**Memory Management in Java**

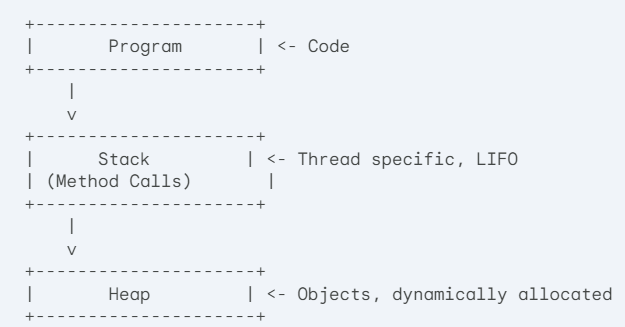
Java offers automatic memory management, a feature that simplifies development by eliminating the need for manual memory allocation and deallocation. This is handled by the Java Virtual Machine (JVM) which frees the programmer from the burden of memory-related issues like memory leaks.

**Memory Areas**

Java programs utilize two primary memory areas: the stack and the heap.

* **Stack:** The stack is a Last-In-First-Out (LIFO) data structure that grows and shrinks dynamically based on method calls. It's used to store primitive data types (like int, float, etc.), method parameters, and local variables. When a method is invoked, a new frame is pushed onto the stack, containing these elements. Once the method execution is complete, the frame is popped, and the memory becomes available again.
* **Heap:** The heap is a dynamically allocated memory pool used for objects. Objects are created using the new keyword, and references to these objects reside on the stack. The heap is not thread-safe, meaning multiple threads can access and modify it concurrently.

**Figure: Memory Areas in Java**

****

**Garbage Collection**

Automatic memory management comes into play with garbage collection. The garbage collector is a background process within the JVM that identifies and reclaims memory occupied by objects no longer referenced by the program. This ensures that memory is not wasted due to forgotten objects.

Here's a simplified breakdown of the garbage collection process:

1. **Marking:** The garbage collector identifies objects that are no longer reachable from the stack or any other references.
2. **Sweeping:** The unreachable objects are marked for deletion.
3. **Compaction (Optional):** In some garbage collection algorithms, the remaining live objects are compacted to reduce memory fragmentation.