Идентификаторы

Идентификаторы используются для обозначения любых объектов Java: переменных, констант, классов, объектов, интерфейсов, методов.

Правила построения идентификаторов:

1. Первый символ – любая буква латинского алфавита (или символ подчеркивания _).
2. Следующие символы: буквы латинского алфавита, цифры или символ подчеркивания.
3. Максимальная длина имени не ограничена, но значащими являются первые 255 символов.
4. Нельзя использовать в качестве идентификаторов зарезервированные слова Java:
   - названия типов: int, long, boolean;
   - ключевые слова операторов: if, else, for, while и т.п.;
5. Java - регистрочувствительный язык. В имени можно использовать заглавные и маленькие буквы (Bbb, BBB, BbB – разные идентификаторы).

Правила записи арифметических выражений

Арифметические операции

Указаны в порядке уменьшения приоритета

Математическое обозначение	Java

a+1
 	
a-1
	a++ постфиксная форма (операция применится к операнду после вычисления всего выражения, в которое операнд входит)
	++a префиксная форма (операция применится до вычисления выражения)
	
	<u>Пример 1:</u>
	int a=10;
	b=a++*2; // b = 20;
	
	<u>Пример 2:</u>
	int a=10;
	b=++a*2; // b = 22;
	a--
	--a

ab	a*b
$\frac{a}{b}$	a/b Результат зависит от типов операндов. Для целочисленных – деление нацело Пример: 15/2=7, 15.0/2 = 7.5
$\left(\frac{a}{b}\right)$ остаток от деления	a % b Пример: 5 % 2 = 1
a + b a - b	a + b a – b

<u>Примеры:</u>

$2x + \frac{3}{4}$ => 2*x + 3.0/4.0

$\frac{2x+3y}{3x^2}$ => (2*x + 3*y)/(3*x*x)

$\frac{3x}{2y}$ => 3*x/(2*y)

Арифметическое выражение вычисляется слева направо с учетом приоритетов арифметических операций.

Встроенные функции

Класс Math.

Математическое обозначение	Java
|x|	Math.abs(x)
$\sqrt{x}$ $\sqrt[3]{x}$	Math.sqrt(x) Math.cbrt(x)
sin x cos x tg x	Math.sin(x) Math.cos(x) Math.tan(x)
arcsin x arccos x arctg x	Math.asin(x) Math.acos(x) Math.atan(x)
ln x lg x $\log_a b$	Math.log(x) Math.log10(x) Math.log(b) / Math.log(a)
e^x a^b	Math.exp(x) Math.pow(a,b)
Число π	Math.PI
e (натуральное основание)	Math.E
Округление до целого	Math.round(x)
Целая часть числа	(int)x
Дробная часть числа	x-((int)x)

Примеры:

$\cos^2 x^6$ => Math.pow(Math.cos(Math.pow(x,6)),2)

$\frac{3\lg x^2 + e^{-3x}}{\sqrt{2x+1}\cdot tg3x}$ =>

(3*Math.log10(Math.pow(x,2))+Math.exp(-3*x))/(Math.sqrt(2*x+1)*Math.tan(3*x))

e – 2x => Math.E-2*x

Чтобы не писать каждый раз Math перед названием функции, можно добавить в начало программы строчку:
import static java.lang.Math.*;

Операции и операторы Java

Операция присваивания

x=2*y+1

Кроме простой операции присваивания еще есть *составные* операции: +=, -=, *=, /=, %=
a+=10; // a=a+10;

В составных операциях перед присваиванием, при необходимости, автоматически производится приведение типа:

```
byte b = 1;
b = b + 10;          // Ошибка - нельзя конвертировать Integer в Byte
b = (byte) (b + 10); // Правильно
b += 10;             // Правильно
```

Операторы ввода-вывода

Для *ввода* с клавиатуры необходимо

1) подключить библиотеку `java.util.*`.

2) создать объект `Scanner` и связать его со стандартным входным потоком System.in

```
Scanner inp = new Scanner(System.in);
```

3) Использовать один из методов:

`nextLine()` – ввод строки текста с клавиатуры

`next()` – ввод слова из строки текста с клавиатуры

`nextInt()` – ввод целочисленного значения из строки текста с клавиатуры

`nextDouble()` – ввод вещественного значения из строки текста с клавиатуры

Пример

```
double d = inp.nextDouble();
int a = inp.nextInt();
```

4) Закрыть консоль

```
inp.close()
```

Для вывода в консоль используется стандартный выходной поток `System.out` и один из методов:

1) неформатированный вывод - методы `println` и `print`. `println` отличается от `print` тем, что после вывода переводит курсор на новую строку.

Пример

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

Результат `x=5`

Пример:

```
System.out.println("a");
System.out.println("b");
System.out.println("c");
```

Результат:

```
a
b
c
```

Пример:

```
System.out.print("a");
System.out.print("b");
System.out.print("c");
```

Результат:
```
abc
```

2) форматированный вывод - метод `printf`

```
printf("строка форматирования" [,аргумент1,аргумент2, … ])
```

Строка форматирования имеет вид:

```
текст%[флаг][число позиций][.точность]символ форматирования
```

Флаги управляют форматом вывода данных (например, выравнивание числовых полей по левому краю, вывод знака для положительных чисел и т.д.).

Символы форматирования:

`d` – целое

Пример: `System.out.printf("x=%3d",x)`

Результат `x=  5`

`f` – вещественное число

Пример: `System.out.printf("x=%8.3f",x)`

Результат `x=   2,955`

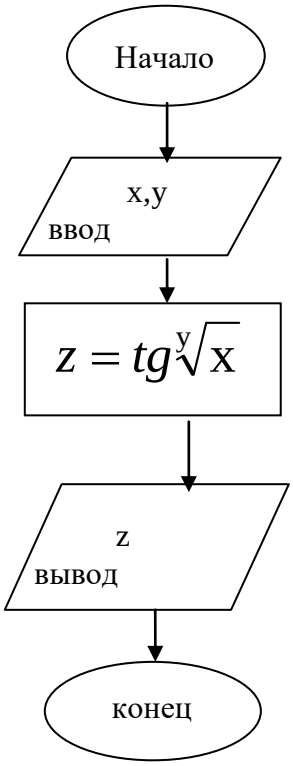
`s` – строка

Пример: `System.out.printf("x=%s",x)`

Результат `x=abc`

Пример программы, вычисляющей арифметическое выражение

Пример: $z = tg\sqrt[y]{x}$



```java
package lab1;

import java.util.Scanner;

public class Primer1 {

	public static void main(String[] args) {
		double x, y, z; // объявление переменных

		// ввод переменных
		Scanner inp = new Scanner(System.in);
		System.out.print("x=");   x = inp.nextDouble();
		System.out.print("y=");   y = inp.nextDouble();
		inp.close();

		// вычисление выражения
		z = Math.tan(Math.pow(x,1/y));

		// вывод результата
		System.out.printf("z=%1.4f",z);
	}

}
```

Составной оператор

```
{
	<операторы>;
}
```

Битовый сдвиг

1) значение << количество - сдвигает последовательность битов на заданное число позиций влево. При каждом сдвиге самый старший бит смещается за пределы допустимого значения и теряется, а освободившиеся справа биты заполняются нулями. Если операнд имеет тип long, биты теряются после сдвига за пределы 63 позиции. Если int - после сдвига за пределы 31 позиции.

20 << 3

Переводим 20 из десятичной системы в двоичную
$20_{10}=16+4=2^4+2^2=10100_2$

Сдвигаем получившееся число на три бита влево
$10100_{2<}\text{-}10100000_2$

Переводим число из двоичной системы в десятичную
$10100000_2=128+32=160_{10}$

2) >> - сдвигает последовательность битов на заданное число позиций вправо. При этом освободившиеся слева биты заполняются единицами при сдвиге отрицательных значений и нулями в случае положительных значений. Биты, которые выходят за разрядную сетку, теряются.

20 >> 3

Выполняется по аналогии с предыдущим примером

$20_{10}=16+4=2^4+2^2=10100_2->10_2=2_{10}$