시험에 나오는 것만 공부한다! 시나공시리즈

1 과목 소프트웨어 설계

핵심 001 소프트웨어 생명 주기 (Software Life Cycle)

소프트웨어 생명 주기는 소프트웨어 개발 방법론의 바탕이 되는 것으로, 소프트웨어를 개발하기 위해 정의하고 운용, 유지보수 등의 과정을 각 단계별로 나눈 것이다.

- 소프트웨어 생명 주기는 소프트웨어 개발 단계와 각 단계별 주요 활동, 그리고 활동의 결과에 대한 산출물로 표현한다. 소프트웨어 수명 주기라고도 한다.
- 소프트웨어 생명 주기를 표현하는 형태를 소프트웨어 생명 주기 모형이라고 하며, 소프트웨어 프로세스 모형 또는 소프트웨어 공학 패러다임이라고도 한다.

※ 소프트웨어 공학의 개념

소프트웨어 공학(SE; Software Engineering)은 소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문이며 여러 가지 방법론과 도구, 관리 기법들을 통하여 소프트웨어의 품질과 생산성 향상을 목적으로 한다.

※ 소프트웨어 공학의 기본 원칙

- 현대적인 프로그래밍 기술을 계속적으로 적용해야 한다.
- 개발된 소프트웨어의 품질이 유지되도록 지속적으로 검증해야 한다.
- 소프트웨어 개발 관련 사항 및 결과에 대한 명확한 기록을 유지해야 한다.

핵심 002 폭포수 모형(Waterfall Model)

폭포수 모형은 폭포에서 한번 떨어진 물은 거슬러 올라갈 수 없듯이 소프트웨어 개발도 이전 단계로 돌아갈 수 없다는 전제하에 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하는 개발 방법론이다.

- 폭포수 모형은 소프트웨어 공학에서 가장 오래되고 가장 폭넓게 사용된 전통적인 소프트웨어 생명 주기 모형으로, 고전적 생명 주기 모형이라고도 한다.
- 소프트웨어 개발 과정의 한 단계가 끝나야만 다음 단계로 넘어갈 수 있는 선형 순차적 모형이다.
- 모형을 적용한 경험과 성공 사례가 많다.
- 각 단계가 끝난 후에는 다음 단계를 수행하기 위한 결과물이 명확하게 산출되어야 한다.

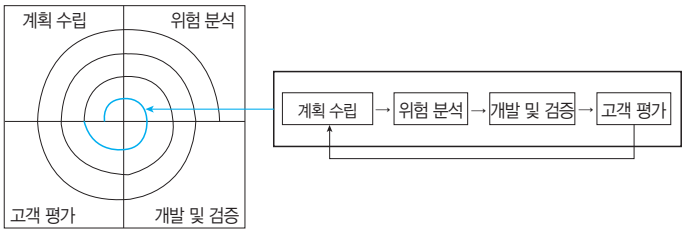
불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

핵심 003 나선형 모형(Spiral Model, 점진적 모형)

나선형 모형은 보헴(Boehm)이 제안한 것으로, 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형이다.

- 나선을 따라 돌듯이 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 최종 소프트웨어를 개발하는 것으로, 점진적 모형이라고도 한다.
- 소프트웨어를 개발하면서 발생할 수 있는 위험을 관리하고 최소화하는 것을 목적으로 한다.
- 점진적으로 개발 과정이 반복되므로 누락되거나 추가된 요구사항을 첨가할 수 있고, 정밀하며, 유지보수 과정이 필요 없다.



핵심 004 애자일 모형(Agile Model)

애자일은 '민첩한', '기민한'이라는 의미로, 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발과정을 진행한다.

- 애자일 모형은 어느 특정 개발 방법론이 아니라 좋은 것을 빠르고 낭비 없게 만들기 위해 고객과의 소통에 초점을 맞춘 방법론을 통칭한다.

- 애자일 모형은 기업 활동 전반에 걸쳐 사용된다.
- 애자일 모형을 기반으로 하는 소프트웨어 개발 모형에는 스크럼(Scrum), XP(eXtreme Programming), 칸반(Kanban), Lean, 크리스탈(Crystal), ASD(Adaptive Software Development), 기능 중심 개발(FDD; Feature Driven Development), DSDM(Dynamic System Development Method), DAD(Disciplined Agile Delivery) 등이 있다.

※ 애자일 개발 4가지 핵심 가치

1. 프로세스와 도구보다는 개인과 상호작용에 더 가치를 둔다.
2. 방대한 문서보다는 실행되는 SW에 더 가치를 둔다.
3. 계약 협상보다는 고객과 협업에 더 가치를 둔다.
4. 계획을 따르기 보다는 변화에 반응하는 것에 더 가치를 둔다.

핵심 005 스크럼의 개요

스크럼이란 럭비에서 반칙으로 경기가 중단된 경우 양 팀의 선수들이 럭비공을 가운데 두고 상대팀을 밀치기 위해 서로 대치해 있는 대형을 말한다. 스크럼은 이처럼 팀이 중심이 되어 개발의 효율성을 높인다는 의미가 내포된 용어이다.

- 스크럼은 팀원 스스로가 스크럼 팀을 구성(self-organizing)해야 하며, 개발 작업에 관한 모든 것을 스스로 해결(cross-functional)할 수 있어야 한다.
- 스크럼 팀은 제품 책임자, 스크럼 마스터, 개발팀으로 구성된다.

제품 책임자(PO; Product Owner)

- 이해관계자들 중 개발될 제품에 대한 이해도가 높고, 요구사항을 책임지고 의사 결정할 사람으로 선정하는데, 주로 개발 의뢰자나 사용자가 담당한다.
- 이해관계자들의 의견을 종합하여 제품에 대한 요구사항을 작성하는 주체다.
- 제품에 대한 테스트를 수행하면서 주기적으로 요구사항의 우선순위를 갱신한다.

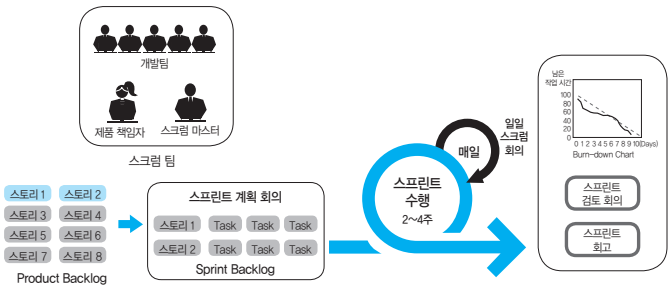
스크럼 마스터(SM; Scrum Master)

- 스크럼 팀이 스크럼을 잘 수행할 수 있도록 객관적인 시각에서 조언을 해주는 가이드 역할을 수행한다. 팀원들을 통제하는 것이 목표가 아니다.
- 일일 스크럼 회의를 주관하여 진행 사항을 점검하고, 개발 과정에서 발생된 장애 요소를 공론화하여 처리한다.

개발팀(DT; Development Team)

- 제품 책임자와 스크럼 마스터를 제외한 모든 팀원으로, 개발자 외에도 디자이너, 테스터 등 제품 개발을 위해 참여하는 모든 사람이 대상이 된다.
- 보통 최대 인원은 7~8명이 적당하다.

핵심 006 스크럼 개발 프로세스



제품 백로그 (Product Backlog)	제품 개발에 필요한 모든 요구사항(User Story)을 우선순위에 따라 나열한 목록이다.
스프린트 계획 회의 (Sprint Planning Meeting)	제품 백로그 중 이번 스프린트에서 수행할 작업을 대상으로 단기 일정을 수립하는 것이다.
스프린트(Sprint)	실제 개발 작업을 진행하는 과정으로, 보통 2 ~ 4주 정도의 기간 내에서 진행한다.
일일 스크럼 회의 (Daily Scrum Meeting)	• 모든 팀원이 매일 약속된 시간에 약 15분 정도의 짧은 시간동안 진행 상황을 점검한다. • 회의는 보통 서서 진행하며, 남은 작업 시간은 소멸 차트(Burn-down Chart)에 표시한다.
스프린트 검토 회의 (Sprint Review)	부분 또는 전체 완성 제품이 요구사항에 잘 부합되는지 사용자가 포함된 참석자 앞에서 테스팅을 수행한다.
스프린트 회고 (Sprint Retrospective)	스프린트 주기를 되돌아보며 정해놓은 규칙을 잘 준수했는지, 개선할 점은 없는지 등을 확인하고 기록한다.

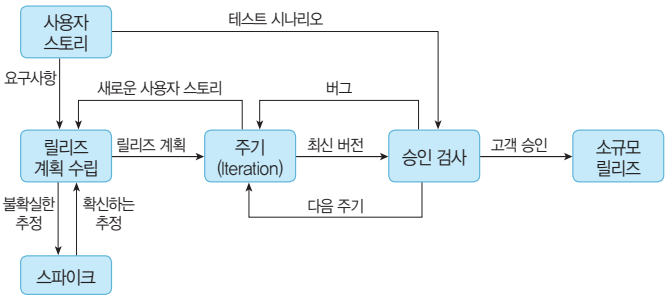
핵심 007 XP(eXtreme Programming) 및 개발 프로세스

XP(eXtreme Programming)

XP(eXtreme Programming)는 수시로 발생하는 고객의 요구사항에 유연하게 대응하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 개발 생산성을 향상시키는 방법이다.

- XP는 짧고 반복적인 개발 주기, 단순한 설계, 고객의 적극적인 참여를 통해 소프트웨어를 빠르게 개발하는 것을 목적으로 한다.
- 릴리즈의 기간을 짧게 반복하면서 고객의 요구사항 반영에 대한 가시성을 높인다.
- XP의 5가지 핵심 가치 : 의사소통(Communication), 단순성(Simplicity), 용기(Courage), 존중(Respect), 피드백(Feedback)

XP 개발 프로세스



사용자 스토리 (User Story)	• 고객의 요구사항을 간단한 시나리오로 표현한 것이다. • 내용은 기능 단위로 구성하며, 필요한 경우 간단한 테스트 사항(Test Case)도 기재한다.
릴리즈 계획 수립(Release Planning)	• 몇 개의 스토리가 적용되어 부분적으로 기능이 완료된 제품을 제공하는 것을 릴리즈라고 한다. • 부분 혹은 전체 개발 완료 시점에 대한 일정을 수립한다.
스파이크 (Spike)	요구사항의 신뢰성을 높이고 기술 문제에 대한 위험을 감소시키기 위해 별도로 만드는 간단한 프로그램이다.
이터레이션 (Iteration)	• 하나의 릴리즈를 더 세분화 한 단위를 이터레이션(Iteration)이라고 한다. • 일반적으로 1~3주 정도의 기간으로 진행된다.
승인 검사 (Acceptance Test, 인수 테스트)	• 하나의 이터레이션 안에서 계획된 릴리즈 단위의 부분 완료 제품이 구현되면 수행하는 테스트이다. • 테스트가 완료되면 다음 이터레이션을 진행한다.
소규모 릴리즈 (Small Release)	릴리즈를 소규모로 하게 되면, 고객의 반응을 기능별로 확인할 수 있어, 고객의 요구사항에 좀 더 유연하게 대응할 수 있다.

핵심 008 XP의 주요 실천 방법(Practice)

Pair Programming (짝 프로그래밍)	다른 사람과 함께 프로그래밍을 수행함으로써 개발에 대한 책임을 공동으로 나눠 갖는 환경을 조성한다.
Collective Ownership (공동 코드 소유)	개발 코드에 대한 권한과 책임을 공동으로 소유한다.
Test-Driven Development (테스트 주도 개발)	• 개발자가 실제 코드를 작성하기 전에 테스트 케이스를 먼저 작성하므로 자신이 무엇을 해야할지를 정확히 파악한다. • 테스트가 지속적으로 진행될 수 있도록 자동화된 테스팅 도구(구조, 프레임워크)를 사용한다.
Whole Team (전체 팀)	개발에 참여하는 모든 구성원(고객 포함)들은 각자 자신의 역할이 있고 그 역할에 대한 책임을 가져야 한다.
Continuous Integration (계속적인 통합)	모듈 단위로 나눠서 개발된 코드들은 하나의 작업이 마무리될 때마다 지속적으로 통합된다.
Design Improvement (디자인 개선) 또는 Refactoring(리팩토링)	프로그램 기능의 변경 없이, 단순화, 유연성 강화 등을 통해 시스템을 재구성한다.
Small Releases (소규모 릴리즈)	릴리즈 기간을 짧게 반복함으로써 고객의 요구 변화에 신속히 대응할 수 있다.

핵심 009 현행 시스템 파악

단계	현행 시스템	내용
1단계	시스템 구성 파악	조직의 주요 업무를 담당하는 기간 업무와 이를 지원하는 지원 업무로 구분하여 기술함
	시스템 기능 파악	현재 제공하는 기능들을 주요 기능과 하부 기능, 세부 기능으로 구분하여 계층형으로 표시함
	시스템 인터페이스 파악	단위 업무 시스템 간에 주고받는 데이터의 종류, 형식, 프로토콜, 연계 유형, 주기 등을 명시함

2단계	아키텍처 구성 파악	최상위 수준에서 계층별로 표현한 아키텍처 구성도를 작성함
	소프트웨어 구성 파악	소프트웨어들의 제품명, 용도, 라이선스 적용 방식, 라이선스 수 등을 명시함
3단계	하드웨어 구성 파악	단위 업무 시스템들이 운용되는 서버의 주요 사양과 수량, 그리고 서버의 이중화의 적용 여부를 명시함
	네트워크 구성 파악	서버의 위치, 서버 간의 네트워크 연결 방식을 네트워크 구성도로 작성함

핵심 010 운영체제(OS, Operating System)

운영체제는 컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어이다.

- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공해준다.
- 컴퓨터 운영체제의 종류에는 Windows, UNIX, Linux, Mac OS 등이, 모바일 운영체제에는 iOS, Android 등이 있다.
- 운영체제 관련 요구사항 식별 시 고려사항
  - 가용성
  - 성능
  - 기술 지원
  - 주변 기기
  - 구축 비용

핵심 011 데이터베이스 관리 시스템(DBMS)

DBMS(DataBase Management System)는 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해 주는 소프트웨어이다.

- DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공용할 수 있도록 관리해 준다.
- DBMS는 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.
- DBMS의 종류에는 Oracle, IBM DB2, Microsoft SQL Server, MySQL, SQLite, MongoDB, Redis 등이 있다.
- DBMS 관련 요구사항 식별 시 고려사항
  - 가용성
  - 성능
  - 기술 지원
  - 상호 호환성
  - 구축 비용

핵심 012 요구사항의 개념 / 요구사항의 유형

요구사항은 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영되는데 필요한 제약조건 등을 나타낸다.

- 요구사항은 소프트웨어 개발이나 유지 보수 과정에서 필요한 기준과 근거를 제공한다.

요구사항의 유형

요구사항은 일반적으로 기술하는 내용에 따라 기능 요구사항(Functional requirements)과 비기능 요구사항(Non-functional requirements)으로 구분하며, 기술 관점과 대상의 범위에 따라 시스템 요구사항(System requirements)과 사용자 요구사항(User requirements)으로 나눈다.

유형	내용
기능 요구사항 (Functional requirements)	• 시스템이 무엇을 하는지, 어떤 기능을 하는지에 대한 사항 • 시스템의 입력이나 출력으로 무엇이 포함되어야 하는지, 시스템이 어떤 데이터를 저장하거나 연산을 수행해야 하는지에 대한 사항 • 시스템이 반드시 수행해야 하는 기능 • 사용자가 시스템을 통해 제공받기를 원하는 기능

비기능 요구사항 (Non-functional requirements)	• 시스템 장비 구성 요구사항 : 하드웨어, 소프트웨어, 네트워크 등의 시스템 장비 구성에 대한 요구사항 • 성능 요구사항 : 처리 속도 및 시간, 처리량, 동적 · 정적 적용량, 가용성 등 성능에 대한 요구사항 • 인터페이스 요구사항 : 시스템 인터페이스와 사용자 인터페이스에 대한 요구사항으로 다른 소프트웨어, 하드웨어 및 통신 인터페이스, 다른 시스템과의 정보 교환에 사용되는 프로토콜과의 연계도 포함하여 기술 • 데이터 요구사항 : 초기 자료 구축 및 데이터 변환을 위한 대상, 방법, 보안이 필요한 데이터 등 데이터를 구축하기 위해 필요한 요구사항 • 테스트 요구사항 : 도입되는 장비의 성능 테스트(BMT)나 구축된 시스템이 제대로 운영되는지를 테스트하고 점검하기 위한 테스트 요구사항 • 보안 요구사항 : 시스템의 데이터 및 기능, 운영 접근을 통제하기 위한 요구사항 • 품질 요구사항 : 관리가 필요한 품질 항목, 품질 평가 대상에 대한 요구사항으로 가용성, 정합성, 상호 호환성, 대응성, 신뢰성, 사용성, 유지 · 관리성, 이식성, 확장성, 보안성 등으로 구분하여 기술 • 제약사항 : 시스템 설계, 구축, 운영과 관련하여 사전에 파악된 기술, 표준, 업무, 법 · 제도 등의 제약조건 • 프로젝트 관리 요구사항 : 프로젝트의 원활한 수행을 위한 관리 방법에 대한 요구사항 • 프로젝트 지원 요구사항 : 프로젝트의 원활한 수행을 위한 지원 사항이나 방안에 대한 요구사항

핵심 013 요구사항 개발 프로세스

요구사항 개발 프로세스는 개발 대상에 대한 요구사항을 체계적으로 도출하고 이를 분석한 후 분석 결과를 명세서(Specification Document)에 정리한 다음 마지막으로 이를 확인 및 검증하는 일련의 구조화된 활동이다.

도출 (Elicitation) → 분석 (Analysis) → 명세 (Specification) → 확인 (Validation)

요구사항 도출(Requirement Elicitation, 요구사항 수집)

요구사항 도출은 시스템, 사용자, 그리고 시스템 개발에 관련된 사람들이 서로 의견을 교환하여 요구사항이 어디에 있는지, 어떻게 수집할 것인지를 식별하고 이해하는 과정이다.

- 요구사항을 도출하는 주요 기법에는 청취와 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스 등이 있다.

요구사항 분석(Requirement Analysis)

요구사항 분석은 개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정이다.

- 요구사항 분석에는 자료 흐름도(DFD), 자료 사전(DD) 등의 도구가 사용된다.

요구사항 명세(Requirement Specification)

요구사항 명세는 분석된 요구사항을 바탕으로 모델을 작성하고 문서화하는 것을 의미한다.

- 구체적인 명세를 위해 소단위 명세서(Mini-Spec)가 사용될 수 있다.

요구사항 확인(Requirement Validation, 요구사항 검증)

요구사항 확인은 개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동이다.

핵심 014 요구사항 명세 기법

구분	정형 명세 기법	비정형 명세 기법
기법	수학적 원리 기반, 모델 기반	상태/기능/객체 중심
작성 방법	수학적 기호, 정형화된 표기법	일반 명사, 동사 등의 자연어를 기반으로 서술 또는 다이어그램으로 작성
특징	• 요구사항을 정확하고 간결하게 표현할 수 있음 • 요구사항에 대한 결과가 작성자에 관계없이 일관성이 있으므로 완전성 검증이 가능함 • 표기법이 어려워 사용자가 이해하기 어려움	• 자연어의 사용으로 인해 요구사항에 대한 결과가 작성자에 따라 다를 수 있어 일관성이 떨어지고, 해석이 달라질 수 있음 • 내용의 이해가 쉬어 의사소통이 용이함
종류	VDM, Z, Petri-net, CSP 등	FSM, Decision Table, ER모델링, State Chart(SADT) 등

핵심 015 요구사항 분석의 개요

요구사항 분석은 소프트웨어 개발의 실제적인 첫 단계로 개발 대상에 대한 사용자의 요구사항을 이해하고 문서화(명세화)하는 활동을 의미한다.

- 사용자 요구의 타당성을 조사하고 비용과 일정에 대한 제약을 설정한다.
- 사용자의 요구를 정확하게 추출하여 목표를 정하고, 어떤 방식으로 해결할 것인지를 결정한다.
- 요구사항 분석을 통한 결과는 소프트웨어 설계 단계에서 필요한 기본적인 자료가 되므로 사용자의 요구사항을 정확하고 일관성 있게 분석하여 문서화해야 한다.
- 소프트웨어 분석가에 의해 요구사항 분석이 수행되며, 이 작업 단계를 요구사항 분석 단계라고 한다.

핵심 016 구조적 분석 기법

구조적 분석 기법은 자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법으로, 다음과 같은 특징이 있다.

- 도형 중심의 분석용 도구와 분석 절차를 이용하여 사용자의 요구사항을 파악하고 문서화한다.
- 도형 중심의 도구를 사용하므로 분석가와 사용자 간의 대화가 용이하다.
- 구조적 분석 기법에서는 자료 흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec.), 개체 관계도(ERD), 상태 전이도(STD), 제어 명세서 등의 도구를 이용하여 모델링한다.

핵심 017 자료 흐름도(DFD)

자료 흐름도(DFD; Data Flow Diagram)는 요구사항 분석에서 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법으로 자료 흐름 그래프, 버블 차트라고도 한다.

- 자료 흐름도에서는 자료의 흐름과 기능을 프로세스(Process), 자료 흐름(Flow), 자료 저장소(Data Store), 단말(Terminator)의 네 가지 기본 기호로 표시한다.

기 호	의 미	표기법	
		Yourdon/ DeMacro	Gane/ Sarson
프로세스 (Process)	• 자료를 변환시키는 시스템의 한 부분(처리 과정)을 나타내며 처리, 기능, 변환, 버블이라고도 한다. • 원이나 둥근 사각형으로 표시하고 그 안에 프로세스 이름을 기입한다.	물품 확인	물품 확인
자료 흐름 (Data Flow)	• 자료의 이동(흐름)이나 연관관계를 나타낸다. • 화살표 위에 자료의 이름을 기입한다.	물품 코드 →	
자료 저장소 (Data Store)	• 시스템에서의 자료 저장소(파일, 데이터베이스)를 나타낸다. • 도형 안에 자료 저장소 이름을 기입한다.	물품대장	ID 물품대장
단말 (Terminator)	• 시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고 출력 데이터를 받는다(정보의 생산자와 소비자). • 도형 안에 이름을 기입한다.	공장	

핵심 018 자료 사전

자료 사전(DD; Data Dictionary)은 자료 흐름도에 있는 자료를 더 자세히 정의하고 기록한 것이며, 이처럼 데이터를 설명하는 데이터를 데이터의 데이터 또는 메타 데이터(Meta Data)라고 한다.

- 자료 흐름도에 시각적으로 표시된 자료에 대한 정보를 체계적이고 조직적으로 모아 개발자나 사용자가 편리하게 사용할 수 있다.

기 호	의 미
=	자료의 정의 : ~로 구성되어 있다(is composed of)
+	자료의 연결 : 그리고(and)
( )	자료의 생략 : 생략 가능한 자료(Optional)
[|]	자료의 선택 : 또는(or)
{ }	자료의 반복 : Iteration of ① $\{\ \}_n$: n번 이상 반복 ② $\{\ \}^n$: 최대로 n번 반복 ③ $\{\ \}_m^n$: m 이상 n 이하로 반복
* *	자료의 설명 : 주석(Comment)

핵심 019 요구사항 분석을 위한 CASE (자동화 도구)

요구사항 분석을 위한 자동화 도구는 요구사항을 자동으로 분석하고, 요구사항 분석 명세서를 기술하도록 개발된 도구를 의미한다.

종류

- SADT(Structured Analysis and Design Technique) : SoftTech 사에서 개발한 것으로 시스템 정의, 소프트웨어 요구사항 분석, 시스템/소프트웨어 설계를 위해 널리 이용되어 온 구조적 분석 및 설계 도구이다.
- SREM(Software Requirements Engineering Methodology) = RSL/REVS
  - TRW 사가 우주 국방 시스템 그룹에 의해 실시간 처리 소프트웨어 시스템에서 요구사항을 명확히 기술하도록 할 목적으로 개발한 것으로, RSL과 REVS를 사용하는 자동화 도구이다.
  - RSL(Requirement Statement Language) : 요소, 속성, 관계, 구조들을 기술하는 요구사항 기술 언어
  - REVS(Requirement Engineering and Validation System) : RSL로 기술된 요구사항들을 자동으로 분석하여 요구사항 분석 명세서를 출력하는 요구사항 분석기
- PSL/PSA : 미시간 대학에서 개발한 것으로 PSL과 PSA를 사용하는 자동화 도구이다.
- TAGS(Technology for Automated Generation of Systems) : 시스템 공학 방법 응용에 대한 자동 접근 방법으로, 개발 주기의 전 과정에 이용할 수 있는 통합 자동화 도구이다.

핵심 020 HIPO

HIPO(Hierarchy Input Process Output)는 시스템의 분석 및 설계나 문서화할 때 사용되는 기법으로, 시스템 실행 과정인 입력, 처리, 출력의 기능을 나타낸다.

- 기본 시스템 모델은 입력, 처리, 출력으로 구성되며, 하향식 소프트웨어 개발을 위한 문서화 도구이다.
- 체계적인 문서 관리가 가능하다.
- 기호, 도표 등을 사용하므로 보기 쉽고 이해하기도 쉽다.
- 기능과 자료의 의존 관계를 동시에 표현할 수 있다.
- 변경, 유지보수가 용이하다.
- 시스템의 기능을 여러 개의 고유 모듈들로 분할하여 이들 간의 인터페이스를 계층 구조로 표현한 것을 HIPO Chart라고 한다.

HIPO Chart의 종류

- **가시적 도표(도식 목차)** : 시스템의 전체적인 기능과 흐름을 보여주는 계층(Tree) 구조도
- **총체적 도표(총괄도표, 개요 도표)** : 프로그램을 구성하는 기능을 기술한 것으로 입력, 처리, 출력에 대한 전반적인 정보를 제공하는 도표
- **세부적 도표(상세 도표)** : 총체적 도표에 표시된 기능을 구성하는 기본 요소들을 상세히 기술하는 도표

핵심 021 UML(Unified Modeling Language)의 개요

UML은 시스템 분석, 설계, 구현 등 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어이다.

- UML은 Rumbaugh(OMT), Booch, Jacobson 등의 객체지향 방법론의 장점을 통합하였으며, 객체 기술에 관한 국제표준화기구인 OMG(Object Management Group)에서 표준으로 지정하였다.
- UML을 이용하여 시스템의 구조를 표현하는 6개의 구조 다이어그램과 시스템의 동작을 표현하는 7개의 행위 다이어그램을 작성할 수 있다.
- 각각의 다이어그램은 사물과 사물 간의 관계를 용도에 맞게 표현한다.
- UML의 구성 요소에는 사물(Things), 관계(Relationships), 다이어그램(Diagram) 등이 있다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 022 관계(Relationships)

관계는 사물과 사물 사이의 연관성을 표현하는 것으로, 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계, 실체화 관계 등이 있다.

연관(Association) 관계

연관 관계는 2개 이상의 사물이 서로 관련되어 있음을 표현한다.

예제1 사람이 집을 소유하는 관계이다. 사람은 자기가 소유하고 있는 집에 대해 알고 있지만 집은 누구에 의해 자신이 소유되고 있는지 모른다는 의미이다.

해설

- '사람' 쪽에 표기된 다중도가 '1'이므로 집은 한 사람에 의해서만 소유될 수 있다.
- '집' 쪽에 표기된 다중도가 '1'이므로 사람은 집을 하나만 소유할 수 있다.

예제2 선생님은 학생을 가르치고 학생은 선생님으로부터 가르침을 받는 것과 같이 선생님과 학생은 서로 관계가 있다.

해설

- '선생님' 쪽에 표기된 다중도가 '1..*'이므로 학생은 한 명 이상의 선생님으로부터 가르침을 받는다.
- '학생' 쪽에 표기된 다중도가 '1..*'이므로 선생님은 한 명 이상의 학생을 가르친다.

집합(Aggregation) 관계

집합 관계는 하나의 사물이 다른 사물에 포함되어 있는 관계를 표현한다.

예제 프린터는 컴퓨터에 연결해서 사용할 수 있으며, 다른 컴퓨터에 연결해서 사용할 수도 있다.

포함(Composition) 관계

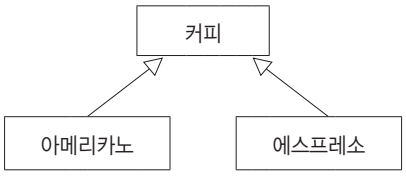
포함 관계는 집합 관계의 특수한 형태로, 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계를 표현한다.

예제 문을 열 수 있는 키는 하나이며, 해당 키로 다른 문은 열 수 없다. 문이 없어지면 키도 더 이상 필요하지 않다.

일반화(Generalization) 관계

일반화 관계는 하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현한다.

예제 아메리카노와 에스프레소는 커피이다. 다시 말하면, 커피에는 아메리카노와 에스프레소가 있다.



의존(Dependency) 관계

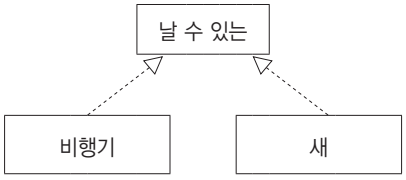
의존 관계는 연관 관계와 같이 사물 사이에 서로 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계를 표현한다.

예제 등급이 높으면 할인율을 적용하고, 등급이 낮으면 할인율을 적용하지 않는다.

실체화(Realization) 관계

실체화 관계는 사물이 할 수 있거나 해야 하는 기능(행위, 인터페이스)으로 서로를 그룹화 할 수 있는 관계를 표현한다.

예제 비행기는 날 수 있고 새도 날 수 있다. 그러므로 비행기와 새는 날 수 있다는 행위로 그룹화 할 수 있다.



시험에 나오는 것만 공부한다! 시나공시리즈

핵심 023 다이어그램(Diagram)

다이어그램은 사물과 관계를 도형으로 표현한 것이다.

- 여러 관점에서 시스템을 가시화한 뷰(View)를 제공함으로써 의사소통에 도움을 준다.
- 정적 모델링에서는 주로 구조적 다이어그램을 사용하고 동적 모델링에서는 주로 행위 다이어그램을 사용한다.
- 구조적(Structural) 다이어그램의 종류

클래스 다이어그램 (Class Diagram)	• 클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다. • 시스템의 구조를 파악하고 구조상의 문제점을 도출할 수 있다.
객체 다이어그램 (Object Diagram)	• 클래스에 속한 사물(객체)들, 즉 인스턴스(Instance)를 특정 시점의 객체와 객체 사이의 관계로 표현한다. • 럼바우(Rumbaugh) 객체지향 분석 기법에서 객체 모델링에 활용된다.
컴포넌트 다이어그램 (Component Diagram)	• 실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현한다. • 구현 단계에서 사용되는 다이어그램이다.
배치 다이어그램 (Deployment Diagram)	• 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현한다. • 노드와 의사소통(통신) 경로로 표현한다. • 구현 단계에서 사용되는 다이어그램이다.
복합체 구조 다이어그램 (Composite Structure Diagram)	클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현한다.
패키지 다이어그램 (Package Diagram)	유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현한다.

- 행위(Behavioral) 다이어그램의 종류

유스케이스 다이어그램 (Use Case Diagram)	• 사용자의 요구를 분석하는 것으로 기능 모델링 작업에 사용한다. • 사용자(Actor)와 사용 사례(Use Case)로 구성되며, 사용 사례 간에는 여러 형태의 관계로 이루어진다.
시퀀스 다이어그램 (Sequence Diagram)	상호 작용하는 시스템이나 객체들이 주고받는 메시지를 표현한다.
커뮤니케이션 다이어그램 (Communication Diagram)	시퀀스 다이어그램과 같이 동작에 참여하는 객체들이 주고받는 메시지를 표현하는데, 메시지뿐만 아니라 객체들 간의 연관까지 표현한다.
상태 다이어그램 (State Diagram)	• 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는지를 표현한다. • 럼바우(Rumbaugh) 객체지향 분석 기법에서 동적 모델링에 활용된다.
활동 다이어그램 (Activity Diagram)	시스템이 어떤 기능을 수행하는지 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현한다.
상호작용 개요 다이어그램(Interaction Overview Diagram)	상호작용 다이어그램 간의 제어 흐름을 표현한다.
타이밍 다이어그램 (Timing Diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

핵심 024 스테레오 타입(Stereotype)

스테레오 타입은 UML에서 표현하는 기본 기능 외에 추가적인 기능을 표현하기 위해 사용한다.

- 길러멧(Guilemet)이라고 부르는 겹화살괄호(《 》) 사이에 표현할 형태를 기술한다.
- 주로 표현되는 형태는 다음과 같다.

《include》	연결된 다른 UML 요소에 대해 포함 관계에 있는 경우
《extend》	연결된 다른 UML 요소에 대해 확장 관계에 있는 경우
《interface》	인터페이스를 정의하는 경우
《exception》	예외를 정의하는 경우
《constructor》	생성자 역할을 수행하는 경우

핵심 025 시퀀스(Sequence) 다이어그램

시퀀스 다이어그램은 시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호 작용하는 과정을 액터, 객체, 메시지 등의 요소를 사용하여 그림으로 표현한 것이다.

시퀀스 다이어그램의 구성 요소

시퀀스 다이어그램은 액터, 객체, 생명선, 실행, 메시지 등으로 구성된다.

액터(Actor)	시스템으로부터 서비스를 요청하는 외부 요소로, 사람이나 외부 시스템을 의미함
객체(Object)	메시지를 주고받는 주체
생명선(Lifeline)	객체가 메모리에 존재하는 기간으로, 객체 아래쪽에 점선을 그어 표현함
실행 상자(Active Box)	객체가 메시지를 주고받으며 구동되고 있음을 표현함
메시지(Message)	객체가 상호 작용을 위해 주고받는 메시지

핵심 026 사용자 인터페이스(UI, User Interface)

- 사용자 인터페이스(UI)는 사용자와 시스템 간의 상호 작용이 원활하게 이뤄지도록 도와주는 장치나 소프트웨어를 의미한다.
- 사용자의 만족도에 가장 큰 영향을 미치는 중요한 요소로, 소프트웨어 영역 중 변경이 가장 많이 발생한다.
- 사용자의 편리성과 가독성을 높임으로써 작업 시간을 단축시키고 업무에 대한 이해도를 높여준다.
- 최소한의 노력으로 원하는 결과를 얻을 수 있게 한다.
- 사용자 인터페이스의 기본 원칙
  - 직관성 : 누구나 쉽게 이해하고 사용할 수 있어야 한다.
  - 유효성 : 사용자의 목적을 정확하고 완벽하게 달성해야 한다.
  - 학습성 : 누구나 쉽게 배우고 익힐 수 있어야 한다.
  - 유연성 : 사용자의 요구사항을 최대한 수용하고 실수를 최소화해야 한다.
- 사용자 인터페이스 개발 시스템의 기능
  - 사용자의 입력을 검증할 수 있어야 한다.
  - 에러 처리와 그와 관련된 에러 메시지를 표시할 수 있어야 한다.
  - 도움과 프롬프트(Prompt)를 제공해야 한다.

핵심 027 UI 설계 도구

UI 설계 도구

UI 설계 도구는 사용자의 요구사항에 맞게 UI의 화면 구조나 화면 배치 등을 설계할 때 사용하는 도구로, 종류에는 와이어프레임, 목업, 스토리보드, 프로토타입, 유스케이스 등이 있다.

와이어프레임(Wireframe)

- 와이어프레임은 기획 단계의 초기에 제작하는 것으로, 페이지에 대한 개략적인 레이아웃이나 UI 요소 등에 대한 뼈대를 설계하는 단계이다.
- 개발자나 디자이너 등이 레이아웃을 협의하거나 현재 진행 상태 등을 공유하기 위해 와이어프레임을 사용한다.
- 와이어프레임 툴 : 손그림, 파워포인트, 키노트, 스케치, 일러스트, 포토샵 등

목업(Mockup)

- 목업은 디자인, 사용 방법 설명, 평가 등을 위해 와이어프레임보다 좀 더 실제 화면과 유사하게 만든 정적인 형태의 모형이다.
- 시각적으로만 구성 요소를 배치하는 것으로 실제로 구현되지는 않는다.
- 목업 툴 : 파워 목업, 발사믹 목업 등

스토리보드(Story Board)

- 스토리보드는 와이어프레임에 콘텐츠에 대한 설명, 페이지 간 이동 흐름 등을 추가한 문서이다.
- 디자이너와 개발자가 최종적으로 참고하는 작업 지침서로, 정책, 프로세스, 콘텐츠 구성, 와이어프레임, 기능 정의 등 서비스 구축을 위한 모든 정보가 들어 있다.
- 스토리보드 툴 : 파워포인트, 키노트, 스케치, Axure 등

프로토타입(Prototype)

- 프로토타입은 와이어프레임이나 스토리보드 등에 인터랙션을 적용함으로써 실제 구현된 것처럼 테스트가 가능한 동적인 형태의 모형이다.
- 프로토타입은 사용성 테스트나 작업자 간 서비스 이해를 위해 작성하는 샘플이다.

유스케이스(Use Case)

- 유스케이스는 사용자 측면에서의 요구사항으로, 사용자가 원하는 목표를 달성하기 위해 수행할 내용을 기술한다.
- 사용자의 요구사항을 빠르게 파악함으로써 프로젝트의 초기에 시스템의 기능적인 요구를 결정하고 그 결과를 문서화할 수 있다.

핵심 028 UI 요구사항 확인

UI 요구사항 확인은 새로 개발할 시스템에 적용할 UI 관련 요구사항을 조사해서 작성하는 단계로, 다양한 경로를 통해 사용자의 요구사항을 조사하고 분석한 후 작성해야 한다.

- UI 요구사항 확인 순서

목표 정의	목표 정의 단계에서는 사용자들을 대상으로 인터뷰를 진행한 후 사용자들의 의견이 수렴된 비즈니스 요구사항을 정의한다. • 인터뷰를 통해 사업적, 기술적인 요구사항을 명확히 이해한다.
활동 사항 정의	활동 사항 정의 단계에서는 조사한 요구사항을 토대로 앞으로 해야 할 활동 사항을 정의한다. • 사용자와 회사의 비전을 일치시키는 작업을 진행한다. • 기술의 발전 가능성을 파악하고 UI 디자인의 방향을 제시한다.
UI 요구사항 작성	UI 요구사항을 작성할 때는 여러 경로를 통해 수집된 사용자들의 요구사항을 검토하고 분석하여 UI 개발 목적에 맞게 작성해야 한다. • UI 요구사항은 반드시 실사용자 중심으로 작성되어야 한다.

- UI 요구사항 작성 순서

요구사항 요소 확인	파악된 요구사항 요소의 종류와 각각의 표현 방식 등을 검토한다. • 요구사항 요소 : 데이터 요구, 기능 요구, 제품/서비스의 품질, 제약 사항
정황 시나리오 작성	정황 시나리오는 사용자의 요구사항을 도출하기 위해 작성하는 것으로, 사용자가 목표를 달성하기 위해 수행하는 방법을 순차적으로 묘사한 것이다. • 정황 시나리오는 요구사항 정의에 사용되는 초기 시나리오이다. • 정황 시나리오는 개발하는 서비스의 모습을 상상하는 첫 번째 단계로 사용자 관점에서 시나리오를 작성해야 한다. • 사용자가 주로 사용하는 기능 위주로 작성해야 하며, 함께 발생되는 기능들은 하나의 시나리오에 통합한다.
요구사항 작성	요구사항은 정황 시나리오를 토대로 작성한다.

핵심 029 품질 요구사항

소프트웨어의 품질은 소프트웨어의 기능, 성능, 만족도 등 소프트웨어에 대한 요구사항이 얼마나 충족하는가를 나타내는 소프트웨어 특성의 총체이다.

- 소프트웨어의 품질은 사용자의 요구사항을 충족시킴으로써 확립된다.
- ISO/IEC 9126 : 소프트웨어의 품질 특성과 평가를 위한 표준 지침으로서 국제 표준으로 널리 사용됨
- ISO/IEC 25010 : 소프트웨어 제품에 대한 국제 표준으로, 2011년에 ISO/IEC 9126을 개정하여 만들었음
- ISO/IEC 12119 : ISO/IEC 9126을 준수한 품질 표준으로, 테스트 절차를 포함하여 규정함
- ISO/IEC 14598 : 소프트웨어 품질의 측정과 평가에 필요 절차를 규정한 표준으로, 개발자, 구매자, 평가자 별로 수행해야 할 제품 평가 활동을 규정함

핵심 030 ISO/IEC 9126의 소프트웨어 품질 특성

기능성 (Functionality)	• 소프트웨어가 사용자의 요구사항을 정확하게 만족하는 기능을 제공하는지 여부를 나타낸다. • 하위 특성 : 적절성/적합성(Suitability), 정밀성/정확성(Accuracy), 상호 운용성(Interoperability), 보안성(Security), 준수성(Compliance)
신뢰성 (Reliability)	• 소프트웨어가 요구된 기능을 정확하고 일관되게 오류 없이 수행할 수 있는 정도를 나타낸다. • 하위 특성 : 성숙성(Maturity), 고장 허용성(Fault Tolerance), 회복성(Recoverability)
사용성 (Usability)	• 사용자와 컴퓨터 사이에 발생하는 어떠한 행위에 대하여 사용자가 정확하게 이해하고 사용하며, 향후 다시 사용하고 싶은 정도를 나타낸다. • 하위 특성 : 이해성(Understandability), 학습성(Learnability), 운용성(Operability), 친밀성(Attractiveness)
효율성 (Efficiency)	• 사용자가 요구하는 기능을 할당된 시간 동안 한정된 자원으로 얼마나 빨리 처리할 수 있는지 정도를 나타낸다. • 하위 특성 : 시간 효율성(Time Behaviour), 자원 효율성(Resource Behaviour)
유지 보수성 (Maintainability)	• 환경의 변화 또는 새로운 요구사항이 발생했을 때 소프트웨어를 개선하거나 확장할 수 있는 정도를 나타낸다. • 하위 특성 : 분석성(Analyzability), 변경성(Changeability), 안정성(Stability), 시험성(Testability)
이식성 (Portability)	• 소프트웨어가 다른 환경에서도 얼마나 쉽게 적용할 수 있는지 정도를 나타낸다. • 하위 특성 : 적용성(Adaptability), 설치성(Installability), 대체성(Replaceability), 공존성(Co-existence)

핵심 031 UI 프로토타입 제작 및 검토

UI 프로토타입의 개요

- 프로토타입은 사용자 요구사항을 기반으로 실제 동작하는 것처럼 만든 동적인 형태의 모형으로, 테스트가 가능하다.
- 프로토타입은 사용자의 요구사항을 개발자가 맞게 해석했는지 검증하기 위한 것으로, 최대한 간단하게 만들어야 한다.
- 프로토타입은 일부 핵심적인 기능만을 제공하지만 최종 제품의 작동 방식을 이해시키는데 필요한 기능은 반드시 포함되어야 한다.
- 프로토타이핑 및 테스트를 거치지 않고는 실제 사용자와 제품 간의 상호 작용 방식을 예측하기 어려우므로 실제 사용자를 대상으로 테스트하는 것이 좋다.

UI 프로토타입의 장 · 단점

장점	• 사용자를 설득하고 이해시키기 쉽다. • 요구사항과 기능의 불일치 등으로 인한 혼선을 예방할 수 있어 개발 시간을 줄일 수 있다. • 사전에 오류를 발견할 수 있다.
단점	• 프로토타입에 사용자의 모든 요구사항을 반영하기 위한 반복적인 개선 및 보완 작업 때문에 작업 시간을 증가 시킬 수 있고, 필요 이상으로 자원을 소모할 수 있다. • 부분적으로 프로토타이핑을 진행하다보면 중요한 작업이 생략될 수 있다.

프로토타이핑의 종류

페이퍼 프로토타입 (Paper Prototype)	• 아날로그적인 방법으로, 스케치, 그림, 글 등을 이용하여 손으로 직접 작성하는 방법이다. • 제작 기간이 짧은 경우, 제작 비용이 적을 경우, 업무 협의가 빠를 경우 사용한다.
디지털 프로토타입 (Digital Prototype)	• 파워포인트, 아크로뱃, 비지오, 옴니그래플 등과 같은 프로그램을 사용하여 작성하는 방법이다. • 재사용이 필요한 경우, 산출물과 비슷한 효과가 필요한 경우, 숙련된 전문가가 있을 경우 사용한다.

핵심 032 UI 설계서 작성

UI 설계서의 개요

- UI 설계서는 사용자의 요구사항을 바탕으로 UI 설계를 구체화하여 작성하는 문서로, 상세 설계 전에 대표적인 화면들을 설계한다.
- UI 설계서는 기획자, 개발자, 디자이너 등과의 원활한 의사소통을 위해 작성한다.
- 작성 순서

UI 설계서 표지 작성	UI 설계서 표지는 다른 문서와 혼동되지 않도록 프로젝트명 또는 시스템명을 포함시켜 작성한다.
UI 설계서 개정 이력 작성	• UI 설계서 개정 이력은 UI 설계서가 수정될 때마다 어떤 부분이 어떻게 수정되었는지를 정리해 놓은 문서이다. • 처음 작성 시 첫 번째 항목을 '초안 작성', 버전(Version)을 1.0으로 설정한다. • UI 설계서에 변경 사항이 있을 때마다 변경 내용을 적고 버전을 0.1씩 높인다.
UI 요구사항 정의서 작성	UI 요구사항 정의서는 사용자의 요구사항을 확인하고 정리한 문서로, 사용자 요구사항의 UI 적용 여부를 요구사항별로 표시한다.
시스템 구조 작성	시스템 구조는 UI 요구사항과 UI 프로토타입에 기초하여 전체 시스템의 구조를 설계한 것으로 사용자의 요구사항이 어떻게 시스템에 적용되는지 알 수 있다.
사이트 맵 (Site Map) 작성	• 사이트 맵은 시스템 구조를 바탕으로 사이트에 표시할 콘텐츠를 한 눈에 알아 볼 수 있도록 메뉴별로 구분하여 설계한 것이다. • 사이트 맵을 작성한 후 사이트 맵의 상세 내용(Site Map Detail)을 표 형태로 작성한다.
프로세스 (Process) 정의서 작성	프로세스 정의서는 사용자 관점에서 사용자가 요구하는 프로세스들을 작업 진행 순서에 맞춰 정리한 것으로 UI의 전체적인 흐름을 파악할 수 있다.
화면 설계	• 화면 설계는 UI 프로토타입과 UI 프로세스를 참고하여 필요한 화면을 페이지별로 설계한 것이다. • 화면을 구분하기 위해 화면별 고유ID를 부여하고 별도 표지를 작성한다.

핵심 033 UI 시나리오 문서

UI 시나리오 문서 개요

- UI 상세 설계는 UI 설계서를 바탕으로 실제 설계 및 구현을 위해 모든 화면에 대한 자세한 설계를 진행하는 단계로, UI 상세 설계를 할 때는 반드시 시나리오를 작성해야 한다.
- UI 시나리오 문서는 사용자 인터페이스의 기능 구조, 대표 화면, 화면 간 인터랙션의 흐름, 다양한 상황에서의 예외 처리 등을 문서로 정리한 것이다.
- UI 설계자 또는 인터랙션 디자이너가 UI 시나리오 문서를 작성하면 그래픽 디자이너가 시나리오를 바탕으로 디자인을 하고 개발자가 UI를 구현한다.

UI 시나리오 문서의 요건

완전성 (Complete)	• 누락되지 않도록 최대한 상세하게 기술해야 한다. • 해당 시스템의 기능보다는 사용자의 태스크에 초점을 맞춰 기술한다.
일관성 (Consistent)	서비스 목표, 시스템 및 사용자의 요구사항, UI 스타일 등이 모두 일관성을 유지해야 한다.
이해성 (Understandable)	• 누구나 쉽게 이해할 수 있도록 설명한다. • 불분명하거나 추상적인 표현은 피한다.
가독성 (Readable)	• 표준화된 템플릿 등을 활용하여 문서를 쉽게 읽을 수 있도록 해야 한다. • v1.0, v2.0 등과 같이 문서 인덱스에 대한 규칙이나 목차를 제공한다. • 읽기 쉽도록 줄 간격, 단락, 들여쓰기 등의 기준을 마련한다. • 시각적인 효과를 위해 여백이나 빈 페이지, 하이라이팅을 일관성 있게 지정한다. • 하이퍼링크 등을 지정하여 문서들이 서로 참조될 수 있도록 지정한다.
수정 용이성 (Modifiable)	시나리오의 수정이나 개선이 쉬워야 한다.
추적 용이성 (Traceable)	변경 사항은 언제, 어떤 부분이, 왜 발생했는지 쉽게 추적할 수 있어야 한다.

핵심 034 HCI / UX / 감성공학

HCI(Human Computer Interaction or Interface)

- HCI는 사람이 시스템을 보다 편리하고 안전하게 사용할 수 있도록 연구하고 개발하는 학문으로, 최종 목표는 시스템을 사용하는데 있어 최적의 사용자 경험(UX)을 만드는 것이다.
- HCI는 어떤 제품이 좋은 제품인지, 어떻게 하면 좋은 제품을 만들 수 있는지 등을 연구한다.

UX(User Experience)

- UX는 사용자가 시스템이나 서비스를 이용하면서 느끼고 생각하게 되는 총체적인 경험을 말한다. 단순히 기능이나 절차상의 만족뿐만 아니라 사용자가 참여, 사용, 관찰하고, 상호 교감을 통해서 알 수 있는 가치 있는 경험을 말한다.
- UX는 기술을 효용성 측면에서만 보는 것이 아니라 사용자의 삶의 질을 향상시키는 하나의 방향으로 보는 새로운 개념이다.
- UI가 사용성, 접근성, 편의성을 중시한다면 UX는 이러한 UI를 통해 사용자가 느끼는 만족이나 감정을 중시한다.
- UX의 특징
  - 주관성(Subjectivity) : 사람들의 개인적, 신체적, 인지적 특성에 따라 다르므로 주관적이다.
  - 정황성(Contextuality) : 경험이 일어나는 상황 또는 주변 환경에 영향을 받는다.
  - 총체성(Holistic) : 개인이 느끼는 총체적인 심리적, 감성적인 결과이다.

감성공학

- 감성공학은 제품이나 작업환경을 사용자의 감성에 알맞도록 설계 및 제작하는 기술로, 인문사회과학, 공학, 의학 등 여러 분야의 학문이 공존하는 종합과학이다.
- '감성'을 과학적으로 측정하기 위해서는 생체계측 기술, 감각계측 기술, 센서, 인공지능, 생체제어 기술 등이 요구된다.
- 감성공학의 목적은 인간의 삶을 편리하고 안전하며 쾌적하게 만드는 것이다.

핵심 035 소프트웨어 아키텍처의 설계

소프트웨어 아키텍처는 소프트웨어의 골격이 되는 기본 구조이자, 소프트웨어를 구성하는 요소들 간의 관계를 표현하는 시스템의 구조 또는 구조체이다.

- 소프트웨어 개발 시 적용되는 원칙과 지침이며, 이해관계자들의 의사소통 도구로 활용된다.
- 소프트웨어 아키텍처 설계의 기본 원리로는 모듈화, 추상화, 단계적 분해, 정보은닉이 있다.

※ 상위 설계와 하위 설계

소프트웨어 개발의 설계 단계는 크게 상위 설계와 하위 설계로 구분할 수 있다.

	상위 설계	하위 설계
별칭	아키텍처 설계, 예비 설계	모듈 설계, 상세 설계
설계 대상	시스템의 전체적인 구조	시스템의 내부 구조 및 행위
세부 목록	구조, DB, 인터페이스	컴포넌트, 자료 구조, 알고리즘

핵심 036 소프트웨어 아키텍처 설계의 기본 원리

모듈화 (Modularity)	모듈화란 소프트웨어의 성능을 향상시키거나 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 시스템의 기능들을 모듈 단위로 나누는 것을 의미한다.
추상화 (Abstraction)	추상화는 문제의 전체적이고 포괄적인 개념을 설계한 후 차례로 세분화하여 구체화시켜 나가는 것이다.
단계적 분해 (Stepwise Refinement)	단계적 분해는 Niklaus Wirth에 의해 제안된 하향식 설계 전략으로, 문제를 상위의 중요 개념으로부터 하위의 개념으로 구체화시키는 분할 기법이다.
정보 은닉 (Information Hiding)	정보 은닉은 한 모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법이다.

핵심 037 협약(Contract)에 의한 설계

컴포넌트를 설계할 때 클래스에 대한 여러 가정을 공유할 수 있도록 명세한 것으로, 소프트웨어 컴포넌트에 대한 정확한 인터페이스를 명세한다.

- 협약에 의한 설계 시 명세에 포함될 조건에는 선행 조건, 결과 조건, 불변 조건이 있다.

선행 조건 (Precondition)	오퍼레이션이 호출되기 전에 참이 되어야 할 조건
결과 조건 (Postcondition)	오퍼레이션이 수행된 후 만족되어야 할 조건
불변 조건 (Invariant)	오퍼레이션이 실행되는 동안 항상 만족되어야 할 조건

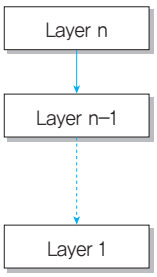
핵심 038 아키텍처 패턴(Patterns)의 개요

아키텍처 패턴은 아키텍처를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제를 의미한다.

- 아키텍처 패턴은 소프트웨어 시스템의 구조를 구성하기 위한 기본적인 윤곽을 제시한다.
- 아키텍처 패턴에는 서브시스템들과 그 역할이 정의되어 있으며, 서브시스템 사이의 관계와 여러 규칙 · 지침 등이 포함되어 있다.
- 아키텍처 패턴을 아키텍처 스타일 또는 표준 아키텍처라고도 한다.
- 아키텍처 패턴의 장점
  - 시행착오를 줄여 개발 시간을 단축시키고, 고품질의 소프트웨어를 생산할 수 있다.
  - 검증된 구조로 개발하기 때문에 안정적인 개발이 가능하다.
  - 이해관계자들이 공통된 아키텍처를 공유할 수 있어 의사소통이 간편해진다.
  - 시스템의 구조를 이해하는 것이 쉬워 개발에 참여하지 않은 사람도 손쉽게 유지보수를 수행할 수 있다.
  - 시스템의 특성을 개발 전에 예측하는 것이 가능해진다.

핵심 039 레이어 패턴(Layers pattern)

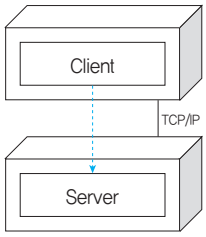
레이어 패턴은 시스템을 계층(Layer)으로 구분하여 구성하는 고전적인 방법 중의 하나다.



- 레이어 패턴은 각각의 서브시스템들이 계층 구조를 이루며, 상위 계층은 하위 계층에 대한 서비스 제공자가 되고, 하위 계층은 상위 계층의 클라이언트가 된다.
- 레이어 패턴은 서로 마주보는 두 개의 계층 사이에서만 상호작용이 이루어지며, 변경 사항을 적용할 때도 서로 마주보는 두 개의 계층에만 영향을 미치므로 변경 작업이 용이하다.
- 레이어 패턴은 특정 계층만을 교체해 시스템을 개선하는 것이 가능하다.
- 대표적으로 OSI 참조 모델이 있다.

핵심 040 클라이언트-서버 패턴 (Client-Server Pattern)

클라이언트-서버 패턴은 하나의 서버 컴포넌트와 다수의 클라이언트 컴포넌트로 구성되는 패턴이다.



- 클라이언트-서버 패턴에서 사용자는 클라이언트와만 의사소통을 한다. 즉 사용자가 클라이언트를 통해 서버에 요청하고 클라이언트가 응답을 받아 사용자에게 제공하는 방식으로 서비스를 제공한다.
- 서버는 클라이언트의 요청에 대비해 항상 대기 상태를 유지해야 한다.
- 클라이언트나 서버는 요청과 응답을 받기 위해 동기화되는 경우를 제외하고는 서로 독립적이다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 041 파이프-필터 패턴 (Pipe-Filter Pattern)

파이프-필터 패턴은 데이터 스트림 절차의 각 단계를 필터(Filter) 컴포넌트로 캡슐화하여 파이프(Pipe)를 통해 데이터를 전송하는 패턴이다.

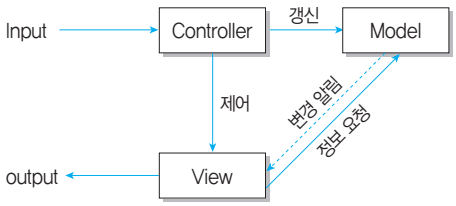
- 필터 컴포넌트는 재사용성이 좋고, 추가가 쉬워 확장이 용이하다.
- 필터 컴포넌트들을 재배치하여 다양한 파이프라인을 구축하는 것이 가능하다.
- 파이프-필터 패턴은 데이터 변환, 버퍼링, 동기화 등에 주로 사용된다.
- 대표적으로 UNIX의 쉘(Shell)이 있다.

핵심 042 모델-뷰-컨트롤러 패턴(Model-View-Controller Pattern)

모델-뷰-컨트롤러 패턴은 서브시스템을 3개의 부분으로 구조화하는 패턴이며, 각 부분의 역할은 다음과 같다.

모델(Model)	서브시스템의 핵심 기능과 데이터를 보관한다.
뷰(View)	사용자에게 정보를 표시한다.
컨트롤러 (Controller)	사용자로부터 받은 입력을 처리한다.

- 모델-뷰-컨트롤러 패턴의 각 부분은 별도의 컴포넌트로 분리되어 있으므로 서로 영향을 받지 않고 개발 작업을 수행할 수 있다.
- 모델-뷰-컨트롤러 패턴에서는 여러 개의 뷰를 만들 수 있으므로 한 개의 모델에 대해 여러 개의 뷰를 필요로 하는 대화형 애플리케이션에 적합하다.



핵심 043 기타 패턴

마스터-슬레이브 패턴 (Master-Slave Pattern)	• 마스터 컴포넌트에서 슬레이브 컴포넌트로 작업을 분할한 후, 슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받는 방식으로 작업을 수행하는 패턴이다. • 장애 허용 시스템과 병렬 컴퓨팅 시스템에서 주로 활용된다.
브로커 패턴 (Broker Pattern)	• 사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커 컴포넌트가 요청에 맞는 컴포넌트와 사용자를 연결해 준다. • 분산 환경 시스템에서 주로 활용된다.
피어-투-피어 패턴 (Peer-To-Peer Pattern)	피어(Peer)를 하나의 컴포넌트로 간주하며, 각 피어는 서비스를 호출하는 클라이언트가 될 수도, 서비스를 제공하는 서버가 될 수도 있는 패턴이다.
이벤트-버스 패턴 (Event-Bus Pattern)	소스가 특정 채널에 이벤트 메시지를 발행(Publish)하면, 해당 채널을 구독(Subscribe)한 리스너들이 메시지를 받아 이벤트를 처리하는 방식이다.
블랙보드 패턴 (Blackboard Pattern)	• 모든 컴포넌트들이 공유 데이터 저장소와 블랙보드 컴포넌트에 접근이 가능한 형태로, 컴포넌트들은 검색을 통해 블랙보드에서 원하는 데이터를 찾을 수 있다. • 음성 인식, 차량 식별, 신호 해석 등에 주로 활용된다.
인터프리터 패턴 (Interpreter Pattern)	프로그램 코드의 각 라인을 수행하는 방법을 지정하고, 기호마다 클래스를 갖도록 구성된다.

핵심 044 객체(Object)

객체는 데이터와 데이터를 처리하는 함수를 묶어 놓은(캡슐화한) 하나의 소프트웨어 모듈이다.

데이터	• 객체가 가지고 있는 정보로 속성이나 상태, 분류 등을 나타낸다. • 속성(Attribute), 상태, 변수, 상수, 자료 구조라고도 한다.
함수	• 객체가 수행하는 기능으로 객체가 갖는 데이터(속성, 상태)를 처리하는 알고리즘이다. • 객체의 상태를 참조하거나 변경하는 수단이 되는 것으로 메소드(Method, 행위), 서비스(Service), 동작(Operation), 연산이라고도 한다.

- 객체의 특성
  - 객체는 독립적으로 식별 가능한 이름을 가지고 있다.
  - 객체가 가질 수 있는 조건을 상태(State)라고 하는데, 일반적으로 상태는 시간에 따라 변한다.
  - 객체와 객체는 상호 연관성에 의한 관계가 형성된다.
  - 객체가 반응할 수 있는 메시지(Message)의 집합을 행위라고 하며, 객체는 행위의 특징을 나타낼 수 있다.
  - 객체는 일정한 기억장소를 가지고 있다.
- 객체의 메소드는 다른 객체로부터 메시지를 받았을 때 정해진 기능을 수행한다.

핵심 045 클래스(Class)

클래스는 공통된 속성과 연산(행위)을 갖는 객체의 집합으로, 객체의 일반적인 타입(Type)을 의미한다.

- 클래스는 각각의 객체들이 갖는 속성과 연산을 정의하고 있는 틀이다.
- 클래스는 객체지향 프로그램에서 데이터를 추상화하는 단위이다.
- 클래스에 속한 각각의 객체를 인스턴스(Instance)라 하며, 클래스로부터 새로운 객체를 생성하는 것을 인스턴스화(Instantiation)라고 한다.

핵심 046 캡슐화(Encapsulation)

캡슐화는 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것을 의미한다.

- 캡슐화된 객체는 인터페이스를 제외한 세부 내용이 은폐(정보 은닉)되어 외부에서의 접근이 제한적이기 때문에 외부 모듈의 변경으로 인한 파급 효과가 적다.
- 캡슐화된 객체들은 재사용이 용이하다.
- 객체들 간의 메시지를 주고받을 때 상대 객체의 세부 내용은 알 필요가 없으므로 인터페이스가 단순해지고, 객체 간의 결합도가 낮아진다.

핵심 047 상속(Inheritance)

상속은 이미 정의된 상위 클래스(부모 클래스)의 모든 속성과 연산을 하위 클래스(자식 클래스)가 물려받는 것이다.

- 상속을 이용하면 하위 클래스는 상위 클래스의 모든 속성과 연산을 자신의 클래스 내에서 다시 정의하지 않고서도 즉시 자신의 속성으로 사용할 수 있다.
- 하위 클래스는 상위 클래스로부터 상속받은 속성과 연산 외에 새로운 속성과 연산을 첨가하여 사용할 수 있다.

핵심 048 연관성(Relationship)

연관성은 두 개 이상의 객체(클래스)들이 상호 참조하는 관계를 말하며 종류는 다음과 같다.

종류	의미	특징
is member of	연관화 (Association)	2개 이상의 객체가 상호 관련되어 있음을 의미함
is instance of	분류화 (Classfication)	동일한 형의 특성을 갖는 객체들을 모아 구성하는 것
is part of	집단화 (Aggregation)	관련 있는 객체들을 묶어 하나의 상위 객체를 구성하는 것
is a	일반화 (Generalization)	공통적인 성질들로 추상화한 상위 객체를 구성하는 것
	특수화/상세화 (Specialization)	상위 객체를 구체화하여 하위 객체를 구성하는 것

핵심 049 객체지향 분석의 방법론

객체지향 분석을 위한 여러 방법론이 제시되었으며 각 방법론은 다음과 같다.

- Rumbaugh(럼바우) 방법 : 가장 일반적으로 사용되는 방법으로 분석 활동을 객체 모델, 동적 모델, 기능 모델로 나누어 수행하는 방법이다.

시험에 나오는 것만 공부한다! 시나공시리즈

- Booch(부치) 방법 : 미시적(Micro) 개발 프로세스와 거시적(Macro) 개발 프로세스를 모두 사용하는 분석 방법으로, 클래스와 객체들을 분석 및 식별하고 클래스의 속성과 연산을 정의한다.
- Jacobson 방법 : Use Case를 강조하여 사용하는 분석 방법이다.
- Coad와 Yourdon 방법 : E-R 다이어그램을 사용하여 객체의 행위를 모델링하며, 객체 식별, 구조 식별, 주제 정의, 속성과 인스턴스 연결 정의, 연산과 메시지 연결 정의 등의 과정으로 구성하는 기법이다.
- Wirfs-Brock 방법 : 분석과 설계 간의 구분이 없고, 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행하는 기법이다.

핵심 050 럼바우(Rumbaugh)의 분석 기법

럼바우의 분석 기법은 모든 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링하는 기법으로, 객체 모델링 기법(OMT, Object-Modeling Technique)이라고도 한다.

- 분석 활동은 '객체 모델링 → 동적 모델링 → 기능 모델링' 순으로 통해 이루어진다.

객체 모델링 (Object Modeling)	정보 모델링이라고도 하며, 시스템에서 요구되는 객체를 찾아내어 속성과 연산 식별 및 객체들 간의 관계를 규정하여 객체 다이어그램으로 표시하는 것이다.
동적 모델링 (Dynamic Modeling)	상태 다이어그램(상태도)을 이용하여 시간의 흐름에 따른 객체들 간의 제어 흐름, 상호 작용, 동작 순서 등의 동적인 행위를 표현하는 모델링이다.
기능 모델링 (Functional Modeling)	자료 흐름도(DFD)를 이용하여 다수의 프로세스들 간의 자료 흐름을 중심으로 처리 과정을 표현한 모델링이다.

핵심 051 객체지향 설계 원칙

객체지향 설계 원칙은 시스템 변경이나 확장에 유연한 시스템을 설계하기 위해 지켜야 할 다섯 가지 원칙으로, 다섯 가지 원칙의 앞 글자를 따 SOLID 원칙이라고도 불린다.

단일 책임 원칙 (SRP, Single Responsibility Principle)	객체는 단 하나의 책임만 가져야 한다는 원칙이다.
개방-폐쇄 원칙 (OCP, Open-Closed Principle)	기존의 코드를 변경하지 않고 기능을 추가할 수 있도록 설계해야 한다는 원칙이다.
리스코프 치환 원칙 (LSP, Liskov Substitution Principle)	자식 클래스는 최소한 자신의 부모 클래스에서 가능한 행위는 수행할 수 있어야 한다는 설계 원칙이다.
인터페이스 분리 원칙 (ISP, Interface Segregation Principle)	자신이 사용하지 않는 인터페이스와 의존 관계를 맺거나 영향을 받지 않아야 한다는 원칙이다.
의존 역전 원칙 (DIP, Dependency Inversion Principle)	각 객체들 간의 의존 관계가 성립될 때, 추상성이 낮은 클래스보다 추상성이 높은 클래스와 의존 관계를 맺어야 한다는 원칙이다.

핵심 052 결합도(Coupling)

결합도는 모듈 간에 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계를 의미한다.

- 다양한 결합으로 모듈을 구성할 수 있으나 결합도가 약할수록 품질이 높고, 강할수록 품질이 낮다.
- 결합도가 강하면 시스템 구현 및 유지보수 작업이 어렵다.
- 결합도의 종류에는 자료 결합도, 스탬프 결합도, 제어 결합도, 외부 결합도, 공통 결합도, 내용 결합도가 있으며 결합도의 정도는 다음과 같다.

자료 결합도	스탬프 결합도	제어 결합도	외부 결합도	공통 결합도	내용 결합도

결합도 약함 ← → 결합도 강함

자료 결합도 (Data Coupling)	모듈 간의 인터페이스가 자료 요소로만 구성될 때의 결합도
스탬프(검인) 결합도 (Stamp Coupling)	모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도
제어 결합도 (Control Coupling)	어떤 모듈이 다른 모듈 내부의 논리적인 흐름을 제어하기 위해 제어 신호를 이용하여 통신하거나 제어 요소(Function Code, Switch, Tag, Flag)를 전달하는 결합도
외부 결합도 (External Coupling)	어떤 모듈에서 선언한 데이터(변수)를 외부의 다른 모듈에서 참조할 때의 결합도
공통(공유) 결합도 (Common Coupling)	공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도
내용 결합도 (Content Coupling)	한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도

핵심 053 응집도(Cohesion)

응집도는 정보 은닉 개념을 확장한 것으로, 명령어나 호출문 등 모듈의 내부 요소들의 서로 관련되어 있는 정도, 즉 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미한다.

- 다양한 기준으로 모듈을 구성할 수 있으나 응집도가 강할수록 품질이 높고, 약할수록 품질이 낮다.
- 응집도의 종류에는 기능적 응집도, 순차적 응집도, 교환(통신)적 응집도, 절차적 응집도, 시간적 응집도, 논리적 응집도, 우연적 응집도가 있으며 응집도의 정도는 다음과 같다.

기능적 응집도	순차적 응집도	교환적 응집도	절차적 응집도	시간적 응집도	논리적 응집도	우연적 응집도

응집도 강함 ← → 응집도 약함

기능적 응집도 (Functional Cohesion)	모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도
순차적 응집도 (Sequential Cohesion)	모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도
교환(통신)적 응집도 (Communication Cohesion)	동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도
절차적 응집도 (Procedural Cohesion)	모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도
시간적 응집도 (Temporal Cohesion)	특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우의 응집도
논리적 응집도 (Logical Cohesion)	유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도
우연적 응집도 (Coincidental Cohesion)	모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우의 응집도

핵심 054 N-S 차트 (Nassi-Schneiderman Chart)

N-S 차트는 논리의 기술에 중점을 둔 도형을 이용한 표현 방법으로 박스 다이어그램, Chapin Chart라고도 한다.

- 연속, 선택 및 다중 선택, 반복 등의 제어 논리 구조를 표현한다.
- GOTO나 화살표를 사용하지 않는다.
- 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는 데 적합하다.
- 선택과 반복 구조를 시각적으로 표현한다.
- 이해하기 쉽고, 코드 변환이 용이하다.
- 읽기는 쉽지만 작성하기가 어려우며, 임의로 제어를 전이하는 것이 불가능하다.
- 총체적인 구조 표현과 인터페이스를 나타내기가 어렵다.
- 단일 입구와 단일 출구로 표현한다.

핵심 055 공통 모듈의 개요

공통 모듈은 여러 프로그램에서 공통적으로 사용할 수 있는 모듈을 의미한다.

- 자주 사용되는 계산식이나 매번 필요한 사용자 인증과 같은 기능들이 공통 모듈로 구성될 수 있다.
- 모듈의 재사용성 확보와 중복 개발 회피를 위해 설계 과정에서 공통 부분을 식별하고 명세를 작성할 필요가 있다.
- 공통 모듈을 구현할 때는 다른 개발자들이 해당 기능을 명확히 이해할 수 있도록 다음의 명세 기법을 준수해야 한다.

정확성 (Correctness)	시스템 구현 시 해당 기능이 필요하다는 것을 알 수 있도록 정확히 작성한다.
명확성 (Clarity)	해당 기능을 이해할 때 중의적으로 해석되지 않도록 명확하게 작성한다.
완전성 (Completeness)	시스템 구현을 위해 필요한 모든 것을 기술한다.
일관성 (Consistency)	공통 기능들 간 상호 충돌이 발생하지 않도록 작성한다.
추적성 (Traceability)	기능에 대한 요구사항의 출처, 관련 시스템 등의 관계를 파악할 수 있도록 작성한다.

핵심 056 재사용(Reuse)

재사용은 비용과 개발 시간을 절약하기 위해 이미 개발된 기능들을 파악하고 재구성하여 새로운 시스템 또는 기능 개발에 사용하기 적합하도록 최적화 시키는 작업이다.

- 재사용을 위해서는 누구나 이해할 수 있고 사용이 가능하도록 사용법을 공개해야 한다.
- 재사용되는 대상은 외부 모듈과의 결합도는 낮고, 응집도는 높아야 한다.
- 재사용 규모에 따른 분류

함수와 객체	클래스나 메소드 단위의 소스 코드를 재사용한다.
컴포넌트	컴포넌트 자체에 대한 수정 없이 인터페이스를 통해 통신하는 방식으로 재사용한다.
애플리케이션	공통된 기능들을 제공하는 애플리케이션을 공유하는 방식으로 재사용한다.

핵심 057 효과적인 모듈 설계 방안

- 결합도는 줄이고 응집도는 높여서 모듈의 독립성과 재사용성을 높인다.
- 모듈의 제어 영역 안에서 그 모듈의 영향 영역을 유지시킨다.
- 복잡도와 중복성을 줄이고 일관성을 유지시킨다.
- 모듈의 기능은 예측이 가능해야 하며 지나치게 제한적이어서는 안 된다.
- 유지보수가 용이해야 한다.
- 모듈 크기는 시스템의 전반적인 기능과 구조를 이해하기 쉬운 크기로 분해한다.
- 효과적인 제어를 위해 모듈 간의 계층적 관계를 정의하는 자료가 제시되어야 한다.

핵심 058 코드(Code)의 개요

코드는 컴퓨터를 이용하여 자료를 처리하는 과정에서 분류 · 조합 및 집계를 용이하게 하고, 특정 자료의 추출을 쉽게 하기 위해서 사용하는 기호이다.

- 코드는 정보를 신속 · 정확 · 명료하게 전달할 수 있게 한다.
- 일반적인 코드의 예로 주민등록번호, 학번, 전화번호 등이 있다.
- 코드의 주요 기능에는 식별 기능, 분류 기능, 배열 기능, 표준화 기능, 간소화 기능이 있다.

식별 기능	데이터 간의 성격에 따라 구분이 가능하다.
분류 기능	특정 기준이나 동일한 유형에 해당하는 데이터를 그룹화 할 수 있다.
배열 기능	의미를 부여하여 나열할 수 있다.
표준화 기능	다양한 데이터를 기준에 맞추어 표현할 수 있다.
간소화 기능	복잡한 데이터를 간소화할 수 있다.

핵심 059 코드의 종류

순차 코드 (Sequence Code)	자료의 발생 순서, 크기 순서 등 일정 기준에 따라서 최초의 자료부터 차례로 일련번호를 부여하는 방법으로, 순서 코드 또는 일련번호 코드라고도 한다. 예 1, 2, 3, 4, …
블록 코드 (Block Code)	코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련번호를 부여하는 방법으로, 구분 코드라고도 한다. 예 1001~1100 : 총무부, 1101~1200 : 영업부
10진 코드 (Decimal Code)	코드화 대상 항목을 0~9까지 10진 분할하고, 다시 그 각각에 대하여 10진 분할하는 방법을 필요한 만큼 반복하는 방법으로, 도서 분류식 코드라고도 한다. 예 1000 : 공학, 1100 : 소프트웨어 공학, 1110 : 소프트웨어 설계
그룹 분류 코드 (Group Classification Code)	코드화 대상 항목을 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련번호를 부여하는 방법이다. 예 1-01-001 : 본사-총무부-인사계, 2-01-001 : 지사-총무부-인사계
연상 코드 (Mnemonic Code)	코드화 대상 항목의 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법이다. 예 TV-40 : 40인치 TV, L-15-220 : 15W 220V의 램프
표의 숫자 코드 (Significant Digit Code)	코드화 대상 항목의 성질, 즉 길이, 넓이, 부피, 지름, 높이 등의 물리적 수치를 그대로 코드에 적용시키는 방법으로, 유효 숫자 코드라고도 한다. 예 120-720-1500 : 두께×폭×길이가 120×720×1500인 강판
합성 코드 (Combined Code)	필요한 기능을 하나의 코드로 수행하기 어려운 경우 2개 이상의 코드를 조합하여 만드는 방법이다. 예 연상 코드 + 순차 코드 KE-711 : 대한항공 711기, AC-253 : 에어캐나다 253기

핵심 060 디자인 패턴(Design Pattern)의 개요

디자인 패턴은 각 모듈의 세분화된 역할이나 모듈들 간의 인터페이스와 같은 코드를 작성하는 수준의 세부적인 구현 방안을 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제를 의미한다.

- 디자인 패턴은 문제 및 배경, 실제 적용된 사례, 재사용이 가능한 샘플 코드 등으로 구성되어 있다.
- '바퀴를 다시 발명하지 마라(Don't reinvent the wheel)'라는 말과 같이, 개발 과정 중에 문제가 발생하면 새로 해결책을 구상하는 것보다 문제에 해당하는 디자인 패턴을 참고하여 적용하는 것이 더 효율적이다.
- GoF의 디자인 패턴은 유형에 따라 생성 패턴 5개, 구조 패턴 7개, 행위 패턴 11개 총 23개의 패턴으로 구성된다.

핵심 061 디자인 패턴 사용의 장 · 단점

- 범용적인 코딩 스타일로 인해 구조 파악이 용이하다.
- 객체지향 설계 및 구현의 생산성을 높이는 데 적합하다.
- 검증된 구조의 재사용을 통해 개발 시간과 비용이 절약된다.
- 초기 투자 비용이 부담될 수 있다.
- 개발자 간의 원활한 의사소통이 가능하다.
- 설계 변경 요청에 대한 유연한 대처가 가능하다.
- 객체지향을 기반으로 한 설계와 구현을 다루므로 다른 기반의 애플리케이션 개발에는 적합하지 않다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 062 생성/구조/행위 패턴

생성 패턴(Creational Pattern)

생성 패턴은 객체의 생성과 관련된 패턴으로 총 5개의 패턴이 있다.

- 생성 패턴은 객체의 생성과 참조 과정을 캡슐화 하여 객체가 생성되거나 변경되어도 프로그램의 구조에 영향을 크게 받지 않도록 하여 프로그램에 유연성을 더해준다.

추상 팩토리 (Abstract Factory)	• 구체적인 클래스에 의존하지 않고, 인터페이스를 통해 서로 연관 · 의존하는 객체들의 그룹으로 생성하여 추상적으로 표현한다. • 연관된 서브 클래스를 묶어 한 번에 교체하는 것이 가능하다.
빌더(Builder)	• 작게 분리된 인스턴스를 건축 하듯이 조합하여 객체를 생성한다. • 객체의 생성 과정과 표현 방법을 분리하고 있어, 동일한 객체 생성에서도 서로 다른 결과를 만들어 낼 수 있다.
팩토리 메소드 (Factory Method)	• 객체 생성을 서브 클래스에서 처리하도록 분리하여 캡슐화한 패턴이다. • 상위 클래스에서 인터페이스만 정의하고 실제 생성은 서브 클래스가 담당한다. • 가상 생성자(Virtual Constructor) 패턴이라고도 한다.
프로토타입 (Prototype)	• 원본 객체를 복제하는 방법으로 객체를 생성하는 패턴이다. • 일반적인 방법으로 객체를 생성하며, 비용이 큰 경우 주로 이용한다.
싱글톤 (Singleton)	• 하나의 객체를 생성하면 생성된 객체를 어디서든 참조할 수 있지만, 여러 프로세스가 동시에 참조할 수는 없다. • 클래스 내에서 인스턴스가 하나뿐임을 보장하며, 불필요한 메모리 낭비를 최소화 할 수 있다.

구조 패턴(Structural Pattern)

구조 패턴은 클래스나 객체들을 조합하여 더 큰 구조로 만들 수 있게 해주는 패턴으로 총 7개의 패턴이 있다.

- 구조 패턴은 구조가 복잡한 시스템을 개발하기 쉽게 도와준다.

어댑터 (Adapter)	• 호환성이 없는 클래스들의 인터페이스를 다른 클래스가 이용할 수 있도록 변환해주는 패턴이다. • 기존의 클래스를 이용하고 싶지만 인터페이스가 일치하지 않을 때 이용한다.
브리지 (Bridge)	• 구현부에서 추상층을 분리하여, 서로가 독립적으로 확장할 수 있도록 구성한 패턴이다. • 기능과 구현을 두 개의 별도 클래스로 구현한다.
컴포지트 (Composite)	• 여러 객체를 가진 복합 객체와 단일 객체를 구분 없이 다루고자 할 때 사용하는 패턴이다. • 객체들을 트리 구조로 구성하여 디렉터리 안에 디렉터리가 있듯이 복합 객체 안에 복합 객체가 포함되는 구조를 구현할 수 있다.
데코레이터 (Decorator)	• 객체 간의 결합을 통해 능동적으로 기능들을 확장할 수 있는 패턴이다. • 임의의 객체에 부가적인 기능을 추가하기 위해 다른 객체들을 덧붙이는 방식으로 구현한다.
퍼싸드 (Facade)	• 복잡한 서브 클래스들을 피해 더 상위에 인터페이스를 구성함으로써 서브 클래스들의 기능을 간편하게 사용할 수 있도록 하는 패턴이다. • 서브 클래스들 사이의 통합 인터페이스를 제공하는 Wrapper 객체가 필요하다.
플라이웨이트 (Flyweight)	• 인스턴스가 필요할 때마다 매번 생성하는 것이 아니고 가능한 한 공유해서 사용함으로써 메모리를 절약하는 패턴이다. • 다수의 유사 객체를 생성하거나 조작할 때 유용하게 사용할 수 있다.
프록시 (Proxy)	• 접근이 어려운 객체와 여기에 연결하려는 객체 사이에서 인터페이스 역할을 수행하는 패턴이다. • 네트워크 연결, 메모리의 대용량 객체로의 접근 등에 주로 이용한다.

행위 패턴(Behavioral Pattern)

행위 패턴은 클래스나 객체들이 서로 상호작용하는 방법이나 책임 분배 방법을 정의하는 패턴으로 총 11개의 패턴이 있다.

- 행위 패턴은 하나의 객체로 수행할 수 없는 작업을 여러 객체로 분배하면서 결합도를 최소화 할 수 있도록 도와준다.

책임 연쇄 (Chain of Responsibility)	• 요청을 처리할 수 있는 객체가 둘 이상 존재하여 한 객체가 처리하지 못하면 다음 객체로 넘어가는 형태의 패턴이다. • 요청을 처리할 수 있는 각 객체들이 고리(Chain)로 묶여 있어 요청이 해결될 때까지 고리를 따라 책임이 넘어간다.
커맨드 (Command)	• 요청을 객체의 형태로 캡슐화하여 재이용하거나 취소할 수 있도록 요청에 필요한 정보를 저장하거나 로그에 남기는 패턴이다. • 요청에 사용되는 각종 명령어들을 추상 클래스와 구체 클래스로 분리하여 단순화한다.
인터프리터 (Interpreter)	• 언어에 문법 표현을 정의하는 패턴이다. • SQL이나 통신 프로토콜과 같은 것을 개발할 때 사용한다.

반복자(Iterator)	• 자료 구조와 같이 접근이 잦은 객체에 대해 동일한 인터페이스를 사용하도록 하는 패턴이다. • 내부 표현 방법의 노출 없이 순차적인 접근이 가능하다.
중재자 (Mediator)	• 수많은 객체들 간의 복잡한 상호작용(Interface)을 캡슐화하여 객체로 정의하는 패턴이다. • 객체 사이의 의존성을 줄여 결합도를 감소시킬 수 있다.
메멘토 (Memento)	• 특정 시점에서의 객체 내부 상태를 객체화함으로써 이후 요청에 따라 객체를 해당 시점의 상태로 돌릴 수 있는 기능을 제공하는 패턴이다. • Ctrl+Z와 같은 되돌리기 기능을 개발할 때 주로 이용한다.
옵서버 (Observer)	• 한 객체의 상태가 변화하면 객체에 상속되어 있는 다른 객체들에게 변화된 상태를 전달하는 패턴이다. • 주로 분산된 시스템 간에 이벤트를 생성 · 발행(Publish)하고, 이를 수신(Subscribe)해야 할 때 이용한다.
상태(State)	• 객체의 상태에 따라 동일한 동작을 다르게 처리해야 할 때 사용하는 패턴이다. • 객체 상태를 캡슐화하고 이를 참조하는 방식으로 처리한다.
전략 (Strategy)	• 동일한 계열의 알고리즘들을 개별적으로 캡슐화하여 상호 교환할 수 있게 정의하는 패턴이다. • 클라이언트는 독립적으로 원하는 알고리즘을 선택하여 사용할 수 있으며, 클라이언트에 영향 없이 알고리즘의 변경이 가능하다.
템플릿 메소드 (Template Method)	• 상위 클래스에서 골격을 정의하고, 하위 클래스에서 세부 처리를 구체화하는 구조의 패턴이다. • 유사한 서브 클래스를 묶어 공통된 내용을 상위 클래스에서 정의함으로써 코드의 양을 줄이고 유지보수를 용이하게 해준다.
방문자 (Visitor)	• 각 클래스들의 데이터 구조에서 처리 기능을 분리하여 별도의 클래스로 구성하는 패턴이다. • 분리된 처리 기능은 각 클래스를 방문(Visit)하여 수행한다.

핵심 063 시스템 인터페이스 요구사항 구성

시스템 인터페이스는 독립적으로 떨어져 있는 시스템들끼리 서로 연동하여 상호 작용하기 위한 접속 방법이나 규칙을 의미한다.

- 시스템 인터페이스 요구사항은 개발을 목표로 하는 시스템과 외부 시스템을 연동하는데 필요한 시스템 인터페이스에 대한 요구사항을 기술한 것이다.
- 시스템 인터페이스 요구사항 명세서의 구성 요소
  - 인터페이스 이름
  - 연계 대상 시스템
  - 연계 범위 및 내용
  - 연계 방식
  - 송신 데이터
  - 인터페이스 주기
  - 기타 고려사항 등

핵심 064 시스템 인터페이스 요구사항 분석 절차

❶ 소프트웨어 요구사항 목록에서 시스템 인터페이스 관련 요구사항을 선별하여 별도로 시스템 인터페이스 요구사항 목록을 만든다.

❷ 시스템 인터페이스와 관련된 요구사항 및 아키텍처 정의서, 현행 시스템의 대 · 내외 연계 시스템 현황 자료 등 시스템 인터페이스 요구사항과 관련된 자료를 준비한다.

❸ 시스템 인터페이스에 대한 요구사항 명세서를 확인하여 기능적인 요구사항과 비기능적인 요구사항으로 분류한다.

❹ 시스템 인터페이스 요구사항 명세서와 시스템 인터페이스 요구사항 목록 및 기타 관련 자료들을 비교하여 요구사항을 분석하고 내용을 추가하거나 수정한다.

❺ 추가 · 수정한 시스템 인터페이스 요구사항 명세서와 시스템 인터페이스 요구사항 목록을 관련 이해관계자에게 전달한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 065 요구사항 검증 방법

- 요구사항 검토(Requirements Review) : 요구사항 명세서의 오류 확인 및 표준 준수 여부 등의 결함 여부를 검토 담당자들이 수작업으로 분석하는 방법으로, 동료검토, 워크스루, 인스펙션 등이 있다.

동료검토 (Peer Review)	요구사항 명세서 작성자가 명세서 내용을 직접 설명하고 동료들이 이를 들으면서 결함을 발견하는 형태의 검토 방법이다.
워크스루 (Walk Through)	검토 회의 전에 요구사항 명세서를 미리 배포하여 사전 검토한 후에 짧은 검토 회의를 통해 결함을 발견하는 형태의 검토 방법이다.
인스펙션 (Inspection)	요구사항 명세서 작성자를 제외한 다른 검토 전문가들이 요구사항 명세서를 확인하면서 결함을 발견하는 형태의 검토 방법이다.

- 프로토타이핑(Prototyping) : 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품(Prototype)을 만들어 최종 결과물을 예측한다.
- 테스트 설계 : 요구사항은 테스트할 수 있도록 작성되어야 하며, 이를 위해 테스트 케이스(Test Case)를 생성하여 이후에 요구사항이 현실적으로 테스트 가능한지를 검토한다.
- CASE 도구 활용 : 일관성 분석(Consistency Analysis)을 통해 요구사항 변경사항의 추적 및 분석, 관리하고, 표준 준수 여부를 확인한다.

핵심 066 인터페이스 요구사항 검증의 주요 항목

완전성 (Completeness)	사용자의 모든 요구사항이 누락되지 않고 완전하게 반영되어 있는가?
일관성 (Consistency)	요구사항이 모순되거나 충돌되는 점 없이 일관성을 유지하고 있는가?
명확성 (Unambiguity)	모든 참여자가 요구사항을 명확히 이해할 수 있는가?
기능성 (Functionality)	요구사항이 '어떻게(How to)' 보다 '무엇을(What)'에 중점을 두고 있는가?
검증 가능성 (Verifiability)	요구사항이 사용자의 요구를 모두 만족하고, 개발된 소프트웨어가 사용자의 요구 내용과 일치하는지를 검증할 수 있는가?
추적 가능성 (Traceability)	요구사항 명세서와 설계서를 추적할 수 있는가?
변경 용이성 (Easily Changeable)	요구사항 명세서의 변경이 쉽도록 작성되었는가?

핵심 067 인터페이스 시스템 식별

개발 시스템 식별	인터페이스 관련 자료들을 기반으로 개발하고자 하는 시스템의 상세 식별 정보를 정의하고 목록을 작성하는 것이다.
내 · 외부 시스템 식별	인터페이스 관련 자료들을 기반으로 개발할 시스템과 연계할 내 · 외부 시스템들의 상세 식별 정보를 정의하고 목록을 작성하는 것이다.
내 · 외부 시스템 환경 및 관리 주체 식별	• 내 · 외부 시스템 환경 : 연계할 시스템 접속에 필요한 IP 또는 URL, Port 정보 등 시스템의 실제 운용 환경을 의미한다. • 내 · 외부 시스템 관리 주체 : 하드웨어를 실제적으로 관리하는 담당자를 의미한다. • 인터페이스 관련 자료들을 기반으로 내 · 외부 시스템의 실제 운용 환경과 하드웨어 관리 주체를 확인한다.
내 · 외부 시스템 네트워크 연결 정보 식별	• 인터페이스 관련 자료들을 기반으로 내 · 외부 시스템을 연계하는데 필요한 네트워크 연결 정보를 확인한다. • 내 · 외부 시스템 네트워크 연결 정보는 시스템 로그인 및 DB 정보를 의미한다.
인터페이스 식별	인터페이스 요구사항 명세서와 인터페이스 요구사항 목록을 기반으로 개발할 시스템과 이와 연계할 내 · 외부 시스템 사이의 인터페이스를 식별하고 인터페이스 목록을 작성하는 것이다.
인터페이스 시스템 식별	인터페이스별로 인터페이스에 참여하는 시스템들을 송신 시스템과 수신 시스템으로 구분하여 작성하는 것이다.

핵심 068 식별 대상 데이터

- 식별 대상 데이터는 송 · 수신 시스템 사이에서 교환되는 데이터로, 규격화된 표준 형식에 따라 전송된다.
- 인터페이스 표준 항목
  - 인터페이스 표준 항목은 송 · 수신 시스템을 연계하는데 표준적으로 필요한 데이터를 의미한다.
  - 인터페이스 표준 항목은 시스템 공통부와 거래 공통부로 나뉜다.

시스템 공통부	• 시스템 간 연동 시 필요한 공통 정보이다. • 구성 정보 : 인터페이스 ID, 전송 시스템 정보, 서비스 코드 정보, 응답 결과 정보, 장애 정보 등
거래 공통부	• 시스템들이 연동된 후 송 · 수신 되는 데이터를 처리할 때 필요한 정보이다. • 구성 정보 : 직원 정보, 승인자 정보, 기기 정보, 매체 정보 등

- 송 · 수신 데이터 항목
  - 송 · 수신 데이터 항목은 송 · 수신 시스템이 업무를 수행하는 데 사용하는 데이터이다.
  - 전송되는 데이터 항목과 순서는 인터페이스별로 다르다.
- 공통 코드
  - 공통 코드는 시스템들에서 공통적으로 사용하는 코드이다.
  - 연계 시스템이나 연계 소프트웨어에서 사용하는 상태 및 오류 코드 등과 같은 항목에 대해 코드값과 코드명, 코드 설명 등을 공통 코드로 관리한다.

핵심 069 송 · 수신 데이터 식별

개발할 시스템과 연계할 내 · 외부 시스템 사이의 정보 흐름과 데이터베이스 산출물을 기반으로 송 · 수신 데이터를 식별한다.

- 인터페이스 표준 항목과 송 · 수신 데이터 항목 식별 : 송 · 수신 시스템 사이의 교환 범위를 확인하고 인터페이스 표준 항목에 대해 송 · 수신 데이터 항목을 식별한다.
- 코드성 데이터 항목 식별 : 코드성 데이터 항목에 대해 코드, 코드명, 코드 설명 등의 코드 정보를 식별한다.
- 코드성 데이터 항목에 대해 송신 시스템에서 사용하는 코드 정보와 수신 시스템에서 사용하는 코드 정보가 동일한 경우 공통 코드 정보를 확보하고, 다른 경우 매핑 필요 대상으로 분류하여 양쪽 시스템에서 사용하는 코드 정보를 확보한다.

핵심 070 시스템 연계 기술

DB Link	DB에서 제공하는 DB Link 객체를 이용하는 방식이다.
API/Open API	송신 시스템의 데이터베이스(DB)에서 데이터를 읽어와 제공하는 애플리케이션 프로그래밍 인터페이스 프로그램이다.
연계 솔루션	EAI 서버와 송 · 수신 시스템에 설치되는 클라이언트(Client)를 이용하는 방식이다.
Socket	서버는 통신을 위한 소켓(Socket)을 생성하여 포트를 할당하고 클라이언트의 통신 요청 시 클라이언트와 연결하여 통신하는 네트워크 기술이다.
Web Service	웹 서비스(Web Service)에서 WSDL과 UDDI, SOAP 프로토콜을 이용하여 연계하는 서비스이다.

핵심 071 인터페이스 통신 유형 / 처리 유형 / 발생 주기

인터페이스 통신 유형

단방향	시스템에서 거래를 요청만 하고 응답이 없는 방식이다.
동기	시스템에서 거래를 요청하고 응답이 올 때까지 대기(Request-Reply)하는 방식이다.
비동기	시스템에서 거래를 요청하고 다른 작업을 수행하다 응답이 오면 처리하는 방식(Send-Receive, Send-Receive-Acknowledge, Publish-Subscribe)이다.

인터페이스 처리 유형

실시간 방식	사용자가 요청한 내용을 바로 처리해야 할 때 사용하는 방식이다.
지연 처리 방식	데이터를 매건 단위로 처리할 경우 비용이 많이 발생할 때 사용하는 방식이다.
배치 방식	대량의 데이터를 처리할 때 사용하는 방식이다.

인터페이스 발생 주기

인터페이스 발생 주기는 업무의 성격과 송 · 수신 데이터 전송량을 고려하여 매일, 수시, 주 1회 등으로 구분한다.

핵심 072 미들웨어(Middleware)

미들웨어는 미들(Middle)과 소프트웨어(Software)의 합성어로, 운영체제와 응용 프로그램, 또는 서버와 클라이언트 사이에서 다양한 서비스를 제공하는 소프트웨어이다.

DB(DataBase)	• DB는 데이터베이스 벤더에서 제공하는 클라이언트에서 원격의 데이터베이스와 연결하기 위한 미들웨어이다. • DB를 사용하여 시스템을 구축하는 경우 보통 2-Tier 아키텍처라고 한다.
RPC(Remote Procedure Call)	RPC(원격 프로시저 호출)는 응용 프로그램의 프로시저를 사용하여 원격 프로시저를 마치 로컬 프로시저처럼 호출하는 방식의 미들웨어이다.
MOM(Message Oriented Middleware)	• MOM(메시지 지향 미들웨어)은 메시지 기반의 비동기형 메시지를 전달하는 방식의 미들웨어이다. • 온라인 업무보다는 이기종 분산 데이터 시스템의 데이터 동기를 위해 많이 사용된다.
TP-Monitor (Transaction Processing Monitor)	• TP-Monitor(트랜잭션 처리 모니터)는 항공기나 철도 예약 업무 등과 같은 온라인 트랜잭션 업무에서 트랜잭션을 처리 및 감시하는 미들웨어이다. • 사용자 수가 증가해도 빠른 응답 속도를 유지해야 하는 업무에 주로 사용된다.
ORB(Object Request Broker)	• ORB(객체 요청 브로커)는 객체 지향 미들웨어로 코바(CORBA) 표준 스펙을 구현한 미들웨어이다. • 최근에는 TP-Monitor의 장점인 트랜잭션 처리와 모니터링 등을 추가로 구현한 제품도 있다.
WAS(Web Application Server)	• WAS(웹 애플리케이션 서버)는 정적인 콘텐츠를 처리하는 웹 서버와 달리 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어이다. • 클라이언트/서버 환경보다는 웹 환경을 구현하기 위한 미들웨어이다.

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

2 과목 소프트웨어 개발

핵심 073 자료 구조

자료 구조의 정의

효율적인 프로그램을 작성할 때 가장 우선적인 고려사항은 저장 공간의 효율성과 실행시간의 신속성이다. 자료 구조는 프로그램에서 사용하기 위한 자료를 기억장치의 공간 내에 저장하는 방법과 저장된 그룹 내에 존재하는 자료 간의 관계, 처리 방법 등을 연구 분석하는 것을 말한다.

자료 구조의 분류

- 선형 구조 (Linear Structure)
  - 배열(Array)
  - 선형 리스트(Linear List)
    - 연속 리스트(Contiguous List)
    - 연결 리스트(Linked List)
  - 스택(Stack)
  - 큐(Queue)
  - 데크(Deque)
- 비선형 구조 (Non-Linear Structure)
  - 트리(Tree)
  - 그래프(Graph)

핵심 074 배열(Array)

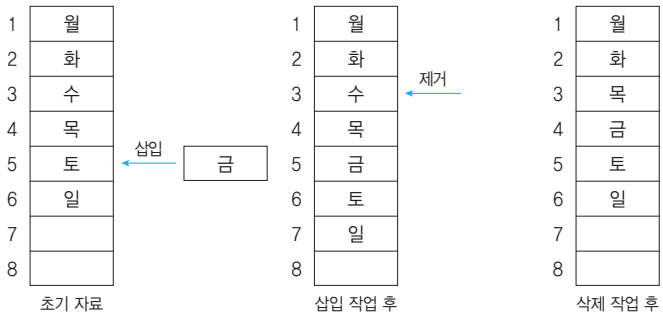
배열은 동일한 자료형의 데이터들이 같은 크기로 나열되어 순서를 갖고 있는 집합이다.

- 배열은 정적인 자료 구조로 기억장소의 추가가 어렵고, 데이터 삭제 시 데이터가 저장되어 있던 기억장소는 빈 공간으로 남아있어 메모리의 낭비가 발생한다.
- 배열은 첨자를 이용하여 데이터에 접근한다.
- 배열은 반복적인 데이터 처리 작업에 적합한 구조이다.
- 배열은 데이터마다 동일한 이름의 변수를 사용하여 처리가 간편하다.
- 배열은 사용한 첨자의 개수에 따라 n차원 배열이라고 부른다.

시험에 나오는 것만 공부한다! 시나공시리즈

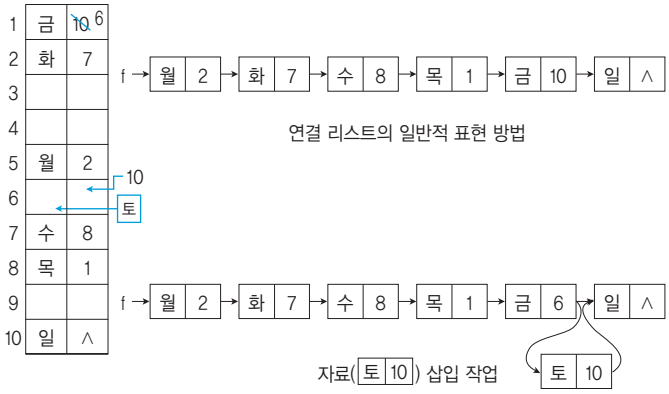
핵심 075 선형 리스트(Linear List)

- 연속 리스트(Contiguous List)
  - 연속 리스트는 배열과 같이 연속되는 기억장소에 저장되는 자료 구조이다.
  - 연속 리스트는 기억장소를 연속적으로 배정받기 때문에 기억장소 이용 효율은 밀도가 1로서 가장 좋다.
  - 연속 리스트는 중간에 데이터를 삽입하기 위해서는 연속된 빈 공간이 있어야 하며, 삽입 · 삭제 시 자료의 이동이 필요하다.



- 연결 리스트(Linked List)
  - 연결 리스트는 자료들을 반드시 연속적으로 배열시키지는 않고 임의의 기억공간에 기억시키되, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료 구조이다.
  - 연결 리스트는 노드의 삽입 · 삭제 작업이 용이하다.
  - 기억 공간이 연속적으로 놓여 있지 않아도 저장할 수 있다.
  - 연결 리스트는 연결을 위한 링크(포인터) 부분이 필요하기 때문에 순차 리스트에 비해 기억 공간의 이용 효율이 좋지 않다.
  - 연결 리스트는 연결을 위한 포인터를 찾는 시간이 필요하기 때문에 접근 속도가 느리다.
  - 연결 리스트는 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘들다.

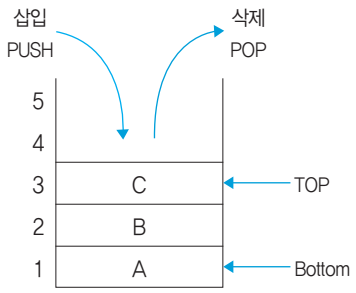
예 연결 리스트 기억장치 내에서의 표현 방법



핵심 076 스택(Stack)

스택은 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조이다.

- 스택은 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO; Last In First Out) 방식으로 자료를 처리한다.
- 스택의 모든 기억 공간이 꽉 채워져 있는 상태에서 데이터가 삽입되면 오버플로(Overflow)가 발생하며, 더 이상 삭제할 데이터가 없는 상태에서 데이터를 삭제하면 언더플로(Underflow)가 발생한다.

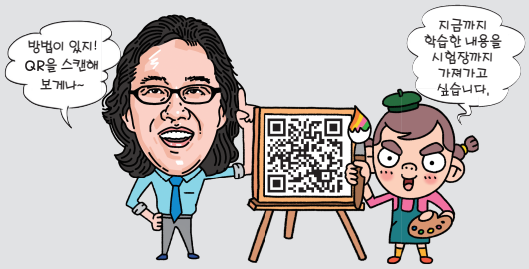


- TOP : 스택으로 할당된 기억 공간에 가장 마지막으로 삽입된 자료가 기억된 위치를 가리키는 요소이다.
- Bottom : 스택의 가장 밑바닥이다.

시험에 나오는 것만 공부한다! 시나공시리즈

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요? 까먹기 전에 다시 한 번 복습하고 싶다고요? 지금까지 공부한 내용을 안전하게 시험장까지 가져가는 완벽한 방법이 있습니다. 지금 당장 QR 코드를 스캔해 보세요.

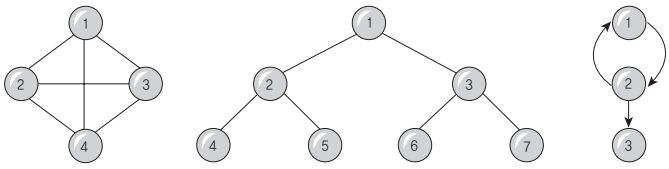


www.membox.co.kr을 직접 입력해도 접속할 수 있습니다.

핵심 077 그래프(Graph)

그래프 G는 정점 V(Vertex)와 간선 E(Edge)의 두 집합으로 이루어진다.

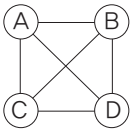
- 간선의 방향성 유무에 따라 방향 그래프와 무방향 그래프로 구분된다.
- 통신망(Network), 교통망, 이항관계, 연립방정식, 유기화학 구조식, 무향선분 해법 등에 응용된다.



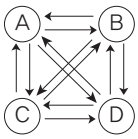
※ 방향/무방향 그래프의 최대 간선 수

n개의 정점으로 구성된 무방향 그래프에서 최대 간선 수는 $n(n-1)/2$이고, 방향 그래프에서 최대 간선 수는 $n(n-1)$이다.

예 정점이 4개인 경우 무방향 그래프와 방향 그래프의 최대 간선 수는 다음과 같다.



무방향 그래프의 최대 간선 수 : $4(4-1)/2 = 6$

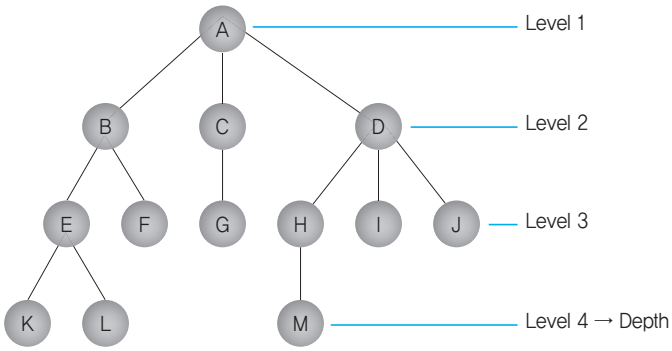


방향 그래프의 최대 간선 수 : $4(4-1) = 12$

핵심 078 트리의 개요

트리는 정점(Node, 노드)과 선분(Branch, 가지)을 이용하여 사이클을 이루지 않도록 구성한 그래프(Graph)의 특수한 형태이다.

- 트리는 가족의 계보(족보), 조직도 등을 표현하기에 적합하다.
- 트리 관련 용어



– 노드(Node) : 트리의 기본 요소로서 자료 항목과 다른 항목에 대한 가지(Branch)를 합친 것

예 A, B, C, D, E, F, G, H, I, J, K, L, M

– 근 노드(Root Node) : 트리의 맨 위에 있는 노드

예 A

– 디그리(Degree, 차수) : 각 노드에서 뻗어 나온 가지의 수

예 A = 3, B = 2, C = 1, D = 3

– 단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드, 즉 디그리가 0인 노드

예 K, L, F, G, M, I, J

– 자식 노드(Son Node) : 어떤 노드에 연결된 다음 레벨의 노드들

예 D의 자식 노드 : H, I, J

– 부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드들

예 E, F의 부모 노드 : B

– 형제 노드(Brother Node, Sibling) : 동일한 부모를 갖는 노드들

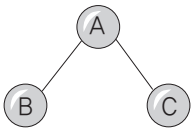
예 H의 형제 노드 : I, J

– 트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수

예 노드 A나 D가 3개의 디그리를 가지므로 앞 트리의 디그리는 3이다.

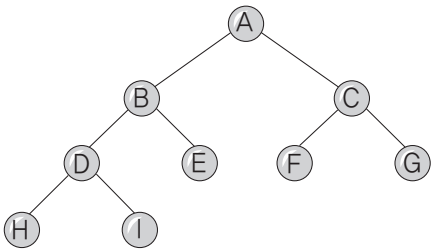
핵심 079 트리의 운행법

이진 트리의 운행법은 다음 세 가지가 있다.

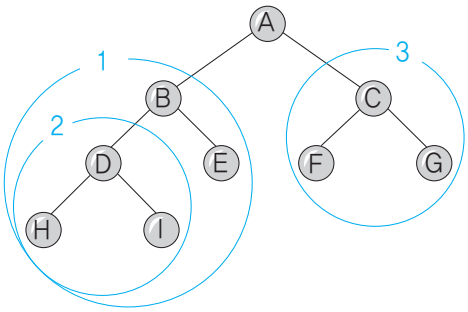


- Preorder 운행 : Root → Left → Right 순으로 운행한다. A, B, C
- Inorder 운행 : Left → Root → Right 순으로 운행한다. B, A, C
- Postorder 운행 : Left → Right → Root 순으로 운행한다. B, C, A

예제 다음 트리를 Inorder, Preorder, Postorder 방법으로 운행했을 때 각 노드를 방문한 순서는?



Preorder 운행법의 방문 순서



❶ Preorder는 Root → Left → Right이므로 A13이 된다.

❷ 1은 B2E이므로 AB2E3이 된다.

❸ 2는 DHI이므로 ABDHIE3이 된다.

❹ 3은 CFG이므로 ABDHIECFG가 된다.

- 방문 순서 : ABDHIECFG

Inorder 운행법의 방문 순서

❶ Inorder는 Left → Root → Right이므로 1A3이 된다.

❷ 1은 2BE이므로 2BEA3이 된다.

❸ 2는 HDI이므로 HDIBEA3이 된다.

❹ 3은 FCG이므로 HDIBEAFCG가 된다.

- 방문 순서 : HDIBEAFCG

Postorder 운행법의 방문 순서

❶ Postorder는 Left → Right → Root이므로 13A가 된다.

❷ 1은 2EB이므로 2EB3A가 된다.

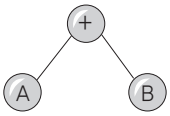
❸ 2는 HID이므로 HIDEB3A가 된다.

❹ 3은 FGC이므로 HIDEBFGCA가 된다.

- 방문 순서 : HIDEBFGCA

핵심 080 수식의 표기법

산술식을 계산하기 위해 기억공간에 기억시키는 방법으로 이진 트리를 많이 사용한다. 이진 트리로 만들어진 수식을 인오더, 프리오더, 포스트오더로 운행하면 각각 중위(Infix), 전위(Prefix), 후위(Postfix) 표기법이 된다.



- 전위 표기법(PreFix) : 연산자 → Left → Right, +AB
- 중위 표기법(InFix) : Left → 연산자 → Right, A+B
- 후위 표기법(PostFix) : Left → Right → 연산자, AB+

Infix 표기를 Postfix나 Prefix로 바꾸기

- Postfix나 Prefix는 스택을 이용하여 처리하므로 Infix는 Postfix나 Prefix로 바꾸어 처리한다.

예제 1 다음과 같이 Infix로 표기된 수식을 Prefix와 Postfix로 변환하시오.

X = A / B * (C + D) + E

- Prefix로 변환하기

❶ 연산 우선순위에 따라 괄호로 묶는다.

(X = (((A / B) * (C + D)) + E))

❷ 연산자를 해당 괄호의 앞(왼쪽)으로 옮긴다.

X = (((A / B) * (C + D)) + E))

→ = (X + (*(/ (AB) + (CD)) E))

시험에 나오는 것만 공부한다! 시나공시리즈

❸ 필요없는 괄호를 제거한다.

prefix 표기 : = X + * / A B + C D E

• Postfix로 변환하기

❶ 연산 우선순위에 따라 괄호로 묶는다.

(X = (((A / B) * (C + D)) + E))

❷ 연산자를 해당 괄호의 뒤(오른쪽)로 옮긴다.

(X = (((A / B) * (C + D)) + E))

→ (X (((A B) / (C D) +) * E) +) =

❸ 필요없는 괄호를 제거한다.

Postfix 표기 : X A B / C D + * E + =

Postfix나 Prefix로 표기된 수식을 Infix로 바꾸기

예제 2 다음과 같이 Postfix로 표기된 수식을 Infix로 변환하시오.

A B C- / D E F + * +

• Postfix는 Infix 표기법에서 연산자를 해당 피연산자 두 개의 뒤로 이동한 것이므로 연산자를 다시 해당 피연산자 두 개의 가운데로 옮기면 된다.

❶ 먼저 인접한 피연산자 두 개와 오른쪽의 연산자를 괄호로 묶는다.

((A (B C −) /) (D (E F +) *) +)

❷ 연산자를 해당 피연산자의 가운데로 이동시킨다.

((A (B C −) /) (D (E F +) *) +)

→ ((A / (B − C)) + (D * (E + F)))

❸ 필요 없는 괄호를 제거한다.

((A / (B − C)) + (D * (E + F))) → A / (B − C) + D * (E + F)

예제 3 다음과 같이 Prefix로 표기된 수식을 Infix로 변환하시오.

+ / A − B C * D + E F

• Prefix는 Infix 표기법에서 연산자를 해당 피연산자 두 개의 앞으로 이동한 것이므로 연산자를 다시 해당 피연산자 두 개의 가운데로 옮기면 된다.

❶ 먼저 인접한 피연산자 두 개와 왼쪽의 연산자를 괄호로 묶는다.

(+ (/ A (− B C)) (* D (+ E F)))

❷ 연산자를 해당 피연산자 사이로 이동시킨다.

(+ (/ A (− B C)) (* D (+ E F)))

→ ((A/(B−C)) + (D * (E+F)))

❸ 필요 없는 괄호를 제거한다.

((A/(B−C)) + (D*(E+F))) → A/(B−C)+D*(E+F)

핵심 081 삽입 정렬(Insertion Sort)

삽입 정렬은 가장 간단한 정렬 방식으로 이미 순서화된 파일에 새로운 하나의 레코드를 순서에 맞게 삽입시켜 정렬한다.

• 두 번째 키와 첫 번째 키를 비교해 순서대로 나열(1회전)하고, 이어서 세 번째 키를 첫 번째, 두 번째 키와 비교해 순서대로 나열(2회전)하고, 계속해서 n번째 키를 앞의 n−1개의 키와 비교하여 알맞은 순서에 삽입하여 정렬하는 방식이다.

• 평균과 최악 모두 수행 시간 복잡도는 $O(n^2)$이다.

예제 8, 5, 6, 2, 4를 삽입 정렬로 정렬하시오.

• 초기 상태 : | 8 | 5 | 6 | 2 | 4 |

• 1회전 : | 8 | 5 | 6 | 2 | 4 | → | 5 | 8 | 6 | 2 | 4 |

두 번째 값을 첫 번째 값과 비교하여 5를 첫 번째 자리에 삽입하고 8을 한 칸 뒤로 이동시킨다.

• 2회전 : | 5 | 8 | 6 | 2 | 4 | → | 5 | 6 | 8 | 2 | 4 |

세 번째 값을 첫 번째, 두 번째 값과 비교하여 6을 8자리에 삽입하고 8은 한 칸 뒤로 이동시킨다.

• 3회전 : | 5 | 6 | 8 | 2 | 4 | → | 2 | 5 | 6 | 8 | 4 |

네 번째 값 2를 처음부터 비교하여 맨 처음에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

• 4회전 : | 2 | 5 | 6 | 8 | 4 | → | 2 | 4 | 5 | 6 | 8 |

다섯 번째 값 4를 처음부터 비교하여 5자리에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 082 선택 정렬(Selection Sort)

선택 정렬은 n개의 레코드 중에서 최소값을 찾아 첫 번째 레코드 위치에 놓고, 나머지 (n−1)개 중에서 다시 최소값을 찾아 두 번째 레코드 위치에 놓는 방식을 반복하여 정렬하는 방식이다.

- 평균과 최악 모두 수행 시간 복잡도는 $O(n^2)$이다.

예제 8, 5, 6, 2, 4를 선택 정렬로 정렬하시오.

- 초기 상태 : 8 5 6 2 4
- 1회전 : 5 8 6 2 4 → 5 8 6 2 4 → 2 8 6 5 4 → 2 8 6 5 4
- 2회전 : 2 6 8 5 4 → 2 5 8 6 4 → 2 4 8 6 5
- 3회전 : 2 4 6 8 5 → 2 4 5 8 6
- 4회전 : 2 4 5 6 8

핵심 083 버블 정렬(Bubble Sort)

버블 정렬은 주어진 파일에서 인접한 두 개의 레코드 키값을 비교하여 그 크기에 따라 레코드 위치를 서로 교환하는 정렬 방식이다.

- 계속 정렬 여부를 플래그 비트(f)로 결정한다.
- 평균과 최악 모두 수행 시간 복잡도는 $O(n^2)$이다.

예제 8, 5, 6, 2, 4를 버블 정렬로 정렬하시오.

- 초기 상태 : 8 5 6 2 4
- 1회전 : 5 8 6 2 4 → 5 6 8 2 4 → 5 6 2 8 4 → 5 6 2 4 8
- 2회전 : 5 6 2 4 8 → 5 2 6 4 8 → 5 2 4 6 8
- 3회전 : 2 5 4 6 8 → 2 4 5 6 8
- 4회전 : 2 4 5 6 8

핵심 084 데이터베이스

데이터베이스는 특정 조직의 업무를 수행하는 데 필요한 상호 관련된 데이터들의 모임으로 다음과 같이 정의할 수 있다.

- **통합된 데이터(Integrated Data)** : 자료의 중복을 배제한 데이터의 모임이다.
- **저장된 데이터(Stored Data)** : 컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료이다.
- **운영 데이터(Operational Data)** : 조직의 고유한 업무를 수행하는 데 존재 가치가 확실하고 없어서는 안 될 반드시 필요한 자료이다.
- **공용 데이터(Shared Data)** : 여러 응용 시스템들이 공동으로 소유하고 유지하는 자료이다.

핵심 085 DBMS(DataBase Management System; 데이터베이스 관리 시스템)

DBMS란 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고, 데이터베이스를 관리해 주는 소프트웨어이다.

- DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공용할 수 있도록 관리해 준다.
- DBMS의 필수 기능에는 정의(Definition), 조작(Manipulation), 제어(Control) 기능이 있다.
  - **정의 기능** : 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 형(Type)과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 기능이다.
  - **조작 기능** : 데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능이다.

– 제어 기능

▶ 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 한다.

▶ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 권한(Authority)을 검사할 수 있어야 한다.

▶ 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 병행 제어(Concurrency Control)를 할 수 있어야 한다.

핵심 086 DBMS의 장 · 단점

장점	• 데이터의 논리적, 물리적 독립성이 보장된다. • 데이터의 중복을 피할 수 있어 기억 공간이 절약된다. • 저장된 자료를 공동으로 이용할 수 있다. • 데이터의 일관성을 유지할 수 있다. • 데이터의 무결성을 유지할 수 있다. • 보안을 유지할 수 있다. • 데이터를 표준화할 수 있다. • 데이터를 통합하여 관리할 수 있다. • 항상 최신의 데이터를 유지한다. • 데이터의 실시간 처리가 가능하다.
단점	• 데이터베이스의 전문가가 부족하다. • 전산화 비용이 증가한다. • 대용량 디스크로의 집중적인 Access로 과부하(Overhead)가 발생한다. • 파일의 예비(Backup)와 회복(Recovery)이 어렵다. • 시스템이 복잡하다.

핵심 087 스키마

스키마(Schema)는 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세(Specification)를 기술(Description)한 메타데이터(Meta-Data)의 집합이다.

- 스키마는 데이터베이스를 구성하는 데이터 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시 데이터 값들이 갖는 제약 조건 등에 관해 전반적으로 정의한다.
- 스키마는 사용자의 관점에 따라 외부 스키마, 개념 스키마, 내부 스키마로 나누어진다.

외부 스키마	사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한 것이다.
개념 스키마	데이터베이스의 전체적인 논리적 구조로서, 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스로 하나만 존재한다.
내부 스키마	물리적 저장장치의 입장에서 본 데이터베이스 구조로서, 실제로 데이터베이스에 저장될 레코드의 형식을 정의하고 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타낸다.

핵심 088 데이터 접속(Data Mapping)

데이터 접속은 소프트웨어의 기능 구현을 위해 프로그래밍 코드와 데이터베이스의 데이터를 연결(Mapping)하는 것을 말하며, 관련 기술로 SQL Mapping과 ORM이 있다.

- SQL Mapping : 프로그래밍 코드 내에 SQL을 직접 입력하여 DBMS의 데이터에 접속하는 기술로, 관련 프레임워크에는 JDBC, ODBC, MyBatis 등이 있다.
- ORM(Object-Relational Mapping) : 객체지향 프로그래밍의 객체(Object)와 관계형(Relational) 데이터베이스의 데이터를 연결(Mapping)하는 기술로, 관련 프레임워크에는 JPA, Hibernate, Django 등이 있다.

핵심 089 트랜잭션(Transaction)

트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.

- 트랜잭션을 제어하기 위해서 사용하는 명령어를 TCL(Transaction Control Language)이라고 하며, TCL의 종류에는 COMMIT, ROLLBACK, SAVEPOINT가 있다.

- COMMIT : 트랜잭션 처리가 정상적으로 종료되어 트랜잭션이 수행한 변경 내용을 데이터베이스에 반영하는 명령어
- ROLLBACK : 하나의 트랜잭션 처리가 비정상으로 종료되어 데이터베이스의 일관성이 깨졌을 때 트랜잭션이 행한 모든 변경 작업을 취소하고 이전 상태로 되돌리는 연산
- SAVEPOINT(=CHECKPOINT) : 트랙잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어

핵심 090 절차형 SQL의 개요

절차형 SQL은 C, JAVA 등의 프로그래밍 언어와 같이 연속적인 실행이나 분기, 반복 등의 제어가 가능한 SQL을 의미한다.

- 절차형 SQL은 일반적인 프로그래밍 언어에 비해 효율은 떨어지지만 단일 SQL 문장으로 처리하기 어려운 연속적인 작업들을 처리하는데 적합하다.
- 절차형 SQL을 활용하여 다양한 기능을 수행하는 저장 모듈을 생성할 수 있다.
- 절차형 SQL은 DBMS 엔진에서 직접 실행되기 때문에 입 · 출력 패킷이 적은 편이다.
- BEGIN ~ END 형식으로 작성되는 블록(Block) 구조로 되어 있기 때문에 기능별 모듈화가 가능하다.
- 절차형 SQL의 종류에는 프로시저, 트리거, 사용자 정의 함수가 있다.
  - 프로시저(Procedure) : 특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다.
  - 트리거(Trigger) : 데이터베이스 시스템에서 데이터의 입력, 갱신, 삭제 등의 이벤트(Event)가 발생할 때마다 관련 작업이 자동으로 수행된다.
  - 사용자 정의 함수 : 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 예약어 Return을 사용하여 처리 결과를 단일값으로 반환한다.

핵심 091 절차형 SQL의 테스트와 디버깅

절차형 SQL은 디버깅을 통해 기능의 적합성 여부를 검증하고, 실행을 통해 결과를 확인하는 테스트 과정을 수행한다.

- 절차형 SQL은 테스트 전에 생성을 통해 구문 오류(Syntax Error)나 참조 오류의 존재 여부를 확인한다.
- 많은 코드로 구성된 절차형 SQL의 특성상 오류 및 경고 메시지가 상세히 출력되지 않으므로 SHOW 명령어를 통해 내용을 확인하고 문제를 수정한다.
- 정상적으로 생성된 절차형 SQL은 디버깅을 통해 로직을 검증하고, 결과를 통해 최종적으로 확인한다.
- 절차형 SQL의 디버깅은 실제로 데이터베이스에 변화를 줄 수 있는 삽입 및 변경 관련 SQL문을 주석으로 처리하고, 출력문을 이용하여 화면에 출력하여 확인한다.

핵심 092 단위 모듈(Unit Module)의 개요

단위 모듈은 소프트웨어 구현에 필요한 여러 동작 중 한 가지 동작을 수행하는 기능을 모듈로 구현한 것이다.

- 단위 모듈로 구현되는 하나의 기능을 단위 기능이라고 부른다.
- 단위 모듈은 사용자나 다른 모듈로부터 값을 전달받아 시작되는 작은 프로그램을 의미하기도 한다.
- 두 개의 단위 모듈이 합쳐질 경우 두 개의 기능을 구현할 수 있다.
- 단위 모듈의 구성 요소에는 처리문, 명령문, 데이터 구조 등이 있다.
- 단위 모듈은 독립적인 컴파일이 가능하며, 다른 모듈에 호출되거나 삽입되기도 한다.
- 단위 모듈을 구현하기 위해서는 단위 기능 명세서를 작성한 후 입 · 출력 기능과 알고리즘을 구현해야 한다.

단위 기능 명세서 작성 → 입 · 출력 기능 구현 → 알고리즘 구현

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 093 IPC(Inter-Process Communication)

IPC는 모듈 간 통신 방식을 구현하기 위해 사용되는 대표적인 프로그래밍 인터페이스 집합으로, 복수의 프로세스를 수행하며 이뤄지는 프로세스 간 통신까지 구현이 가능하다.

- IPC의 대표 메소드 5가지

Shared Memory	다수의 프로세스가 공유 가능한 메모리를 구성하여 프로세스 간 통신을 수행
Socket	네트워크 소켓을 이용하여 네트워크를 경유하는 프로세스들 간 통신을 수행
Semaphores	공유 자원에 대한 접근 제어를 통해 프로세스 간 통신을 수행
Pipes&named Pipes	• 'Pipe'라고 불리는 선입선출 형태로 구성된 메모리를 여러 프로세스가 공유하여 통신을 수행 • 하나의 프로세스가 Pipe를 이용 중이라면 다른 프로세스는 접근할 수 없음
Message Queueing	메시지가 발생하면 이를 전달하는 형태로 프로세스 간 통신을 수행

핵심 094 단위 모듈 테스트의 개요

단위 모듈 테스트는 프로그램의 단위 기능을 구현하는 모듈이 정해진 기능을 정확히 수행하는지 검증하는 것이다.

- 단위 모듈 테스트는 단위 테스트(Unit Test)라고도 하며, 화이트박스 테스트와 블랙박스 테스트 기법을 사용한다.
- 단위 모듈 테스트를 수행하기 위해서는 모듈을 단독적으로 실행할 수 있는 환경과 테스트에 필요한 데이터가 모두 준비되어야 한다.
- 모듈의 통합 이후에는 오랜 시간 추적해야 발견할 수 있는 에러들도 단위 모듈 테스트를 수행하면 쉽게 발견하고 수정할 수 있다.
- 단위 모듈 테스트의 기준은 단위 모듈에 대한 코드이므로 시스템 수준의 오류는 잡아낼 수 없다.

핵심 095 테스트 케이스(Test Case)

테스트 케이스는 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서로, 명세 기반 테스트의 설계 산출물에 해당된다.

- 단위 모듈을 테스트하기 전에 테스트에 필요한 입력 데이터, 테스트 조건, 예상 결과 등을 모아 테스트 케이스를 만든다.
- 테스트 케이스를 이용하지 않고 수행하는 직관적인 테스트는 특정 요소에 대한 검증이 누락되거나 불필요한 검증의 반복으로 인해 인력과 시간을 낭비할 수 있다.
- ISO/IEC/IEEE 29119-3 표준에 따른 테스트 케이스의 구성 요소는 다음과 같다.

식별자(Identifier)	항목 식별자, 일련번호
테스트 항목 (Test Item)	테스트 대상(모듈 또는 기능)
입력 명세 (Input Specification)	입력 데이터 또는 테스트 조건
출력 명세 (Output Specification)	테스트 케이스 수행 시 예상되는 출력 결과
환경 설정 (Environmental Needs)	필요한 하드웨어나 소프트웨어의 환경
특수 절차 요구 (Special Procedure Requirement)	테스트 케이스 수행 시 특별히 요구되는 절차
의존성 기술 (Inter-case Dependencies)	테스트 케이스 간의 의존성

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 096 통합 개발 환경(IDE; Integrated Development Environment)

통합 개발 환경은 개발에 필요한 환경, 즉 편집기(Editor), 컴파일러(Compiler), 디버거(Debugger) 등의 다양한 툴을 하나의 인터페이스로 통합하여 제공하는 것을 의미한다.

- 통합 개발 환경 도구는 통합 개발 환경을 제공하는 소프트웨어를 의미한다.
- 통합 개발 환경 도구는 코드의 자동 생성 및 컴파일이 가능하고 추가 기능을 위한 도구들을 다운로드하여 추가할 수 있다.
- 통합 개발 환경을 지원하는 도구는 플랫폼, 운영체제, 언어별로 다양하게 존재하며, 대표적인 도구는 다음과 같다.

프로그램	개발사	플랫폼	운영체제	지원 언어
이클립스 (Eclipse)	Eclipse Foundation, IBM	크로스 플랫폼	Windows, Linux, MacOS 등	Java, C, C++, PHP, JSP 등
비주얼 스튜디오 (Visual Studio)	Microsoft	Win32, Win64	Windows	Basic, C, C++, C#, .NET 등
엑스 코드 (Xcode)	Apple	Mac, iPhone	MacOS, iOS	C, C++, C#, Java, AppleScript 등
안드로이드 스튜디오 (Android Studio)	Google	Android	Windows, Linux, MacOS	Java, C, C++
IDEA	JetBrains (이전 IntelliJ)	크로스 플랫폼	Windows, Linux, MacOS	Java, JSP, XML, Go, Kotlin, PHP 등

핵심 097 기타 협업 도구

협업 도구는 개발에 참여하는 사람들이 서로 다른 작업 환경에서 원활히 프로젝트를 수행할 수 있도록 도와주는 도구(Tool)로, 협업 소프트웨어, 그룹웨어(Groupware) 등으로도 불린다.

- 협업 도구에는 일정 관리, 업무흐름 관리, 정보 공유, 커뮤니케이션 등의 업무 보조 도구가 포함되어 있다.
- 협업 도구는 웹 기반, PC, 스마트폰 등 다양한 플랫폼에서 사용할 수 있도록 제공된다.
- 협업 도구에 익숙하지 않거나 이용할 의지가 없으면 협업 도구가 오히려 협업의 방해 요소가 될 수 있다.

핵심 098 소프트웨어 패키징의 개요

소프트웨어 패키징이란 모듈별로 생성한 실행 파일들을 묶어 배포용 설치 파일을 만드는 것을 말한다.

- 개발자가 아니라 사용자를 중심으로 진행한다.
- 소스 코드는 향후 관리를 고려하여 모듈화하여 패키징한다.
- 사용자가 소프트웨어를 사용하게 될 환경을 이해하여, 다양한 환경에서 소프트웨어를 손쉽게 사용할 수 있도록 일반적인 배포 형태로 패키징한다.

핵심 099 패키징 시 고려사항

- 사용자의 시스템 환경, 즉 운영체제(OS), CPU, 메모리 등에 필요한 최소 환경을 정의한다.
- UI(User Interface)는 사용자가 눈으로 직접 확인할 수 있도록 시각적인 자료와 함께 제공하고 매뉴얼과 일치시켜 패키징한다.
- 소프트웨어는 단순히 패키징하여 배포하는 것으로 끝나는 것이 아니라 하드웨어와 함께 관리될 수 있도록 Managed Service 형태로 제공하는 것이 좋다.
- 사용자에게 배포되는 소프트웨어이므로 내부 콘텐츠에 대한 암호화 및 보안을 고려한다.

- 다른 여러 콘텐츠 및 단말기 간 DRM(디지털 저작권 관리) 연동을 고려한다.
- 사용자의 편의성을 위한 복잡성 및 비효율성 문제를 고려한다.
- 제품 소프트웨어 종류에 적합한 암호화 알고리즘을 적용한다.

핵심 100 릴리즈 노트(Release Note)의 개요

릴리즈 노트는 개발 과정에서 정리된 릴리즈 정보를 소프트웨어의 최종 사용자인 고객과 공유하기 위한 문서이다.

- 릴리즈 노트를 통해 테스트 진행 방법에 대한 결과와 소프트웨어 사양에 대한 개발팀의 정확한 준수 여부를 확인할 수 있다.
- 소프트웨어에 포함된 전체 기능, 서비스의 내용, 개선 사항 등을 사용자와 공유할 수 있다.
- 릴리즈 노트를 이용해 소프트웨어의 버전 관리나 릴리즈 정보를 체계적으로 관리할 수 있다.
- 릴리즈 노트는 소프트웨어의 초기 배포 시 또는 출시 후 개선 사항을 적용한 추가 배포 시에 제공한다.

핵심 101 릴리즈 노트 초기 버전 작성 시 고려사항

릴리즈 노트의 초기 버전은 다음의 사항을 고려하여 작성한다.

- 릴리즈 노트는 정확하고 완전한 정보를 기반으로 개발팀에서 직접 현재 시제로 작성해야 한다.
- 신규 소스, 빌드 등의 이력이 정확하게 관리되어 변경 또는 개선된 항목에 대한 이력 정보들도 작성되어야 한다.
- 릴리즈 노트 작성에 대한 표준 형식은 없지만 일반적으로 다음과 같은 항목이 포함된다.
  - Header(머릿말)
  - 개요
  - 목적
  - 문제 요약
  - 재현 항목
  - 수정/개선 내용
  - 사용자 영향도
  - SW 지원 영향도
  - 노트
  - 면책 조항
  - 연락처

핵심 102 릴리즈 노트 추가 버전 작성 시 고려사항

소프트웨어의 테스트 과정에서 베타 버전이 출시되거나 긴급한 버그 수정, 업그레이드와 같은 자체 기능 향상, 사용자 요청 등의 특수한 상황이 발생하는 경우 릴리즈 노트를 추가로 작성한다.

- 중대한 오류가 발생하여 긴급하게 수정하는 경우에는 릴리즈 버전을 출시하고 버그 번호를 포함한 모든 수정된 내용을 담아 릴리즈 노트를 작성한다.
- 소프트웨어에 대한 기능 업그레이드를 완료한 경우에는 릴리즈 버전을 출시하고 릴리즈 노트를 작성한다.
- 사용자로부터 접수된 요구사항에 의해 추가나 수정된 경우 자체 기능 향상과는 다른 별도의 릴리즈 버전으로 출시하고 릴리즈 노트를 작성한다.

핵심 103 저작권의 개요

저작권이란 소설, 시, 논문, 강연, 연술, 음악, 연극, 무용, 회화, 서예, 건축물, 사진, 영상, 지도, 도표, 컴퓨터 프로그램 저작물 등에 대하여 창작자가 가지는 배타적 독점적 권리로 타인의 침해를 받지 않을 고유한 권한이다.

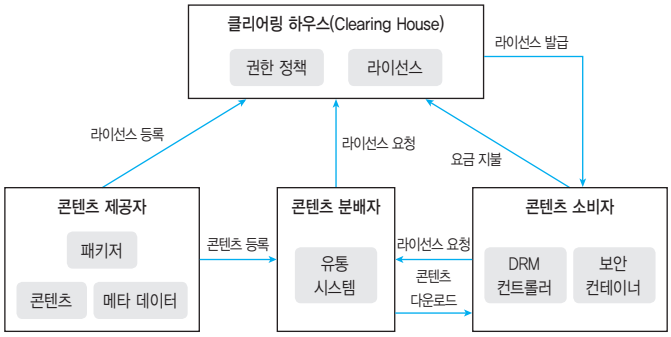
- 컴퓨터 프로그램들과 같이 복제하기 쉬운 저작물에 대해 불법 복제 및 배포 등을 막기 위한 기술적인 방법을 통칭해 저작권 보호 기술이라고 한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 104 디지털 저작권 관리(DRM; Digital Right Management)

디지털 저작권 관리는 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 디지털 콘텐츠의 생성, 유통, 이용까지의 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술이다.

디지털 저작권 관리의 흐름 및 구성 요소



클리어링 하우스 (Clearing House)	저작권에 대한 사용 권한, 라이선스 발급, 암호화된 키 관리, 사용량에 따른 결제 관리 등을 수행하는 곳
콘텐츠 제공자 (Contents Provider)	콘텐츠를 제공하는 저작권자
패키저(Packager)	콘텐츠를 메타 데이터와 함께 배포 가능한 형태로 묶어 암호화하는 프로그램
콘텐츠 분배자 (Contents Distributor)	암호화된 콘텐츠를 유통하는 곳이나 사람
콘텐츠 소비자 (Customer)	콘텐츠를 구매해서 사용하는 주체
DRM 컨트롤러 (DRM Controller)	배포된 콘텐츠의 이용 권한을 통제하는 프로그램
보안 컨테이너 (Security Container)	콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치

핵심 105 디지털 저작권 관리의 기술 요소

디지털 저작권 관리를 위해 사용되는 기술은 다음과 같다.

구성 요소	설명
암호화 (Encryption)	콘텐츠 및 라이선스를 암호화하고 전자 서명을 할 수 있는 기술
키 관리 (Key Management)	콘텐츠를 암호화한 키에 대한 저장 및 분배 기술
암호화 파일 생성 (Packager)	콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
식별 기술 (Identification)	콘텐츠에 대한 식별 체계 표현 기술
저작권 표현 (Right Expression)	라이선스의 내용 표현 기술
정책 관리 (Policy Management)	라이선스 발급 및 사용에 대한 정책 표현 및 관리 기술
크랙 방지 (Tamper Resistance)	크랙에 의한 콘텐츠 사용 방지 기술
인증 (Authentication)	라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술

핵심 106 소프트웨어 설치 매뉴얼의 개요

소프트웨어 설치 매뉴얼은 개발 초기에서부터 적용된 기준이나 사용자가 소프트웨어를 설치하는 과정에 필요한 내용을 기록한 설명서와 안내서이다.

- 설치 매뉴얼은 사용자 기준으로 작성한다.
- 설치 시작부터 완료할 때까지의 전 과정을 빠짐없이 순서대로 설명한다.
- 설치 과정에서 표시될 수 있는 오류 메시지 및 예외 상황에 관한 내용을 별도로 분류하여 설명한다.
- 소프트웨어 설치 매뉴얼에는 목차 및 개요, 서문, 기본 사항 등이 기본적으로 포함되어야 한다.

핵심 107 설치 매뉴얼 작성 방법

설치 매뉴얼은 사용자가 설치 과정을 이해하기 쉽도록 설치 화면을 누락 없이 캡처하고 순서대로 상세히 설명한다.

- 설치 매뉴얼에는 설치 화면 및 UI, 설치 이상 메시지, 설치 완료 및 결과, FAQ, 설치 시 점검 사항, Network 환경 및 보안, 고객 지원 방법, 준수 정보 및 제한 보증 등에 대한 내용을 기술한다.

설치 화면 및 UI	설치 실행과 메인 화면 및 안내창에 대한 내용을 기술한다.
설치 이상 메시지 설명	설치 방법이나 설치 환경이 잘못된 경우 표시될 수 있는 메시지에 대해 설명한다.
설치 완료 및 결과	설치 완료 화면을 수록하여 설치가 정상적으로 마무리되었음을 사용자에게 최종적으로 알려준다.
FAQ	설치 과정에서 사용자가 직면할 수 있는 문제 상황에 대비할 수 있도록, 설치 시 발생할 수 있는 다양한 상황을 FAQ로 정리하여 수록한다.
설치 시 점검 사항	설치 전 사용자의 설치 환경에 따라 점검해야 할 사항들이 무엇인지 설명한다.
Network 환경 및 보안	네트워크 오류로 인해 설치 시 문제가 발생하지 않도록 사전에 필요한 네트워크 연결 상태를 점검하도록 안내한다.
고객 지원 방법 (Customer Support)	설치와 관련하여 기술적인 지원이나 소프트웨어에 대한 서비스를 원할 경우 국가, 웹 사이트, 전화번호, 이메일 등 문의할 수 있는 연락처를 안내한다.
준수 정보 & 제한 보증 (Compliance Information & Limited Warranty)	Serial 보존, 불법 등록 사용 금지 등에 대한 준수 사항을 안내한다.

핵심 108 소프트웨어 사용자 매뉴얼의 개요

소프트웨어 사용자 매뉴얼은 사용자가 소프트웨어를 사용하는 과정에서 필요한 내용을 문서로 기록한 설명서와 안내서이다.

- 사용자 매뉴얼은 사용자가 소프트웨어 사용에 필요한 절차, 환경 등의 제반 사항이 모두 포함되도록 작성한다.
- 소프트웨어 배포 후 발생될 수 있는 오류에 대한 패치나 기능에 대한 업그레이드를 위해 매뉴얼의 버전을 관리한다.
- 개별적으로 동작이 가능한 컴포넌트 단위로 매뉴얼을 작성한다.
- 사용자 매뉴얼은 컴포넌트 명세서와 컴포넌트 구현 설계서를 토대로 작성한다.
- 사용자 매뉴얼에는 목차 및 개요, 서문, 기본 사항 등이 기본적으로 포함되어야 한다.

핵심 109 사용자 매뉴얼 작성 방법

사용자 매뉴얼은 사용자가 사용 방법을 이해하기 쉽도록 상황별로 누락 없이 캡처하여 순서대로 상세히 설명한다.

- 사용자 매뉴얼에는 사용자 화면 및 UI, 주요 기능 분류, 응용 프로그램 및 설정, 장치 연동, Network 환경, Profile 안내, 고객 지원 방법, 준수 정보 및 제한 보증 등에 대한 내용을 기술한다.

사용자 화면 및 UI	주의 사항과 참고 사항을 기술한다.
주요 기능 분류	기능이 실행되는 화면을 순서대로 캡처하여 기능에 대한 사용법을 설명한다.
응용 프로그램 (Program) 및 설정(Setting)	소프트웨어 구동 시 함께 실행해도 되는 응용 프로그램, 또는 함께 실행되면 안 되는 응용 프로그램에 대해 설명한다.
장치 연동	소프트웨어가 특정 장치(Device)에 내장되는 경우 연동되는 장치(Device)에 대해 설명한다.
Network 환경	Network에 접속되어 사용되는 소프트웨어인 경우 정상적인 연결을 위한 설정값 등을 설명한다.

시험에 나오는 것만 공부한다! 시나공시리즈

Profile 안내	Profile은 소프트웨어의 구동 환경을 점검하는 파일로, 사용자가 Profile의 경로를 변경하거나 위치를 이동하지 않도록 안내한다.
고객 지원 방법 (Customer Support)	사용과 관련하여 기술적인 지원이나 소프트웨어에 대한 서비스를 원할 경우 국가, 웹 사이트, 전화번호, 이메일 등 문의할 수 있는 연락처를 안내한다.
준수 정보 & 제한 보증(Compliance Information & Limited Warranty)	Serial 보존, 불법 등록 사용 금지 등에 대한 준수 사항을 안내한다.

핵심 110 소프트웨어 패키징의 형상 관리

형상 관리(SCM; Software Configuration Management)는 소프트웨어의 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동이다.

- 소프트웨어 변경의 원인을 알아내고 제어하며, 적절히 변경되고 있는지 확인하여 해당 담당자에게 통보한다.
- 형상 관리는 소프트웨어 개발의 전 단계에 적용되는 활동이며, 유지보수 단계에서도 수행된다.
- 형상 관리는 소프트웨어 개발의 전체 비용을 줄이고, 개발 과정의 여러 방해 요인이 최소화되도록 보증하는 것을 목적으로 한다.
- 관리 항목에는 소스 코드뿐만 아니라 각종 정의서, 지침서, 분석서 등이 포함된다.

핵심 111 형상 관리의 중요성

- 지속적인 소프트웨어의 변경 사항을 체계적으로 추적하고 통제할 수 있다.
- 제품 소프트웨어에 대한 무절제한 변경을 방지할 수 있다.
- 제품 소프트웨어에서 발견된 버그나 수정 사항을 추적할 수 있다.
- 소프트웨어는 형태가 없어 가시성이 결핍되므로 진행 정도를 확인하기 위한 기준으로 사용될 수 있다.
- 소프트웨어의 배포본을 효율적으로 관리할 수 있다.
- 소프트웨어를 여러 명의 개발자가 동시에 개발할 수 있다.

핵심 112 소프트웨어의 버전 등록 관련 주요 기능

항목	설명
저장소 (Repository)	최신 버전의 파일들과 변경 내역에 대한 정보들이 저장되어 있는 곳이다.
가져오기 (Import)	버전 관리가 되고 있지 않은 아무것도 없는 저장소(Repository)에 처음으로 파일을 복사한다.
체크아웃 (Check-Out)	• 프로그램을 수정하기 위해 저장소(Repository)에서 파일을 받아온다. • 소스 파일과 함께 버전 관리를 위한 파일들도 받아온다.
체크인 (Check-In)	체크아웃 한 파일의 수정을 완료한 후 저장소(Repository)의 파일을 새로운 버전으로 갱신한다.
커밋 (Commit)	체크인을 수행할 때 이전에 갱신된 내용이 있는 경우에는 충돌(Conflict)을 알리고 diff 도구를 이용해 수정한 후 갱신을 완료한다.
동기화 (Update)	저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화한다.

핵심 113 공유 폴더 방식 / 클라이언트/서버 방식 / 분산 저장소 방식

공유 폴더 방식

공유 폴더 방식은 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 방식으로, 다음과 같은 특징이 있다.

- 개발자들은 개발이 완료된 파일을 약속된 공유 폴더에 매일 복사한다.
- 담당자는 공유 폴더의 파일을 자기 PC로 복사한 후 컴파일 하여 이상 유무를 확인한다.
- 종류에는 SCCS, RCS, PVCS, QVCS 등이 있다.

클라이언트/서버 방식

클라이언트/서버 방식은 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식으로, 다음과 같은 특징이 있다.

- 서버의 자료를 개발자별로 자신의 PC(클라이언트)로 복사하여 작업한 후 변경된 내용을 서버에 반영한다.
- 모든 버전 관리는 서버에서 수행된다.
- 종류에는 CVS, SVN(Subversion), CVSNT, Clear Case, CMVC, Perforce 등이 있다.

분산 저장소 방식

분산 저장소 방식은 버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 방식으로, 다음과 같은 특징이 있다.

- 개발자별로 원격 저장소의 자료를 자신의 로컬 저장소로 복사하여 작업한 후 변경된 내용을 로컬 저장소에서 우선 반영(버전 관리)한 다음 이를 원격 저장소에 반영한다.
- 종류에는 Git, GNU arch, DCVS, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM 등이 있다.

핵심 114 Subversion(서브버전, SVN)

Subversion은 CVS를 개선한 것으로, 아파치 소프트웨어 재단에서 2000년에 발표하였다.

- 클라이언트/서버 구조로, 서버(저장소, Repository)에는 최신 버전의 파일들과 변경 내역이 관리된다.
- 서버의 자료를 클라이언트로 복사해와 작업한 후 변경 내용을 서버에 반영(Commit)한다.
- 모든 개발 작업은 trunk 디렉터리에서 수행되며, 추가 작업은 branches 디렉터리 안에 별도의 디렉터리를 만들어 작업을 완료한 후 trunk 디렉터리와 병합(merge)한다.
- 커밋(Commit)할 때마다 리비전(Revision)이 1씩 증가한다.
- 클라이언트는 대부분의 운영체제에서 사용되지만, 서버는 주로 유닉스를 사용한다.
- 소스가 오픈되어 있어 무료로 사용할 수 있다.
- CVS의 단점이었던 파일이나 디렉터리의 이름 변경, 이동 등이 가능하다.

핵심 115 Subversion의 주요 명령어

명령어	의미
add	• 새로운 파일이나 디렉터리를 버전 관리 대상으로 등록한다. • add로 등록되지 않은 대상은 commit이 적용되지 않는다.
commit	버전 관리 대상으로 등록된 클라이언트의 소스 파일을 서버의 소스 파일에 적용한다.
update	• 서버의 최신 commit 이력을 클라이언트의 소스 파일에 적용한다. • commit 전에는 매번 update를 수행하여 클라이언트에 적용되지 않은 서버의 변동 내역을 클라이언트에 적용한다.
checkout	버전 관리 정보와 소스 파일을 서버에서 클라이언트로 받아온다.
lock/unlock	서버의 소스 파일이나 디렉터리를 잠그거나 해제한다.
import	아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장하는 명령으로, 한 번 사용하면 다시 사용하지 않는다.
export	버전 관리에 대한 정보를 제외한 순수한 소스 파일만을 서버에서 받아온다.
info	지정한 파일에 대한 위치나 마지막 수정 일자 등에 대한 정보를 표시한다.
diff	지정된 파일이나 경로에 대해 이전 리비전과의 차이를 표시한다.
merge	다른 디렉터리에서 작업된 버전 관리 내역을 기본 개발 작업과 병합한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 116 Git(깃)

Git은 리누스 토발즈(Linus Torvalds)가 2005년 리눅스 커널 개발에 사용할 관리 도구로 개발한 이후 주니오 하마노(Junio Hamano)에 의해 유지 보수되고 있다.

- Git은 분산 버전 관리 시스템으로 2개의 저장소, 즉 지역(로컬) 저장소와 원격 저장소가 존재한다.
- 지역 저장소는 개발자들이 실제 개발을 진행하는 장소로, 버전 관리가 수행된다.
- 원격 저장소는 여러 사람들이 협업을 위해 버전을 공동 관리하는 곳으로, 자신의 버전 관리 내역을 반영하거나 다른 개발자의 변경 내용을 가져올 때 사용한다.
- 버전 관리가 지역 저장소에서 진행되므로 버전 관리가 신속하게 처리되고, 원격 저장소나 네트워크에 문제가 있어도 작업이 가능하다.
- 브랜치를 이용하면 기본 버전 관리 틀에 영향을 주지 않으면서 다양한 형태의 기능 테스팅이 가능하다.
- 파일의 변화를 스냅샷(Snapshot)으로 저장하는데, 스냅샷은 이전 스냅샷의 포인터를 가지므로 버전의 흐름을 파악할 수 있다.

핵심 117 Git의 주요 명령어

명령어	의미
add	• 작업 내역을 지역 저장소에 저장하기 위해 스테이징 영역(Staging Area)에 추가한다. • '--all' 옵션으로 작업 디렉터리의 모든 파일을 스테이징 영역에 추가할 수 있다.
commit	작업 내역을 지역 저장소에 저장한다.
branch	• 새로운 브랜치를 생성한다. • 최초로 commit을 하면 마스터(master) 브랜치가 생성된다. • commit 할 때마다 해당 브랜치는 가장 최근의 commit한 내용을 가리키게 된다. • '-d' 옵션으로 브랜치를 삭제할 수 있다.
checkout	• 지정한 브랜치로 이동한다. • 현재 작업 중인 브랜치는 HEAD 포인터가 가리키는데, checkout 명령을 통해 HEAD 포인터를 지정한 브랜치로 이동시킨다.
merge	지정한 브랜치의 변경 내역을 현재 HEAD 포인터가 가리키는 브랜치에 반영함으로써 두 브랜치를 병합한다.
init	지역 저장소를 생성한다.
remote add	원격 저장소에 연결한다.
push	로컬 저장소의 변경 내역을 원격 저장소에 반영한다.
fetch	원격 저장소의 변경 이력만을 지역 저장소로 가져와 반영한다.
clone	원격 저장소의 전체 내용을 지역 저장소로 복제한다.
fork	지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제한다.

핵심 118 빌드 자동화 도구의 개념

빌드란 소스 코드 파일들을 컴파일한 후 여러 개의 모듈을 묶어 실행 파일로 만드는 과정이며, 이러한 빌드를 포함하여 테스트 및 배포를 자동화하는 도구를 빌드 자동화 도구라고 한다.

- 애자일 환경에서는 하나의 작업이 마무리될 때마다 모듈 단위로 나눠서 개발된 코드들이 지속적으로 통합되는데, 이러한 지속적인 통합(Continuous Integration) 개발 환경에서 빌드 자동화 도구는 유용하게 활용된다.
- 빌드 자동화 도구에는 Ant, Make, Maven, Gradle, Jenkins 등이 있으며, 이중 Jenkins와 Gradle이 가장 대표적이다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 119 Jenkins/Gradle

Jenkins

Jenkins는 JAVA 기반의 오픈 소스 형태로, 가장 많이 사용되는 빌드 자동화 도구이다.

- 서블릿 컨테이너에서 실행되는 서버 기반 도구이다.
- SVN, Git 등 대부분의 형상 관리 도구와 연동이 가능하다.
- 친숙한 Web GUI 제공으로 사용이 쉽다.
- 여러 대의 컴퓨터를 이용한 분산 빌드나 테스트가 가능하다.

Gradle

Gradle은 Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구로, 안드로이드 앱 개발 환경에서 사용된다.

- 안드로이드 뿐만 아니라 플러그인을 설정하면, JAVA, C/C++, Python 등의 언어도 빌드가 가능하다.
- Groovy를 사용해서 만든 DSL(Domain Specific Language)을 스크립트 언어로 사용한다.
- Gradle은 실행할 처리 명령들을 모아 태스크(Task)로 만든 후 태스크 단위로 실행한다.
- 이전에 사용했던 태스크를 재사용하거나 다른 시스템의 태스크를 공유할 수 있는 빌드 캐시 기능을 지원하므로 빌드의 속도를 향상시킬 수 있다.

핵심 120 애플리케이션 테스트

애플리케이션 테스트의 개념

- 애플리케이션 테스트는 애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차이다.
- 애플리케이션 테스트는 개발된 소프트웨어가 고객의 요구사항을 만족시키는지 확인(Validation)하고 소프트웨어가 기능을 정확히 수행하는지 검증(Verification)한다.

애플리케이션 테스트의 기본 원리

- 애플리케이션 테스트는 소프트웨어의 잠재적인 결함을 줄일 수 있지만 소프트웨어에 결함이 없다고 증명할 수는 없다. 즉 완벽한 소프트웨어 테스팅은 불가능하다.
- 애플리케이션의 결함은 대부분 개발자의 특성이나 애플리케이션의 기능적 특징 때문에 특정 모듈에 집중되어 있다. 애플리케이션의 20%에 해당하는 코드에서 전체 80%의 결함이 발견된다고 하여 파레토 법칙을 적용하기도 한다.
- 애플리케이션 테스트에서는 동일한 테스트 케이스로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 '살충제 패러독스(Pesticide Paradox)' 현상이 발생한다. 살충제 패러독스를 방지하기 위해서 테스트 케이스를 지속적으로 보완 및 개선해야 한다.
- 애플리케이션 테스트는 소프트웨어 특징, 테스트 환경, 테스터 역량 등 정황(Context)에 따라 테스트 결과가 달라질 수 있으므로, 정황에 따라 테스트를 다르게 수행해야 한다.
- 소프트웨어의 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 말할 수 없다. 이것을 오류-부재의 궤변(Absence of Errors Fallacy)이라고 한다.
- 테스트와 위험은 반비례한다. 테스트를 많이 하면 할수록 미래에 발생할 위험을 줄일 수 있다.
- 테스트는 작은 부분에서 시작하여 점점 확대하며 진행해야 한다.
- 테스트는 개발자와 관계없는 별도의 팀에서 수행해야 한다.

핵심 121 애플리케이션 테스트의 분류

프로그램 실행 여부에 따른 테스트

정적 테스트	• 프로그램을 실행하지 않고 명세서나 소스 코드를 대상으로 분석하는 테스트이다. • 소프트웨어 개발 초기에 결함을 발견할 수 있어 소프트웨어의 개발 비용을 낮추는데 도움이 된다. • 종류 : 워크스루, 인스펙션, 코드 검사 등
동적 테스트	• 프로그램을 실행하여 오류를 찾는 테스트로, 소프트웨어 개발의 모든 단계에서 테스트를 수행할 수 있다. • 종류 : 블랙박스 테스트, 화이트박스 테스트

테스트 기반(Test Bases)에 따른 테스트

명세 기반 테스트	• 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 만들어 구현하고 있는지 확인하는 테스트이다. • 종류 : 동등 분할, 경계 값 분석 등
구조 기반 테스트	• 소프트웨어 내부의 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트이다. • 종류 : 구문 기반, 결정 기반, 조건 기반 등
경험 기반 테스트	• 유사 소프트웨어나 기술 등에 대한 테스터의 경험을 기반으로 수행하는 테스트이다. • 경험 기반 테스트는 사용자의 요구사항에 대한 명세가 불충분하거나 테스트 시간에 제약이 있는 경우 수행하면 효과적이다. • 종류 : 에러 추정, 체크 리스트, 탐색적 테스팅

시각에 따른 테스트

검증(Verification) 테스트	개발자의 시각에서 제품의 생산 과정을 테스트하는 것으로, 제품이 명세서대로 완성됐는지를 테스트한다.
확인(Validation) 테스트	사용자의 시각에서 생산된 제품의 결과를 테스트하는 것으로, 사용자가 요구한대로 제품이 완성됐는지, 제품이 정상적으로 동작하는지를 테스트한다.

목적에 따른 테스트

회복(Recovery) 테스트	시스템에 여러 가지 결함을 주어 실패하도록 한 후 올바르게 복구되는지를 확인하는 테스트이다.
안전(Security) 테스트	시스템에 설치된 시스템 보호 도구가 불법적인 침입으로부터 시스템을 보호할 수 있는지를 확인하는 테스트이다.
강도(Stress) 테스트	시스템에 과도한 정보량이나 빈도 등을 부과하여 과부하 시에도 소프트웨어가 정상적으로 실행되는지를 확인하는 테스트이다.
성능(Performance) 테스트	소프트웨어의 실시간 성능이나 전체적인 효율성을 진단하는 테스트로, 소프트웨어의 응답 시간, 처리량 등을 테스트한다.
구조(Structure) 테스트	소프트웨어 내부의 논리적인 경로, 소스 코드의 복잡도 등을 평가하는 테스트이다.
회귀(Regression) 테스트	소프트웨어의 변경 또는 수정된 코드에 새로운 결함이 없음을 확인하는 테스트이다.
병행(Parallel) 테스트	변경된 소프트웨어와 기존 소프트웨어에 동일한 데이터를 입력하여 결과를 비교하는 테스트이다.

핵심 122 화이트박스 테스트(White Box Test)

화이트박스 테스트는 모듈의 원시 코드를 오픈시킨 상태에서 원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법이다.

- 모듈 안의 작동을 직접 관찰한다.
- 원시 코드(모듈)의 모든 문장을 한 번 이상 실행함으로써 수행된다.

화이트박스 테스트의 종류

기초 경로 검사 (Base Path Testing)	• 대표적인 화이트박스 테스트 기법이다. • 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법으로, 테스트 측정 결과는 실행 경로의 기초를 정의하는 데 지침으로 사용된다.
제어 구조 검사 (Control Structure Testing)	• 조건 검사(Condition Testing) : 프로그램 모듈 내에 있는 논리적 조건을 테스트하는 테스트 케이스 설계 기법 • 루프 검사(Loop Testing) : 프로그램의 반복(Loop) 구조에 초점을 맞춰 실시하는 테스트 케이스 설계 기법 • 데이터 흐름 검사(Data Flow Testing) : 프로그램에서 변수의 정의와 변수 사용의 위치에 초점을 맞춰 실시하는 테스트 케이스 설계 기법

화이트박스 테스트의 검증 기준

문장 검증 기준 (Statement Coverage)	소스 코드의 모든 구문이 한 번 이상 수행되도록 테스트 케이스 설계
분기 검증 기준 (Branch Coverage)	소스 코드의 모든 조건문이 한 번 이상 수행되도록 테스트 케이스 설계
조건 검증 기준 (Condition Coverage)	소스 코드의 모든 조건문에 대해 조건이 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스 설계
분기/조건 기준 (Branch/Condition Coverage)	소스 코드의 모든 조건문과 각 조건문에 포함된 개별 조건식의 결과가 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스 설계

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 123 블랙박스 테스트(Black Box Test)

블랙박스 테스트는 소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 테스트로, 기능 테스트라고도 한다.

- 사용자의 요구사항 명세를 보면서 테스트하는 것으로, 주로 구현된 기능을 테스트한다.
- 소프트웨어 인터페이스에서 실시되는 테스트이다.

블랙박스 테스트의 종류

동치 분할 검사 (Equivalence Partitioning Testing, 동치 클래스 분해)	• 입력 자료에 초점을 맞춰 테스트 케이스(동치 클래스)를 만들고 검사하는 방법으로 동등 분할 기법이라고도 한다. • 프로그램의 입력 조건에 타당한 입력 자료와 타당하지 않은 입력 자료의 개수를 균등하게 하여 테스트 케이스를 정하고, 해당 입력 자료에 맞는 결과가 출력되는지 확인하는 기법이다.
경계값 분석 (Boundary Value Analysis)	• 입력 자료에만 치중한 동치 분할 기법을 보완하기 위한 기법이다. • 입력 조건의 중간값보다 경계값에서 오류가 발생될 확률이 높다는 점을 이용하여 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하는 기법이다.
원인-효과 그래프 검사 (Cause-Effect Graphing Testing)	입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법이다.
오류 예측 검사 (Error Guessing)	• 과거의 경험이나 확인자의 감각으로 테스트하는 기법이다. • 다른 블랙 박스 테스트 기법으로는 찾아낼 수 없는 오류를 찾아내는 일련의 보충적 검사 기법이며, 데이터 확인 검사라고도 한다.
비교 검사 (Comparison Testing)	여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법이다.

핵심 124 개발 단계에 따른 애플리케이션 테스트

단위 테스트 (Unit Test)	• 코딩 직후 소프트웨어 설계의 최소 단위인 모듈이나 컴포넌트에 초점을 맞춰 테스트하는 것이다. • 인터페이스, 외부적 I/O, 자료 구조, 독립적 기초 경로, 오류 처리 경로, 경계 조건 등을 검사한다.
통합 테스트 (Integration Test)	• 단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트를 의미한다. • 모듈 간 또는 통합된 컴포넌트 간의 상호 작용 오류를 검사한다.
시스템 테스트 (System Test)	• 개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행되는가를 점검하는 테스트이다. • 기능적 요구사항과 비기능적 요구사항으로 구분하여 각각을 만족하는지 테스트한다.
인수 테스트 (Acceptance Test)	• 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트하는 방법이다. • 개발한 소프트웨어를 사용자가 직접 테스트한다.

- 인수 테스트 종류

사용자 인수 테스트	사용자가 시스템 사용의 적절성 여부를 확인한다.
운영상의 인수 테스트	시스템 관리자가 시스템 인수 시 수행하는 테스트 기법으로, 백업/복원 시스템, 재난 복구, 사용자 관리, 정기 점검 등을 확인한다.
계약 인수 테스트	계약상의 인수/검수 조건을 준수하는지 여부를 확인한다.
규정 인수 테스트	소프트웨어가 정부 지침, 법규, 규정 등 규정에 맞게 개발되었는지 확인한다.
알파 테스트	• 개발자의 장소에서 사용자가 개발자 앞에서 행하는 테스트 기법이다. • 테스트는 통제된 환경에서 행해지며, 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인하면서 기록한다.
베타 테스트	• 선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법이다. • 실업무를 가지고 사용자가 직접 테스트하는 것으로, 개발자에 의해 제어되지 않은 상태에서 테스트가 행해지며, 발견된 오류와 사용상의 문제점을 기록하고 개발자에게 주기적으로 보고한다.

핵심 125 하향식 통합 테스트 (Top Down Integration Test)

하향식 통합 테스트는 프로그램의 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트하는 기법이다.

- 주요 제어 모듈을 기준으로 하여 아래 단계로 이동하면서 통합하는데, 이때 깊이 우선 통합법이나 넓이 우선 통합법을 사용한다.
- 테스트 초기부터 사용자에게 시스템 구조를 보여줄 수 있다.
- 상위 모듈에서는 테스트 케이스를 사용하기 어렵다.
- 하향식 통합 방법은 다음과 같은 절차로 수행된다.

❶ 주요 제어 모듈은 작성된 프로그램을 사용하고, 주요 제어 모듈의 종속 모듈들은 스텁(Stub)으로 대체한다.

※ 테스트 스텁(Test Stub) : 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로, 일시적으로 필요한 조건만을 가지고 있는 시험용 모듈이다.

❷ 깊이 우선 또는 넓이 우선 등의 통합 방식에 따라 하위 모듈인 스텁들이 한 번에 하나씩 실제 모듈로 교체된다.

❸ 모듈이 통합될 때마다 테스트를 실시한다.

❹ 새로운 오류가 발생하지 않음을 보증하기 위해 회귀 테스트를 실시한다.

핵심 126 상향식 통합 테스트 (Bottom Up Integration Test)

상향식 통합 테스트는 프로그램의 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트하는 기법이다.

- 가장 하위 단계의 모듈부터 통합 및 테스트가 수행되므로 스텁(Stub)은 필요하지 않지만, 하나의 주요 제어 모듈과 관련된 종속 모듈의 그룹인 클러스터(Cluster)가 필요하다.
- 상향식 통합 방법은 다음과 같은 절차로 수행된다.

❶ 하위 모듈들을 클러스터(Cluster)로 결합한다.

❷ 상위 모듈에서 데이터의 입 · 출력을 확인하기 위해 더미 모듈인 드라이버(Driver)를 작성한다.

※ 테스트 드라이버(Test Driver) : 테스트 드라이버는 테스트 대상의 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 도구이다.

❸ 통합된 클러스터 단위로 테스트한다.

❹ 테스트가 완료되면 클러스터는 프로그램 구조의 상위로 이동하여 결합하고 드라이버는 실제 모듈로 대체된다.

핵심 127 애플리케이션 테스트 프로세스

- 애플리케이션 테스트 프로세스는 개발된 소프트웨어가 사용자의 요구대로 만들어졌는지, 결함은 없는지 등을 테스트하는 절차이다.
- 순서

테스트 계획	프로젝트 계획서, 요구 명세서 등을 기반으로 테스트 목표를 정의하고 테스트 대상 및 범위를 결정한다.
테스트 분석 및 디자인	테스트의 목적과 원칙을 검토하고 사용자의 요구사항을 분석한다.
테스트 케이스 및 시나리오 작성	테스트 케이스의 설계 기법에 따라 테스트 케이스를 작성하고 검토 및 확인한 후 테스트 시나리오를 작성한다.
테스트 수행	• 테스트 환경을 구축한 후 테스트를 수행한다. • 테스트의 실행 결과를 측정하여 기록한다.
테스트 결과 평가 및 리포팅	테스트 결과를 비교 분석하여 테스트 결과서를 작성한다.
결함 추적 및 관리	테스트를 수행한 후 결함이 어디에서 발생했는지, 어떤 종류의 결함인지 등 결함을 추적하고 관리한다.

- 결함 관리 프로세스
  ❶ 에러 발견 : 에러가 발견되면 테스트 전문가와 프로젝트팀이 논의한다.
  ❷ 에러 등록 : 발견된 에러를 결함 관리 대장에 등록한다.
  ❸ 에러 분석 : 등록된 에러가 실제 결함인지 아닌지를 분석한다.
  ❹ 결함 확정 : 등록된 에러가 실제 결함이면 결함 확정 상태로 설정한다.
  ❺ 결함 할당 : 결함을 해결할 담당자에게 결함을 할당하고 결함 할당 상태로 설정한다.
  ❻ 결함 조치 : 결함을 수정하고, 수정이 완료되면 결함 조치 상태로 설정한다.
  ❼ 결함 조치 검토 및 승인 : 수정이 완료된 결함에 대해 확인 테스트를 수행하고, 이상이 없으면 결함 조치 완료 상태로 설정한다.

핵심 128 테스트 케이스 / 테스트 시나리오 / 테스트 오라클

테스트 케이스(Test Case)

- 테스트 케이스는 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서로, 명세 기반 테스트의 설계 산출물에 해당된다.
- 테스트 케이스를 미리 설계하면 테스트 오류를 방지할 수 있고 테스트 수행에 필요한 인력, 시간 등의 낭비를 줄일 수 있다.

테스트 시나리오(Test Scenario)

- 테스트 시나리오는 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스들을 묶은 집합으로, 테스트 케이스들을 적용하는 구체적인 절차를 명세한 문서이다.
- 테스트 시나리오에는 테스트 순서에 대한 구체적인 절차, 사전 조건, 입력 데이터 등이 설정되어 있다.

테스트 오라클(Test Oracle)

- 테스트 오라클은 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법 및 활동을 말한다.
- 테스트 오라클은 결과를 판단하기 위해 테스트 케이스에 대한 예상 결과를 계산하거나 확인한다.
- 테스트 오라클의 특징
  - 제한된 검증 : 테스트 오라클을 모든 테스트 케이스에 적용할 수 없다.
  - 수학적 기법 : 테스트 오라클의 값을 수학적 기법을 이용하여 구할 수 있다.
  - 자동화 가능 : 테스트 대상 프로그램의 실행, 결과 비교, 커버리지 측정 등을 자동화 할 수 있다.

테스트 오라클의 종류

참(True) 오라클	모든 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하는 오라클로, 발생된 모든 오류를 검출할 수 있다.
샘플링(Sampling) 오라클	특정한 몇몇 테스트 케이스의 입력 값들에 대해서만 기대하는 결과를 제공하는 오라클이다.
추정(Heuristic) 오라클	샘플링 오라클을 개선한 오라클로, 특정 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하고, 나머지 입력 값들에 대해서는 추정으로 처리하는 오라클이다.
일관성 검사(Consistent) 오라클	애플리케이션의 변경이 있을 때, 테스트 케이스의 수행 전과 후의 결과 값이 동일한지를 확인하는 오라클이다.

핵심 129 테스트 자동화 도구

테스트 자동화의 개념

테스트 자동화는 사람이 반복적으로 수행하던 테스트 절차를 스크립트 형태로 구현하는 자동화 도구를 적용함으로써 쉽고 효율적으로 테스트를 수행할 수 있도록 한 것이다.

테스트 자동화 도구의 유형

정적 분석 도구 (Static Analysis Tools)	프로그램을 실행하지 않고 분석하는 도구로, 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도 및 남은 결함 등을 발견하기 위해 사용된다.
테스트 실행 도구 (Test Execution Tools)	• 스크립트 언어를 사용하여 테스트를 실행하는 방법으로, 테스트 데이터와 테스트 수행 방법 등이 포함된 스크립트를 작성한 후 실행한다. • 데이터 주도 접근 방식 : 스프레드시트에 테스트 데이터를 저장하고, 이를 읽어 실행하는 방식이다. • 키워드 주도 접근 방식 : 스프레드시트에 테스트를 수행할 동작을 나타내는 키워드와 테스트 데이터를 저장하여 실행하는 방식이다.
성능 테스트 도구 (Performance Test Tools)	애플리케이션의 처리량, 응답 시간, 경과 시간, 자원 사용률 등을 인위적으로 적용한 가상의 사용자를 만들어 테스트를 수행함으로써 성능의 목표 달성 여부를 확인한다.
테스트 통제 도구 (Test Control Tools)	테스트 계획 및 관리, 테스트 수행, 결함 관리 등을 수행하는 도구로, 종류에는 형상 관리 도구, 결함 추적/관리 도구 등이 있다.
테스트 하네스 도구 (Test Harness Tools)	• 테스트 하네스는 애플리케이션의 컴포넌트 및 모듈을 테스트하는 환경의 일부분으로, 테스트를 지원하기 위해 생성된 코드와 데이터를 의미한다. • 테스트 하네스 도구는 테스트가 실행될 환경을 시뮬레이션 하여 컴포넌트 및 모듈이 정상적으로 테스트되도록 한다.

※ 테스트 하네스(Test Harness)의 구성 요소

- **테스트 드라이버(Test Driver)** : 테스트 대상의 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 도구
- **테스트 스텁(Test Stub)** : 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로, 일시적으로 필요한 조건만을 가지고 있는 테스트용 모듈
- **테스트 슈트(Test Suites)** : 테스트 대상 컴포넌트나 모듈, 시스템에 사용되는 테스트 케이스의 집합
- **테스트 케이스(Test Case)** : 사용자의 요구사항을 정확하게 준수했는지 확인하기 위한 입력 값, 실행 조건, 기대 결과 등으로 만들어진 테스트 항목의 명세서
- **테스트 스크립트(Test Script)** : 자동화된 테스트 실행 절차에 대한 명세서
- **목 오브젝트(Mock Object)** : 사전에 사용자의 행위를 조건부로 입력해 두면, 그 상황에 맞는 예정된 행위를 수행하는 객체

테스트 수행 단계별 테스트 자동화 도구

테스트 단계	자동화 도구	설명
테스트 계획	요구사항 관리	사용자의 요구사항 정의 및 변경사항 등을 관리하는 도구
테스트 분석/설계	테스트 케이스 생성	테스트 기법에 따른 테스트 데이터 및 테스트 케이스 작성을 지원하는 도구
테스트 수행	테스트 자동화	테스트의 자동화를 도와주는 도구로 테스트의 효율성을 높임
	정적 분석	코딩 표준, 런타임 오류 등을 검증하는 도구
	동적 분석	대상 시스템의 시뮬레이션을 통해 오류를 검출하는 도구
	성능 테스트	가상의 사용자를 생성하여 시스템의 처리 능력을 측정하는 도구
	모니터링	CPU, Memory 등과 같은 시스템 자원의 상태 확인 및 분석을 지원하는 도구
테스트 관리	커버리지 분석	테스트 완료 후 테스트의 충분성 여부 검증을 지원하는 도구
	형상 관리	테스트 수행에 필요한 다양한 도구 및 데이터를 관리하는 도구
	결함 추적/관리	테스트 시 발생한 결함 추적 및 관리 활동을 지원하는 도구

핵심 130 결함 관리

결함(Fault)의 정의

결함은 오류 발생, 작동 실패 등과 같이 소프트웨어가 개발자가 설계한 것과 다르게 동작하거나 다른 결과가 발생되는 것을 의미한다.

- 사용자가 예상한 결과와 실행 결과 간의 차이나 업무 내용과의 불일치 등으로 인해 변경이 필요한 부분도 모두 결함에 해당된다.

결함 상태 추적

테스트에서 발견된 결함은 지속적으로 상태 변화를 추적하고 관리해야 한다.

- 발견된 결함에 대해 결함 관리 측정 지표의 속성 값들을 분석하여 향후 결함이 발견될 모듈 또는 컴포넌트를 추정할 수 있다.
- 결함 관리 측정 지표

결함 분포	모듈 또는 컴포넌트의 특정 속성에 해당하는 결함 수 측정
결함 추세	테스트 진행 시간에 따른 결함 수의 추이 분석
결함 에이징	특정 결함 상태로 지속되는 시간 측정

결함 추적 순서

결함 추적은 결함이 발견된 때부터 결함이 해결될 때까지 전 과정을 추적하는 것으로 순서는 다음과 같다.

❶ 결함 등록(Open) : 테스터와 품질 관리(QA) 담당자에 의해 발견된 결함이 등록된 상태

❷ 결함 검토(Reviewed) : 등록된 결함을 테스터, 품질 관리(QA) 담당자, 프로그램 리더, 담당 모듈 개발자에 의해 검토된 상태

❸ 결함 할당(Assigned) : 결함을 수정하기 위해 개발자와 문제 해결 담당자에게 결함이 할당된 상태

❹ 결함 수정(Resolved) : 개발자가 결함 수정을 완료한 상태

❺ 결함 조치 보류(Deferred) : 결함의 수정이 불가능해 연기된 상태로, 우선순위, 일정 등에 따라 재오픈을 준비중인 상태

❻ 결함 종료(Closed) : 결함이 해결되어 테스터와 품질 관리(QA) 담당자가 종료를 승인한 상태

❼ 결함 해제(Clarified) : 테스터, 프로그램 리더, 품질 관리(QA) 담당자가 종료 승인한 결함을 검토하여 결함이 아니라고 판명한 상태

결함 분류

시스템 결함	시스템 다운, 애플리케이션의 작동 정지, 종료, 응답 시간 지연, 데이터베이스 에러 등 주로 애플리케이션 환경이나 데이터베이스 처리에서 발생된 결함
기능 결함	사용자의 요구사항 미반영/불일치, 부정확한 비즈니스 프로세스, 스크립트 오류, 타 시스템 연동 시 오류 등 애플리케이션의 기획, 설계, 업무 시나리오 등의 단계에서 유입된 결함
GUI 결함	UI 비일관성, 데이터 타입의 표시 오류, 부정확한 커서/메시지 오류 등 사용자 화면 설계에서 발생된 결함
문서 결함	사용자의 요구사항과 기능 요구사항의 불일치로 인한 불완전한 상태의 문서, 사용자의 온라인/오프라인 매뉴얼의 불일치 등 기획자, 사용자, 개발자 간의 의사소통 및 기록이 원활하지 않아 발생된 결함

결함 심각도

결함 심각도는 애플리케이션에 발생한 결함이 전체 시스템에 미치는 치명도를 나타내는 척도이다.

- 결함 심각도를 우선순위에 따라 High, Medium, Low 또는 치명적(Critical), 주요(Major), 보통(Normal), 경미(Minor), 단순(Simple) 등으로 분류된다.

결함 우선순위

결함의 우선순위는 발견된 결함 처리에 대한 신속성을 나타내는 척도로, 결함의 중요도와 심각도에 따라 설정되고 수정 여부가 결정된다.

핵심 131 애플리케이션 성능 분석

애플리케이션 성능이란 사용자가 요구한 기능을 최소한의 자원을 사용하여 최대한 많은 기능을 신속하게 처리하는 정도를 나타낸다.

- 애플리케이션 성능 측정 지표

처리량 (Throughput)	일정 시간 내에 애플리케이션이 처리하는 일의 양
응답 시간 (Response Time)	애플리케이션에 요청을 전달한 시간부터 응답이 도착할 때까지 걸린 시간
경과 시간 (Turn Around Time)	애플리케이션에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
자원 사용률 (Resource Usage)	애플리케이션이 의뢰한 작업을 처리하는 동안의 CPU 사용량, 메모리 사용량, 네트워크 사용량 등 자원 사용률

- 애플리케이션의 성능 분석 도구는 애플리케이션의 성능을 테스트하는 도구와 시스템을 모니터링하는 도구로 분류된다.
  - 성능 테스트 도구 : 애플리케이션의 성능을 테스트하기 위해 애플리케이션에 부하나 스트레스를 가하면서 애플리케이션의 성능 측정 지표를 점검하는 도구
  - 시스템 모니터링 도구 : 애플리케이션이 실행되었을 때 시스템 자원의 사용량을 확인하고 분석하는 도구

핵심 132 복잡도

복잡도(Complexity)는 시스템이나 시스템 구성 요소 또는 소프트웨어의 복잡한 정도를 나타나는 말로, 시스템 또는 소프트웨어를 어느 정도의 수준까지 테스트해야 하는지 또는 개발하는데 어느 정도의 자원이 소요되는지 예측하는데 사용된다.

시간 복잡도

시간 복잡도는 알고리즘의 실행시간, 즉 알고리즘을 수행하기 위해 프로세스가 수행하는 연산 횟수를 수치화한 것을 의미한다.

- 시간 복잡도가 낮을수록 알고리즘의 실행시간이 짧고, 높을수록 실행시간이 길어진다.
- 점근 표기법의 종류

빅오 표기법 (Big-O Notation)	• 알고리즘의 실행시간이 최악일 때를 표기하는 방법이다. • 입력값에 대해 알고리즘을 수행했을 때 명령어의 실행 횟수는 어떠한 경우에도 표기 수치보다 많을 수 없다.
세타 표기법 (Big-θ Notation)	• 알고리즘의 실행시간이 평균일 때를 표기하는 방법이다. • 입력값에 대해 알고리즘을 수행했을 때 명령어 실행 횟수의 평균적인 수치를 표기한다.
오메가 표기법 (Big-Ω Notation)	• 알고리즘의 실행시간이 최상일 때를 표기하는 방법이다. • 입력값에 대해 알고리즘을 수행했을 때 명령어의 실행 횟수는 어떠한 경우에도 표기 수치보다 적을 수 없다.

빅오 표기법(Big-O Notation)

빅오 표기법은 알고리즘의 실행시간이 최악일 때를 표기하는 방법으로, 신뢰성이 떨어지는 오메가 표기법이나 평가하기 까다로운 세타 표기법에 비해 성능을 예측하기 용이하여 주로 사용된다.

- 일반적인 알고리즘에 대한 최악의 시간 복잡도를 빅오 표기법으로 표현하면 다음과 같다.

O(1)	입력값(n)에 관계 없이 일정하게 문제 해결에 하나의 단계만을 거친다. 예 스택의 삽입(Push), 삭제(Pop)
$O(\log_2 n)$	문제 해결에 필요한 단계가 입력값(n) 또는 조건에 의해 감소한다. 예 이진 트리(Binary Tree), 이진 검색(Binary Search)
O(n)	문제 해결에 필요한 단계가 입력값(n)과 1:1의 관계를 가진다. 예 for문
$O(n\log_2 n)$	문제 해결에 필요한 단계가 $n(\log_2 n)$번만큼 수행된다. 예 힙 정렬(Heap Sort), 2-Way 합병 정렬(Merge Sort)
$O(n^2)$	문제 해결에 필요한 단계가 입력값(n)의 제곱만큼 수행된다. 예 삽입 정렬(Insertion Sort), 쉘 정렬(Shell Sort), 선택 정렬(Selection Sort), 버블 정렬(Bubble Sort), 퀵 정렬(Quick Sort)
$O(2^n)$	문제 해결에 필요한 단계가 2의 입력값(n) 제곱만큼 수행된다. 예 피보나치 수열(Fibonacci Sequence)

순환 복잡도

순환 복잡도(Cyclomatic Complexity)는 한 프로그램의 논리적인 복잡도를 측정하기 위한 소프트웨어의 척도로, 맥케이브 순환도(McCabe's Cyclomatic) 또는 맥케이브 복잡도 메트릭(McCabe's Complexity Metrics)라고도 하며, 제어 흐름도 이론에 기초를 둔다.

- 순환 복잡도를 이용하여 계산된 값은 프로그램의 독립적인 경로의 수를 정의하고, 모든 경로가 한 번 이상 수행되었음을 보장하기 위해 행해지는 테스트 횟수의 상한선을 제공한다.
- 제어 흐름도 G에서 순환 복잡도 V(G)는 다음과 같은 방법으로 계산할 수 있다.

  방법 1 순환 복잡도는 제어 흐름도의 영역 수와 일치하므로 영역 수를 계산한다.

  방법 2 $V(G) = E - N + 2$: E는 화살표 수, N은 노드의 수

핵심 133 애플리케이션 성능 개선

소스 코드 최적화

소스 코드 최적화는 나쁜 코드(Bad Code)를 배제하고, 클린 코드(Clean Code)로 작성하는 것이다.

- 클린 코드(Clean Code) : 누구나 쉽게 이해하고 수정 및 추가할 수 있는 단순, 명료한 코드, 즉 잘 작성된 코드를 의미한다.
- 나쁜 코드(Bad Code)
  - 프로그램의 로직(Logic)이 복잡하고 이해하기 어려운 코드로, 스파게티 코드와 외계인 코드가 여기에 해당한다.
  - 스파게티 코드 : 코드의 로직이 서로 복잡하게 얽혀 있는 코드
  - 외계인 코드 : 아주 오래되거나 참고문서 또는 개발자가 없어 유지보수 작업이 어려운 코드
- 클린 코드 작성 원칙

가독성	• 누구든지 코드를 쉽게 읽을 수 있도록 작성한다. • 코드 작성 시 이해하기 쉬운 용어를 사용하거나 들여쓰기 기능 등을 사용한다.
단순성	• 코드를 간단하게 작성한다. • 한 번에 한 가지를 처리하도록 코드를 작성하고 클래스/메소드/함수 등을 최소 단위로 분리한다.
의존성 배제	• 코드가 다른 모듈에 미치는 영향을 최소화한다. • 코드 변경 시 다른 부분에 영향이 없도록 작성한다.
중복성 최소화	• 코드의 중복을 최소화한다. • 중복된 코드는 삭제하고 공통된 코드를 사용한다.
추상화	상위 클래스/메소드/함수에서는 간략하게 애플리케이션의 특성을 나타내고, 상세 내용은 하위 클래스/메소드/함수에서 구현한다.

소스 코드 최적화 유형

- 클래스 분할 배치 : 하나의 클래스는 하나의 역할만 수행하도록 응집도를 높이고, 크기를 작게 작성한다.
- 느슨한 결합(Loosely Coupled) : 인터페이스 클래스를 이용하여 추상화된 자료 구조와 메소드를 구현함으로써 클래스 간의 의존성을 최소화한다.

소스 코드 품질 분석 도구

소스 코드 품질 분석 도구는 소스 코드의 코딩 스타일, 코드에 설정된 코딩 표준, 코드의 복잡도, 코드에 존재하는 메모리 누수 현상, 스레드 결함 등을 발견하기 위해 사용하는 분석 도구로, 크게 정적 분석 도구와 동적 분석 도구로 나뉜다.

- 정적 분석 도구
  - 작성한 소스 코드를 실행하지 않고 코딩 표준이나 코딩 스타일, 결함 등을 확인하는 코드 분석 도구이다.
  - 종류 : pmd, cppcheck, SonarQube, checkstyle, ccm, cobertura 등
- 동적 분석 도구
  - 작성한 소스 코드를 실행하여 코드에 존재하는 메모리 누수, 스레드 결함 등을 분석하는 도구이다.
  - 종류 : Avalanche, Valgrind 등

핵심 134 인터페이스 설계서

인터페이스 설계서는 시스템 사이의 데이터 교환 및 처리를 위해 교환 데이터 및 관련 업무, 송 · 수신 시스템 등에 대한 내용을 정의한 문서이다.

일반적인 인터페이스 설계서

시스템의 인터페이스 목록, 각 인터페이스의 상세 데이터 명세, 각 기능의 세부 인터페이스 정보를 정의한 문서이다.

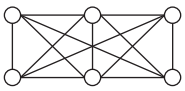
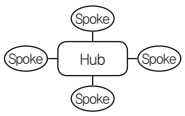
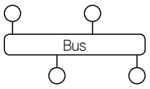
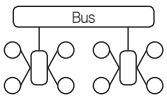
- 시스템 인터페이스 설계서 : 시스템 인터페이스 목록을 만들고 각 인터페이스 목록에 대한 상세 데이터 명세를 정의하는 것이다.
- 상세 기능별 인터페이스 명세서
  - 각 기능의 세부 인터페이스 정보를 정의한 문서이다.
  - 인터페이스를 통한 각 세부 기능의 개요, 세부 기능이 동작하기 전에 필요한 사전/사후 조건, 인터페이스 데이터, 호출 이후 결과를 확인하기 위한 반환값 등으로 구성된다.

정적 · 동적 모형을 통한 인터페이스 설계서

- 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 이용하여 만든 문서이다.
- 시스템을 구성하는 주요 구성 요소 간의 트랜잭션을 통해 해당 인터페이스가 시스템의 어느 부분에 속하고, 해당 인터페이스를 통해 상호 교환되는 트랜잭션의 종류를 확인할 수 있다.

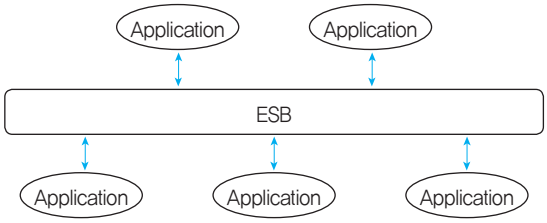
핵심 135 EAI(Enterprise Application Integration)

- EAI는 기업 내 각종 애플리케이션 및 플랫폼 간의 정보 전달, 연계, 통합 등 상호 연동이 가능하게 해주는 솔루션이다.
- EAI는 비즈니스 간 통합 및 연계성을 증대시켜 효율성 및 각 시스템 간의 확정성(Determinacy)을 높여 준다.
- EAI의 구축 유형은 다음과 같다.

유형	기능
Point-to-Point	• 가장 기본적인 애플리케이션 통합 방식으로, 애플리케이션을 1:1로 연결한다. • 변경 및 재사용이 어렵다. 
Hub & Spoke	• 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식이다. • 확장 및 유지 보수가 용이하다. • 허브 장애 발생 시 시스템 전체에 영향을 미친다. 
Message Bus (ESB 방식)	• 애플리케이션 사이에 미들웨어를 두어 처리하는 방식이다. • 확장성이 뛰어나며 대용량 처리가 가능하다. 
Hybrid	• Hub & Spoke와 Message Bus의 혼합 방식이다. • 그룹 내에서는 Hub & Spoke 방식을, 그룹 간에는 Message Bus 방식을 사용한다. • 필요한 경우 한 가지 방식으로 EAI 구현이 가능하다. • 데이터 병목 현상을 최소화할 수 있다. 

핵심 136 ESB(Enterprise Service Bus)

- ESB는 애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스를 제공하는 솔루션이다.
- ESB는 애플리케이션 통합 측면에서 EAI와 유사하지만 애플리케이션 보다는 서비스 중심의 통합을 지향한다.
- ESB는 특정 서비스에 국한되지 않고 범용적으로 사용하기 위하여 애플리케이션과의 결합도(Coupling)를 약하게(Loosely) 유지한다.
- 관리 및 보안 유지가 쉽고, 높은 수준의 품질 지원이 가능하다.



핵심 137 모듈 간 인터페이스 데이터 표준 확인

- 인터페이스 데이터 표준은 모듈 간 인터페이스에 사용되는 데이터의 형식을 표준화 하는 것이다.
- 인터페이스 데이터 표준은 기존의 데이터 중에서 공통 영역을 추출하거나 어느 한쪽의 데이터를 변환하여 정의한다.
- 확인된 인터페이스 데이터 표준은 인터페이스 기능 구현을 정의하는데 사용된다.

데이터 인터페이스 확인	• 데이터 표준을 위해 식별된 데이터 인터페이스에서 입 · 출력값의 의미와 데이터의 특성 등을 구체적으로 확인한다. • 확인된 데이터 인터페이스의 각 항목을 통해 데이터 표준을 확인한다.
인터페이스 기능 확인	• 데이터 표준을 위해 식별된 인터페이스 기능을 기반으로 인터페이스 기능 구현을 위해 필요한 데이터 항목을 확인한다. • 확인된 데이터 항목과 데이터 인터페이스에서 확인된 데이터 표준에서 수정 · 추가 · 삭제될 항목이 있는지 확인한다.
인터페이스 데이터 표준 확인	• 데이터 인터페이스에서 확인된 데이터 표준과 인터페이스 기능을 통해 확인된 데이터 항목들을 검토하여 최종적으로 데이터 표준을 확인한다. • 확인된 데이터 표준은 항목별로 데이터 인터페이스와 인터페이스 기능 중 출처를 구분하여 기록한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 138 인터페이스 기능 구현 정의

- 인터페이스 기능 구현의 정의는 인터페이스를 실제로 구현하기 위해 인터페이스 기능에 대한 구현 방법을 기능별로 기술한 것이다.
- 인터페이스 기능 구현 정의 순서

❶ 컴포넌트 명세서를 확인한다.

❷ 인터페이스 명세서를 확인한다.

❸ 일관된 인터페이스 기능 구현을 정의한다.

❹ 정의된 인터페이스 기능 구현을 정형화한다.

- 모듈 세부 설계서
  - 모듈 세부 설계서는 모듈의 구성 요소와 세부적인 동작 등을 정의한 설계서이다.
  - 대표적인 모듈 세부 설계서에는 컴포넌트 명세서와 인터페이스 명세서가 있다.

컴포넌트 명세서	컴포넌트의 개요 및 내부 클래스의 동작, 인터페이스를 통해 외부와 통신하는 명세 등을 정의한 것이다.
인터페이스 명세서	컴포넌트 명세서의 항목 중 인터페이스 클래스의 세부 조건 및 기능 등을 정의한 것이다.

- 인터페이스의 기능, 인터페이스 데이터 표준, 모듈 세부 설계서를 기반으로 일관성 있고 정형화된 인터페이스 기능 구현에 대해 정의한다.
- 일관성 있는 인터페이스 기능 구현 정의
  - 인터페이스의 기능, 인터페이스 데이터 표준, 모듈 세부 설계서를 통해 인터페이스의 기능 구현을 정의한다.
  - 정의한 인터페이스 기능 구현에 대해 송 · 수신 측에서 진행해야 할 절차까지 다시 세부적으로 정의한다.
- 정의된 인터페이스 기능 구현 정형화
  - 정의한 인터페이스 기능 구현을 특정 하드웨어나 소프트웨어에 의존적이지 않게 사람들이 보기 쉽고 표준화되도록 정형화한다.
  - 가독성을 높이려면 프로세스 형태나 유스케이스 다이어그램 형태로 정형화한다.

핵심 139 인터페이스 구현

인터페이스 구현은 송 · 수신 시스템 간의 데이터 교환 및 처리를 실현해 주는 작업을 의미한다.

- 정의된 인터페이스 기능 구현을 기반으로 구현 방법 및 범위 등을 고려하여 인터페이스 구현 방법을 분석한다.

데이터 통신을 이용한 인터페이스 구현

데이터 통신을 이용한 인터페이스 구현은 애플리케이션 영역에서 인터페이스 형식에 맞춘 데이터 포맷을 인터페이스 대상으로 전송하고 이를 수신 측에서 파싱(Parsing)하여 해석하는 방식이다.

- 주로 JSON이나 XML 형식의 데이터 포맷을 사용하여 인터페이스를 구현한다.

인터페이스 엔티티를 이용한 인터페이스 구현

인터페이스 엔티티를 이용한 인터페이스 구현은 인터페이스가 필요한 시스템 사이에 별도의 인터페이스 엔티티를 두어 상호 연계하는 방식이다.

- 일반적으로 인터페이스 테이블을 엔티티로 활용한다.
- 인터페이스 테이블은 한 개 또는 송신 및 수신 인터페이스 테이블을 각각 두어 활용한다.
- 송신 및 수신 인터페이스 테이블의 구조는 대부분 같지만 상황에 따라 서로 다르게 설계할 수도 있다.

※ JSON(JavaScript Object Notation)

JSON은 속성-값 쌍(Attribute-Value Pairs)으로 이루어진 데이터 객체를 전달하기 위해 사람이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다.

- 비동기 처리에 사용되는 AJAX에서 XML을 대체하여 사용되고 있다.

※ XML(eXtensible Markup Language)

XML은 특수한 목적을 갖는 마크업 언어를 만드는 데 사용되는 다목적 마크업 언어이다.

- 웹 페이지의 기본 형식인 HTML의 문법이 각 웹 브라우저에서 상호 호환적이지 못하다는 문제와 SGML의 복잡함을 해결하기 위하여 개발되었다.

※ AJAX(Asynchronous JavaScript and XML)

AJAX는 자바 스크립트(JavaScript) 등을 이용하여 클라이언트와 서버 간에 XML 데이터를 교환 및 제어함으로써 이용자가 웹 페이지와 자유롭게 상호 작용할 수 있도록 하는 비동기 통신 기술을 의미한다.

핵심 140 인터페이스 예외 처리

인터페이스 예외 처리는 구현된 인터페이스가 동작하는 과정에서 기능상 예외 상황이 발생 했을 때 이를 처리하는 절차를 말한다.

- 인터페이스 예외 처리는 인터페이스를 구현하는 방법에 따라 데이터 통신을 이용한 방법과 인터페이스 엔티티를 이용한 방법이 있다.

데이터 통신을 이용한 인터페이스 예외 처리

데이터 통신을 이용한 인터페이스 예외 처리 방법은 JSON, XML 등 인터페이스 객체를 이용해 구현한 인터페이스 동작이 실패할 경우를 대비한 것이다.

- 인터페이스 객체의 송 · 수신 시 발생할 수 있는 예외 케이스를 정의하고 각 예외 케이스마다 예외 처리 방법을 기술한다.
- 시스템 환경, 송 · 수신 데이터, 프로그램 자체 원인 등 다양한 원인으로 인해 예외 상황이 발생한다.

인터페이스 엔티티를 이용한 인터페이스 예외 처리

인터페이스 엔티티를 이용한 예외 처리 방법은 인터페이스 동작이 실패할 경우를 대비하여 해당 엔티티에 인터페이스의 실패 상황과 원인 등을 기록하고, 이에 대한 조치를 취할 수 있도록 사용자 및 관리자에서 알려주는 방식으로 예외 처리 방법을 정의한다.

핵심 141 인터페이스 보안

- 인터페이스는 시스템 모듈 간 통신 및 정보 교환을 위한 통로로 사용되므로 충분한 보안 기능을 갖추지 않으면 시스템 모듈 전체에 악영향을 주는 보안 취약점이 될 수 있다.
- 인터페이스의 보안성 향상을 위해서는 인터페이스의 보안 취약점을 분석한 후 적절한 보안 기능을 적용한다.
- 인터페이스 보안 기능은 일반적으로 네트워크, 애플리케이션, 데이터베이스 영역에 적용한다.

네트워크 영역	• 인터페이스 송 · 수신 간 스니핑(Sniffing) 등을 이용한 데이터 탈취 및 변조 위협을 방지하기 위해 네트워크 트래픽에 대한 암호화를 설정한다. • 암호화는 인터페이스 아키텍처에 따라 IPSec, SSL, S-HTTP 등의 다양한 방식으로 적용한다.
애플리케이션 영역	소프트웨어 개발 보안 가이드를 참조하여 애플리케이션 코드 상의 보안 취약점을 보완하는 방향으로 애플리케이션 보안 기능을 적용한다.
데이터베이스 영역	• 데이터베이스, 스키마, 엔티티의 접근 권한과 프로시저(Procedure), 트리거(Trigger) 등 데이터베이스 동작 객체의 보안 취약점에 보안 기능을 적용한다. • 개인 정보나 업무상 민감한 데이터의 경우 암호화나 익명화 등 데이터 자체의 보안 방안도 고려한다.

핵심 142 데이터 무결성 검사 도구

- 데이터 무결성 검사 도구는 시스템 파일의 변경 유무를 확인하고, 파일이 변경되었을 경우 이를 관리자에게 알려주는 도구로, 인터페이스 보안 취약점을 분석하는데 사용된다.
- 크래커나 허가받지 않은 내부 사용자들이 시스템에 침입하면 백도어를 만들어 놓거나 시스템 파일을 변경하여 자신의 흔적을 감추는데, 무결성 검사 도구를 이용하여 이를 감지할 수 있다.
- 해시(Hash) 함수를 이용하여 현재 파일 및 디렉토리의 상태를 DB에 저장한 후 감시하다가 현재 상태와 DB의 상태가 달라지면 관리자에게 변경 사실을 알려준다.
- 대표적인 데이터 무결성 검사 도구에는 Tripwire, AIDE, Samhain, Claymore, Slipwire, Fcheck 등이 있다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 143 연계 테스트

연계 테스트는 구축된 연계 시스템과 연계 시스템의 구성 요소가 정상적으로 동작하는지 확인하는 활동이다.

연계 테스트 케이스 작성	• 연계 시스템 간의 데이터 및 프로세스의 흐름을 분석하여 필요한 테스트 항목을 도출하는 과정이다. • 송 · 수신용 연계 응용 프로그램의 단위 테스트 케이스와 연계 테스트 케이스를 각각 작성한다.
연계 테스트 환경 구축	테스트의 일정, 방법, 절차, 소요 시간 등을 송 · 수신 기관과의 협의를 통해 결정하는 것이다.
연계 테스트 수행	연계 응용 프로그램을 실행하여 연계 테스트 케이스의 시험 항목 및 처리 절차 등을 실제로 진행하는 것이다.
연계 테스트 수행 결과 검증	연계 테스트 케이스의 시험 항목 및 처리 절차를 수행한 결과가 예상 결과와 동일한지를 확인하는 것이다.

핵심 144 인터페이스 구현 검증 도구

- 인터페이스 구현을 검증하기 위해서는 인터페이스 단위 기능과 시나리오 등을 기반으로 하는 통합 테스트가 필요하다.
- 통합 테스트는 다음과 같은 테스트 자동화 도구를 이용하면 효율적으로 수행할 수 있다.

도구	기능
xUnit	Java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크이다.
STAF	• 서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크이다. • 크로스 플랫폼, 분산 소프트웨어 테스트 환경을 조성할 수 있도록 지원한다. • 분산 소프트웨어의 경우 각 분산 환경에 설치된 데몬이 프로그램 테스트에 대한 응답을 대신하며, 테스트가 완료되면 이를 통합하고 자동화하여 프로그램을 완성한다.
FitNesse	웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크이다.
NTAF	FitNesse의 장점인 협업 기능과 STAF의 장점인 재사용 및 확장성을 통합한 NHN(Naver)의 테스트 자동화 프레임워크이다.
Selenium	다양한 브라우저 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크이다.
watir	Ruby를 사용하는 애플리케이션 테스트 프레임워크이다.

핵심 145 인터페이스 구현 감시 도구

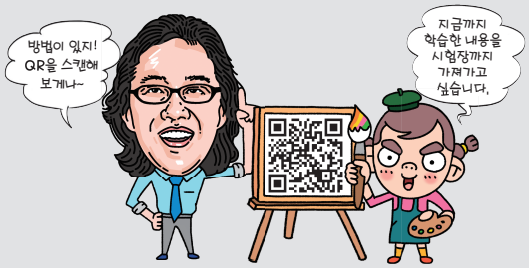
- 인터페이스 동작 상태는 APM을 사용하여 감시(Monitoring)할 수 있다.
- 애플리케이션 성능 관리 도구를 통해 데이터베이스와 웹 애플리케이션의 트랜잭션, 변수값, 호출 함수, 로그 및 시스템 부하 등 종합적인 정보를 조회하고 분석할 수 있다.
- 대표적인 애플리케이션 성능 관리 도구에는 스카우터(Scouter), 제니퍼(Jennifer) 등이 있다.

※ APM(Application Performance Management/Monitoring)

- APM은 애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구를 의미한다.
- APM은 리소스 방식과 엔드투엔드(End-to-End)의 두 가지 유형이 있다.
  - **리소스 방식** : Nagios, Zabbix, Cacti 등
  - **엔드투엔드 방식** : VisualVM, 제니퍼, 스카우터 등

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요? 까먹기 전에 다시 한 번 복습하고 싶다고요? 지금까지 공부한 내용을 안전하게 시험장까지 가져가는 완벽한 방법이 있습니다. 지금 당장 QR 코드를 스캔해 보세요.



www.membox.co.kr을 직접 입력해도 접속할 수 있습니다.

시험에 나오는 것만 공부한다! 시나공시리즈

3 과목 데이터베이스 구축

핵심 146 데이터베이스 설계의 개념 및 고려사항

데이터베이스 설계란 사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현하여 일반 사용자들이 사용하게 하는 것이다.

데이터베이스 설계 시 고려사항

무결성	삽입, 삭제, 갱신 등의 연산 후에도 데이터베이스에 저장된 데이터가 정해진 제약 조건을 항상 만족해야 한다.
일관성	데이터베이스에 저장된 데이터들 사이나, 특정 질의에 대한 응답이 처음부터 끝까지 변함없이 일정해야 한다.
회복	시스템에 장애가 발생했을 때 장애 발생 직전의 상태로 복구할 수 있어야 한다.
보안	불법적인 데이터의 노출 또는 변경이나 손실로부터 보호할 수 있어야 한다.
효율성	응답시간의 단축, 시스템의 생산성, 저장 공간의 최적화 등이 가능해야 한다.
데이터베이스 확장	데이터베이스 운영에 영향을 주지 않으면서 지속적으로 데이터를 추가할 수 있어야 한다.

핵심 147 데이터베이스 설계 순서

요구 조건 분석	요구 조건 명세서 작성
↓	
개념적 설계	개념 스키마, 트랜잭션 모델링, E-R 모델
↓	
논리적 설계	목표 DBMS에 맞는 논리 스키마 설계, 트랜잭션 인터페이스 설계
↓	
물리적 설계	목표 DBMS에 맞는 물리적 구조의 데이터로 변환
↓	
구현	목표 DBMS의 DDL(데이터 정의어)로 데이터베이스 생성, 트랜잭션 작성

핵심 148 개념적 설계(정보 모델링, 개념화)

개념적 설계란 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.

- 개념적 설계 단계에서는 개념 스키마 모델링과 트랜잭션 모델링을 병행 수행한다.
- 개념적 설계 단계에서는 요구 분석 단계에서 나온 결과인 요구 조건 명세를 DBMS에 독립적인 E-R 다이어그램으로 작성한다.
- DBMS에 독립적인 개념 스키마를 설계한다.

핵심 149 논리적 설계(데이터 모델링)

논리적 설계 단계란 현실 세계에서 발생하는 자료를 컴퓨터가 이해하고 처리할 수 있는 물리적 저장장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적 자료 구조로 변환(mapping)시키는 과정이다.

- 개념 세계의 데이터를 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계로 표현되는 논리적 구조의 데이터로 모델화한다.
- 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계에서는 개념 스키마를 평가 및 정제하고 DBMS에 따라 서로 다른 논리적 스키마를 설계하는 단계이다.
- 트랜잭션의 인터페이스를 설계한다.
- 관계형 데이터베이스라면 테이블을 설계하는 단계이다.

핵심 150 물리적 설계(데이터 구조화)

물리적 설계란 논리적 설계 단계에서 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.

- 물리적 설계 단계에서는 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정한다.
- 저장 레코드의 형식, 순서, 접근 경로, 조회가 집중되는 레코드와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.

핵심 151 데이터 모델

데이터 모델은 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형이다.

- 데이터 모델은 데이터, 데이터의 관계, 데이터의 의미 및 일관성, 제약 조건 등을 기술하기 위한 개념적 도구들의 모임이다.
- 현실 세계를 데이터베이스에 표현하는 중간 과정, 즉 데이터베이스 설계 과정에서 데이터의 구조(Schema)를 논리적으로 표현하기 위해 사용되는 지능적 도구이다.

데이터 모델의 구성 요소

개체(Entity)	데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.
속성(Attribute)	데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드에 해당한다.
관계(Relationship)	개체 간의 관계 또는 속성 간의 논리적인 연결을 의미한다.

데이터 모델에 표시할 요소

구조(Structure)	논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질을 표현한다.
연산(Operation)	데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구이다.
제약 조건(Constraint)	데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건이다.

핵심 152 개체(Entity)

개체의 정의 및 특징

개체(Entity)는 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.

- 개체는 실세계에 독립적으로 존재하는 유형, 무형의 정보로서 서로 연관된 몇 개의 속성으로 구성된다.
- 유일한 식별자(Unique Identifier)에 의해 식별이 가능하다.
- 다른 개체와 하나 이상의 관계(Relationship)가 있다.

개체 선정 방법

- 업무 분석에 관한 내용을 구체적으로 설명한 업무 기술서를 이용한다.
- 실제 업무를 담당하고 있는 담당자와 인터뷰를 한다.
- 업무 기술서와 인터뷰에서 확인하지 못한 정보가 있는지 실제 업무를 직접 견학하여 확인한다.
- 실제 업무에 사용되고 있는 장부와 전표를 이용한다.
- 이미 구축된 시스템이 있는 경우 해당 시스템의 산출물을 검토한다.

개체명 지정 방법

- 일반적으로 해당 업무에서 사용하는 용어로 지정한다.
- 약어 사용은 되도록 제한한다.
- 가능하면 단수 명사를 사용한다.
- 모든 개체명은 유일해야 한다.
- 가능하면 개체가 생성되는 의미에 따라 이름을 부여한다.

핵심 153 속성의 정의 및 특징

- 속성(Attribute)은 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당한다.
- 속성은 개체를 구성하는 항목이다.
- 속성은 개체의 특성을 기술한다.
- 속성의 수를 디그리(Degree) 또는 차수라고 한다.

핵심 154 속성의 종류

속성의 특성에 따른 분류

기본 속성 (Basic Attribute)	• 업무 분석을 통해 정의한 속성이다. • 속성 중 가장 많고 일반적이다. • 업무로부터 분석한 속성이라도 업무상 코드로 정의한 속성은 기본 속성에서 제외된다.
설계 속성 (Designed Attribute)	• 원래 업무상 존재하지 않고 설계 과정에서 도출해내는 속성이다. • 업무에 필요한 데이터 외에 데이터 모델링을 위해 업무를 규칙화하려고 속성을 새로 만들거나 변형하여 정의하는 속성이다.

파생 속성 (Derived Attribute)	• 다른 속성으로부터 계산이나 변형 등의 영향을 받아 발생하는 속성이다. • 파생 속성은 되도록 적은 수를 정의하는 것이 좋다.

개체 구성 방식에 따른 분류

기본키 속성(Primary Key Attribute)	개체를 식별할 수 있는 속성이다.
외래키 속성(Foreign Key Attribute)	다른 개체와의 관계에서 포함된 속성이다.
일반 속성	개체에 포함되어 있고 기본키, 외래키에 포함되지 않은 속성이다.

핵심 155 속성명 지정 원칙

- 속성명은 웹이나 클라이언트/서버(Client/Server) 등 어떠한 환경에서든 사용자 인터페이스에 나타나기 때문에 정확하고 혼란이 없도록 지정해야 한다.
- 속성명 지정 원칙
  - 해당 업무에서 사용하는 용어로 지정한다.
  - 서술형으로 지정하지 않는다.
  - 가급적이면 약어의 사용은 제한한다.
  - 개체명은 속성명으로 사용할 수 없다.
  - 개체에서 유일하게 식별 가능하도록 지정한다.

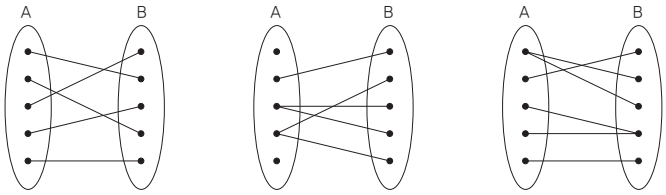
핵심 156 관계(Relationship)

관계는 개체와 개체 사이의 논리적인 연결을 의미한다.

- 관계에는 개체 간의 관계와 속성 간의 관계가 있다.

관계의 형태

일 대 일(1:1)	개체 집합 A의 각 원소가 개체 집합 B의 원소 한 개와 대응하는 관계이다.
일 대 다(1:N)	개체 집합 A의 각 원소는 개체 집합 B의 원소 여러 개와 대응하고 있지만, 개체 집합 B의 각 원소는 개체 집합 A의 원소 한 개와 대응하는 관계이다.
다 대 다(N:M)	개체 집합 A의 각 원소는 개체 집합 B의 원소 여러 개와 대응하고, 개체 집합 B의 각 원소도 개체 집합 A의 원소 여러 개와 대응하는 관계이다.



일 대 일(1:1) 일 대 다(1:N) 다 대 다(N:M)

관계의 종류

종속 관계 (Dependent Relationship)	두 개체 사이의 주 · 종 관계를 표현한 것으로, 식별 관계와 비식별 관계가 있다.
중복 관계 (Redundant Relationship)	두 개체 사이에 2번 이상의 종속 관계가 발생하는 관계이다.
재귀 관계 (Recursive Relationship)	개체가 자기 자신과 관계를 갖는 것으로, 순환 관계(Recursive Relationship)라고도 한다.
배타 관계 (Exclusive Relationship)	• 개체의 속성이나 구분자를 기준으로 개체의 특성을 분할하는 관계로, 배타 AND 관계와 배타 OR 관계로 구분한다. • 배타 AND 관계는 하위 개체들 중 속성이나 구분자 조건에 따라 하나의 개체만을 선택할 수 있고, 배타 OR 관계는 하나 이상의 개체를 선택할 수 있다.

핵심 157 식별자(Identifier)

분류	식별자
대표성 여부	• 주 식별자(Primary Identifier) : 개체를 대표하는 유일한 식별자 • 보조 식별자(Alternate Identifier) : 주 식별자를 대신하여 개체를 식별할 수 있는 속성
스스로 생성 여부	• 내부 식별자(Internal Identifier) : 개체 내에서 스스로 만들어지는 식별자 • 외부 식별자(Foreign Identifier) : 다른 개체와의 관계(Relationship)에 의해 외부 개체의 식별자를 가져와 사용하는 식별자
단일 속성 여부	• 단일 식별자(Single Identifier) : 주 식별자가 한 가지 속성으로만 구성된 식별자 • 복합 식별자(Composit Identifier) : 주 식별자가 두 개 이상의 속성으로 구성된 식별자
대체 여부	• 원조 식별자(Original Identifier) : 업무에 의해 만들어지는 가공되지 않은 원래의 식별자로, 본질 식별자라고도 함 • 대리 식별자(Surrogate Identifier) : 주 식별자의 속성이 두 개 이상인 경우 속성들을 하나의 속성으로 묶어 사용하는 식별자로, 인조 식별자라고도 함

※ 주 식별자의 특징

특징	내용
유일성	주 식별자에 의해 개체 내에 모든 인스턴스들이 유일하게 구분되어야 함
최소성	주 식별자를 구성하는 속성의 수는 유일성을 만족하는 최소 수가 되어야 함
불변성	주 식별자가 한 번 특정 개체에 지정되면 그 식별자는 변하지 않아야 함
존재성	주 식별자가 지정되면 식별자 속성에 반드시 데이터 값이 존재해야 함

핵심 158 E-R 모델의 개요

E-R 모델은 개념적 데이터 모델의 가장 대표적인 것으로, 1976년 피터 첸(Peter Chen)에 의해 제안되고 기본적인 구성 요소가 정립되었다.

- E-R 모델은 개체 타입(Entity Type)과 이들 간의 관계 타입(Relationship Type)을 이용해 현실 세계를 개념적으로 표현한다.
- E-R 모델에서는 데이터를 개체(Entity), 관계(Relationship), 속성(Attribute)으로 묘사한다.
- E-R 모델은 특정 DBMS를 고려한 것은 아니다.
- E-R 다이어그램으로 표현하며, 1:1, 1:N, N:M 등의 관계 유형을 제한 없이 나타낼 수 있다.

핵심 159 E-R 다이어그램

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	이중 타원	다중값 속성(복합 속성)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 성명은 성과 이름으로 구성
N M	관계	1:1, 1:N, N:M 등의 개체 간 관계에 대한 대응수를 선 위에 기술함
	선, 링크	개체 타입과 속성을 연결

핵심 160 관계형 데이터 모델

관계형 데이터 모델은 가장 널리 사용되는 데이터 모델로, 2차원적인 표(Table)를 이용해서 데이터 상호 관계를 정의하는 DB 구조를 말한다.

- 파일 구조처럼 구성한 테이블들을 하나의 DB로 묶어서 테이블 내에 있는 속성들 간의 관계(Relationship)를 설정하거나 테이블 간의 관계를 설정하여 이용한다.
- 기본키(Primary Key)와 이를 참조하는 외래키(Foreign Key)로 데이터 간의 관계를 표현한다.
- 계층 모델과 망 모델의 복잡한 구조를 단순화시킨 모델이다.
- 관계형 모델의 대표적인 언어는 SQL이다.
- 1:1, 1:N, N:M 관계를 자유롭게 표현할 수 있다.

핵심 161 관계형 데이터베이스의 Relation 구조

릴레이션은 데이터들을 표(Table)의 형태로 표현한 것으로 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성된다.

튜플(Tuple)

- 튜플은 릴레이션을 구성하는 각각의 행을 말한다.
- 튜플은 속성의 모임으로 구성된다.
- 파일 구조에서 레코드와 같은 의미이다.
- 튜플의 수를 카디널리티(Cardinality) 또는 기수, 대응수라고 한다.

속성(Attribute)

- 속성은 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당된다.
- 속성은 개체의 특성을 기술한다.
- 속성의 수를 디그리(Degree) 또는 차수라고 한다.

도메인(Domain)

- 도메인은 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합이다.
- 도메인은 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는데에도 이용된다.

  예 성별 애트리뷰트의 도메인은 '남'과 '여'로, 그 외의 값은 입력될 수 없다.

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

핵심 162 릴레이션의 특징

- 한 릴레이션에는 똑같은 튜플이 포함될 수 없으므로 릴레이션에 포함된 튜플들은 모두 상이하다.

  예 〈학생〉 릴레이션을 구성하는 김예소 레코드는 김예소에 대한 학적 사항을 나타내는 것으로 〈학생〉 릴레이션 내에서는 유일하다.
- 한 릴레이션에 포함된 튜플 사이에는 순서가 없다.

  예 〈학생〉 릴레이션에서 김예소 레코드와 고강민 레코드의 위치가 바뀌어도 상관없다.
- 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다.

  예 〈학생〉 릴레이션에 새로운 학생의 레코드를 삽입하거나 기존 학생에 대한 레코드를 삭제함으로써 테이블은 내용 면에서나 크기 면에서 변하게 된다.
- 릴레이션 스키마를 구성하는 속성들 간의 순서는 중요하지 않다.

  예 학번, 이름 등의 속성을 나열하는 순서가 이름, 학번 순으로 바뀌어도 데이터 처리에는 아무런 영향을 미치지 않는다.
- 속성의 유일한 식별을 위해 속성의 명칭은 유일해야 하지만, 속성을 구성하는 값은 동일한 값이 있을 수 있다.

  예 각 학생의 학년을 기술하는 속성인 '학년'은 다른 속성명들과 구분되어 유일해야 하지만 '학년' 속성에는 2, 1, 2, 4 등이 입력된 것처럼 동일한 값이 있을 수 있다.
- 릴레이션을 구성하는 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키(Key)로 설정한다.

  예 〈학생〉 릴레이션에서는 '학번'이나 '이름'이 튜플들을 구분하는 유일한 값인 키가 될 수 있다.
- 속성의 값은 논리적으로 더 이상 쪼갤 수 없는 원자값만을 저장한다.

  예 '학년'에 저장된 1, 2, 4 등은 더 이상 세분화할 수 없다.

핵심 163 키(Key)

키(Key)는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 튜플들을 서로 구분할 수 있는 기준이 되는 애트리뷰트를 말한다.

후보키 (Candidate Key)	• 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말한다. • 후보키는 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소성을 만족시켜야 한다.
기본키 (Primary Key)	• 후보키 중에서 특별히 선정된 주키(Main Key)로 중복된 값을 가질 수 없다. • 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성이다. • 기본키는 NULL 값을 가질 수 없다. 즉 튜플에서 기본키로 설정된 속성에는 NULL 값이 있어서는 안 된다.
대체키 (Alternate Key)	• 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키를 의미한다. • 보조키라고도 한다.
슈퍼키 (Super Key)	• 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키로서 릴레이션을 구성하는 모든 튜플들 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않는다. • 슈퍼키는 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족시키지만, 최소성은 만족시키지 못한다.
외래키 (Foreign Key)	• 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미한다. • 한 릴레이션에 속한 속성 A와 참조 릴레이션의 기본키인 B가 동일한 도메인 상에서 정의되었을 때의 속성 A를 외래키라고 한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 164 무결성(Integrity)

무결성이란 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제값이 일치하는 정확성을 의미한다.

개체 무결성(Entity Integrity, 실체 무결성)	기본 테이블의 기본키를 구성하는 어떤 속성도 Null 값이나 중복값을 가질 수 없다는 규정이다.
도메인 무결성(Domain Integrity, 영역 무결성)	주어진 속성 값이 정의된 도메인에 속한 값이어야 한다는 규정이다.
참조 무결성 (Referential Integrity)	외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 한다. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정이다.
사용자 정의 무결성 (User-Defined Integrity)	속성 값들이 사용자가 정의한 제약조건에 만족해야 한다는 규정이다.

핵심 165 관계대수의 개요

관계대수는 관계형 데이터베이스에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어이다.

- 관계대수는 릴레이션을 처리하기 위해 연산자와 연산 규칙을 제공하는 언어로 피연산자가 릴레이션이고, 결과도 릴레이션이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.
- 관계대수에는 관계 데이터베이스에 적용하기 위해 특별히 개발한 순수 관계 연산자와 수학적 집합 이론에서 사용하는 일반 집합 연산자가 있다.
- 순수 관계 연산자
  - Select
  - Project
  - Join
  - Division
- 일반 집합 연산자
  - UNION(합집합)
  - INTERSECTION(교집합)
  - DIFFERENCE(차집합)
  - CARTESIAN PRODUCT(교차곱)

핵심 166 순수 관계 연산자

Select	릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만드는 연산이다. • 릴레이션의 행(가로)에 해당하는 튜플을 구하는 것이므로 수평 연산이라고도 한다. • 연산자의 기호는 그리스 문자 시그마(σ)를 사용한다.
Project	주어진 릴레이션에서 속성 리스트(Attribute List)에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 연산이다. 단 연산 결과에 중복이 발생하면 중복이 제거된다. • 릴레이션의 열(세로)에 해당하는 Attribute를 추출하는 것이므로 수직 연산자라고도 한다. • 연산자의 기호는 그리스 문자 파이(π)를 사용한다.
Join	공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산이다. • 연산자의 기호는 ⋈를 사용한다.
Division	X⊃Y인 두 개의 릴레이션 R(X)와 S(Y)가 있을 때, R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구하는 연산이다. • 연산자의 기호는 ÷를 사용한다.

핵심 167 관계해석(Relational Calculus)

관계해석은 관계 데이터 모델의 제안자인 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.

- 관계해석은 관계 데이터의 연산을 표현하는 방법으로, 원하는 정보를 정의할 때는 계산 수식을 사용한다.
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.
- 튜플 관계해석과 도메인 관계해석이 있다.
- 기본적으로 관계해석과 관계대수는 관계 데이터베이스를 처리하는 기능과 능력면에서 동등하며, 관계대수로 표현한 식은 관계해석으로 표현할 수 있다.
- 질의어로 표현한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 168 정규화의 개요

정규화란 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.

- 하나의 종속성이 하나의 릴레이션에 표현될 수 있도록 분해해가는 과정이라 할 수 있다.
- 정규형에는 제1정규형, 제2정규형, 제3정규형, BCNF형, 제4정규형, 제5정규형이 있으며, 차수가 높아질수록 만족시켜야 할 제약 조건이 늘어난다.
- 정규화는 데이터베이스의 논리적 설계 단계에서 수행한다.
- 정규화는 논리적 처리 및 품질에 큰 영향을 미친다.
- 정규화된 데이터 모델은 일관성, 정확성, 단순성, 비중복성, 안정성 등을 보장한다.

핵심 169 정규화의 목적

- 데이터 구조의 안정성 및 무결성을 유지한다.
- 어떠한 릴레이션이라도 데이터베이스 내에서 표현 가능하게 만든다.
- 효과적인 검색 알고리즘을 생성할 수 있다.
- 데이터 중복을 배제하여 이상(Anomaly)의 발생 방지 및 자료 저장 공간의 최소화가 가능하다.
- 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다.
- 데이터 모형의 단순화가 가능하다.
- 속성의 배열 상태 검증이 가능하다.
- 개체와 속성의 누락 여부 확인이 가능하다.
- 자료 검색과 추출의 효율성을 추구한다.

핵심 170 이상(Anomaly)의 개념 및 종류

정규화를 거치지 않으면 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 못한 곤란한 현상이 발생하는데, 이를 이상(Anomaly)이라 하며 삽입 이상, 삭제 이상, 갱신 이상이 있다.

삽입 이상 (Insertion Anomaly)	릴레이션에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들도 함께 삽입되는 현상이다.
삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄가 일어나는 현상이다.
갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상이다.

핵심 171 정규화 과정

비정규 릴레이션
↓ 도메인이 원자값
1NF
↓ 부분적 함수 종속 제거
2NF
↓ 이행적 함수 종속 제거
3NF
↓ 결정자이면서 후보키가 아닌 것 제거
BCNF
↓ 다치 종속 제거
4NF
↓ 조인 종속성 이용
5NF

정규화 단계 암기 요령

두부를 좋아하는 정규화가 두부가게에 가서 가게에 있는 두부를 다 달라고 말하니 주인이 깜짝 놀라며 말했다.

두부이걸다줘? ≒ 도부이결다조

도메인이 원자값
부분적 함수 종속 제거
이행적 함수 종속 제거
결정자이면서 후보키가 아닌 것 제거
다치 종속 제거
조인 종속성 이용

※ 이행적 종속(Transitive Dependency) 관계 : A → B이고 B → C일 때 A → C를 만족하는 관계를 의미

※ 함수적 종속(Functional Dependency)

- 함수적 종속은 데이터들이 어떤 기준값에 의해 종속되는 것을 의미한다.
- 예를 들어 〈수강〉 릴레이션이 (학번, 이름, 과목명)으로 되어 있을 때, '학번'이 결정되면 '과목명'에 상관없이 '학번'에는 항상 같은 '이름'이 대응된다. '학번'에 따라 '이름'이 결정될 때 '이름'을 '학번'에 함수종속적이라고 하며 '학번 → 이름'과 같이 쓴다.

핵심 172 반정규화의 개념

반정규화란 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로, 의도적으로 정규화 원칙을 위배하는 행위이다.

- 반정규화를 수행하면 시스템의 성능이 향상되고 관리 효율성은 증가하지만 데이터의 일관성 및 정합성이 저하될 수 있다.
- 과도한 반정규화는 오히려 성능을 저하시킬 수 있다.
- 반정규화를 위해서는 사전에 데이터의 일관성과 무결성을 우선으로 할지, 데이터베이스의 성능과 단순화를 우선으로 할지를 결정해야 한다.
- 반정규화 방법에는 테이블 통합, 테이블 분할, 중복 테이블 추가, 중복 속성 추가 등이 있다.

핵심 173 반정규화 방법

테이블 통합	• 두 개의 테이블이 조인(Join)되는 경우가 많아 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행한다. • 두 개의 테이블에서 발생하는 프로세스가 동일하게 자주 처리되는 경우, 두 개의 테이블을 이용하여 항상 조회를 수행하는 경우 테이블 통합을 고려한다. • 테이블 통합의 종류에는 1:1 관계 테이블 통합, 1:N 관계 테이블 통합, 슈퍼타입/서브타입 테이블 통합이 있다.
테이블 분할	• 테이블을 수직 또는 수평으로 분할하는 것이다. • 수평 분할(Horizontal Partitioning) : 레코드(Record)를 기준으로 테이블을 분할하는 것이다. • 수직 분할(Vertical Partitioning) : 하나의 테이블에 속성이 너무 많을 경우 속성을 기준으로 테이블을 분할하는 것이다.
중복 테이블 추가	• 여러 테이블에서 데이터를 추출해서 사용해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가하여 작업의 효율성을 향상시킬 수 있다. • 중복 테이블 추가 방법 : 집계 테이블의 추가, 진행 테이블의 추가, 특정 부분만을 포함하는 테이블의 추가
중복 속성 추가	조인해서 데이터를 처리할 때 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추가하는 것이다.

핵심 174 시스템 카탈로그

시스템 카탈로그는 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.

- 시스템 카탈로그 내의 각 테이블은 사용자를 포함하여 DBMS에서 지원하는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지 관리하는 시스템 테이블이다.
- 카탈로그들이 생성되면 데이터 사전(Data Dictionary)에 저장되기 때문에 좁은 의미로는 카탈로그를 데이터 사전이라고도 한다.
- 시스템 카탈로그에 저장된 정보를 메타 데이터(Meta-Data)라고 한다.
- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자도 SQL을 이용하여 내용을 검색해 볼 수 있다.
- INSERT, DELETE, UPDATE문으로 카탈로그를 갱신하는 것은 허용되지 않는다.
- 데이터베이스 시스템에 따라 상이한 구조를 갖는다.
- 카탈로그는 DBMS가 스스로 생성하고 유지한다.
- **카탈로그의 갱신** : 사용자가 SQL문을 실행시켜 기본 테이블, 뷰, 인덱스 등에 변화를 주면 시스템이 자동으로 갱신한다.

※ Data Directory

- 데이터 사전에 수록된 데이터를 실제로 접근하는 데 필요한 정보를 관리 유지하는 시스템이다.
- 시스템 카탈로그는 사용자와 시스템 모두 접근할 수 있지만 데이터 디렉터리는 시스템만 접근할 수 있다.

핵심 175 물리 데이터베이스 설계

물리 데이터베이스 설계는 논리적 구조로 표현된 논리적 데이터베이스를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.

- 물리적 데이터베이스 구조의 기본적인 데이터 단위는 저장 레코드(Stored Record)이다.
- 물리적 설계 단계에 꼭 포함되어야 할 것은 저장 레코드의 양식 설계, 레코드 집중(Record Clustering)의 분석 및 설계, 접근 경로 설계 등이다.
- 물리적 데이터베이스 구조는 여러 가지 타입의 저장 레코드 집합이라는 면에서 단순한 파일과 다르다.
- 물리적 데이터베이스 구조는 데이터베이스 시스템의 성능에 중대한 영향을 미친다.
- 물리적 설계 시 고려 사항
  - 인덱스 구조
  - 레코드 크기
  - 파일에 존재하는 레코드 개수
  - 파일에 대한 트랜잭션의 갱신과 참조 성향
  - 성능 향상을 위한 개념 스키마의 변경 여부 검토
  - 빈번한 질의와 트랜잭션들의 수행속도를 높이기 위한 고려
  - 시스템 운용 시 파일 크기의 변화 가능성

핵심 176 데이터베이스 저장 공간 설계

테이블(Table)

테이블은 데이터베이스의 가장 기본적인 객체로 로우(Row, 행)와 컬럼(Column, 열)으로 구성되어 있다.

- 테이블의 종류

일반 테이블	현재 사용되는 대부분의 DBMS에서 표준 테이블로 사용되는 테이블 형태이다.
클러스터드 인덱스 테이블(Clustered Index Table)	기본키(Primary Key)나 인덱스키의 순서에 따라 데이터가 저장되는 테이블이다.
파티셔닝 테이블 (Partitioning Table)	대용량의 테이블을 작은 논리적 단위인 파티션(Partition)으로 나눈 테이블이다.
외부 테이블 (External Table)	데이터베이스에서 일반 테이블처럼 이용할 수 있는 외부 파일로, 데이터베이스 내에 객체로 존재한다.
임시 테이블 (Temporary Table)	트랜잭션이나 세션별로 데이터를 저장하고 처리할 수 있는 테이블이다.

컬럼(Column)

컬럼은 테이블의 열을 구성하는 요소로 데이터 타입(Data Type), 길이(Length) 등으로 정의된다.

- 데이터 타입은 데이터의 일관성 유지를 위해 사용되는 가장 기본적인 것으로, 도메인을 정의한 경우 도메인에 따라 데이터의 타입과 길이가 정의된다.
- 두 컬럼을 비교하는 연산에서 두 컬럼의 데이터 타입이나 길이가 다르면 DBMS 내부적으로 데이터 타입을 변환한 후 비교 연산을 수행한다.
- 참조 관계인 컬럼들은 데이터 타입과 길이가 일치해야 한다.

테이블스페이스(Tablespace)

테이블스페이스는 테이블이 저장되는 논리적인 영역으로, 하나의 테이블스페이스에 하나 또는 그 이상의 테이블을 저장할 수 있다.

- 테이블을 저장하면 논리적으로는 테이블스페이스에 저장되고, 물리적으로는 해당 테이블스페이스와 연관된 데이터 파일(Data File)에 저장된다.
- 데이터베이스를 테이블, 테이블스페이스, 데이터 파일로 나눠 관리하면 논리적 구성이 물리적 구성에 종속되지 않아 투명성이 보장된다.
- 테이블스페이스는 데이터베이스에 저장되는 내용에 따라 테이블, 인덱스, 임시(Temporary) 등의 용도로 구분하여 설계한다.

핵심 177 트랜잭션 분석 / CRUD 분석

트랜잭션(Transaction)

트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.

- 트랜잭션은 데이터베이스 시스템에서 병행 제어 및 회복 작업 시 처리되는 작업의 논리적 단위로 사용된다.

- 트랜잭션의 특성

Atomicity (원자성)	• 트랜잭션의 연산은 데이터베이스에 모두 반영되도록 완료(Commit)되든지 아니면 전혀 반영되지 않도록 복구(Rollback)되어야 한다. • 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 한다.
Consistency (일관성)	• 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다. • 시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 한다.
Isolation(독립성, 격리성, 순차성)	• 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없다. • 수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없다.
Durability (영속성, 지속성)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 한다.

CRUD 분석

CRUD는 '생성(Create), 읽기(Read), 갱신(Update), 삭제(Delete)'의 앞 글자만 모아서 만든 용어이며, CRUD 분석은 데이터베이스 테이블에 변화를 주는 트랜잭션의 CRUD 연산에 대해 CRUD 매트릭스를 작성하여 분석하는 것이다.

- CRUD 분석으로 테이블에 발생되는 트랜잭션의 주기별 발생 횟수를 파악하고 연관된 테이블들을 분석하면 테이블에 저장되는 데이터의 양을 유추할 수 있다.

CRUD 매트릭스

CRUD 매트릭스는 2차원 형태의 표로서, 행(Row)에는 프로세스를, 열(Column)에는 테이블을, 행과 열이 만나는 위치에는 프로세스가 테이블에 발생시키는 변화를 표시하는 업무 프로세스와 데이터 간 상관 분석표이다.

- CRUD 매트릭스를 통해 프로세스의 트랜잭션이 테이블에 수행하는 작업을 검증한다.

트랜잭션 분석

트랜잭션 분석의 목적은 CRUD 매트릭스를 기반으로 테이블에 발생하는 트랜잭션 양을 분석하여 테이블에 저장되는 데이터의 양을 유추하고 이를 근거로 DB 용량을 산정하고 DB 구조를 최적화하는 것이다.

- 트랜잭션 분석은 업무 개발 담당자가 수행한다.

트랜잭션 분석서

트랜잭션 분석서는 단위 프로세스와 CRUD 매트릭스를 이용하여 작성하며, 구성 요소에는 단위 프로세스, CRUD 연산, 테이블명, 컬럼명, 테이블 참조 횟수, 트랜잭션 수, 발생 주기 등이 있다.

핵심 178 인덱스(Index)

인덱스는 데이터 레코드를 빠르게 접근하기 위해 〈키 값, 포인터〉 쌍으로 구성되는 데이터 구조이다.

- 인덱스는 데이터가 저장된 물리적 구조와 밀접한 관계가 있다.
- 인덱스는 레코드가 저장된 물리적 구조에 접근하는 방법을 제공한다.
- 인덱스를 통해서 파일의 레코드에 대한 액세스를 빠르게 수행할 수 있다.
- 레코드의 삽입과 삭제가 수시로 일어나는 경우에는 인덱스의 개수를 최소로 하는 것이 효율적이다.
- 인덱스의 종류

트리 기반 인덱스	인덱스를 저장하는 블록들이 트리 구조를 이루고 있는 것으로, 상용 DBMS에서는 트리 구조 기반의 B+ 트리 인덱스를 주로 활용한다.
비트맵 인덱스	인덱스 컬럼의 데이터를 Bit 값인 0 또는 1로 변환하여 인덱스 키로 사용하는 방법이다.
함수 기반 인덱스	• 컬럼의 값 대신 컬럼에 특정 함수(Function)나 수식(Expression)을 적용하여 산출된 값을 사용하는 것으로, B+ 트리 인덱스 또는 비트맵 인덱스를 생성하여 사용한다. • 데이터를 입력하거나 수정할 때 함수를 적용해야 하므로 부하가 발생할 수 있다.
비트맵 조인 인덱스	다수의 조인된 객체로 구성된 인덱스로, 단일 객체로 구성된 일반적인 인덱스와 액세스 방법이 다르다.
도메인 인덱스	개발자가 필요한 인덱스를 직접 만들어 사용하는 것으로, 확장형 인덱스(Extensible Index)라고도 한다.

※ 클러스터드 인덱스 / 넌클러스터드 인덱스

- 클러스터드 인덱스(Clustered Index) : 인덱스 키의 순서에 따라 데이터가 정렬되어 저장되는 방식
- 넌클러스터드 인덱스(Non-Clustered Index) : 인덱스의 키 값만 정렬되어 있을 뿐 실제 데이터는 정렬되지 않는 방식

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 179 뷰(View)

뷰는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다.

- 뷰는 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- 뷰는 데이터 보정 작업, 처리 과정 시험 등 임시적인 작업을 위한 용도로 활용된다.

뷰(View)의 특징

- 뷰는 기본 테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조를 사용하며, 조작도 기본 테이블과 거의 같다.
- 뷰는 가상 테이블이기 때문에 물리적으로 구현되어 있지 않다.
- 데이터의 논리적 독립성을 제공할 수 있다.
- 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해진다.
- 뷰를 통해서만 데이터에 접근하게 하면 뷰에 나타나지 않는 데이터를 안전하게 보호하는 효율적인 기법으로 사용할 수 있다.
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다.
- 일단 정의된 뷰는 다른 뷰의 정의에 기초가 될 수 있다.
- 뷰를 정의할 때는 CREATE문, 제거할 때는 DROP문을 사용한다.

뷰(View)의 장 · 단점

장점	단점
• 논리적 데이터 독립성을 제공한다. • 동일 데이터에 대해 동시에 여러 사용자의 상이한 응용이나 요구를 지원해 준다. • 사용자의 데이터 관리를 간단하게 해준다. • 접근 제어를 통한 자동 보안이 제공된다.	• 독립적인 인덱스를 가질 수 없다. • 뷰의 정의를 변경할 수 없다. • 뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따른다.

핵심 180 클러스터(Cluster)

클러스터는 데이터 저장 시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법이다.

- 클러스터링키로 지정된 컬럼 값의 순서대로 저장되고, 여러 개의 테이블이 하나의 클러스터에 저장된다.

클러스터(Cluster)의 특징

- 클러스터링 된 테이블은 데이터 조회 속도는 향상시키지만 데이터 입력, 수정, 삭제에 대한 성능은 저하시킨다.
- 클러스터는 데이터의 분포도가 넓을수록 유리하다.
- 데이터 분포도가 넓은 테이블을 클러스터링 하면 저장 공간을 절약할 수 있다.
- 클러스터링된 테이블은 클러스터링키 열을 공유하므로 저장 공간이 줄어든다.
- 대용량을 처리하는 트랜잭션은 전체 테이블을 스캔하는 일이 자주 발생하므로 클러스터링을 하지 않는 것이 좋다.
- 파티셔닝된 테이블에는 클러스터링을 할 수 없다.

핵심 181 파티션(Patition)

데이터베이스에서 파티션은 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것을 말한다.

- 대용량 DB의 경우 중요한 몇 개의 테이블에만 집중되어 데이터가 증가되므로, 이런 테이블들을 작은 단위로 나눠 분산시키면 성능 저하를 방지할 뿐만 아니라 데이터 관리도 쉬워진다.

파티션의 장 · 단점

장점	• 데이터 접근 시 액세스 범위를 줄여 쿼리 성능이 향상된다. • 파티션별로 데이터가 분산되어 저장되므로 디스크의 성능이 향상된다. • 파티션별로 백업 및 복구를 수행하므로 속도가 빠르다. • 시스템 장애 시 데이터 손상 정도를 최소화할 수 있다. • 데이터 가용성이 향상된다. • 파티션 단위로 입 · 출력을 분산시킬 수 있다.
단점	• 하나의 테이블을 세분화하여 관리하므로 세심한 관리가 요구된다. • 테이블간 조인에 대한 비용이 증가한다. • 용량이 작은 테이블에 파티셔닝을 수행하면 오히려 성능이 저하된다.

파티션의 종류

범위 분할 (Range Partitioning)	지정한 열의 값을 기준으로 분할한다. 예 일별, 월별, 분기별 등
해시 분할 (Hash Partitioning)	• 해시 함수를 적용한 결과 값에 따라 데이터를 분할한다. • 특정 파티션에 데이터가 집중되는 범위 분할의 단점을 보완한 것으로, 데이터를 고르게 분산할 때 유용하다. • 특정 데이터가 어디에 있는지 판단할 수 없다. • 고객번호, 주민번호 등과 같이 데이터가 고른 컬럼에 효과적이다.
조합 분할 (Composite Partitioning)	• 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할하는 방식이다. • 범위 분할한 파티션이 너무 커서 관리가 어려울 때 유용하다.

핵심 182 데이터베이스 용량 설계

데이터베이스 용량 설계

데이터베이스 용량 설계는 데이터가 저장될 공간을 정의하는 것이다.

- 데이터베이스 용량을 설계할 때는 테이블에 저장할 데이터양과 인덱스, 클러스터 등이 차지하는 공간 등을 예측하여 반영해야 한다.

데이터베이스 용량 설계의 목적

- 데이터베이스의 용량을 정확히 산정하여 디스크의 저장 공간을 효과적으로 사용하고 확장성 및 가용성을 높인다.
- 디스크의 특성을 고려하여 설계함으로써 디스크의 입·출력 부하를 분산시키고 채널의 병목 현상을 최소화한다.
- 디스크에 대한 입·출력 경합이 최소화되도록 설계함으로써 데이터 접근성이 향상된다.
- 데이터 접근성을 향상시키는 설계 방법
  - 테이블의 테이블스페이스와 인덱스의 테이블스페이스를 분리하여 구성한다.
  - 테이블스페이스와 임시 테이블스페이스를 분리하여 구성한다.
  - 테이블을 마스터 테이블과 트랜잭션 테이블로 분류한다.
- 데이터베이스에 생성되는 오브젝트의 익스텐트 발생을 최소화하여 성능을 향상시킨다.
- 데이터베이스 용량을 정확히 분석하여 테이블과 인덱스에 적합한 저장 옵션을 지정한다.

핵심 183 분산 데이터베이스 설계

분산 데이터베이스는 논리적으로는 하나의 시스템에 속하지만 물리적으로는 네트워크를 통해 연결된 여러 개의 컴퓨터 사이트(Site)에 분산되어 있는 데이터베이스를 말한다.

분산 데이터베이스의 목표

- 위치 투명성(Location Transparency) : 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있다.
- 중복 투명성(Replication Transparency) : 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행한다.
- 병행 투명성(Concurrency Transparency) : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않는다.

분산 데이터베이스의 장·단점

장점	단점
• 지역 자치성이 높다. • 자료의 공유성이 향상된다. • 분산 제어가 가능하다. • 시스템 성능이 향상된다. • 중앙 컴퓨터의 장애가 전체 시스템에 영향을 끼치지 않는다. • 효용성과 융통성이 높다. • 신뢰성 및 가용성이 높다. • 점진적 시스템 용량 확장이 용이하다.	• DBMS가 수행할 기능이 복잡하다. • 데이터베이스 설계가 어렵다. • 소프트웨어 개발 비용이 증가한다. • 처리 비용이 증가한다. • 잠재적 오류가 증가한다.

분산 데이터베이스 설계

분산 데이터베이스 설계는 애플리케이션이나 사용자가 분산되어 저장된 데이터에 접근하게 하는 것을 목적으로 한다.

- 분산 설계 방법에는 테이블 위치 분산, 분할(Fragmentation), 할당(Allocation)이 있다.
  - 테이블 위치 분산 : 데이터베이스의 테이블을 각기 다른 서버에 분산시켜 배치하는 방법
  - 분할 : 테이블의 데이터를 분할하여 분산시키는 것

    ▶ 주요 분할 방법

수평 분할	특정 속성의 값을 기준으로 행(Row) 단위로 분할
수직 분할	데이터 컬럼(속성) 단위로 분할

  - 할당 : 동일한 분할을 여러 개의 서버에 생성하는 분산 방법으로, 중복이 없는 할당(Allocation)과 중복이 있는 할당(Allocation)으로 나뉨

핵심 184 데이터베이스 이중화 / 서버 클러스터링

데이터베이스 이중화(Database Replication)

데이터베이스 이중화는 시스템 오류로 인한 데이터베이스 서비스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리하는 것이다.

- 데이터베이스 이중화를 수행하면 하나 이상의 데이터베이스가 항상 같은 상태를 유지하므로 데이터베이스에 문제가 발생하면 복제된 데이터베이스를 이용하여 즉시 문제를 해결할 수 있다.
- 데이터베이스 이중화의 분류

Eager 기법	트랜잭션 수행 중 데이터 변경이 발생하면 이중화된 모든 데이터베이스에 즉시 전달하여 변경 내용이 즉시 적용되도록 하는 기법
Lazy 기법	트랜잭션의 수행이 종료되면 변경 사실을 새로운 트랜잭션에 작성하여 각 데이터베이스에 전달되는 기법으로, 데이터베이스마다 새로운 트랜잭션이 수행되는 것으로 간주된다.

- 데이터베이스 이중화 구성 방법

활동-대기 (Active-Standby) 방법	• 한 DB가 활성 상태로 서비스하고 있으면 다른 DB는 대기하고 있다가 활성 DB에 장애가 발생하면 대기 상태에 있던 DB가 자동으로 모든 서비스를 대신 수행한다. • 구성 방법과 관리가 쉬워 많은 기업에서 이용된다.
활동-활동 (Active-Active) 방법	• 두 개의 DB가 서로 다른 서비스를 제공하다가 둘 중 한쪽 DB에 문제가 발생하면 나머지 다른 DB가 서비스를 제공한다. • 두 DB가 모두 처리를 하기 때문에 처리율이 높지만 구성 방법 및 설정이 복잡하다.

클러스터링(Clustering)

클러스터링은 두 대 이상의 서버를 하나의 서버처럼 운영하는 기술이다.

- 클러스터링은 서버 이중화 및 공유 스토리지를 사용하여 서버의 고가용성을 제공한다.
- 클러스터링에는 고가용성 클러스터링과 병렬 처리 클러스터링이 있다.
  - 고가용성 클러스터링 : 하나의 서버에 장애가 발생하면 다른 노드(서버)가 받아 처리하여 서비스 중단을 방지하는 방식으로, 일반적으로 언급되는 클러스터링이 고가용성 클러스터링이다.
  - 병렬 처리 클러스터링 : 전체 처리율을 높이기 위해 하나의 작업을 여러 개의 서버에서 분산하여 처리하는 방식이다.

핵심 185 암호화(Encryption)

암호화는 데이터를 보낼 때 송신자가 지정한 수신자 이외에는 그 내용을 알 수 없도록 평문을 암호문으로 변환하는 것이다.

- 암호화(Encryption) 과정 : 암호화되지 않은 평문을 정보 보호를 위해 암호문으로 바꾸는 과정
- 복호화(Decryption) 과정 : 암호문을 원래의 평문으로 바꾸는 과정
- 암호화 기법에는 개인키 암호 방식과 공개키 암호 방식이 있다.

개인키 암호 방식(Private Key Encryption) = 비밀키 암호 방식

비밀키 암호화 기법은 동일한 키로 데이터를 암호화하고 복호화한다.

- 비밀키 암호화 기법은 대칭 암호 방식 또는 단일키 암호화 기법이라고도 한다.
- 비밀키는 제3자에게는 노출시키지 않고 데이터베이스 사용 권한이 있는 사용자만 나누어 가진다.
- 종류 : 전위 기법, 대체 기법, 대수 기법, 합성 기법(DES, LUCIFER)

공개키 암호 방식(Public Key Encryption)

공개키 암호화 기법은 서로 다른 키로 데이터를 암호화하고 복호화한다.

- 데이터를 암호화할 때 사용하는 키(공개키, Public Key)는 데이터베이스 사용자에게 공개하고, 복호화할 때의 키(비밀키, Secret Key)는 관리자가 비밀리에 관리 하는 방법이다.
- 공개키 암호화 기법은 비대칭 암호 방식이라고도 하며, 대표적으로 RSA(Rivest Shamir Adleman)가 있다.

핵심 186 데이터베이스 보안 – 접근통제

접근통제

접근통제는 데이터가 저장된 객체와 이를 사용하려는 주체 사이의 정보 흐름을 제한하는 것이다.

- 접근통제 기술

임의 접근통제 (DAC, Discretionary Access Control)	• 임의 접근통제는 데이터에 접근하는 사용자의 신원에 따라 접근 권한을 부여하는 방식이다. • 데이터 소유자가 접근통제 권한을 지정하고 제어한다. • 객체를 생성한 사용자가 생성된 객체에 대한 모든 권한을 부여받고, 부여된 권한을 다른 사용자에게 허가할 수도 있다. • 임의 접근통제에 사용되는 SQL 명령어에는 GRANT와 REVOKE가 있다.
강제 접근통제 (MAC, Mandatory Access Control)	• 강제 접근통제는 주체와 객체의 등급을 비교하여 접근 권한을 부여하는 방식이다. • 시스템이 접근통제 권한을 지정한다. • 데이터베이스 객체별로 보안 등급을 부여할 수 있고, 사용자별로 인가 등급을 부여할 수 있다. • 주체는 자신보다 보안 등급이 높은 객체에 대해 읽기, 수정, 등록이 모두 불가능하고, 보안 등급이 같은 객체에 대해서는 읽기, 수정, 등록이 가능하고, 보안 등급이 낮은 객체는 읽기가 가능하다.
역할기반 접근통제 (RBAC, Role Based Access Control)	• 역할기반 접근통제는 사용자의 역할에 따라 접근 권한을 부여하는 방식이다. • 중앙관리자가 접근 통제 권한을 지정한다. • 임의 접근통제와 강제 접근통제의 단점을 보완하였으며, 다중 프로그래밍 환경에 최적화된 방식이다. • 중앙관리자가 역할마다 권한을 부여하면, 책임과 자질에 따라 역할을 할당받은 사용자들은 역할에 해당하는 권한을 사용할 수 있다.

- 접근통제의 3요소 : 접근통제 정책, 접근통제 메커니즘, 접근통제 보안모델

접근통제 정책

접근통제 정책은 어떤 주체가(Who)가 언제(When), 어디서(Where), 어떤 객체(What)에게, 어떤 행위(How)에 대한 허용 여부를 정의하는 것으로, 신분 기반 정책, 규칙 기반 정책, 역할 기반 정책이 있다.

신분 기반 정책	• 주체나 그룹의 신분에 근거하여 객체의 접근을 제한하는 방법으로, IBP와 GBP가 있다. • IBP(Individual-Based Policy) : 최소 권한 정책으로, 단일 주체에게 하나의 객체에 대한 허가를 부여한다. • GBP(Group-Based Policy) : 복수 주체에 하나의 객체에 대한 허가를 부여한다.
규칙 기반 정책	• 주체가 갖는 권한에 근거하여 객체의 접근을 제한하는 방법으로, MLP와 CBP가 있다. • MLP(Multi-Level Policy) : 사용자 및 객체별로 지정된 기밀 분류에 따른 정책 • CBP(Compartment-Based Policy) : 집단별로 지정된 기밀 허가에 따른 정책
역할 기반 정책	GBP의 변형된 정책으로, 주체의 신분이 아니라 주체가 맡은 역할에 근거하여 객체의 접근을 제한하는 방법이다. 예 인사 담당자, DBA 등

접근통제 매커니즘

접근통제 매커니즘은 정의된 접근통제 정책을 구현하는 기술적인 방법으로, 접근통제 목록, 능력 리스트, 보안 등급, 패스워드, 암호화 등이 있다.

접근통제 보안 모델

접근통제 보안 모델은 보안 정책을 구현하기 위한 정형화된 모델로, 기밀성 모델, 무결성 모델, 접근통제 모델이 있다.

기밀성 모델	• 군사적인 목적으로 개발된 최초의 수학적 모델로, 기밀성 보장이 최우선인 모델이다. • 기밀성 모델은 군대 시스템 등 특수 환경에서 주로 사용된다.

무결성 모델	• 기밀성 모델에서 발생하는 불법적인 정보 변경을 방지하기 위해 무결성을 기반으로 개발된 모델이다. • 무결성 모델은 데이터의 일관성 유지에 중점을 두어 개발되었다.
접근통제 모델	접근통제 메커니즘을 보안 모델로 발전시킨 것으로, 대표적으로 접근통제 행렬(Access Control Matrix)이 있다.

핵심 187 데이터베이스 백업

데이터베이스 백업

데이터베이스 백업은 전산 장비의 장애에 대비하여 데이터베이스에 저장된 데이터를 보호하고 복구하기 위한 작업으로, 치명적인 데이터 손실을 막기 위해서는 데이터베이스를 정기적으로 백업해야 한다.

- 데이터베이스 관리 시스템(DBMS)은 데이터베이스 파괴 및 실행 중단이 발생하면 이를 복구할 수 있는 기능을 제공한다.

로그 파일

로그 파일은 데이터베이스의 처리 내용이나 이용 상황 등 상태 변화를 시간의 흐름에 따라 모두 기록한 파일로, 데이터베이스의 복구를 위해 필요한 가장 기본적인 자료이다.

- 로그 파일을 기반으로 데이터베이스를 과거 상태로 복귀(UNDO)시키거나 현재 상태로 재생(REDO)시켜 데이터베이스 상태를 일관성 있게 유지할 수 있다.
- 로그 파일은 트랜잭션 시작 시점, Rollback 시점, 데이터 입력, 수정 삭제 시점 등에서 기록된다.

데이터베이스 복구 알고리즘

NO-UNDO/ REDO	• 데이터베이스 버퍼의 내용을 비동기적으로 갱신한 경우의 복구 알고리즘 • NO-UNDO : 트랜잭션 완료 전에는 변경 내용이 데이터베이스에 기록되지 않으므로 취소할 필요가 없다. • REDO : 트랜잭션 완료 후 데이터베이스 버퍼에는 기록되어 있고, 저장매체에는 기록되지 않았으므로 트랜잭션 내용을 다시 실행해야 한다.
UNDO/ NO-REDO	• 데이터베이스 버퍼의 내용을 동기적으로 갱신한 경우의 복구 알고리즘 • UNDO : 트랜잭션 완료 전에 시스템이 파손되었다면 변경된 내용을 취소한다. • NO-REDO : 트랜잭션 완료 전에 데이터베이스 버퍼 내용을 이미 저장 매체에 기록했으므로 트랜잭션 내용을 다시 실행할 필요가 없다.
UNDO/ REDO	• 데이터베이스 버퍼의 내용을 동기/비동기적으로 갱신한 경우의 복구 알고리즘 • 데이터베이스 기록 전에 트랜잭션이 완료될 수 있으므로 완료된 트랜잭션이 데이터베이스에 기록되지 못했다면 다시 실행해야 한다.
NO-UNDO/ NO-REDO	• 데이터베이스 버퍼의 내용을 동기적으로 저장 매체에 기록하지만 데이터베이스와는 다른 영역에 기록한 경우의 복구 알고리즘 • NO-UNDO : 변경 내용은 데이터베이스와 다른 영역에 기록되어 있으므로 취소할 필요가 없다. • NO-REDO : 다른 영역에 이미 기록되어 있으므로 트랜잭션을 다시 실행할 필요가 없다.

백업 종류

백업 종류는 복구 수준에 따라서 운영체제를 이용하는 물리 백업과 DBMS 유틸리티를 이용하는 논리 백업으로 나뉜다.

- **물리 백업** : 데이터베이스 파일을 백업하는 방법으로, 백업 속도가 빠르고 작업이 단순하지만 문제 발생 시 원인 파악 및 문제 해결이 어렵다.
- **논리 백업** : DB 내의 논리적 객체들을 백업하는 방법으로, 복원 시 데이터 손상을 막고 문제 발생 시 원인 파악 및 해결이 수월하지만 백업/복원 시 시간이 많이 소요된다.

구분	설명	복구 수준
물리 백업	로그 파일 백업 실시	완전 복구
	로그 파일 백업 없음	백업 시점까지 복구
논리 백업	DBMS 유틸리티	

핵심 188 스토리지

스토리지(Storage)의 개요

스토리지는 단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술이다.

DAS(Direct Attached Storage)

DAS는 서버와 저장장치를 전용 케이블로 직접 연결하는 방식으로, 일반 가정에서 컴퓨터에 외장하드를 연결하는 것이 여기에 해당된다.

- 서버에서 저장장치를 관리한다.
- 저장장치를 직접 연결하므로 속도가 빠르고 설치 및 운영이 쉽다.

- 초기 구축 비용 및 유지보수 비용이 저렴하다.
- 확장성 및 유연성이 상대적으로 떨어진다.

NAS(Network Attached Storage)

- NAS는 서버와 저장장치를 네트워크를 통해 연결하는 방식이다.
- 별도의 파일 관리 기능이 있는 NAS Storage가 내장된 저장장치를 직접 관리한다.
- Ethernet 스위치를 통해 다른 서버에서도 스토리지에 접근할 수 있어 파일 공유가 가능하고, 장소에 구애받지 않고 저장장치에 쉽게 접근할 수 있다.
- DAS에 비해 확장성 및 유연성이 우수하다.
- 접속 증가 시 성능이 저하될 수 있다.

SAN(Storage Area Network)

SAN은 DAS의 빠른 처리와 NAS의 파일 공유 장점을 혼합한 방식으로, 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성하는 방식이다.

- 파이버 채널 스위치는 서버나 저장장치를 광케이블로 연결하므로 처리 속도가 빠르다.
- 서버들이 저장장치 및 파일을 공유할 수 있다.
- 확장성, 유연성, 가용성이 뛰어나다.

핵심 189 논리 데이터 모델의 물리 데이터 모델 변환

엔티티(Entity)를 테이블로 변환

논리 데이터 모델에서 정의된 엔티티를 물리 데이터 모델의 테이블로 변환하는 것이다.

- 변환 규칙

논리적 설계(데이터 모델링)	물리적 설계
엔티티(Entity)	테이블(Table)
속성(Attribute)	컬럼(Column)
주 식별자(Primary Identifier)	기본키(Primary Key)
외부 식별자(Foreign Identifier)	외래키(Foreign Key)
관계(Relationship)	관계(Relationship)

슈퍼타입/서브타입을 테이블로 변환

슈퍼타입/서브타입은 논리 데이터 모델에서 이용되는 형태이므로 물리 데이터 모델을 설계할 때는 슈퍼타입/서브타입을 테이블로 변환해야 한다.

- 슈퍼타입 기준 테이블 변환 : 서브타입을 슈퍼타입에 통합하여 하나의 테이블로 만드는 것
  - 장 · 단점

장점	• 데이터의 액세스가 상대적으로 용이하다. • 뷰를 이용하여 각각의 서브타입만을 액세스하거나 수정할 수 있다. • 서브타입 구분이 없는 임의 집합에 대한 처리가 용이하다. • 여러 테이블을 조인하지 않아도 되므로 수행 속도가 빨라진다. • SQL 문장 구성이 단순해진다.
단점	• 테이블의 컬럼이 증가하므로 디스크 저장 공간이 증가한다. • 처리마다 서브타입에 대한 구분(TYPE)이 필요한 경우가 많이 발생한다. • 인덱스 크기의 증가로 인덱스의 효율이 떨어진다.

- 서브타입 기준 테이블 변환 : 슈퍼타입 속성들을 각각의 서브타입에 추가하여 서브타입들을 개별적인 테이블로 만드는 것
  - 장 · 단점

장점	• 각 서브타입 속성들의 선택 사양이 명확한 경우에 유리하다. • 처리할 때마다 서브타입 유형을 구분할 필요가 없다. • 여러 개의 테이블로 통합하므로 테이블당 크기가 감소하여 전체 테이블 스캔시 유리하다.
단점	• 수행 속도가 감소할 수 있다. • 복잡한 처리를 하는 SQL의 통합이 어렵다. • 부분 범위에 대한 처리가 곤란해진다. • 여러 테이블을 통합한 뷰는 조회만 가능하다. • UID(Unique Identifier, 식별자)의 유지 관리가 어렵다.

- 개별타입 기준 테이블 변환 : 슈퍼타입과 서브타입들을 각각의 개별적인 테이블로 변환하는 것이다.
  - 슈퍼타입과 서브타입 테이블들 사이에는 각각 1:1 관계가 형성된다.
  - 장 · 단점

장점	• 저장 공간이 상대적으로 작다. • 슈퍼타입 또는 서브타입 각각의 테이블에 속한 정보만 조회하는 경우 문장 작성이 용이하다.
단점	슈퍼타입 또는 서브타입의 정보를 같이 처리하면 항상 조인이 발생하여 성능이 저하된다.

속성을 컬럼으로 변환

논리 데이터 모델에서 정의한 속성을 물리 데이터 모델의 컬럼으로 변환한다.

- Primary UID를 기본키로 변환 : 논리 데이터 모델에서의 Primary UID는 물리 데이터 모델의 기본키로 만든다.
- Primary UID(관계의 UID Bar)를 기본키로 변환 : 다른 엔티티와의 관계로 인해 생성된 Primary UID는 물리 데이터 모델의 기본키로 만든다.
- Secondary(Alternate) UID를 유니크키로 변환 : 논리 모델링에서 정의된 Secondary UID 및 Alternate Key는 물리 모델에서 유니크키로 만든다.

관계를 외래키로 변환

논리 데이터 모델에서 정의된 관계는 기본키와 이를 참조하는 외래키로 변환한다.

- 다음은 개체 A, B로 이루어진 E-R 모델을 관계 테이블로 변환하는 방법이다.
  - 1:1 관계 : 개체 A의 기본키를 개체 B의 외래키로 추가하거나 개체 B의 기본키를 개체 A의 외래키로 추가하여 표현한다.
  - 1:M 관계 : 개체 A의 기본키를 개체 B의 외래키로 추가하여 표현하거나 별도의 테이블로 표현한다.
  - N:M 관계 : 릴레이션 A와 B의 기본키를 모두 포함한 별도의 릴레이션으로 표현한다. 이때 생성된 별도의 릴레이션을 교차 릴레이션(Intersection Relation) 또는 교차 엔티티(Intersection Entity)라고 한다.
  - 1:M 순환 관계 : 개체 A에 개체 A의 기본키를 참조하는 외래키 컬럼을 추가하여 표현한다. 데이터의 계층 구조를 표현하기 위해 주로 사용된다.

핵심 190 물리 데이터 모델 품질 검토

물리 데이터 모델 품질 검토

- 물리 데이터 모델의 품질 검토는 물리 데이터 모델을 설계하고 데이터베이스 객체를 생성한 후 개발 단계로 넘어가기 전에 모델러와 이해관계자들이 모여 수행한다.
- 물리 데이터 모델은 시스템 성능에 직접적인 영향을 미치므로 향후 발생할 문제에 대해 면밀히 검토해야 한다.
- 물리 데이터 모델 품질 검토의 목적은 데이터베이스의 성능 향상과 오류 예방이다.
- 물리 데이터 모델을 검토하려면 모든 이해관계자가 동의하는 검토 기준이 필요하다.

물리 데이터 모델 품질 기준

정확성	데이터 모델이 요구사항이나 업무 규칙, 표기법에 따라 정확하게 표현되었음을 의미한다.
완전성	데이터 모델이 데이터 모델의 구성 요소를 누락 없이 정의하고 요구사항이나 업무 영역을 누락 없이 반영하였음을 의미한다.
준거성	데이터 모델이 데이터 표준, 표준화 규칙, 법적 요건 등을 정확하게 준수하였음을 의미한다.
최신성	데이터 모델이 최근의 이슈나 현행 시스템을 반영하고 있음을 의미한다.
일관성	데이터 모델이 표현상의 일관성을 유지하고 있음을 의미한다.
활용성	작성된 모델과 설명을 사용자가 충분히 이해할 수 있고, 업무 변화에 따른 데이터 구조의 변경이 최소화될 수 있도록 설계되었음을 의미한다.

- 물리 데이터 모델의 품질 기준은 조직 혹은 업무 상황에 따라 가감하거나 변형하여 사용한다.

핵심 191 DDL(Data Define Language, 데이터 정의어)

- DDL은 SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
- 논리적 데이터 구조와 물리적 데이터 구조의 사상을 정의한다.
- 데이터베이스 관리자나 데이터베이스 설계자가 사용한다.
- DDL(데이터 정의어)의 세 가지 유형

명령어	기능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의한다.
ALTER	TABLE에 대한 정의를 변경하는 데 사용한다.
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제한다.

핵심 192 DML(Data Manipulation Language, 데이터 조작어)

- DML은 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다.
- 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
- DML(데이터 조작어)의 네 가지 유형

명령어	기능
SELECT	테이블에서 조건에 맞는 튜플을 검색한다.
INSERT	테이블에 새로운 튜플을 삽입한다.
DELETE	테이블에서 조건에 맞는 튜플을 삭제한다.
UPDATE	테이블에서 조건에 맞는 튜플의 내용을 변경한다.

핵심 193 DCL(Data Control Language, 데이터 제어어)

- DCL은 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다.
- 데이터베이스 관리자가 데이터 관리를 목적으로 사용한다.
- DCL(데이터 제어어)의 종류

명령어	기능
COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려준다.
ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구한다.
GRANT	데이터베이스 사용자에게 사용 권한을 부여한다.
REVOKE	데이터베이스 사용자의 사용 권한을 취소한다.

핵심 194 CREATE TABLE

CREATE TABLE은 테이블을 정의하는 명령문이다.

표기 형식

```
CREATE TABLE 테이블명
        (속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], …
        [, PRIMARY KEY(기본키_속성명, …)]
        [, UNIQUE(대체키_속성명, …)]
        [, FOREIGN KEY(외래키_속성명, …)]
                [REFERENCES 참조테이블(기본키_속성명, …)]
                [ON DELETE 옵션]
                [ON UPDATE 옵션]
        [, CONSTRAINT 제약조건명] [CHECK (조건식)]);
```

- 기본 테이블에 포함될 모든 속성에 대하여 속성명과 그 속성의 데이터 타입, 기본값, NOT NULL 여부를 지정한다.
- PRIMARY KEY : 기본키로 사용할 속성 또는 속성의 집합을 지정한다.
- UNIQUE : 대체키로 사용할 속성 또는 속성의 집합을 지정하는 것으로 UNIQUE로 지정한 속성은 중복된 값을 가질 수 없다.
- FOREIGN KEY ~ REFERENCES ~
  - 참조할 다른 테이블과 그 테이블을 참조할 때 사용할 외래키 속성을 지정한다.
  - 외래키가 지정되면 참조 무결성의 CASCADE 법칙이 적용된다.
  - ON DELETE 옵션 : 참조 테이블의 튜플이 삭제되었을 때 기본 테이블에 취해야 할 사항을 지정한다. 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT가 있다.
  - ON UPDATE 옵션 : 참조 테이블의 참조 속성 값이 변경되었을 때 기본 테이블에 취해야 할 사항을 지정한다. 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT가 있다.

- CONSTRAINT : 제약 조건의 이름을 지정한다. 이름을 지정할 필요가 없으면 CHECK절만 사용하여 속성 값에 대한 제약 조건을 명시한다.
- CHECK : 속성 값에 대한 제약 조건을 정의한다.

핵심 195 ALTER TABLE

ALTER TABLE은 테이블에 대한 정의를 변경하는 명령문이다.

표기 형식

```
ALTER TABLE 테이블명 ADD 속성명 데이터_타입 [DEFAULT '기본값';
ALTER TABLE 테이블명 ALTER 속성명 [SET DEFAULT '기본값';
ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];
```

- ADD : 새로운 속성(열)을 추가할 때 사용한다.
- ALTER : 특정 속성의 Default 값을 변경할 때 사용한다.
- DROP COLUMN : 특정 속성을 삭제할 때 사용한다.

예제 1 〈학생〉 테이블에 최대 3문자로 구성되는 '학년' 속성 추가하시오.

```
ALTER TABLE 학생 ADD 학년 VARCHAR(3);
```

예제 2 〈학생〉 테이블의 '학번' 필드의 데이터 타입과 크기를 VARCHAR(10)으로 하고 NULL 값이 입력되지 않도록 변경하시오.

```
ALTER TABLE 학생 ALTER 학번 VARCHAR(10) NOT NULL;
```

핵심 196 DROP

DROP은 스키마, 도메인, 기본 테이블, 뷰 테이블, 인덱스, 제약 조건 등을 제거하는 명령문이다.

표기 형식

```
DROP SCHEMA 스키마명 [CASCADE | RESTRICT];
DROP DOMAIN 도메인명 [CASCADE | RESTRICT];
DROP TABLE 테이블명 [CASCADE | RESTRICT];
DROP VIEW 뷰명 [CASCADE | RESTRICT];
DROP INDEX 인덱스명 [CASCADE | RESTRICT];
DROP CONSTRAINT 제약조건명;
```

- CASCADE : 제거할 요소를 참조하는 다른 모든 개체를 함께 제거한다. 즉 주 테이블의 데이터 제거 시 각 외래키와 관계를 맺고 있는 모든 데이터를 제거하는 참조 무결성 제약 조건을 설정하기 위해 사용된다.
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소한다.

예제 〈학생〉 테이블을 제거하되, 〈학생〉 테이블을 참조하는 모든 데이터를 함께 제거하시오.

```
DROP TABLE 학생 CASCADE;
```

핵심 197 DCL(Data Control Language, 데이터 제어어)의 개념

DCL(데이터 제어어)는 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어이다.

- DCL은 데이터베이스 관리자(DBA)가 데이터 관리를 목적으로 사용한다.
- DCL에는 GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT 등이 있다.

핵심 198 GRANT / REVOKE

데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.

- GRANT : 권한 부여를 위한 명령어
- REVOKE : 권한 취소를 위한 명령어
- 사용자등급 지정 및 해제

> – **GRANT** 사용자등급 **TO** 사용자_ID_리스트 [IDENTIFIED BY 암호];
> – **REVOKE** 사용자등급 **FROM** 사용자_ID_리스트;

예제 1 사용자 ID가 "NABI"인 사람에게 데이터베이스 및 테이블을 생성할 수 있는 권한을 부여하는 SQL문을 작성하시오.

```
GRANT RESOURCE TO NABI;
```

예제 2 사용자 ID가 "STAR"인 사람에게 단순히 데이터베이스에 있는 정보를 검색할 수 있는 권한을 부여하는 SQL문을 작성하시오.

```
GRANT CONNECT TO STAR;
```

- 테이블 및 속성에 대한 권한 부여 및 취소

> – **GRANT** 권한_리스트 **ON** 개체 **TO** 사용자 [WITH GRANT OPTION];
> – **REVOKE** [GRANT OPTION FOR] 권한_리스트 **ON** 개체 **FROM** 사용자 [CASCADE];

– 권한 종류 : ALL, SELECT, INSERT, DELETE, UPDATE, ALTER 등
– WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여함
– GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소함
– CASCADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소함

예제 3 사용자 ID가 "NABI"인 사람에게 〈고객〉 테이블에 대한 모든 권한과 다른 사람에게 권한을 부여할 수 있는 권한까지 부여하는 SQL문을 작성하시오.

```
GRANT ALL ON 고객 TO NABI WITH GRANT OPTION;
```

예제 4 사용자 ID가 "STAR"인 사람에게 부여한 〈고객〉 테이블에 대한 권한 중 UPDATE 권한을 다른 사람에게 부여할 수 있는 권한만 취소하는 SQL문을 작성하시오.

```
REVOKE GRANT OPTION FOR UPDATE ON 고객 FROM STAR;
```

핵심 199 COMMIT

트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성(Consistency) 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영하여야 하는데, 이때 사용하는 명령이 COMMIT이다.

- COMMIT 명령을 실행하지 않아도 DML문이 성공적으로 완료되면 자동으로 COMMIT되고, DML이 실패하면 자동으로 ROLLBACK이 되도록 Auto Commit 기능을 설정할 수 있다.

핵심 200 ROLLBACK

ROLLBACK은 아직 COMMIT되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어이다.

- 트랜잭션 전체가 성공적으로 끝나지 못하면 일부 변경된 내용만 데이터베이스에 반영되는 비일관성(Inconsistency)인 상태를 가질 수 있기 때문에 일부분만 완료된 트랜잭션은 롤백(Rollback)되어야 한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 201 삽입문(INSERT INTO~)

삽입문은 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.

일반 형식

```
INSERT INTO 테이블명([속성명1, 속성명2,…])
VALUES (데이터1, 데이터2,… );
```

- 대응하는 속성과 데이터는 개수와 데이터 유형이 일치해야 한다.
- 기본 테이블의 모든 속성을 사용할 때는 속성명을 생략할 수 있다.
- SELECT문을 사용하여 다른 테이블의 검색 결과를 삽입할 수 있다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 1 〈사원〉 테이블에 (이름 – 홍승현, 부서 – 인터넷)을 삽입하시오.

```
INSERT INTO 사원(이름, 부서) VALUES ('홍승현', '인터넷');
```

예제 2 〈사원〉 테이블에 (장보고, 기획, 05/03/73, 홍제동, 90)을 삽입하시오.

```
INSERT INTO 사원 VALUES ('장보고', '기획', #05/03/73#, '홍제동', 90);
```

예제 3 〈사원〉 테이블에 있는 편집부의 모든 튜플을 편집부원(이름, 생일, 주소, 기본급) 테이블에 삽입하시오.

```
INSERT INTO 편집부원(이름, 생일, 주소, 기본급)
SELECT 이름, 생일, 주소, 기본급
FROM 사원
WHERE 부서 = '편집';
```

핵심 202 삭제문(DELETE FROM~)

삭제문은 기본 테이블에 있는 튜플들 중에서 특정 튜플(행)을 삭제할 때 사용한다.

일반 형식

```
DELETE
FROM 테이블명
[WHERE 조건];
```

- 모든 레코드를 삭제할 때는 WHERE절을 생략한다.
- 모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 1 〈사원〉 테이블에서 "임꺽정"에 대한 튜플을 삭제하시오.

```
DELETE
FROM 사원
WHERE 이름 = '임꺽정';
```

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 203 갱신문(UPDATE~ SET~)

갱신문은 기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용한다.

일반 형식

> UPDATE 테이블명
> SET 속성명 = 데이터[, 속성명=데이터, …]
> [WHERE 조건];

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 〈사원〉 테이블에서 "황진이"의 '부서'를 "기획부"로 변경하고 '기본급'을 5만 원 인상시키시오.

> UPDATE 사원
> SET 부서 = '기획', 기본급 = 기본급 + 5
> WHERE 이름 = '황진이';

핵심 204 데이터 조작문의 네 가지 유형

- SELECT(검색) : SELECT~ FROM~ WHERE~
- INSERT(삽입) : INSERT INTO~ VALUES~
- DELETE(삭제) : DELETE~ FROM~ WHERE~
- UPDATE(변경) : UPDATE~ SET~ WHERE~

핵심 205 일반 형식

> SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속성명, …]
> [, 그룹함수(속성명) [AS 별칭]]
> [, Window함수 OVER (PARTITION BY 속성명1, 속성명2, …
> ORDER BY 속성명3, 속성명4, …)]
> FROM 테이블명[, 테이블명, …]
> [WHERE 조건]
> [GROUP BY 속성명, 속성명, …]
> [HAVING 조건]
> [ORDER BY 속성명 [ASC | DESC]];

- SELECT절
  - PREDICATE : 불러올 튜플 수를 제한할 명령어를 기술한다.
    - ▶ ALL : 모든 튜플을 검색할 때 지정하는 것으로, 주로 생략한다.
    - ▶ DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색한다.
    - ▶ DISTINCTROW : 중복된 튜플을 제거하고 한 개만 검색하지만 선택된 속성의 값이 아닌, 튜플 전체를 대상으로 한다.
  - 속성명 : 검색하여 불러올 속성(열) 또는 속성을 이용한 수식을 지정한다.
    - ▶ 기본 테이블을 구성하는 모든 속성을 지정할 때는 '*'를 기술한다.
    - ▶ 두 개 이상의 테이블을 대상으로 검색할 때는 '테이블명.속성명'으로 표현한다.
  - AS : 속성 및 연산의 이름을 다른 제목으로 표시하기 위해 사용된다.
- FROM절 : 질의에 의해 검색될 데이터들을 포함하는 테이블명을 기술한다.
- WHERE절 : 검색할 조건을 기술한다.
- ORDER BY절 : 특정 속성을 기준으로 정렬하여 검색할 때 사용한다.

– 속성명 : 정렬의 기준이 되는 속성명을 기술한다.

– [ASC | DESC] : 정렬 방식으로서 'ASC'는 오름차순, 'DESC'는 내림차순이다. 생략하면 오름차순으로 지정된다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 1 〈사원〉 테이블에서 '주소'만 검색하되 같은 '주소'는 한 번만 출력하시오.

```
SELECT DISTINCT 주소
FROM 사원;
```

〈결과〉

주소
대흥동
망원동
상암동
서교동
연남동
합정동

예제 2 〈사원〉 테이블에서 "기획" 부서에 근무하면서 "대흥동"에 사는 사람의 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 부서 = '기획' AND 주소 = '대흥동';
```

〈결과〉

이름	부서	생일	주소	기본급
성춘향	기획	02/20/64	대흥동	100

핵심 206 조건 연산자 / 연산자 우선순위

조건 연산자

- 비교 연산자

연산자	=	〈〉	〉	〈	〉=	〈=
의미	같다	같지 않다	크다	작다	크거나 같다	작거나 같다

- 논리 연산자 : NOT, AND, OR
- LIKE 연산자 : 대표 문자를 이용해 지정된 속성의 값이 문자 패턴과 일치하는 튜플을 검색하기 위해 사용된다.

대표 문자	%	_	#
의미	모든 문자를 대표함	문자 하나를 대표함	숫자 하나를 대표함

연산자 우선순위

종류	연산자	우선순위
산술 연산자	×, /, +, –	왼쪽에서 오른쪽으로 갈수록 낮아진다.
관계 연산자	=, 〈〉, 〉, 〉=, 〈, 〈=	모두 같다.
논리 연산자	NOT, AND, OR	왼쪽에서 오른쪽으로 갈수록 낮아진다.

※ 산술, 관계, 논리 연산자가 함께 사용되었을 때는 산술 〉 관계 〉 논리 연산자 순서로 연산자 우선순위가 정해진다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 〈사원〉 테이블에서 성이 '김'인 사람의 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 이름 LIKE "김%";
```

〈결과〉

이름	부서	생일	주소	기본급
김선달	편집	10/22/73	망원동	90

핵심 207 하위 질의

하위 질의는 조건절에 주어진 질의를 먼저 수행하여 그 검색 결과를 조건절의 피연산자로 사용한다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 1 '취미'가 "나이트댄스"인 사원의 '이름'과 '주소'를 검색하시오.

```
SELECT 이름, 주소
FROM 사원
WHERE 이름 = (SELECT 이름 FROM 여가활동
WHERE 취미 = '나이트댄스') ;
```

〈결과〉

이름	주소
성춘향	대흥동

시험에 나오는 것만 공부한다! 시나공시리즈

예제 2 취미활동을 하는 사원들의 부서를 검색하시오.

```
SELECT 부서
FROM 사원
WHERE EXISTS (SELECT 이름 FROM 여가활동
WHERE 여가활동.이름 = 사원.이름);
```

〈결과〉

부서
인터넷
편집
기획
기획

핵심 208 일반 형식

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭]
[, [테이블명.]속성명, …]
[, 그룹함수(속성명) [AS 별칭]]
[, WINDOW함수 OVER (PARTITION BY 속성명1,
속성명2, …
                ORDER BY 속성명3, 속성명4, …)
[AS 별칭]]
FROM 테이블명[, 테이블명, …]
[WHERE 조건]
[GROUP BY 속성명, 속성명, …]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- 그룹함수 : GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 함수를 기술한다.
- WINDOW 함수 : GROUP BY절을 이용하지 않고 속성의 값을 집계할 함수를 기술한다.
  - PARTITION BY : WINDOW 함수가 적용될 범위로 사용할 속성을 지정한다.
  - ORDER BY : PARTITION 안에서 정렬 기준으로 사용할 속성을 지정한다.
- GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 사용한다. 일반적으로 GROUP BY절은 그룹 함수와 함께 사용된다.
- HAVING절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정한다.

〈상여금〉

부서	이름	상여내역	상여금
기획	홍길동	연장근무	100
기획	일지매	연장근무	100
기획	최준호	야간근무	120
기획	장길산	특별근무	90
인터넷	강건달	특별근무	90
인터넷	서국현	특별근무	90
인터넷	박인식	연장근무	30
편집	김선달	특별근무	80
편집	황종근	연장근무	40
편집	성춘향	야간근무	80
편집	임꺽정	야간근무	80
편집	황진이	야간근무	50

예제 〈상여금〉 테이블에서 '상여금'이 100 이상인 사원이 2명 이상인 '부서'의 튜플 수를 구하시오.

```
SELECT 부서, COUNT(*) AS 사원수
FROM 상여금
WHERE 상여금 >= 100
GROUP BY 부서
HAVING COUNT(*) >= 2;
```

〈결과〉

부서	사원수
기획	3

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 209 그룹 함수

GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 때 사용된다.

- COUNT(속성명) : 그룹별 튜플 수를 구하는 함수
- SUM(속성명) : 그룹별 합계를 구하는 함수
- AVG(속성명) : 그룹별 평균을 구하는 함수
- MAX(속성명) : 그룹별 최대값을 구하는 함수
- MIN(속성명) : 그룹별 최소값을 구하는 함수
- STDDEV(속성명) : 그룹별 표준편차를 구하는 함수
- VARIANCE(속성명) : 그룹별 분산을 구하는 함수

핵심 210 집합 연산자를 이용한 통합 질의

집합 연산자를 사용하여 2개 이상의 테이블의 데이터를 하나로 통합한다.

표기 형식

```
SELECT 속성명1, 속성명2, …
FROM 테이블명
UNION | UNION ALL | INTERSECT | EXCEPT
SELECT 속성명1, 속성명2, …
FROM 테이블명
[ORDER BY 속성명 [ASC | DESC]];
```

- 두 개의 SELECT문에 기술한 속성들은 개수와 데이터 유형이 서로 동일해야 한다.

〈사원〉

사원	직급
김형석	대리
홍영선	과장
류기선	부장
김현천	이사

〈직원〉

사원	직급
신원섭	이사
이성호	대리
홍영선	과장
류기선	부장

예제 〈사원〉 테이블과 〈직원〉 테이블을 통합하는 질의문을 작성하시오. (단, 같은 레코드가 중복되어 나오지 않게 하시오.)

```
SELECT *
FROM 사원
UNION
SELECT *
FROM 직원;
```

〈결과〉

사원	직급
김현천	이사
김형석	대리
류기선	부장
신원섭	이사
이성호	대리
홍영선	과장

핵심 211 JOIN의 개념

JOIN(조인)은 2개의 테이블에 대해 연관된 튜플들을 결합하여, 하나의 새로운 릴레이션을 반환한다.

- JOIN은 크게 INNER JOIN과 OUTER JOIN으로 구분된다.
- JOIN은 일반적으로 FROM절에 기술하지만, 릴레이션이 사용되는 어느 곳에서나 사용할 수 있다.

핵심 212 INNER JOIN

INNER JOIN은 일반적으로 EQUI JOIN과 NON-EQUI JOIN으로 구분된다.

- 조건이 없는 INNER JOIN을 수행하면 CROSS JOIN과 동일한 결과를 얻을 수 있다.

- EQUI JOIN
  - EQUI JOIN은 JOIN 대상 테이블에서 공통 속성을 기준으로 '='(equal) 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 JOIN 방법이다.
  - WHERE절을 이용한 EQUI JOIN의 표기 형식

> **SELECT** [테이블명1.]속성명, [테이블명2.]속성명, …
> **FROM** 테이블명1, 테이블명2, …
> **WHERE** 테이블명1.속성명 = 테이블명2.속성명;

  - NATURAL JOIN절을 이용한 EQUI JOIN의 표기 형식

> **SELECT** [테이블명1.]속성명, [테이블명2.]속성명, …
> **FROM** 테이블명1 **NATURAL JOIN** 테이블명2;

  - JOIN ~ USING절을 이용한 EQUI JOIN의 표기 형식

> **SELECT** [테이블명1.]속성명, [테이블명2.]속성명, …
> **FROM** 테이블명1 **JOIN** 테이블명2 **USING**(속성명);

〈학생〉

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

〈학과〉

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

〈성적등급〉

등급	최저	최고
A	90	100
B	80	89
C	60	79
D	0	59

예제 1 〈학생〉 테이블과 〈학과〉 테이블에서 '학과코드' 값이 같은 튜플을 JOIN하여 '학번', '이름', '학과코드', '학과명'을 출력하는 SQL문을 작성하시오.

- SELECT 학번, 이름, 학생.학과코드, 학과명
  FROM 학생, 학과
  WHERE 학생.학과코드 = 학과.학과코드;
- SELECT 학번, 이름, 학생.학과코드, 학과명
  FROM 학생 NATURAL JOIN 학과;
- SELECT 학번, 이름, 학생.학과코드, 학과명
  FROM 학생 JOIN 학과 USING(학과코드);

〈결과〉

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어

핵심 213 SELF JOIN

- SELF JOIN은 같은 테이블에서 2개의 속성을 연결하여 EQUI JOIN을 하는 JOIN 방법이다.
- 표기 형식

> - **SELECT** [별칭1.]속성명, [별칭1.]속성명, …
> **FROM** 테이블명1 [AS] 별칭1 JOIN 테이블명1 [AS] 별칭2
> **ON** 별칭1.속성명 = 별칭2.속성명;
> - **SELECT** [별칭1.]속성명, [별칭1.]속성명, …
> **FROM** 테이블명1 [AS] 별칭1, 테이블명1 [AS] 별칭2
> **WHERE** 별칭1.속성명 = 별칭2.속성명;

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 214 프로시저(Procedure)의 개요

프로시저란 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다.

- 프로시저를 만들어 데이터베이스에 저장하면 여러 프로그램에서 호출하여 사용할 수 있다.
- 프로시저는 데이터베이스에 저장되어 수행되기 때문에 스토어드(Stored) 프로시저라고도 불린다.
- 프로시저는 시스템의 일일 마감 작업, 일괄(Batch) 작업 등에 주로 사용된다.

프로시저의 구성도

데이터 ➡

프로시저
DECLARE (필수)
BEGIN (필수)
• CONTROL
• SQL
• EXCEPTION
• TRANSACTION
END (필수)

➡ 결과

- DECLARE : 프로시저의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부이다.
- BEGIN / END : 프로시저의 시작과 종료를 의미한다.
- CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다.
- SQL : DML, DCL이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 수행한다.
- EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의한다.
- TRANSACTION : 수행된 데이터 작업들을 DB에 적용할지 취소할지를 결정하는 처리부이다.

핵심 215 트리거(Trigger)의 개요

트리거는 데이터베이스 시스템에서 데이터의 삽입(Insert), 갱신(Update), 삭제(Delete) 등의 이벤트(Event)가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL이다.

- 트리거는 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력 등의 목적으로 사용된다.
- 트리거의 구문에는 DCL(데이터 제어어)을 사용할 수 없으며, DCL이 포함된 프로시저나 함수를 호출하는 경우에도 오류가 발생한다.
- 트리거에 오류가 있는 경우 트리거가 처리하는 데이터에도 영향을 미치므로 트리거를 생성할 때 세심한 주의가 필요하다.

핵심 216 트리거의 구성

트리거는 선언, 이벤트, 시작, 종료로 구성되며, 시작과 종료 구문 사이에는 제어(CONTROL), SQL, 예외(EXCEPTION)가 포함된다.

트리거 구성도

이벤트 ➡

트리거
DECLARE (필수)
EVENT (필수)
BEGIN (필수)
• CONTROL
• SQL
• EXCEPTION
END (필수)

➡ 데이터 변경

- DECLARE : 트리거의 명칭, 변수 및 상수, 데이터 타입을 정의하는 선언부이다.
- EVENT : 트리거가 실행되는 조건을 명시한다.
- BEGIN / END : 트리거의 시작과 종료를 의미한다.
- CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다.

- SQL : DML문이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 수행한다.
- EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의한다.

핵심 217 사용자 정의 함수

사용자 정의 함수의 개요

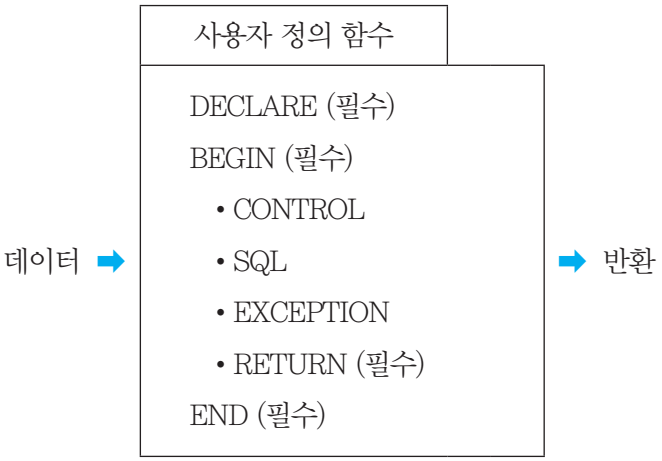
사용자 정의 함수는 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 처리 결과를 단일값으로 반환하는 절차형 SQL이다.

- 사용자 정의 함수는 데이터베이스에 저장되어 SELECT, INSERT, DELETE, UPDATE 등 DML문의 호출에 의해 실행된다.
- 사용자 정의 함수는 예약어 RETURN을 통해 값을 반환하기 때문에 출력 파라미터가 없다.
- 사용자 정의 함수는 INSERT, DELETE, UPDATE를 통한 테이블 조작은 할 수 없고 SELECT를 통한 조회만 할 수 있다.
- 사용자 정의 함수는 프로시저를 호출하여 사용할 수 없다.
- 사용자 정의 함수는 SUM( ), AVG( ) 등의 내장 함수처럼 DML문에서 반환값을 활용하기 위한 용도로 사용된다.

사용자 정의 함수의 구성

사용자 정의 함수의 구성은 프로시저와 유사하다. 프로시저의 구성에서 RETURN만 추가하면 된다.

- 사용자 정의 함수의 구성도

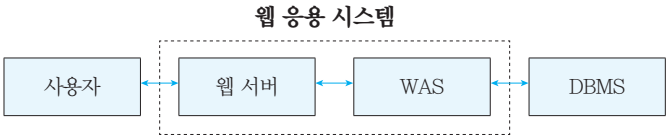


- DECLARE : 사용자 정의 함수의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부이다.
- BEGIN / END : 사용자 정의 함수의 시작과 종료를 의미한다.
- CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다.
- SQL : SELECT문이 삽입되어 데이터 조회 작업을 수행한다.
- EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의한다.
- RETURN : 호출 프로그램에 반환할 값이나 변수를 정의한다.

핵심 218 DBMS 접속의 개요

DBMS 접속은 사용자가 데이터를 사용하기 위해 응용 시스템을 이용하여 DBMS에 접근하는 것을 의미한다.

- 응용 시스템은 사용자로부터 매개 변수를 전달받아 SQL을 실행하고 DBMS로부터 전달받은 결과를 사용자에게 전달하는 매개체 역할을 수행한다.
- 인터넷을 통해 구동되는 웹 응용 프로그램은 웹 응용 시스템을 통해 DBMS에 접근한다.
- 웹 응용 시스템은 웹 서버와 웹 애플리케이션 서버(WAS)로 구성되며, 서비스 규모가 작은 경우 웹 서버와 웹 애플리케이션 서버를 통합하여 하나의 서버만으로 운용할 수 있다.



- 사용자는 웹 서버에 접속하여 데이터를 주고받는다.
- 웹 서버는 많은 수의 서비스 요청을 처리하기 때문에 사용자가 대용량의 데이터를 요청하면 직접 처리하지 않고 WAS에게 해당 요청을 전달한다.
- WAS는 수신한 요청을 트랜잭션 언어로 변환한 후 DBMS에 전달하여 데이터를 받는다. 이렇게 받은 데이터는 처음 요청한 웹 서버로 다시 전달되어 사용자에게까지 도달하게 된다.

핵심 219 DBMS 접속 기술

DBMS 접속 기술은 DBMS에 접근하기 위해 사용하는 API 또는 API의 사용을 편리하게 도와주는 프레임워크 등을 의미한다.

JDBC(Java DataBase Connectivity)

JDBC는 Java 언어로 다양한 종류의 데이터베이스에 접속하고 SQL문을 수행할 때 사용되는 표준 API이다.

- 1997년 2월 썬 마이크로시스템에서 출시했다.
- 접속하려는 DBMS에 대한 드라이버가 필요하다.

ODBC(Open DataBase Connectivity)

ODBC는 데이터베이스에 접근하기 위한 표준 개방형 API로, 개발 언어에 관계없이 사용할 수 있다.

- 1992년 9월 마이크로소프트에서 출시했다.
- ODBC도 접속하려는 DBMS에 맞는 드라이버가 필요하지만, 접속하려는 DBMS의 인터페이스를 알지 못하더라도 ODBC 문장을 사용하여 SQL을 작성하면 ODBC에 포함된 드라이버 관리자가 해당 DBMS의 인터페이스에 맞게 연결해 주므로 DBMS의 종류를 몰라도 된다.

MyBatis

MyBatis는 JDBC 코드를 단순화하여 사용할 수 있는 SQL Mapping 기반 오픈 소스 접속 프레임워크이다.

- JDBC로 데이터베이스에 접속하려면 다양한 메소드를 호출하고 해제해야 하는데, MyBatis는 이를 간소화 했고 접속 기능을 더욱 강화하였다.
- MyBatis는 SQL 문장을 분리하여 XML 파일을 만들고, Mapping을 통해 SQL을 실행한다.
- MyBatis는 SQL을 거의 그대로 사용할 수 있어 SQL 친화적인 국내 환경에 적합하여 많이 사용된다.

핵심 220 동적 SQL(Dynamic SQL)

동적 SQL은 개발 언어에 삽입되는 SQL 코드를 문자열 변수에 넣어 처리하는 것으로, 조건에 따라 SQL 구문을 동적으로 변경하여 처리할 수 있다.

- 동적 SQL은 사용자로부터 SQL문의 일부 또는 전부를 입력받아 실행할 수 있다.
- 동적 SQL은 값이 입력되지 않을 경우 사용하는 NVL 함수를 사용할 필요가 없다.
- 동적 SQL은 응용 프로그램 수행 시 SQL이 변형될 수 있으므로 프리컴파일 할 때 구문 분석, 접근 권한 확인 등을 할 수 없다.
- 동적 SQL은 정적 SQL에 비해 속도가 느리지만, 상황에 따라 다양한 조건을 첨가하는 등 유연한 개발이 가능하다.

핵심 221 SQL 테스트의 개요

SQL 테스트는 SQL이 작성 의도에 맞게 원하는 기능을 수행하는지 검증하는 과정이다.

- 단문 SQL은 SQL 코드를 직접 실행한 후 결과를 확인하는 것으로 간단히 테스트가 가능하다.
- 절차형 SQL은 테스트 전에 생성을 통해 구문 오류(Syntax Error)나 참조 오류의 존재 여부를 확인한다.
- 정상적으로 생성된 절차형 SQL은 디버깅을 통해 로직을 검증하고, 결과를 통해 최종적으로 확인한다.

핵심 222 단문 SQL 테스트

단문 SQL 테스트는 DDL, DML, DCL이 포함되어 있는 SQL과 TCL을 테스트하는 것으로, 직접 실행하여 결과물을 확인한다.

- 실행 시 오류나 경고가 발생할 경우 메시지를 참조하여 문제를 해결한다.
- DESCRIBE 명령어를 이용하면 DDL로 작성된 테이블이나 뷰의 속성, 자료형, 옵션들을 바로 확인할 수 있다.
  - DESC [개체명];
- DML로 변경한 데이터는 SELECT문으로 데이터의 정상적인 변경 여부를 확인할 수 있다.
- DCL로 설정된 사용자 권한은 사용자 권한 정보가 저장된 테이블을 SELECT로 조회하거나, SHOW 명령어로 확인할 수 있다.

핵심 223 절차형 SQL 테스트

프로시저, 사용자 정의 함수, 트리거 등의 절차형 SQL은 디버깅을 통해 기능의 적합성 여부를 검증하고, 실행을 통해 결과를 확인하는 테스트를 수행한다.

- 많은 코드로 구성된 절차형 SQL의 특성상 오류 및 경고 메시지가 상세히 출력되지 않으므로 SHOW 명령어를 통해 오류 내용을 확인하고 문제를 수정한다.
  - 형식 : SHOW ERRORS;
- 데이터베이스에 변화를 줄 수 있는 SQL문은 주석으로 처리하고, 출력문을 이용하여 화면에 출력하여 확인한다.
- 디버깅이 완료되면 출력문을 삭제하고, 주석 기호를 삭제한 후 절차형 SQL을 실행하여 결과를 검토한다.

핵심 224 ORM(Object-Relational Mapping)의 개요

ORM은 객체지향 프로그래밍의 객체(Object)와 관계형 데이터베이스(Relational Database)의 데이터를 연결(Mapping)하는 기술을 의미한다.

- ORM은 객체지향 프로그래밍에서 사용할 수 있는 가상의 객체지향 데이터베이스를 만들어 프로그래밍 코드와 데이터를 연결한다.
- ORM으로 생성된 가상의 객체지향 데이터베이스는 프로그래밍 코드 또는 데이터베이스와 독립적이므로 재사용 및 유지보수가 용이하다.
- ORM은 SQL 코드를 직접 입력하지 않고 선언문이나 할당 같은 부수적인 코드가 생략되기 때문에 직관적이고 간단하게 데이터를 조작할 수 있다.

핵심 225 ORM 프레임워크

ORM 프레임워크는 ORM을 구현하기 위한 구조와 구현을 위해 필요한 여러 기능들을 제공하는 소프트웨어를 의미한다.

- ORM 프레임워크의 종류

JAVA	JPA, Hibernate, EclipseLink, DataNucleus, Ebean 등
C++	ODB, QxOrm 등
Python	Django, SQLAlchemy, Storm 등
iOS	DatabaseObjects, Core Data 등
.NET	NHibernate, DatabaseObjects, Dapper 등
PHP	Doctrine, Propel, RedBean 등

핵심 226 쿼리 성능 최적화의 개요

쿼리 성능 최적화는 데이터 입 · 출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화하는 것이다.

- 쿼리 성능을 최적화하기 전에 성능 측정 도구인 APM을 사용하여 최적화 할 쿼리를 선정해야 한다.
- 최적화 할 쿼리에 대해 옵티마이저가 수립한 실행 계획을 검토하고 SQL 코드와 인덱스를 재구성한다.

※ RBO vs CBO

RBO(Rule Based Optimizer)는 규칙 기반 옵티마이저이고, CBO(Cost Based Optimizer)는 비용 기반 옵티마이저로서 다음과 같은 차이점이 있다.

	RBO	CBO
최적화 기준	규칙에 정의된 우선순위	액세스 비용
성능 기준	개발자의 SQL 숙련도	옵티마이저의 예측 성능
특징	실행 계획 예측이 쉬움	성능 통계치 정보 활용, 예측이 복잡함
고려사항	개발자의 규칙 이해도, 규칙의 효율성	비용 산출 공식의 정확성

핵심 227 쿼리 성능 최적화

쿼리 성능 최적화는 실행 계획에 표시된 연산 순서, 조인 방식, 테이블 조회 방법 등을 참고하여 SQL문이 더 빠르고 효율적으로 작동하도록 SQL 코드와 인덱스를 재구성하는 것을 의미한다.

SQL 코드 재구성

- WHERE 절을 추가하여 일부 레코드만 조회하게 함으로써 조회에 들어가는 비용을 줄인다.
- WHERE 절에 연산자가 포함되면 INDEX를 활용하지 못하므로 가능한 한 연산자 사용을 자제한다.
- 서브 쿼리에 특정 데이터가 존재하는지 확인할 때는 IN보다 EXISTS를 활용한다.
- 옵티마이저의 실행 계획이 잘못되었다고 판단되는 경우 힌트를 활용하여 실행 계획의 액세스 경로 및 조인 순서를 변경한다.

인덱스 재구성

- SQL 코드에서 조회되는 속성과 조건들을 고려하여 인덱스를 구성한다.
- 실행 계획을 참고하여 인덱스를 추가하거나 기존 인덱스의 열 순서를 변경한다.
- 인덱스의 추가 및 변경은 해당 테이블을 참조하는 다른 SQL문에도 영향을 줄 수 있으므로 신중히 결정한다.
- 단일 인덱스로 쓰기나 수정 없이 읽기로만 사용되는 테이블의 경우 IOT(Index-Organized Table)로 구성하는 것을 고려한다.
- 불필요한 인덱스를 제거한다.

핵심 228 데이터 전환의 정의

데이터 전환이란 운영 중인 기존 정보 시스템에 축적되어 있는 데이터를 추출(Extraction)하여 새로 개발할 정보 시스템에서 운영 가능하도록 변환(Transformation)한 후, 적재(Loading)하는 일련의 과정을 말한다.

- 데이터 전환을 ETL(Extraction, Transformation, Load), 즉 추출, 변환, 적재 과정이라고 한다.
- 데이터 전환을 데이터 이행(Data Migration) 또는 데이터 이관이라고도 한다.

핵심 229 데이터 검증

데이터 검증이란 원천 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정을 말한다.

- 데이터 전환 검증은 검증 방법과 검증 단계에 따라 분류할 수 있다.

핵심 230 검증 방법에 따른 분류

데이터 검증은 검증 방법에 따라 로그 검증, 기본 항목 검증, 응용 프로그램 검증, 응용 데이터 검증, 값 검증으로 분류할 수 있다.

로그 검증	데이터 전환 과정에서 작성하는 추출, 전환, 적재 로그를 검증한다.
기본 항목 검증	로그 검증 외에 별도로 요청된 검증 항목에 대해 검증한다.
응용 프로그램 검증	응용 프로그램을 통한 데이터 전환의 정합성을 검증한다.
응용 데이터 검증	사전에 정의된 업무 규칙을 기준으로 데이터 전환의 정합성을 검증한다.
값 검증	숫자 항목의 합계 검증, 코드 데이터의 범위 검증, 속성 변경에 따른 값 검증을 수행한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 231 오류 데이터 측정 및 정제의 개요

오류 데이터 측정 및 정제는 고품질의 데이터를 운영 및 관리하기 위해 수행한다.

- 오류 데이터 측정 및 정제는 '데이터 품질 분석 → 오류 데이터 측정 → 오류 데이터 정제' 순으로 진행한다.

핵심 232 오류 데이터 정제

오류 데이터 정제는 오류 관리 목록의 각 항목을 분석하여 원천 데이터를 정제하거나 전환 프로그램을 수정하는 것이다.

오류 데이터 분석

- 오류 관리 목록의 오류 데이터를 분석하여 오류 상태, 심각도, 해결 방안을 확인 및 기재한다.
- 상태

Open	오류가 보고만 되고 분석되지 않은 상태
Assigned	오류의 영향 분석 및 수정을 위해 개발자에게 오류를 전달한 상태
Fixed	개발자가 오류를 수정한 상태
Closed	수정된 오류에 대해 테스트를 다시 했을 때 오류가 발견되지 않은 상태
Deferred	오류 수정을 연기한 상태
Classified	보고된 오류를 관련자들이 확인했을 때 오류가 아니라고 확인된 상태

핵심 233 데이터 정제요청서의 개요

데이터 정제요청서는 원천 데이터의 정제와 전환 프로그램의 수정을 위해 요청사항 및 조치사항 등 데이터 정제와 관련된 전반적인 내용을 문서로 작성한 것이다.

- 오류 관리 목록을 기반으로 데이터 정제 요건 목록을 작성하고 이 목록의 항목별로 데이터 정제요청서를 작성한다.

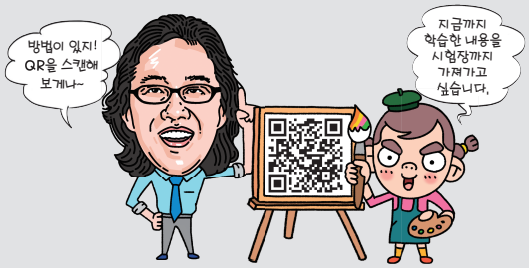
핵심 234 데이터 정제보고서의 개요 및 작성

데이터 정제보고서는 데이터 정제요청서를 통해 정제된 원천 데이터가 정상적으로 정제되었는지 확인한 결과를 문서로 작성한 것이다.

- 정제 요청 데이터와 정제된 데이터 항목을 육안으로 직접 비교하여 확인한다.
- 정제된 데이터를 데이터 전환 프로그램을 이용하여 전환한 후 오류의 발생 여부를 확인하고 목적 데이터베이스에서 전환된 데이터를 확인한다.
- 정제된 데이터의 확인 결과를 반영한 데이터 정제보고서를 정제ID별로 작성한다.
- 데이터 정제보고서에는 데이터 전환 결과 외에도 오류 데이터의 원인, 실제 데이터 정제 건수, 향후 대응 방안 등을 포함한다.

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요? 까먹기 전에 다시 한 번 복습하고 싶다고요? 지금까지 공부한 내용을 안전하게 시험장까지 가져가는 완벽한 방법이 있습니다. 지금 당장 QR 코드를 스캔해 보세요.



www.membox.co.kr을 직접 입력해도 접속할 수 있습니다.

시험에 나오는 것만 공부한다! 시나공시리즈

4 과목 프로그래밍 언어 활용

핵심 235 하드웨어 환경

하드웨어 환경은 사용자와의 인터페이스 역할을 하는 클라이언트(Client) 그리고 클라이언트와 통신하여 서비스를 제공하는 서버(Server)로 구성된다.

- 클라이언트에는 PC, 스마트폰 등이 있다.
- 서버는 사용 목적에 따라 웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버 등으로 나뉜다.
  - 웹 서버(Web Server) : 클라이언트로부터 직접 요청을 받아 처리하는 서버로, 저용량의 정적 파일들을 제공한다.
  - 웹 애플리케이션 서버(WAS; Web Application Server) : 사용자에게 동적 서비스를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업을 수행하거나, 웹 서버와 데이터베이스 서버 또는 웹 서버와 파일 서버 사이에서 인터페이스 역할을 수행하는 서버이다.
  - 데이터베이스 서버(DB Server) : 데이터베이스와 이를 관리하는 DBMS를 운영하는 서버이다.
  - 파일 서버(File Server) : 데이터베이스에 저장하기에는 비효율적이거나, 서비스 제공을 목적으로 유지하는 파일들을 저장하는 서버이다.

핵심 236 소프트웨어 환경

소프트웨어 환경은 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성된다.

- 시스템 소프트웨어에는 운영체제(OS), 웹 서버 및 WAS 운용을 위한 서버 프로그램, DBMS 등이 있다.
- 개발 소프트웨어에는 요구사항 관리 도구, 설계/모델링 도구, 구현 도구, 빌드 도구, 테스트 도구, 형상 관리 도구 등이 있다.
  - 요구사항 관리 도구 : 요구사항의 수집과 분석, 추적 등을 편리하게 도와주는 소프트웨어
  - 설계/모델링 도구 : UML(통합 모델링 언어)을 지원하며, 개발의 전 과정에서 설계 및 모델링을 도와주는 소프트웨어
  - 구현 도구 : 개발 언어를 통해 애플리케이션의 실제 구현을 지원하는 소프트웨어
  - 빌드 도구 : 구현 도구를 통해 작성된 소스의 빌드 및 배포, 라이브러리 관리를 지원하는 소프트웨어
  - 테스트 도구 : 모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 소프트웨어
  - 형상 관리 도구 : 산출물들을 버전별로 관리하여 품질 향상을 지원하는 소프트웨어

핵심 237 서버 개발 프레임워크

서버 개발 프레임워크는 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어를 의미한다.

- 서버 개발 프레임워크에 따라 지원하는 프로그래밍 언어가 제한적이므로 선정할 수 있는 프레임워크도 제한적이다.
- 서버 개발 프레임워크의 대부분은 모델-뷰-컨트롤러(MVC) 패턴을 기반으로 개발되었다.
- 대표적인 서버 개발 프레임워크의 종류는 다음과 같다.

프레임워크	특징
Spring	JAVA를 기반으로 만들어진 프레임워크로, 전자정부 표준 프레임워크의 기반 기술로 사용되고 있다.
Node.js	JavaScript를 기반으로 만들어진 프레임워크로, 비동기 입·출력 처리와 이벤트 위주의 높은 처리 성능을 갖고 있어 실시간으로 입·출력이 빈번한 애플리케이션에 적합하다.
Django	Python을 기반으로 만들어진 프레임워크로, 컴포넌트의 재사용과 플러그인화를 강조하여 신속한 개발이 가능하도록 지원한다.
Codeigniter	PHP를 기반으로 만들어진 프레임워크로, 인터페이스가 간편하며 서버 자원을 적게 사용한다.
Ruby on Rails	Ruby를 기반으로 만들어진 프레임워크로, 테스트를 위한 웹 서버를 지원하며 데이터베이스 작업을 단순화, 자동화시켜 개발 코드의 길이가 짧아 신속한 개발이 가능하다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 238 소프트웨어 개발 보안 점검 항목

소프트웨어 개발 보안 점검 항목은 소프트웨어 개발의 각 단계에서 점검되어야 할 보안 항목들을 말한다.

세션 통제	세션은 서버와 클라이언트의 연결을 말하며, 세션 통제는 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것을 의미한다.
입력 데이터 검증 및 표현	입력 데이터에 대한 유효성 검증체계를 갖추고, 검증 실패 시 이를 처리할 수 있도록 코딩하는 것을 의미한다.
보안 기능	인증, 접근제어, 기밀성, 암호화 등의 기능을 의미한다.
시간 및 상태	동시 수행을 지원하는 병렬 처리 시스템이나 다수의 프로세스가 동작하는 환경에서 시간과 실행 상태를 관리하여 시스템이 원활히 동작되도록 코딩하는 것을 의미한다.
에러처리	• 소프트웨어 실행 중 발생할 수 있는 오류들을 사전에 정의하여 에러로 인해 발생할 수 있는 문제들을 예방하는 것을 의미한다. • 보안 약점에는 오류 메시지를 통한 정보 노출, 오류 상황 대응 부재 등이 있다.
코드 오류	개발자들이 코딩 중 실수하기 쉬운 형(Type) 변환, 자원의 반환 등을 고려하며 코딩하는 것을 의미한다.
캡슐화	데이터(속성)와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것을 의미한다.
API 오용	API를 잘못 사용하거나 보안에 취약한 API를 사용하지 않도록 고려하여 코딩하는 것을 의미한다.

핵심 239 API(Application Programming Interface)

API는 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스를 의미한다.

- API는 프로그래밍 언어에서 특정한 작업을 수행하기 위해 사용되거나, 운영체제의 파일 제어, 화상 처리, 문자 제어 등의 기능을 활용하기 위해 사용된다.
- API는 개발에 필요한 여러 도구를 제공하기 때문에 이를 이용하면 원하는 기능을 쉽고 효율적으로 구현할 수 있다.
- API의 종류에는 Windows API, 단일 유닉스 규격(SUS), Java API, 웹 API 등이 있으며, 누구나 무료로 사용할 수 있게 공개된 API를 Open API라고 한다.

핵심 240 배치 프로그램

배치 프로그램(Batch Program)의 개요

배치 프로그램은 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 의미한다.

- 배치 프로그램이 갖추어야 하는 필수 요소는 다음과 같다.

대용량 데이터	대량의 데이터를 가져오거나, 전달하거나, 계산하는 등의 처리가 가능해야 한다.
자동화	심각한 오류가 발생하는 상황을 제외하고는 사용자의 개입 없이 수행되어야 한다.
견고성	잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 한다.
안정성/신뢰성	오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있어야 한다.
성능	다른 응용 프로그램의 수행을 방해하지 않아야 하고, 지정된 시간 내에 처리가 완료되어야 한다.

핵심 241 패키지 소프트웨어

패키지 소프트웨어는 기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 소프트웨어를 의미한다.

- 기업에서는 패키지 소프트웨어를 구입하여 기업 환경에 적합하게 커스터마이징(Customizing)하여 사용한다.
- 패키지 소프트웨어를 이용하여 시스템을 구축하는 방식을 패키지 개발 방식이라고 한다.
- 기능 요구사항을 70% 이상 충족시키는 패키지 소프트웨어가 있을 때만 사용하는 것이 적합하다.

※ 패키지 소프트웨어와 전용 개발 소프트웨어의 비교

	패키지 소프트웨어	전용 개발 소프트웨어
기능 요구사항	70% 이상 충족시키는 패키지 소프트웨어가 있는 경우 이용	모든 기능 요구사항 반영 가능
안정성	품질이 검증되었고, 업계 표준 준용	개발자의 역량에 따라 달라짐
라이선스	판매자	회사
생산성	개발을 위한 인력과 시간이 절약됨	개발을 위한 인력과 시간이 필요함
호환성	보장이 안 됨	설계 단계부터 고려하여 개발
유지보수	결함 발생 시 즉시 대응이 어려움	결함 발생 시 즉시 대응이 가능

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

핵심 242 C/C++의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기
문자	char	1Byte
부호없는 문자형	unsigned char	1Byte
정수	short	2Byte
	int	4Byte
	long	4Byte
	long long	8Byte
실수	float	4Byte
	double	8Byte
	long double	8Byte

핵심 243 C언어의 구조체

배열이 자료의 형과 크기가 동일한 변수의 모임이라면 구조체는 자료의 종류가 다른 변수의 모임이라고 할 수 있다. 예를 들어 이름, 직위, 급여 등의 필드가 필요한 사원 자료를 하나의 단위로 관리하려면 이름과 직위는 문자, 급여는 숫자와 같이 문자와 숫자가 혼용되므로 배열로는 처리할 수 없다. 이런 경우 구조체를 사용하면 간단하게 처리할 수 있다.

- 구조체를 정의한다는 것은 int나 char 같은 자료형을 하나 만드는 것을 의미한다.
- 구조체는 'structure(구조)'의 약어인 'struct'를 사용하여 정의한다.
- 구조체 정의 예

```
struct sawon {
   char name[10];
   char position[10];
   int pay;
}
```

핵심 244 JAVA의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기
문자	char	2Byte
정수	byte	1Byte
	short	2Byte
	int	4Byte
	long	8Byte
실수	float	4Byte
	double	8Byte
논리	boolean	1Byte

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 245 변수의 개요 / 변수명 작성 규칙

변수의 개요

변수(Variable)는 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로, 변할 수 있는 값을 의미한다.

- 변수는 저장하는 값에 따라 정수형, 실수형, 문자형, 포인터형 등으로 구분한다.

변수명 작성 규칙

- 영문자, 숫자, _(under bar)를 사용할 수 있다.
- 첫 글자는 영문자나 _(under bar)로 시작해야 하며, 숫자는 올 수 없다.
- 글자 수에 제한이 없다.
- 공백이나 *, +, −, / 등의 특수문자를 사용할 수 없다.
- 대 · 소문자를 구분한다.
- 예약어를 변수명으로 사용할 수 없다.
- 변수 선언 시 문장 끝에 반드시 세미콜론(;)을 붙여야 한다.
- 변수 선언 시 변수명에 데이터 타입을 명시하는 것을 헝가리안 표기법(Hungarian Notation)이라고 한다.

핵심 246 기억 클래스

변수 선언 시 메모리 내에 변수의 값을 저장하기 위한 기억영역이 할당되는데, 할당되는 기억영역에 따라 사용 범위에 제한이 있다. 이러한 기억영역을 결정하는 작업을 기억 클래스(Storage Class)라 한다.

- C언어에서는 다음과 같이 5가지 종류의 기억 클래스를 제공한다.

종류	기억영역	예약어	생존기간	사용 범위
자동 변수	메모리(스택)	auto	일시적	지역적
레지스터 변수	레지스터	register		
정적 변수(내부)	메모리(데이터)	static	영구적	
정적 변수(외부)				전역적
외부 변수		extern		

자동 변수(Automatic Variable)

자동 변수는 함수나 코드의 범위를 한정하는 블록 내에서 선언되는 변수이다.

- 함수나 블록이 실행되는 동안에만 존재하며 이를 벗어나면 자동으 소멸된다.

외부 변수(External Variable)

외부 변수는 현재 파일이나 다른 파일에서 선언된 변수나 함수를 참조(reference)하기 위한 변수이다.

- 외부 변수는 함수 밖에서 선언한다.

정적 변수(Static Variable)

정적 변수는 함수나 블록 내에서 선언하는 내부 정적 변수와 함수 외부에서 선언하는 외부 정적 변수가 있다.

- 내부 정적 변수는 선언한 함수나 블록 내에서만 사용할 수 있고, 외부 정적 변수는 모든 함수에서 사용할 수 있다.

레지스터 변수(Register Variable)

레지스터 변수는 메모리가 아닌 CPU 내부의 레지스터에 기억영역을 할당받는 변수이다.

- 자주 사용되는 변수를 레지스터에 저장하여 처리 속도를 높이기 위해 사용한다.

핵심 247 연산자의 종류

산술 연산자

산술 연산자는 가, 감, 승, 제 등의 산술 계산에 사용되는 연산자를 말한다.

- 산술 연산자에는 일반 산술식과 달리 한 변수의 값을 증가하거나 감소시키는 증감 연산자가 있다.

연산자	의미	비고
+	덧셈	
−	뺄셈	
*	곱셈	
/	나눗셈	

%	나머지	
++	증가 연산자	전치 : 변수 앞에 증감 연산자가 오는 형태로 먼저 변수의 값을 증감시킨 후 변수를 연산에 사용한다(++a, --a). 후치 : 변수 뒤에 증감 연산자가 오는 형태로 먼저 변수를 연산에 사용한 후 변수의 값을 증감시킨다(a++, a--).
--	감소 연산자	

관계 연산자

관계 연산자는 두 수의 관계를 비교하여 참(true) 또는 거짓(false)을 결과로 얻는 연산자이다.

- 거짓은 0, 참은 1로 사용되지만 0외의 모든 숫자도 참으로 간주된다.

연산자	의미
==	같다
!=	같지 않다
>	크다
>=	크거나 같다
<	작다
<=	작거나 같다

비트 연산자

비트 연산자는 비트별(0, 1)로 연산하여 결과를 얻는 연산자이다.

연산자	의미	비고
&	and	모든 비트가 1일 때만 1
^	xor	모든 비트가 같으면 0, 하나라도 다르면 1
|	or	모든 비트 중 한 비트라도 1이면 1
~	not	각 비트의 부정, 0이면 1, 1이면 0
<<	왼쪽 시프트	비트를 왼쪽으로 이동
>>	오른쪽 시프트	비트를 오른쪽으로 이동

논리 연산자

논리 연산자는 두 개의 논리 값을 연산하여 참(true) 또는 거짓(false)을 결과로 얻는 연산자이다. 관계 연산자와 마찬가지로 거짓은 0, 참은 1이다.

연산자	의미	비고
!	not	부정
&&	and	모두 참이면 참
||	or	하나라도 참이면 참

대입 연산자

연산 후 결과를 대입하는 연산식을 간략하게 입력할 수 있도록 대입 연산자를 제공한다. 대입 연산자는 산술, 관계, 비트, 논리 연산자에 모두 적용할 수 있다.

연산자	예	의미
+=	a += 1	a = a + 1
-=	a -= 1	a = a - 1
*=	a *= 1	a = a * 1
/=	a /= 1	a = a / 1
%=	a %= 1	a = a % 1
<<=	a <<= 1	a = a << 1
>>=	a >>= 1	a = a >> 1

조건 연산자

조건 연산자는 조건에 따라 서로 다른 수식을 수행한다.

형식

조건 ? 수식1 : 수식2;	'조건'의 수식이 참이면 '수식1'을, 거짓이면 '수식2'를 실행한다.

기타 연산자

연산자	의미
sizeof	자료형의 크기를 표시한다.
,(콤마)	콤마로 구분하여 한 줄에 두 개 이상의 수식을 작성하거나 변수를 정의한다.
(자료형)	• 사용자가 자료형을 다른 자료형으로 변환할 때 사용하는 것으로, cast(캐스트) 연산자라고 부른다. • 변환할 자료형을 괄호 안에 넣어서 변환할 값이나 변수명 앞에 놓는다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 248 연산자 우선순위

- 한 개의 수식에 여러 개의 연산자가 사용되면 기본적으로 아래 표의 순서대로 처리된다.
- 아래 표의 한 줄에 가로로 나열된 연산자는 우선순위가 같기 때문에 결합규칙에 따라 ←는 오른쪽에 있는 연산자부터, →는 왼쪽에 있는 연산자부터 차례로 계산된다.

대분류	중분류	연산자	결합 규칙	우선 순위
단항 연산자	단항 연산자	!(논리 not) ~(비트 not) ++(증가) --(감소) sizeof(기타)	←	높음 ↑
이항 연산자	산술 연산자	* / %(나머지)	→	
		+ -		
	시프트 연산자	《 》		
	관계 연산자	〈 〈= 〉= 〉		
		==(같다) !=(같지 않다)		
	비트 연산자	&(비트 and) ^(비트 xor) |(비트 or)		
	논리 연산자	&&(논리 and) ||(논리 or)		
삼항 연산자	조건 연산자	? :	→	↓
대입 연산자	대입 연산자	= += -= *= /= %= 《= 》= 등	←	
순서 연산자	순서 연산자	,	→	낮음

핵심 249 scanf( ) 함수

scanf( ) 함수는 C언어의 표준 입력 함수로, 키보드로 입력받아 변수에 저장하는 함수이다.

형식

scanf (서식 문자열, 변수의 주소)	• 서식 문자열 : 입력받을 데이터의 자료형을 지정한다. • 변수의 주소 : 데이터를 입력받을 변수를 적는다. 변수의 주소로 입력받아야 하기 때문에 변수에 주소연산자 &를 붙인다.

예 scanf("%3d", &a);

▶ % : 서식 문자임을 지정

▶ 3 : 입력 자릿수를 3자리로 지정

▶ d : 10진수로 입력

▶ &a : 입력받은 데이터를 변수 a의 주소에 저장

특징

- 입력받을 데이터의 자료형, 자릿수 등을 지정할 수 있다.
- 한 번에 여러 개의 데이터를 입력 받을 수 있다.
- 서식 문자열과 변수의 자료형은 일치해야 한다.

  예 scanf("%d %f", &i, &j); → '%d'와 i, "%f"와 j는 자료형이 일치해야 한다.

서식 문자열

서식 문자열은 printf( ) 함수로 출력할 때도 동일하게 적용된다.

서식 문자열	의미
%d	정수형 10진수를 입 · 출력하기 위해 지정한다.
%u	부호없는 정수형 10진수를 입 · 출력하기 위해 지정한다.
%o	정수형 8진수를 입 · 출력하기 위해 지정한다.
%x	정수형 16진수를 입 · 출력하기 위해 지정한다.
%c	문자를 입 · 출력하기 위해 지정한다.
%s	문자열을 입 · 출력하기 위해 지정한다.
%f	소수점을 포함하는 실수를 입 · 출력하기 위해 지정한다.
%e	지수형 실수를 입 · 출력하기 위해 지정한다.
%ld	long형 10진수를 입 · 출력하기 위해 지정한다.
%lo	long형 8진수를 입 · 출력하기 위해 지정한다.
%lx	long형 16진수를 입 · 출력하기 위해 지정한다.
%p	주소를 16진수로 입 · 출력하기 위해 지정한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 250 printf( ) 함수

printf( ) 함수는 C언어의 표준 출력 함수로, 인수로 주어진 값을 화면에 출력하는 함수이다.

형식

printf (서식 문자열, 변수)	• 서식 문자열 : 변수의 자료형에 맞는 서식 문자열을 입력한다. • 변수 : 서식 문자열의 순서에 맞게 출력할 변수를 적는다. scanf( )와 달리 주소 연산자 &를 붙이지 않는다.

예 printf("%-8.2f", 200.2);
(∨는 빈 칸을 의미함)

```
200.20∨∨
```

▶ % : 서식 문자임을 지정

▶ - : 왼쪽부터 출력

▶ 8 : 출력 자릿수를 8자리로 지정

▶ 2 : 소수점 이하를 2자리로 지정

▶ f : 실수로 출력

주요 제어문자

제어문자란 입력 혹은 출력 내용을 제어하는 문자이다.

문자	의미	기능
\n	new line	커서를 다음 줄 앞으로 이동한다.
\b	backspace	커서를 왼쪽으로 한 칸 이동한다.
\t	tab	커서를 일정 간격 띄운다.
\r	carriage return	커서를 현재 줄의 처음으로 이동한다.
\0	null	널 문자를 출력한다.
\'	single quote	작은따옴표를 출력한다.
\"	double quote	큰따옴표를 출력한다.
\a	alert	스피커로 벨 소리를 출력한다.
\\	backslash	역 슬래시를 출력한다.
\f	form feed	한 페이지를 넘긴다.

핵심 251 JAVA에서의 표준 출력

• JAVA에서 값을 화면에 출력할 때는 System 클래스의 서브 클래스인 out 클래스의 메소드 print( ), println( ), printf( ) 등을 사용하여 출력한다.

• 형식 1 : 서식 문자열에 맞게 변수의 내용을 출력한다.

```
System.out.printf(서식 문자열, 변수)
```

– printf( ) 메소드는 C언어의 printf( ) 함수와 사용법이 동일하다.

예 System.out.printf("%-8.2f", 200.2);
(∨는 빈 칸을 의미함)

```
200.20∨∨
```

▶ % : 서식 문자임을 지정

▶ - : 왼쪽부터 출력

▶ 8 : 출력 자릿수를 8자리로 지정

▶ 2 : 소수점 이하를 2자리로 지정

▶ f : 실수로 출력

• 형식 2 : 값이나 변수의 내용을 형식없이 출력한다.

```
System.out.print( )
```

– 문자열을 출력할 때는 큰따옴표로 묶어줘야 한다.

– 문자열 또는 문자열 변수를 연속으로 출력할 때는 +를 이용한다.

예 System.out.print("abc123" + "def");

```
abc123def
```

• 형식 3 : 값이나 변수의 내용을 형식없이 출력한 후 커서를 다음 줄의 처음으로 이동한다.

```
System.out.println( )
```

– println( ) 메소드는 출력 후 다음 줄로 이동한다는 것을 제외하면 print( ) 메소드와 사용법이 동일하다.

예 System.out.print("abc123" + "def");

```
abc123def
```

커서의 위치

핵심 252 기타 표준 입 · 출력 함수

입력	getchar( )	키보드로 한 문자를 입력받아 변수에 저장하는 함수
	gets( )	키보드로 문자열을 입력받아 변수에 저장하는 함수로, Enter를 누르기 전까지를 하나의 문자열로 인식하여 저장함
출력	putchar( )	인수로 주어진 한 문자를 화면에 출력하는 함수
	puts( )	인수로 주어진 문자열을 화면에 출력한 후 커서를 자동으로 다음 줄 앞으로 이동하는 함수

핵심 253 단순 if문

if문은 조건에 따라서 실행할 문장을 달리하는 제어문이며, 단순 if문은 조건이 한 개 일 때 사용하는 제어문이다.

- 조건이 참일 때만 실행할 문장을 지정할 수도 있고, 참과 거짓에 대해 각각 다른 실행문을 지정할 수도 있다.
- 형식1 : 조건이 참일 때만 실행한다.
  - 조건이 참일 때 실행할 문장이 하나인 경우

```
if(조건)
    실행할 문장;
```

if는 조건 판단문에 사용되는 예약어이므로 그대로 적는다.
조건은 참(1) 또는 거짓(0)이 결과로 나올 수 있는 수식을 ( ) 안에 입력한다.

조건이 참일 경우 실행할 문장을 적는다.

  - 조건이 참일 때 실행할 문장이 두 문장 이상인 경우

```
if(조건)
{
    실행할 문장1;
    실행할 문장2;
    ⋮
}
```

{ } 사이에 조건이 참일 경우 실행할 문장을 적는다.

예제 1 a가 10보다 크면 a에서 10을 빼기

```
#include <stdio.h>
main( )
{
  int a = 15, b;
  if (a > 10) ❶
     b = a - 10; ❷
  printf("%d\n", b); ❸
```

a가 10보다 크면 ❷번 문장을 실행하고, 아니면 ❸번 문장으로 이동해서 실행을 계속한다.

❶번의 조건식이 참일 경우 실행할 문장이다. b는 5가 된다.

여기서는 ❶번의 조건식이 거짓일 경우 실행할 문장이 없다. 조건 판단문을 벗어나면 무조건 ❸번으로 온다.

결과 5

형식2 : 조건이 참일 때와 거짓 때 실행할 문장이 다르다.

```
if(조건)
   실행할 문장1;
else
   실행할 문장2;
```

조건이 참일 경우 실행할 문장을 적는다. 참일 경우 실행할 문장이 두 문장 이상이면 { }를 입력하고 그 사이에 문장을 적는다.

조건이 거짓일 경우 실행할 문장을 적는다. 두 문장 이상인 경우 {}를 입력하고 그 사이에 문장을 적는다.

예제 2 a가 b보다 크면 'a-b', 아니면 'b-a'를 수행하기

```
#include <stdio.h>
main( )
{
  int a = 10, b = 20, cha;
  if (a > b) ❶
     cha = a - b; ❷
  else ❸
     cha = b - a; ❹
  printf("%d\n", cha);
}
```

a가 b보다 크면 ❷번 문장을 실행하고, 아니면 ❸번의 다음 문장인 ❹번 문장을 실행한다.

❶번의 조건식이 참일 경우 실행할 문장이다. 참이 아니기 때문에 초기화 시키지 않은 cha에는 알 수 없는 값이 그대로 있게 된다.

❶번의 조건식이 거짓일 경우 실행할 문장의 시작점이다.

❶번의 조건식이 거짓일 경우 실행할 실제 처리문이다. cha는 10이 된다.

결과 10

핵심 254 switch문

switch문은 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문이다.

• 형식

```
switch(수식) ❶
{ ❷
  case 레이블1: ❸
    실행할 문장1;
    break;
  case 레이블2: ❹
    실행할 문장2;
    break;
      ⋮
```

• switch는 switch문에 사용되는 예약어로 그대로 입력한다.
• 수식 : '레이블1' ~ '레이블n'의 값 중 하나를 도출하는 변수나 수식을 입력한다.

❷~❺번이 switch문의 범위이다. '{'로 시작해서 '}'로 끝난다. 반드시 입력해야 한다.

• case는 switch문에서 레이블을 지정하기 위한 예약어로 그대로 입력해야 한다.
• 레이블1 : ❶번 식의 결과가 될 만한 값 중 하나를 입력한다. 결과가 '레이블1'과 일치하면 이곳으로 찾아온다. 식의 결과가 5종류로 나타나면 case문이 5번 나와야한다.

❶번 식의 결과가 ❸번의 '레이블1'과 일치할 때 실행할 문장이다.

switch문을 탈출하여 ❺번으로 간다.

❶번의 식의 결과가 '레이블2'와 일치하면 찾아오는 곳이다.

❶번의 식의 결과가 ❹번의 '레이블2'와 일치할 때 실행할 문장이다.

switch문을 탈출하여 ❺번으로 간다.

```
  default:                    ❶번의 식의 결과가 '레이블1' ~ '레이블n'에 해당하지 않는 경우 찾아오는 곳이다.
    실행할 문장3;
} ❺
```

- case문의 레이블에는 한 개의 상수만 지정할 수 있으며, int, char, enum형의 상수만 가능하다.
- case문의 레이블에는 변수를 지정할 수 없다.
- break문은 생략이 가능하지만 break문이 생략되면 수식과 레이블이 일치할 때 실행할 문장부터 break문 또는 switch 문이 종료될 때까지 모든 문장이 실행된다.

예제 점수(jum)에 따라 등급 표시하기

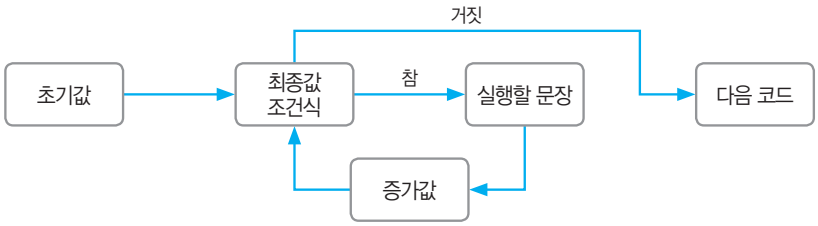
```
#include <stdio.h>
main( )
{
  int jum = 85;
  switch (jum / 10)           jum을 10으로 나눠 결과에 해당하는 숫자를 찾아간다. 85/10은 8.5지만 C 언어에서 정수 나눗셈은 결과도 정수이므로 결과는 8이다. 8에 해당하는 ❺번으로 이동하여 ❻, ❼번을 실행한다.
  { ❶                         ❶~❽번까지가 switch 조건문의 범위이다.
    case 10:                  100점일 경우 'jum/10'의 결과인 10이 찾아오는 곳이지만 할 일은 'case 9:'와 같으므로 아무것도 적지 않는다. 아무것도 적지 않으면 다음 문장인 ❷번으로 이동한다.
    case 9: ❷                 'jum/10'이 9일 경우 찾아오는 곳이다. ❸, ❹번을 실행한다.
      printf("학점은 A입니다.\n"); ❸   "학점은 A입니다."를 출력한다.
      break; ❹                break를 만나면 switch문을 탈출하여 ❾번으로 이동한다.
    case 8: ❺                 'jum/10'이 8일 경우 찾아오는 곳이다. ❻, ❼번을 실행한다.
      printf("학점은 B입니다.\n"); ❻   "학점은 B입니다."를 출력한다.
      break; ❼                switch문을 탈출하여 ❾번으로 이동한다.
    case 7:
      printf("학점은 C입니다.\n");
      break;
    case 6:
      printf("학점은 D입니다.\n");
      break;
    default:                  case 10~6에 해당되지 않는 경우, 즉 jum이 59 이하인 경우 찾아오는 곳이다.
      printf("학점은 F입니다.\n");   "학점은 F입니다."를 출력한다.
    } ❽
} ❾
```

결과 학점은 B입니다.

핵심 255 for문

for문은 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문이다.

- for문은 초기값을 정한 다음 최종값에 대한 조건이 참이면 실행할 문장을 실행한 후 초기값을 증가값 만큼 증가시키면서 최종값에 대한 조건이 참인 동안 실행할 문장을 반복 수행한다.



for문의 동작 과정

- 형식

for(식1; 식2; 식3)

- for는 반복문을 의미하는 예약어로 그대로 입력한다.
- 식1 : 초기값을 지정할 수식을 입력한다.
- 식2 : 최종값을 지정할 수식을 입력한다.
- 식3 : 증가값으로 사용할 수식을 입력한다.

실행할 문장;

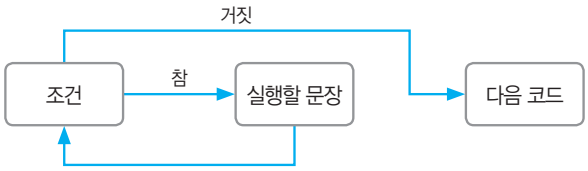
식2가 참일 동안 실행할 문장을 입력한다. 실행할 문장이 두 문장 이상일 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

- for문은 처음부터 최종값에 대한 조건식을 만족하지 못하면 한 번도 수행하지 않는다.

핵심 256 while문

while문은 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문이다.

- while문은 조건이 참인동안 실행할 문장을 반복 수행하다가 조건이 거짓이면 while문을 끝낸 후 다음 코드를 실행한다.
- while문은 조건이 처음부터 거짓이면 한 번도 수행하지 않는다.



while문의 동작 과정

- 형식

while(조건)

- while은 반복문에 사용되는 예약어로 그대로 입력한다.
- (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참(1)을 직접 입력할 수도 있다.

실행할 문장;

조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

예제 다음은 1~5까지의 합을 더하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main( )
{
   int a = 0, hap = 0;
   while (a < 5) ❶
   { ❷
      a++; ❸
      hap += a; ❹
   } ❺
   printf("%d, %d\n", a, hap); ❻
}
```

a가 5보다 작은 동안 ❷~❺번 문장을 반복하여 수행한다.

❷~❺번까지가 반복문의 범위이다. 반복문에서 실행할 문장이 하나인 경우는 { }를 생략해도 된다.

'a = a + 1;'과 동일하다. a의 값을 1씩 증가시킨다.

'hap = hap + a'와 동일하다. a의 값을 hap에 누적시킨다.

반복문의 끝이다.

결과 5, 15

a가 5가 되었을 때 5를 hap에 누적한 다음 while 문을 벗어나기 때문에 a는 5로 끝난다.

핵심 257 do~while문

do~while문은 조건이 참인 동안 정해진 문장을 반복 수행하다가 조건이 거짓이면 반복문을 벗어나는 while문과 같은 동작을 하는데, 다른 점은 do~while문은 실행할 문장을 무조건 한 번 실행한 다음 조건을 판단하여 탈출 여부를 결정한다는 것이다.

- do~while문은 실행할 문장을 우선 실행한 후 조건을 판별하여 조건이 참이면 실행할 문장을 계속 반복 수행하고, 거짓이면 do~while문을 끝낸 후 다음 코드를 실행한다.

do~while문의 동작 과정

- 형식

do	do는 do~while문에 사용되는 예약어로, do~while의 시작 부분에 그대로 입력한다.
실행할 문장;	조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 실행할 문장들을 입력한다.
while(조건);	• while은 do~while문에 사용되는 예약어로, do~while의 끝 부분에 그대로 입력한다. • (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참(1)을 직접 입력할 수도 있다.

예제 다음은 1부터 10까지 홀수의 합을 더하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main ( )
{
   int a = 1, hap = 0;
   do ❶
   { ❷
      hap += a; ❸
      a += 2; ❹
   } while(a < 10); ❺
   printf("%d, %d\n", a, hap); ❻

}
```

do~while 반복문의 시작점이다. ❷~❺번 사이의 문장을 반복하여 수행한다.

❷~❺번까지가 반복문의 범위이다.

'hap = hap + a'와 동일하다. a의 값을 hap에 누적시킨다.

'a = a + 2;'와 동일하다. a의 값을 2씩 증가시킨다.

a가 10보다 작은 동안 ❷~❺번 사이의 문장을 반복하여 수행한다.

결과 11, 25

a가 9가 되었을 때 9를 hap에 누적한 다음 a에 2를 더해 a가 11이 되었을 때 do~while문을 벗어나기 때문에 a는 11로 끝난다.

핵심 258 배열

배열의 개념

배열은 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 조합해서 하나의 이름으로 정의해 사용하는 것을 말한다.

- 배열은 하나의 이름으로 여러 기억장소를 가리키기 때문에 배열에서 개별적인 요소들의 위치는 첨자를 이용하여 지정한다.
- 배열은 변수명 뒤에 대괄호 [ ]를 붙이고 그 안에 사용할 개수를 지정한다.
- C언어에서 배열의 위치는 0부터 시작된다.
- 배열은 행 우선으로 데이터가 기억장소에 할당된다.
- C 언어에서 배열 위치를 나타내는 첨자 없이 배열 이름을 사용하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다.

1차원 배열

- 1차원 배열은 변수들을 일직선상의 개념으로 조합한 배열이다.
- 형식

자료형 변수명[개수];	• 자료형 : 배열에 저장할 자료의 형을 지정한다. • 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다. • 개수 : 배열의 크기를 지정하는 것으로 생략할 수 있다.

예 int a[5] : 5개의 요소를 갖는 정수형 배열 a

	첫 번째	두 번째	세 번째	네 번째	다섯 번째
배열 a	a[0]	a[1]	a[2]	a[3]	a[4]

※ a[3] : a는 배열의 이름이고, 3은 첨자로서 배열 a에서의 위치를 나타낸다. a[3]에 4를 저장시키려면 'a[3] = 4'와 같이 작성한다.

예제 1 1차원 배열 a의 각 요소에 10, 11, 12, 13, 14를 저장한 후 출력하기

```
#include <stdio.h>
main( )
{
  int a[5];
```

5개의 요소를 갖는 정수형 배열 a를 선언한다. 선언할 때는 사용할 개수를 선언하고, 사용할 때는 첨자를 0부터 사용하므로 주의해야 한다.

	첫 번째	두 번째	세 번째	네 번째	다섯 번째
배열 a	a[0]	a[1]	a[2]	a[3]	a[4]

```
  int i;
```

정수형 변수 i를 선언한다

```
  for (i = 0; i < 5; i++)
```

반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ❶번 문장을 반복하여 수행한다. 그러니까 ❶번 문장을 5회 반복하는 것이다.

```
    a[i] = i + 10; ❶
```

배열 a의 i번째에 i+10을 저장시킨다. i는 0~4까지 변하므로 배열 a에 저장된 값은 다음과 같다.

배열 a	10	11	12	13	14
	a[0]	a[1]	a[2]	a[3]	a[4]

```
  for (i = 0; i < 5; i++)
```

반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ❷번 문장을 반복하여 수행한다.

```
    printf("%d ", a[i]); ❷
```

배열 a의 i번째를 출력한다. i는 0~4까지 변하므로 출력 결과는 다음과 같다. 서식 문자열에 '\n'이 없기 때문에 한 줄에 붙여서 출력한다.

```
}
```

결과 10 11 12 13 14

핵심 259 2차원 배열

- 2차원 배열은 변수들을 평면, 즉 행과 열로 조합한 배열이다.
- 형식

자료형 변수명[행개수][열개수]	• 자료형 : 배열에 저장할 자료의 형을 지정한다. • 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다. • 행개수 : 배열의 행 크기를 지정한다. • 열개수 : 배열의 열 크기를 지정한다.

예 int b[3][3] : 3개의 행과 3개의 열을 갖는 정수형 배열 b

배열 b

열

0, 0	0, 1	0, 2	b[0][2]
1, 0	1, 1	1, 2	
2, 0	2, 1	2, 2	

행

b[0][2] : b는 배열의 이름이고, 0은 행 첨자, 2는 열 첨자로서 배열 b에서의 위치를 나타낸다.

예제 3행 4열의 배열에 다음과 같이 숫자 저장하기

배열 a

1	2	3	4
5	6	7	8
9	10	11	12

```
#include <stdio.h>
main( )
{
  int a[3][4];
  int i, j, k = 0;
  for (i = 0; i < 3; i++) ❶

  { ❷
    for (j = 0; j < 4; j++) ❸

    { ❹
      k++; ❺
      a[i][j] = k; ❻
    } ❼
  } ❽
}
```

3행 4열의 크기를 갖는 정수형 배열 a를 선언한다.

정수형 변수 i, j, k를 선언하고 k를 0으로 초기화 한다.

반복 변수 i가 0에서 시작하여 1씩 증가하면서 3보다 작은 동안 ❷~❽번을 반복하여 수행한다. 결국 ❸번 문장을 3회 반복한다.

❷~❽이 ❶번 반복문의 반복 범위이지만 실제 실행할 문장은 ❸번 하나이다.

반복 변수 j가 0에서 시작하여 1씩 증가하면서 4보다 작은 동안 ❹~❼번을 반복하여 수행한다.
- i가 0일 때 j는 0에서 3까지 4회 반복
- i가 1일 때 j는 0에서 3까지 4회 반복
- i가 2일 때 j는 0에서 3까지 4회 반복 수행하므로 ❺~❻번을 총 12회 수행한다.

❹~❼이 ❸번 반복문의 반복 범위이다.

k를 1씩 증가시킨다. k는 총 12회 증가하므로 1~12까지 변한다.

배열 a의 i행 j열에 k를 기억시킨다. a[0][0]~a[2][3]까지 1~12가 저장된다.

❹번의 짝이다.

❷번의 짝이다.

첫 번째 { 의 짝이자 프로그램의 끝이다.

핵심 260 배열의 초기화

- 배열 선언 시 초기값을 지정할 수 있다.
- 배열을 선언할 때 배열의 크기를 생략하는 경우에는 반드시 초기값을 지정해야 초기값을 지정한 개수 만큼의 배열이 선언된다.

예 1차원 배열 초기화

방법 1 char a[3] = {'A', 'B', 'C'}

방법 2 char a[ ] = {'A', 'B', 'C'}

배열 a

A	B	C
a[0]	a[1]	a[2]

예 2차원 배열 초기화

방법 1 int a[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };

방법 2 int a[2][4] = {10, 20, 30, 40, 50, 60, 70, 80}

배열 a

a[0][0]	a[0][1]	a[0][2]	a[0][3]
10	20	30	40
50	60	70	80
a[1][0]	a[1][1]	a[1][2]	a[1][3]

- 배열의 개수보다 적은 수로 배열을 초기화하면 입력된 값만큼 지정한 숫자가 입력되고, 나머지 요소에는 0이 입력된다.

예 int a[5] = { 3, }; 또는 int a[5] = { 3 };

배열 a

3	0	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

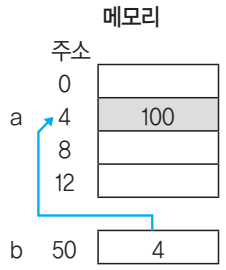
시험에 나오는 것만 공부한다! 시나공시리즈

핵심 261 포인터와 포인터 변수

포인터는 변수의 주소를 말하며, C언어에서는 주소를 제어할 수 있는 기능을 제공한다.

- C언어에서 변수의 주소를 저장할 때 사용하는 변수를 포인터 변수라 한다.
- 포인터 변수를 선언할 때는 자료의 형을 먼저 쓰고 변수명 앞에 간접 연산자 *를 붙인다(예 int *a;).
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 알아낼 때는 변수 앞에 번지 연산자 &를 붙인다(예 a = &b;).
- 실행문에서 포인터 변수에 간접 연산자 *를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 말한다(예 c = *a;).
- 포인터 변수는 필요에 의해 동적으로 할당되는 메모리 영역인 힙 영역에 접근하는 동적 변수이다.

예를 들어, a 변수에 100을 저장시키고, a 변수의 주소를 포인터 변수 b에 기억시켰다면 다음 그림과 같이 표현하고 말할 수 있다.



- a는 메모리의 4번지에 대한 이름이다.
- a 변수의 주소는 4다.
- a 변수에는 100이 기억되어 있다.
- 4번지에는 100이 기억되어 있다.
- &a는 a 변수의 주소를 말한다. 즉 &a는 4다.
- 포인터 변수 b는 a 변수의 주소를 기억하고 있다.
- 포인터 변수가 가리키는 곳의 값을 말할 때는 *을 붙인다.
- *b는 b에 저장된 주소가 가리키는 곳에 저장된 값을 말하므로 100이다.

예제 1 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main( )
{
  int a = 50; ❶
  int *b; ❷
  b = &a; ❸
  *b = *b+20; ❹
  printf("%d, %d", a, *b); ❺
}
```

❶ 정수형 변수 a를 선언하고 50으로 초기화한다.

❷ 정수형 변수가 저장된 곳의 주소를 기억할 포인터 변수 b를 선언한다.

❸ 정수형 변수 a의 주소를 포인터 변수 b에 기억시킨다. b에는 a의 주소가 저장된다.

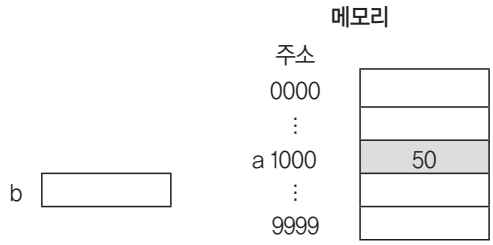
❹ b가 가리키는 곳의 값에 20을 더한다. b가 가리키는 곳이 a이므로 결국 a의 값도 바뀌는 것이다.

❺ 결과 70, 70

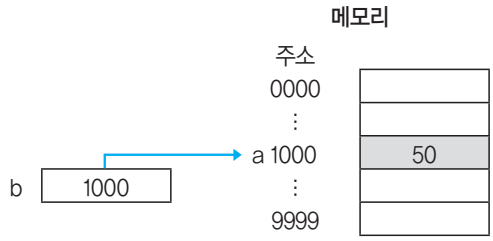
- ❷와 같이 선언할 때 *는 해당 변수가 포인터 변수라는 것을 의미한다.
- ❹, ❺와 같이 사용할 때 *를 붙이면 그 포인터 변수가 가리키는 곳의 값을 의미한다.

위 코드의 실행 과정에 따라 메모리의 변화를 그려보면 다음과 같다.

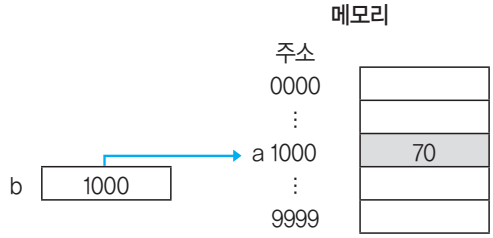
❶, ❷번 수행 : 주기억장치의 빈 공간 어딘가에 a라는 이름을 붙이고 그 곳에 50을 저장한다.



❸번 수행 : 변수 a의 주소가 b에 기억된다는 것은 b가 변수 a의 주소를 가리키고 있다는 의미이다.



❹번 수행 : b가 가리키는 곳의 값에 20을 더해 다시 b가 가리키는 곳에 저장한다. 그곳은 변수 a의 주소이므로 변수 a의 값도 저절로 변경되는 것이다.



핵심 262 Python의 기초

- 변수의 자료형에 대한 선언이 없다.
- 문장의 끝을 의미하는 세미콜론(;)을 사용할 필요가 없다.
- 변수에 연속하여 값을 저장하는 것이 가능하다.

  예 x, y, z = 10, 20, 30
- if나 for와 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다.
- 여백은 일반적으로 4칸 또는 한 개의 탭만큼 띄워야하며, 같은 수준의 코드들은 반드시 동일한 여백을 가져야 한다.

input( ) 함수

- input( ) 함수는 Python의 표준 입력 함수로, 키보드로 입력받아 변수에 저장하는 함수이다.

• 형식

변수 = input(출력문자)

• '출력문자'는 생략이 가능하며, '변수'는 사용자가 임의로 지정할 수 있다.
• 값을 입력하고 Enter를 누르면, 입력한 값이 '변수'에 저장된다.

예 a = input('입력하세요.') → 화면에 입력하세요.가 출력되고 그 뒤에서 커서가 깜빡거리며 입력을 기다린다. 키보드로 값을 입력하면 변수 a에 저장된다.

print( ) 함수

• 형식1

print(출력값1, 출력값2, …, sep = 분리문자, end = 종료문자)

• '출력값'에는 숫자, 문자, 문자열, 변수 등 다양한 값이나 식이 올 수 있다.
• 'sep'는 여러 값을 출력할 때 값과 값 사이를 구분하기 위해 출력하는 문자로, 생략할 경우 기본값은 공백 한 칸(' ')이다.
• 'end'는 맨 마지막에 표시할 문자로, 생략할 경우 기본값은 줄 나눔이다.

예 print(82, 24, sep = '–', end = ',') → 82와 24 사이에 분리문자 '–'가 출력되고, 마지막에 종료문자 ','가 출력된다.

결과 82–24,

슬라이스(Slice)

슬라이스는 문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라(slicing) 반환하는 기능이다.

• 형식

객체명[초기위치:최종위치]

'초기위치'에서 '최종위치'–1까지의 요소들을 가져온다.

객체명[초기위치:최종위치:증가값]

• '초기위치'에서 '최종위치'–1까지 '증가값'만큼 증가하면서 해당 위치의 요소들을 가져온다.
• '증가값'이 음수인 경우 '초기위치'에서 '최종위치'+1까지 '증가값' 만큼 감소하면서 해당 위치의 요소들을 가져온다.

• 슬라이스는 일부 인수를 생략하여 사용할 수 있다.

객체명[:] 또는 객체명[::]

객체의 모든 요소를 반환한다.

객체명[초기위치:]

객체의 '초기위치'에서 마지막 위치까지의 요소들을 반환한다.

객체명[:최종위치]

객체의 0번째 위치에서 '최종위치'–1까지의 요소들을 반환한다.

객체명[::증가값]

객체의 0번째 위치에서 마지막 위치까지 '증가값'만큼 증가하면서 해당 위치의 요소들을 반환한다.

예 a = ['a', 'b', 'c', 'd', 'e']일 때

a[1:3] → ['b', 'c']

a[0:5:2] → ['a', 'c', 'e']

a[3:] → ['d', 'e']

a[:3] → ['a', 'b', 'c']

a[::3] → ['a', 'd']

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 263 절차적 프로그래밍 언어

절차적 프로그래밍 언어의 개요

절차적 프로그래밍 언어는 일련의 처리 절차를 정해진 문법에 따라 순서대로 기술해 나가는 언어이다.

- 절차적 프로그래밍 언어는 프로그램이 실행되는 절차를 중요시 한다.
- 절차적 프로그래밍 언어는 데이터를 중심으로 프로시저를 구현하며, 프로그램 전체가 유기적으로 연결되어 있다.
- 절차적 프로그래밍 언어는 자연어에 가까운 단어와 문장으로 구성된다.
- 절차적 프로그래밍 언어는 과학 계산이나 하드웨어 제어에 주로 사용된다.

절차적 프로그래밍 언어의 장 · 단점

- 컴퓨터의 처리 구조와 유사하여 실행 속도가 빠르다.
- 같은 코드를 복사하지 않고 다른 위치에서 호출하여 사용할 수 있다.
- 모듈 구성이 용이하며, 구조적인 프로그래밍이 가능하다.
- 프로그램을 분석하기 어렵다.
- 유지 보수나 코드의 수정이 어렵다.

핵심 264 절차적 프로그래밍 언어의 종류

언어	특징
C	• 1972년 미국 벨 연구소의 데니스 리치에 의해 개발되었다. • 시스템 소프트웨어를 개발하기 편리하여 시스템 프로그래밍 언어로 널리 사용된다. • 자료의 주소를 조작할 수 있는 포인터를 제공한다. • 고급 프로그래밍 언어이면서 저급 프로그램 언어의 특징을 모두 갖췄다. • UNIX의 일부가 C 언어로 구현되었다. • 컴파일러 방식의 언어이다. • 이식성이 좋아 컴퓨터 기종에 관계없이 프로그램을 작성할 수 있다.
ALGOL	• 수치 계산이나 논리 연산을 위한 과학 기술 계산용 언어이다. • PASCAL과 C 언어의 모체가 되었다.
COBOL	• 사무 처리용 언어이다. • 영어 문장 형식으로 구성되어 있어 이해와 사용이 쉽다. • 4개의 DIVISION으로 구성되어 있다.
FORTRAN	• 과학 기술 계산용 언어이다. • 수학과 공학 분야의 공식이나 수식과 같은 형태로 프로그래밍 할 수 있다.

핵심 265 객체지향 프로그래밍 언어

객체지향 프로그래밍 언어의 개요

객체지향 프로그래밍 언어는 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체로 만들어, 기계적인 부품들을 조립하여 제품을 만들 듯이 소프트웨어를 개발할 때도 객체들을 조립해서 프로그램을 작성할 수 있도록 한 프로그래밍 기법이다.

- 프로시저보다는 명령과 데이터로 구성된 객체를 중심으로 하는 프로그래밍 기법으로, 한 프로그램을 다른 프로그램에서 이용할 수 있도록 한다.

객체지향 프로그래밍 언어의 장 · 단점

- 상속을 통한 재사용과 시스템의 확장이 용이하다.
- 코드의 재활용성이 높다.
- 자연적인 모델링에 의해 분석과 설계를 쉽고 효율적으로 할 수 있다.
- 사용자와 개발자 사이의 이해를 쉽게 해준다.
- 대형 프로그램의 작성이 용이하다.
- 소프트웨어 개발 및 유지보수가 용이하다.
- 프로그래밍 구현을 지원해 주는 정형화된 분석 및 설계 방법이 없다.
- 구현 시 처리 시간이 지연된다.

시험에 나오는 것만 공부한다! 시나공시리즈

객체지향 프로그래밍 언어의 종류

언어	특징
JAVA	• 분산 네트워크 환경에 적용이 가능하며, 멀티스레드 기능을 제공하므로 여러 작업을 동시에 처리할 수 있다. • 운영체제 및 하드웨어에 독립적이며, 이식성이 강하다. • 캡슐화가 가능하고 재사용성 높다.
C++	• C 언어에 객체지향 개념을 적용한 언어이다. • 모든 문제를 객체로 모델링하여 표현한다.
Smalltalk	• 1세대 객체지향 프로그래밍 언어 중 하나로 순수한 객체지향 프로그래밍 언어이다. • 최초로 GUI를 제공한 언어이다.

핵심 266 스크립트 언어

스크립트 언어(Script Language)의 개요

스크립트 언어는 HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로, 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 동작하게 하는 언어이다.

- 게시판 입력, 상품 검색, 회원 가입 등과 같은 데이터베이스 처리 작업을 수행하기 위해 주로 사용한다.
- 스크립트 언어는 클라이언트의 웹 브라우저에서 해석되어 실행되는 클라이언트용 스크립트 언어와 서버에서 해석되어 실행된 후 결과만 클라이언트로 보내는 서버용 스크립트 언어가 있다.
  - 서버용 스크립트 언어 : ASP, JSP, PHP, 파이썬
  - 클라이언트용 스크립트 언어 : 자바 스크립트(JAVA Script), VB 스크립트(Visual Basic Script)

스크립트 언어의 장 · 단점

- 컴파일 없이 바로 실행하므로 결과를 바로 확인할 수 있다.
- 배우고 코딩하기 쉽다.
- 개발 시간이 짧다.
- 소스 코드를 쉽고 빠르게 수정할 수 있다.
- 코드를 읽고 해석해야 하므로 실행 속도가 느리다.
- 런타임 오류가 많이 발생한다.

스크립트 언어의 종류

자바 스크립트 (JAVA Script)	• 웹 페이지의 동작을 제어하는 데 사용되는 클라이언트용 스크립트 언어로, 클래스가 존재하지 않으며 변수 선언도 필요 없다. • 서버에서 데이터를 전송할 때 아이디, 비밀번호, 수량 등의 입력 사항을 확인하기 위한 용도로 많이 사용된다.
VB 스크립트 (Visual Basic Script)	마이크로소프트 사에서 자바 스크립트에 대응하기 위해 제작한 언어로, Active X를 사용하여 마이크로소프트 사의 애플리케이션들을 컨트롤할 수 있다.
ASP(Active Server Page)	• 서버 측에서 동적으로 수행되는 페이지를 만들기 위한 언어로 마이크로 소프트 사에서 제작하였다. • Windows 계열에서만 수행 가능한 프로그래밍 언어이다.
JSP(Java Server Page)	JAVA로 만들어진 서버용 스크립트로, 다양한 운영체제에서 사용이 가능하다.
PHP (Professional Hypertext Preprocessor)	• 서버용 스크립트 언어로, Linux, Unix, Windows 운영체제에서 사용 가능하다. • C, Java 등과 문법이 유사하므로 배우기 쉬워 웹 페이지 제작에 많이 사용된다.
파이썬(Python)	객체지향 기능을 지원하는 대화형 인터프리터 언어로, 플랫폼에 독립적이고 문법이 간단하여 배우기 쉽다.
쉘 스크립트	• 유닉스/리눅스 계열의 쉘(Shell)에서 사용되는 명령어들의 조합으로 구성된 스크립트 언어이다. • 컴파일 단계가 없어 실행 속도가 빠르다. • 저장 시 확장자로 '.sh'가 붙는다. • 쉘의 종류 : Bash Shell, Bourne Shell, C Shell, Korn Shell 등 • 쉘 스크립트에서 사용되는 제어문 – 선택형 : if, case – 반복형 : for, while, until
Basic	절차지향 기능을 지원하는 대화형 인터프리터 언어로, 초보자도 쉽게 사용할 수 있는 문법 구조를 갖는다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 267 선언형 언어

선언형 언어

선언형 언어는 명령형 언어와 반대되는 개념의 언어로, 명령형 언어가 문제를 해결하기 위한 방법을 기술한다면 선언형 언어는 프로그램이 수행해야 하는 문제를 기술하는 언어이다.

- 선언형 언어는 목표를 명시하고 알고리즘은 명시하지 않는다.
- 선언형 언어에는 함수형 언어와 논리형 언어 등이 있다.

함수형 언어	• 수학적 함수를 조합하여 문제를 해결하는 언어로, 알려진 값을 함수에 적용하는 것을 기반으로 한다. • 적용형 언어라고도 한다. • 종류 : LISP
논리형 언어	• 기호 논리학에 기반을 둔 언어로, 논리 문장을 이용하여 프로그램을 표현하고 계산을 수행한다. • 선언적 언어라고도 한다. • 종류 : PROLOG

※ 명령형 언어

명령형 언어는 순차적인 명령 수행을 기본으로 하는 언어로, 문제를 처리하기 위한 방법에 초점을 두고 코드를 작성한다.

- 폰노이만 구조에 개념적인 기초를 두고 있다.
- 명령형 언어는 알고리즘을 명시하고 목표는 명시하지 않는다.
- 명령형 언어에는 절차적 언어와 객체지향 언어가 있다.
- 종류 : FORTRAN, COBOL, C, JAVA 등

선언형 프로그래밍 언어 종류

HTML	인터넷의 표준 문서인 하이퍼텍스트 문서를 만들기 위해 사용하는 언어로, 특별한 데이터 타입이 없는 단순한 텍스트이므로 호환성이 좋고 사용이 편리하다.
LISP	• 인공지능 분야에 사용되는 언어이다. • 기본 자료 구조가 연결 리스트 구조이며, 재귀(Recursion) 호출을 많이 사용한다.
PROLOG	논리학을 기초로 한 고급 언어로, 인공 지능 분야에서의 논리적인 추론이나 리스트 처리 등에 주로 사용된다.
XML	• 기존 HTML의 단점을 보완하여 웹에서 구조화된 폭넓고 다양한 문서들을 상호 교환할 수 있도록 설계된 언어이다. • HTML에 사용자가 새로운 태그(Tag)를 정의할 수 있으며, 문서의 내용과 이를 표현하는 방식이 독립적이다.
Haskell	• 함수형 프로그래밍 언어로 부작용(Side Effect)이 없다. • 코드가 간결하고 에러 발생 가능성이 낮다.

핵심 268 라이브러리

라이브러리의 개념

라이브러리는 프로그램을 효율적으로 개발할 수 있도록 자주 사용하는 함수나 데이터들을 미리 만들어 모아 놓은 집합체이다.

- 자주 사용하는 함수들의 반복적인 코드 작성을 피하기 위해 미리 만들어 놓은 것으로, 필요할 때는 언제든지 호출하여 사용할 수 있다.
- 라이브러리에는 표준 라이브러리와 외부 라이브러리가 있다.
  - 표준 라이브러리 : 프로그래밍 언어에 기본적으로 포함되어 있는 라이브러리로, 여러 종류의 모듈이나 패키지 형태이다.
  - 외부 라이브러리 : 개발자들이 필요한 기능들을 만들어 인터넷 등에 공유해 놓은 것으로, 외부 라이브러리를 다운받아 설치한 후 사용한다.

C언어의 대표적인 표준 라이브러리

C언어는 라이브러리를 헤더 파일로 제공하는데, 각 헤더 파일에는 응용 프로그램 개발에 필요한 함수들이 정리되어 있다.

- C언어에서 헤더 파일을 사용하려면 '#include 〈stdio.h〉'와 같이 include문을 이용해 선언한 후 사용해야 한다.
- 종류 : stdio.h, math.h, string.h, stdlib.h, time.h

JAVA의 대표적인 표준 라이브러리

JAVA는 라이브러리를 패키지에 포함하여 제공하는데, 각 패키지에는 JAVA 응용 프로그램 개발에 필요한 메소드들이 클래스로 정리되어 있다.

- JAVA에서 패키지를 사용하려면 'import java.util'과 같이 import문을 이용해 선언한 후 사용해야 한다.
- 종류 : java.lang, java.util, java.io, java.net, java.awt

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 269 예외 처리

프로그램의 정상적인 실행을 방해하는 조건이나 상태를 예외(Exception)라고 하며, 이러한 예외가 발생했을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴을 수행하도록 하는 것을 예외 처리(Exception Handling)라고 한다.

- 예외가 발생했을 때 일반적인 처리 루틴은 프로그램을 종료시키거나 로그를 남기도록 하는 것이다.
- C++, Ada, JAVA, 자바스크립트와 같은 언어에는 예외 처리 기능이 내장되어 있으며, 그 외의 언어에서는 필요한 경우 조건문을 이용해 예외 처리 루틴을 작성한다.
- 예외의 원인에는 컴퓨터 하드웨어 문제, 운영체제의 설정 실수, 라이브러리 손상, 사용자의 입력 실수, 받아들일 수 없는 연산, 할당하지 못하는 기억장치 접근 등 다양하다.

핵심 270 프로토타입(Prototype)

프로그래밍 언어에서 프로토타입이란 함수 원형(Function Prototype)이라는 의미로, 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것이다.

- 함수가 호출되기 전에 함수가 미리 정의되는 경우에는 프로토타입을 정의하지 않아도 된다.
- 프로토타입은 본문이 없다는 점을 제외하고 함수 정의와 형태가 동일하다.
- 프로토타입에 정의된 반환 형식은 함수 정의에 지정된 반환 형식과 반드시 일치해야 한다.

핵심 271 운영체제의 정의 및 목적

운영체제의 정의

운영체제(OS; Operating System)는 컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임이다.

- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공해준다.

운영체제의 목적

운영체제의 목적에는 처리 능력 향상, 사용 가능도 향상, 신뢰도 향상, 반환 시간 단축 등이 있다.

- 처리 능력, 반환 시간, 사용 가능도, 신뢰도는 운영체제의 성능을 평가하는 기준이 된다.

처리 능력 (Throughput)	일정 시간 내에 시스템이 처리하는 일의 양
반환 시간 (Turn Around Time)	시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
사용 가능도 (Availability)	시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
신뢰도(Reliability)	시스템이 주어진 문제를 정확하게 해결하는 정도

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 272 운영체제의 기능

- 프로세서(처리기, Processor), 기억장치(주기억장치, 보조기억장치), 입 · 출력장치, 파일 및 정보 등의 자원을 관리한다.
- 자원을 효율적으로 관리하기 위해 자원의 스케줄링 기능을 제공한다.
- 사용자와 시스템 간의 편리한 인터페이스를 제공한다.
- 시스템의 각종 하드웨어와 네트워크를 관리 · 제어한다.
- 데이터를 관리하고, 데이터 및 자원의 공유 기능을 제공한다.
- 시스템의 오류를 검사하고 복구한다.
- 자원 보호 기능을 제공한다.
- 입 · 출력에 대한 보조 기능을 제공한다.
- 가상 계산기 기능을 제공한다.

핵심 273 Windows

- Windows는 1990년대 마이크로소프트(Microsoft) 사가 개발한 운영체제이다.
- Windows의 주요 특징

그래픽 사용자 인터페이스 (GUI; Graphic User Interface)	키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행하는 방식을 말한다.
선점형 멀티태스킹 (Preemptive Multi-Tasking)	동시에 여러 개의 프로그램을 실행하는 멀티태스킹을 하면서 운영체제가 각 작업의 CPU 이용 시간을 제어하여 응용 프로그램 실행중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식을 말한다.
PnP (Plug and Play, 자동 감지 기능)	컴퓨터 시스템에 프린터나 사운드 카드 등의 하드웨어를 설치했을 때, 해당 하드웨어를 사용하는데 필요한 시스템 환경을 운영체제가 자동으로 구성해 주는 기능이다.
OLE (Object Linking and Embedding)	다른 여러 응용 프로그램에서 작성된 문자나 그림 등의 개체(Object)를 현재 작성 중인 문서에 자유롭게 연결(Linking)하거나 삽입(Embedding)하여 편집할 수 있게 하는 기능이다.
255자의 긴 파일명	• Windows에서는 파일 이름을 지정할 때 VFAT(Virtual File Allocation Table)를 이용하여 최대 255자까지 지정할 수 있다. • 파일 이름으로는 ₩ / : * ? " 〈 〉 | 를 제외한 모든 문자 및 공백을 사용할 수 있으며, 한글의 경우 127자까지 지정할 수 있다.
Single-User 시스템	컴퓨터 한 대를 한 사람만이 독점해서 사용한다.

핵심 274 UNIX의 개요 및 특징

UNIX는 1960년대 AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제이다.

- 시분할 시스템(Time Sharing System)을 위해 설계된 대화식 운영체제로, 소스가 공개된 개방형 시스템(Open System)이다.
- 대부분 C 언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.
- 크기가 작고 이해하기가 쉽다.
- 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)을 지원한다.
- 많은 네트워킹 기능을 제공하므로 통신망(Network) 관리용 운영체제로 적합하다.
- 트리 구조의 파일 시스템을 갖는다.
- 전문적인 프로그램 개발에 용이하다.
- 다양한 유틸리티 프로그램들이 존재한다.

※ 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)

- 다중 사용자(Multi-User)는 여러 사용자가 동시에 시스템을 사용하는 것이고, 다중 작업(Multi-Tasking)은 여러 개의 작업이나 프로그램을 동시에 수행하는 것을 의미한다.
- 하나 이상의 작업을 백그라운드에서 수행하므로 여러 작업을 동시에 처리할 수 있다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 275 UNIX 시스템의 구성

커널(Kernel)

- UNIX의 가장 핵심적인 부분이다.
- 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행된다.
- 하드웨어를 보호하고, 프로그램과 하드웨어 간의 인터페이스 역할을 담당한다.
- 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다.

쉘(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다.
- 시스템과 사용자 간의 인터페이스를 담당한다.
- DOS의 COMMAND.COM과 같은 기능을 수행한다.
- 주기억장치에 상주하지 않고, 명령어가 포함된 파일 형태로 존재하며 보조 기억장치에서 교체 처리가 가능하다.
- 파이프라인 기능을 지원하고 입 · 출력 재지정을 통해 출력과 입력의 방향을 변경할 수 있다.
- 공용 Shell(Bourne Shell, C Shell, Korn Shell)이나 사용자 자신이 만든 Shell을 사용할 수 있다.

Utility Program

- 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용한다.
- DOS에서의 외부 명령어에 해당된다.
- 유틸리티 프로그램에는 에디터, 컴파일러, 인터프리터, 디버거 등이 있다.

핵심 276 기억장치 관리 전략

기억장치의 관리 전략은 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 적재 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용하기 위한 것으로 반입(Fetch) 전략, 배치(Placement) 전략, 교체(Replacement) 전략이 있다.

반입(Fetch) 전략

반입 전략은 보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략이다.

요구 반입 (Demand Fetch)	실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법이다.
예상 반입 (Anticipatory Fetch)	실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법이다.

배치(Placement) 전략

배치 전략은 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략이다.

최초 적합 (First Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법이다.
최적 적합 (Best Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법이다.
최악 적합 (Worst Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법이다.

교체(Replacement) 전략

교체 전략은 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략이다.

- 교체 전략에는 FIFO, OPT, LRU, LFU, NUR, SCR 등이 있다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 277 주기억장치 할당의 개념

주기억장치 할당 기법은 프로그램이나 데이터를 실행시키기 위해 주기억장치에 어떻게 할당할 것인지에 대한 내용이며, 연속 할당 기법과 분산 할당 기법으로 분류할 수 있다.

연속 할당 기법	프로그램을 주기억장치에 연속으로 할당하는 기법으로, 단일 분할 할당 기법과 다중 분할 할당 기법이 있다. • 단일 분할 할당 기법 : 오버레이, 스와핑 • 다중 분할 할당 기법 : 고정 분할 할당 기법, 동적 분할 할당 기법
분산 할당 기법	프로그램을 특정 단위의 조각으로 나누어 주기억장치 내에 분산하여 할당하는 기법으로, 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.

핵심 278 다중 분할 할당 기법

고정 분할 할당(Multiple contiguous Fixed parTition allocation, MFT) 기법 = 정적 할당(Static Allocation) 기법

고정 분할 할당은 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법이다.

- 프로그램을 실행하려면 프로그램 전체가 주기억장치에 위치해야 한다.
- 프로그램이 분할된 영역보다 커서 영역 안에 들어갈 수 없는 경우가 발생할 수 있다.
- 일정한 크기의 분할 영역에 다양한 크기의 프로그램이 할당되므로 내부 단편화 및 외부 단편화가 발생하여 주기억장치의 낭비가 많다.
- 실행할 프로그램의 크기를 미리 알고 있어야 한다.
- 다중 프로그래밍을 위해 사용되었으나 현재는 사용되지 않는다.

가변 분할 할당(Multiple contiguous Variable parTition allocation, MVT) 기법 = 동적 할당(Dynamic Allocation) 기법

고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할해 놓는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법이다.

- 주기억장치를 효율적으로 사용할 수 있으며, 다중 프로그래밍의 정도를 높일 수 있다.
- 고정 분할 할당 기법에 비해 실행될 프로세스 크기에 대한 제약이 적다.
- 단편화를 상당 부분 해결할 수 있으나 영역과 영역 사이에 단편화가 발생될 수 있다.

핵심 279 가상기억장치

가상기억장치는 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다.

- 주기억장치의 용량보다 큰 프로그램을 실행하기 위해 사용한다.
- 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- 가상기억장치에 저장된 프로그램을 실행하려면 가상기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환 작업이 필요하다.
- 블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있다.
- 가상기억장치의 일반적인 구현 방법에는 블록의 종류에 따라 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.

페이징(Paging) 기법

- 페이징 기법은 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램(페이지)을 동일하게 나눠진 주기억장치의 영역(페이지 프레임)에 적재시켜 실행하는 기법이다.

시험에 나오는 것만 공부한다! 시나공시리즈

- 프로그램을 일정한 크기로 나눈 단위를 페이지(Page)라고 하고, 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임(Page Frame)이라고 한다.
- 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있다.
- 주소 변환을 위해서 페이지의 위치 정보를 가지고 있는 페이지 맵 테이블(Page Map Table)이 필요하다.

세그먼테이션(Segmentation) 기법

- 세그먼테이션 기법은 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법이다.
- 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트(Segment)라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다.
- 주소 변환을 위해서 세그먼트가 존재하는 위치 정보를 가지고 있는 세그먼트 맵 테이블(Segment Map Table)이 필요하다.
- 내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있다.

핵심 280 페이지 교체 알고리즘

페이지 교체 알고리즘은 페이지 부재(Page Fault)가 발생했을 때 가상기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데, 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할 것인지를 결정하는 기법이다.

OPT (OPTimal replacement, 최적 교체)	• 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법이다. • 벨레이디(Belady)가 제안한 것으로, 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘이다.
FIFO (First In First Out)	• 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이다. • 이해하기 쉽고, 프로그래밍 및 설계가 간단하다.
LRU (Least Recently Used)	• 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이다. • 각 페이지마다 계수기(Counter)나 스택(Stack)을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체한다.
LFU (Least Frequently Used)	• 사용 빈도가 가장 적은 페이지를 교체하는 기법이다. • 활발하게 사용되는 페이지는 사용 횟수가 많아 교체되지 않고 사용된다.
SCR (Second Chance Replacement, 2차 기회 교체)	가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법이다.
NUR (Not Used Recently)	• LRU와 비슷한 알고리즘으로, 최근에 사용하지 않은 페이지를 교체하는 기법이다. • 최근에 사용되지 않은 페이지는 향후에도 사용되지 않을 가능성이 높다는 것을 전제로, LRU에서 나타나는 시간적인 오버헤드를 줄일 수 있다. • 최근의 사용 여부를 확인하기 위해서 각 페이지마다 두 개의 비트, 즉 참조 비트(Reference Bit)와 변형 비트(Modified Bit, Dirty Bit)가 사용된다.

핵심 281 페이지 크기

페이지 크기가 작을 경우

- 페이지 단편화가 감소되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 줄어든다.
- 불필요한 내용이 주기억장치에 적재될 확률이 적으므로 효율적인 워킹 셋을 유지할 수 있다.
- Locality에 더 일치할 수 있기 때문에 기억장치 효율이 높아진다.
- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 매핑 속도가 늦어진다.
- 디스크 접근 횟수가 많아져서 전체적인 입 · 출력 시간은 늘어난다.

페이지 크기가 클 경우

- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고, 매핑 속도가 빨라진다.
- 디스크 접근 횟수가 줄어들어 전체적인 입 · 출력의 효율성이 증가된다.
- 페이지 단편화가 증가되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어난다.
- 프로세스(프로그램) 수행에 불필요한 내용까지도 주기억장치에 적재될 수 있다.

핵심 282 Locality

Locality(국부성, 지역성, 구역성, 국소성)는 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론이다.

- 스래싱을 방지하기 위한 워킹 셋 이론의 기반이 되었다.
- Locality의 종류에는 시간 구역성(Temporal Locality)과 공간 구역성(Spatial Locality)이 있다.

시간 구역성(Temporal Locality)

- 시간 구역성은 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상이다.
- 한 번 참조한 페이지는 가까운 시간 내에 계속 참조할 가능성이 높음을 의미한다.
- **시간 구역성이 이루어지는 기억 장소** : Loop(반복, 순환), 스택(Stack), 부 프로그램(Sub Routine), Counting(1씩 증감), 집계(Totaling)에 사용되는 변수(기억장소)

공간 구역성(Spatial Locality)

- 공간 구역성은 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상이다.
- 어느 하나의 페이지를 참조하면 그 근처의 페이지를 계속 참조할 가능성이 높음을 의미한다.
- **공간 구역성이 이루어지는 기억장소** : 배열 순회(Array Traversal, 배열 순례), 순차적 코드의 실행, 프로그래머들이 관련된 변수(데이터를 저장할 기억장소)들을 서로 근처에 선언하여 할당되는 기억장소, 같은 영역에 있는 변수를 참조할 때 사용

핵심 283 워킹 셋(Working Set)

워킹 셋은 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합이다.

- 데닝(Denning)이 제안한 프로그램의 움직임에 대한 모델로, 프로그램의 Locality 특징을 이용한다.
- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정된다.
- 시간이 지남에 따라 자주 참조하는 페이지들의 집합이 변화하기 때문에 워킹 셋은 시간에 따라 변경된다.

핵심 284 스래싱(Thrashing)

스래싱은 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상이다.

- 다중 프로그래밍 시스템이나 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정중 자주 페이지 부재가 발생함으로써 나타나는 현상으로, 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU의 이용률은 어느 특정 시점까지는 높아지지만, 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고, CPU의 이용률은 급격히 감소하게 된다.
- 스래싱 현상 방지 방법
  - 다중 프로그래밍의 정도를 적정 수준으로 유지한다.
  - 페이지 부재 빈도(Page Fault Frequency)를 조절하여 사용한다.
  - 워킹 셋을 유지한다.
  - 부족한 자원을 증설하고, 일부 프로세스를 중단시킨다.
  - CPU 성능에 대한 자료의 지속적 관리 및 분석으로 임계치를 예상하여 운영한다.

※ 페이지 부재(Page Fault)

- 페이지 부재는 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상이며, 페이지 부재 빈도(PFF; Page Fault Frequency)는 페이지 부재가 일어나는 횟수를 의미한다.
- **페이지 부재 빈도 방식** : 페이지 부재율(Page Fault Rate)에 따라 주기억장치에 있는 페이지 프레임의 수를 늘리거나 줄여 페이지 부재율을 적정 수준으로 유지하는 방식

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 285 프로세스(Process)의 정의

프로세스는 일반적으로 프로세서(처리기, CPU)에 의해 처리되는 사용자 프로그램, 시스템 프로그램, 즉 실행중인 프로그램을 의미하며, 작업(Job), 태스크(Task)라고도 한다.

- 프로세스는 다음과 같이 여러 형태로 정의할 수 있다.
  - PCB를 가진 프로그램
  - 실기억장치에 저장된 프로그램
  - 프로세서가 할당되는 실체로서, 디스패치가 가능한 단위
  - 프로시저가 활동중인 것
  - 비동기적 행위를 일으키는 주체
  - 지정된 결과를 얻기 위한 일련의 계통적 동작
  - 목적 또는 결과에 따라 발생되는 사건들의 과정
  - 운영체제가 관리하는 실행 단위

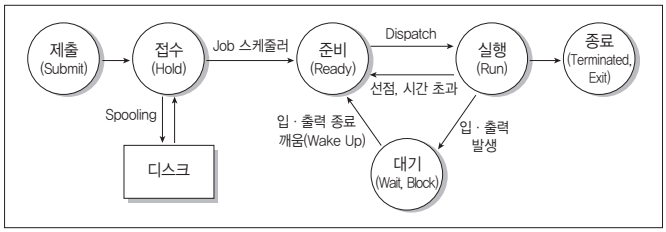
핵심 286 PCB

PCB(Process Control Block, 프로세스 제어 블록)는 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳으로, Task Control Block 또는 Job Control Block이라고도 한다.

- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고, 프로세스가 완료되면 PCB는 제거된다.
- PCB에 저장되어 있는 정보
  - 프로세스의 현재 상태
  - 포인터
    - ▶ 부모 프로세스에 대한 포인터
    - ▶ 자식 프로세스에 대한 포인터
    - ▶ 프로세스가 위치한 메모리에 대한 포인터
    - ▶ 할당된 자원에 대한 포인터
  - 프로세스 고유 식별자
  - 스케줄링 및 프로세스의 우선순위
  - CPU 레지스터 정보
  - 주기억장치 관리 정보
  - 입 · 출력 상태 정보
  - 계정 정보

핵심 287 프로세스 상태 전이

프로세스 상태 전이는 프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것을 의미하며, 프로세스의 상태를 다음과 같이 상태 전이도로 표시할 수 있다.



- 프로세스의 상태는 제출, 접수, 준비, 실행, 대기 상태로 나눌 수 있으며, 이 중 주요 세 가지 상태는 준비, 실행, 대기 상태이다.
- **제출(Submit)** : 작업을 처리하기 위해 사용자가 작업을 시스템에 제출한 상태이다.
- **접수(Hold)** : 제출된 작업이 스풀 공간인 디스크의 할당 위치에 저장된 상태이다.
- **준비(Ready)** : 프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태이다.
- **실행(Run)** : 준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태이다.
- **대기(Wait), 보류, 블록(Block)** : 프로세스에 입 · 출력 처리가 필요하면 현재 실행 중인 프로세스가 중단되고, 입 · 출력 처리가 완료될 때까지 대기하고 있는 상태이다.
- **종료(Terminated, Exit)** : 프로세스의 실행이 끝나고 프로세스 할당이 해제된 상태이다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 288 프로세스 상태 전이 관련 용어

Dispatch	준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정이다.
Wake Up	입·출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이 되는 과정이다.
Spooling	입·출력장치의 공유 및 상대적으로 느린 입·출력장치의 처리 속도를 보완하고 다중 프로그래밍 시스템의 성능을 향상시키기 위해 입·출력할 데이터를 직접 입·출력장치에 보내지 않고 나중에 한꺼번에 입·출력하기 위해 디스크에 저장하는 과정이다.
교통량 제어기 (Traffic Controller)	프로세스의 상태에 대한 조사와 통보를 담당한다.

핵심 289 스레드(Thread)

스레드는 프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위이다.

- 하나의 프로세스에 하나의 스레드가 존재하는 경우에는 단일 스레드, 하나 이상의 스레드가 존재하는 경우에는 다중 스레드라고 한다.
- 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.
- 스레드 기반 시스템에서 스레드는 독립적인 스케줄링의 최소 단위로서 프로세스의 역할을 담당한다.
- 동일 프로세스 환경에서 서로 독립적인 다중 수행이 가능하다.
- 스레드의 분류

사용자 수준의 스레드	• 사용자가 만든 라이브러리를 사용하여 스레드를 운용한다. • 속도는 빠르지만 구현이 어렵다.
커널 수준의 스레드	• 운영체제의 커널에 의해 스레드를 운용한다. • 구현이 쉽지만 속도가 느리다

- 스레드 사용의 장점
  - 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.
  - 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.
  - 응용 프로그램의 응답 시간(Response Time)을 단축시킬 수 있다.
  - 실행 환경을 공유시켜 기억장소의 낭비가 줄어든다.
  - 프로세스들 간의 통신이 향상된다.
  - 스레드는 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.

핵심 290 스케줄링 / 문맥 교환

스케줄링의 정의

스케줄링(Scheduling)은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미한다.

스케줄링의 목적

- 공정성
- 처리율(량) 증가
- CPU 이용률 증가
- 우선순위 제도
- 오버헤드 최소화
- 응답 시간(Response Time, 반응 시간) 최소화
- 반환 시간(Turn Around Time) 최소화
- 대기 시간 최소화
- 균형 있는 자원의 사용
- 무한 연기 회피

문맥 교환

- 문맥 교환은 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것이다.
- 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장한다.
- 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 한다.

※ 스케줄링의 성능 평가 기준 : 스케줄링의 목적 중 CPU 이용률, 처리율, 반환 시간, 대기 시간, 응답 시간은 여러 종류의 스케줄링 성능을 비교하는 기준이 됨

핵심 291 프로세스 스케줄링의 기법

비선점(Non-Preemptive) 스케줄링

- 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법이다.
- 프로세스가 CPU를 할당받으면 해당 프로세스가 완료될 때까지 CPU를 사용한다.
- 모든 프로세스에 대한 요구를 공정하게 처리할 수 있다.
- 프로세스 응답 시간의 예측이 용이하며, 일괄 처리 방식에 적합하다.
- 중요한 작업(짧은 작업)이 중요하지 않은 작업(긴 작업)을 기다리는 경우가 발생할 수 있다.
- 비선점 스케줄링의 종류에는 FCFS, SJF, 우선순위, HRN, 기한부 등의 알고리즘이 있다.

선점(Preemptive) 스케줄링

- 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법이다.
- 우선순위가 높은 프로세스를 빠르게 처리할 수 있다.
- 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템에 사용된다.
- 많은 오버헤드(Overhead)를 초래한다.
- 선점이 가능하도록 일정 시간 배당에 대한 인터럽트용 타이머 클록(Clock)이 필요하다.
- 선점 스케줄링의 종류에는 Round Robin, SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등의 알고리즘이 있다.

핵심 292 주요 스케줄링 알고리즘

FCFS(First Come First Service, 선입 선출) = FIFO(First In First Out)

FCFS는 준비상태 큐(대기 큐, 준비 완료 리스트, 작업준비 큐, 스케줄링 큐)에 도착한 순서에 따라 차례로 CPU를 할당하는 기법으로, 가장 간단한 알고리즘이다.

- 먼저 도착한 것이 먼저 처리되어 공평성은 유지되지만 짧은 작업이 긴 작업을, 중요한 작업이 중요하지 않은 작업을 기다리게 된다.

SJF(Shortest Job First, 단기 작업 우선)

SJF는 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법이다.

- 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘이다.

HRN(Hightest Response-ratio Next)

실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법이다.

- 우선순위 계산 공식을 이용하여 서비스(실행) 시간이 짧은 프로세스나 대기 시간이 긴 프로세스에게 우선순위를 주어 CPU를 할당한다.
- 서비스 실행 시간이 짧거나 대기 시간이 긴 프로세스일 경우 우선순위가 높아진다.
- 우선순위를 계산하여 그 숫자가 가장 높은 것부터 낮은 순으로 우선순위가 부여된다.
- 우선순위 계산식

$$\text{우선순위 계산식} = \frac{\text{대기 시간+서비스 시간}}{\text{서비스 시간}}$$

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 293 Windows의 주요 환경 변수

Windows에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞뒤에 '%'를 입력해야 한다.

- Windows에서 set을 입력하면 모든 환경 변수와 값을 출력한다.

환경 변수	용도
%ALLUSERPROFILE%	모든 사용자의 프로필이 저장된 폴더
%APPDATA%	설치된 프로그램의 필요 데이터가 저장된 폴더
%COMSPEC%	기본 명령 프롬프트로 사용할 프로그램명
%HOMEDRIVE%	로그인한 계정의 정보가 저장된 드라이브
%HOMEPATH%	로그인한 계정의 기본 폴더
%LOGONSERVER%	로그인한 계정이 접속한 서버명
%PATH%	실행 파일을 찾는 경로
%PATHEXT%	cmd에서 실행할 수 있는 파일의 확장자 목록
%PROGRAMFILES%	기본 프로그램의 설치 폴더
%SYSTEMDRIVE%	Windows가 부팅된 드라이브
%SYSTEMROOT%	부팅된 운영체제가 들어 있는 폴더
%TEMP% 또는 %TMP%	임시 파일이 저장되는 폴더
%USERDOMAIN%	로그인한 시스템의 도메인명
%USERNAME%	로그인한 계정 이름
%USERPROFILE%	로그인한 유저의 프로필이 저장된 폴더명

핵심 294 UNIX / LINUX의 주요 환경 변수

UNIX나 LINUX에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞에 '$'를 입력해야 한다.

- UINIX나 LINUX에서는 set, env, printenv, setenv 중 하나를 입력하면 모든 환경 변수와 값을 표시한다.

환경 변수	용도
$DISPLAY	현재 X 윈도 디스플레이 위치
$HOME	사용자의 홈 디렉터리
$LANG	프로그램 사용 시 기본적으로 지원되는 언어
$MAIL	메일을 보관하는 경로
$PATH	실행 파일을 찾는 경로
$PS1	쉘 프롬프트 정보
$PWD	현재 작업하는 디렉터리
$TERM	로긴 터미널 타입
$USER	사용자의 이름

핵심 295 Windows 기본 명령어

명령어	기능
DIR	파일 목록을 표시한다.
COPY	파일을 복사한다.
TYPE	파일의 내용을 표시한다.
REN	파일의 이름을 변경한다.
DEL	파일을 삭제한다.
MD	디렉터리를 생성한다.
CD	디렉터리의 위치를 변경한다.
CLS	화면의 내용을 지운다.
ATTRIB	파일의 속성을 변경한다.
FIND	파일을 찾는다.
CHKDSK	디스크 상태를 점검한다.
FORMAT	디스크 표면을 트랙과 섹터로 나누어 초기화한다.
MOVE	파일을 이동한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 296 UNIX / LINUX 기본 명령어

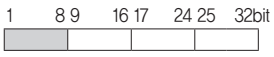
명령어	기능
cat	파일 내용을 화면에 표시한다.
chdir	현재 사용할 디렉터리의 위치를 변경한다.
chmod	파일의 보호 모드를 설정하여 파일의 사용 허가를 지정한다.
chown	소유자를 변경한다.
cp	파일을 복사한다.
exec	새로운 프로세스를 수행한다.
find	파일을 찾는다.
fork	새로운 프로세스를 생성한다(하위 프로세스 호출, 프로세스 복제 명령).
fsck	파일 시스템을 검사하고 보수한다.
getpid	자신의 프로세스 아이디를 얻는다.
getppid	부모 프로세스 아이디를 얻는다.
ls	현재 디렉터리 내의 파일 목록을 확인한다.
mount/ unmount	파일 시스템을 마운팅한다/마운팅 해제한다.
rm	파일을 삭제한다.
wait	fork 후 exec에 의해 실행되는 프로세스의 상위 프로세스가 하위 프로세스 종료 등의 event를 기다린다.

핵심 297 인터넷

IP 주소(Internet Protocol Address)

IP 주소는 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소이다.

- 숫자로 8비트씩 4부분, 총 32비트로 구성되어 있다.
- IP 주소는 네트워크 부분의 길이에 따라 다음과 같이 A 클래스에서 E 클래스까지 총 5단계로 구성되어 있다.

A Class	국가나 대형 통신망에 사용(0~127로 시작) 2^{24} = 16,777,216개의 호스트 사용 가능	
B Class	중대형 통신망에 사용(128~191로 시작) 2^{16} = 65,536개의 호스트 사용 가능	
C Class	소규모 통신망에 사용(192~223으로 시작) 2^{8} = 256개의 호스트 사용 가능	
D Class	멀티캐스트용으로 사용(224~239로 시작)	네트워크 부분 호스트 부분
E Class	실험적 주소이며 공용되지 않음	

서브네팅(Subnetting)

서브네팅은 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용하는 것을 말한다.

- 4바이트의 IP 주소 중 네트워크 주소와 호스트 주소를 구분하기 위한 비트를 서브넷 마스크(Subnet Mask)라고 하며, 이를 변경하여 네트워크 주소를 여러 개로 분할하여 사용한다.
- 서브넷 마스크는 각 클래스마다 다르게 사용된다.

IPv6(Internet Protocol version 6)

IPv6은 현재 사용하고 있는 IP 주소 체계인 IPv4의 주소 부족 문제를 해결하기 위해 개발되었다.

- 128비트의 긴 주소를 사용하여 주소 부족 문제를 해결할 수 있으며, IPv4에 비해 자료 전송 속도가 빠르다.
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제를 해결할 수 있다.
- IPv4와 호환성이 뛰어나다.
- 주소의 확장성, 융통성, 연동성이 뛰어나며, 실시간 흐름 제어로 향상된 멀티미디어 기능을 지원한다.
- 패킷 크기를 확장할 수 있으므로 패킷 크기에 제한이 없다.

IPv6의 구성

- 16비트씩 8부분, 총 128비트로 구성되어 있다.
- 각 부분을 16진수로 표현하고, 콜론(:)으로 구분한다.
- IPv6은 다음과 같이 세 가지 주소 체계로 나누어진다.

유니캐스트 (Unicast)	단일 송신자와 단일 수신자 간의 통신(1 대 1 통신에 사용)
멀티캐스트 (Multicast)	단일 송신자와 다중 수신자 간의 통신(1 대 다 통신에 사용)
애니캐스트 (Anycast)	단일 송신자와 가장 가까이 있는 단일 수신자 간의 통신(1 대 1 통신에 사용)

도메인 네임(Domain Name)

도메인 네임은 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것이다.

- 호스트 컴퓨터 이름, 소속 기관 이름, 소속 기관의 종류, 소속 국가명 순으로 구성되며, 왼쪽에서 오른쪽으로 갈수록 상위 도메인을 의미한다.

핵심 298 OSI 참조 모델

- 다른 시스템 간의 원활한 통신을 위해 ISO(국제표준화기구)에서 제안한 통신 규약(Protocol)이다.
- OSI 7계층은 1~3 계층을 하위 계층, 4~7 계층을 상위 계층이라고 한다.
  - 하위 계층 : 물리 계층 → 데이터 링크 계층 → 네트워크 계층
  - 상위 계층 : 전송 계층 → 세션 계층 → 표현 계층 → 응용 계층

물리 계층 (Physical Layer)	전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의한다.
데이터 링크 계층 (Data Link Layer)	• 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정과 유지 및 종료를 담당한다. • 송신 측과 수신 측의 속도 차이를 해결하기 위한 흐름 제어 기능을 한다. • 프레임의 시작과 끝을 구분하기 위한 프레임의 동기화 기능을 한다. • 오류의 검출과 회복을 위한 오류 제어 기능을 한다.
네트워크 계층 (Network Layer, 망 계층)	• 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 한다. • 네트워크 연결을 설정, 유지, 해제하는 기능을 한다. • 경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 수행한다.
전송 계층 (Transport Layer)	• 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 한다. • 종단 시스템(End-to-End) 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능을 한다. • 주소 설정, 다중화(분할 및 재조립), 오류 제어, 흐름 제어를 수행한다.
세션 계층 (Session Layer)	• 송 · 수신 측 간의 관련성을 유지하고 대화 제어를 담당한다. • 대화(회화) 구성 및 동기 제어, 데이터 교환 관리 기능을 한다.
표현 계층 (Presentation Layer)	• 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능을 한다. • 서로 다른 데이터 표현 형태를 갖는 시스템 간의 상호 접속을 위해 필요한 계층이다. • 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식(포맷) 변환, 문맥 관리 기능을 한다.
응용 계층 (Application Layer)	사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 서비스를 제공한다.

핵심 299 네트워크 관련 장비

네트워크 인터페이스 카드 (NIC; Network Interface Card)	컴퓨터와 컴퓨터 또는 컴퓨터와 네트워크를 연결하는 장치로, 정보 전송 시 정보가 케이블을 통해 전송될 수 있도록 정보 형태를 변경한다.
허브(Hub)	• 한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치로, 각 회선을 통합적으로 관리하며, 신호 증폭 기능을 하는 리피터의 역할도 포함한다. • 허브의 종류에는 더미 허브, 스위칭 허브가 있다.
리피터(Repeater)	전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행한다.
브리지(Bridge)	• LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행한다. • 네트워크를 분산적으로 구성할 수 있어 보안성을 높일 수 있다.

스위치(Switch)	• 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치이다. • 하드웨어를 기반으로 처리하므로 전송 속도가 빠르다.
라우터(Router)	브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가된 것으로, 서로 다른 LAN이나 LAN과 WAN의 연결도 수행한다.
게이트웨이(Gateway)	• 전 계층(1~7계층)의 프로토콜 구조가 다른 네트워크의 연결을 수행한다. • LAN에서 다른 네트워크에 데이터를 보내거나 다른 네트워크로부터 데이터를 받아들이는 출입구 역할을 한다.

핵심 300 프로토콜의 개념

- 프로토콜은 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약이다.
- 프로토콜의 기본 요소
  - 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
  - 의미(Semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
  - 시간(Timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정
- 프로토콜의 기능

단편화와 재결합	송신 측에서 전송할 데이터를 전송에 알맞은 일정 크기의 작은 블록으로 자르는 작업을 단편화(Fragmentation)라 하고, 수신 측에서 단편화된 블록을 원래의 데이터로 모으는 것을 재결합(Reassembly)이라 한다.
캡슐화	캡슐화(Encapsulation)는 단편화된 데이터에 송·수신지 주소, 오류 검출 코드, 프로토콜 기능을 구현하기 위한 프로토콜 제어 정보 등의 정보를 부가하는 것으로, 요약화라고도 한다.
흐름 제어	• 흐름 제어(Flow Control)는 수신 측의 처리 능력에 따라 송신 측에서 송신하는 데이터의 전송량이나 전송 속도를 조절하는 기능이다. • 정지-대기(Stop-and-Wait), 슬라이딩 윈도우(Sliding Window) 방식을 이용한다.
오류 제어	오류 제어(Error Control)는 전송중에 발생하는 오류를 검출하고 정정하여 데이터나 제어 정보의 파손에 대비하는 기능이다.
동기화	동기화(Synchronization)는 송·수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능이다.
순서 제어	순서 제어(Sequencing)는 전송되는 데이터 블록(PDU)에 전송 순서를 부여하는 기능으로, 연결 위주의 데이터 전송 방식에만 사용된다.
주소 지정	주소 지정(Addressing)은 데이터가 목적지까지 정확하게 전송될 수 있도록 목적지 이름, 주소, 경로를 부여하는 기능이다.
다중화	다중화(Multiplexing)는 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능이다.
경로 제어	경로 제어(Routing)는 송·수신 측 간의 송신 경로 중에서 최적의 패킷 교환 경로를 설정하는 기능이다.
전송 서비스	전송하려는 데이터가 사용하도록 하는 별도의 부가 서비스이다.

핵심 301 TCP/IP

- TCP/IP는 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜이다.
- TCP/IP는 TCP 프로토콜과 IP 프로토콜이 결합된 것을 의미한다.
- TCP/IP는 응용 계층, 전송 계층, 인터넷 계층, 네트워크 액세스 계층으로 이루어져 있다.
- 응용 계층의 주요 프로토콜

FTP (File Transfer Protocol)	컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
SMTP(Simple Mail Transfer Protocol)	전자 우편을 교환하는 서비스
TELNET	• 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스 • 프로그램을 실행하는 등 시스템 관리 작업을 할 수 있는 가상의 터미널(Virtual Terminal) 기능을 수행

시험에 나오는 것만 공부한다! 시나공시리즈

SNMP (Simple Network Management Protocol)	TCP/IP의 네트워크 관리 프로토콜로, 라우터나 허브 등 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는 데 사용되는 표준 통신 규약
DNS (Domain Name System)	도메인 네임을 IP 주소로 매핑(Mapping)하는 시스템
HTTP(HyperText Transfer Protocol)	월드 와이드 웹(WWW)에서 HTML 문서를 송수신 하기 위한 표준 프로토콜

• 전송 계층의 주요 프로토콜

TCP (Transmission Control Protocol)	• 양방향 연결(Full Duplex Connection)형 서비스를 제공한다. • 스트림 위주의 전달(패킷 단위)을 한다. • 신뢰성 있는 경로를 확립하고 메시지 전송을 감독한다. • 순서 제어, 오류 제어, 흐름 제어 기능을 한다. • TCP 프로토콜의 헤더는 기본적으로 20Byte에서 60Byte까지 사용할 수 있는데, 선택적으로 40Byte를 더 추가할 수 있으므로 최대 100Byte까지 크기를 확장할 수 있다.
UDP (User Datagram Protocol)	• 데이터 전송 전에 연결을 설정하지 않는 비연결형 서비스를 제공한다. • TCP에 비해 상대적으로 단순한 헤더 구조를 가지므로, 오버헤드가 적고, 흐름제어나 순서 제어가 없어 전송 속도가 빠르다. • 실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에서 사용된다.
RTCP (Real-Time Control Protocol)	• RTP(Real-time Transport Protocol) 패킷의 전송 품질을 제어하기 위한 제어 프로토콜이다. • 세션(Session)에 참여한 각 참여자들에게 주기적으로 제어 정보를 전송한다.

• 인터넷 계층의 주요 프로토콜

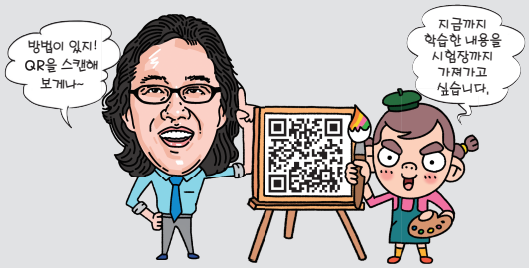
IP(Internet Protocol)	• 전송할 데이터에 주소를 지정하고, 경로를 설정하는 기능을 한다. • 비연결형인 데이터그램 방식을 사용하는 것으로 신뢰성이 보장되지 않는다.
ICMP (Internet Control Message Protocol, 인터넷 제어 메시지 프로토콜)	IP와 조합하여 통신중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리하는 역할을 하며, 헤더는 8Byte로 구성된다.
IGMP(Internet Group Management Protocol, 인터넷 그룹 관리 프로토콜)	멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용된다.
ARP (Address Resolution Protocol, 주소 분석 프로토콜)	호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소(MAC Address)로 바꾼다.
RARP (Reverse Address Resolution Protocol)	ARP와 반대로 물리적 주소를 IP 주소로 변환하는 기능을 한다.

• 네트워크 액세스 계층의 주요 프로토콜

Ethernet (IEEE 802.3)	CSMA/CD 방식의 LAN
IEEE 802	LAN을 위한 표준 프로토콜
HDLC	비트 위주의 데이터 링크 제어 프로토콜
X.25	패킷 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜
RS-232C	공중 전화 교환망(PSTN)을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요? 까먹기 전에 다시 한 번 복습하고 싶다고요? 지금까지 공부한 내용을 안전하게 시험장까지 가져가는 완벽한 방법이 있습니다. 지금 당장 QR 코드를 스캔해 보세요.



www.membox.co.kr을 직접 입력해도 접속할 수 있습니다.

시험에 나오는 것만 공부한다! 시나공시리즈

5 과목 정보시스템 구축 관리

핵심 302 구조적 방법론

구조적 방법론은 정형화된 분석 절차에 따라 사용자 요구사항을 파악하여 문서화하는 처리(Precess) 중심의 방법론이다.

- 1960년대까지 가장 많이 적용되었던 소프트웨어 개발 방법론이다.
- 쉬운 이해 및 검증이 가능한 프로그램 코드를 생성하는 것이 목적이다.
- 복잡한 문제를 다루기 위해 분할과 정복(Divide and Conquer) 원리를 적용한다.
- 구조적 방법론의 절차

타당성 검토 단계 → 계획 단계 → 요구사항 단계 → 설계 단계 → 구현 단계 → 시험 단계 → 운용/유지보수 단계

핵심 303 정보공학 방법론

정보공학 방법론은 정보 시스템의 개발을 위해 계획, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합 및 적용하는 자료(Data) 중심의 방법론이다.

- 정보 시스템 개발 주기를 이용하여 대규모 정보 시스템을 구축하는데 적합하다.
- 정보공학 방법론의 절차

정보 전략 계획 수립 단계 → 업무 영역 분석 단계 → 업무 시스템 설계 단계 → 업무 시스템 구축 단계

핵심 304 객체지향 방법론

객체지향 방법론은 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체(Object)로 만들어, 소프트웨어를 개발할 때 기계의 부품을 조립하듯이 객체들을 조립해서 필요한 소프트웨어를 구현하는 방법론이다.

- 객체지향 방법론은 구조적 기법의 문제점으로 인한 소프트웨어 위기의 해결책으로 채택되었다.
- 객체지향 방법론의 구성 요소에는 객체(Object), 클래스(Class), 메시지(Message) 등이 있다.
- 객체지향 방법론의 기본 원칙에는 캡슐화(Encapsulation), 정보 은닉(Information Hiding), 추상화(Abstraction), 상속성(Inheritance), 다형성(Polymorphism) 등이 있다.
- 객체지향 방법론의 절차

요구 분석 단계 → 설계 단계 → 구현 단계 → 테스트 및 검증 단계 → 인도 단계

핵심 305 컴포넌트 기반 방법론

컴포넌트 기반(CBD; Component Based Design) 방법론은 기존의 시스템이나 소프트웨어를 구성하는 컴포넌트를 조합하여 하나의 새로운 애플리케이션을 만드는 방법론이다.

- 컴포넌트의 재사용(Reusability)이 가능하여 시간과 노력을 절감할 수 있다.
- 새로운 기능을 추가하는 것이 간단하여 확장성이 보장된다.
- 유지 보수 비용을 최소화하고 생산성 및 품질을 향상시킬 수 있다.
- 컴포넌트 기반 방법론의 절차

개발 준비 단계 → 분석 단계 → 설계 단계 → 구현 단계 → 테스트 단계 → 전개 단계 → 인도 단계

핵심 306 애자일 방법론

애자일(Agile)은 '민첩한', '기민한'이라는 의미로, 애자일 방법론은 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발 과정을 진행하는 방법론이다.

- 소규모 프로젝트, 고도로 숙달된 개발자, 급변하는 요구사항에 적합하다.
- 애자일 방법론의 대표적인 종류에는 익스트림 프로그래밍(XP; eXtreme Programming), 스크럼(Scrum), 칸반(Kanban), 크리스탈(Crytal) 등이 있다.
- 애자일 방법론의 절차

사용자 스토리 → [반복 주기: 계획 → 개발 → 승인 테스트]

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 307 소프트웨어 재사용의 개요

소프트웨어 재사용(Software Reuse)은 이미 개발되어 인정받은 소프트웨어의 전체 혹은 일부분을 다른 소프트웨어 개발이나 유지에 사용하는 것이다.

- 소프트웨어 개발의 품질과 생산성을 높이기 위한 방법으로, 기존에 개발된 소프트웨어와 경험, 지식 등을 새로운 소프트웨어에 적용한다.
- 재사용의 이점
  - 개발 시간과 비용을 단축시킨다.
  - 소프트웨어 품질을 향상시킨다.
  - 소프트웨어 개발의 생산성을 향상시킨다.
  - 프로젝트 실패의 위험을 감소시킨다.
  - 시스템 구축 방법에 대한 지식을 공유하게 된다.
  - 시스템 명세, 설계, 코드 등 문서를 공유하게 된다.

핵심 308 소프트웨어 재사용 방법

소프트웨어 재사용 방법에는 합성 중심 방법과 생성 중심 방법이 있다.

합성 중심 (Composition-Based)	전자 칩과 같은 소프트웨어 부품, 즉 블록(모듈)을 만들어서 끼워 맞추어 소프트웨어를 완성시키는 방법으로, 블록 구성 방법이라고도 한다.
생성 중심 (Generation-Based)	추상화 형태로 쓰여진 명세를 구체화하여 프로그램을 만드는 방법으로, 패턴 구성 방법이라고도 한다.

핵심 309 소프트웨어 재공학의 개요

소프트웨어 재공학(Software Reengineering)은 새로운 요구에 맞도록 기존 시스템 을 이용하여 보다 나은 시스템을 구축하고, 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것이다.

- 유지보수 비용이 소프트웨어 개발 비용의 대부분을 차지하는 문제를 염두에 두어 기존 소프트웨어의 데이터와 기능들의 개조 및 개선을 통해 유지보수성과 품질을 향상 시키려는 기술이다.
- 유지보수 생산성 향상을 통해 소프트웨어 위기를 해결하는 방법이다.
- 기존 소프트웨어의 기능을 개조하거나 개선하므로, 예방(Preventive) 유지보수 측면에서 소프트웨어 위기를 해결하는 방법이라고 할 수 있다.
- 소프트웨어 재공학도 자동화된 도구를 사용하여 소프트웨어를 분석하고 수정하는 과정을 포함한다.
- 소프트웨어의 수명이 연장되고, 소프트웨어 기술이 향상될 뿐만 아니라 소프트웨어의 개발 기간도 단축된다.
- 소프트웨어에서 발생할 수 있는 오류가 줄어들고, 비용이 절감된다.

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

핵심 310 CASE의 개요

CASE(Computer Aided Software Engineering)는 소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것이다.

- 객체지향 시스템, 구조적 시스템 등 다양한 시스템에서 활용되는 자동화 도구(CASE Tool)이다.
- 소프트웨어, 하드웨어, 데이터베이스, 테스트 등을 통합하여 소프트웨어를 개발하는 환경을 조성한다.
- 소프트웨어 생명 주기의 전체 단계를 연결해 주고 자동화해 주는 통합된 도구를 제공해 주는 기술이다.
- 소프트웨어 개발 도구와 방법론이 결합된 것으로, 정형화된 구조 및 방법(메커니즘)을 소프트웨어 개발에 적용하여 생산성 향상을 구현하는 공학 기법이다.
- 소프트웨어 개발의 모든 단계에 걸쳐 일관된 방법론을 제공하는 자동화 도구들을 지원하고, 개발자들은 이 도구를 사용하여 소프트웨어 개발의 표준화를 지향하며, 자동화의 이점을 얻을 수 있게 해준다.
- CASE의 주요 기능 : 소프트웨어 생명 주기 전 단계의 연결, 다양한 소프트웨어 개발 모형 지원, 그래픽 지원 등

핵심 311 소프트웨어 비용 산정

소프트웨어 비용 산정은 소프트웨어의 개발 규모를 소요되는 인원, 자원, 기간 등으로 확인하여 실행 가능한 계획을 수립하기 위해 필요한 비용을 산정하는 것이다.

- 소프트웨어 비용 산정을 너무 높게 산정할 경우 예산 낭비와 일의 효율성 저하를 초래할 수 있고, 너무 낮게 산정할 경우 개발자의 부담이 가중되고 품질문제가 발생할 수 있다.
- 소프트웨어 비용 산정 기법에는 하향식 비용 산정 기법과 상향식 비용 산정 기법이 있다.
- 소프트웨어 비용을 결정하는 요소에는 프로젝트 요소, 자원 요소, 생산성 요소가 있다.

프로젝트 요소	제품 복잡도, 시스템 크기, 요구되는 신뢰도
자원 요소	인적 자원, 하드웨어 자원, 소프트웨어 자원
생산성 요소	개발자 능력, 개발 기간

핵심 312 하향식 비용 산정 기법

하향식 비용 산정 기법은 과거의 유사한 경험을 바탕으로 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정하는 비과학적인 방법이다.

전문가 감정 기법	• 조직 내에 있는 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 기법이다. • 가장 편리하고 신속하게 비용을 산정할 수 있으며, 의뢰자로부터 믿음을 얻을 수 있다. • 새로운 프로젝트에는 과거의 프로젝트와 다른 요소들이 있다는 것을 간과할 수 있다. • 새로운 프로젝트와 유사한 프로젝트에 대한 경험이 없을 수 있다. • 개인적이고 주관적일 수 있다.
델파이 기법	• 전문가 감정 기법의 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법이다. • 전문가들의 편견이나 분위기에 지배되지 않도록 한 명의 조정자와 여러 전문가로 구성된다.

- 비용 산정 순서

❶ 조정자는 각 비용 산정 요원에게 시스템 정의서와 산정한 비용 내역을 기록할 서식을 제공한다.

❷ 산정 요원들은 정의서를 분석하여 익명으로 그들 나름대로의 비용을 산정한다.

❸ 조정자는 산정 요원들의 반응을 요약하여 배포한다.

❹ 산정 요원들은 이전에 산정한 결과를 이용하여 다시 익명으로 산정한다.

❺ 요원들 간의 의견이 거의 일치할 때까지 이 과정을 반복한다.

핵심 313 상향식 비용 산정 기법

상향식 비용 산정 기법은 프로젝트의 세부적인 작업 단위별로 비용을 산정한 후 집계하여 전체 비용을 산정하는 방법이다.

LOC(원시 코드 라인 수, source Line Of Code) 기법

LOC 기법은 소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법이다.

- 측정이 용이하고 이해하기 쉬워 가장 많이 사용된다.
- 예측치를 이용하여 생산성, 노력, 개발 기간 등의 비용을 산정한다.

$$\text{예측치} = \frac{a+4m+b}{6}$$ 단, a : 낙관치, b : 비관치, m : 기대치(중간치)

- 산정 공식
  - 노력(인월) = 개발 기간 × 투입 인원
    = LOC / 1인당 월평균 생산 코드 라인 수
  - 개발 비용 = 노력(인월) × 단위 비용(1인당 월평균 인건비)
  - 개발 기간 = 노력(인월) / 투입 인원
  - 생산성 = LOC / 노력(인월)

개발 단계별 인월수(Effort Per Task) 기법

개발 단계별 인월수 기법은 LOC 기법을 보완하기 위한 기법으로, 각 기능을 구현시키는 데 필요한 노력을 생명주기의 각 단계별로 산정한다.

- LOC 기법보다 더 정확하다.

핵심 314 수학적 산정 기법의 개요

수학적 산정 기법은 상향식 비용 산정 기법으로, 경험적 추정 모형, 실험적 추정 모형 이라고도 하며, 개발 비용 산정의 자동화를 목표로 한다.

- 비용을 자동으로 산정하기 위해 사용되는 공식은 과거 유사한 프로젝트를 기반으로하여 경험적으로 유도된 것이다.
- 수학적 산정 기법에는 COCOMO 모형, Putnam 모형, 기능 점수(FP) 모형 등이 있으며 각 모형에서는 지정된 공식을 사용하여 비용을 산정한다.

핵심 315 COCOMO의 소프트웨어 개발 유형

조직형 (Organic Mode)	• 기관 내부에서 개발된 중 · 소 규모의 소프트웨어로 일괄 자료 처리나 과학 기술 계산용, 비즈니스 자료 처리용으로 5만(50KDSI) 라인 이하의 소프트웨어를 개발하는 유형이다. • 사무 처리용, 업무용, 과학용 응용 소프트웨어 개발에 적합하다.
반분리형 (Semi-Detached Mode)	• 조직형과 내장형의 중간형으로 트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템 등의 30만(300KDSI) 라인 이하의 소프트웨어를 개발하는 유형이다. • 컴파일러, 인터프리터와 같은 유틸리티 개발에 적합하다.
내장형 (Embedded Mode)	• 초대형 규모의 트랜잭션 처리 시스템이나 운영체제 등의 30만(300KDSI)라인 이상의 소프트웨어를 개발하는 유형이다. • 신호기 제어 시스템, 미사일 유도 시스템, 실시간 처리 시스템 등의 시스템 프로그램 개발에 적합하다.

핵심 316 COCOMO 모형의 종류

기본(Basic)형 COCOMO	소프트웨어의 크기(생산 코드 라인 수)와 개발 유형만을 이용하여 비용을 산정하는 모형이다.
중간 (Intermediate)형 COCOMO	기본형 COCOMO의 공식을 토대로 사용하나, 다음 4가지 특성의 15가지 요인에 의해 비용을 산정하는 모형이다. • 제품의 특성 : 요구되는 신뢰도, 데이터베이스 크기, 제품의 복잡도 • 컴퓨터의 특성 : 수행 시간의 제한, 기억장소의 제한, 가상 기계의 안정성, Turn Around Time • 개발 요원의 특성 : 분석가의 능력, 개발 분야의 경험, 가상 기계의 경험, 프로그래머의 능력, 프로그래밍 언어의 경험 • 프로젝트 특성 : 소프트웨어 도구의 이용, 프로젝트 개발 일정, 최신 프로그래밍 기법의 이용
발전(Detailed)형 COCOMO	중간(Intermediate)형 COCOMO를 보완하여 만들어진 방법으로 개발 공정별로 보다 자세하고 정확하게 노력을 산출하여 비용을 산정하는 모형이다.

핵심 317 Putnam 모형

Putnam 모형은 소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 가정해 주는 모형이다.

- 푸트남(Putnam)이 제안한 것으로 생명 주기 예측 모형이라고도 한다.
- 시간에 따른 함수로 표현되는 Rayleigh-Norden 곡선의 노력 분포도를 기초로 한다.
- 대형 프로젝트의 노력 분포 산정에 이용되는 기법이다.
- 개발 기간이 늘어날수록 프로젝트 적용 인원의 노력이 감소한다.

핵심 318 기능 점수(FP) 모형

기능 점수(Function Point) 모형은 알브레히트(Albrecht)가 제안한 것으로, 소프트웨어의 기능을 증대시키는 요인별로 가중치를 부여하고, 요인별 가중치를 합산하여 총 기능 점수를 산출하며 총 기능 점수와 영향도를 이용하여 기능 점수(FP)를 구한 후 이를 이용해서 비용을 산정하는 기법이다.

기능 점수(FP) = 총 기능 점수 × [0.65 + (0.1 × 총 영향도)]

- 소프트웨어 기능 증대 요인
  - 자료 입력(입력 양식)
  - 정보 출력(출력 보고서)
  - 명령어(사용자 질의수)
  - 데이터 파일
  - 필요한 외부 루틴과의 인터페이스

※ 자동화 추정 도구

- SLIM : Rayleigh-Norden 곡선과 Putnam 예측 모델을 기초로 하여 개발된 자동화 추정 도구
- ESTIMACS : 다양한 프로젝트와 개인별 요소를 수용하도록 FP 모형을 기초로 하여 개발된 자동화 추정 도구

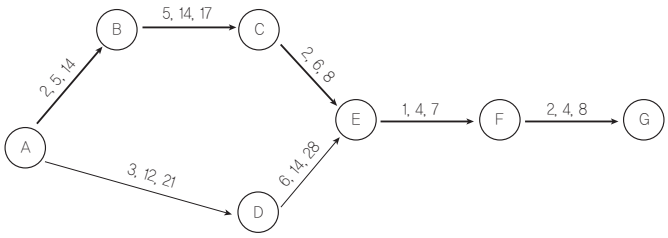
핵심 319 PERT

PERT(Program Evaluation and Review Technique, 프로그램 평가 및 검토 기술)는 프로젝트에 필요한 전체 작업의 상호 관계를 표시하는 네트워크로 각 작업별로 낙관적인 경우, 가능성이 있는 경우, 비관적인 경우로 나누어 각 단계별 종료 시기를 결정하는 방법이다.

- 과거에 경험이 없어서 소요 기간 예측이 어려운 소프트웨어에서 사용한다.
- 노드와 간선으로 구성되며 원 노드에는 작업을, 간선(화살표)에는 낙관치, 기대치, 비관치를 표시한다.
- 결정 경로, 작업에 대한 경계 시간, 작업 간의 상호 관련성 등을 알 수 있다.
- 다음과 같은 PERT 공식을 이용하여 작업 예측치를 계산한다.

$$\text{작업 예측치} = \frac{\text{비관치} + 4 \times \text{기대치} + \text{낙관치}}{6}$$

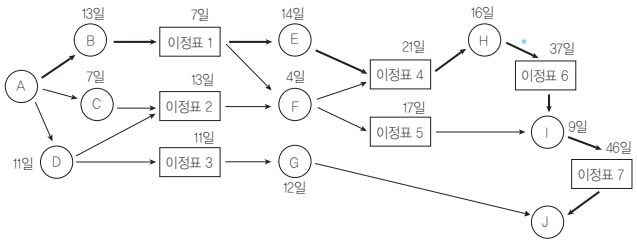
$$\text{평방 편차} = \left[\frac{(\text{비관치} - \text{낙관치})}{6}\right]^2$$



핵심 320 CPM

CPM(Critical Path Method, 임계 경로 기법)은 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는데 사용하는 기법이다.

- CPM은 노드와 간선으로 구성된 네트워크로 노드는 작업을, 간선은 작업 사이의 전후 의존 관계를 나타낸다.
- 원형 노드는 각 작업을 의미하며 각 작업 이름과 소요 기간을 표시하고, 박스 노드 는 이정표를 의미하며 박스 노드 위에는 예상 완료 시간을 표시한다.
- 간선을 나타내는 화살표의 흐름에 따라 각 작업이 진행되며, 전 작업이 완료된 후 다음 작업을 진행할 수 있다.
- 임계 경로는 최장 경로를 의미한다.



핵심 321 간트 차트

간트 차트는 프로젝트의 각 작업들이 언제 시작하고 언제 종료되는지에 대한 작업 일 정을 막대 도표를 이용하여 표시하는 프로젝트 일정표로, 시간선(Time-Line) 차트라고도 한다.

- 중간 목표 미달성 시 그 이유와 기간을 예측할 수 있게 한다.
- 사용자와의 문제점이나 예산의 초과 지출 등도 관리할 수 있게 한다.
- 자원 배치와 인원 계획에 유용하게 사용된다.
- 다양한 형태로 변경하여 사용할 수 있다.
- 작업 경로는 표시할 수 없으며, 계획의 변화에 대한 적응성이 약하다.
- 계획 수립 또는 수정 때 주관적 수치에 기울어지기 쉽다.
- 간트 차트는 이정표, 작업 일정, 작업 기간, 산출물로 구성되어 있다.
- 수평 막대의 길이는 각 작업(Task)의 기간을 나타낸다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 322 소프트웨어 개발 방법론 결정의 개요

소프트웨어 개발 방법론의 결정은 프로젝트 관리와 재사용 현황을 소프트웨어 개발 방법론에 반영하고, 확정된 소프트웨어 생명 주기와 개발 방법론에 맞춰 소프트웨어 개발 단계, 활동, 작업, 절차 등을 정의하는 것이다.

※ 프로젝트 관리(Project Management)

프로젝트 관리는 주어진 기간 내에 최소의 비용으로 사용자를 만족시키는 시스템을 개발하기 위한 전반적인 활동이다.

관리 유형	주요 내용
일정 관리	작업 순서, 작업 기간 산정, 일정 개발, 일정 통제
비용 관리	비용 산정, 비용 예산 편성, 비용 통제
인력 관리	프로젝트 팀 편성, 자원 산정, 프로젝트 조직 정의, 프로젝트 팀 개발, 자원 통제, 프로젝트 팀 관리
위험 관리	위험 식별, 위험 평가, 위험 대처, 위험 통제
품질 관리	품질 계획, 품질 보증 수행, 품질 통제 수행

핵심 323 ISO/IEC 12207

ISO/IEC 12207은 ISO(International Organization for Standardization, 국제표준화기구)에서 만든 표준 소프트웨어 생명 주기 프로세스로, 소프트웨어의 개발, 운영, 유지보수 등을 체계적으로 관리하기 위한 소프트웨어 생명 주기 표준을 제공한다.

- ISO/IEC 12207은 기본 생명 주기 프로세스, 지원 생명 주기 프로세스, 조직 생명 주기 프로세스로 구분한다.

기본 생명 주기 프로세스	획득, 공급, 개발, 운영, 유지보수 프로세스
지원 생명 주기 프로세스	품질 보증, 검증, 확인, 활동 검토, 감사, 문서화, 형상 관리, 문제 해결 프로세스
조직 생명 주기 프로세스	관리, 기반 구조, 훈련, 개선 프로세스

핵심 324 CMMI(Capability Maturity Model Integration)

CMMI(능력 성숙도 통합 모델)는 소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델로, 미국 카네기멜론 대학교의 소프트웨어 공학연구소(SEI)에서 개발하였다.

- CMMI의 소프트웨어 프로세스 성숙도는 초기, 관리, 정의, 정량적 관리, 최적화의 5단계로 구분한다.

단계	프로세스	특징
초기(Initial)	정의된 프로세스 없음	작업자 능력에 따라 성공 여부 결정
관리(Managed)	규칙화된 프로세스	특정한 프로젝트 내의 프로세스 정의 및 수행
정의(Defined)	표준화된 프로세스	조직의 표준 프로세스를 활용하여 업무 수행
정량적 관리(Quantitatively Managed)	예측 가능한 프로세스	프로젝트를 정량적으로 관리 및 통제
최적화(Optimizing)	지속적 개선 프로세스	프로세스 역량 향상을 위해 지속적인 프로세스 개선

핵심 325 SPICE(Software Process Improvement and Capability dEtermination)

SPICE(소프트웨어 처리 개선 및 능력 평가 기준)는 정보시스템 분야에서 소프트웨어의 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준으로, 공식 명칭은 ISO/IEC 15504이다.

- SPICE는 5개의 프로세스 범주와 40개의 세부 프로세스로 구성된다.
- SPICE는 프로세스 수행 능력 단계를 불완전, 수행, 관리, 확립, 예측, 최적화의 6단계로 구분한다.

단계	특징
불완전(Incomplete)	프로세스가 구현되지 않았거나 목적을 달성하지 못한 단계이다.
수행(Performed)	프로세스가 수행되고 목적이 달성된 단계이다.
관리(Managed)	정의된 자원의 한도 내에서 그 프로세스가 작업 산출물을 인도하는 단계이다.

확립 (Established)	소프트웨어 공학 원칙에 기반하여 정의된 프로세스가 수행되는 단계이다.
예측 (Predictable)	프로세스가 목적 달성을 위해 통제되고, 양적인 측정을 통해서 일관되게 수행되는 단계이다.
최적화 (Optimizing)	프로세스 수행을 최적화하고, 지속적인 개선을 통해 업무 목적을 만족시키는 단계이다.

핵심 326 소프트웨어 개발 방법론 테일러링의 개요 및 고려사항

소프트웨어 개발 방법론 테일러링은 프로젝트 상황 및 특성에 맞도록 정의된 소프트웨어 개발 방법론의 절차, 사용기법 등을 수정 및 보완하는 작업이다.

- 소프트웨어 개발 방법론 테일러링 작업 시 고려해야 할 사항에는 내부적 기준과 외부적 기준이 있다.

내부적 기준	• 목표 환경 : 시스템의 개발 환경과 유형이 서로 다른 경우 테일러링이 필요하다. • 요구사항 : 프로젝트의 생명 주기 활동에서 개발, 운영, 유지보수 등 프로젝트에서 우선적으로 고려할 요구사항이 서로 다른 경우 테일러링이 필요하다. • 프로젝트 규모 : 비용, 인력, 기간 등 프로젝트의 규모가 서로 다른 경우 테일러링이 필요하다. • 보유 기술 : 프로세스, 개발 방법론, 산출물, 구성원의 능력 등이 서로 다른 경우 테일러링이 필요하다.
외부적 기준	• 법적 제약사항 : 프로젝트별로 적용될 IT Compliance가 서로 다른 경우 테일러링이 필요하다. • 표준 품질 기준 : 금융, 제도 등 분야별 표준 품질 기준이 서로 다른 경우 테일러링이 필요하다.

핵심 327 소프트웨어 개발 프레임워크

프레임워크(Framework)는 소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러 가지 기능들을 제공해주는 반제품 형태의 소프트웨어 시스템이다.

- 선행 사업자의 기술에 의존하지 않은 표준화된 개발 기반으로 인해 사업자 종속성이 해소된다.
- 프레임워크의 주요 기능에는 예외 처리, 트랜잭션 처리, 메모리 공유, 데이터 소스 관리, 서비스 관리, 쿼리 서비스, 로깅 서비스, 사용자 인증 서비스 등이 있다.
- 프레임워크의 종류

스프링 프레임워크 (Spring Framework)	자바 플랫폼을 위한 오픈 소스 경량형 애플리케이션 프레임워크이다.
전자정부 프레임워크	우리나라의 공공부문 정보화 사업 시 효율적인 정보 시스템의 구축을 지원하기 위해 필요한 기능 및 아키텍처를 제공하는 프레임워크이다.
닷넷 프레임워크 (.NET Framework)	Windows 프로그램의 개발 및 실행 환경을 제공하는 프레임워크로, Microsoft 사에서 통합 인터넷 전략을 위해 개발하였다.

핵심 328 프레임워크의 특성

모듈화 (Modularity)	• 프레임워크는 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상시킨다. • 프레임워크는 개발표준에 의한 모듈화로 인해 유지 보수가 용이하다.
재사용성 (Reusability)	프레임워크는 재사용 가능한 모듈들을 제공함으로써 예산 절감, 생산성 향상, 품질 보증이 가능하다.
확장성 (Extensibility)	프레임워크는 다형성(Polymorphism)을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션 개발이 가능하다.
제어의 역흐름 (Inversion of Control)	개발자가 관리하고 통제해야 하는 객체들의 제어를 프레임워크에 넘김으로써 생산성을 향상시킨다.

핵심 329 네트워크 관련 신기술

IoT(Internet of Things, 사물 인터넷)	정보 통신 기술을 기반으로 실세계(Physical World)와 가상 세계(Virtual World)의 다양한 사물들을 인터넷으로 서로 연결하여 진보된 서비스를 제공하기 위한 서비스 기반 기술이다.
M2M(Machine to Machine, 사물 통신)	• 무선 통신을 이용한 기계와 기계 사이의 통신이다. • M2M은 변압기 원격 감시, 전기, 가스 등의 원격 검침, 무선 신용카드 조회기, 무선 보안단말기, 버스 운행 시스템, 위치 추적 시스템, 시설물 관리 등을 무선으로 통합하여 상호 작용하는 통신이다.
모바일 컴퓨팅 (Mobile Computing)	휴대형 기기로 이동하면서 자유로이 네트워크에 접속하여 업무를 처리할 수 있는 환경을 말한다.

클라우드컴퓨팅 (Cloud Computing)	• 각종 컴퓨팅 자원을 중앙 컴퓨터에 두고 인터넷 기능을 갖는 단말기로 언제 어디서나 인터넷을 통해 컴퓨터 작업을 수행할 수 있는 환경을 의미한다. • 중앙 컴퓨터는 복수의 데이터 센터를 가상화 기술로 통합한 대형 데이터 센터로, 각종 소프트웨어, 데이터, 보안 솔루션 기능 등 컴퓨팅 자원을 보유하고 있다.
모바일 클라우드 컴퓨팅 (MCC; Mobile Cloud Computing)	클라우드 서비스를 이용하여 소비자와 소비자의 파트너가 모바일 기기로 클라우드 컴퓨팅 인프라를 구성하여 여러 가지 정보와 자원을 공유하는 ICT(Information and Communications Technologies) 기술을 의미한다.
인터클라우드 컴퓨팅 (Inter-Cloud Computing)	각기 다른 클라우드 서비스를 연동하거나 컴퓨팅 자원의 동적 할당이 가능하도록 여러 클라우드 서비스 제공자들이 제공하는 클라우드 서비스나 자원을 연결하는 기술을 말한다.
메시 네트워크 (Mesh Network)	차세대 이동통신, 홈네트워킹, 공공 안전 등 특수 목적을 위한 새로운 방식의 네트워크 기술로, 대규모 디바이스의 네트워크 생성에 최적화되어 있다.
와이선(Wi-SUN)	스마트 그리드와 같은 장거리 무선 통신을 필요로 하는 사물 인터넷(IoT) 서비스를 위한 저전력 장거리(LPWA; Low-Power Wide Area) 통신 기술이다.
NDN (Named Data Networking)	콘텐츠 자체의 정보와 라우터 기능만으로 데이터 전송을 수행하는 기술로, 클라이언트와 서버가 패킷의 헤더에 내장되어 있는 주소 정보를 이용하여 연결되던 기존의 IP(Internet Protocol) 망을 대체할 새로운 인터넷 아키텍처로 떠오르고 있다.
NGN(Next Generation Network, 차세대 통신망)	ITU-T에서 개발하고 있는 유선망 기반의 차세대 통신망으로, 유선망뿐만 아니라 이동 사용자를 목표로 하며, 이동통신에서 제공하는 완전한 이동성(Full Mobility) 제공을 목표로 개발되고 있다.
SDN(Software Defined Networking, 소프트웨어 정의 네트워킹)	네트워크를 컴퓨터처럼 모델링하여 여러 사용자가 각각의 소프트웨어들로 네트워킹을 가상화하여 제어하고 관리하는 네트워크이다.
NFC (Near Field Communication, 근거리 무선 통신)	• 고주파(HF)를 이용한 근거리 무선 통신 기술이다. • NFC는 Ecma 340, ISO/IEC 18092 표준으로, 아주 가까운 거리에서 양방향 통신을 지원하는 RFID 기술의 일종이다.
UWB(Ultra WideBand, 초광대역)	짧은 거리에서 많은 양의 디지털 데이터를 낮은 전력으로 전송하기 위한 무선 기술로 무선 디지털 펄스라고도 하며, 블루투스와 비교되는 기술이다.

피코넷 (PICONET)	여러 개의 독립된 통신장치가 블루투스 기술이나 UWB 통신 기술을 사용하여 통신망을 형성하는 무선 네트워크 기술이다.
WBAN (Wireless Body Area Network)	웨어러블(Wearable) 또는 몸에 심는(Implant) 형태의 센서나 기기를 무선으로 연결하는 개인 영역 네트워킹 기술이다.
GIS(Geographic Information System, 지리 정보 시스템)	지리적인 자료를 수집 · 저장 · 분석 · 출력할 수 있는 컴퓨터 응용 시스템으로, 위성을 이용해 모든 사물의 위치 정보를 제공해 주는 것을 말한다.
USN(Ubiquitous Sensor Network, 유비쿼터스 센서 네트워크)	각종 센서로 수집한 정보를 무선으로 수집할 수 있도록 구성한 네트워크를 말한다. 즉 필요한 모든 것(곳)에 RFID 태그를 부착하고, 이를 통하여 사물의 인식 정보는 물론 주변의 환경정보까지 탐지하여 이를 네트워크에 연결하여 정보를 관리하는 것을 의미한다.
SON(Self Organizing Network, 자동 구성 네트워크)	• 주변 상황에 맞추어 스스로 망을 구성하는 네트워크를 말한다. • SON의 목적은 통신망 커버리지 및 전송 용량 확장의 경제성 문제를 해결하고, 망의 운영과 관리의 효율성을 높이는 것이다.
애드 혹 네트워크 (Ad-hoc Network)	재난 현장과 같이 별도의 고정된 유선망을 구축할 수 없는 장소에서 모바일 호스트(Mobile Host)만을 이용하여 구성한 네트워크로, 망을 구성한 후 단기간 사용되는 경우나 유선망을 구성하기 어려운 경우에 적합하다.
네트워크 슬라이싱 (Network Slicing)	3GPP를 포함한 여러 글로벌 이동통신 표준화 단체가 선정한 5G(IMT-2020)의 핵심기술 중 하나로, 네트워크에서 하나의 물리적인 코어 네트워크 인프라(Infrastructure)를 독립된 다수의 가상 네트워크로 분리하여 각각의 네트워크를 통해 다양한 고객 맞춤형 서비스를 제공하는 것을 목적으로 하는 네트워크 기술이다.
저전력 블루투스 기술 (BLE; Bluetooth Low Energy)	일반 블루투스와 동일한 2.4GHz 주파수 대역을 사용하지만 연결되지 않은 대기 상태에서는 절전 모드를 유지하는 기술이다.
지능형 초연결망	과학기술정보통신부 주관으로 추진 중인 사업으로, 스마트 시티, 스마트 스테이션 등 4차 산업혁명 시대를 맞아 새로운 변화에 따라 급격하게 증가하는 데이터 트래픽을 효과적으로 수용하기 위해 시행되는 정부 주관 사업이다.
파장 분할 다중화(WDM, Wavelength Division Multiplexing)	• 광섬유를 이용한 통신 기술의 하나로, 파장이 서로 다른 복수의 신호를 보냄으로써 여러 대의 단말기가 동시에 통신 회선을 사용할 수 있도록 하는 것이다. • 파장이 다른 광선끼리는 서로 간섭을 일으키지 않는 성질을 이용한 기술이다.

시험에 나오는 것만 공부한다! 시나공시리즈

소프트웨어 정의 데이터 센터 (SDDC, Software Defined Data Center)	데이터 센터의 모든 자원을 가상화하여 인력의 개입 없이 소프트웨어 조작만으로 관리 및 제어되는 데이터 센터를 의미한다.
개방형 링크드 데이터 (LOD, Linked Open Data)	Linked Data와 Open Data의 합성어로, 누구나 사용할 수 있도록 웹상에 공개된 연계 데이터를 의미한다.

핵심 330 네트워크(Network) 설치 구조

성형 (Star, 중앙 집중형)	• 중앙에 중앙 컴퓨터가 있고, 이를 중심으로 단말장치들이 연결되는 중앙 집중식의 네트워크 구성 형태이다. • 포인트 투 포인트(Point-to-Point) 방식으로 회선을 연결한다.
링형 (Ring, 루프형)	• 컴퓨터와 단말장치들을 서로 이웃하는 것끼리 포인트 투 포인트(Point-to-Point) 방식으로 연결시킨 형태이다. • 분산 및 집중 제어 모두 가능하다. • 데이터는 단방향 또는 양방향으로 전송할 수 있으며, 단방향 링의 경우 컴퓨터, 단말장치, 통신 회선 중 어느 하나라도 고장나면 전체 통신망에 영향을 미친다.
버스형(Bus)	• 한 개의 통신 회선에 여러 대의 단말장치가 연결되어 있는 형태이다. • 물리적 구조가 간단하고, 단말장치의 추가와 제거가 용이하다. • 단말장치가 고장나더라도 통신망 전체에 영향을 주지 않기 때문에 신뢰성을 높일 수 있다.
계층형 (Tree, 분산형)	중앙 컴퓨터와 일정 지역의 단말장치까지는 하나의 통신 회선으로 연결시키고, 이웃하는 단말장치는 일정 지역 내에 설치된 중간 단말장치로부터 다시 연결시키는 형태이다.
망형(Mesh)	• 모든 지점의 컴퓨터와 단말장치를 서로 연결한 형태로, 노드의 연결성이 높다. • 많은 단말장치로부터 많은 양의 통신을 필요로 하는 경우에 유리하다. • 보통 공중 데이터 통신망에서 사용되며, 통신 회선의 총 경로가 가장 길다. • 모든 노드를 망형으로 연결하려면 노드의 수가 n개일 때, n(n-1)/2개의 회선이 필요하고 노드당 n-1개의 포트가 필요하다.

핵심 331 네트워크 분류

네트워크는 각 사이트들이 분포되어 있는 지리적 범위에 따라 LAN과 WAN으로 분류된다.

근거리 통신망 (LAN; Local Area Network)	• 회사, 학교, 연구소 등에서 비교적 가까운 거리에 있는 컴퓨터, 프린터, 테이프 등과 같은 자원을 연결하여 구성한다. • 주로 자원 공유를 목적으로 사용한다. • 사이트 간의 거리가 짧아 데이터의 전송 속도가 빠르고, 에러 발생율이 낮다. • 근거리 통신망에서는 주로 버스형이나 링형 구조를 사용한다.
광대역 통신망 (WAN; Wide Area Network)	• 국가와 국가 혹은 대륙과 대륙 등과 같이 멀리 떨어진 사이트들을 연결하여 구성한다. • 사이트 간의 거리가 멀기 때문에 통신 속도가 느리고, 에러 발생률이 높다. • 일정한 지역에 있는 사이트들을 근거리 통신망으로 연결한 후 각 근거리 통신망을 연결하는 방식을 사용한다.

핵심 332 LAN의 표준안

IEEE 802의 주요 표준 규격

IEEE 802 위원회에서 지정한 LAN의 표준 규격은 다음과 같다.

표준 규격	내용
802.1	전체의 구성, OSI 참조 모델과의 관계, 통신망 관리 등에 관한 규약이다.
802.2	논리 링크 제어(LLC) 계층에 관한 규약이다.
802.3	CSMA/CD 방식의 매체 접근 제어 계층에 관한 규약이다.
802.4	토큰 버스 방식의 매체 접근 제어 계층에 관한 규약이다.
802.5	토큰 링 방식의 매체 접근 제어 계층에 관한 규약이다.
802.6	도시형 통신망(MAN)에 관한 규약이다.
802.9	종합 음성/데이터 네트워크에 관한 규약이다.
802.11	무선 LAN에 관한 규약이다.

802.11의 버전

802.11 (초기 버전)	2.4GHz 대역 전파와 CSMA/CA 기술을 사용해 최고 2Mbps까지의 전송 속도를 지원한다.
802.11a	5GHz 대역의 전파를 사용하며, OFDM 기술을 사용해 최고 54Mbps까지의 전송 속도를 지원한다.

802.11b	802.11 초기 버전의 개선안으로 등장하였으며, 초기 버전의 대역 전파와 기술을 사용해 최고 11Mbps의 전송 속도로 기존에 비해 5배 이상 빠르게 개선되었다.
802.11e	802.11의 부가 기능 표준으로, QoS 기능이 지원되도록 하기 위해 매체 접근 제어(MAC) 계층에 해당하는 부분을 수정하였다.
802.11g	2.4GHz 대역의 전파를 사용하지만 5GHz 대역의 전파를 사용하는 802.11a와 동일한 최고 54Mbps까지의 전송 속도를 지원한다.
802.11n	2.4GHz 대역과 5GHz 대역을 사용하는 규격으로, 최고 600Mbps까지의 전송 속도를 지원한다.

핵심 333 스위치

스위치(Switch) 분류

스위치는 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치로, OSI 7 계층의 Layer에 따라 L2, L3, L4, L7으로 분류된다.

L2 스위치	• OSI의 2계층에 속하는 장비이다. • 일반적으로 부르는 스위치는 L2 스위치를 의미한다. • MAC 주소를 기반으로 프레임을 전송한다. • 동일 네트워크 간의 연결만 가능하다.
L3 스위치	• OSI의 3계층에 속하는 장비이다. • L2 스위치에 라우터 기능이 추가된 것으로, IP 주소를 기반으로 패킷을 전송한다. • 서로 다른 네트워크 간의 연결이 가능하다.
L4 스위치	• OSI 4계층에 속하는 장비이다. • 로드밸런서가 달린 L3 스위치로, IP 주소 및 TCP/UDP를 기반으로 사용자들의 요구를 서버의 부하가 적은 곳에 배분하는 로드밸런싱 기능을 제공한다.
L7 스위치	• OSI 7계층에 속하는 장비이다. • IP 주소, TCP/UDP 포트 정보에 패킷 내용까지 참조하여 세밀하게 로드밸런싱한다.

스위칭(Switch) 방식

스위치가 프레임을 전달하는 방식에 따라 Store and Forwarding, Cut-through, Fragment Free가 있다.

Store and Forwarding	데이터를 모두 받은 후 스위칭하는 방식
Cut-through	데이터의 목적지 주소만을 확인한 후 바로 스위칭하는 방식
Fragment Free	Store and Forwarding과 Cut-through 방식의 장점을 결합한 방식

핵심 334 경로 제어(Routing)

경로 제어는 송 · 수신 측 간의 전송 경로 중에서 최적 패킷 교환 경로를 결정하는 기능이다.

- 최적 패킷 교환 경로란 어느 한 경로에 데이터의 양이 집중하는 것을 피하면서, 최저의 비용으로 최단 시간에 송신할 수 있는 경로를 의미한다.
- 경로 제어 요소 : 성능 기준, 경로의 결정 시간과 장소, 정보 발생지, 경로 정보의 갱신 시간
- 경로 제어 프로토콜(Routing Protocol)

GP(Interior Gateway Protocol, 내부 게이트웨이 프로토콜)	• 하나의 자율 시스템(AS) 내의 라우팅에 사용되는 프로토콜이다. • RIP(Routing Information Protocol) – 현재 가장 널리 사용되는 라우팅 프로토콜로 거리 벡터 라우팅 프로토콜이라고도 불리며, 최단 경로 탐색에 Bellman-Ford 알고리즘이 사용된다. – 소규모 동종의 네트워크(자율 시스템, AS) 내에서 효율적인 방법이다. – 최대 홉(Hop) 수를 15로 제한하므로 15 이상의 경우는 도달할 수 없는 네트워크를 의미하는데 이것은 대규모 네트워크에서는 RIP를 사용할 수 없음을 의미한다. • OSPF(Open Shortest Path First protocol) – RIP의 단점을 해결하여 새로운 기능을 지원하는 인터넷 프로토콜로, 대규모 네트워크에서 많이 사용된다. – 인터넷 망에서 이용자가 최단 경로를 선정할 수 있도록 라우팅 정보에 노드 간의 거리 정보, 링크 상태 정보를 실시간으로 반영하여 최단 경로로 라우팅을 지원한다. – 최단 경로 탐색에 다익스트라(Dijkstra) 알고리즘을 사용한다. – 라우팅 정보에 변화가 생길 경우 변화된 정보만 네트워크 내의 모든 라우터에 알린다. – 하나의 자율 시스템(AS)에서 동작하면서 내부 라우팅 프로토콜의 그룹에 도달한다.
EGP(Exterior Gateway Protocol, 외부 게이트웨이 프로토콜)	자율 시스템(AS) 간의 라우팅, 즉 게이트웨이 간의 라우팅에 사용되는 프로토콜이다.
BGP(Border Gateway Protocol)	• 자율 시스템(AS) 간의 라우팅 프로토콜로, EGP의 단점을 보완하기 위해 만들어졌다. • 초기에 BGP 라우터들이 연결될 때에는 전체 경로 제어표(라우팅 테이블)를 교환하고, 이후에는 변화된 정보만을 교환한다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 335 트래픽 제어(Traffic Control)

트래픽 제어는 네트워크의 보호, 성능 유지, 네트워크 자원의 효율적인 이용을 위해 전송되는 패킷의 흐름 또는 그 양을 조절하는 기능으로 흐름 제어, 폭주(혼합) 제어, 교착상태 방지 기법이 있다.

- 흐름 제어(Flow Control) : 흐름 제어란 네트워크 내의 원활한 흐름을 위해 송 · 수신 측 사이에 전송되는 패킷의 양이나 속도를 규제하는 기능이다.

정지-대기 (Stop-and-Wait)	• 수신 측의 확인 신호(ACK)를 받은 후에 다음 패킷을 전송하는 방식이다. • 한 번에 하나의 패킷만을 전송할 수 있다.
슬라이딩 윈도우 (Sliding Window)	• 확인 신호, 즉 수신 통지를 이용하여 송신 데이터의 양을 조절하는 방식이다. • 수신 측의 확인 신호를 받지 않더라도 미리 정해진 패킷의 수만큼 연속적으로 전송하는 방식으로, 한 번에 여러 개의 패킷을 전송할 수 있어 전송 효율이 좋다. • 송신 측은 수신 측으로부터 확인 신호(ACK) 없이도 보낼 수 있는 패킷의 최대치를 미리 약속받는데, 이 패킷의 최대치가 윈도우 크기(Window Size)를 의미한다. • 윈도우 크기(Window Size)는 상황에 따라 변한다. 즉, 수신 측으로부터 이전에 송신한 패킷에 대한 긍정 수신 응답(ACK)이 전달된 경우 윈도우 크기는 증가하고, 수신 측으로부터 이전에 송신한 패킷에 대한 부정 수신 응답(NAK)이 전달된 경우 윈도우 크기는 감소한다.

- 폭주(혼잡) 제어(Congestion Control) : 흐름 제어(Flow Control)가 송 · 수신 측 사이의 패킷 수를 제어하는 기능이라면, 폭주 제어는 네트워크 내의 패킷 수를 조절하여 네트워크의 오버플로(Overflow)를 방지하는 기능을 한다.

느린 시작 (Slow Start)	• 윈도우의 크기를 1, 2, 4, 8, …과 같이 2배씩 지수적으로 증가시켜 초기에는 느리지만 갈수록 빨라진다. • 전송 데이터의 크기가 임계 값에 도달하면 혼잡 회피 단계로 넘어간다.
혼잡 회피 (Congestion Avoidance)	느린 시작(Slow Start)의 지수적 증가가 임계 값에 도달되면 혼잡으로 간주하고 회피를 위해 윈도우의 크기를 1씩 선형적으로 증가시켜 혼잡을 예방하는 방식이다.

- 교착상태(Dead Lock) 방지 : 교착상태란 교환기 내에 패킷들을 축적하는 기억 공간이 꽉 차 있을 때 다음 패킷들이 기억 공간에 들어가기 위해 무한정 기다리는 현상을 말한다.

핵심 336 SW 관련 신기술

인공지능 (AI; Artificial Intelligence)	인간의 두뇌와 같이 컴퓨터 스스로 추론, 학습, 판단 등 인간지능적인 작업을 수행하는 시스템이다.
뉴럴링크 (Neuralink)	• 미국의 전기자동차 회사 테슬라(Tesla)의 CEO 일론 머스크(Elon Musk)가 사람의 뇌와 컴퓨터를 결합하는 기술을 개발하기 위해 2017년 3월 설립한 회사이다. • 뉴럴링크가 개발하고 있는 기술은 '신경 레이스(Neural Lace)'로, 작은 전극을 뇌에 이식함으로써 생각을 업로드하고 다운로드하는 것을 목표로 삼고 있다.
딥 러닝 (Deep Learning)	인간의 두뇌를 모델로 만들어진 인공 신경망(ANN; Artificial Neural Network)을 기반으로 하는 기계 학습 기술이다.
전문가 시스템 (Expert System)	의료 진단 등과 같은 특정 분야의 전문가가 수행하는 고도의 업무를 지원하기 위한 컴퓨터 응용 프로그램이다.
증강현실 (AR; Augmented Reality)	실제 촬영한 화면에 가상의 정보를 부가하여 보여주는 기술이다.
블록체인 (Blockchain)	P2P 네트워크를 이용하여 온라인 금융 거래 정보를 온라인 네트워크 참여자(Peer)의 디지털 장비에 분산 저장하는 기술을 의미한다.
분산 원장 기술 (DLT; Distributed Ledger Technology)	중앙 관리자나 중앙 데이터 저장소가 존재하지 않고 P2P 망 내의 참여자들에게 모든 거래 목록이 분산 저장되어 거래가 발생할 때마다 지속적으로 갱신되는 디지털 원장을 의미한다.
해시(Hash)	임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것을 말한다.
양자 암호키 분배 (QKD; Quantum Key Distribution)	양자 통신을 위해 비밀키를 분배하여 관리하는 기술로, 두 시스템이 암호 알고리즘 동작을 위한 비밀키를 안전하게 공유하기 위해 양자 암호키 분배 시스템을 설치하여 운용하는 방식으로 활용된다.
프라이버시 강화 기술 (PET; Privacy Enhancing Technology)	• 개인정보 위험 관리 기술이다. • PET는 최근 심각한 위험으로 대두되고 있는 개인정보 침해 위험을 관리하기 위한 핵심 기술로, 암호화, 익명화 등 개인정보를 보호하는 기술에서 사용자가 직접 개인정보를 통제하기 위한 기술까지 다양한 사용자 프라이버시 보호 기술을 통칭한다.

디지털 저작권 관리(DRM; Digital Rights Management)	인터넷이나 기타 디지털 매체를 통해 유통되는 데이터의 저작권을 보호를 위해 데이터의 안전한 배포를 활성화하거나 불법 배포를 방지하기 위한 시스템이다.
공통 평가 기준 (CC; Common Criteria)	• 1999년 6월 8일 ISO 15408 표준으로 채택된 정보 보호 제품 평가 기준이다. • 공통 평가 기준은 정보화 순기능 역할을 보장하기 위해 정보화 제품의 정보보호 기능과 이에 대한 사용 환경 등급을 정한 기준이다.
개인정보 영향평가 제도 (PIA; Privacy Impact As sessment)	개인 정보를 활용하는 새로운 정보시스템의 도입 및 기존 정보시스템의 중요한 변경 시 시스템의 구축 · 운영이 기업의 고객은 물론 국민의 사생활에 미칠 영향에 대해 미리 조사 · 분석 · 평가하는 제도이다.
그레이웨어 (Grayware)	소프트웨어를 제공하는 입장에서는 악의적이지 않은 유용한 소프트웨어라고 주장할 수 있지만 사용자 입장에서는 유용할 수도 있고 악의적일 수도 있는 애드웨어, 트랙웨어, 기타 악성 코드나 악성 공유웨어를 말한다.
매시업(Mashup)	웹에서 제공하는 정보 및 서비스를 이용하여 새로운 소프트웨어나 서비스, 데이터베이스 등을 만드는 기술이다. 즉 다수의 정보원이 제공하는 콘텐츠를 조합하여 하나의 서비스로 제공하는 웹 사이트 또는 애플리케이션을 말한다.
리치 인터넷 애플리케이션 (RIA; Rich Internet Application)	플래시 애니메이션 기술과 웹 서버 애플리케이션 기술을 통합하여 기존 HTML 보다 역동적이고 인터랙티브한 웹페이지를 제공하는 신개념의 플래시 웹페이지 제작 기술이다.
시맨틱 웹 (Semantic Web)	컴퓨터가 사람을 대신하여 정보를 읽고 이해하고 가공하여 새로운 정보를 만들어 낼 수 있도록 이해하기 쉬운 의미를 가진 차세대 지능형 웹이다.
증발품 (Vaporware)	판매 계획 또는 배포 계획은 발표되었으나 실제로 고객에게 판매되거나 배포되지 않고 있는 소프트웨어이다.
오픈 그리드 서비스 아키텍처 (OGSA; Open Grid Service Architecture)	애플리케이션 공유를 위한 웹 서비스를 그리드 상에서 제공하기 위해 만든 개방형 표준이다.
서비스 지향 아키텍처 (SOA; Service Oriented Architecture)	• 기업의 소프트웨어 인프라인 정보시스템을 공유와 재사용이 가능한 서비스 단위나 컴포넌트 중심으로 구축하는 정보기술 아키텍처이다. • SOA 기반 애플리케이션 구성 계층 : 표현(Presentation) 계층, 업무 프로세스(Biz-Process) 계층, 서비스 중간(Service Intermediary) 계층, 애플리케이션(Application) 계층, 데이터 저장(Persistency) 계층
서비스형 소프트웨어 (SaaS; Software as a Service)	소프트웨어의 여러 기능 중에서 사용자가 필요로 하는 서비스만 이용할 수 있도록 한 소프트웨어이다.
소프트웨어 에스크로(임치) (Software Escrow)	소프트웨어 개발자의 지식재산권을 보호하고 사용자는 저렴한 비용으로 소프트웨어를 안정적으로 사용 및 유지보수 받을 수 있도록 소스 프로그램과 기술 정보 등을 제3의 기관에 보관하는 것이다.
복잡 이벤트 처리 (CEP; Complex Event Processing)	실시간으로 발생하는 많은 사건들 중 의미가 있는 것만을 추출할 수 있도록 사건 발생 조건을 정의하는 데이터 처리 방법이다.
디지털 트윈 (Digital Twin)	• 현실속의 사물을 소프트웨어로 가상화한 모델로, 자동차, 항공, 에너지, 국방, 헬스케어 등 여러 분야에서 주목 받고 있다. • 실제 물리적인 자산을 소프트웨어로 가상화함으로써 실제 자산의 특성에 대한 정확한 정보를 얻을 수 있고, 자산 최적화, 돌발사고 최소화, 생산성 증가 등 설계부터 제조, 서비스에 이르는 모든 과정의 효율성을 향상시킬 수 있다.

핵심 337 소프트웨어 개발 직무별 보안 활동

안전한 소프트웨어 개발을 위해서는 프로젝트 관련자들이 보안 활동을 수행할 수 있도록 각 직무(Role)별로 수행해야 할 보안 활동을 정의해야 한다.

- 소프트웨어 개발 참여자의 직무는 프로젝트 관리자, 요구사항 분석가, 아키텍트, 설계자, 구현 개발자, 테스트 분석가, 보안 감사자로 구분한다.

프로젝트 관리자 (Project Manager)	• 응용 프로그램에 대한 보안 전략을 조직 구성원들에게 전달한다. • 조직 구성원들에게 응용 프로그램 보안 영향을 이해시킨다.
요구사항 분석가 (Requirement Specifier)	• 아키텍트가 고려해야 할 보안 관련 비즈니스 요구사항을 설명한다. • 프로젝트 팀이 고려해야 할 구조 정의 및 해당 구조에 존재하는 자원에 대한 보안 요구사항을 정의한다.

아키텍트 (Architect)	• 보안 오류가 발생하지 않도록 보안 기술 문제를 충분히 이해한다. • 시스템에 사용되는 모든 리소스 정의 및 각 리소스별로 적절한 보안 요구사항을 적용한다.
설계자 (Designer)	• 특정 기술에 대해 보안 요구사항의 만족성 여부를 확인한다. • 문제 발생 시 최선의 문제 해결 방법을 결정한다. • 애플리케이션 보안 수준에 대한 품질 측정을 지원한다.
구현 개발자 (Implementer)	구조화된 소프트웨어 개발 환경에서 프로그램을 원활히 구현할 수 있도록 시큐어 코딩 표준을 준수하여 개발한다.
테스트 분석가 (Test Analyst)	• 소프트웨어 개발 요구사항과 구현 결과를 반복적으로 확인한다. • 테스트 분석가는 반드시 보안 전문가일 필요는 없지만 보안 위험에 대한 학습이나 툴(Tool) 사용법 정도는 숙지하고 있어야 한다.
보안 감사자 (Security Auditor)	• 소프트웨어 개발 프로젝트의 현재 상태의 보안을 보장한다. • 요구사항 검토 시 요구사항의 적합성과 완전성을 확인한다. • 소프트웨어 개발 프로젝트의 전체 단계에서 활동한다.

핵심 338 개인정보 보호 관련 법령

- **개인정보 보호법** : 개인정보의 처리 및 보호에 관한 사항을 정함으로써 개인의 자유와 권리를 보호한다.
- **정보통신망 이용촉진 및 정보보호 등에 관한 법률** : 정보통신망의 이용 촉진 및 정보통신 서비스를 이용하는 이용자들의 개인정보를 보호한다.
- **신용정보의 이용 및 보호에 관한 법률** : 개인 신용정보의 효율적 이용과 체계적인 관리를 통해 정보의 오남용을 방지한다.
- **위치정보의 보호 및 이용 등에 관한 법률** : 개인 위치정보의 안전한 이용 환경을 조성하여 정보의 유출이나 오남용을 방지한다.
- **표준 개인정보 보호 지침** : 개인정보의 처리에 관한 기준, 개인정보 침해의 유형 및 예방조치 등에 관한 세부사항을 규정한다.
- **개인정보의 안전성 확보 조치 기준** : 개인정보 처리자가 개인정보를 처리하는데 있어 개인정보가 분실, 도난, 유출, 위조, 변조, 훼손되지 않도록 안전성 확보에 필요한 기술적, 관리적, 물리적 안전조치에 관한 최소한의 기준을 규정한다.
- **개인정보 영향평가에 관한 고시** : 개인정보 영향평가를 위한 평가기관의 지정, 영향평가의 절차 등에 관한 세부 기준을 규정한다.

핵심 339 HW 관련 신기술

고가용성 (HA; High Availability)	• 긴 시간동안 안정적인 서비스 운영을 위해 장애 발생 시 즉시 다른 시스템으로 대체 가능한 환경을 구축하는 메커니즘을 의미한다. • 가용성(Availability)을 극대화하는 방법으로는 클러스터, 이중화 등이 있다.
3D Printing (Three Dimension Printing)	대상을 평면에 출력하는 것이 아니라 손으로 만질 수 있는 실제 물체로 만들어내는 것을 말한다.
4D Printing (Fourth Dimension Printing)	특정 시간이나 환경 조건이 갖추어지면 스스로 형태를 변화시키거나 제조되는 자가 조립(Self-Assembly) 기술이 적용된 제품을 3D Printing하는 기술을 의미한다.
RAID(Redundant Array of Inexpensive Disk, Redundant Array of Independent Disk)	여러 개의 하드디스크로 디스크 배열을 구성하여 파일을 구성하고 있는 데이터 블록들을 서로 다른 디스크들에 분산 저장할 경우 그 블록들을 여러 디스크에서 동시에 읽거나 쓸 수 있으므로 디스크의 속도가 매우 향상되는데, 이 기술을 RAID라고 한다.
4K 해상도	• 차세대 고화질 모니터의 해상도를 지칭하는 용어이다. • 4K 해상도는 가로 픽셀 수가 3840이고, 세로 픽셀 수가 2160인 영상의 해상도를 말하는데, 이는 Full HDTV(1920×1080)의 가로·세로 2배, 총 4배에 해당하는 초고화질의 영상이다.
앤 스크린 (N-Screen)	N개의 서로 다른 단말기에서 동일한 콘텐츠를 자유롭게 이용할 수 있는 서비스를 말한다.
컴패니언 스크린 (Companion Screen)	앤 스크린(N Screen)의 한 종류로, TV 방송 시청 시 방송 내용을 공유하며 추가적인 기능을 수행할 수 있는 스마트폰, 태블릿PC 등을 의미한다. 세컨드 스크린(Second Screen)이라고도 불린다.

신 클라이언트 PC (Thin Client PC)	하드디스크나 주변 장치 없이 기본적인 메모리만 갖추고 서버와 네트워크로 운용되는 개인용 컴퓨터를 말하는 것으로, 서버 기반 컴퓨팅과 관계가 깊다.
패블릿(Phablet)	폰(Phone)과 태블릿(Tablet)의 합성어로, 태블릿 기능을 포함한 5인치 이상의 대화면 스마트폰을 말한다.
C형 유에스비(Universal Serial Bus Type-C, USB Type-C, USB-C)	• 범용 인터페이스 규격인 유에스비(USB; Universal Serial Bus)의 표준 중 하나로, 2014년 8월 USB IF(Implementers Forum)에서 발표되었다. • C형 유에스비는 기존 A형에 비하여 크기가 작고, 24핀으로 위아래의 구분이 없어 어느 방향으로든 연결이 가능하다.
멤스(MEMS; Micro-Electro Mechanical Systems)	초정밀 반도체 제조 기술을 바탕으로 센서, 액추에이터(Actuator) 등 기계 구조를 다양한 기술로 미세 가공하여 전기기계적 동작을 할 수 있도록 한 초미세 장치이다.
트러스트존 기술 (TrustZone Technology)	칩 설계회사인 ARM(Advanced RISC Machine)에서 개발한 기술로, 하나의 프로세서(Processor) 내에 일반 애플리케이션을 처리하는 일반 구역(Normal World)과 보안이 필요한 애플리케이션을 처리하는 보안 구역(Secure World)으로 분할하여 관리하는 하드웨어 기반의 보안 기술이다.
엠디스크(M-DISC, Millennial DISC)	한 번의 기록만으로 자료를 영구 보관할 수 있는 광 저장 장치이다.
멤리스터(Memristor)	메모리(Memory)와 레지스터(Resister)의 합성어로, 전류의 방향과 양 등 기존의 경험을 모두 기억하는 특별한 소자이다.

핵심 340 Secure OS의 개요

Secure OS는 기존의 운영체제(OS)에 내재된 보안 취약점을 해소하기 위해 보안 기능을 갖춘 커널을 이식하여 외부의 침입으로부터 시스템 자원을 보호하는 운영체제를 의미한다.

- 보호 방법을 구현하기 복잡한 것부터 차례로 분류하면 다음과 같다.
  - 암호적 분리(Cryptographic Separation) : 내부 정보를 암호화하는 방법
  - 논리적 분리(Logical Separation) : 프로세스의 논리적 구역을 지정하여 구역을 벗어나는 행위를 제한하는 방법
  - 시간적 분리(Temporal Separation) : 동일 시간에 하나의 프로세스만 수행되도록 하여 동시 실행으로 발생하는 보안 취약점을 제거하는 방법
  - 물리적 분리(Physical Separation) : 사용자별로 특정 장비만 사용하도록 제한하는 방법

※ 참조 모니터(Reference Monitor)

참조 모니터는 보호대상의 객체에 대한 접근통제를 수행하는 추상머신이며, 이것을 실제로 구현한 것이 보안 커널이다.

- 참조 모니터는 보안 커널 데이터베이스(SKDB; Security Kernel Database)를 참조하여 객체에 대한 접근 허가 여부를 결정한다.
- 참조 모니터와 보안 커널의 특징
  - 격리성(Isolation) : 부정 조작이 불가능해야 함
  - 검증가능성(Verifiability) : 적절히 구현되었다는 것을 확인할 수 있어야 함
  - 완전성(Completeness) : 우회가 불가능해야 함

핵심 341 DB 관련 신기술

빅데이터 (Big Data)	기존의 관리 방법이나 분석 체계로는 처리하기 어려운 막대한 양의 정형 또는 비정형 데이터 집합으로, 스마트 단말의 빠른 확산, 소셜 네트워크 서비스의 활성화, 사물 네트워크의 확대로 데이터 폭발이 더욱 가속화되고 있다.
브로드 데이터 (Broad Data)	다양한 채널에서 소비자와 상호 작용을 통해 생성된, 기업 마케팅에 있어 효율적이고 다양한 데이터이며, 이전에 사용하지 않거나 알지 못했던 새로운 데이터나, 기존 데이터에 새로운 가치가 더해진 데이터를 말한다.
메타 데이터 (Meta Data)	일련의 데이터를 정의하고 설명해 주는 데이터이다. 컴퓨터에서는 데이터 사전의 내용, 스키마 등을 의미하고, HTML 문서에서는 메타 태그 내의 내용이 메타 데이터이다. 방송에서는 방대한 분량의 저작물을 신속하게 검색하기 위한 촬영 일시, 장소, 작가, 출연자 등과 음원의 검색을 위한 작곡자나 가수명 등을 메타 데이터로 처리한다.
디지털 아카이빙 (Digital Archiving)	디지털 정보 자원을 장기적으로 보존하기 위한 작업을 말한다. 아날로그 콘텐츠는 디지털로 변환한 후 압축해서 저장하고, 디지털 콘텐츠도 체계적으로 분류하고 메타 데이터를 만들어 DB화하는 작업이다.

하둡 (Hadoop)	• 오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼이다. • 일반 PC급 컴퓨터들로 가상화된 대형 스토리지를 형성하고 그 안에 보관된 거대한 데이터 세트를 병렬로 처리할 수 있도록 개발된 자바 소프트웨어 프레임워크로, 구글, 야후 등에 적용되고 있다.
맵리듀스 (MapReduce)	대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델로, 흩어져 있는 데이터를 연관성 있는 데이터 분류로 묶는 Map 작업을 수행한 후 중복 데이터를 제거하고 원하는 데이터를 추출하는 Reduce 직업을 수행한다.
타조(Tajo)	오픈 소스 기반 분산 컴퓨팅 플랫폼인 아파치 하둡(Apache Hadoop) 기반의 분산 데이터 웨어하우스 프로젝트로, 우리나라가 주도하여 개발하고 있다.
데이터 다이어트 (Data Diet)	데이터를 삭제하는 것이 아니라 압축하고, 중복된 정보는 중복을 배제하고, 새로운 기준에 따라 나누어 저장하는 작업이다.
데이터 마이닝 (Data Mining)	• 데이터 웨어하우스에 저장된 데이터 집합에서 사용자의 요구에 따라 유용하고 가능성 있는 정보를 발견하기 위한 기법이다. • 대량의 데이터를 분석하여 데이터 속에 내재되어 있는 변수 사이의 상호관계를 규명하여 패턴화함으로써 효율적인 데이터 추출이 가능하다.
OLAP(Online Analytical Processing)	• 다차원으로 이루어진 데이터로부터 통계적인 요약 정보를 분석하여 의사결정에 활용하는 방식을 말한다. • OLAP 연산 : Roll-up, Drill-down, Drill-through, Drill-across, Pivoting, Slicing, Dicing

핵심 342 회복(Recovery)

- 회복은 트랜잭션들을 수행하는 도중 장애가 발생하여 데이터베이스가 손상되었을 때 손상되기 이전의 정상 상태로 복구하는 작업이다.
- 회복 기법

연기 갱신 기법 (Deferred Update)	• 트랜잭션이 성공적으로 완료될 때까지 데이터베이스에 대한 실질적인 갱신을 연기하는 방법이다. • 트랜잭션이 수행되는 동안 갱신된 내용은 일단 Log에 보관된다. • 트랜잭션의 부분 완료(성공적인 완료 직전) 시점에 Log에 보관한 갱신 내용을 실제 데이터베이스에 기록한다. • 트랜잭션이 부분 완료되기 전에 장애가 발생하여 트랜잭션이 Rollback되면 트랜잭션이 실제 데이터베이스에 영향을 미치지 않았기 때문에 어떠한 갱신 내용도 취소(Undo)시킬 필요 없이 무시하면 된다. • Redo 작업만 가능하다.
즉각 갱신 기법 (Immediate Update)	• 트랜잭션이 데이터를 갱신하면 트랜잭션이 부분 완료되기 전이라도 즉시 실제 데이터베이스에 반영하는 방법이다. • 장애가 발생하여 회복 작업할 경우를 대비하여 갱신된 내용들은 Log에 보관시킨다. • 회복 작업을 할 경우에는 Redo와 Undo 모두 사용 가능하다.
그림자 페이지 대체 기법 (Shadow Paging)	• 갱신 이전의 데이터베이스를 일정 크기의 페이지 단위로 구성하여 각 페이지마다 복사본인 그림자 페이지로 별도 보관해 놓고, 실제 페이지를 대상으로 트랜잭션에 의한 갱신 작업을 하다가 장애가 발생하여 트랜잭션 작업을 Rollback시킬 때, 갱신된 이후의 실제 페이지 부분에 그림자 페이지를 대체하여 회복시키는 기법이다. • 로그, Undo 및 Redo 알고리즘이 필요 없다.
검사점 기법 (Check Point)	트랜잭션 실행 중 특정 단계에서 재실행할 수 있도록 갱신 내용이나 시스템에 대한 상황 등에 관한 정보와 함께 검사점을 로그에 보관해 두고, 장애 발생 시 트랜잭션 전체를 철회하지 않고 검사점부터 회복 작업을 하여 회복시간을 절약하도록 하는 기법이다.

핵심 343 병행제어(Concurrency Control)

병행제어란 다중 프로그램의 이점을 활용하여 동시에 여러 개의 트랜잭션을 병행수행할 때, 동시에 실행되는 트랜잭션들이 데이터베이스의 일관성을 파괴하지 않도록 트랜잭션 간의 상호 작용을 제어하는 것이다.

- 병행제어 기법의 종류

로킹(Locking)	• 주요 데이터의 액세스를 상호 배타적으로 하는 것이다. • 트랜잭션들이 어떤 로킹 단위를 액세스하기 전에 Lock(잠금)을 요청해서 Lock이 허락되어야만 그 로킹 단위를 액세스할 수 있도록 하는 기법이다.
타임 스탬프 순서 (Time Stamp Ordering)	• 직렬성 순서를 결정하기 위해 트랜잭션 간의 처리 순서를 미리 선택하는 기법들 중에서 가장 보편적인 방법이다. • 트랜잭션과 트랜잭션이 읽거나 갱신한 데이터에 대해 트랜잭션이 실행을 시작하기 전에 시간표(Time Stamp)를 부여하여 부여된 시간에 따라 트랜잭션 작업을 수행하는 기법이다. • 교착상태가 발생하지 않는다.

최적 병행수행 (검증 기법, 확인 기법, 낙관적 기법)	병행수행하고자 하는 대부분의 트랜잭션이 판독 전용(Read Only) 트랜잭션일 경우, 트랜잭션 간의 충돌률이 매우 낮아서 병행제어 기법을 사용하지 않고 실행되어도 이 중의 많은 트랜잭션은 시스템의 상태를 일관성 있게 유지한다는 점을 이용한 기법이다.
다중 버전 기법	• 타임 스탬프의 개념을 이용하는 기법으로, 다중 버전 타임 스탬프 기법이라고도 한다. • 타임 스탬프 기법은 트랜잭션 및 데이터들이 이용될 때의 시간을 시간표로 관리하지만, 다중 버전 기법은 갱신될 때마다의 버전을 부여하여 관리한다.

※ 로킹 단위(Locking Granularity)

- 병행제어에서 한꺼번에 로킹할 수 있는 객체의 크기를 의미한다.
- 데이터베이스, 파일, 레코드, 필드 등이 로킹 단위가 될 수 있다.
- 로킹 단위가 크면 로크 수가 작아 관리하기 쉽지만 병행성 수준이 낮아지고, 로킹 단위가 작으면 로크 수가 많아 관리하기 복잡해 오버헤드가 증가하지만 병행성 수준이 높아진다.

핵심 344 교착상태

교착상태(Dead Lock)는 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상을 의미한다.

- 교착상태 발생의 필요 충분 조건

상호 배제 (Mutual Exclusion)	한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 한다.
점유와 대기 (Hold and Wait)	최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용되고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 한다.
비선점 (Non-preemption)	다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 한다.
환형 대기 (Circular Wait)	공유 자원과 공유 자원을 사용하기 위해 대기하는 프로세스들이 원형으로 구성되어 있어 자신에게 할당된 자원을 점유하면서 앞이나 뒤에 있는 프로세스의 자원을 요구해야 한다.

- 교착상태의 해결 방법

예방 기법 (Prevention)	• 교착상태가 발생하지 않도록 사전에 시스템을 제어하는 방법으로, 교착상태 발생의 네 가지 조건 중에서 어느 하나를 제거(부정)함으로써 수행된다. • 자원의 낭비가 가장 심한 기법이다.
회피 기법 (Avoidance)	• 교착상태가 발생할 가능성을 배제하지 않고 교착상태가 발생하면 적절히 피해나가는 방법으로, 주로 은행원 알고리즘(Banker's Algorithm)이 사용된다. • 은행원 알고리즘(Banker's Algorithm) : E. J. Dijkstra가 제안한 것으로, 은행에서 모든 고객의 요구가 충족되도록 현금을 할당하는 데서 유래한 기법
발견 기법 (Detection)	• 시스템에 교착상태가 발생했는지 점검하여 교착상태에 있는 프로세스와 자원을 발견하는 것을 의미한다. • 교착상태 발견 알고리즘과 자원 할당 그래프 등을 사용할 수 있다.
회복 기법 (Recovery)	교착상태를 일으킨 프로세스를 종료하거나 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것을 의미한다.

핵심 345 데이터 표준화

데이터 표준화는 시스템을 구성하는 데이터 요소의 명칭, 정의, 형식, 규칙에 대한 원칙을 수립하고 적용하는 것을 의미한다.

- 데이터 표준화의 대상으로는 데이터 명칭, 데이터 정의, 데이터 형식, 데이터 규칙이 있다.
- 데이터 표준화의 구성 요소에는 데이터 표준, 데이터 관리 조직, 데이터 표준화 절차가 있다.

데이터 표준	• 데이터 모델이나 DB에서 정의할 수 있는 모든 오브젝트를 대상으로 데이터 표준화를 수행해야 한다. • 데이터 표준의 종류 – 표준 단어 : 업무에서 사용하고 일정한 의미를 갖고 있는 최소 단위의 단어를 의미한다. – 표준 도메인 : 문자형, 숫자형, 날짜형, 시간형과 같이 컬럼을 성질에 따라 그룹핑한 개념이다. – 표준 코드 : 선택할 수 있는 값을 정형화하기 위해 기준에 맞게 이미 정의된 코드 값으로, 도메인의 한 유형이다. – 표준 용어 : 단어, 도메인, 코드 표준이 정의되면 이를 바탕으로 표준 용어를 구성한다.

데이터 관리 조직	• 데이터 표준 원칙이나 데이터 표준의 준수 여부 등을 관리하는 사람들로, 대표적으로 데이터 관리자가 있다. • 데이터 관리자는 조직 내의 데이터에 대한 정의, 체계화, 감독 등의 업무를 담당한다.
데이터 표준화 절차	데이터 표준화 요구사항 수집, 데이터 표준 정의, 데이터 표준 확정, 데이터 표준 관리 순으로 진행된다.

- 데이터 표준화의 기대 효과
  - 동일한 데이터에 대해 동일한 명칭을 지정하면 명확한 의사소통이 가능하다.
  - 표준화된 데이터를 사용하면 필요한 데이터의 의미나 위치 등을 쉽게 파악할 수 있다.
  - 데이터 표준에 따라 데이터 형식 및 규칙을 적용하면 입력 오류를 방지하고 잘못된 데이터로 인한 의사 결정의 오류를 줄여 데이터 품질을 향상시킬 수 있다.
  - 데이터 표준에 따라 데이터를 전사적으로 관리하면 시스템 간 데이터 공유 시 데이터 변환이나 정제 작업을 수행하지 않아도 된다.
  - 향후 데이터 유지보수 및 운영의 효율성, 관리 비용을 절감할 수 있다.

핵심 346 Secure SDLC의 개요

Secure SDLC는 보안상 안전한 소프트웨어를 개발하기 위해 SDLC에 보안 강화를 위한 프로세스를 포함한 것을 의미한다.

- Secure SDLC는 소프트웨어의 유지 보수 단계에서 보안 이슈를 해결하기 위해 소모되는 많은 비용을 최소화하기 위해 등장하였다.
- Secure SDLC는 요구사항 분석, 설계, 구현, 테스트, 유지 보수 등 SDLC 전체 단계에 걸쳐 수행되어야 할 보안 활동을 제시한다.
- Secure SDLC의 대표적인 방법론

CLASP	• Secure Software 사에서 개발하였으며, SDLC의 초기 단계에서 보안을 강화하기 위해 개발된 방법론이다. • 활동 중심, 역할 기반의 프로세스로 구성되어 있으며, 현재 운용 중인 시스템에 적용하기에 적합하다.
SDL	• 마이크로소프트 사에서 안전한 소프트웨어 개발을 위해 기존의 SDLC를 개선한 방법론이다. • 전통적인 나선형 모델을 기반으로 한다.
Seven Touchpoints	• 소프트웨어 보안의 모범사례를 SDLC에 통합한 방법론이다. • 설계 및 개발 과정의 모든 산출물에 대해 위험 분석 및 테스트를 수행한다. • SDLC의 각 단계에 관련된 7개의 보안 강화 활동을 수행한다.

핵심 347 보안 요소

보안 요소는 소프트웨어 개발에 있어 충족시켜야할 요소 및 요건을 의미한다.

- 보안 3대 요소에는 기밀성(Confidentiality), 무결성(Integrity), 가용성(Availability)이 있으며, 그 외에도 인증(Authentication), 부인 방지(NonRepudiation) 등이 있다.

기밀성	• 시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용된다. • 정보가 전송 중에 노출되더라도 데이터를 읽을 수 없다.
무결성	시스템 내의 정보는 오직 인가된 사용자만 수정할 수 있다.
가용성	인가받은 사용자는 언제라도 사용할 수 있다.
인증	• 시스템 내의 정보와 자원을 사용하려는 사용자가 합법적인 사용자인지를 확인하는 모든 행위를 말한다. • 대표적 방법으로는 패스워드, 인증용 카드, 지문 검사 등이 있다.
부인 방지	데이터를 송 · 수신한 자가 송 · 수신 사실을 부인할 수 없도록 송 · 수신 증거를 제공한다.

핵심 348 세션 통제

세션 통제의 개요

세션은 서버와 클라이언트의 연결을 의미하고, 세션 통제는 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것을 의미한다.

- 세션 통제는 소프트웨어 개발 과정 중 요구사항 분석 및 설계 단계에서 진단해야 하는 보안 점검 내용이다.
- 세션 통제의 보안 약점에는 불충분한 세션 관리, 잘못된 세션에 의한 정보 노출이 있다.

세션 설계시 고려 사항

- 시스템의 모든 페이지에서 로그아웃이 가능하도록 UI(User Interface)를 구성한다.
- 로그아웃 요청 시 할당된 세션이 완전히 제거되도록 한다.
- 세션 타임아웃은 중요도가 높으면 2~5분, 낮으면 15~30분으로 설정한다.
- 이전 세션이 종료되지 않으면 새 세션이 생성되지 못하도록 설계한다.
- 중복 로그인을 허용하지 않은 경우 클라이언트의 중복 접근에 대한 세션 관리 정책을 수립한다.
- 패스워드 변경 시 활성화된 세션을 삭제하고 재할당한다.

세션ID의 관리 방법

- 세션ID는 안전한 서버에서 최소 128비트의 길이로 생성한다.
- 세션ID의 예측이 불가능하도록 안전한 난수 알고리즘을 적용한다.
- 세션ID가 노출되지 않도록 URL Rewrite 기능을 사용하지 않는 방향으로 설계한다.
- 로그인 시 로그인 전의 세션ID를 삭제하고 재할당한다.
- 장기간 접속하고 있는 세션ID는 주기적으로 재할당되도록 설계한다.

핵심 349 입력 데이터 검증 및 표현

입력 데이터 검증 및 표현의 개요

입력 데이터 검증 및 표현은 입력 데이터로 인해 발생하는 문제들을 예방하기 위해 구현 단계에서 검증해야 하는 보안 점검 항목들이다.

입력 데이터 검증 및 표현의 보안 약점

SQL 삽입	• 웹 응용 프로그램에 SQL을 삽입하여 내부 데이터베이스(DB) 서버의 데이터를 유출 및 변조하고, 관리자 인증을 우회하는 보안 약점이다. • 동적 쿼리에 사용되는 입력 데이터에 예약어 및 특수문자가 입력되지 않게 필터링 되도록 설정하여 방지할 수 있다.
경로 조작 및 자원 삽입	• 데이터 입출력 경로를 조작하여 서버 자원을 수정 · 삭제할 수 있는 보안 약점이다. • 사용자 입력값을 식별자로 사용하는 경우, 경로 순회 공격을 막는 필터를 사용하여 방지할 수 있다.
크로스사이트 스크립팅(XSS)	• 웹페이지에 악의적인 스크립트를 삽입하여 방문자들의 정보를 탈취하거나, 비정상적인 기능 수행을 유발하는 보안 약점이다. • HTML 태그의 사용을 제한하거나 스크립트에 삽입되지 않도록 '〈', '〉', '&' 등의 문자를 다른 문자로 치환함으로써 방지할 수 있다.
운영체제 명령어 삽입	• 외부 입력값을 통해 시스템 명령어의 실행을 유도함으로써 권한을 탈취하거나 시스템 장애를 유발하는 보안 약점이다. • 웹 인터페이스를 통해 시스템 명령어가 전달되지 않도록 하고, 외부 입력값을 검증 없이 내부 명령어로 사용하지 않음으로써 방지할 수 있다.
위험한 형식 파일 업로드	• 악의적인 명령어가 포함된 스크립트 파일을 업로드함으로써 시스템에 손상을 주거나, 시스템을 제어할 수 있는 보안 약점이다. • 업로드 되는 파일의 확장자 제한, 파일명의 암호화, 웹사이트와 파일 서버의 경로 분리, 실행 속성을 제거하는 등의 방법으로 방지할 수 있다.
신뢰되지 않는 URL 주소로 자동접속 연결	• 입력 값으로 사이트 주소를 받는 경우 이를 조작하여 방문자를 피싱 사이트로 유도하는 보안 약점이다. • 연결되는 외부 사이트의 주소를 화이트 리스트로 관리함으로써 방지할 수 있다.
메모리 버퍼 오버플로	• 연속된 메모리 공간을 사용하는 프로그램에서 할당된 메모리의 범위를 넘어선 위치에서 자료를 읽거나 쓰려고 할 때 발생하는 보안 약점이다. • 프로그램의 오동작을 유발시키거나, 악의적인 코드를 실행시켜 공격자가 프로그램을 통제할 수 있는 권한을 획득하게 한다. • 메모리 버퍼를 사용할 경우 적절한 버퍼의 크기를 설정하고, 설정된 범위의 메모리 내에서 올바르게 읽거나 쓸 수 있도록 함으로써 방지할 수 있다.

핵심 350 보안 기능

보안 기능의 개요

보안 기능은 소프트웨어 개발의 구현 단계에서 코딩하는 기능인 인증, 접근제어, 기밀성, 암호화 등을 올바르게 구현하기 위한 보안 점검 항목들이다.

보안 기능의 보안 약점

적절한 인증 없이 중요기능 허용	• 보안검사를 우회하여 인증과정 없이 중요한 정보 또는 기능에 접근 및 변경이 가능하다. • 중요정보나 기능을 수행하는 페이지에서는 재인증 기능을 수행하도록 하여 방지할 수 있다.

부적절한 인가	• 접근제어 기능이 없는 실행경로를 통해 정보 또는 권한을 탈취할 수 있다. • 모든 실행경로에 대해 접근제어 검사를 수행하고, 사용자에게는 반드시 필요한 접근 권한만을 부여하여 방지할 수 있다.
중요한 자원에 대한 잘못된 권한 설정	• 권한 설정이 잘못된 자원에 접근하여 해당 자원을 임의로 사용할 수 있다. • 소프트웨어 관리자만 자원들을 읽고 쓸 수 있도록 설정하고, 인가되지 않은 사용자의 중요 자원에 대한 접근 여부를 검사함으로써 방지할 수 있다.
취약한 암호화 알고리즘 사용	• 암호화된 환경설정 파일을 해독하여 비밀번호 등의 중요정보를 탈취할 수 있다. • 안전한 암호화 알고리즘을 이용하고, 업무관련 내용이나 개인정보 등에 대해서는 IT보안인증사무국이 안정성을 확인한 암호모듈을 이용함으로써 방지할 수 있다.
중요정보 평문 저장 및 전송	• 암호화되지 않은 평문 데이터를 탈취하여 중요한 정보를 획득할 수 있다. • 중요한 정보를 저장하거나 전송할 때는 반드시 암호화 과정을 거치도록 하고, HTTPS 또는 SSL과 같은 보안 채널을 이용함으로써 방지할 수 있다.
하드코드된 비밀번호	• 소스코드 유출 시 내부에 하드코드된 패스워드를 이용하여 관리자 권한을 탈취할 수 있다. • 패스워드는 암호화하여 별도의 파일에 저장하고, 디폴트 패스워드나 디폴드 키의 사용을 피함으로써 방지할 수 있다.

핵심 351 시간 및 상태

시간 및 상태의 개요

시간 및 상태는 동시 수행을 지원하는 병렬 처리 시스템이나 다수의 프로세스가 동작하는 환경에서 시간과 실행 상태를 관리하여 시스템이 원활하게 동작되도록 하기 위한 보안 검증 항목들이다.

TOCTOU 경쟁 조건	• 검사 시점(Time Of Check)과 사용 시점(Time Of Use)을 고려하지 않고 코딩하는 경우 발생하는 보안 약점이다. • 검사 시점에는 사용이 가능했던 자원이 사용 시점에는 사용할 수 없게 된 경우에 발생한다.
종료되지 않는 반복문 또는 재귀함수	• 반복문이나 재귀함수에서 종료 조건을 정의하지 않았거나 논리 구조상 종료될 수 없는 경우 발생하는 보안 약점이다. • 반복문이나 재귀함수가 종료되지 않을 경우 시스템 자원이 끊임없이 사용되어 자원고갈로 인한 서비스 장애 또는 시스템 장애가 발생한다.

핵심 352 에러처리

에러처리의 개요

에러처리는 소프트웨어 실행 중 발생할 수 있는 오류(Error)들을 사전에 정의하여 오류로 인해 발생할 수 있는 문제들을 예방하기 위한 보안 점검 항목들이다.

- 각 프로그래밍 언어의 예외처리 구문을 통해 오류에 대한 사항을 정의한다.
- 예외처리 구문으로 처리하지 못한 오류들은 중요정보를 노출시키거나, 소프트웨어의 실행이 중단되는 등 예기치 못한 문제를 발생시킬 수 있다.
- 에러처리의 미비로 인한 코딩이 유발하는 보안 약점에는 오류 메시지를 통한 정보노출, 오류 상황 대응 부재, 부적절한 예외처리가 있다.

오류 메시지를 통한 정보노출	오류 발생으로 실행 환경, 사용자 정보, 디버깅 정보 등의 중요 정보를 소프트웨어가 메시지로 외부에 노출하는 보안 약점이다.
오류 상황 대응 부재	소프트웨어 개발 중 예외처리를 하지 않았거나 미비로 인해 발생하는 보안 약점이다.
부적절한 예외처리	함수의 반환값 또는 오류들을 세분화하여 처리하지 않고 광범위하게 묶어 한 번에 처리하거나, 누락된 예외가 존재할 때 발생하는 보안 약점이다.

핵심 353 코드 오류

코드 오류의 개요

코드 오류는 소프트웨어 구현 단계에서 개발자들이 코딩 중 실수하기 쉬운 형(Type) 변환, 자원 반환 등의 오류를 예방하기 위한 보안 점검 항목들이다.

널 포인터(Null Pointer) 역참조	널 포인터가 가리키는 메모리에 어떠한 값을 저장할 때 발생하는 보안 약점이다.
부적절한 자원 해제	자원을 반환하는 코드를 누락하거나 프로그램 오류로 할당된 자원을 반환하지 못했을 때 발생하는 보안 약점이다.
해제된 자원 사용	이미 사용이 종료되어 반환된 메모리를 참조하는 경우 발생하는 보안 약점이다.
초기화되지 않은 변수 사용	변수 선언 후 값이 부여되지 않은 변수를 사용할 때 발생하는 보안 약점이다.

※ 스택 가드(Stack Guard)

- 널 포인터 역참조와 같이 주소가 저장되는 스택에서 발생하는 보안 약점을 막는 기술 중 하나이다.
- 메모리상에서 프로그램의 복귀 주소와 변수 사이에 특정 값을 저장한 후 그 값이 변경되었을 경우 오버플로우 상태로 판단하여 프로그램 실행을 중단함으로써 잘못된 복귀 주소의 호출을 막는 기술이다.

핵심 354 캡슐화

캡슐화의 개요

캡슐화는 정보 은닉이 필요한 중요한 데이터와 기능을 불충분하게 캡슐화하거나 잘못 사용함으로써 발생할 수 있는 문제를 예방하기 위한 보안 점검 항목들이다.

잘못된 세션에 의한 정보 노출	다중 스레드(Multi-Thread) 환경에서 멤버 변수에 정보를 저장할 때 발생하는 보안 약점이다.
제거되지 않고 남은 디버그 코드	개발 중에 버그 수정이나 결과값 확인을 위해 남겨둔 코드들로 인해 발생하는 보안 약점이다.
시스템 데이터 정보 노출	시스템의 내부 정보를 시스템 메시지 등을 통해 외부로 출력하도록 코딩했을 때 발생하는 보안 약점이다.
Public 메소드로부터 반환된 Private 배열	선언된 클래스 내에서만 접근이 가능한 Private 배열을 모든 클래스에서 접근이 가능한 Public 메소드에서 반환할 때 발생하는 보안 약점이다.
Private 배열에 Public 데이터 할당	Private 배열에 Public으로 선언된 데이터 또는 메소드의 파라미터를 저장할 때 발생하는 보안 약점이다.

※ 접근 지정자(접근 제어자)

접근 지정자는 프로그래밍 언어에서 특정 개체를 선언할 때 외부로부터의 접근을 제한하기 위해 사용되는 예약어이다(접근 가능 : ○, 접근 불가능 : ×).

한정자	클래스 내부	패키지 내부	하위 클래스	패키지 외부
Public	○	○	○	○
Protected	○	○	○	×
Default	○	○	×	×
Private	○	×	×	×

핵심 355 API 오용

API 오용의 개요

API 오용은 소프트웨어 구현 단계에서 API를 잘못 사용하거나 보안에 취약한 API를 사용하지 않도록 하기 위한 보안 검증 항목들이다.

DNS Lookup에 의존한 보안 결정

도메인명에 의존하여 인증이나 접근 통제 등의 보안 결정을 내리는 경우 발생하는 보안 약점이다.

- DNS 엔트리를 속여 동일한 도메인에 속한 서버인 것처럼 위장하거나, 사용자와 서버 간의 네트워크 트래픽을 유도하여 악성 사이트를 경유하도록 조작할 수 있다.

취약한 API 사용

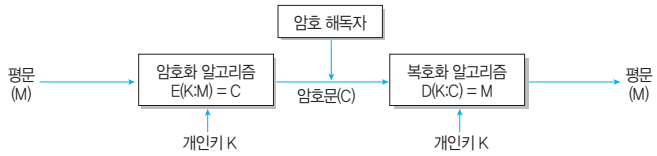
보안 문제로 사용이 금지된 API를 사용하거나, 잘못된 방식으로 API를 사용했을 때 발생하는 보안 약점이다.

- 보안 문제로 금지된 대표적인 API에는 C언어의 문자열 함수 strcat( ), strcpy( ), sprintf( ) 등이 있다.

핵심 356 개인키 암호화(Private Key Encryption) 기법

개인키 암호화 기법은 동일한 키로 데이터를 암호화하고 복호화한다.

- 데이터베이스 사용자는 평문의 정보 M을 암호화 알고리즘 E와 개인키(Private Key) K를 이용하여 암호문 C로 바꾸어 저장시켜 놓으면 사용자는 그 데이터베이스에 접근하기 위해 복호화 알고리즘 D와 개인키 K를 이용하여 다시 평문의 정보 M으로 바꾸어 이용하는 방법이다.



- 개인키 암호화 기법은 대칭 암호 기법 또는 단일키 암호화 기법이라고도 한다.
- 개인키 암호화 기법은 한 번에 하나의 데이터 블록을 암호화 하는 블록 암호화 방식과, 평문과 동일한 길이의 스트림을 생성하여 비트 단위로 암호화 하는 스트림 암호화 방식으로 분류된다.

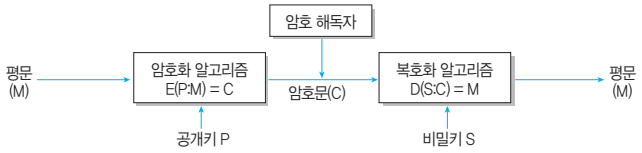
시험에 나오는 것만 공부한다! 시나공시리즈

- 종류
  - 블록 암호화 방식 : DES, SEED, AES, ARIA
  - 스트림 암호화 방식 : LFSR, RC4

핵심 357 공개키 암호화(Public Key Encryption) 기법

공개키 암호화 기법은 데이터를 암호화할 때 사용하는 공개키(Public Key)는 데이터베이스 사용자에게 공개하고, 복호화할 때의 비밀키(Secret Key)는 관리자가 비밀리에 관리한다.

- 데이터베이스 사용자는 평문의 정보 M을 암호화 알고리즘 E와 공개키(Public Key) P를 이용하여 암호문 C로 바꾸어 저장시켜 놓고, 이를 복호화하기 위해서는 비밀키와 복호화 알고리즘에 권한이 있는 사용자만이 복호화 알고리즘 D와 비밀키(Secret Key) S를 이용하여 다시 평문의 정보 M으로 바꿀 수 있는 기법이다.



- 공개키 암호화 기법은 비대칭 암호 기법이라고도 하며, 대표적으로는 RSA(Rivest Shamir Adleman) 기법이 있다.

핵심 358 양방향 알고리즘 종류

개인키 암호화 방식과 공개키 암호화 방식에서 사용되는 주요 암호화 알고리즘에는 SEED, ARIA 등이 있다.

SEED	• 1999년 한국인터넷진흥원(KISA)에서 개발한 블록 암호화 알고리즘 • 블록 크기는 128비트이며, 키 길이에 따라 128, 256으로 분류된다.
ARIA(Academy, Research Institute, Agency)	• 2004년 국가정보원과 산학연협회가 개발한 블록 암호화 알고리즘 • ARIA는 학계(Academy), 연구기관(Research Institute), 정부(Agency)의 영문 앞 글자로 구성되었다. • 블록 크기는 128비트이며, 키 길이에 따라 128, 192, 256으로 분류된다.
DES(Data Encryption Standard)	• 1975년 미국 NBS에서 발표한 개인키 암호화 알고리즘 • DES를 3번 적용하여 보안을 더욱 강화한 3DES(Triple DES)도 있다. • 블록 크기는 64비트이며, 키 길이는 56비트이다.
AES(Advanced Encryption Standard)	• 2001년 미국 표준 기술 연구소(NIST)에서 발표한 개인키 암호화 알고리즘 • DES의 한계를 느낀 NIST에서 공모한 후 발표하였다. • 블록 크기는 128비트이며, 키 길이에 따라 128, 192, 256으로 분류된다.
RSA(Rivest Shamir Adleman)	• 1978년 MIT의 라이베스트(Rivest), 샤미르(Shamir), 애들먼(Adelman)에 의해 제안된 공개키 암호화 알고리즘 • 큰 숫자를 소인수분해 하기 어렵다는 것에 기반하여 만들어졌다. • 공개키와 비밀키를 사용하는데, 여기서 키란 메시지를 열고 잠그는 상수(Constant)를 의미한다.

핵심 359 해시(Hash)

해시는 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것을 의미한다.

- 해시 알고리즘을 해시 함수라고 부르며, 해시 함수로 변환된 값이나 키를 해시값 또는 해시키라고 부른다.
- 데이터의 암호화, 무결성 검증을 위해 사용될 뿐만 아니라 정보보호의 다양한 분야에서 활용된다.
- 해시 함수의 종류에는 SHA 시리즈, MD5, N-NASH, SNEFRU 등이 있다.

SHA 시리즈	• 1993년 미국 국가안보국(NSA)이 처음 설계했으며, 미국 국립표준기술연구소(NIST)에 의해 발표되었다. • 초기 개발된 SHA-0 이후 SHA-1이 발표되었고, 다시 SHA-2라고 불리는 SHA-224, SHA-256, SHA-384, SHA-512가 발표되었다.
MD5	• 1991년 R.Rivest가 MD4를 대체하기 위해 고안한 암호화 해시 함수이다. • 블록 크기는 512비트이며, 키 길이는 128비트이다.
N-NASH	• 1989년 일본의 전신전화주식회사(NTT)에서 발표한 암호화 해시 함수이다. • 블록 크기와 키 길이가 모두 128비트이다.
SNEFRU	• 1990년 R.C.Merkle가 발표한 해시 함수이다. • 32비트 프로세서에서 구현을 용이하게 할 목적으로 개발되었다. • 블록 크기는 512비트이며, 키 길이에 따라 128과 256으로 분류된다.

핵심 360 SMURFING(스머핑)

SMURFING은 IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄으로써 네트워크를 불능 상태로 만드는 공격 방법이다.

- 공격자는 송신 주소를 공격 대상지의 IP 주소로 위장하고 해당 네트워크 라우터의 브로드캐스트 주소를 수신지로 하여 패킷을 전송하면, 라우터의 브로드캐스트 주소로 수신된 패킷은 해당 네트워크 내의 모든 컴퓨터로 전송된다.
- 해당 네트워크 내의 모든 컴퓨터는 수신된 패킷에 대한 응답 메시지를 송신 주소인 공격 대상지로 집중적으로 전송하게 되는데, 이로 인해 공격 대상지는 네트워크 과부하로 인해 정상적인 서비스를 수행할 수 없게 된다.
- SMURFING 공격을 무력화하는 방법 중 하나는 각 네트워크 라우터에서 브로드캐스트 주소를 사용할 수 없게 미리 설정해 놓는 것이다.

핵심 361 DDoS(Distributed Denial of Service, 분산 서비스 거부) 공격

DDoS 공격은 여러 곳에 분산된 공격 지점에서 한 곳의 서버에 대해 분산 서비스 공격을 수행하는 것으로, 네트워크에서 취약점이 있는 호스트들을 탐색한 후 이들 호스트들에 분산 서비스 공격용 툴을 설치하여 에이전트(Agent)로 만든 후 DDoS 공격에 이용한다.

- 공격의 범위를 확대하기 위해 일부 호스트에 다수의 에이전트를 관리할 수 있는 핸들러(Handler) 프로그램을 설치하여 마스터(Master)로 지정한 후 공격에 이용하기도 한다.

분산 서비스 공격용 툴

에이전트(Agent)의 역할을 수행하도록 설계된 프로그램으로 데몬(Daemon)이라고 부르며, 다음과 같은 종류가 있다.

- Trin00 : 가장 초기 형태의 데몬으로, 주로 UDP Flooding 공격을 수행함
- TFN(Tribe Flooding Network) : UDP Flooding 뿐만 아니라 TCP SYN Flood 공격, ICMP 응답 요청, 스머핑 공격 등을 수행함
- TFN2K : TFN의 확장판
- Stacheldraht : 이전 툴들의 기능을 유지하면서, 공격자, 마스터, 에이전트가 쉽게 노출되지 않도록 암호화된 통신을 수행하며, 툴이 자동으로 업데이트되도록 설계됨

핵심 362 정보 보안 침해 공격 관련 용어

용어	의미
좀비(Zombie) PC	악성코드에 감염되어 다른 프로그램이나 컴퓨터를 조종하도록 만들어진 컴퓨터로, C&C(Command & Control) 서버의 제어를 받아 주로 DDoS 공격 등에 이용된다.
C&C 서버	해커가 원격지에서 감염된 좀비 PC에 명령을 내리고 악성코드를 제어하기 위한 용도로 사용하는 서버를 말한다.
봇넷(Botnet)	악성 프로그램에 감염되어 악의적인 의도로 사용될 수 있는 다수의 컴퓨터들이 네트워크로 연결된 형태를 말한다.
웜(Worm)	네트워크를 통해 연속적으로 자신을 복제하여 시스템의 부하를 높임으로써 결국 시스템을 다운시키는 바이러스의 일종으로, 분산 서비스 거부 공격, 버퍼 오버플로 공격, 슬래머 등이 웜 공격의 한 형태이다.
제로 데이 공격 (Zero Day Attack)	보안 취약점이 발견되었을 때 발견된 취약점의 존재 자체가 널리 공표되기도 전에 해당 취약점을 통하여 이루어지는 보안 공격으로, 공격의 신속성을 의미한다.
키로거 공격 (Key Logger Attack)	컴퓨터 사용자의 키보드 움직임을 탐지해 ID, 패스워드, 계좌번호, 카드번호 등과 같은 개인의 중요한 정보를 몰래 빼가는 해킹 공격
랜섬웨어 (Ransomware)	인터넷 사용자의 컴퓨터에 잠입해 내부 문서나 파일 등을 암호화해 사용자가 열지 못하게 하는 프로그램으로, 암호 해독용 프로그램의 전달을 조건으로 사용자에게 돈을 요구하기도 한다.
백도어 (Back Door, Trap Door)	• 시스템 설계자가 서비스 기술자나 유지 보수 프로그램 작성자(Programmer)의 액세스 편의를 위해 시스템 보안을 제거하여 만들어놓은 비밀 통로로, 컴퓨터 범죄에 악용되기도 한다. • 백도어 탐지 방법 : 무결성 검사, 열린 포트 확인, 로그 분석, SetUID 파일 검사 등
트로이 목마 (Trojan Horse)	정상적인 기능을 하는 프로그램으로 위장하여 프로그램 내에 숨어 있다가 해당 프로그램이 동작할 때 활성화되어 부작용을 일으키는 것으로, 자기 복제 능력은 없다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 363 보안 서버의 개념

보안 서버란 인터넷을 통해 개인정보를 암호화하여 송 · 수신할 수 있는 기능을 갖춘 서버를 말한다.

- '개인정보의 기술적 · 관리적 보호조치 기준'에 따르면 보안 서버는 다음과 같은 기능을 갖춰야 한다.
  - 서버에 SSL(Secure Socket Layer) 인증서를 설치하여 전송 정보를 암호화하여 송 · 수신하는 기능
  - 서버에 암호화 응용 프로그램을 설치하고 전송 정보를 암호화하여 송 · 수신하는 기능
- 스니핑(Sniffing)을 이용한 정보 유출, 피싱(Phishing)을 이용한 위조 사이트 등에 대비하기 위해 보안 서버 구축이 필요하다.

핵심 364 인증(認證, Authentication)의 개념

인증은 다중 사용자 컴퓨터 시스템이나 네트워크 시스템에서 로그인을 요청한 사용자의 정보를 확인하고 접근 권한을 검증하는 보안 절차이다.

- 인증에는 네트워크를 통해 컴퓨터에 접속하는 사용자의 등록 여부를 확인하는 것과 전송된 메시지의 위 · 변조 여부를 확인하는 것이 있다.
- 인증의 주요 유형
  - 지식 기반 인증(Something You Know) : 사용자가 기억하고 있는 정보를 기반으로 인증을 수행하는 것
  - 소유 기반 인증(Something You Have) : 사용자가 소유하고 있는 것을 기반으로 인증을 수행하는 것
  - 생체 기반 인증(Something You Are) : 사용자의 고유한 생체 정보를 기반으로 인증을 수행하는 것
  - 위치 기반 인증(Somewhere You Are) : 인증을 시도하는 위치의 적절성 확인하는 것

핵심 365 보안 아키텍처(Security Architecture)

보안 아키텍처란 정보 시스템의 무결성(Integrity), 가용성(Availability), 기밀성(Confidentiality)을 확보하기 위해 보안 요소 및 보안 체계를 식별하고 이들 간의 관계를 정의한 구조를 말한다.

- 보안 아키텍처를 통해 관리적, 물리적, 기술적 보안 개념의 수립, 보안 관리 능력의 향상, 일관된 보안 수준의 유지를 기대할 수 있다.
- 보안 아키텍처는 보안 수준에 변화가 생겨도 기본 보안 아키텍처의 수정 없이 지원할 수 있어야 한다.
- 보안 아키텍처는 보안 요구사항의 변화나 추가를 수용할 수 있어야 한다.

핵심 366 보안 프레임워크(Security Framework)

프레임워크는 '뼈대', '골조'를 의미하는 용어이며, 보안 프레임워크는 안전한 정보 시스템 환경을 유지하고 보안 수준을 향상시키기 위한 체계를 말한다.

- ISO 27001은 정보보안 관리를 위한 국제 표준으로, 일종의 보안 인증이자 가장 대표적인 보안 프레임워크이다.
- ISO 27001은 영국의 BSI(British Standards Institute)가 제정한 BS 7799를 기반으로 구성되어 있다.
- ISO 27001은 조직에 대한 정보보안 관리 규격이 정의되어 있어 실제 심사/인증용으로 사용된다.

핵심 367 리눅스(LINUX) 로그

리눅스에서는 시스템의 모든 로그를 var/log 디렉터리에서 기록하고 관리한다.

- 로그 파일을 관리하는 syslogd 데몬은 etc/syslog.conf 파일을 읽어 로그 관련 파일들의 위치를 파악한 후 로그 작업을 시작한다.
- syslog.conf 파일을 수정하여 로그 관련 파일들의 저장 위치와 파일명을 변경할 수 있다.

시험에 나오는 것만 공부한다! 시나공시리즈

핵심 368 리눅스의 주요 로그 파일

로그	파일명	데몬	내용
커널 로그	/dev/console	kernel	커널에 관련된 내용을 관리자에게 알리기 위해 파일로 저장하지 않고 지정된 장치에 표시한다.
부팅 로그	/var/log/boot.log	boot	부팅 시 나타나는 메시지들을 기록한다.
크론 로그	/var/log/cron	crond	작업 스케줄러인 crond의 작업 내역을 기록한다.
시스템 로그	/var/log/messages	syslogd	커널(kernel)에서 실시간으로 보내오는 메시지들을 기록한다.
보안 로그	/var/log/secure	xinetd	시스템의 접속에 대한 로그를 기록한다.
FTP 로그	/var/log/xferlog	ftpd	FTP로 접속하는 사용자에 대한 로그를 기록한다.
메일 로그	/var/log/maillog	sendmail popper	송수신 메일에 대한 로그를 기록한다.

핵심 369 Windows 이벤트 뷰어의 로그

로그	내용
응용 프로그램	• 응용 프로그램에서 발생하는 이벤트가 기록된다. • 기록되는 이벤트는 응용 프로그램 개발자에 의해 결정된다.
보안	로그온 시도, 파일이나 객체 생성, 조회, 제거 등의 리소스 사용과 관련된 이벤트가 기록된다.
시스템	Windows 시스템 구성 요소에 의해 발생하는 이벤트가 기록된다.
Setup	프로그램 설치와 관련된 이벤트가 기록된다.
Forwarded Events	다른 컴퓨터와의 상호 작용으로 발생하는 이벤트가 기록된다.

핵심 370 보안 솔루션

보안 솔루션의 개념

보안 솔루션이란 접근 통제, 침입 차단 및 탐지 등을 수행하여 외부로부터의 불법적인 침입을 막는 기술 및 시스템을 말한다.

- 주요 보안 솔루션에는 방화벽, 침입 탐지 시스템(IDS), 침입 방지 시스템(IPS), 데이터 유출 방지(DLP), 웹 방화벽, VPN, NAC 등이 있다.

방화벽(Firewall)	기업이나 조직 내부의 네트워크와 인터넷 간에 전송되는 정보를 선별하여 수용 · 거부 · 수정하는 기능을 가진 침입 차단 시스템이다.
침입 탐지 시스템 (IDS; Intrusion Detection System)	컴퓨터 시스템의 비정상적인 사용, 오용, 남용 등을 실시간으로 탐지하는 시스템이다.
침입 방지 시스템 (IPS; Intrusion Prevention System)	• 방화벽과 침입 탐지 시스템을 결합한 것이다. • 비정상적인 트래픽을 능동적으로 차단하고 격리하는 등의 방어 조치를 취하는 보안 솔루션이다.
데이터 유출 방지(DLP; Data Leakage/Loss Prevention)	데이터 유출 방지는 내부 정보의 외부 유출을 방지하는 보안 솔루션이다.
웹 방화벽(Web Firewall)	일반 방화벽이 탐지하지 못하는 SQL 삽입 공격, Cross-Site Scripting(XSS) 등의 웹 기반 공격을 방어할 목적으로 만들어진 웹 서버에 특화된 방화벽이다.
VPN(Virtual Private Network, 가상 사설 통신망)	가상 사설 네트워크로서 인터넷 등 통신 사업자의 공중 네트워크와 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안 솔루션이다.
NAC(Network Access Control)	네트워크에 접속하는 내부 PC의 MAC 주소를 IP 관리 시스템에 등록한 후 일관된 보안 관리 기능을 제공하는 보안 솔루션이다.
ESM(Enterprise Security Management)	다양한 장비에서 발생하는 로그 및 보안 이벤트를 통합하여 관리하는 보안 솔루션이다.

불합격 방지용 안전장치 기억상자
틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.