java provides 3 types of control flow ststements thgey are:

Desicion making staments:

if statements

switch statements

loop statements:

for loop

while loop

do while loop

for each loop (check it in arrays)

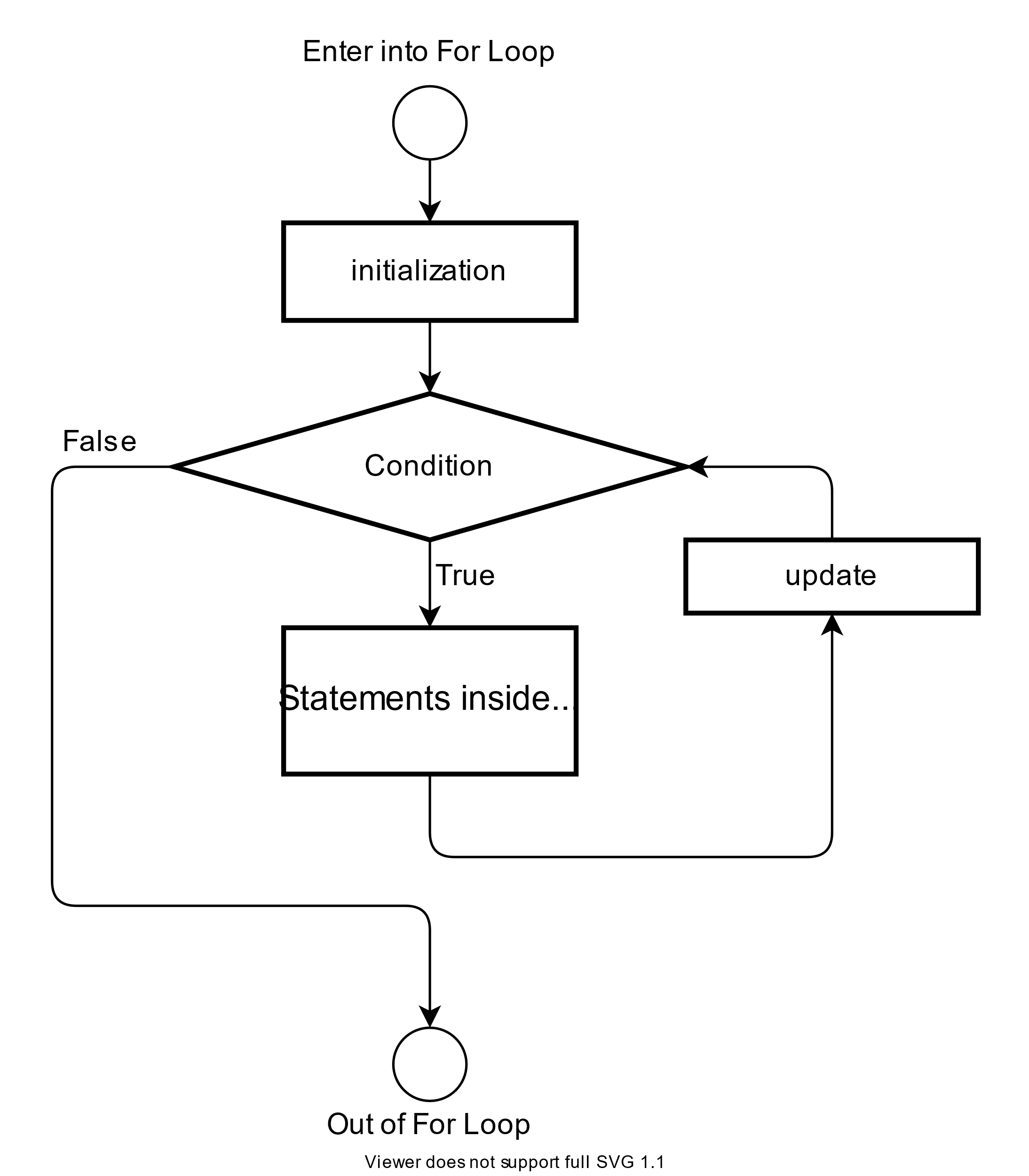
jump statements :

break statement

continue statement

Loop statements

1. for loop
2. while loop



1. do while loop
2. initialization
3. condition check
4. body
5. update again goes to condition check , if condition satisfies loop goes to 2nd step and again flow continues.

If condition does not satisfy loop terminates

Eg: For\_Loop

Eg: For\_Loop\_Eg2

// if there is single line statements we can omit curly braces { }

In the program for loop has only one statement that is If has only one statement that is x increment so its curly braces also omitted

Print statement is outside the for loop .

The Short circuit OR operator checks only for true , if true comes first , it does not check the second expression.

Go through the program flow

Eg: For\_Loop\_Eg3

For(stmnt-1 ; stmnt-2 ; stmnt-3){

Stmnt-4

}

Stmnt-1 can be any statement, but suggested for initialization

Stmnt-2 Should be Boolean statement only

Stmnt-3 can be any statement, but suggested for Increment/decrement.

Stmnt-4 can be any statement , but suggested for repeatative

logic.

Stmnt-1 is executed only once , so print statement int stmnt-1 is executed once , after that increment/decrement and condition is checked for every iteration

Eg: For\_Loop\_Eg4

// go through the code and previous example above explanation.

Eg: For\_Loop\_Eg5

// if there is no condition compiler evaluates the condition to “true” to enable the program to execute ,since there is no incrementation / decrementation the compiler will always check the condition , which it has marked as true

So it becomes a infinite loop

Note: for a boolean variable the default value is false , and for a literal it is set to true by compiler to execute the program.

Eg: For\_Loop\_Eg6

// we have used boolean literal compiler will evaluate during compile time , and since condition is true for loop becomes infinite loop and compiler says code below for loop will not get executed (unreachable code).

Eg: For\_Loop\_Eg7

// go through the code

Eg: For\_Loop\_Eg8

// since no condition compiler sets boolean literal value true to execute the loop , since it becomes infinite loop code below for loop is unreachable

Eg: For\_Loop\_Eg9

// since condition in the code is operand compiler will not evaluate the condition jvm will evaluate the condition, compiler will evaluate the condition only if final variables (or) literals during compile time.

Note:

if variables are marked as final ,then those values are known to compiler, we can say them as compile time constants.

If we try to change / reassign final variables it would be compile time error.

In java memory for a variable is given by jvm as per data type and value will also be assigned by jvm only, compiler will not allocate memory for variables and initialize them

Eg: For\_Loop\_Eg10

// since if consists of single statement braces are removed .

Whenever break statement is invoked control comes out of the loop and loop is terminated

Eg: For\_Loop\_Eg11

For a%b here a is dividend and b is divisor

Note :

0 % any number = 0

small number % big number = small number .

Eg: For\_Loop\_Eg12 // go through the program

Note : break statement moves the control out of the current working loop

continue skips the current iteration and further iterations work normally

Eg: For\_Loop\_Eg13

// here since break statement is used for label 1 control comes out of outer for loop and no output.

Eg: For\_Loop\_Eg14

//go through the program

Eg: For\_Loop\_Eg16

In loop condition if post increment is made first condition is evaluated and then incremented (line 3).

In line 5 the post increment and assignment is happening in same variable , so there is no scope for increment /increment is not valid only assignment is valid.

If assignment and post increment is made on different variable then first assignment is done and increment will also be valid .

Eg: For\_Loop\_Eg17

In for loop you omit initialization , condition and increment which will become infinite for loop. There will be no compilation error .

If you want to use variable in the loop initialization before using it must be compulsory.

if intialization is given condition and increment is not given it will be infinite loop.

If condition or incrementation/decrementation either one of them is not given it will lead to infinite loop .

Incrementation / decrementation can be done in loop body also, but condition should be given only in the loop declaration .

Eg: For\_Loop\_Eg18

// go through the code .

Eg: For\_Loop\_Eg19

// go through the code

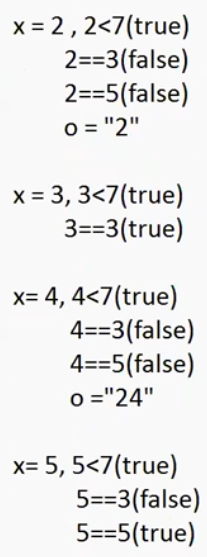
Eg: For\_Loop\_Eg20

// go through the code

Eg: For\_Loop\_Eg21

// go through the code

Eg: For\_Loop\_Eg22



// go through the code

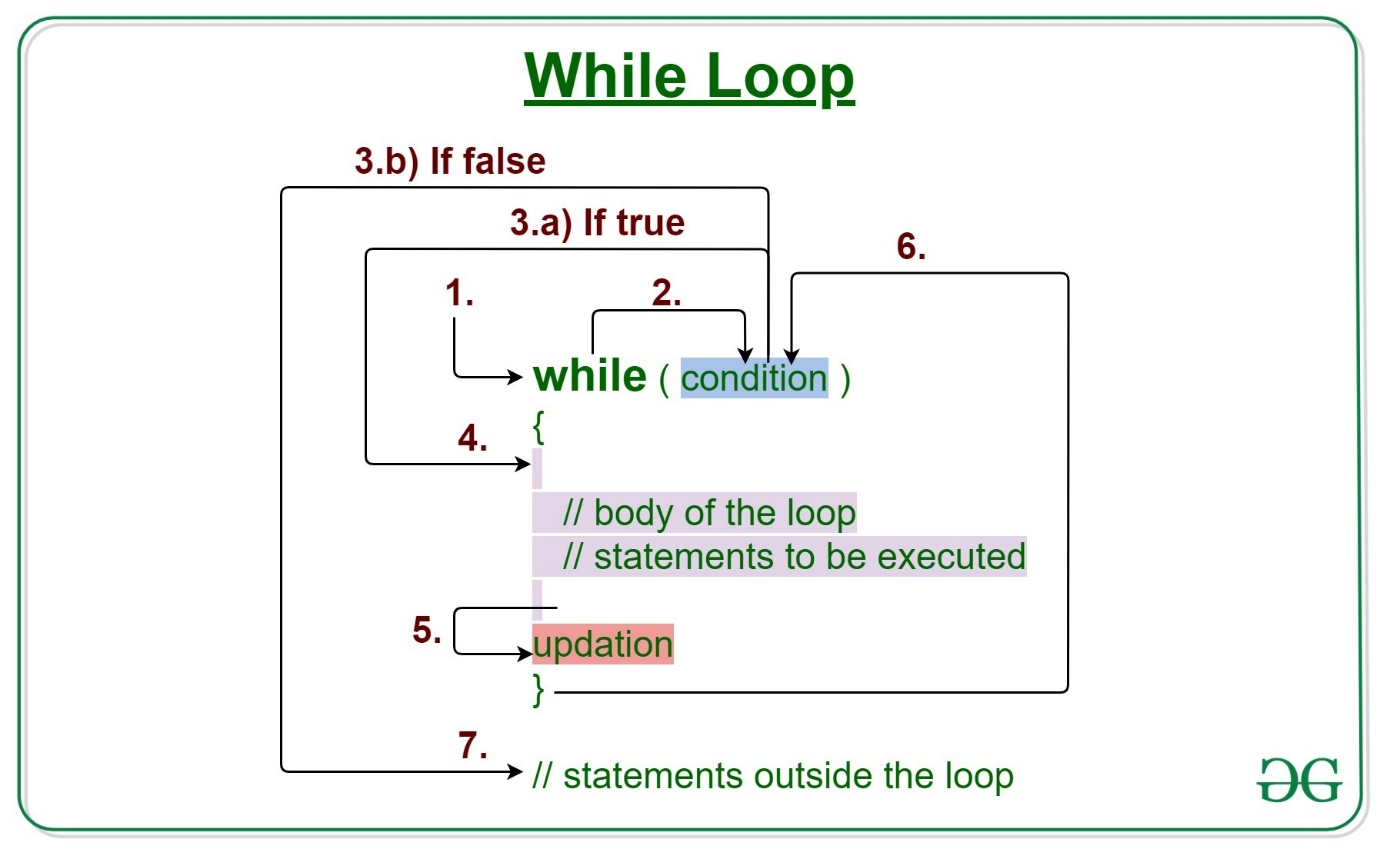
while loop:

In while loop ( increment , update , initialization ) is optional

It can just work with condition.

While loop is considered as the repeating if statement

If no .of iterations is not known we can use while loop



Eg: While\_Loop

Eg: While\_Loop\_Eg1

// since the condition is true while loop is executed , again compiler checks for condition ,since there is no boolean value false , the compiler keeps on executing the while loop. Which leads to infinite loop.

Eg: While\_Loop\_Eg2

// condition should always be boolean for an while loop it cannot be other data type .

Eg: While\_Loop\_Eg3

// if there is one statement in the body we can omit curly braces , but that single statement should not be a declarative statement (int a = 10) .

But if curly braces are there the loop can also have only declaration , no compile time error .

; -> semi colon is also considered as statement . so curly braces are omitted

Eg: While\_Loop\_Eg4

// compiler knows if the condition is true the while loop will execute infinite times , so whatever below the while loop is not at all executed , and it says as unreachable statement (compile time error )

Eg:While\_Loop\_Eg5

// compiler says if the condition is false the while loop will never be executed so what is the need to write the while loop

It says unreachable statement ,compile time error

Eg:While\_Loop\_Eg6

// if we use variables/ operands in expression the condition is evaluated at runtime by jvm . but if we use direct literals the compiler will evaluate the condition

In this program the compiler will not evaluate the condition since they are operands involved they are evaluated at runtime by jvm .

Eg: While\_Loop\_Eg7

// whenever we use a final variables the compiler will get to know the value of the variables at the compile time itself , if the values are known to the compiler at the compile time , the condition gets evaluated at the compile time

So in the program if the condition is false , the compiler says if you are writing a false condition what is the need of writing the loop (compile time error ) .

If the condition is true and you have written the code below the loop , it says the loop will execute infinite times and code below loop will never gets executed then what is the need of writing the code (compile time error )

Eg: While\_Loop\_Eg8

Line : 6 is unreachable statement (compile time error) because, since literal is used in the while loop , the compiler evaluate the condition during compile time and, it gets to know while loop is executed infinite times and code below while loop will never gets executed.

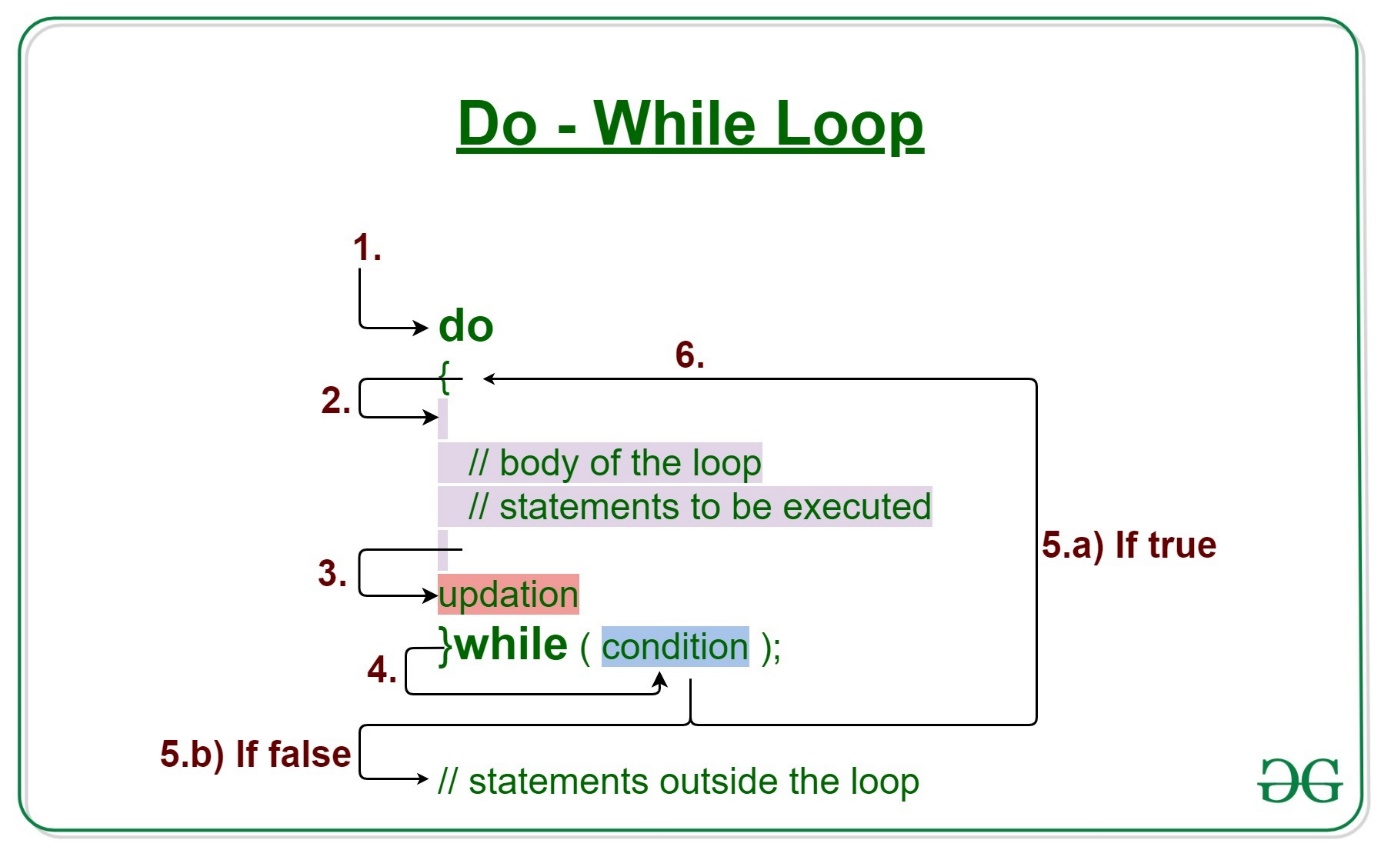
Eg: While\_Loop\_Eg9

//here isAvailable() (below while loop) variable x is a local variable so its scope is only within block , so decrement changes will not reflect in while loop , it only returns boolean , when condition x<1 , it returns false and while loop stops iteration .

isAvailable() in whileloop use local variable of main method x = 5 , and it is only decremented in print statement . five is printed and then decremented to 4 , 4 passed as parameter to isAvailabe(x) and isAvailable() below whileloop takes value will check x-- >0 , returns true until x >0 . when it is false , while loop stops execution .

do while :

irrespective of the condition the loop will execute once .



Whenever there is a need to execute the loop once even though the condition is false in that case we use do while loop.

Java do-while loop is called an exit control loop.

Eg: Do\_While

Eg: Do\_While\_Eg2

// we can omit braces if one statement in for loop, while loop as well as do while loop

Eg: Do\_While\_Eg3

// if you are executing do while loop there should be atleast one statement in the body otherwise , it leads to compile time error.

Eg: Do\_While \_Eg4

Semicolon is also considered as the statement , so no erorr.

Note : if there is no curly braces you cannot use declarative statements in loops ,

If curly braces are present and it is a single statement declarative statements are allowed .

Eg: Do\_While\_Eg5

If the loops are written in single line first split them .

Here inner while loop is the single line statement for do block , println statement is the single line statement for inner while loop.

And while is asusually terminated with semi colon no so error.

Eg: Do\_While\_Eg6

// go through the code.

Eg: Do\_While\_Eg7

// since do while loop has false condition , code in do executes once and control comes out of loop for other statements in the body.

Eg: Do\_While\_Eg8

// since literals are not involved compiler will not directly involve , so there is no compile time error and at “hi” print statement . hello is executed infinite times

Eg: Do\_While\_Eg9

// go through the code , similar code as Eg8

Eg: Do\_While\_Eg10

// go through the code

Eg: Do\_While\_Eg11

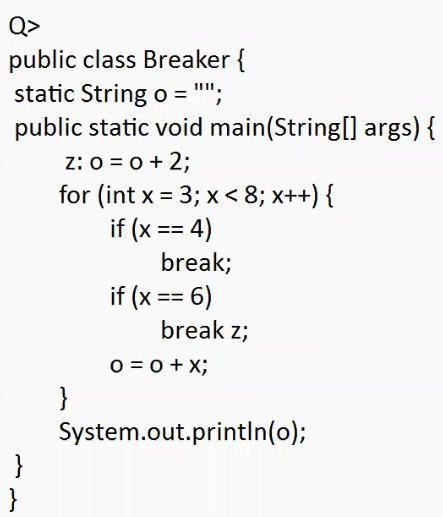
// go through the code

Eg: Do\_While\_Eg12

The continue statement is skips only the loop , but not the block.

Eg: Do\_While\_Eg13

If post increment is found in print statement , loops , conditional statements , and assigning to other variables first their value is loaded and then incremented.



Here z is a label , we cant use label for a statement. Label is used only with loop , to come out of the loop. So it results in compile time error.