**Memory allocation** in java – specifies the mechanism where the computer program and services are assigned dedicated virtual memory spaces. The **Java Virtual Machine** splits the memory into stack and heap memory.

**Stack memory** – is used for static memory allocation and thread execution. Methods, local variables and reference variables are all stored in stack memory.

It gets erased after its execution.

**Characteristics of stack memory**:-

* It expands and contracts as new methods are called and returned.
* Variables within the stack only last as long as the method’s scope remains.
* When the method is executed, it is properly allocated and deallocated.
* Java throws java.lang.StackOverFlowError if this memory is full.
* Since each thread runs in its own stack, this memory is thread-safe.
* As compared to heap memory, access to this memory is fast.

**Heap memory** – any time an object is created and allocated in java heap space, it is used. In heap memory, new objects are often formed and references to these objects are stored in stack memory. Garbage collection, a discrete function, keeps flushing the memory used by previous objects that have no reference. A heap space object can have unrestricted access throughout the program.

These objects are accessible from anywhere in the program and have global access.

This memory model is divided into generations:-

* Young generation
* Old or tenured generation
* Permanent generation

**Characteristics of heap memory**:-

* Complex memory storage methods such as young, old and permanent generation are used to access it.
* Java throws java.lang.OutOfMemoryError if heap memory is full.
* Access to this memory is slower than access to stack.
* Unlike stack this memory is not immediately deallocated. Garbage collectors are used to free up unused objects in order to maintain memory efficiency.
* Heap unlike stack is not thread safe and must be protected by synchronizing the code properly.

Only a reference is created in java when we only declare a variable of a class type (memory is not allocated for the object). We must use new() keyword to assign memory to an object. As a result a heap memory is always assigned to the object.

Class test {

Void show() {

System.out.println(“inside test show()”);

}

}

Public class main {

Public static void main(String[] args) {

//all objects are dynamically allocated

Test t = new test();

t.show();

}

}

Output:- inside test show()

NOTE – here ‘t’ is a reference variable which refers to the test object which is in the heap memory.

**Garbage collection** – is a process of destroying runtime unused objects. It destroys the objects automatically. A garbage collector’s key goal is to allow effective use of memory.

Ways to make an object eligible for garbage collector

Three ways to make an object eligible for garbage collection –

* + - Nullifying the reference variable.
      * Student obj new Student();
      * obj = null;
    - by accessing a reference variable to another
      * Student obj1 = new Student();
      * Student obj2 = new Student();
      * obj1 = obj2;
    - by anonymous object
      * new Student();