1. Heap Generations Recap

- **Young Generation**:
    - Divided into **Eden** + **Survivor spaces (S0, S1)**.
    - Most new objects are allocated in **Eden**.
    - GC here is **minor GC** (fast, frequent).
- **Old Generation (Tenured)**:
    - Stores long-lived objects.
    - Collected during **major/full GC** (slower, less frequent).

---

2. Object Lifecycle in GC

1. New objects → Eden
    - When you create an object, it usually goes to **Eden space**.
2. Minor GC happens
    - If Eden fills up, a **minor GC** runs.
    - Live objects in Eden are copied to a **Survivor space (S0 or S1)**.
    - Dead objects (unreachable) are cleared.
3. Survivor ↔ Survivor copying
    - On each minor GC, live objects in **Eden + one Survivor space** are copied into the **other Survivor space**.
    - Survivor spaces flip roles (from–space, to–space).
4. Promotion (tenuring) to Old Generation
    - Objects don’t stay in Survivor forever. They eventually get **promoted** to the Old Generation when:
5. Promotion rules:

- **Age threshold exceeded**:
  - Each time an object survives a minor GC, its **age increases** (stored in the object header).
  - If age ≥ `MaxTenuringThreshold` (default usually 15), it's promoted.
- **Survivor space overflow**:
  - If Survivor space doesn't have enough room, some objects are promoted early.
- **Large objects**:
  - Very large objects may be allocated **directly in Old Generation** if they don't fit in Eden/Survivor.

---

3. Example Walkthrough

- You create 1000 small objects → go into **Eden**.
- Minor GC runs → survivors copied to **S0**. (Age=1)
- Next Minor GC → survivors copied to **S1**. (Age=2)
- Repeat… until `MaxTenuringThreshold` is reached.
- Then → promoted to **Old Generation**.

---

4. Why do we promote?

- Young gen is designed for **short-lived objects** (most die fast).
- Old gen is for **long-lived objects** (cached data, sessions, etc.).
- This separation makes GC more efficient:
    - **Minor GC** = frequent but cheap.
    - **Major GC** = rare but expensive.

⚡ In short:

- **Eden → Survivor (S0/S1) → Old Generation**.
- Promotion happens when object **ages out**, **Survivor space is full**, or object is **too large**.

5. Minor GC

- **What it collects**:
    - Only the **Young Generation** (Eden + Survivor spaces).

- **Trigger**:
    - Happens when **Eden space is full**.
- **Process**:
    - Mark live objects in Eden and Survivor.
    - Copy survivors to the other Survivor space (or Old Gen if needed).
    - Clear Eden and the used Survivor.
- **Cost**:
    - Usually **fast** (small region, most objects die young).
- **Pause**:
    - Causes a **stop-the-world pause**, but very short.

👉 Example: If your program keeps creating many short-lived objects (like strings, request objects), Minor GC will happen frequently.

---

6. Major GC (a.k.a. Old GC)

- **What it collects**:
    - The **Old Generation** (tenured space).
- **Trigger**:
    - Happens when Old Gen is full (after objects get promoted from Young Gen).
- **Process**:
    - Can use **Mark-Sweep-Compact** or concurrent collectors (CMS, G1, ZGC, etc.).
- **Cost**:
    - Much **slower** than Minor GC (more objects to scan, compaction may be needed).
- **Pause**:
    - Usually longer stop-the-world pauses, though concurrent GCs (CMS, G1, ZGC, Shenandoah) reduce this.