Task 4: Research and present a comparison of different garbage collection algorithms (Serial, Parallel, CMS, G1, ZGC) in Java.

1. **Serial Garbage Collector**

   Characteristics:

   - Uses a single thread for garbage collection.
   - Simple and straightforward implementation.
   - Operates with stop-the-world pauses.

   Strengths:

   - Low overhead and easy to implement.
   - Suitable for single-threaded applications and environments with limited resources.

   Weaknesses:

   - Can cause long pause times, making it unsuitable for applications requiring low latency.
   - Not efficient for multi-threaded applications or large heaps.

   Use Cases:

   - Small applications or systems with a single CPU core.

2. **Parallel Garbage Collector**

   Characteristics:

   - Utilizes multiple threads for garbage collection in the young generation.
   - Also known as “Throughput Collector.”

   Strengths:

   - Reduces garbage collection time by using multiple threads.
   - Good for applications where high throughput is more important than pause time.

   Weaknesses:

   - Still experience stop-the-world pauses, which may be unacceptable latency-sensitive applications.

   Use Cases:

   - Applications with high throughput demands where pause time is less critical.

3. **Concurrent Mark-Sweep (CMS) Collector**

Characteristics:

- Designed to minimize pause time by performing most of the work concurrently with application threads.
- Uses multiple threads for garbage collection.

Strengths:

- Reduces pause time significantly compared to Serial and Parallel collectors.
- Works well for applications that require shorter pause times.

Weaknesses:

- Can lead to fragmentation as it does not compact the heap.
- Requires more CPU resources due to concurrent execution.
- More complex to tune and may result in more frequent GCs if not configured properly.

Use Cases:

- Applications where low pause times are more critical than throughput, such as interactive applications.

4. **G1 Garbage Collector**

Characteristics:

- Divides the heap into regions and performs garbage collection incrementally.
- Aims to provide predictable pause times.

Strengths:

- Provide better control over pause times compared to CMS.
- Compacts the heap as it collects, reducing fragmentation issues.

Weaknesses:

- More complex configuration and tuning compared to simpler collectors.
- Slightly high overhead due to the complexity of region-based collection.

Use Cases:

- Large applications requiring predictable pause times and reduced fragmentation.

5. **Z Garbage Collector (ZGC)**

Characteristics:

- A scalable low-latency garbage collector designed for very large heaps.
- Performs most of its work concurrently.

Strengths:

- Extremely low pause times (usually under 10 ms).
- Scales well with large heaps, up to terabytes in size.

Weaknesses:

- Higher CPU usage due to concurrent operations.
- Newer and less mature compared to older collectors.

Use Cases:

- Applications requiring minimal latency and capable of handling large amounts of data, such as high-frequency trading systems.