The Java Language Specification

Lexical Structure of Literals

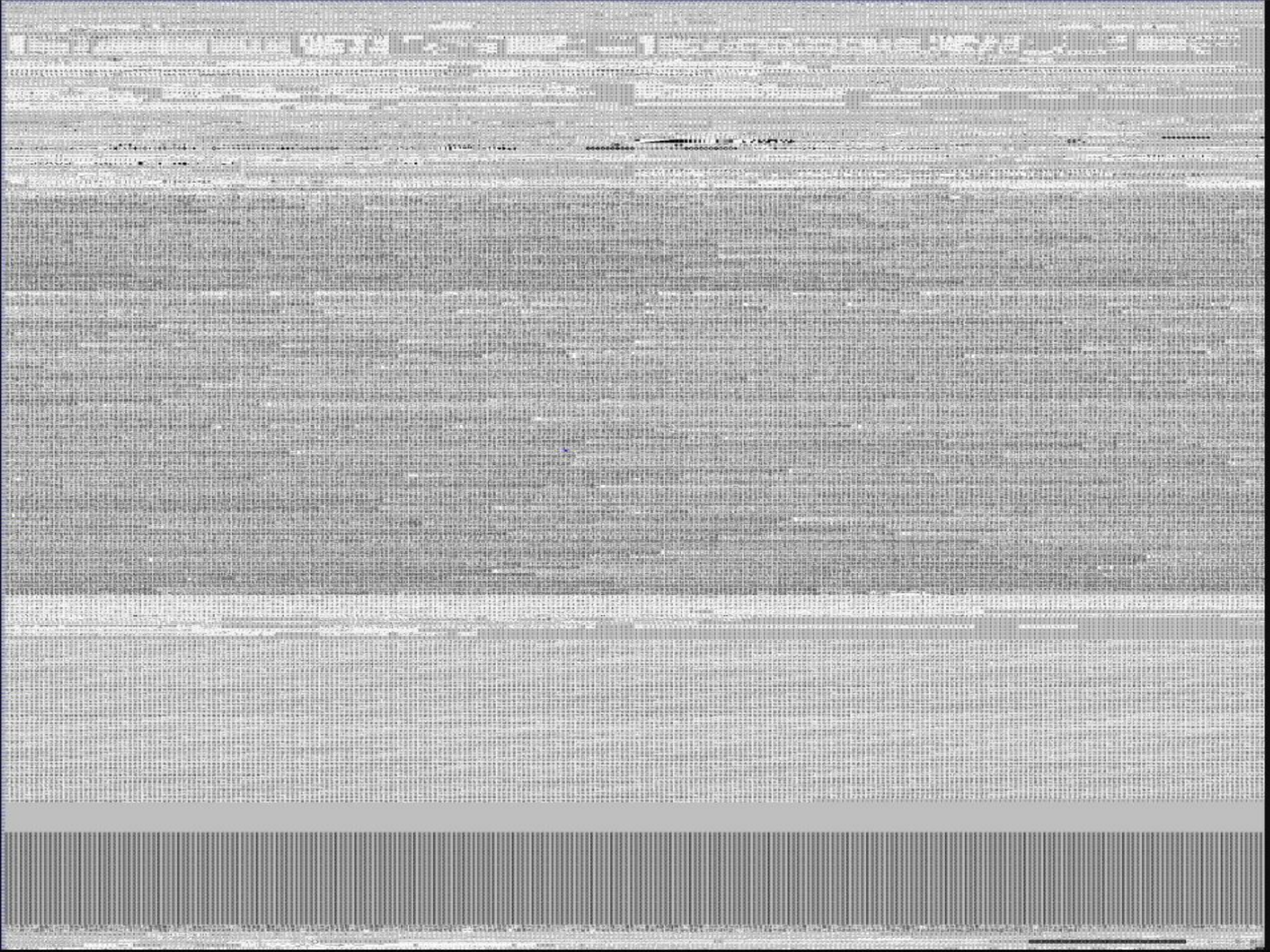
CFG

A grammar (when the context is not given, often called a formal grammar for clarity) is a set of production rules for strings in a formal language

A context-free grammar (CFG) is a set of recursive rewriting rules (or productions) used to generate patterns of strings

Unicode

The Unicode Standard provides a unique number for every character(All 1,112,064 of them), no matter what platform, device, application or language



Lexical Translations

A raw Unicode character stream is translated into a sequence of tokens, using the following three lexical translation steps:

1. A translation of Unicode escapes in the raw stream of Unicode characters to the corresponding Unicode character
2. A translation of the Unicode stream resulting from step 1 into a stream of input characters and line terminators
3. A translation of the stream of input characters and line terminators resulting from step 2 into a sequence of input elements

Removing the white spaces and comments, we get tokens that are terminal symbols of the syntactic grammar.

White spaces

WhiteSpace:

the ASCII SP character, also known as "space"

the ASCII HT character, also known as "horizontal tab"

the ASCII FF character, also known as "form feed"

Comments

Comments can be of two types:

/* text */ (All enclosed characters are ignored)

// (All characters till end of line are ignored)

Comments have the following properties:

1. Comments do not nest
2. /* and */ have no special meaning in comments that begin with //
3. // has no special meaning in comments that begin with /* or /**

Identifiers and Keywords

An identifier is an unlimited-length sequence of Java letters and Java digits, the first of which must be a Java letter.

A java letter consists of ASCII A-Z, ASCII a-z and ASCII _ and $.

Java digits include ASCII 0-9

50 keywords are reserved and cannot be used as identifiers.

A few common ones are : if, for, new, return, private, public, while, do etc.

Literals

A literal is the source code representation of a value of a primitive type, the string type or the null type

The different types of literals in java are:

1. IntegerLiteral
2. FloatingPointLiteral
3. BooleanLiteral
4. CharacterLiteral
5. StringLiteral
6. NullLiteral

Integer Literals

IntegerLiteral:

- DecimalIntegerLiteral
- HexIntegerLiteral
- OctalIntegerLiteral
- BinaryIntegerLiteral

DecimalIntegerLiteral:
    DecimalNumeral [IntegerTypeSuffix]

HexIntegerLiteral:
    HexNumeral [IntegerTypeSuffix]

OctalIntegerLiteral:
    OctalNumeral [IntegerTypeSuffix]

BinaryIntegerLiteral:
    BinaryNumeral [IntegerTypeSuffix]

IntegerTypeSuffix:

l L

DecimalNumeral:

0

NonZeroDigit [Digits]

NonZeroDigit Underscores Digits

NonZeroDigit:

1 2 3 4 5 6 7 8 9

Digits:

Digit

Digit [DigitsAndUnderscores] Digit

Digit:

0

NonZeroDigit

—

```
DigitsAndUnderscores:
    DigitOrUnderscore{DigitOrUnderscore}

DigitOrUnderscore:
    Digit
    _

Underscores:
    _ {_}
```

HexNumeral:

0 x HexDigits

0 X HexDigits

HexDigits:

HexDigit

HexDigit[HexDigitsAndUnderscores] HexDigit

HexDigit:
(one of) 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

HexDigitsAndUnderscores:
HexDigitOrUnderscore{HexDigitOrUnderscore}

—

HexDigitOrUnderscore:
    HexDigit
    _

___

OctalNumeral:

- 0 OctalDigits
- 0 Underscores OctalDigits

OctalDigits:

- OctalDigit
- OctalDigit[OctalDigitsAndUnderscores]OctalDigit

OctalDigit:
(one of) 0 1 2 3 4 5 6 7

OctalDigitsAndUnderscores:
OctalDigitOrUnderscore{OctalDigitOrUnderscore}

OctalDigitOrUnderscore:
  OctalDigit
  _

___

BinaryNumeral:

0 b BinaryDigits

0 B BinaryDigits

BinaryDigits:

BinaryDigit

BinaryDigit[BinaryDigitsAndUnderscores]BinaryDigit

BinaryDigit:

(one of) 0 1

———

BinaryDigitsAndUnderscores:
    BinaryDigitOrUnderscore{BinaryDigitOrUnderscore}

BinaryDigitOrUnderscore:
BinaryDigit
_

Floating-Point Literals

FloatingPointLiteral:

DecimalFloatingPointLiteral

HexadecimmalFloatingPointLiteral

DecimalFloatingPointLiteral:

Digits . [Digits] [ExponentPart] [FloatTypeSuffix]

. Digits [ExponentPart] [FloatTypeSuffix]

Digits ExponentPart [FloatTypeSuffix]

Digits [ExponentPart] FloatTypeSuffix

ExponentPart:

ExponentIndicator SignedInteger

ExponentIndicator:

e E

SignedInteger
    [Sign] Digits

Sign:
    + -

FloatTypeSuffix:
f F d D

HexadecimalFloatingPointLiteral:
    HexSignificand BinaryExponent [FloatTypeSuffix]

HexSignificand:
    HexNumeral [.]
    0 x [HexDigits] . HexDigits
    0 X [HexDigits] . HexDigits

BinaryExponent:

BinaryExponentIndicator SignedInteger

BinaryExponentIndicator:

p P

Boolean Literals

BooleanLiteral:
(one of) true false

Character Literal

CharacterLiteral:

'SingleCharacter'

'EscapeSequence'

SingleCharacter:

InputCharacter but not ' or \

Null Literal

NullLiteral:

Null

Escape Sequence

EscapeSequence:

\b (backspace)

\t (horizontal tab)

\n (linefeed)

\f (formfeed)

\r (carriage return)

\" (double quote)

Separator

It helps to define the structure of a program.

( ) - encloses argument

{ } - define block of code

[ ] - declare array type

; - terminate statement

: - used after loop label

. - select a field or method

Operator

Operator:(one of)

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

String literals

A string literal consists of zero or more characters enclosed in double quotes.

Characters may be represented by escape sequences.

StringLiteral:
" {StringCharacter} "

StringCharacter:
InputCharacter but not " or \
EscapeSequence

The characters CR and LF are never an InputCharacter.

A long string literal can always be broken using the string concatenation operator +

Expressions

Expression names
Primary expressions
Unary operator expressions
Binary operator expressions
Ternary operator expressions
Lambda expressions

AssignmentExpression:

ConditionalExpression

Assignment

Assignment:

LeftHandSide Assignment Operator Expression

—

LeftHandSide:

ExpressionName

FieldAccess

ArrayAccess

AssignmentOperator:(one of)

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

|=

Thank You