Java Tokens

- Reserved Keywords
- Identifiers
- Literals
- Operators
- Separators

Keywords

Keywords are essential part of a language definition. They implement specific features of the language. Java language has reserved 50 words as keywords. These keywords, combined with operators and separators according to syntax, form definition of the Java language. Examples of keywords are: abstract, byte do extends, for, import etc.

Since keywords have specific meaning in Java, we cannot use them as names for variables, classes, methods. All keywords are written in lower-case letters.

Identifiers:

Identifiers are programmer defined tokens used for naming classes, methods, variables, objects, labels, packages and interfaces in a program. Java identifiers follow the following rules:

i. They can have alphabets, digits and the underscore and dollar sign characters
ii. They must not begin with a digit
iii. Uppercase and lowercase letters are distinct
iv. They can be of any length.

Naming conventions followed by java developer:

- Name of all public methods and instance variables start with a leading lowercase letter. Ex: average, sum.
- When more than one words are used in a name, the second and subsequent words are marked with a leading uppercase letters. Examples:
  dayTemperature
  firstDay0fMonth
  totalMarks
- All private and local variables use only lowercase letters combined with underscores. Examples:
  length

batch_strength

- All classes and interfaces start with a leading uppercase letter (and each subsequent word with a leading uppercase letter). Examples:
  Student
  HelloJava
  Vehicle
  MotorCycle

- Variables that represent constant values use all uppercase letters and underscores between words. Examples:
  TOTAL
  F_MAX
  PRINCIPAL_AMOUNT

Literals

Literals in Java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables. Java language specifies five major types of literals. They are:

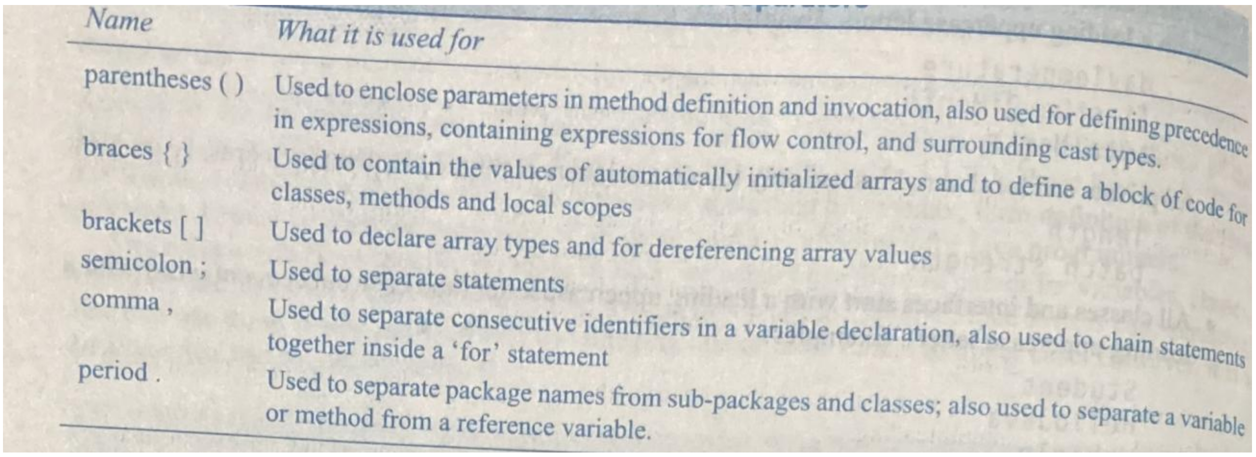
- Integer literals
- Floating_point literals
- Character literals
- String literals
- Boolean literals

Operators

An operator is a symbol that takes one or more arguments and operates on them to produce a result.

Separators

Separators are symbols used to indicate where groups of code are divided and arranged. They basically define the shape and function of our code. Below Table lists separators and their functions.



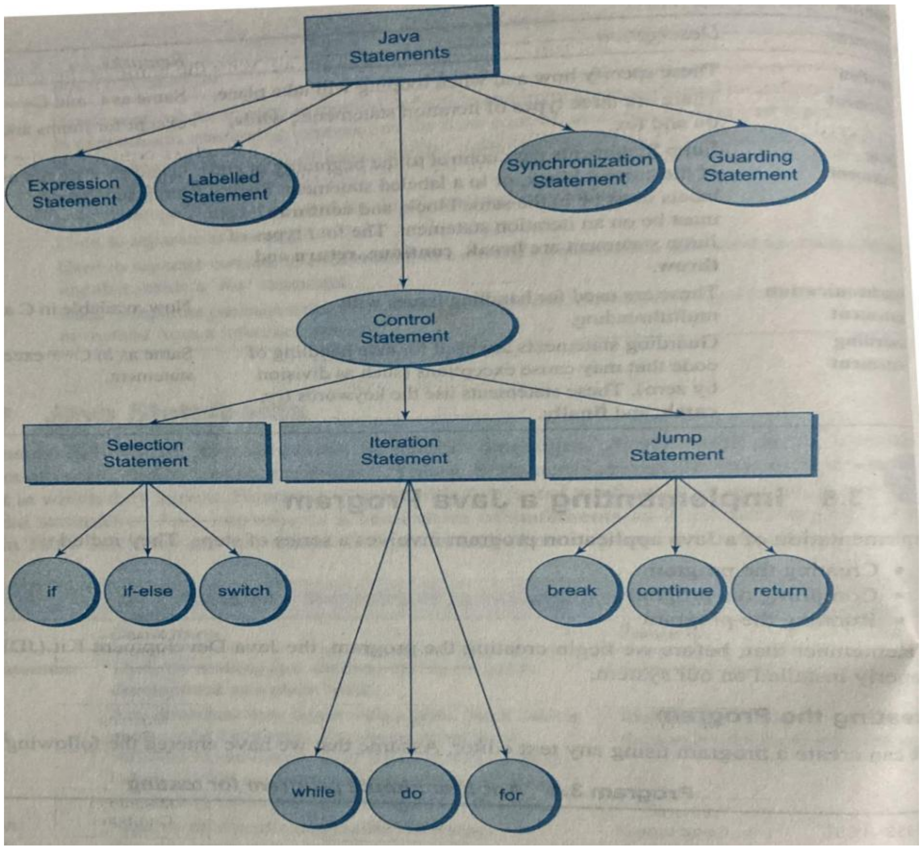
Name	What it is used for
parentheses ( )	Used to enclose parameters in method definition and invocation, also used for defining precedence in expressions, containing expressions for flow control, and surrounding cast types.
braces { }	Used to contain the values of automatically initialized arrays and to define a block of code for classes, methods and local scopes
brackets [ ]	Used to declare array types and for dereferencing array values
semicolon ;	Used to separate statements
comma ,	Used to separate consecutive identifiers in a variable declaration, also used to chain statements together inside a 'for' statement
period .	Used to separate package names from sub-packages and classes; also used to separate a variable or method from a reference variable.

Java statements

The statements in Java are like sentences in natural languages. A statement is an executable combination of tokens ending with a semicolon (;) mark. Statements are usually executed in sequence in the order in which they appear. However, it is possible to control the flow of execution, if necessary, using special statements. Java implements several types of statements as described in below table:

Statement	Description	Remarks
Empty statement	These do nothing and are used during program development as a place holder	Same as C and C++
Labeled statement	Any Statement may begin with a label. Such labels must not be keywords, already declared local variables or previously used labels in this module. Labels in Java are used as the arguments of Jump statements, which are described later in this list.	Identical to C and C++ except their use with jump statements
Expression statement	Most statements are expression statements. Java has seven types of Expression statements: **Assignment, Pre-Increment, Pre-Decrement, Post-Increment, Post-Decrement, Method Call and Allocation Expression.**	Same as C++
Selection statement	These select one of several control flows. There are Three types of selection statements in Java: **if, if-else and switch**	Same as C and C++

Iteration statement	These specify how and when looping will take place. There are three types of iteration statements; **while, do** and **for.**	Same as C and C++ except for jump and labels
Jump statement	Jump Statements pass control to the beginning or end of the current block, or to a labeled statement. Such labels must be in the same block, and **continue** labels must be on an iteration statement. The four types of Jump statement are **break, continue, return** and **throw.**	C and C++ do not use labels with jump statement
Synchronization statemet	These are used for handling issues with multithreading.	Now available in C and C++
Guarding statement	Guarding statements are used for safe handling of code that may cause exceptions (such as division by zero). These statements use the keywords **try, catch,** and **finally.**	Same as in C++ except finally statement


Java Statements
Expression Statement
Labelled Statement
Synchronization Statement
Guarding Statement
Control Statement
Selection Statement
Iteration Statement
Jump Statement
if
if-else
switch
break
continue
return
while
do
for