Checkpoint Metacognitivo – Fase PROFUNDIZA

1. ¿Qué decisión de diseño de los lenguajes reales me sorprendió más?

La que más sorprende es la de **Python list con sobreasignación adaptativa**.

- No usa un factor fijo como Java o C++, sino que ajusta dinámicamente cuánto crecer según el tamaño actual.
- Esto rompe la idea clásica de que los arreglos dinámicos siempre crecen con un factor constante.
- Me sorprendió porque refleja un diseño pragmático: optimizar para cargas reales y reducir picos de memoria, en lugar de seguir una regla matemática rígida.

2. ¿Cómo el ejercicio de diseñar un arreglo especializado cambió mi perspectiva sobre los trade-offs?

- Antes pensaba que un ArrayList genérico era suficiente para casi todo.
- Al diseñar el MessageBuffer circular, entendí que los trade-offs dependen del patrón de uso real:
    - Si solo necesito los últimos 100 mensajes, un buffer circular es más eficiente que un arreglo dinámico.
    - Los trade-offs no son teóricos, sino prácticos: memoria exacta, operaciones O(1), evicción automática.
- Ahora veo que diseñar estructuras a medida puede ser más simple y eficiente que usar una genérica.

3. ¿Qué concepto de esta semana anticipo que será más útil en las siguientes semanas?

El concepto de amortización del costo en operaciones de arreglos dinámicos.

- Entender que una operación cara (redimensionar) se distribuye entre muchas operaciones baratas.
- Esto será clave cuando estudiemos otras estructuras como hash tables o heaps, que también dependen de costos amortizados.

4. Si empezara la semana de nuevo con lo que sé ahora, ¿qué haría diferente?

- Haría más pruebas de estrés desde el inicio (casos extremos de inserción, eliminación, redimensionamiento).
- Me enfocaría en patrones de uso reales en lugar de solo ejemplos académicos.
- Documentaría mejor los trade-offs de cada decisión de diseño, porque eso me ayuda a comparar alternativas con más claridad.