**2.8 Java Iteration :-**

**1) Iteration** means the act of repeating a process with the aim of approaching a desired goal, target or result. Each repetition of the process is also called an "iteration," and the results of one iteration are used as the starting point for the next iteration.

A pentagon iteration. Connecting alternate corners of a regular pentagon produces a [pentagram](http://en.wikipedia.org/wiki/Pentagram) which encloses a smaller inverted pentagon. Iterating the process produces a sequence of nested pentagons and pentagrams.

**Iteration** in mathematics may refer to the process of [iterating a function](http://en.wikipedia.org/wiki/Iterated_function) i.e. applying a function repeatedly, using the output from one iteration as the input to the next. Iteration of apparently simple functions can produce complex behaviours and difficult problems - for examples, see the [Collatz conjecture](http://en.wikipedia.org/wiki/Collatz_conjecture) and [juggler sequences](http://en.wikipedia.org/wiki/Juggler_sequence).

Another use of iteration in mathematics is in [iterative methods](http://en.wikipedia.org/wiki/Iterative_method) which are used to produce approximate numerical solutions to certain mathematical problems. [Newton's method](http://en.wikipedia.org/wiki/Newton%27s_method) is an example of an iterative method.

**Iteration** in computing is the repetition of a block of statements within a [computer program](http://en.wikipedia.org/wiki/Computer_program). It can be used both as a general term, synonymous with repetition, and to describe a specific form of repetition with a [mutable](http://en.wikipedia.org/wiki/Mutable_object) state.

When used in the first sense, [recursion](http://en.wikipedia.org/wiki/Recursion) is an example of *iteration*, but typically using a *recursive notation*, which is typically not the case for *iteration*.

However, when used in the second (more restricted) sense, iteration describes the style of programming used in imperative programming languages. This contrasts with recursion, which has a more declarative approach.

Here is an example of iteration relying on [destructive assignment](http://en.wikipedia.org/wiki/Assignment_%28computer_science%29), in imperative [pseudocode](http://en.wikipedia.org/wiki/Pseudocode):

a = 0

for i from 1 to 3 // loop three times

{

a = a + i // add the current value of i to a

}

print a // the number 6 is printed (0 + 1; 1 + 2; 3 + 3)

In this program fragment, the value of the variable *i* changes over time, taking the values 1, 2 and 3. This changing value—or *mutable state*—is characteristic of iteration.

Iteration can be approximated using recursive techniques in [functional programming languages](http://en.wikipedia.org/wiki/Functional_programming_language). The following example is in [Scheme](http://en.wikipedia.org/wiki/Scheme_%28programming_language%29). Note that the following is recursive (a special case of iteration) because the definition of "how to iterate", the iter function, calls itself in order to solve the problem instance. Specifically it uses [tail recursion](http://en.wikipedia.org/wiki/Tail_recursion) so it does not use large amounts of stack space.

(let iterate ((i 1) (a 0))

(if (<= i 3)

(iterate (+ i 1) (+ a i))

(display a)))

An [iterator](http://en.wikipedia.org/wiki/Iterator) is an object that provides iteration as a generic service, allowing iteration to be done in the same way for a range of different data structures. Conversely, an [iteratee](http://en.wikipedia.org/wiki/Iteratee) is an abstraction which accepts or rejects data during an iteration process (controlled externally by an [enumerator](http://en.wikipedia.org/wiki/Iteratee#Enumerators) - so unlike with code that uses iterators, the iteratee code is not "in charge" of the iteration process).

Iteration is also performed using a worksheet, or by using solver or goal seek functions available in Excel. Many implicit equations like the Colebrook equation can be solved in the convenience of a worksheet by designing suitable calculation algorithms.[[1]](http://en.wikipedia.org/wiki/Iteration#cite_note-1)

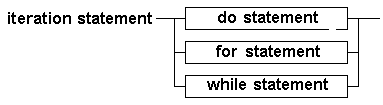
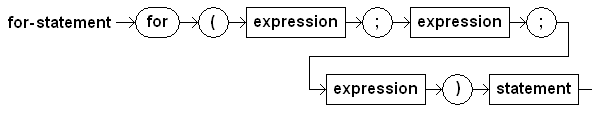
Many of the engineering problems like solving Colebrook equations reaches 8-digit accuracy in as small as 12 iterations and a maximum of 100 iterations is sufficient to reach a 15-digit accurate result.[[2]](http://en.wikipedia.org/wiki/Iteration#cite_note-2) .

**Iterations** in a project context may refer to the technique of developing and delivering incremental components of business functionality, product development or process design. This is most often associated with [agile software development](http://en.wikipedia.org/wiki/Agile_software_development), but could potentially be any material. A single iteration results in one or more bite-sized but complete packages of project work that can perform some tangible business function. Multiple iterations recurse to create a fully integrated product. This is often contrasted with the [waterfall model](http://en.wikipedia.org/wiki/Waterfall_model) approach.[*[citation needed](http://en.wikipedia.org/wiki/Wikipedia:Citation_needed" \o "Wikipedia:Citation needed)*]

**2) Java Iteration :-** These statements provide Iteration control structures to execute a section of code repeatedly if and only if an explicit run-time condition is met.  For the *do* statement and the *while* statement, the condition is an expression which evaluates to a boolean value, that is, either true or false.  For the *for* statement, the initialization expression can be any expression, the for-control is an expression which evaluates to an integral value, and the for iteration expression can be any expression.

Iteration statements are also referred to as iterative or looping statements.

**Syntax Diagrams**

[do Statement Syntax Diagram](http://java.comsci.us/syntax/statement/dowhile.html)[](http://java.comsci.us/syntax/statement/for.html)[While Statement Syntax Diagram](http://java.comsci.us/syntax/statement/while.html)

**BNF Syntax**

<iteration-statement>

::= <do-statement>

::= <for-statement>

::= <while-statement>

<do-statement>

::= **'do**' <body-statement> **'while**' **'(**' <control-expression> **')**' **';**'

<for-statement>

::= **'for**' **'(**' <for-initialization> **';**' <for-control> **';**' <for-iteration> **')**' <statement>

<while-statement>

::= **'while**' **'(**' <expression> **')**' <statement>

**EBNF Syntax**

<iteration-statement>

::= <do-statement>

  |  <for-statement>

  |  <while-statement>

<do-statement>

::= **'do**' <body-statement> **'while**' **'(**' <control-expression> **')**' **';**'

<for-statement>

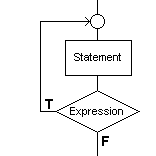
::= **'for**' **'(**' <for-initialization> **';**' <for-control> **';**' <for-iteration> **')**' <statement>

<while-statement>

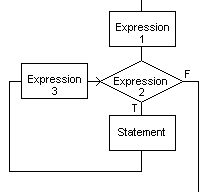
::= **'while**' **'(**' <expression> **')**' <statement>

**Flowcharts**

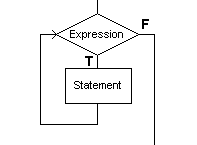
<do-statement>

[](http://java.comsci.us/syntax/statement/dowhile.html)

<for-statement>

[](http://java.comsci.us/syntax/statement/for.html)

<while-statement>

[](http://java.comsci.us/syntax/statement/while.html)

**Remarks**

* Iteration statements are also referred to as iterative, looping, or repetition statements.
* The statement part of an iteration statement is usually a compound statement, that is, a statement list enclosed by braces, {..}.
* The for statement is shorthand for a while loop:

for (A; B; C) D;

is the same as

A;

while (B) {

D;

C;

}