Garbage collection (GC) in Java is an automatic memory management process that manages the allocation and deallocation of memory for Java applications. Here’s a simplified explanation of how garbage collection works in Java:

1. **Heap Memory**: Java programs use the heap memory area for dynamic memory allocation of objects. When you create objects in Java using new, memory is allocated on the heap.
2. **Memory Management**: Unlike languages like C or C++, where developers must manually allocate and deallocate memory (using malloc() and free()), Java handles memory management automatically through garbage collection.
3. **Garbage Collector**: The garbage collector is a part of the Java Virtual Machine (JVM) responsible for reclaiming memory occupied by objects that are no longer reachable or needed by the application.
4. **Reachability**: An object is considered reachable if it can be accessed directly or indirectly from the application’s root references (such as local variables, static fields, or method parameters).
5. **Garbage Collection Process**:
   * **Mark Phase**: The garbage collector traverses all reachable objects starting from the root references and marks them as live.
   * **Sweep Phase**: It then sweeps through the heap and deallocates memory of objects that were not marked as live (i.e., unreachable objects).
   * **Compact Phase (optional)**: Some garbage collectors may include compaction, where live objects are moved closer together to reduce heap fragmentation.
6. **Generational Garbage Collection**: Modern JVMs typically use a generational garbage collection algorithm, which divides the heap into different generations (young generation, old generation, and sometimes perm/gen space). This allows more efficient memory management by focusing garbage collection efforts on specific areas of the heap.
7. **Tuning and Configuration**: Java provides options to configure and tune garbage collection behavior based on application requirements and performance goals. This includes selecting different garbage collector algorithms (e.g., Serial GC, Parallel GC, CMS, G1 GC) and adjusting heap size parameters (-Xms, -Xmx, -XX:NewSize, -XX:MaxNewSize, etc.).
8. **Impact on Performance**: While garbage collection automates memory management and reduces the risk of memory leaks and dangling pointers, improper tuning or inefficient use of objects can lead to performance issues such as frequent pauses (stop-the-world events) during garbage collection cycles.

In summary, garbage collection in Java is a fundamental feature that simplifies memory management by automatically reclaiming memory from unreachable objects, ensuring efficient and reliable memory usage in Java applications.