- **화이트 박스 테스트 커버리지**
    - 설계 절차에 조첨. 구조적 테스트
    - 테스트 과정 초기

    1. 기초 경로 검사
        - 대표적 기법
        - 논리적 복잡성 측정
        - 실행 경로 기초 정의
    2. 제어 구조 검사
        - 조건 검사 condition testing : 논리적 조건 초점
        - 루프 검사 loop testing : 반복 구조 초점
        - 데이터 흐름검사 data flow testing : 변수 정의, 사용 위치에 초점

    - 검증 기준 (코드 커버리지)
      [코드 커버리지 Code Coverage 종류와 예시 코드 커버리지(Code Coverage)]
        1. 문장 검증 statement : 모든 구문 한 번 이상 수행
        2. 분기검증 branch (결정 decision) : 모든 조건문 True, False 한 번 이상 수행
        3. 조건 검증 condition : 모든 조건문의 내부 조건식 True, False 한번 이상 수행
        4. 분기/조건 검증 : 분기를 고려하지 않는 조건 검증 개선. 모든 조건문 한번 이상 + 각 조건문의 조건 식 결과 True, False 한번 이상 수행
        5. 변형 분기/조건 검증(MC/DC) : 모든 결정 포인트 내 전체 조건식은
        6. 다중 조건 검증 : 모든 개별 조건식의 논리적 조합

    - 검증 기준 종류
        1. 기능 기반 커버리지
        2. 라인 커버리지 : 수행한 코드 라인 수
        3. 코드 커버리지 : 코드 자체 테스트

- **객체 지향 방법론**
    - 럼바우 분석 Rumbaugh
        - 소프트웨어 구성 요소를 그래픽 표기법 이용해 모델링
        - 객체모델링기법 OMT 라고도 함
        - 객체 / 동적 / 기능 모델링
            1. 객체 모델링 Object M
                - 객체 다이어그램으로 표시

2. 동적 모델링 Dynamic M
    - 상태 다이어그램으로 시간 흐름에 따라 객체간 동적 행위 표현
3. 기능 모델링 Functional M
    - 자료 흐름도 (DFD)로 다수 프로세스 간 자료 흐름 중심으로 처리
    - 어떤 데이터 입력해 어떤 결과 구할까

- 부치 분석 Booch
    - 미시적, 거시적 개발 프로세스 모두 사용
    - 클래스 객체 분석 식별해 클래스 속성과 연산 정의
- Jacobson 분석
    - 유스케이스 사용해 분석
    - 사용자, 외부 시스템, 타 요소들이 시스템과 상호작용하는 방법 기술
- Coad와 Yourdon 분석
    - E-R 다이어그램으로 객체 행위 모델링
    - 객체 식별, 구조 식별, 주제 정의, 속성, 인스턴스 연결 정의 등 관계 분석
- Wirfs-Brock 분석
    - 분석 설계간 구분 없음
    - 고객 명세서 평가해 설계 작업까지 연속적으로 수행

- **객체 지향 기법 관계성 종류**
    - is member of 연관성 Association
    - part-whole, is part of 집단화 aggregation
    - is a 일반화 Generalization, 특수화 Specialization

- **버퍼 오버플로우 대응 방안**
    - 스택가드 : 카나리(무결성 체크용 값) 복귀 주소와 변수 사이에 삽입. 버퍼 오버플로우 발생 시 카나리 체크해 변할 경우 복귀 주소 호출 안 함
    - 스택쉴드 : 함수 시작 시 복귀 주소를 Global RET(특수 스택) 에 저장. 함수 종료시 저장된 값과 RET값 비교해 다를 경우 프로그램 중단
    - ASLR : 주소 공간 배치를 난수화. 실행 시 메모리 주소 변경. 특정 주소 호출 못함. Address Space Layout Randomization

- CASE
    - 소프트웨어공학 작업 자동화 도구
        - 부품 재사용성 향상
        - 품질 일관성 효율적 관리

        - 생명 주기 연결 자동화
        - 유지보수 용이
        - 문서화 명세화그래픽
        - 다이어그램 작성
    - 주요 기능
        - 생명 주기 연결
        - 모델 간 모순 검사
        - 모델 오류 검증
        - 자료흐름도 등 다이어그램 작성
        - 다양한 SW 개발 모형 지원
        - 문서화 명세화 그래픽 지원
    - 원천 기술
        - 구조적 기법
        - 프로토타이핑 기술
        - 자동프로그래밍 기술
        - 정보 저장소기술
        - 분산 처리 기술
    - 종류
        - SADT : softtech사. 구조적 요구 분석. 블록다이어그램
        - SREM : trw사. RSL, REVS 이용
        - PSL/PSA : 미시간 대학
        - TAGS : IORL 언어. SDLC 전 과정

- **소프트웨어 생명주기 모델**
    - 구조적 Structured: 프로세스 중심. 하향식. 나씨 슈나이더만 차트
    - 정보공학 Information Engineering : 정보시스템 개발 절차 작업 기법 체계화. 개발 주기 이용 대형 프로젝트 수행.
    - 객체지향 Object-Oriented : 객체 기본 단위로 시스템 분석 설계
    - 컴포넌트기반 CBD Component Based : 소프트웨어 구성하는 컴포넌트 조립. 개발기간 단축(생산성). 기능 추가 쉬움(확장성). 재사용 가능.
    - 애자일 Agile: 절차보다 사람. 변화 우연 신속. 효율적. 경량
    - 제품 계열 Product Line : 특정 제품에 적용할 공통기능 정의

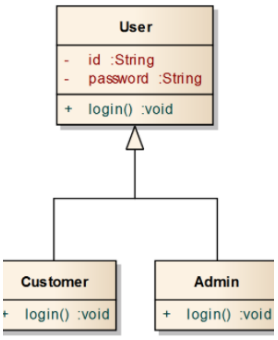
- **소프트웨어 설계 유형**
    - 상위 설계 : 자료 구조 / 아키텍처(예비설계. 시스템 전체 구조. 컴포넌트 간 관계1) / 인터페이스(소프트웨어와 상호작용하는 컴퓨터 시스템, 사용자가 어떻게

통신하는지) / 프로시저 (프로그램 아키텍처의 컴포넌트를 소프트웨어 컴포넌트의 프로시저 서술로 변환)

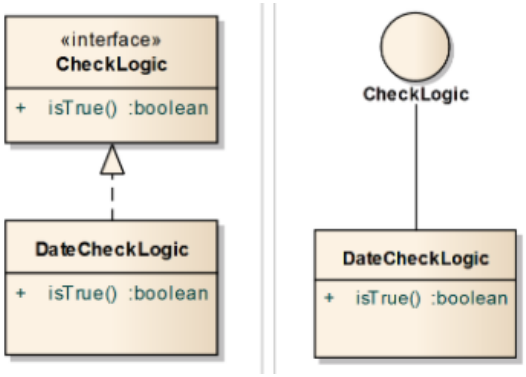
- 하위 설계 : 모듈

- **소프트웨어 아키텍처 패턴**
  - 계층화 패턴 Layered : 시스템 계층으로 구분. 하위 모듈은 특정 수준 추상화 제공. 각 계층은 다음 상위 계층에 서비스 제공. 마주보는 계층만 상호작용
  - 클라이언트 서버 : 하나의 서버 다수의 클라이언트
  - 파이프 필터 : 데이터 스트림 생성 처리. 서브 시스템이 입력 데이터 받아 처리. 결과를 다음 서브 시스템으로 넘기는 과정 반복
  - 브로커 : 분리 컴포넌트로 구성된 분산 시스템. 컴포넌트는 원격 서비스 실행해 상호작용. 컴포넌트 간 통신 조정. 서버는 자신의기능을 브로커에게 넘김. 클라이언트는 브로커에 서비스 요청. 브로커는 클라이언트를 자기 레지스트리에 있는 서비스로 리다이렉션. Apache Kaf-ka, JBoss Messaging
  - 모델 뷰 컨트롤러 : 모델(핵심 기능, 데이터 보관), 뷰 (사용자에게정보 표시), 컨트롤러 (사용자에게 요청 입력받아 처리)

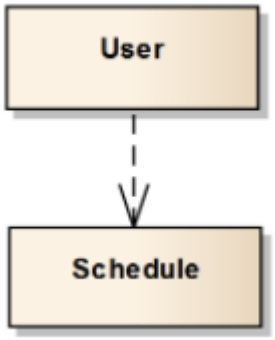
- **UML 관계**
  - 일반화 Generalization
    부모 클래스와 자식 클래스의 상속관계
    자식 클래스가 주체. 자식 클래스를 부모 클래스로 일반화.
    하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지 표현
    (반대의 경우, 부모 클래스를 자식 클래스로 구체화 Specialize)



  - 실체화 Realization
    인터페이스 메서드를 실제 기능으로 구현하는 것
    사물이 할 수 있거나 해야 하는 기능(행위, 인터페이스)으로 서로를 그룹화



- 의존 Dependency
  제일 많이 사용
  어떤 클래스가 다른 클래스를 참조
  해당 객체의 참조를 계속 유지하지 않음
  사물 사이에 서로 연관은 있으나 필요에 의해 서로 영향을 줌. 짧은 시간 동안 관계 유지.
  실체



유저 클래스에서 스케쥴 객체 생성

- 연관 Association, 방향성 있는 연관 Directed Association
  보통 다른 객체 참조를 가지는 필드… ?
  2개 이상의 사물이 서로 관련되어 있음. 실선과 화살표

- 집합 Aggregation
  전체와 부분의 관계
  하나의 사물이 다른 사물에 포함되어 있음
  전체 쪽에 빈 다이아몬드 표기
  전체와 부분은 독립적
  전체가 부분을 빌려 씀

- 합성, 포함 Composition
  전체와 부분 집합을 나타내는데 위의 집합보다 더 강한 집합
  포함하는 사물의 변화가 포함되는 사물에게 영향을 미침
  종속적. 부분이 집합의 소유

- **디자인 패턴**
  - 생성
    Builder / Prototype / Factory Method / Abstract Factory / Singleton
  - 구조
    Bridge / Decorator / Facade / Flyweight / Proxy / Composite / Adaptor
  - 행위
    Interpreter / Template Method / Chain of Responsibility / Command / Iterator / Mediator / Observer / State / Strategy / Visitor

- **정적 / 동적 다이어그램 (럼바우 객체 지향 분석 기법)**
  - 정적
    클래스, 객체, 컴포넌트, 배치(결과물, 프로세스, 컴포넌트 등 물리 요소들 위치 표현), 복합체 구조(Composite Structure. 복잡한 구조 클래스 컴포넌트 내부 구조 표현), 패키지(유스케이스, 클래스 등 모델 요소들 그룹화한 패키지 관계 표현)
  - 행위
    유스케이스(사용자 요구 분석해 기능 모델링에 사용), 시퀀스(상호작용하는 시스템,객체들이 주고받는 메시지 표현), 커뮤니케이션(메시지 뿐 아니라 객체들 연관까지 표현), 상태(한 객체가 속한 클래스의 상태 변화 또는 다른 객체와의 상호작용에 따라 어떻게 변하는지), 활동(객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현), 타이밍(객체 상태 변화와 시간 제약 명시적 표현)

- **정규화 단계**
  - 1. 원자화
  - 2. 부분 함수 종속 제거
  - 3. 이행 함수 종속 제거

- BCNF. 결정자 함수 종속 제거
- 4. 다치 종속성 제거
- 5. 조인 종속성 제거

- **프로세스 상태**
  - 생성 Create, 준비 Ready, 실행 Running, 대기 waiting, 완료 Complete

- **스케줄링 알고리즘**
  - 선점형
    라운드 로빈 / SRT Shortest Remaining Time First / 다단계 큐 / 다단계 피드백 큐
  - 비선점
    우선순위, 기한부, FCFS, HRN High Response Ratio Next, SJF Shortest Job First

- **소프트웨어 아키텍처 4 + 1 뷰**
  - 논리 뷰 logical
    시스템 기능 요구사항을 클래스, 컴포넌트 종류와 관계로 설명
    설계가 실제로 구현되는지 설명
    설계자 관점
    클래스/시퀀스 다이어그램
  - 프로세스 뷰 process
    비기능 속성(자원 사용, 병행 실행, 비동기, 이벤트 처리) 표현
    성능, 확장성, 효율성
    시스템 통합자 관점
    시퀀스/협력 다이어그램
  - 구현 뷰 implementation
    개별 환경 내 정적인 소프트웨어 모듈 구성 설명
    컴포넌트 구조, 의존성, 부가 정보
    실제 구현할 수 있는지 확인
    개발자 관점
    컴포넌트 다이어그램
  - 배포 뷰 deployment
    컴포넌트가 물리적 환경에서 어떻게 배치 연결되는지 매핑
    여기에 뭐가 들어가고… 하는 배치를 결정
    배치(배포) 다이어그램
  - 유스케이스 뷰

다른 뷰 검증
사용자, 설계자, 개발자, 테스트 관점
유스케이스 다이어그램

- **품질 표준**
  - 국제 제품 품질 : 9126 / 14598 / 12119 / 25000
  - 국제 프로세스 품질 : 9001 / 12207 / 15504(SPICE) / CMMi

  - ISO/IEC 9126
    사용자 관점. SW 품질 측정, 평가. 기신사효유이
    기능성 : 적절(적합), 정확, 상호 운용, 보안, 호환
    신뢰성 : 성숙, 결함 허용, 회복
    사용성 : 이해, 학습, 운용, 친밀
    효율성 : 시간 효율, 자원 효율
    유지보수성 : 분석, 변경, 안정, 시험
    이식성 : 적용, 설치, 대체, 공존

  - ISO/IEC 14598
    제품 평가 프로세스/모듈 제공. 개발과정/완료제품 품질 평가 표준. 반재공객
    반복성Repeatability / 재현성Reproducibility / 공정성Impartiality /
    객관성Objectivity

  - ISO/IEC 12119
    품질 요구사항/테스트 국제 표준

  - ISO/IEC 25000
    SQuaRe. 통합. 관모측요평
    2500n : 품질관리 / 1 : 품질 모델 / 2 : 품질 측정 / 3 : 품질 요구 / 4 : 품질 평가

  - ISO/IEC 9001
    설계/개발, 생산, 설치, 서비스 과정 품질 보증 모델

  - ISO/IEC 12207
    소프트웨어 생명주기 프로세스
    획득, 공급, 개발, 운영, 유지보수 체계적으로 관리. 기조지
    기본 프로세스, 조직 프로세스, 지원 프로세스

- ISO/IEC 15504 SPICE
  SW 프로세스 평가, 개선. 품질 및 생산성 향상
  수행 능력 수준 : 불안정 / 수행 / 관리 / 확립 / 예측 / 최적화

- CMMi
  기존 CMM 모델 통합. SW-CMM + SE-CMM
  SPICE 준수. 개발 능력, 성숙도 평가.
  단계적 표현 : 조직 전체 성숙도. 초기화 / 관리 / 정의 / 정량적 관리 / 최적화
  연속적 표현 : 프로세스 영역별.
  영역별 능력평가.

- **비트연산**

1. AND 연산 &
   둘 다 1일 경우 1로 변환
   1111 & 1000 = 1000

2. OR 연산 |
   둘 중 1이 있으면 1로 변환
   1111 | 1000 = 1111

3. XOR 연산 ^
   두 수가 다르면 1로 변환
   1111 ^ 1000 = 0111 = 111

4. NOT 연산 ~
   각 자리 수 반대로 변환

5. SHIFT 연산 <<, >>, >>>
   << : 각자리를 왼쪽으로 N칸 밀고 제일 첫째 자리는 0으로 채우기

   ex) 1111 << 2 = 111100

   1111(15) 이 111100(60)로 바뀐것을 보면 x * 2^y 값을 반환

   >> : 각자리를 오른쪽으로 N칸 밀고 남는 자리수는 삭제

   ex) 1111 >> 2 = 11

   1111(15) 이 11(3)로 변형되는 것을 보면 x / 2^y 으로 나머지

>>>: >> 연산할 때 남는 공간 0으로 채움. 항상 양수만 고려.

- **관계 대수 & 관계 해석**
  - 관계 대수 : 절차적 (순서 명시). 일반집합 연산자 / 순수관계 연산자

1. 관계대수의 개념

- 관계대수는 관계형 데이터베이스에서 원하는 정보와 그 정보를 어떻게 유도하는가를 기술하는 **절차적 언어**이다.

2. 관계대수의 특징

- 상용 관계 DBMS 들에서 널리 사용되는 **SQL의 이론적인 기초**
- SQL을 구현하고 최적화하기 위해 **DBMS의 내부 언어**로도 사용

3. 관계대수의 연산자

구분	유형	표기법	설명
일반 집합 연산 (Set Operations)	합집합	U	이항 연산으로 관계성이 있는 두 개의 릴레이션을 합집합하여 하나의 릴레이션을 만들어 내는 연산
	교집합	∩	이항 연산으로 관계성이 있는 두개의 릴레이션에서 중복되어 있는 내용을 선택하여 새로운 릴레이션을 만들어 내는 연산
	차집합	-	이항 연산으로 관계성이 있는 두개의 릴레이션이 있을 때 그 중 하나의 릴레이션에서 또 다른 릴레이션의 내용과 겹치는 내용을 제거해서 새로운 릴레이션을 생성하는 연산
	카티션 프로덕트	X	이항 연산으로 두 릴레이션의 현재 투플로 구성 가능한 모든 조합 만드는 연산
순수 관계 연산 (Relational Operations)	Select	ó	릴레이션의 주어진 조건을 만족하는 튜플을 선택하는 연산
	Project	π	단항 연산으로 릴레이션에서 참조하고자 하는 어트리뷰트를 선택하여 분리해 내는 연산
	Join	⋈	두 릴레이션 간의 에트리뷰트 값이 동일한 튜플을 연결하는 연산
	Division	÷	두 개의 릴레이션 A와 B가 있을 때 B의 릴레이션의 모든 조건을 만족하는 경우의 튜플들을 릴레이션 A에서 분리해 내어 프로젝션하는 연산

  - 관계 해석 : 비절차적. 프레디킷 해석. 튜플 관계 해석 / 도메인 해석

구분	구성요소	기호	설명
연산자	OR 연산	V	원자식 간 "또는"이라는 관계로 연결
	AND 연산	∧	원자식 간 "그리고"라는 관계로 연결
	NOT 연산	¬	원자식에 대해 부정
정량자	전칭 정량자 (Universal Quantifier	∀	모든 가능한 튜플 "For All" # All의 'A'를 뒤집은 형태 ★
	존재 정량자 (Existential Quantifier)	∃	어떤 튜플 하나라도 존재 "There Exists" # Exists의 'E'를 뒤집은 형태 ★

- **서비스 공격**

  * DoS 공격 : 자원부족하게 함

    - SYN Flooding : 3-way-handshake, 서버 대기, SYN 패킷만 보냄
    - UDP Flooding : 대량의 UDP패킷, 임의 포트번호
    - SMURPING : IP, ICMP 데이터 집중, Echo
    - Ping of Death : ping명령, 큰사이즈 패킷, 프로토콜 공격
    - Land : IP주소 집중, 자기자신 무한 응답
    - TearDrop : Fragment Offset, 패킷 재조립, 과부하
    - Bonk : 같은 시퀀스번호 계속보냄
    - Boink : 시퀀스번호 빈공간 생성

  * DDos 공격 : 분산 지점에서 한곳의 서버 공격, 네트워크 취약점 호스트, 분산 서비스 공격

    - Trinoo : UDP flood 유발
    - TFN(Tribe Flood Network) : Trinoo와 비슷
    - Stacheldraht : 분산 서비스거부 에이전트 역할

    DDoS 공격 구성요소 (HAMAD)

구성요소	설명
핸들러(Handler)	마스터 시스템의 역할을 수행하는 프로그램
에이전트(Agent)	공격 대상에 직접 공격을 가하는 시스템
마스터(Master)	공격자에게서 직접 명령을 받는 시스템
공격자(Attacker)	공격을 주도하는 해커의 컴퓨터
데몬 프로그램(Daemon)	에이전트 시스템의 역할을 수행하는 프로그램

  * 애플리케이션공격 :

    - HTTP GET Flooding : 캐시옵션 조작, 웹서버 자원소진,
    - Slowloris(Slow HTTP Header Dos) : 헤더 끝 개행문자열 전송 x 연결자원 소진,
    - RUDY(Slow HTTP POST Dos) : contentlength 매우 크게 설정. 메시지 바디부분 소량으로 보냄
    - Slow HTTP Read Dos : TCP 윈도크기, 처리율 감소
    - Hulk Dos : URL 지속 변경 다량으로 GET 요청
    - Hash Dos : 해시충돌발생

* 네트워크 침해 공격 :

- Sniffing : 직접공격 x 데이터 몰래 들여다봄
- 트로이목마 : 겉보기에는 정상 실행하면 악성코드

+ 보안관련 용어

- 스미싱 : 문자메시지, 개인신용정보
- 스피어피싱 : 이메일, 개인정보 탈취
- APT : 특정기업이나 조직 네트워크
- 랜섬웨어 : 몸값요구
- 공급망(Supply Chain Attack) : 개발사의 네트워크 침투

- **보안 도구**
  - Ping : ICMP에서 확인용 명령어
  - tracer : 데이터가 목적지까지 도달했는지 확인
  - cron : 특정 시간 특정 작업 수행하게 하는 스케줄링 명령어
  - tcpdump : 스니핑 도구의 일종. 자기 컴퓨터에 들어오는 모든 내용 도청
  - tripwire : 크래커 방지. 비교 통해 차이점 감지. 무결성 검증

- **백도어 탐지 기법**
  - 현재 동작중인 프로세스, 열린 포트 확인
  - setuid 파일 검사
  - 백신 및 백도어 탐지 툴 이용
  - 무결성 검사
  - 로그 분석

- **페이징 기법**
  - 페이지 크기가 작으면
    페이지 사상 테이블에 많은 공간 필요
    내부 단편화 감소
    사용하는 페이지 집합 효율적 운영
    특정 참조 구역성만 포함해 기억 장치 효율적
    페이지 정보 있는 페이지 맵 테이블 크기가 커져 매핑 속도 늦어짐
    디스크 접근 횟수 많아져 전체적 입출력 시간 증가

  - 페이지 크기 크면
    테이블 크기 작아져 주기억 장치 공간 절약

참조하는 정보와는 무관한 많은 양의 정보가 주기억 장치에 남음
페이지 테이블이 복잡하지 않아 관리 용이
페이지 맵 테이블 크기 작아져 매핑 속도 빨라짐
디스크 접근 횟수 줄어 전체적인 입출력 시간 감소
페이지 단편화 증가, 기억 공간 낭비 초래

- **결함 조치 상태**
  - 열린 Open : 오류 보고. 분석 안 됨.
  - 할당됨 Assigned : 수정 위해 개발자에게 할당
  - 연기됨 Deferred : 낮은 우선순위로 수정 연기
  - 종료됨 Closed : 재 테스트 시 오류 사라짐
  - 수정됨 Fixed : 오류 수정
  - 분류됨 Classified : 오류 아니라고 확인함

- **순위 함수**
  - RANK : 동일 순서 레코드 -> 후순위는 넘어감. 1>2>2>3
  - DENSE_RANK : 후 순위 안 넘어감. 1>2>2>3
  - ROW_NUMBER : 동일 순서 무관 연속 번호 부여. 1>2>3>4

- **행 순서 함수**
  - FIRST_VALUE : 가장 먼저 나온 값. MIN
  - LAST_VALUE : 가장 뒤에 나온 값. MAX
  - LAG : 이전 행 값
  - LEAD : 이후 행 값

- **트랜잭션 고립화**
  - 0 : Read Uncommitted : 트랜잭션 처리 중인 데이터 읽을 수 있음
    dirty read : 변경된, 커밋되지 않은 것도 읽을 수 있음
  - 1 : Read committed : 커밋한 데이터만 읽을 수 있음
  - 2: Repeatable Read : 선행 트랜잭션에서 읽은 데이터, 후행 트랜잭션이 수정 못함. 선행 트랜잭션이 다시 읽었을 때 동일한 결과를 보여주려고. but 신규 데이터 입력 못 막음. 갑자기 데이터 늘어나는 Phantom Read.
  - 3 : Serializable Read : 직렬화 읽기. 2와 달리 후행이 수정 가능. 후행 트랜잭션에서 수정 입력 삭제한 데이터가 선행 트랜잭션에 영향을 안 줌. 선 트랜이 14개 조회했으면 후 트랜이 뭘 수정 추가해서 변경됐어도 그대로 14임.

- **집합 연산**
  - INTERSECT : 교집합
  - MINUS : 차집합
  - UNION : 합집합
  - UNION ALL : 중복 허용 합집합

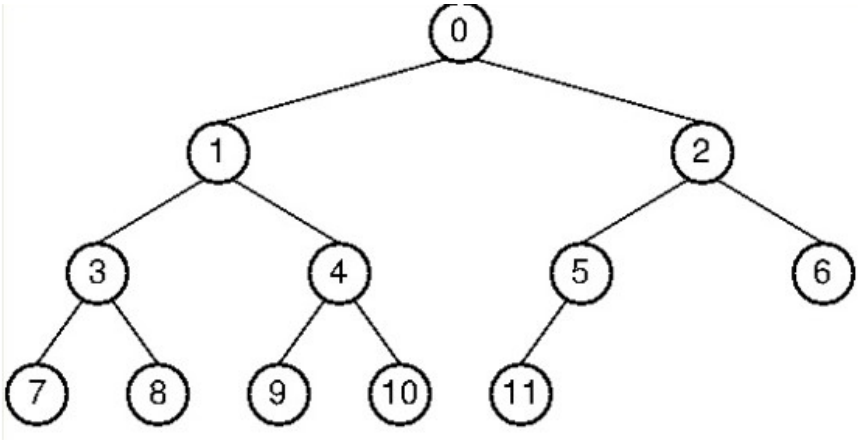
- **소프트웨어 비용 산정 모델**
  - SAAM : 최초 개발. 변경 용이성. 가능성 중심 평가.
  - ATAM : SAAM 다음 버전. 품질(가용, 보안, 성능, 사용)속성 이해 상충관계 평가. 상충 관계 유지. 변경 가능성. 시험 가능성
  - CBAM : ATAM 바탕. 시스템 아키텍처 분석 중심. 경제적 가치 평가
  - ADR : 컴포넌트, 모듈 단위 응집도 평가
  - ARID : 전체 아닌 특정 부분 품질 요소에 집중

- **프로젝트 비용 산정 모델**
  - LOC Lines of Code : 원시 코드 라인 수 낙관치, 중간치, 비관치로 예측치
  - man Month ; 1개월간 할 수 있는 일의 양
  - COCOMO Constructive Cost Model : 보헴. 프로그램 규모따라 필요 노력 Man-Month로 산정. 조직형 단순형 Organic / 반분리 중간형 Semi-Detached / 임베디드 Embeded
  - Putnam : Rayleigh-Norden 곡선 노력 분포도. 자동화 추정 도구 SLIM
  - FP Functional Point 기능 점수 : 자료입력(입력 양식) / 정보 출력(출력보고서) / 명령어(사용자 질의 수) / 데이터 파일 / 필요한 외부 루틴과의 인터페이스

- **XP 기본 원리**
  - Pair Programming 짝 프로그래밍
  - Collective Ownership 공동 코드 소유
  - Continuous Integration 지속적인 통합
  - Planning Process 계획 세우기
  - Small Release 작은 릴리즈
  - Metaphor 메타포
  - Simple Design 간단 디자인
  - TDD Test Driven Develop 테스트 기반 개발
  - Refactoring 리팩토링
  - 40-Hour Work 40시간 작업
  - On Site Customer 고객 상주

- Coding Standard 코드 표준

- **이진 트리 순회**
  - Preorder 전위 : 루트 먼저. 루트 > 왼쪽 > 오른쪽
  - Inorder 중위 : 루트 중간. 왼쪽 > 루트 > 오른쪽
  - Postorder 후위 : 루트 마지막. 왼쪽 > 오른쪽 > 루트
  - Levelorder 층별 : 위에서 아래로.



위와 같은 트리가 있다고 한다면 각각 순회방법은 다음과 같습니다.

전위 순회 : 0->1->3->7->8->4->9->10->2->5->11->6

중위 순회 : 7->3->8->1->9->4->10->0->11->5->2->6

후위 순회 : 7->8->3->9->10->4->1->11->5->6->2->0

층별 순회 : 0->1->2->3->4->5->6->7->8->9->10->11

- **스택 & 큐**
  - 스택
    쌓아올림.
    push / pop
    마지막 삽입 자료 먼저 삭제
    후입선출 Last-In-First-Out LIFO
    웹 브라우저 방문 기록 / 역순 문자열 만들기 / 실행 취소 / 후위 표기법 계산 / 수식 괄호 검사
  - 큐
    줄 서서 기다리다
    enQueue / deQueue

선입선출 First-In-First-Out FIFO
우선순위가 같은 작업 예약 / 프린터 인쇄 대기열 / 은행 업무 / 콜센터 고객
대기시간 / 프로세스 관리 / 너비 우선 탐색 Breadth-First-Search / 캐시 구현

- **빌드 자동화 도구**
  - 코드 컴파일해 실행 파일로 만들고 테스트, 배포 자동화하는 도구
  - 지속적통합 개발 환경(애자일)에서 좋음
  - Ant, Make, Maven, Gradle, Jenkins

  - Jenkins
    자바 기반 오픈 소스.
    서블릿 컨테이너에서 실행. 서버 기반
    SVN, Git 등 형상 관리 도구와 연동
    Web Gui 제공. 쉬움
    분산 빌드, 테스트 가능

  - Gradle
    Groovy 기반 오픈 소스 자동화 도구
    안드로이드 앱 개발 환경
    플러그 인 쓰면 Java, C, C++, Python 도 가능
    Groovy로 만든 DSL을 스크립트 언어로 사용
    실행 처리 명령 모은 태스크 단위로 실행
    빌드 캐시 기능(태스크 재사용, 공유) -> 빌드 속도 향상

- **교착 상태**
  - 상호배제 / 점유와 대기 / 비선점 / 환형 대기
  - Prevention 예방 : 점유 자원 해제 후 새 자원 요청
  - Avoidance 회피 : Banker's Algorithm, wait-die, wound-wait
  - Detection 발견 : 자원 할당 그래프, wait for graph
  - Recovery 회복 : 프로세스 킬, 자원 선점

- **미들웨어 솔루션 유형**
  - DB 미들웨어 : 애플리케이션과 DB간 통신 원활하게 함
  - RPC 원격 프로시저 호출 : 원격 프로시저를 로컬 프로시저처럼 호출
  - MOM 메시지지향 : 비동기형 메시지 전달 방식. 이기종 분산 DB시스템의 데이터 동기 사용

- TP-Monitor : 온라인 업무에서 트랜잭션 처리. 빠른 응답 속도
- Lagacyware : 새로운 업데이트 기능 덧붙일 때
- ORB 객체 기반 : CORBA 표준 스펙 구현
- WAS : 동적 콘텐츠, 웹 환경 구현

- **McCabe의 Cyclomatic 수 계산식**
  - V(G) = 간선 - 노드 + 2

- **프로젝트 일정관리**
  - CPM Critical Path Method: 주 공정법. 임계경로. 최장거리
  - CCPM Critical Chain Project Management: 자원제약사항 고려
  - PERT Program Evaluation Technique: 비관치, 중간치, 낙관치

- **컴포넌트 설계 시 협약에 의한 설계**
  - 클래스에 대한 가정 공유
  - 소프트웨어 컴포넌트에 대한 정확한 인터페이스 명세를 위해 조건 설계
  - 선행 조건 : 컴포넌트 오퍼레이션 사용 전 참 조건
  - 결과 조건 : 사용 후 만족 조건
  - 불변 조건 : 실행 동안 항상 만족 조건

- **TCP/UDP**
  - TCP : 흐름 혼잡제어 신뢰
  - UDP : 비신뢰. 데이터그램지향. 멀티캐스팅으로 여러 다수 지점 전송 가능

- **접근 통제 유형**
  - DAC 임의적 : 데이터 소유자에게 권한. 신분 기반. 변경용이. 유연
  - MAC 강제적 : 시스템에게 권한. 보안 등급 기준. 고정. 변경 어려워. 안정. 중앙 집중. 규칙
  - RBAC : 중앙관리자에게 권한. 역할 기반. 변경 관리 용이
  - ABAC : 속성 기반

- **보안 프레임워크**
  - 구성 요소 : 정보보호 요소 / 전략 및 거버넌스 / 침해사고 관리 / 계층적 보안 기술
  - 정보보호 요소 : 컴플라이언스 / 보안 위협 / 비즈니스 요구사항 / 비즈니스 기회

- **취약점 분석 절차**

- 1. 자산 조사 및 분석
- 2. 진단 대상 선정
- 3. 제약사항 확인
- 4. 진단 수행
- 5. 결과 분석 /보고서 작성

- **시간 복잡도 알고리즘 분류**
  - O(1): 해시 함수
  - O(n) : 순차 탐색
  - O(logn) : 이진 탐색
  - O(Nlog2N) : 퀵, 병합 정렬
  - O(n2) : 거품, 삽입, 선택 정렬

- **해싱 함수**
  - 제산법 : 나머지 연산자 %
  - 제곱법 : 키값을 제곱
  - 숫자 분석 : 어떤 분포인지 조사
  - 폴딩법 : 여러 부분으로 나눠 사용
  - 기수 변환 : 다른 진법으로 간주 변환
  - 무작위방법 : 난수

- **HIPO 차트**
  - 체계적 문서관리
  - 기호 도표 쉽다
  - 기능과 자료 의존 관계 동시 표현
  - 변경 유지보수 용이
  - 시스템 기능을 고유 모듈로 분할. 이들 간 인터페이스를 계층구조로 표현
  - 가시적 : 시스템 전체 기능 흐름. 계층 구조도
  - 총체적 : 입력 처리 출력 정보 제공. 프로그램 구성 기술
  - 세부적 : 총체적 도표의 기능 구성하는 기본 요소 상세히 기술

- **네트워크**
  - 7계층
    - 1. 물리 : 실제 장치 연결. 전기 물리 세부사항
    - 2. 데이터링크 : 링크 설정 유지 종료. 노드 간 오류 흐름 회선 제어

        - 3. 네트워크 : 다양한 길이 패킷을 네트워크로 전달. 전송 계층이 요구하는 서비스 품질을 위한수단 제공. IP ARP RARP ICMP Telnet
        - 4. 전송 : 상위 계층이 데이터 전달 유효성이나 효율성 걱정 안하게 함. 종단간 사용자들이 신뢰성있는데이터 전달하게 오류 검출 복구 흐름제어. TCP(기본헤더20바이트. 60까지 확장) UDP
        - 5. 세션 : 응용 프로그램 간대화 유지. RPC, NetBIOS
        - 6. 표현 : 애플리케이션이 다루는 정보 통신에 알맞은형태로 만듦. 하위 계층 데이터를 사용자가 이해할 수 있게 만듦. JPEG, MPEG
        - 7. 응용 : 응용 프로세스와 직접 관계. HTTP SMTP FTP
    - 네트워크 프로토콜
        - IP : internet protocol : 송신 수신 간 패킷 단위로 데이터 교환하는 네트워크에서 사용. 비신뢰. 비연결. 에러 흐름제어 없음. 20~40바이트.
            - IPv4 : 32비트, 헤더 크기 가변. 8비트 씩 4부분 10진수
            - IPv6 : 128비트. 헤더 크기 고정. 16비트씩 8부분 16진수
            - 둘 전환 : 듀얼 스택, 터널링, 주소 변환
        - ARP : Address Resolution Protocol : IP 네트워크 상에서 IP를 MAC으로 변환
        - RARP : Reverse ARP : MAC으로 서버에게 IP주소 요청
        - ICMP 인터넷 제어 메시지 : Internet Control Message protocol : IP 패킷 처리할 때 문제 생기면 알려줌
        - IGMP : Internet Group Management Protocol : 인터넷 그룹 관리. 호스트 컴퓨터와 인접 라우터가 멀티캐스트 그룹 멤버십 구성할 때 사용
        - 라우팅 프로토콜 : 테이터 전송 시 목적지까지 최적 경로 설정. 라우터 간 상호 통신규약.
            - 내부 : RIP Routing Information Protocol, OSPF Open Shortest Path First
            - 외부 : EGP, BGP Border Gateway Protocol

- **Nosql 유형**

유형	
Key-Value Store	Redis, DynamoDB
Column Family Data Store	HBase, Cas-sandra
Document Store	MongoDB, Couchbase
Graph Store	Neo4j, AllegroGraph

- **빅데이터 기술**

구분	기술	설명
비정형 데이터 수집	척와 (Chukwa)	분산된 각 서버에서 에이전트를 실행하고, 컬렉터(Collector)가 에이전트로부터 데이터를 받아 HDFS에 저장
정형 데이터 수집	스쿱 (Sqoop)	커넥터(Connector)를 사용하여 관계형 데이터베이스 시스템(RDBMS)에서 HDFS로 데이터를 수집
분산 데이터 저장	HDFS	대용량 파일을 분산된 서버에 저장하고, 그 저장된 데이터를 빠르게 처리할 수 있게 하는 하둡 파일 시스템
	하둡 (Hadoop)	오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼이다. [2020년 1회] 일반 PC급 컴퓨터들로 가상화된 대형 스토리지를 형성한다. 다양한 소스를 통해 생성된 빅데이터를 효율적으로 저장하고 처리한다.
분산 데이터 처리	맵리듀스 (MapReduce)	대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델이다. Google에 의해 고안된 기술로써 대표적인 대용량 데이터 처리를 위한 병렬 처리 기법을 제공한다. [2020년 4회] 임의의 순서로 정렬된 데이터를 분산 처리하고 이를 다시 합치는 과정을 거친다.
분산 데이터베이스	HBase	컬럼 기반 저장소로 HDFS와 인터페이스 제공
데이터 가공	피그 (Pig)	대용량 데이터 집합을 분석하기 위한 플랫폼으로 하둡을 이용하여 맵리듀스를 사용하기 위한 높은 수준의 스크립트 언어인 피그 라틴이라는 자체 언어를 제공
	하이브 (Hive)	하둡 기반의 DW 솔루션 SQL과 매우 유사한 HiveQL이라는 쿼리를 제공
데이터 분석 및 시각화	R	통계 프로그래밍 언어인 S 언어를 기반으로 만들어진 오픈 소스 프로그래밍 언어