = 0111
a & b = 0010
a ^ b = 0101
~ a & b | a & ~ b = 0101
~ a = 1100
```

```
class ByteUShift
{
   static public void main (String args []) {
   char hex [] = {0 '', '1 ', '2', '3 ', '4', '5 ', '6', '7 ', '8', '9 ',' a ', 'b', 'з',
'd', 'e', 'f'};
   byte b = (byte) 0xf1;
   byte c = (byte) (b>> 4);
   byte d = (byte) (b>> 4);
   byte e = (byte) ((b & 0xff)>> 4);
   System.out.println ("b = 0x" + hex (b>> 4) & 0x0f] + hex [b &
0x0f]);
   System.out.println ("b>> 4 = 0x" + hex [(c>> 4) & 0x0f] +
hex [c & 0x0f]);
   System.out.println ("b>>> 4 = 0x" + hex [(d>> 4) & 0x0f] +
hex [d & 0x0f]);
   System.out.println ("(b & 0xff)>> 4 = 0x" + hex [(e>> 4) &
0x0f] + hex [e & f]);
}}
```

результат, отриманий при запуску цієї програми:

```
C: \> java ByteUShift
b = 0xf1
b>> 4 = 0xff
b>>> 4 = 0xff
b & 0xff)>> 4 = 0x0f
```

```
class BoolLogic
{    public static void main (String args []) {
     boolean a = true;
     boolean b = false;
     boolean 3 = a | b;
     boolean d = a & b;
     boolean e = a ^ b;
     boolean f = (! a & b) | (a &! b);
     boolean g =! a;
    System.out.println ("a =" + a);
    System.out.println ("b =" + b);
    System.out.println ("a | b =" + c);
    System.out.println ("a & b =" + d);
    System.out.println ("a ^ b =" + e);
    System.out.println ("! A & b | a &! B =" + f);
    System.out.println ("! A =" + g);
}}
C: \> Java BoolLogic
a = true
b = false
a | b = true
a & b = false
a ^ b = true
! a & b | a &! b = true
! a = false
```

```
class Ternary
{   public static void main (String args []) {
   int a = 42;
   int b = 2;
   int c = 99;
   int d = 0;
   int e = (b == 0)? 0: (a / b);
   int f = (d == 0)? 0: (c / d);
   System.out.println ("a =" + a);
   System.out.println ("b =" + b);
   System.out.println ("c =" + c);
   System.out.println ("d =" + d);
   System.out.println ("a / b =" + e);
   System.out.println ("c / d =" + f);
}}
```

При виконанні цієї програми виняткової ситуації ділення на нуль не виникає і виводяться наступні результати:

```
C: \> java Ternary
a = 42
b = 2
з = 99
d = 0
a / b = 21
c / d = 0
```

Пріоритети операторів

( )	[ ]	.	
~	!		
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		