01.02

한달간 피드백

#12일

Solid

1) Single responsibity priciple 단일 책임 원칙

> 클래스는 하나의 기능만을 가지고 클래스가 제공하는 모든 서비스는 하나의 책임을 수행하는데 집중되어 있어야 한다.

2) Open close principle 개방폐쇄원칙

확장에는 개방되어 있고, 변경에는 폐쇄되어야 한다.

3) Liskov Substitution Principle 리스코프 치환법칙

서브 타입은 언제나 기반 타입으로 교체할 수 있어야 한다.

4) Interface segregation principle 인터페이스 분리법칙

하나의 일반적인 인터페이스보다는 여러개의 구체적인 인터페이스가 낫다

5) Dependency inversion principle 의존역전법칙

추상화에 의존해야지 구체화에 의존하면 안된다.

HashMap에서 충돌을 핸들링하는 방법

- Open address

1) 다음칸을 계속 확인하며 비어있는 칸 찾기

2) 제곱수의 칸을 확인하며 비어있는 칸 찾기

3) 2차 해시함수 적용하기

- Seperate chaining

1) linkedlist 사용하기

2) Tree 사용하기

#17일

SHA-1

- 문서변조여부를 확인하기 위한 해싱에서 해시함수 중 하나
- 입력을 받고 메시지 다이제스트라는 160비트(20바이트) 해시값을 만드는 암호화 해시 함수
- **인터넷 보안 프로토콜과 공개키 인증서에 적용(TLS/SSL 인증서)**
- **SHA-1은 160비트의 메세지 다이제스트를 생성하는데 무차별 대입 공격으로 동일한 해시를 만들 수 있는 취약점**
- 해시 충돌의 위험이 있음 (해싱되는 결과값이 같아지는 경우가 존재한다.)
- 해시함수를 통해 압축 적용단계가 많아질수록 복잡해져서 문서의 변조를 막는다.

#25일

cms 순서

1) **initial mark** : GC Root로부터 바로 참조되거나, young 영역의 살아있는 객체로부터 바로 참조되는 old 영역의 객체를 mark (STW 발생)

2) **Concurrent mark** : initial mark단계에서 mark한 객체부터 시작해서 **old영역을 순회**하면서 살아있는 모든 객체들을 markt한다. (STW 발생하지 않고 애플리케이션 스레드를 멈추지 않고 동작)

3) **Concurrent reclean** :

Card라는 힙영역 안에 dirty로 표시한 객체로부터 참조되고 있는 객체를 mark (STW 발생하지 않고 애플리케이션 스레드를 멈추지 않고 동작)

4) **Concurrent aborable Preclean**

변화한 객체들을 계속 **스캔**

(STW 발생하지 않고 애플리케이션 스레드를 멈추지 않고 동작, Final Remark의 수행시간을 최대한 짧게 만든다)

5) **Final Remark**

STW를 통해 애플리케이션 스레드를 잠시 멈춤으로써 **객체들의 상태를 완전히 반영**한다. 이 단계를 young 영역이 거의 비워져있을 때 수행하게 하도록 하여 STW을 일으키는 동작이 연쇄적으로 발생하는 것을 방지

6) **Concurrent sweep**

애플리케이션 스레드와 병렬적으로 수행, 사용하지 않는 객체들을 정리하여 **빈공간을 확보**한다

→ CMS는 컴팩션 안한다

#26일

NIO

- 읽는 데이터를 무조건 buffer에 저장해서 버퍼내 데이터의 위치를 이동해 필요한 부분만 읽고 쓴다.
- 데이터 IO를 channel을 통해 읽는다.
  - channel : non-blocking read를 할 수 있도록 지원하는 connection
  - 양방향으로 입력과 출력이 가능하다.
  - 채널에서 데이터를 주고 받을 때 사용하는 것이 버퍼이다.
  - ServerSocketChannel과 SocketChannel을 이용하여 TCP 네트워크 프로그램에 이용한다.
- 넌블로킹, 블로킹 둘다 가짐
  - 파일을 읽는 File I/O 는 blocking. 그 외는 non blocking
- selector
  - 여러개의 채널에서 발생하는 이벤트(연결이 생성됨, 데이터가 도착함 등)를 **모니터링할 수 있는 객체**다. 하나의 selector(스레드)에서 여러 채널에 대해 지속적으로 모니터링 한다.
  - 실제 수신되는 데이터가 없음에도 무한루프를 돌며 버퍼에 데이터 여부를 계속 확인하게 되는데 이를 방지하기 위해 사용된다.

- 연결 클라이언트가 많고 IO가 작은 경우
    - NIO는 버퍼할당크기가 문제가 되고, 모든 입출력 작업에 버퍼를 무조건 사용해야 하므로 즉시 처리하는 IO보다 성능 저하가 발생할 수 있다.

#27일

- critical section을 보호하기 위한 방법 : lock 변수, 세마포어 변수