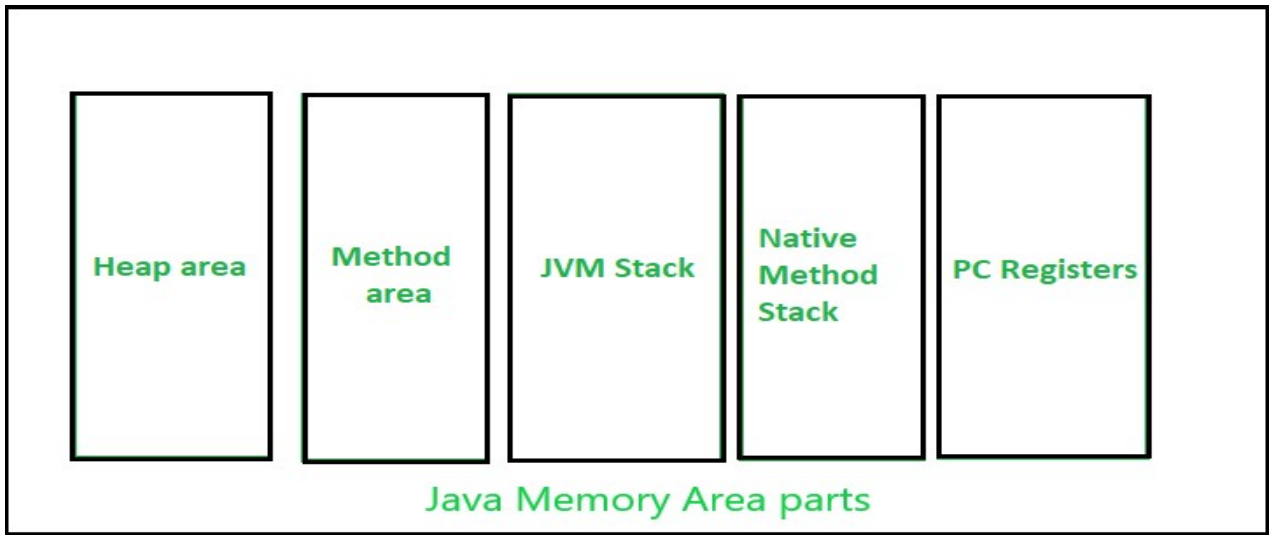
Week-4: Memory Allocation in Java

Introduction:

- In every programming language, the memory is a vital resource and is also scarce in nature. Hence it's essential that the memory is managed thoroughly without any leaks.
- Allocation and deallocation of memory is a critical task and requires a lot of care and consideration.
- However in Java, unlike other programming language, the JVM and to be specific Garbage Collector has the role of managing memory allocation so that the programmer needs not to.
- Whereas in other programming languages such as C the programmer has direct access to the memory who allocates memory in his code, thereby creating a lot of scope for leaks.
- The major concepts in Java Memory Management :
  - JVM Memory Structure
  - Working of Garbage Collector

Java Memory Structure:

- JVM defines various run time data area which are used during execution of a program. Some of the areas are created by the JVM whereas some are created by the threads that are used in a program.
- However, the memory area created by JVM is destroyed only when the JVM exits. The data areas of thread are created during instantiation and destroyed when the thread exits.



JVM Memory area parts

Heap Area:

- It is a shared runtime data area and stores the actual object in a memory. It is instantiated during the virtual machine startup.
- This memory is allocated for all class instances and array. Heap can be of fixed or dynamic size depending upon the system's configuration.
- JVM provides the user control to initialize or vary the size of heap as per the requirement. When a new keyword is used, object is assigned a space in heap, but the reference of the same exists onto the stack.

- There exists one and only one heap for a running JVM process.

Scanner sc = new Scanner(System.in);

  - The above statement creates the object of Scanner class which gets allocated to heap whereas the reference 'sc' gets pushed to the stack.
  - ***Note:*** *Garbage collection in heap area is mandatory.*

Method Area:

- It is a logical part of the heap area and is created on virtual machine startup.
- This memory is allocated for class structures, method data and constructor field data, and also for interfaces or special method used in class. Heap can be of fixed or dynamic size depending upon the system's configuration.
- Can be of a fixed size or expanded as required by the computation. Needs not to be contiguous.

JVM Stacks:

- A stack is created at the same time when a thread is created and is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.
- Stacks can either be of fixed or dynamic size. The size of a stack can be chosen independently when it is created.
- The memory for stack needs not to be contiguous.

Native method Stacks:

- Also called as C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when its created. And it can be of fixed or dynamic nature.

Program counter (PC) registers:

- Each JVM thread which carries out the task of a specific method has a program counter register associated with it. The non native method has a PC which stores the address of the available JVM instruction whereas in a native method, the value of program counter is undefined.
- PC register is capable of storing the return address or a native pointer on some specific platform.

Working of a Garbage Collector:

- Garbage collection in Java is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine.
- When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

What is Garbage Collection?

- In C/C++, a programmer is responsible for both the creation and destruction of objects. Usually, programmer neglects the destruction of useless objects. Due to this negligence, at a certain point, sufficient memory may not be available to create new objects, and the entire program will terminate abnormally, causing **OutOfMemoryErrors**.
- But in Java, the programmer need not care for all those objects which are no longer in use. Garbage collector destroys these objects. The main objective of Garbage Collector is to free heap memory by destroying **unreachable objects**. The garbage collector is the best example of the Daemon thread as it is always running in the background.

How Does Garbage Collection in Java works?

- Java garbage collection is an automatic process. Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.
- An in-use object, or a referenced object, means that some part of the program still maintains a pointer to that object.
- An unused or unreferenced object is no longer referenced by any part of the program. So the memory used by an unreferenced object can be reclaimed. The programmer does not need to mark objects to be deleted explicitly. The garbage collection implementation lives in the JVM.

Types of Garbage Collection:

There are five types of garbage collection are as follows:

i) **Serial GC:** It uses the mark and sweeps approach for young and old generations, which is minor and major GC.

ii) **Parallel GC:** It is similar to serial GC except that, it spawns N (the number of CPU cores in the system) threads for young generation garbage collection.

iii) **Parallel Old GC:** It is similar to parallel GC, except that it uses multiple threads for both generations.

iv) **Concurrent Mark Sweep (CMS) Collector:** It does the garbage collection for the old generation. We can limit the number of threads in CMS collector using **XX:ParalleCMSThreads=JVM option**. It is also known as Concurrent Low Pause Collector.

v) **G1 Garbage Collector:** It introduced in Java 7. Its objective is to replace the CMS collector. It is a parallel, concurrent, and CMS collector. There is no young and old generation space. It divides the heap into several equal sized heaps. It first collects the regions with lesser live data.

Advantages of Garbage Collection in Java

- It makes java memory-efficient because the garbage collector removes the unreferenced objects from heap memory.
- It is automatically done by the garbage collector(a part of JVM), so we don't need extra effort.

finalize () method:

- Just before destroying an object, Garbage Collector calls ***finalize()*** method on the object to perform cleanup activities. Once *finalize()* method completes, Garbage Collector destroys that object.
- *finalize()* method is present in Object class with the following prototype.

  protected void finalize() throws Throwable

- Based on our requirement, we can override *finalize()* method for performing our cleanup activities like closing connection from the database.
- The *finalize()* method is called by Garbage Collector, not JVM. However, Garbage Collector is one of the modules of JVM.
- Object class *finalize()* method has an empty implementation. Thus, it is recommended to override the *finalize()* method to dispose of system resources or perform other cleanups.
- The *finalize()* method is never invoked more than once for any object.
- If an uncaught exception is thrown by the *finalize()* method, the exception is ignored, and the finalization of that object terminates.