1. Difference Between While And Do While

|  |  |
| --- | --- |
| While | Do while |
| While the loop is an entry control loop because firstly, the condition is checked, then the loop's body is executed.  The statement of while loop may not be executed at all  The while loop terminates when the condition becomes false  In a while loop, the test condition variable must be initialized first to check the test condition in the loop.  In a while loop, at the end of the condition, there is no semicolon. **Syntax:**  while (condition)  In a while loop, the number of executions depends on the condition defined in the while block | The do-while loop is an exit control loop because in this, first of all, the body of the loop is executed then the condition is checked true or false  The statement of the do-while loop must be executed at least once  As long as the condition is true, the compiler keeps executing the loop in the do-while loop  In a do-while loop, the variable of test condition Initialized in the loop also  In this, at the end of the condition, there is a semicolon. **Syntax:**  while (condition);  In a do-while loop, irrespective of the condition mentioned, a minimum of 1 execution occurs |

|  |
| --- |
|  |

Difference Between For And While Loop

|  |  |
| --- | --- |
| **For Loop** | **While Loop** |
| In the **for** loop, the **initialization**, checking of the condition, and the iteration statement are all written atop the loop  The **'for'** loop is more appropriate for use in case the number of iterations is known in advance  In case the condition is absent in the**'for' loop**, then the loop will iterate countless times.  In the **'for'** loop, once the initialization is carried out, it is not repeated | In the case of the **while** loop, only **initialization** and checking of the condition is carried out atop the loop  The**'while'** loop is preferably used in case the number of iterations is not specified beforehand  In case of the condition being absent in the**'while' loop**, a compilation error will be provided by the program.  In the case of the **‘while’** loop, in case the initialization is carried out during the stage of condition checking. The initialization has to be performed every time the loop iterates |

# 3)Difference between & and &&

|  |  |
| --- | --- |
| Bitwise And Operator (&) | Logical And Operator (&&) |
| It evaluates both left and right side of the expression It operates on "Boolean datatype" as well as operates on "bits".  Use to check logical condition and also used to mask off certain bits | It evaluates both left and right side of the expression It operates only on "Boolean datatype"  Used only to check logical condition |

1. Difference Between Break and Continue

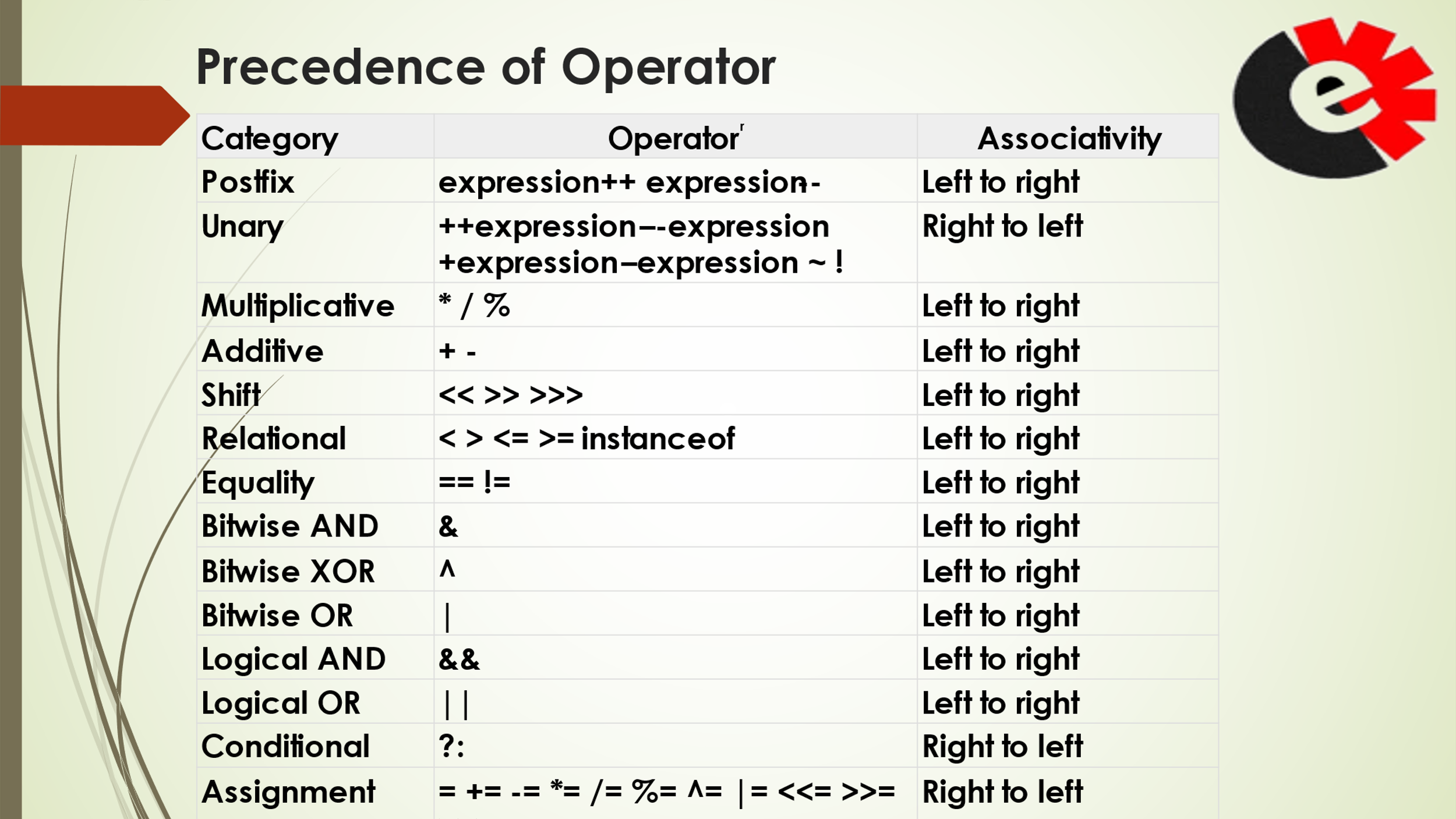
|  |  |
| --- | --- |
| **Break** | **Continue** |
| It terminates the execution of remaining iteration of the loop  break' resumes the control of the program to the end of loop enclosing that 'break  It causes early termination of loop  break' stops the continuation of loop  break' can be used with 'switch', 'label | It terminates only the current iteration of the loop  continue' resumes the control of the program to the next iteration of that loop enclosing 'continue'  It causes early execution of the next iteration  continue' do not stops the continuation of loop, it only stops the current iteration  'continue' can not be executed with 'switch' and 'labels' |

# Java Operator Precedence

* Operator precedence determines the order in which the operators in an expression are evaluated.
* When two operators share a common operand, 4 in this case, the operator with the highest precedence is operated

### Associativity of Operators in Java

* If an expression has two operators with similar precedence, the expression is evaluated according to its associativity (either left to right, or right to left).



### 

1. Nested For Loop

* Nested Loops are loops where a loop appears inside the body of another loop.

• The loop inside the body is called the inner loop. The other is called the outer loop.

• The inner loop completes all passes for a single pass of the outer loop

• This is very useful for many types of algorithms, especially with data that has more than one dimension