

# Лексика языка Java

- Лекция посвящена описанию лексики языка Java. Лексика описывает, из чего состоит текст программы, каким образом он записывается и на какие простейшие слова (лексемы) компилятор разбивает программу при анализе.
- Лексемы (или `tokens` в английском варианте) – это основные "кирпичики", из которых строится любая программа на языке Java.
- Эта тема раскрывает многие детали внутреннего устройства языка, и невозможно написать ни одной строки кода, не затронув ее.

# Темы лекции 2. Лексика языка

Кодировка

Анализ программы

- Утверждения и выражения Java.
- Переменные и примитивные типы данных.
- Константы. Комментарии. Литералы.
- Арифметика.
- Сравнение.
- Логические операторы.

Математика

# Кодировка в программах Java

- Для записи лексем используется код ASCII – American Standard Code for Information Interchange, Американский стандартный код обмена информацией 128
- Для комментариев, идентификаторов, символьных и строковых литералов используется Unicode.  
От `\u0000` до `\uFFFF`
- Какая версия Unicode используется в настоящее время можно узнать на сайте <http://www.unicode.org/>

# Анализ программы

- пробелы (white spaces);
- комментарии (comments);
- основные лексемы (tokens).

# Пробелы в Java

- **ASCII-символ SP**, space, пробел, `\u0020`, десятичный код 32;
- **ASCII-символ HT**, horizontal tab, символ горизонтальной табуляции, `\u0009`, десятичный код 9;
- **ASCII-символ FF**, form feed, символ перевода страницы (был введен для работы с принтером), `\u000c`, десятичный код 12;
- **завершение строки.**

# Комментарии в Java

Комментарии не влияют на результирующий бинарный код и используются только для ввода пояснений к программе.

В Java комментарии бывают двух видов:

- Строчные - `//`
- Блочные - `/* */` и комментарий разработчика `/** */` для документирования, ставятся перед объявлением класса интерфейсов, полей, методов и конструкторов

# Комментарии в Java

```
// В этом примере текст /*...*/ станет просто  
// частью строки s  
String s = "text/*just text*/";
```

```
/* Следующая строка станет причиной ошибки  
   при компиляции, так как комментарий  
   разбил имя метода getRadius()  
*/ circle.get/*comment*/Radius();
```

```
// Комментарий может разделять вызовы функций  
circle./*comment*/getRadius();
```

# Комментарии в Java

В следующем примере компилятор выдаст ошибку:

```
/*  
    comment  
    /*  
    more comments  
    */  
    finish  
*/
```



# Основные лексемы в Java

- идентификаторы (identifiers);
- ключевые слова (key words);
- литералы (literals);
- разделители (separators);
- операторы (operators).

# Идентификаторы в Java

**Идентификаторы** – это имена, которые даются различным элементам языка для упрощения доступа к ним. Имена имеют пакеты, классы, интерфейсы, поля, методы, аргументы и локальные переменные.

**Идентификаторы** можно записывать символами Unicode, то есть на любом удобном языке. Длина имени не ограничена.

# Идентификаторы в Java

Идентификатор состоит из букв и цифр.

В нотации Бэкуса-Науэра это правило может быть записано так:

$$\langle \text{идентификатор} \rangle ::= \langle \text{буква} \rangle \\ \{ \langle \text{буква} \rangle \mid \langle \text{цифра} \rangle \} [\dots n]$$

# Зарезервированные слова и методы языка Java

- Зарезервированные слова - это специальные идентификаторы, которые в языке Java используются для того, чтобы идентифицировать **встроенные** типы, модификаторы и средства управления выполнением программы.
- На сегодняшний день в языке Java имеется 50 зарезервированных слов

# Зарезервированные слова языка Java

<code>abstract</code>	<code>continue</code>	<code>for</code>	<code>new</code>	<code>switch</code>
<code>assert<sup>***</sup></code>	<code>default</code>	<code>goto<sup>*</sup></code>	<code>package</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>case</code>	<code>enum<sup>****</sup></code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>catch</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>strictfp<sup>**</sup></code>	<code>volatile</code>
<code>const<sup>*</sup></code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>

# Литералы (константы) в Java

**Литералы** позволяют задать в программе значения для числовых, символьных и строковых выражений, а также null-литералов. Всего в Java определено 6 видов литералов:

- **целочисленный** (`integer`) ;
- **дробный** (`floating-point`) ;
- **булевский** (`boolean`) ;
- **символьный** (`character`) ;
- **строковый** (`string`) ;
- **null-литерал** (`null-literal`) .

# Целочисленные литералы в Java

Целочисленные литералы позволяют задавать целочисленные значения в виде

- **десятеричном** – традиционная запись
- **восьмеричном** – запись начинается с нуля, запрещены 8 и 9
- **шестнадцатеричном** – запись начинается с 0x или 0X, кроме цифр от 0 до 9 используются A, B, C, D, E, F

# Целочисленные литералы в Java

Для записи очень больших целочисленных чисел (тип **long**) следует в конце числа писать L или l, в противном случае компилятор выдаст сообщение об ошибке.

**Максимальные допустимые значения:**

```
9223372036854775807L
```

```
07777777777777777777777777777777L
```

```
0x7fffffffffffffffffffffffL
```

**// наибольшие отрицательные значения:**

```
-9223372036854775808L
```

```
-01000000000000000000000000000000L
```

```
-0x800000000000000000000000L
```



# Дробные литералы в Java

Дробные литералы представляют собой числа с плавающей десятичной точкой. Правила записи таких чисел такие же, как и в большинстве современных языков программирования.

## Примеры:

3.14

2.

.5

7e10

3.1E-20

# Дробные литералы в Java

дробный литерал состоит из следующих составных частей:

- целая часть;
- десятичная точка (используется ASCII-символ точка);
- дробная часть;
- порядок (состоит из латинской ASCII-буквы E в произвольном регистре и целого числа с опциональным знаком + или -);
- окончание-указатель типа.

# Дробные литералы в Java

Целая и дробная части записываются десятичными цифрами, а **указатель** типа (аналог указателя L или l для целочисленных литералов типа long) имеет два возможных значения –

- латинская ASCII-буква D (для типа double)
- или F (для типа float) в произвольном регистре.

Если указатель типа опущен, считается что литерал имеет тип double.

# Дробные литералы в Java

В спецификация IEEE 754 (полное название – IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985 (IEEE, New York)) – описаны не только конечные дробные величины, но и еще несколько особых значений, а именно:

- положительная и отрицательная **бесконечности** (positive/negative infinity);
- значение "**не число**", Not-a-Number, сокращенно NaN;
- положительный и отрицательный **нули**.

# Предельные значения

Максимальное положительное конечное значение дробного литерала:

- для float:  $3.40282347e+38f$
- для double:  $1.79769313486231570e+308$

Минимальное положительное ненулевое значение:

- для float:  $1.40239846e-45f$
- для double:  $4.94065645841246544e-324$

# Булевские литералы в Java

Логические литералы имеют два возможных значения – **true** и **false**.

Эти два зарезервированных слова не являются ключевыми, но также не могут использоваться в качестве идентификатора.

# Символьные литералы в Java

Символьные литералы описывают один символ из набора Unicode, заключенный в **одиночные кавычки**, или апострофы (ASCII-символ single quote, \u0027). Например:

'a' // латинская буква а

' ' // пробел

'К' // греческая буква каппа

Допускается также запись:

'\u0041' // латинская буква А

'\u0410' // русская буква А

'\u0391' // греческая буква А

# Символьные литералы в Java

- Символьный литерал должен содержать строго один символ, или специальную последовательность, начинающуюся с `\`. Для записи специальных символов (неотображаемых и служебных, таких как `"`, `'`, `\`) используются следующие обозначения:

```
\b  \u0008  backspace BS - забой
\t  \u0009  horizontal tab HT - табуляция
\n  \u000a  linefeed LF - конец строки
\f  \u000c  form feed FF - конец страницы
\r  \u000d  carriage return CR - возврат каретки
\"  \u0022  double quote " - двойная кавычка
\'  \u0027  single quote ' - одинарная кавычка
\\  \u005c  backslash
\ - обратная косая черта
\шестнадцатеричный код от \u0000 до \u00ff символа
    в шестнадцатеричном формате.
```



# Строковые литералы в Java

- Строковые литералы состоят из набора символов и записываются в **двойных кавычках**. Длина может быть нулевой или сколь угодно большой. Любой символ может быть представлен специальной последовательностью, начинающейся с \.

`" "` // литерал нулевой длины

`"\"` // литерал, состоящий из одного символа `"`

`"Простой текст"` // литерал длины 13

# null-литерал в Java

- Null-литерал может принимать всего одно значение: `null`. Это литерал ссылочного типа, причем эта ссылка никуда не ссылается, объект отсутствует.
- Разумеется, его можно применять к ссылкам любого объектного типа данных.
- Типы данных рассматриваются далее в лекции.

# Разделители в Java

- **Разделители** – это специальные символы, которые используются в служебных целях языка. Назначение каждого из них будет рассмотрено по ходу изложения курса. Вот их полный список:

( ) [ ] { } ; . ,

# Операторы в языке Java

- Операторы используются в различных операциях – арифметических, логических, битовых, операциях сравнения и присваивания. Следующие 37 лексем (все состоят только из ASCII-символов) являются операторами языка Java:

=	>	<	!	~	?	:==	<=	>=
!=	&&		++	--	+	-	*	/
%	^	%	<<	>>	>>>	+=	--	*=
/=	&=	=	^=	%=	<<=	>>=	>>>=	

# Работа с операторами

- Операторы присваивания и сравнения
- Арифметические операции
- Логические операции

# Присвоение и сравнение

оператор присваивания = и

оператор сравнения == различаются .

```
x = 1; // присваиваем переменной x значение 1
```

```
x == 1 // сравниваем значение переменной x с единицей
```

Оператор сравнения всегда возвращает  
булевское значение `true` или `false`.

Оператор присваивания возвращает значение  
правого операнда.

# Арифметика

Наряду с четырьмя обычными арифметическими операциями  $+$ ,  $-$ ,  $*$ ,  $/$ , существует оператор получения остатка от деления  $\%$ , который может быть применен как к целочисленным аргументам, так и к дробным.

Унарные операторы **инкрементации**  $++$  и **декрементации**  $--$ , как обычно, можно использовать как справа, так и слева.

Битовые операции

# Логические операторы

Логические операторы "и" и "или" (& и |) можно использовать в двух вариантах.

Первый вариант операторов (&, |) всегда вычисляет оба операнда,

Второй – ( &&, || ) не будет продолжать вычисления, если значение выражения уже очевидно.



# Математика - пакет java.Math

- `Math.pow` – возведение в степень
- `Math.sqrt` – квадратный корень
- `Math.tan` – тангенс
- `Math.sin` – синус
- `Math.cos` – косинус
- `Math.atan` – арктангенс
- `Math.exp` – экспонента
- `Math.log` – логарифм
- `Math.PI` – число пи
- `Math.E` – число e

# Типы данных

- Целочисленные
  - byte
  - short
  - int
  - long
  - char (также является целочисленным типом)
- Дробные
  - float
  - double
- Булевские