1주차

진행 상태	시작 전
완료 확인	☐

▼ 01장_디자인 패턴과 프로그래밍 패러다임

1.1 디자인패턴

1.1.1 싱글톤 패턴

- 하나의 클래스에 오직 하나의 인스턴스만 가지는패턴
  - 장점: 하나의 인스턴스를 만들어 놓고 해당 인스턴스를 다른 모듈들이 공유하며 사용하기 때문에 인스턴스를 생성할 때 드는 비용이 줄어듬
  - 단점: 의존성이 높음
- 싱글톤 패턴 단점
  - 사용하기 쉽고 실용적 but 모듈 간의 결합을 강하게 만들 수 있다는 단점 존재

    ⇒ 의존성 주입으로 해결 가능

> ❓ 의존성이란?
>
> - 의존성, 종송석을 의미
> - 코드에서 두 모듈(클래스)간의 연결 혹은 관계를 말한다.
> - 클래스 A가 다른 클래스 B를 이용할 때 A가 B에 의존한다고 한다. 이런 관계에서 A는 B없이 작동할 수 없다.
> - 의존 대상 B가 변하면, 그것이 A에 영향을 미친다.
>
> 단점
>
> - 모듈들이 더 분리되므로 클래스 수가 늘어나 복잡성이 증가될 수 있고
>
> 런타임 패널티 생김

1.1.2 팩토리 패턴

- 객체를 사용하는 코드에서 객체 생성 부분을 떼어내 추상화한 패턴
- 상속 관계에 있는 두 클래스에서 상위 클래스: 중요한 뼈대 결정 / 하위 클래스: 객체 생성에 관한 구체적인내용 결정
- 상위 , 하위 클래스 분리 되기 때문에 느슨한 결합

1.1.3 전략 패턴

- 전략 패턴 = 정책 패턴
- 객체의 행위를 바꾸고 싶은 경우 '직접' 수정하지 않고 전략이라고 부르는'캡슐화한 알고리즘' ⇒ 컨텍스트 안에서 바꿔주면서 상호 교체 가능하게 만드는 패턴

> ❓ 컨텍스트?
>
> 상황, 맥락, 문맥 의미
>
> 개발자가 어떠한 작업을 완료하는 데 필요한 모든 관련 정보

- 전략 패턴 라이브러리
  - passport
    - Node.js 에서 인증 모듈을 구현할 때 쓰는 미들웨어 라이브러리

        - 여러 가지 '전략'을 기반으로 인증

1.1.4 옵저버 패턴

- 주체가 어떤 객체(subject)의 상태 변화를 관찰하다가 상태 변화가 있을 때마다 메서드 등을 통해 옵저버 목록에 있는 옵저버들에게 변화를 알려줌
    - 주체?
        - 객체의 상태 변화를 보고 있는 관찰자
    - 옵저버?
        - 객체의 상태 변화에 따라 전달되는 메서드 등을 기반으로 '추가 변화 사항"이 생기는 개체들
    - 트위터?
        - 주체와 객체를 따로 두지 않고 상태각 변경되는 객체를 기반으로 구축
    - MVC 패턴 사용
        - 주체 = 모델　　⇒ 뷰 ⇒　　컨드롤러

            update()　　(옵저버)
- 자바 스크립트에서 옵저버 패턴
    - 프록시 객체
        - 어떠한 대상의 기본적인 동작의 작업을 가로챌 수 있는 객체를ㄹ뜻
            - target: 프록시할 대상
            - handler: 프록시 객체의 target 동작을 가로채서 정의할 동작들이 정해져 있는 함수

- 자바 : 상속과 구현
    - 상속(extends)
        - 자식 클래스가 부모 클래스의 메서드 등을 상속 받아 사용
        - 자식 클래스에서 추가 및 확장 가능
        - 재사용성 , 중복성의 최소화
    - 구현(implements)
        - 부모 인터페이스를 자식 클래스에서 재정의하여 구현
        - 반드시 부모클래스의 메서드를 재정의하여 구현

1.1.5 프록시 패턴과 프록시 서버

- **프록시 패턴**
    - 대상 객체에 접근하기 전 그 접근에 대한 흐름을 가로채 대상 객체 앞단의 인터페이스 역할을 하는 디자인 패턴 ⇒ 객체의 속성, 변환 등을 보완하며 보안, 데이터 검증, 캐싱, 로깅에 사용
- **프록시 서버**
    - 서버와 클라이언트 사이에서 클라이언트가 자신을 통해 다른 네트워크 서비스에 간접적으로 접속할 수 있게 해주는 컴퓨터 시스템이나 응용 프로그램
        - 프록시 서버로 쓰는 ngnix
            - 비동기 이벤트 기반의 구조와 다수의 연결을 효과적으로 처리 가능한 웹 서버

❓
- 버퍼 오버플로우?
  - 버퍼 = 데이터가 저장되는 메모리 공간
  - 메모리 공간을 벗어나는 경우
- gzip 압축
  - LZ77과 Huffman 코딩의 조합인 DEFLATE 알고리즘을 리반으로 한 압축 기술
  - 이를 이용하면 데이터 전송량 줄일 수 있지만, 압축을 해제했을 때 서버에서의 CPU 오버헤드도 생각해서 gzip 압축 사용 유무를 결정

- 프록시 서버로 쓰는 CloudFlare
  - 전 세계적으로 분산된 서버가 있고 이를 통해 어떠한 시스템의 콘텐츠 전달을 빠르게 할 수 있는CDN 서비스

❓ CDN?

Content Delivery Network

- 각 사용자가 인터넷에 접속하는 곳과 가까운 곳에서 콘텐츠를 캐싱 또는 배포하는 서버 네트워크를 뜻함
- 사용자가 웹 서버로부터 콘텐츠를 다운로드하는 시간을 줄일 수 있음

- DDOS 공격 방어
  - 짧은 기간 동안 네트워크에 많은 요청을 보내 네트워크를 마비시켜 웹 사이트의 가용성을 방해하는 사이버 공격 유형
  - 의심스러운 트래픽, 사용자가 접속하는 것이 아닌 시스템을 통해 오는 트래픽을 자동으로 차단해서 DDOS 공격으로부터 보호
- HTTPS 구축
  - 서버에서 HTTPS를 구축할 때 인증서를 기반으로 구축할 수도 있음
  - CloudFalre를 사용하면 별도의 인증서 설치 없이 구축 쉽게 가능

- CROS와 프론트엔드의 프록시 서버
  - 서버가 웹 브라우저에서 리소스를 로드할 때 다른 오리진을 통해 로드하지 못하게 하는 HTTP 헤더 기반 메커니즘

❓ 오리진?

- 프로토콜과 호스트 이름, 포트의 조합

ex) https:/naver.com:12020/test 에서

————————————-→ 이부분을 뜻함

1.1.6 이터레이터 패턴

- iterator를 사용하여 컬렉션의 요소들에 접근하는 디자인 패턴

❓ 이터레이터 프로토콜

- 이터러블한 객체들을 순화할 때 쓰이는 규칙

이터러블한 객체

- 반복 가능한 객체로 배열을 일반화한 객체

1.1.7 노출모듈 패턴

- 즉시 실행 함수를 통해 private, public 같은 접근 제어자를 만드는 패턴
- 자바스크립트는 private / public 같은 접근 제어자가 존재하지 않고 전역 범위에서 스크립트 실행
    - 그래서 노출모듈 패턴을 통해 private나 public 접근 제어자 구현

1.1.8 MVC 패턴

- 모델/ 뷰/ 컨트롤러 로 이루어진 디자인패턴
    - 장점: 재사용성과 확장성 용이
    - 단점: 애플리케이션이 복잡해질수록 모델과 뷰의 관계 복잡
- 모델
    - 애플리케이션의 데이터인 데이터베이스/상수/변수 뜻
    - 뷰에서 데이터를 생성하거나 수정하면 컨트롤러를 통해 모델을 생성하거나 갱신
- 뷰
    - 사용자 인터페이스 요소
    - 모델을 기반으로 사용자가 볼 수 있는 화면
- 컨트롤러
    - 하나 이상의 모델과 하나 이상의 뷰를 있는 다리 역할

1.1.9 MVP 패턴

- MVC 에서 C에 해당하는 컨트롤러가 프레젠터로 교체된 패턴
- 뷰와 프레젠터는 일대일 관계로 MVC 패턴보다 더 강한 결합

1.1.10 MVVM 패턴

- MVC의 C에 해당하는컨트롤러가 뷰모델로 바뀐 패턴
- 뷰모델 : 뷰를 더 추상화한 계층
- 커맨드와 데이터 바인딩을 가지는 것이 특징으로 뷰와 뷰모델 사이의 양방향 데이터바인딩을 지원
- UI를 별도의 코드 수정 없이 재사용할 수 있고 단위 테스팅하기 쉬움 ex) Vue.js

❓ 커맨드

- 여러 가지요소에 대한 처리를 하나의 액션으로 처리할 수 있게 하는 기법

데이터 바인딩

- 화면에 보이는 데이터와 웹 브라우저의 메모리 데이터를일치시키는 기법
- 뷰모델을 변경하면 뷰가 변경

1.2 프로그래밍 패러다임

: 프로그래머에게 프로그래밍의 관점을 갖게 해주는 역할을 하는 개발방법론

1.2.1 선언형과 함수형 프로그래밍

- 선언형 프로그래밍
    - ‘무엇을’ 풀어내는가에집중하는 패러다임
    - ‘프로그램은 함수로 이루어진 것이다’ 라는 명제가 담겨 있는 패러다임
- 함수형 프로그래밍
    - 선언형 패러다임의 일종
    - 작은 ‘순수 함수’들을 블록처럼 쌓아 로직을 구현하고 ‘고차 함수’를 통해 재사용성을 높인 프로그래밍 패러다임
        - 순수함수: 출력이 입력에만 의존하는 것

        - 고차함수: 함수가 함수를 값처럼 매개변수로 받아 로직을 생성할 수 있는 것

1.2.2 객체지향 프로그래밍

- 객체들의 집합으로 프로그램의 상호 작용을 표현하며 데이터를 객체로 취급하여 객체 내부에 선언된 메서드를 활용하는 방식
- 설계에 많은 시간 소요 / 처리 속도가 다른 프로그래밍 패러다임에 비해 상대적 느림
- 특징
    - 추상화
        - 복잡한 시스템으로 부터 핵심적인 개념 또는 기능을 간추려내는 것
    - 캡슐화
        - 객체의 속성과 메소드를 하나로 묶고 일부를 외부에 감추어 은닉하는 것
    - 상속성
        - 상위 클래스의 특성을 하위 클래스가 이어받아서 재사용하거나 추가 또는 확장
        - 코드의 재사용 , 계층적인 관계 생성 , 유지 보수성 측면에서 중요
    - 다형성
        - 하나의 메서드나 클래스가 다양한 방법으로 동작하는 것
        - ex) 오버로딩 / 오버라이딩
            - 오버로딩 : 같은 이름을 가진 메서드를 여러 개 두는 것(정적 다형성)
                - 메서드의 타입, 매개변수의 유형, 개수 등 여러 개를 둘 수 있음
            - 오버라이딩: 주로 메서드 오버라이딩을 말하며 상위 클래스로부터 상속받은 메서드를 하위 클래스가 ***재정의*** 하는 것을 의미 (동적 다형성)
- 설계 원칙(SOLID)
    - S(SRP) 단일 책임 원칙
        - 모든 클래스는 각각 하나의 책임만 가져야 하는 원칙
        - ex) A라는 로직이 존재하면 어떤 클래스는 A에 관한 클래스여야 하고 이를 수정해도 A와 관련된 수정
    - O(OCP) 개방-폐쇄 원칙
        - 유지 보수 사항이 생긴다면 코드를 쉽게 확장할 수 있도록 하고 수정할 때는 닫혀 있어야함
        - 기존의 코드는 잘 변경하지 않으면서 확장은 쉽게 가능
    - L(LSP) 리스코프 치환 원칙
        - 프로그램의 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 하는 것
        - 클래스는 상속이 되고, 부모 - 자식이라는 계층 관계가 만들어짐
            - 이때, 부모 객체이 자식 객체를 넣어도 시스템이 문제없이 돌아가게 만드는 것
    - I(ISP) 인터페이스 분리 원칙
        - 하나의 일반적인 인터페이스보다 구체적인 여러 개의 인터페이스를 만들어야 하는 원칙
    - D(DIP) 의존 역전 원칙
        - 자신보다 변하기 쉬운 것에 의존 하던 것을 추상화된 인터페이스 , 상위 클래스를 두어 변하기 쉬운 것의 변화에 영향받지 않게 하는 원칙
        - ex) 타이어 틀 만들고 타이어 교체 가능 ⇒ 상위 계층은 하위 계층의 변화에 독립

1.2.3 절차형 프로그래밍

- 로직이 수행되어야 할 연속적인 계산 과정으로 이루어짐
- 장점: 일이 진행되는 방식으로 그저 코드를 구현하기만 하면 되기 때문에 코드의 가독성이 좋고 실행 속도 빠름
- 단점: 모듈화하기가 어렵고 유지 보수성 떨어짐

1.2.4 패러다임의 혼합

- 비지니스 로직이나 서비스의 특징을 고려해서 패러다임 정하는 것이 좋음

▼ 예상 질문

▼ 옵저버 패턴을 어떻게 구현하나요?

▼ 프록시 서버를 설명하고 사용 사례에 대해 설명해보세요.

▼ MVC 패턴을 설명하고 MVVM 패턴과의 차이는 무엇인지 설명해보세요.

▼ 싱글톤 패턴이 무엇이고, 어떤 장단점이 있나요?

답변:싱글톤 패턴은 하나의 클래스에 오직 하나의 인스턴스만 가지는 디자인 패턴입니다.

- **장점**: 메모리 절약, 인스턴스 생성 비용 감소
- **단점**: 의존성이 높아져 결합도 증가
- **해결책**: 의존성 주입(DI)으로 결합도를 낮출 수 있습니다

실무 예시:
데이터베이스 연결 객체나 로거(Logger) 같은 경우에 싱글톤 패턴을 자주 사용합니다.

▼ MVC, MVP, MVVM 패턴의 차이점을 설명해주세요.

답변:
세 패턴 모두 애플리케이션의 관심사를 분리하는 아키텍처 패턴입니다.

- **MVC**: Model-View-Controller로 구성, Controller가 중재자 역할
- **MVP**: Controller 대신 Presenter 사용, View와 1:1 강한 결합
- **MVVM**: ViewModel이 데이터 바인딩으로 View와 양방향 통신, Vue.js가 대표적
- **핵심 차이**: 각 컴포넌트 간 통신 방식과 결합도가 다름

▼ 객체지향 프로그래밍의 SOLID 원칙에 대해 설명해주세요.

답변:
SOLID는 객체지향 설계의 5가지 원칙입니다.

- **S(단일책임)**: 클래스는 하나의 책임만
- **O(개방폐쇄)**: 확장엔 열려있고 수정엔 닫혀있게
- **L(리스코프)**: 자식이 부모를 완전히 대체 가능해야
- **I(인터페이스분리)**: 필요한 메서드만 구현하도록 인터페이스 분리
- **D(의존역전)**: 구체적 구현이 아닌 추상화에 의존

한 줄 요약: 유지보수와 확장이 쉬운 코드를 만들기 위한 5가지 설계 원칙입니다.

▼ 02장_네트워크

2.1 네트워크의 기초

- 네트워크: 노드와 링크가 서로 연결되어 있거나 연결되어 있지 않은 집합체
  - 노드: 서버 , 라우터 , 스위치 등 네트워크 장치를 의미
  - 링크: 유선 또는 무선

2.1.1 처리량과 지연 시간

- 처리량: 링크를 통해 전달되는 단위 = 시간당 데이터양 (bps 단위)
  - 트래픽, 네트워크 장치 간의 대역폭, 네트워크 중간에 발생하는 에러, 장치의 하드웨어 스펙 영향 받음
- 지연 시간: 요청이 처리되는 시간
  - 어떤 메시지가 두 장치 사이를 왕복하는 데 걸린 시간
  - 매체 타입(무선,유선), 패킷 크기, 라우터의 패킷 처리 시간 영향 받음

2.1.2 네트워크 토폴로지와 병목 현상

- 네트워크 토폴리지
  - 노드와 링크가 어떻게 배치되어 있는지에 대한 방식 & 연결 형태
- 트리 토폴로지
  - 계층형 토폴로지
  - 트리 형태로 배치한 네트워크 구성
  - 장점: 노드의 추가, 삭제 쉬움
  - 단점: 특정 노드에 트래픽이 집중될 때 하위 노드에 영향 끼칠 수 있음
- 버스 토폴로지
  - 중앙 통신 회선 하나에 여러 개의 노드가 연결되어 공유하는 네트워크 구성
  - 근거리 통신망(LAN)에서 사용
  - 장점: 설치 비용 적음 , 신뢰성 우수, 중앙 통신 회선에 노드 추가 및 삭제 쉬움
  - 단점: 스푸핑이 가능
    - 스푸핑?
      - LAN상에서 송신부의 패킷을 송신과 관련 없는 다른 호스트에 가지 않도록 하는 스위칭 기능을 마비 또는 속여서 특정 노드에 해당 패킷이 오도록 하는 것
- 스타 토폴로지
  - 중앙에 있는 노드에 모두 연결된 네트워크 구성
  - 장점
    - 노드를 추가하기 쉬움
    - 에러 탐지하기 쉬움
    - 패킷의 충동 발생 가능성 적음
    - 어떠한 노드에 장애가 발생해도 쉽게 에러 발견
    - 장애 노드가 중앙 노드가 아닐 경우 다른 노드에 영향 끼치는 것이 적음
  - 단점
    - 중앙 노드에 장애 발생 ⇒ 전체 네트워크 사용 불가
    - 설치 비용 고가
- 링형 토폴로지
  - 각각의 노드가 양 옆의 두 노드와 연결하여 전체적으로 고리처럼 하나의 연속된 길을 통해 통신을 하는 망 구성 방식
  - 데이터는 노드→ 노드 이동
  - 각각의 노드는 고리 모양의 길을 통해 패킷 처리
  - 장점
    - 노드 수가 증가 → 네트워크 손실 거의 없고 충돌이 발생되는 가능성 적음
    - 노드의 고장 발견 빠르게 찾음
  - 단점
    - 네트워크 구성 변경이 어려움
    - 회선에 장애 발생 ⇒ 전체 네트워크에 영행 끼침
- 메시 토폴로지
  - 망형 토폴로지
  - 그물망처럼 연결되어 있는 구조
  - 장점
    - 한 단말 장치에 장애 발생해도 여러 개의 경로 존재 ⇒ 네트워크 계속 사용 가능 및 트래픽 분산 처리 가능
  - 단점

        - 노드의 추가 어렵
        - 구축 비용과 운용 비용 고가
- 병목 현상
    - 전체 시스템의 성능이나 용량이 하나의 구성 요소로 인해 제한을 받는 현상
        - (네트워크 구조) 토폴리지 중요한 이유 ⇒ 병목 현상을 찾을 때 중요한 기준임
        - 네트워크가 어떤 토폴로지를 갖는지 / 어떠한 경로로 이루어져 있는지 알아야 병목 현상 해결 가능

2.1.3 네트워크 분류

- LAN < MAN < WAN (by 네트워크 규모)
- LAN
    - 근거리 통신망
    - 같은 건물이나 캠퍼스 같은좁은 공간에서 운영
    - 전송 속도 빠르고, 혼잡 x
- MAN
    - 대도시 지역 네트워크
    - 전송 속도 평균, LAN 보다 혼잡
- WAN
    - 광역 네트워크 (국가, 대륙)
    - 전송 속도 낮음, MAN 보다 혼잡

2.1.4 네트워크 성능 분석 명령어

- 병목 현상 원인
    - 네트워크 대역폭
    - 네트워크 토폴로지
    - 서버 CPU, 메모리 사용량
    - 비효율적인 네트워크 구성
- 네트워크로부터 발생한 문제점 ⇒ 네트워크 성능 분석 명령어
    - ping
        - 네트워크 상태를 확인하려는 대상 노드를 향해 일정 크기의 패킷을 전송하는 명령어
        - TCP/IP 프로토콜 중에 ICMP 프로토콜을 통해 동작
        - ICMP 프로토콜 지원하지 않는 기기를 대상으로 실행 x
    - netstat
        - 접속되어 있는 서비스들의 네트워크 상태 표시
        - 네트워크 접속, 라우팅 테이블, 네트워크 프로토콜 리스트 보여줌
        - 서비스의 포트가 열려 있는지 확인
    - nslookup
        - DNS 관련된 내용 확인
        - 특정 도메인에 매핑된 ip 확인
    - tracert
        - 윈도우 tracert / 리눅스 traceroute
        - 목적지 노드까지 네트워크 경로를 확인할 때 사용하는 명령어
        - 목적지 노드까지 구간들 중 어느 구간에서 응답 시간이 느려지는지 확인

2.1.5 네트워크 프로토콜 표준화

- 다른 장치들끼리 데이터를 주고받기 위해 설정된 공통된 인터페이스
- 기업이나 개인이 발표해서 정하는 것이 아니라 IEEE 또는 IETF 라는 표준화 단체가 정함

2.2 TCP/IP 4계층 모델

2.2.1 계층 구조

- **애플리케이션 계층**
    - 웹 서비스 , 이메일 등 서비스를 실질적으로 사람들에게 제공하는 층
        - FTP
        - SSH
        - HTTP
        - SMTP
        - DNS
- **전송 계층**
    - 송신자와 수신자를 연결하는 통신 서비스 제공
    - 연결 지향 데이터 스트림 지원, 신뢰성, 흐름 제어 제공
    - 애플리케이션과 인터넷 계층 사이의 데이터가 전달될 때의 중계 역할
        - TCP
            - 가상회선 패킷 교환방식
                - 각 패킷에는 가상회선 식별자가 포함되며 모든 패킷을 전송하면 가상회선 해제되고
                - 패킷들은 전송된 '순서대로' 도착하는 방식
            - TCP 연결 성립 과정
                - 3-웨이 핸드셰이크 작업 진행 ⇒ TCP는 이 과정이 있기 때문에 신뢰성 있는 계층 / UDP는 없음

                1. **SYN 단계 (연결 요청)**
                    - 클라이언트가 서버에 클라이언트의 ISN(Initial Sequence Number)을 담아 SYN을 보냄
                    - ISN은 새로운 TCP 연결의 첫 번째 패킷에 할당된 임의의 시퀀스 번호
                2. **SYN + ACK 단계 (응답)**
                    - 서버가 클라이언트의 SYN을 수신
                    - 서버의 ISN을 보내며, 승인번호로 클라이언트의 ISN + 1을 보냄
                3. **ACK 단계 (확인)**
                    - 클라이언트가 서버의 ISN + 1한 값인 승인번호를 담아 ACK를 서버에 보냄

                결과: 3-웨이 핸드셰이크 과정 완료 후 신뢰성 구축 → 데이터 전송 시작
            - TCP 연결 해제 과정
                - 4-웨이 핸드셰이크 과정 발생

                1. **FIN 전송**
                    - 클라이언트가 연결을 닫으려고 할 때 FIN 세그먼트 보냄
                    - 클라이언트는 FIN_WAIT_1 상태로 전환
                2. **ACK 응답**
                    - 서버가 클라이언트로 ACK 승인 세그먼트 보냄
                    - 서버는 CLOSE_WAIT 상태, 클라이언트는 FIN_WAIT_2 상태
                3. **FIN 전송**
                    - 서버가 일정 시간 후 클라이언트에 FIN 세그먼트 보냄
                4. **최종 ACK**

                - 클라이언트가 TIME_WAIT 상태가 되고 서버로 ACK 보냄
                - 서버는 CLOSED 상태, 클라이언트는 일정 시간 대기 후 연결 해제

            TIME_WAIT의 필요성:

            - 지연 패킷 처리를 위해 (데이터 무결성 보장)
            - 양쪽 장치의 연결 종료 확인을 위해
        - UDP
            - 데이터그램 패킷 교환방식
                - 패킷이 독립적으로 이동하며 최적의 경로를 선택하여 가는데
                - 하나의 메시지에서 분할된 여러 패킷은 서로 다른 경로로 전송될 수 있고
                - 도착한 '순서가 다를 수' 있는 방식
- 인터넷 계층
    - 기능: 장치로부터 받은 네트워크 패킷을 IP 주소로 지정된 목적지로 전송
    - 주요 프로토콜: IP, ARP, ICMP 등
    - 특징:
        - 패킷을 수신해야 할 상대의 주소를 지정하여 데이터 전달
        - 상대방이 제대로 받았는지 보장하지 않는 비연결형적 특징
- 링크 계층
    - 기능: 실질적으로 데이터를 전달하며 장치 간 신호를 주고받는 '규칙'을 정하는 계층
    - 별칭: 네트워크 접근 계층
    - 세부 분류:
        - 물리 계층: 무선/유선 LAN을 통해 0과 1로 이루어진 데이터 전송
        - 데이터 링크 계층: 이더넷 프레임을 통해 에러 확인, 흐름 제어, 접근 제어
            - 유선 LAN (IEEE802.3)
                - 이더넷 프로토콜 사용
                - 전이중화 통신: 양쪽 장치가 동시에 송수신 가능 (송신로와 수신로 분리)
                - 케이블 종류:
                    - 트위스트 페어 케이블: 8개 구리선을 2개씩 꼬아서 묶은 케이블 (UTP, STP)
                    - 광섬유 케이블: 레이저 이용, 장거리 고속 통신 가능 (100Gbps)
            - 무선 LAN (IEEE802.11)
                - 반이중화 통신: 수신과 송신에 같은 채널 사용, 한 번에 한 방향만 통신
                - CSMA/CA: 데이터 전송 전 충돌 방지 메커니즘
                    1. 무선 매체 상태 확인
                    2. 캐리어 감지로 회선 비어있는지 판단
                    3. IFS(랜덤 값 기반 대기시간) 적용
                    4. 데이터 송신
                - 주파수 대역: 2.4GHz (장애물에 강하지만 간섭 多) / 5GHz (깨끗한 전파환경)
- 와이파이 구조
    - **BSS (Basic Service Set)**: 단일 AP 기반, 이동성 제한
    - **ESS (Extended Service Set)**: 여러 BSS 연결, 중단 없는 이동성 제공
- 이더넷 프레임 구조
    - **Preamble**: 프레임 시작 알림

        - **SFD**: MAC 주소 필드 시작 알림
        - **DMAC/SMAC**: 수신/송신 MAC 주소
        - **EtherType**: 상위 계층 IP 프로토콜 정의
        - **Payload**: 전달받은 데이터
        - **CRC**: 에러 확인 비트
- 계층 간 데이터 송수신 과정
    - 캡슐화 과정 (Encapsulation)

      상위 계층 → 하위 계층으로 데이터 전달 시 헤더 추가

      1. **애플리케이션 계층**: 메시지
      2. **전송 계층**: 세그먼트/데이터그램 (TCP/UDP 헤더 추가)
      3. **인터넷 계층**: 패킷 (IP 헤더 추가)
      4. **링크 계층**: 프레임 (프레임 헤더와 트레일러 추가)
    - 비캡슐화 과정 (Decapsulation)

      하위 계층 → 상위 계층으로 데이터 전달 시 헤더 제거

      1. **링크 계층**: 프레임 헤더/트레일러 제거 → 패킷
      2. **인터넷 계층**: IP 헤더 제거 → 세그먼트/데이터그램
      3. **전송 계층**: TCP/UDP 헤더 제거 → 메시지
      4. **애플리케이션 계층**: 최종 메시지를 사용자에게 전달

      결과: 송신 측에서 캡슐화된 데이터가 수신 측에서 비캡슐화되어 원본 메시지로 복원

2.2.2 PDU

- **PDU**: 네트워크의 어떤 계층에서 다른 계층으로 데이터가 전달될 때의 한 덩어리 단위
- **구성**: 제어 관련 정보가 포함된 '헤더' + 데이터를 의미하는 '페이로드'
- 계층별 PDU 명칭
    - **애플리케이션 계층**: 메시지
    - **전송 계층**: 세그먼트(TCP) / 데이터그램(UDP)
    - **인터넷 계층**: 패킷
    - **링크 계층**: 프레임(데이터 링크 계층) / 비트(물리 계층)

2.3 네트워크 기기

2.3.1 네트워크 기기의 처리 범위

- **상위 계층 기기는 하위 계층 처리 가능** (역은 불가능)
- **계층별 기기**:
    - 애플리케이션 계층: L7 스위치
    - 인터넷 계층: 라우터, L3 스위치
    - 데이터 링크 계층: L2 스위치, 브리지
    - 물리 계층: NIC, 리피터, AP

2.3.2 애플리케이션 계층을 처리하는 기기

- **L7 스위치 (로드밸런서)**
    - **기능**: 서버 부하 분산, URL/서버/캐시/쿠키 기반 트래픽 분산
    - **필터링**: 바이러스, 불필요한 외부 데이터 차단
    - **헬스 체크**: 정상/비정상 서버 판별

- **L4 vs L7 스위치**
  - L4: IP와 포트 기반 분산
  - L7: URL, HTTP 헤더, 쿠키 기반 분산
- **서버 이중화**: 가상 IP로 2대 이상 서버 관리하여 안정적 서비스 제공

2.3.3 인터넷 계층을 처리하는 기기

- **라우터**
  - **기능**: 여러 네트워크 연결/분할/구분
  - **라우팅**: 최적 경로로 패킷 포워딩
- **L3 스위치**
  - L2 스위치 기능 + 라우팅 기능
  - 하드웨어 기반 라우팅 (소프트웨어 기반 라우터보다 빠름)

2.3.4 데이터 링크 계층을 처리하는 기기

- **L2 스위치**
  - **기능**: MAC 주소 테이블 관리, MAC 주소 기반 스위칭
  - IP 주소 이해 불가, 단순 MAC 주소 기반 패킷 전달
- **브리지**
  - **기능**: 두 개의 LAN 상호 접속
  - 통신망 범위 확장, 서로 다른 LAN을 하나로 통합

2.3.5 물리 계층을 처리하는 기기

- **NIC (Network Interface Card)**
  - **기능**: 네트워크 연결용 확장 카드
  - 고유한 MAC 주소 보유
- **리피터**
  - **기능**: 약해진 신호 증폭하여 전달
  - 광케이블 보급으로 현재는 거의 사용 안함
- **AP (Access Point)**
  - **기능**: 유선 LAN을 무선 LAN으로 변환
  - 패킷 복사 기능

2.4 IP 주소

2.4.1 ARP

- **ARP (Address Resolution Protocol)**
  - **기능**: IP 주소 → MAC 주소 변환
  - **과정**:
    1. ARP Request 브로드캐스트로 IP에 해당하는 MAC 주소 요청
    2. 해당 장치가 ARP Reply 유니캐스트로 MAC 주소 응답
- **RARP**: MAC 주소 → IP 주소 변환

2.4.2 홉바이홉 통신

- **홉바이홉 통신**: 패킷이 여러 라우터를 건너가며 최종 목적지까지 전달
- **라우팅 테이블**: 목적지 정보와 다음 라우터 정보가 저장된 리스트
- **게이트웨이**: 서로 다른 네트워크/프로토콜 간 통신을 가능하게 하는 관문

2.4.3 IP 주소 체계

- **IPv4 vs IPv6**
  - IPv4: 32비트, 8비트씩 점으로 구분 (123.45.67.89)
  - IPv6: 64비트, 16비트씩 콜론으로 구분
- **클래스 기반 할당 방식**
  - A, B, C, D, E 클래스로 구분
  - **문제점**: 사용하는 주소보다 버리는 주소가 많음
- **해결 방안**:
  - **DHCP**: IP 주소 자동 할당
  - **NAT**: 사설 IP ↔ 공인 IP 변환
    - **장점**: 주소 절약, 내부 네트워크 보안
    - **단점**: 접속 속도 저하 가능

2.4.4 IP 주소를 이용한 위치 정보

- IP 주소를 통해 동/구 단위까지 위치 추적 가능

2.5 HTTP

2.5.1 HTTP/1.0

- **특징**: 한 연결당 하나의 요청 처리
- **문제점**: RTT 증가 (매번 TCP 3-웨이 핸드셰이크 필요)

> ❓ RTT?
>
> 패킷이 목적지에 도달하고 나서 다시 출발지로 돌아오기까지 걸리는 시간 & 패킷 왕복 시간

- **해결 방안**:
  - **이미지 스플리팅**: 여러 이미지를 하나로 합쳐 다운로드 횟수 감소
  - **코드 압축**: 개행문자, 빈칸 제거로 코드 크기 최소화
  - **Base64 인코딩**: 이미지를 문자열로 변환 (37% 크기 증가하지만 HTTP 요청 불필요)

2.5.2 HTTP/1.1

- **개선점**: keep-alive 옵션으로 한 번의 TCP 연결로 여러 파일 송수신
- **문제점**:
  - **HOL Blocking**: 첫 번째 패킷 지연 시 뒤의 패킷들도 대기
  - **무거운 헤더**: 쿠키 등 메타데이터로 인한 압축되지 않는 헤더

2.5.3 HTTP/2

- **핵심 기능**:
  - **멀티플렉싱**: 여러 스트림으로 병렬 송수신 → HOL Blocking 해결

> ❓ 스트림?
>
> 시간이 지남에 따라 사용할 수 있게 되는일련의 데이터 요소를 가리키는 데이터 흐름

  - **헤더 압축**: HPACK(허프만 코딩) 압축 사용
  - **서버 푸시**: 클라이언트 요청 없이 서버가 리소스 전송

- **허프만 코딩**: 빈도 높은 정보는 적은 비트, 빈도 낮은 정보는 많은 비트 사용

2.5.4 HTTPS

- **SSL/TLS**: 애플리케이션과 전송 계층 사이의 신뢰 계층
- **TLS 핸드셰이크 (1-RTT)**:
  1. **사이퍼 슈트**: 프로토콜 + AEAD 사이퍼 모드 + 해싱 알고리즘
  2. **인증 메커니즘**: CA에서 발급한 인증서로 서버 신뢰성 검증
  3. **키 교환**: 디피-헬만 알고리즘으로 암호키 교환
  4. **해싱**: SHA-256 등으로 데이터 무결성 보장

> ❓ 해시?
>
> - 다양한 길이를 가진 데이터를 고정된 길이를 가진 데이터로 매핑한 값
>
> 해싱?
>
> - 임의의 데이터를 해시로 바꿔주는 일이며 해시 함수가 이를 담당
>
> 해시 함수?
>
> - 임의의 데이터를 입력으로 받아 일정한 길이의 데이터로 바꿔주는 함수

- **디피-헬만 키 교환**: 공개값 공유 → 각자 비밀값과 혼합 → 공통 암호키 생성
- **SEO 이점**:
  - 구글 검색 순위 우대
  - **필수 요소**: 캐노니컬 설정, 메타 설정, 페이지 속도 개선, 사이트맵 관리

2.5.5 HTTP/3

- **QUIC 프로토콜**: UDP 기반 (기존 TCP 대신)
- **장점**:
  - **초기 연결**: 1-RTT (3-웨이 핸드셰이크 불필요)
  - **FEC (순방향 오류 수정)**: 패킷 손실 시 수신 측에서 에러 검출/수정
  - **멀티플렉싱**: HTTP/2의 장점 유지

▼ 예상 질문

▼ OSI 7계층과 TCP/IP 4계층의 차이점은 무엇인가요?

▼ HTTP/2를 설명하고 장점 두 가지를 설명하세요.

▼ www.naver.com을 주소창에 입력하면 어떻게 될까요?

▼ HTTP와 HTTPS의 차이점을 설명하시오

- HTTP는 웹에서 데이터를 주고받는 기본 프로토콜이고, HTTPS는 HTTP + SSL/TLS 보안 계층이 추가된 프로토콜

▼ 쿠키와 세션의 차이점을 설명하시오.

- 쿠키는 클라이언트(브라우저)에 저장되는 데이터 세션은 서버에 저장되는 데이터
  - 세션의 단점: 서버 비용 발생

▼ 로드 밸런서란 무엇이고, L4와 L7 로드 밸런서의 차이점을 설명해주세요.

- 로드 밸런서는 여러 서버에 트래픽을 분산시켜 부하를 균등하게 분배하는 장비로

  L4 로드 밸런서는 IP주소와 포트번호 기반으로 트래픽 분산시키고

  L7 로드 밸런서는 HTTP 헤더, URL, 쿠키 등 애플리케이션 레벨 정보로 분산시킵니다.