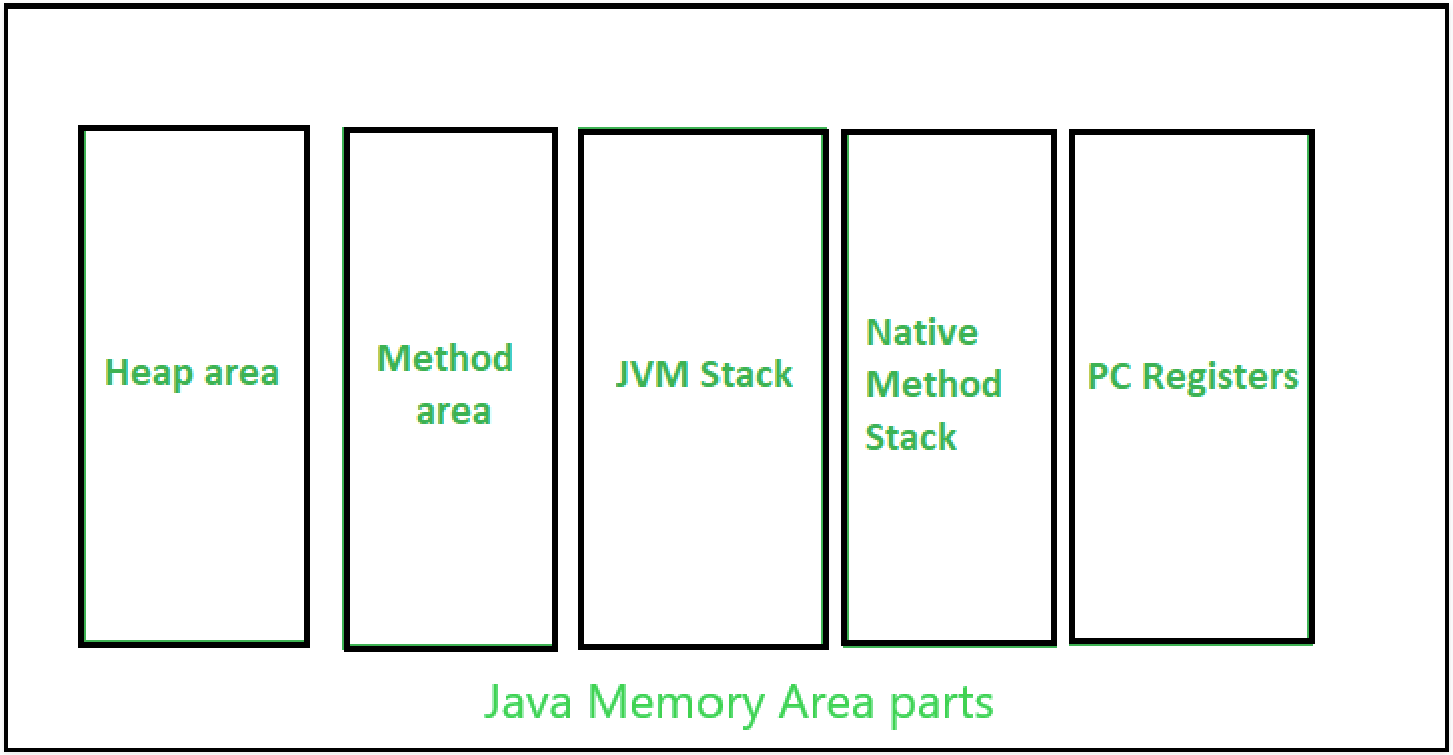
JAVA MEMORY MANAGEMENT

- **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 :**
- 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.

- **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.