In Java, the JVM provides several types of garbage collectors, each designed to manage memory differently to optimize performance for various use cases. Here are the main types of garbage collectors in Java:

1. Serial Garbage Collector:

* Uses a single thread for garbage collection.
* Suitable for single-threaded applications or small heaps.
* Enabled with the JVM option -XX:+UseSerialGC.

1. Parallel Garbage Collector (Throughput Collector):

* Uses multiple threads for garbage collection to minimize pause times.
* Designed for applications that run on multi-core processors.
* Enabled with the JVM option -XX:+UseParallelGC.

1. G1 Garbage Collector (Garbage-First):

* Aims to achieve both high throughput and low pause times.
* Divides the heap into regions and performs garbage collection on regions that contain the most garbage first.
* Enabled with the JVM option -XX:+UseG1GC.

1. CMS Garbage Collector (Concurrent Mark-Sweep):

* Focuses on low pause times.
* Mark-and-sweep phases run concurrently with application threads.
* Enabled with the JVM option -XX:+UseConcMarkSweepGC.
* Deprecated and replaced by G1GC in newer Java versions.

1. Shenandoah Garbage Collector:

* A low-pause-time garbage collector.
* Runs concurrently with the application threads.
* Available in JDK 12 and later,
* Enabled with the JVM option -XX:+UseShenandoahGC.

1. Z Garbage Collector (ZGC):

* Designed for low latency and can handle large heaps.
* Performs most of the work concurrently without stopping the application threads.
* Available in JDK 11 and later
* Enabled with the JVM option -XX:+UseZGC.

Each garbage collector has its own advantages and trade-offs, and the choice of which to use depends on the specific requirements and characteristics of the application.