**Stack Memory:**

* The stack is a region of memory used for managing method calls and local variables. Each thread of execution has its own stack.
* It stores method call frames (activation records) which contain information such as local variables, method parameters, and return addresses. These frames are pushed onto the stack when a method is called and popped off when the method returns.
* Stack memory is typically faster to allocate and deallocate compared to heap memory, as it involves moving the stack pointer.
* Local variables, primitive data types, and references to objects are often stored on the stack.

**Heap Memory:**

* The heap is a region of memory used for dynamic memory allocation. It's where objects are allocated when you use the new keyword to create instances of classes.
* Objects in the heap can be accessed from anywhere in the program, and their lifetimes are managed by the Java Virtual Machine (JVM) through a process known as garbage collection.
* The heap is larger in size compared to the stack and is typically used for storing objects with longer lifetimes, such as objects representing the main data structures of your program.
* Java's automatic memory management (garbage collection) helps reclaim memory occupied by objects that are no longer reachable, preventing memory leaks.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Garbage Collector**

All objects are allocated on the heap area managed by the JVM. As long as an object is being referenced, the JVM considers it alive. Once an object is no longer referenced and therefore is not reachable by the application code, the garbage collector removes it and reclaims the unused memory.