Занятие № 2.

Основные элементы синтаксиса, типы данных, переменные

Компиляция, анализ и исправление ошибок
Основные элементы синтаксиса языка
Типы данных(примитивные)
Ключевые слова
Программа вывода форматированного текста
Объявления переменных и их инициализация

Ключевые слова

Зарезервированные ключевые слова - это специалъные идентификаторы, которые в языке Java исполъзуются для того, чтобы идентифицироватъ встроенные типы, модификаторы и средства управления выполнением программы. На сегодняшний день в языке Java имеется 59 зарезервированных слов. Эти ключевые слова совместно с синтаксисом операторов и разделителей входят в описание языка. Они могут применяться только по назначению, их нельзя исполъзовать в качестве идентификаторов для имен переменных, классов или методов.

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	

В Java существуют следующие базовые типы данных:

Тип данных	Объём занимаемой памяти	Диапазон допустимых значений	Значение по умолчанию (для полей класса)
byte	1 байт	Целые из [-128;127] что эквивалентно $[-2^7;2^7]$	0
short	2 байта	Целые из [-32768; 32767] или $[-2^{15};2^{15}]$	0
int	4 байта	Целые из [-2147483648; 2147483647] или $[-2^{31};2^{31}]$	0
long	8 байт	Целые из $[-2^{63};2^{63}]$	0L
float	4 байта	Вещественные положительные и отрицательные числа от $\sim1,4*10^{-45}$ до $\sim3,4*10^{38}$ по стандарту IEEE 754	0.0f
double	8 байт	Вещественные положительные и отрицательные числа от $\sim4,9*10^{-324}$ до $\sim1,8*10^{308}$ по стандарту IEEE 754	0.0d
char	2 байта	Натуральные из [0;65535], интерпретируются как коды символов по таблице Unicode	'\u0000'
boolean	Для хранения значения этого типа достаточно 1 бита, но в реальности память такими порциями не выделяется, поэтому переменные этого типа могут быть по-разному упакованы виртуальной машиной	false либо true	false

Методы printf и format

Пакет java.io содержит класс PrintStream, который содержит эти два метода, эти методы можно использовать вместо print и println. Объект System.out, который вы уже использовали — это объект PrintStream, поэтому вы можете вызывать методы PrintStream используя System.out. Например:

```
System.out.format(.....);
```

Синтаксис методов одинаков:

```
public PrintStream format(String format, Object... args)
```

где format — это строка которая определяет шаблон, согласно которому будет происходить форматирование, args — это список переменных, для печати по заданному шаблону. Простой пример:

```
System.out.format("The value of " + "the float variable is " +
     "%f, while the value of the " + "integer variable is %d, " +
     "and the string is %s", floatVar, intVar, stringVar);
```

Строка format содержит обычный текст и специальные форматирующие символы. Эти символы начинаются со знака процента (%) и заканчиваются конвертером — символом, который определяет тип переменной для форматирования. Пример:

```
int i = 461012;
System.out.format("The value of i is: %d%n", i);
```

Спецификатор %d определяет одну десятичную целую переменную. %n — переход на новую линию. Данный пример выведет:

```
The value of i is: 461012
```

Пример

Следующая таблица содержит некоторые флаги, которые используются в программе ниже.

Конвертер	Флаг	Описание
d		Десятичное целое.
f		Float.
n		Символ новой строки в зависимости от платформы, на которой запущена программа. Вместо \n лучше использовать %n.
tB		Дата и время — полное название месяца в зависимости от языка.
td, te		Дата и время — 2 цифры дня месяца. td — с ведущими нулями, te — без.
ty, tY		Дата и время — ty = год из 2х цифр, tY = год из 4х цифр.
tl		Дата и время — часы в 12 часовом формате.
tM		Дата и время — минуты из 2х цифр с ведущими нулями.
tp		Дата и время — am/pm в зависимости от языка(в нижнем регистре).
tm		Дата и время — месяц — 2 цифры с ведущими нулями.
tD		Дата и время — дата в формате %tm%td%ty
	08	Восемь символов с ведущими нулями при необходимости.
	+	Включить знак (положительный или отрицательный).
	—	По левому краю
	.3	Три символа после запятой
	10.3	Десять символов до запятой и три — после.

Рассмотрим пример, в котором рассчитывается и выводится на консоль таблица умножения:

```
int[][] multiplyTab  = new int[10][10];

for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        multiplyTab[i][j] = (i+1)*(j+1);
        //вывод ряда чисел разделенных знаком табуляции
        System.out.print(multiplyTab[i][j] + "\t");
   }
   System.out.println();
}
```

В данном случае для разделения чисел мы использовали знак табуляции, это дало следующий результат:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Таблица выглядит ровно, но она слишком широкая. Для того, чтобы сделать таблицу более компактной, будем использовать метод printf(). Заменим в предыдущем коде строку

```
System.out.print(multiplyTab[i][j] + "\t");
```

на строку

```
System.out.printf("%4d", multiplyTab[i][j]);
```

Получим следующий результат:

3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Как мы видим, таблица стала компактнее. Более того, теперь мы можем уменьшать или увеличивать промежутки между числами по нашему желанию. Для этого нужно лишь заменить число 4 в выражении «%4d».

Использование printf для форматирования в Java

Метод ***printf()*** принадлежит классу **PrintStream**, который отвечает за вывод информации. Уже знакомые нам методы ***print()*** и ***println()*** также являются методами класса **PrintStream.**

Метод **printf** определен следующим образом:

```
printf(String format, Object... args)
```

Первый аргумент **format** это строка, определяющая шаблон, согласно которому будет происходить форматирование. Для ее записи существуют определенные правила.

В предыдущем примере формат был **«%4d»,** где **d** означает вывод десятичного целого числа, а **4** — означает то, что если количество знаков в числе меньше, чем 4, то оно будет спереди дополнено пробелами на недостающее (до 4-х) количество знаков (тем самым подвинуто вправо).

Для наглядности приведем еще один пример, который выводит столбиком несколько чисел.

System.out.printf("%6d%n%6d%n%6d%n%6d%n%6d%n%6d", 666666, 55555, 4444, 333, 22, 1);

На консоль будет выведено:

```
666666
 55555
  4444
   333
    22
     1
```

Здесь в строке форматирования мы использовали несколько раз %6d%n, где каждое %6d задает формат для одного из чисел, указанных далее в качестве аргументов. Первое %6d форматирует число 666666, второе %6d форматирует 55555 и т.д. %n означает перевод строки. Поскольку ко всем числам было применено форматирование %6d, то числа, которые содержат менее 6 знаков подвинуты вправо на недостающее количество знаков и тем самым красиво выровнены.

Данный пример иллюстрирует также то, что метод printf может содержать несколько аргументов после строки с форматом. На что указывает Object… args в сигнатуре метода. Эти аргументы должны соответствовать ссылкам на них в строке формата. Например, если в строке формата стоит символ d, отвечающий за вывод целого десятичного числа, а далее в аргументе вы укажете строку, при компиляции произойдет ошибка преобразования формата java.util.IllegalFormatConversionException. Если аргументов больше, чем определено в формате, то лишние аргументы будут игнорироваться.

Объявления переменных и их инициализация

```
public class Simple {
    private int i;
    private boolean b;
    private String s;

    Simple() {
        int y;

        if(i  > 0) y = 5;
        System.out.println("i = " + i + " b = " + b
                + " s = " + s + " y = " + y) ;
    }
}
```

Инициализация переменных примитивного типа или объекта в полях происходит автоматически, им присваивается значение по-умолчанию. Например, переменная i будет равна нулю, логическая переменная b получит значение по умолчанию false, ссылочная переменная s получит null.
Однако, важно запомнить, что локальную переменную (в теле метода или конструктора) необходимо обязательно инициализировать.
По условиям задачи, переменная i будет равна нулю, но локальная переменная y не будет инициализирована, т.к. условие if(i > 0) не будет выполнено. Следовательно, это вызовет это ошибку компилятора.
Запомните, что локальные переменные в Java должны быть инициализированы всегда.
Если подправить код:

```
if (i  > 0) y = 5;
else y = 0;
```

то инициализация переменной будет гарантирована