LEXICAL STRUCTURE

Кодировка

- Кодировка это соответствие между символами и числами.
- Каждый символ кодировки имеет **фиксированный уникальный** числовой код.
- Кодировку можно представить в виде таблицы.

Кодировка ASCII

- ASCII включает в себя управляющие символы, знаки препинания, десятичные цифры, латинский алфавит.
- Коды символов ASCII лежат в диапазоне от 0 до 127 включительно.
- Практически все распространенные кодировки включают в себя ASCII составной частью.

Управляющие символы ASCII

- Возврат каретки
  - символ с кодом 0x0D (13 в десятичной системе счисления), '\r', CR.
- Перевод строки
  - символ с кодом 0x0A (10 в десятичной системе счисления), '\n', LF.

Unicode

Стандарт кодирования символов.

Стандарт состоит из двух частей:

- кодировка Unicode;
- формат преобразования Unicode (UTF - Unicode transformation format).

Кодовая точка Unicode

Каждый символ Unicode имеет фиксированный числовой код, т.н. кодовую точку (code point), в виде неотрицательного целого числа.

Нотация обозначения символов Unicode

Кодовая точка ==> Обозначение

0 - FFFF ==> U+xxxx
10000 - FFFFF ==> U+xxxxx
100000 - 10FFFF ==> U+xxxxxx

Диапазоны символов Unicode

- [U+0000, U+007F]
  - Совпадает с ASCII
- [U+0000, U+FFFF]
  - BMP - базовая мультиязыковая плоскость
- [U+10000, U+10FFFF]
  - Дополнительные символы (supplimentary characters).

Количество символов Unicode

Зависит от версии стандарта Unicode.

Текущая версия 6.2, стандарт 2012.

Максимальная кодовая точка кодировки Unicode: 10FFFF

Количество символов Unicode меньше этого значения, т.к. *некоторым кодам символы в соответствие не поставлены*.

UTF

- Формат преобразования Unicode.
- Взаимооднозначное соответствие между кодовыми точками символов Unicode и последовательностью байт.
- UTF определяет, *как кодовые точки будут представлены байтами*.

Виды UTF

- UTF-8
- UTF-16 (BE/LE варианты)
- UTF-32 (BE/LE варианты).

Количество байт на символ в разных UTF

- UTF-8
  - от 1 до 6 байт на символ
  - для записи ASCII использует один байт
- UTF-16
  - 2 б. для символов [U+0000, U+FFFF]
  - 4 б. для символов [U+10000, U+10FFFF]
- UTF-32
  - используется ровно четыре байта.

Метка порядка байт (BOM)

- UTF-16
  - BE ==> FEFF
  - LE ==> FFFE
- UTF-32
  - BE ==> 0000 FEFF
  - LE ==> FFFE 0000
- UTF-8
  - EF BB BF

Порядок байт BE

Прямой порядок байт
(он же big endian - BE).

Старший (более значимый) байт в слове находится впереди младшего (менее значимого) байта.

Запись BOM в UTF-16BE: FEFF
Запись BOM в UTF-32BE: 0000 FEFF

Порядок байт LE

Обратный порядок байт
(little endian - BE).

Младший (менее значимый) байт в слове расположен впереди старшего (более значимого) байта.

Запись BOM в UTF-16LE: FFFE
Запись BOM в UTF-32LE: FFFE 0000

Использование метки порядков байт (BOM)

- Стандарт Unicode определяет использование метки порядков байт как *опциональное*.
- В том случае, когда метка отсутствует, порядок байт *по умолчанию будет принят BE*.

Диапазоны суррогатных заменителей UTF-16

Диапазоны суррогатный заменителей:

- [U+D800, U+DBFF] - верхний;
- [U+DC00, U+DFFF] - нижний.

Каждый символ из [10000, 10FFFF] будет представлен парой символов из этих диапазонов:

- первый из верхнего
- второй из нижнего

Представление дополнительных символов

Каждый дополнительный символ Unicode ([U+10000, U+10FFFF]) кодируют двумя суррогатными символами.

Таким образом, доп. символы представлены четырьмя байтами

- первые два из диапазона [D800, DBFF]
- вторые два из диапазона [DC00, DFFF]

Unicode escape последовательности Java

\uXXXX

где XXXX - шестнадцатеричный код символа в кодировке UTF-16BE.

Регистры цифр не имеют значения.
Буква u **В НИЖНЕМ РЕГИСТРЕ!**

Escape последовательности для дополнительных символов

Для записи дополнительных символов Unicode с помощью Unicode escape последовательностей Java используют две подряд идущие escape последовательности, в которых записаны коды *соответствующих суррогатных заменителей*:

U+1D120 ===> \uD834\uDD20

Кодировка исходного текста программы

По умолчанию компилятор интерпретирует входные символы используя т.н. кодировку по умолчанию операционной системы в которой он запущен.

При этом будет осуществлено преобразование (перекодирование): КПУ ===> UTF-16BE.

Кодировка по умолчанию в Windows

Windows русской локализации

Cp1251, она же Windows-1251, *однобайтная кодировка с кириллицей*.

Для консоли:

Cp866 (неофициальное название - DOS кодировка), *однобайтная кодировка с кириллицей*.

Кодировки KOI8, Cp1251, Cp866

- KOI8 - однобайтная кодировка, содержит кириллицу. Есть подвиды: KOI8-R (=Cp20866, рус. алф.), KOI8-U (=Cp21866, укр. алф.).
- Windows-1251 (=Cp1251) - однобайтная кодировка, содержит кириллицу, КПУ во всех Windows рус. локализации.
- Cp866 - однобайтная кодировка, содержит кириллицу, КПУ консоли Windows рус. локализации.

Кодировка ISO-8859-1

Она же Latin-1, CP819.

Однобайтная кодировка, совпадает с первыми 256 символами Unicode.

По умолчанию кодировка java properties файлов.

Лексическая трансляция кода программы

1) Подстановка: \uXXXX ===> символ Unicode с кодовой точкой XXXX;

2) определение входных Unicode символов и ограничителей строк;

3) определение входных элементов (пробельные символы, комментарии, лексемы).

Ограничители строк

- Символ U+000A, он же ASCII символ LF (перевод строки)
- Символ U+000D, он же ASCII символ CR (возврат каретки)
- Последовательность (*упорядоченная*) из двух последовательно идущих символов U+000D и U+000A.

Входные элементы языка Java

- Пробельные символы
- Комментарии
- Лексемы

Лексемы отделены друг от друга пробельными символами или комментариями.

Разделители лексем

- Пробельные символы
- Комментарии

```
int/*коммент. разделяет лексемы*/x;
```

Пробельные символы

Служат для разделения лексем.

- Пробел (SP)
- Горизонтальная табуляция (HT)
- Перевод страницы (FF)
- Ограничители строк (\u000A, \u000D, \u000D\u000A).

Комментарии в Java

Обычно выделяют три вида:

1) однострочный: // текст
2) многострочный: /* текст */
3) документатора: /** документация */

По последней спецификации комментарий документатора это многострочный комментарий.

Лексемы языка Java

- Идентификаторы (Unicode)
- Литералы (Unicode)
- Ключевые слова (ASCII)
- Разделители (ASCII)
- Знаки операций (ASCII)

Идентификаторы

Идентификаторы используют для именования:

- типов (классы, интерфейсы)
- пакетов
- методов
- полей
- локальных переменных

Структура идентификаторов

Последовательность неограниченной длины букв и цифр языка Java.

На первом месте в последовательности должна быть буква.

Идентификатор не может иметь то же самое написание что и

- ключевые слова
- литералы true, false, null

Буква в языке Java

Символ, для которого метод Character.isJavaIdentifierStart возвращает значение true.

Примеры:

- латинские буквы
- символ подчеркивания _
- символ доллара $.

Буква или цифра Java

Символ, для которого метод Character.isJavaIdentifierPart возвращает значение true.

Примеры:

- Латинские буквы
- Кирилические буквы
- Цифры от 0 до 9 (коды: U+0030 - U+0039)
- $, _

Ключевые слова

50 ключевых слов (JSE 7)

Примитивные типы данных

- Целые числа: byte short int long char
- Вещественные числа: float double
- Логический тип: boolean

Модификаторы уровня доступа

- public
- protected
- private

Используемые в операторах выбора

- if else
- switch case default

Используемые в циклах

- for
- while
- do

Используемые при работе с исключениями

- throw
- throws
- try catch finally

Неиспользуемые

- goto
- const

Использование данных ключевых слов вызовет ошибку на этапе компиляции.

Литералы

Литералы - это представления в исходном коде программы значений:

- примитивных типов
  - int long float double boolean
- типа String
- null - литерал нул типа

Замечание: экземпляры Class<Type> также называют литералами типа Type.

Числовые литералы

Числовые литералы – константы типов:

- int long (*целые*)
- float double (*вещественные*)

В записи литералов допустимо использовать знак подчеркивания для разделения разрядов

- *только между цифрами*
- *любое число знаков _*

10_000 0_7777 1_____2_3E1_2

Числовой литерал со знаком

Если числовой литерал предваряет знак + или - то знак "+/-" не входит в состав литерала: -34 +3

Верно для любых числовых литералов (целых и вещественных).

Целые литералы

Целые литералы могут быть записаны с помощью одной из четырех систем счисления:

- десятичной
- шестнадцатеричной
- восьмеричной
- бинарной

Тип целого литерала

Если в конце целого литерала стоит суффикс L или l, то тип литерала long.

Если суффикс отсутствует, тип литерала int.

Представление отрицательных чисел с помощью литералов

Отрицательные числа могут быть представлены *только* с помощью бинарных, восьмеричных или шестнадцатеричных литералов. Три следующих литерала представляют -1

0b11111111_11111111_11111111_11111111
037_777_777_777
0xFF_FF_FF_FF

Десятичные литералы *не могут представлять отрицательные числа* (*только положительные или ноль*).

Максимальные десятичные целые литералы

int: 2^31 = -2147483648
long: 2^63 = -9223372036854775808L
могут быть использованы *только с унарной операцией изменения знака*: -

Без минуса:
int: 2^31 - 1 = 2147483647
long: 2^63 - 1 = 9223372036854775807L

Диапазоны десятичных целых литералов

- int ===> [0, 2^31]
- long ===> [0, 2^63]

Структура шестнадцатеричных целых литералов

- Обязателен признак 0x или 0X.
- Минимум одна шестнадцатиричная цифра (0 - 9, a - f, A - F).
- Опциональный суффикс L/l.

Примеры: 0xABL; 0X0; 0x123L; 0X123

Структура десятичных целых литералов

- Минимум одна десятичная цифра.
- Опциональный суффикс L/l.
- Если цифр больше чем одна, то *первая не может быть нулем*.

Примеры: 0; 123; 0L; 0l; 72L; 5
Но: 00; 0034 - *целые восьмеричные литералы!*

Структура восьмеричных целых литералов

- Обязателен признак восмеричного литерала 0.
- Минимум одна восьмеричная цифра (0-7).
- Опциональный суффикс L/l.

Примеры: 00; 00000L; 017; 0777L; 0123

Структура бинарных целых литералов

- Обязателен признак 0b или 0B.
- Минимум одна цифра из множества {0, 1}
- Опциональный суффикс L/l.

Примеры: 0b101L; 0B00000

Вещественные литералы

Вещественные литералы могут быть записаны с помощью систем счисления:

- десятичной
- шестнадцатеричной

Тип вещественных литералов

По умолчанию double или если в конце литерала поставлен суффикс D (или d).

Суффикс F (или f) указывает, что литерал имеет тип float.

Структура десятичных вещественных литералов

Общий вид (порядок важен):
[цифры] [точка] [цифры] [десятичная_экспонента] [суффикс] (1.2E-3D)

Четыре варианта структуры (остальные компоненты опциональны):
1) [цифры] [точка] (12.; 1.2; 1.e+2; 1.2f)
2) [точка] [цифры] (.12; .1; .1E2; 1.23)
3) [цифры] [десятичная_экспонента] (12E3; 1e-2d; 1.2E+3)
4) [цифры] [суффикс] (1f; 12D; .1D, 1E2D)

Цифры - десятичные, суффиксы D/d, F/f.

Структура десятичной экспоненты

- Обязателен признак десятичной экспоненты E или e.
- Необязательный знак экспоненты + или -.
- Минимум одна *десятичная* цифра.

Примеры: E1; e+1; E-123
123E-45 ==> 123 * 10^-45

Структура шестнадцатеричного вещественного литерала

Общий вид (порядок важен):
[0X или 0x] [цифры] [точка] [цифры] [бинарная экспонента] [суффикс] (0X12.34P-5D)

[бинарная экспонента] обязательна; [суффикс] опционален; [цифры] [точка] [цифры] не обязательны, но если стоит точка, то должна быть хотя бы одна цифра.

Цифры - шестнадцатеричные (0-9, A-F, a-f);

суффиксы d/D, f/F.

Примеры: 0x24P1; 0x1.2p-3F
Замечание: Если цифр перед экспонентой нет, то число равно нулю (0xP+37 = 0).

Структура бинарной экспоненты

- Обязателен признак бинарной экспоненты P или p.
- Необязательный знак экспоненты + или -.
- Минимум одна *десятичная* цифра.

Примеры: p1 p+1 P-99

23.4P2 ==> (2*16^1 + 3*16^0 + 4*16^-1) * 2^2

Булевы литералы и литерал нул-типа

- boolean
  - true
  - false
- Нул-тип
  - null

Символьные литералы

Символ Unicode из диапазона [U+0000, U+FFFF] заключенный в одинарные кавычки ' (U+0027), за исключением:

- одинарной кавычки ' (U+0027)
- обратного слеша \ (U+005C)
- \u000A
- \u000D

Символьные литералы имеют тип char.

Примеры: 'a'; 'T'; '\u0065'; '\77'; '\\'

Строковые литералы

Ноль или более символов Unicode (*допустимы символы из всего диапазона Unicode*), заключенные в двойные кавычки " (U+0022), за исключением:

- двойной кавычки " (U+0022)
- обратного слеша \ (U+005C)
- \u000A
- \u000D

Строковые литералы имеют тип String.

Примеры: ""; "ab\"c"; "\u0065bc\123"

Способы представления символов в литералах

В символьных и строковых литералах символ может быть представлен в виде:

- знака символа
- Java Unicode escape последовательности
  - \uXXXX (кроме \u000D \u000A !)
  - двумя для доп. символов (*только в строковых летералах!*)
- восьмеричной escape последовательностью
  - \X \XX \XXX (*только символы ISO-8859-1!*)
- символьной escape последовательностью
  - \\ \r \n \’ \” \t \b \f (*только эти 8 символов*)

Конкатенация строковых литералов

Длинный строковый литерал может быть записан при помощи оператора конкатенации строк +, результат конкатенации - строковый литерал.

Конкатенация двух строковых литералов - выражение, а не строковый литерал, однако, результат такого выражения будет вычислен *на этапе компиляции*.

Символьные escape последовательности

\t U+0009, горизонтальная табуляция

\n U+000A, перевод строки

\r U+000D, возврат каретки

\f U+000C, перевод страницы

\' U+0027, одинарная кавычка

\" U+0022, двойная кавычка

\\ U+005C, обратный слеш

\b U+0008, забой (backspace)

Восьмеричные escape последовательности

\A символ с кодом 0A
\AB символ с кодом 0AB
\ZAB символ с кодом 0ZAB

Z - цифра из множества [0, 3]; A, B – восьмеричные цифры [0, 7].

Примеры: \7 \20 \377
(\377 ===> 255=FF)

Разделители

Всего существует 9 символов разделителей:

[] () {}

квадратные, круглые, фигурные скобки

. , ;

точка, запятая, точка с запятой

Операции

Всего существует 36 операций:

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

=   +=     -= *= /= %=
&=    |= ^= <<=  >>=  >>>=
```