가비지 컬렉션

이것이 C#이다

Contents

- 가비지 컬렉터를 아시나요?
- 개처럼 할당하고 정승처럼 수거하라
- 세대별 가비지 컬렉션
- 가비지 컬렉션을 이해했습니다. 우리는 뭘 해야 하죠?

22.1 가비지 컬렉터를 아시나요?

- **프로그래머가 쉽게 저지르는 실수**
  - 해제한 메모리 액세스
- **메모리 관리의 실수를 줄이기 위한 C#의 해결 책**
  - 자동 메모리 관리 기능 → 가비지 컬렉션 (쓰레기 수거)
  - 가비지 컬렉션을 담당하는 친구 → 가비지 컬렉터
- **가비지 컬렉터도 소프트웨어다.**
  - 컴퓨팅 자원을 사용한다.
  - 최소한의 자원 사용을 추구하자.
  - →가비지 컬렉터의 동작 메커니즘 이해 필요
  - →그에 따른 코딩 가이드라인 수립

22.2 개처럼 할당하고 정승처럼 수거하라

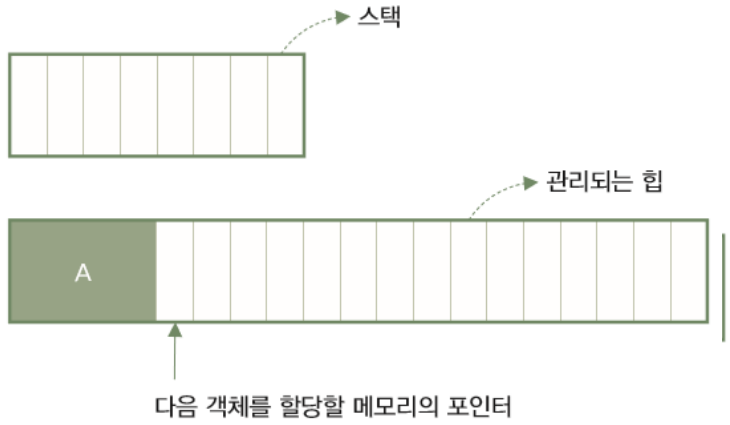
❖ CLR에서 메모리에 객체 할당 방법

- 관리되는 힙을 위한 메모리 공간 확보
- object A = new object();
- object B = new object();

```
if ( true )
{
    object a = new object()
}
```

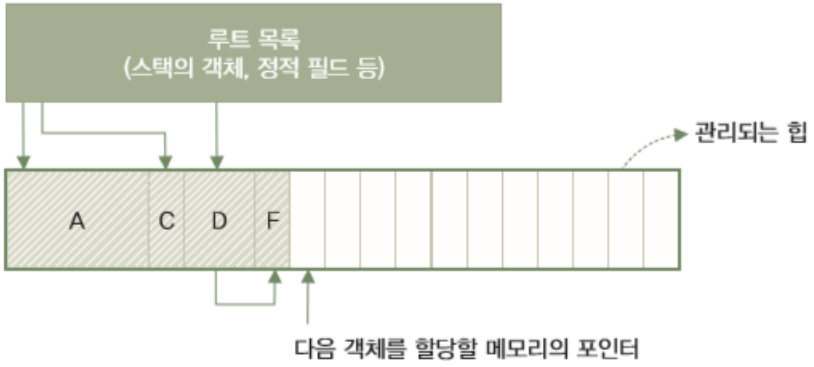
❖ 할당한 객체의 메모리 해제 과정

- 값 형식-스택, 참조 형식- 힙
- If 블록 안의 참조 a와 메모리
- If 블록이 끝난 경우의 a와 메모리



❖ 루트 목록으로 GC의 쓰레기 객체 정리 과정

- 할당된 메모리 위치 참조 객체
- JIT 컴파일러에서 루트 목록 만듦
- CLR에서 루트목록 관리 및 상태 갱신



22.3 세대별 가비지 컬렉션

❖ CLR은 메모리를 3개의 세대(0,1,2)로 관리

- 0 세대: 가비지 컬렉션을 한번도 겪지 않은 '갓 생성된' 객체
- 2 세대: 최소 2회 이상 가비지 컬렉션을 겪고도 살아남은, 산전수전 다 겪은 객체
- 1 세대: 0세대에서 2세대로 넘어가는 과도기의 객체
- 가비지 컬렉션 빈도: 2세대 < 1세대 < 0세대

❖ Full GC

- 2세대 힙이 가득 차는 경우
- →CLR에서 애플리케이션 실행 일시 중단
- →0-2세대 전체 메모리에 걸쳐 쓰레기 수집
- →애플리케이션이 차지하는 메모리가 클 수록 장시간 실행 중지

12.4 가비지 컬렉션을 이해했습니다. 우리는 뭘 해야 하죠?

- **가비지 컬렉션 메커니즘의 이해를 바탕으로 적절한 작전 수립**
- **객체를 너무 많이 할당하지 마세요**
  - 너무 많은 객체는 관리되는 힙의 각 세대에 메모리 포화 초래
  - →빈번한 가비지 컬렉션 호출
- **너무 큰 객체 할당을 피하세요**
  - 85KB 이상 객체를 할당하는 대형 객체 힙(LOH)과 소형 객체 힙(SOH)
  - 대형 객체 힙은 할당 시의 성능과 메모리 공간 효율이 좋지 않다.
  - LOH는 2세대 힙으로 간주 → 전 세 대에 가비지 컬렉션 촉발 → 일시적 중지
- **너무 복잡한 참조 관계는 만들지 마세요.**
  - 참조 관계가 많은 객체는 가비지 컬렉션 후에 살아남았을 때 문제.
  - → 객체의 각 필드 간 참조 관계를 모두 조사, 참조하는 메모리 주소 모두 수정
  - 가비지 컬렉션을 회피 하기 위해 오버헤드가 큰 쓰기 장벽(Write barrier)생성 문제
- **루트를 너무 많이 만들지 마세요.**
  - 루트목록이 작을 수록 컬렉터 검사 횟수가 줄어 가비지 컬렉션을 더 빨리 끝냄.

Thank You !

이것이 C#이다