Day 9:

Task 1: Research and present a comparison of different garbage collection algorithms (Serial, Parallel, CMS, G1, ZGC) in Java.

Garbage collection is a crucial component of the Java Virtual Machine (JVM) that automatically manages memory by reclaiming the memory occupied by objects that are no longer in use. Java provides several garbage collection algorithms, each with its own strengths and trade-offs. Here's a comparison of the different garbage collection algorithms in Java:

1. Serial Garbage Collector:
   - This is the simplest and oldest garbage collector in Java.
   - It freezes all application threads during the garbage collection process, which can cause significant pauses (stop-the-world events).
   - Suitable for single-threaded applications or applications with small data sets.
   - Not recommended for server or multi-threaded applications due to long pause times.

2. Parallel Garbage Collector:
   - This collector uses multiple threads to perform garbage collection in parallel, which can reduce pause times compared to the Serial GC.
   - It is designed for applications with medium to large data sets running on multi-CPU or multi-core systems.
   - It still causes stop-the-world events, but the pause times are generally shorter than the Serial GC.
   - Suitable for applications with high throughput requirements.

3. Concurrent Mark Sweep (CMS) Garbage Collector:
   - This collector aims to minimize pause times by performing most of its work concurrently with the application threads.
   - It uses multiple threads to scan the heap and find live objects, and then performs a stop-the-world pause to reclaim the unreachable objects.
   - The pause times are generally shorter compared to the Parallel GC, but it can have a higher CPU overhead due to concurrent operations.

   - Suitable for applications with strict pause time requirements and large data sets.

4. Garbage-First (G1) Garbage Collector:
   - This is a low-pause-time collector designed for multi-threaded applications running on multi-core systems with large memory.
   - It divides the heap into multiple regions and performs garbage collection on a subset of regions at a time, minimizing pause times.
   - It can handle large heaps efficiently and provides better control over pause times compared to other collectors.
   - Suitable for applications with strict pause time requirements and large data sets, especially those running on multi-core systems.

5. ZGC (Z Garbage Collector):
   - This is a low-pause-time collector introduced in Java 11 and designed for large data sets and multi-core systems.
   - It uses a load barrier and concurrent marking to minimize pause times and provide high throughput.
   - It can handle large heaps efficiently and provides better control over pause times compared to other collectors.
   - Suitable for applications with strict pause time requirements, large data sets, and running on multi-core systems with a large amount of memory.

When choosing a garbage collection algorithm, consider factors such as the application's pause time requirements, throughput needs, heap size, and available CPU and memory resources. Generally, for applications with strict pause time requirements and large data sets, G1 or ZGC are recommended. For applications with medium to large data sets and high throughput requirements, the Parallel GC can be a good choice. CMS can be suitable for applications with strict pause time requirements but lower throughput needs. The Serial GC is generally not recommended for server or multi-threaded applications due to its long pause times.

Here's a table comparing the different garbage collection algorithms in Java:

Garbage Collector	Pause Times	Throughput	Heap Size	Use Case
Serial GC	Long	Low	Small	Single-threaded applications, small data sets
Parallel GC	Moderate	High	Medium to Large	Applications with high throughput requirements, multi-CPU/core systems
CMS GC	Short	Moderate	Large	Applications with strict pause time requirements, large data sets
G1 GC	Short	High	Large	Applications with strict pause time requirements, large data sets, multi-core systems
ZGC	Very Short	High	Large	Applications with strict pause time requirements, large data sets, multi-core systems, large memory

- Pause Times: Indicates the typical pause times or stop-the-world events caused by the garbage collector during its operation.
- Throughput: Refers to the overall application throughput or performance when using the garbage collector.
- Heap Size: Indicates the recommended heap size or data set size for which the garbage collector is optimized.
- Use Case: Suggests the typical scenarios or applications where the garbage collector is most suitable.