2026 시나공

기출문제집

정보처리기사

실기

핵심 요약

1장 요구사항 확인

2장 데이터 입·출력 구현

3장 데이터 입·출력 구현

4장 서버 프로그램 구현

5장 인터페이스 구현

6장 화면 설계

7장 애플리케이션 테스트 관리

8장 SQL 응용

9장 소프트웨어 개발 보안 구축

10장 프로그래밍 언어 활용

11장 응용 SW 기초 기술 활용

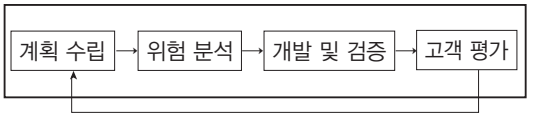
12장 제품 소프트웨어 패키징

1장 요구사항 확인

340002 필기 25.8, 25.5, 25.2, 24.5, 23.7, 23.5, 23.2, 22.7, 21.8, 21.3, 20.9, 20.8, 20.6

001 나선형 모형 B

- 나선형 모형(Spiral Model, 점진적 모형)은 나선을 따라 돌듯이 **여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로** 완벽한 최종 소프트웨어를 **개발하는 모형**이다.
- 보헴(Boehm)이 제안하였다.
- 4가지 주요 활동



340003 필기 24.7, 24.2, 21.8, 20.9, 20.8, 20.6

002 폭포수 모형 B

- 폭포수 모형(Waterfall Model)은 이전 단계로 돌아갈 수 없다는 전제하에 **각 단계를 확실히 매듭짓고 그 결과를** 철저하게 **검토하여 승인 과정을 거친 후에 다음 단계를 진행하는 개발 방법론**이다.
- 가장 오래되고 가장 폭넓게 사용된 전통적인 소프트웨어 생명 주기 모형이다.
- 고전적 생명 주기 모형이라고도 한다.

궁금해요 ?

Q 실기 책에 왜 필기의 기출 년월이 표시되어 있나요?

A 정보처리기사 시험은 필기와 실기가 시험 범위가 같습니다. 동일한 내용이 객관식으로 필기시험에 나올 수도 있고, 단답형이나 서술식으로 실기시험에 나올 수도 있습니다. 공부하다 보면 알겠지만 필기시험과 실기시험에 중복해서 나온 내용이 많습니다. 자격 시험에는 나온 문제가 또 나온다는 걸 명심하세요.

340005 20.7, 필기 25.8, 23.7, 22.4, 22.3, 21.8, 21.5, 20.8

003 애자일 모형 A

- 애자일(Agile)은 '민첩한', '기민한'이라는 의미로, 고객의 **요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발하는 모형**이다.
- 어느 특정 개발 방법론이 아니라 좋은 것을 빠르고 낭비 없게 만들기 위해 고객과의 소통에 초점을 맞춘 방법론을 통칭한다.
- 대표적인 개발 모형
  - 스크럼(Scrum)
  - XP(eXtreme Programming)
  - 칸반(Kanban)
  - Lean
  - 기능 중심 개발(FDD; Feature Driven Development)

340006 필기 25.8, 24.7, 23.2, 22.3, 21.3, 20.8

004 애자일 개발 4가지 핵심 가치 B

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

340010

필기 25.8, 25.2, 24.5, 24.2, 23.7, 23.5, 23.2, 22.7, 22.4, 20.9, 20.6

005 XP Ⓑ

- XP(eXtreme Programming)는 수시로 발생하는 고객의 **요구사항에 유연하게 대응하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 개발 생산성을 향상시키는 방법**이다.
- 짧고 반복적인 개발 주기, 단순한 설계, 고객의 적극적인 참여를 통해 소프트웨어를 빠르게 개발하는 것을 목적으로 한다.
- XP의 5가지 핵심 가치
  - 의사소통(Communication)
  - 단순성(Simplicity)
  - 용기(Courage)
  - 존중(Respect)
  - 피드백(Feedback)

440009

20.10, 필기 24.7, 24.5, 22.4, 20.9

006 XP의 주요 실천 방법(Practice) Ⓐ

필기 20.9

Pair Programming(짝 프로그래밍)

다른 사람과 함께 프로그래밍을 수행함으로써 개발에 대한 책임을 공동으로 나눠 갖는 환경을 조성한다.

필기 20.9

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.

필기 20.9

Continuous Integration(계속적인 통합)

모듈 단위로 나눠서 개발된 코드들은 하나의 작업이 마무리될 때마다 지속적으로 통합된다.

20.10, 필기 24.7, 24.5, 22.4

Refactoring(리팩토링)

- 프로그램 기능의 변경 없이 시스템을 재구성한다.
- **목적** : 프로그램을 쉽게 이해하고 쉽게 수정하여 빠르게 개발할 수 있도록 하기 위함임

Small Releases(소규모 릴리즈)

릴리즈 기간을 짧게 반복함으로써 고객의 요구 변화에 신속히 대응할 수 있다.

340016

21.4, 필기 22.4

007 기능 요구사항 Ⓐ

- 기능 요구사항(Functional Requirements)은 시스템이 무엇을 하는지, 어떤 기능을 하는지 등의 기능이나 수행과 관련된 요구사항이다.
- 시스템의 입력이나 출력으로 무엇이 포함되어야 하는지에 대한 사항
- 시스템이 어떤 데이터를 저장하거나 연산을 수행해야 하는지에 대한 사항
- 시스템이 반드시 수행해야 하는 기능
- 사용자가 시스템을 통해 제공받기를 원하는 기능

340017

21.4, 필기 25.5, 24.5, 23.2, 22.4, 21.8

008 비기능 요구사항 Ⓐ

- 비기능 요구사항(Non-functional Requirements)은 품질이나 제약사항과 관련된 요구사항이다.
- 시스템 장비 구성 요구사항
- 성능 요구사항
- 인터페이스 요구사항
- 데이터를 구축하기 위해 필요한 요구사항
- 테스트 요구사항
- 보안 요구사항
- **품질 요구사항** : 가용성, 정합성, 상호 호환성, 대응성, 이식성, 확장성, 보안성 등

440015

필기 24.7, 24.2, 22.3, 20.6

009 요구사항 분석 C

- 요구사항 분석(Requirement Analysis)은 소프트웨어 개발의 실제적인 첫 단계로, **개발 대상에 대한 사용자의 요구사항을 이해하고 문서화하는 활동**을 의미한다.
- 사용자 요구의 타당성을 조사하고 비용과 일정에 대한 제약을 설정한다.
- 사용자의 요구를 정확하게 추출하여 목표를 정한다.

340023

필기 25.5, 25.2, 24.5, 23.2, 22.7, 22.3, 20.9, 20.8, 20.6

010 자료 흐름도의 구성 요소 A

기호	의미
필기 25.5, 25.2, 24.5, 23.2, 22.7, … 프로세스(Process)	자료를 변환시키는 시스템의 한 부분(처리 과정)을 나타내며 처리, 기능, 변환, 버블이라고도함
필기 25.5, 25.2, 24.5, 23.2, 22.7, … 자료 흐름(Data Flow)	자료의 이동(흐름)이나 연관관계를 나타냄
필기 25.5, 25.2, 24.5, 23.2, 22.7, … 자료 저장소(Data Store)	시스템에서의 자료 저장소(파일, 데이터베이스)를 나타냄
필기 25.5, 24.5, 23.2, 22.7, 22.3, … 단말(Terminator)	시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고 출력 데이터를 받음

340027

필기 25.2, 24.2, 23.7, 23.2, 22.7, 20.9

011 HIPO B

- HIPO(Hierarchy Input Process Output)는 시스템의 분석 및 설계, 또는 문서화에 사용되는 기법으로, **시스템 실행 과정인 입력 · 처리 · 출력의 기능을 표현한 것**이다.
- 하향식 소프트웨어 개발을 위한 문서화 도구이다.
- 시스템의 기능을 여러 개의 고유 모듈로 분할하여 이들 간의 인터페이스를 계층 구조로 표현한 것을 HIPO Chart라고 한다.

340030

24.10

012 연관 관계 A

- 연관(Association) 관계는 **2개 이상의 사물이 서로 관련되어 있는 관계**이다.
- 사물 사이를 실선으로 연결하여 표현한다.
- 방향성은 화살표로 표현한다.
- 양방향 관계의 경우 화살표를 생략하고 실선으로만 연결한다.
- 다중도를 선 위에 표기한다.

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

340031

21.10

013 집합 관계 A

- 집합(Aggregation) 관계는 **하나의 사물이 다른 사물에 포함되어 있는 관계**이다.
- 포함하는 쪽(전체, Whole)과 포함되는 쪽(부분, Part)은 서로 독립적이다.
- 포함되는 쪽(부분, Part)에서 포함하는 쪽(전체, Whole)으로 속이 빈 마름모를 연결하여 표현한다.

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

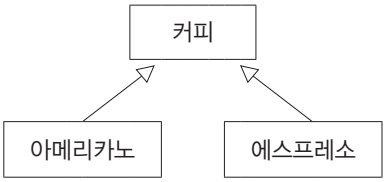
340033

24.10, 21.10, 필기 24.2, 20.8

014 일반화 관계 Ⓐ

- 일반화(Generalization) 관계는 **하나의 사물이 다른 사물에 비해 더 일반적이거나 구체적인 관계**이다.
- 보다 일반적인 개념을 상위(부모), 보다 구체적인 개념을 하위(자식)라고 부른다.
- 구체적(하위)인 사물에서 일반적(상위)인 사물 쪽으로 속이 빈 화살표를 연결하여 표현한다.

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



340034

24.10, 필기 23.5, 22.7, 21.8

015 의존 관계 Ⓐ

- 의존(Dependency) 관계는 연관 관계와 같이 사물 사이에 서로 연관은 있으나 필요에 의해 **서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계**이다.
- 하나의 사물과 다른 사물이 소유 관계는 아니지만 사물의 변화가 다른 사물에도 영향을 미치는 관계이다.
- 일반적으로 한 클래스가 다른 클래스를 오퍼레이션의 매개 변수로 사용하는 경우에 나타나는 관계이다.
- 영향을 받는 사물(이용자)이 영향을 주는 사물(제공자) 쪽으로 점선 화살표를 연결하여 표현한다.

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

340037

필기 25.8, 23.5, 23.2, 22.3, 21.5, 21.3, 20.6

016 구조적 다이어그램의 종류 Ⓑ

필기 25.8, 23.2, 21.5, 21.3, 20.6

클래스 다이어그램(Class Diagram)

클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다.

필기 20.6

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.

필기 23.5, 22.3, 20.6

컴포넌트 다이어그램(Component Diagram)

- 실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현한다.
- 구현 단계에서 사용된다.

필기 22.3

배치 다이어그램(Deployment Diagram)

- 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현한다.
- 구현 단계에서 사용된다.

복합체 구조 다이어그램(Composite Structure Diagram)

클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현한다.

필기 22.3

패키지 다이어그램(Package Diagram)

유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현한다.

340038

필기 25.8, 23.5, 23.2, 22.3, 21.5, 21.3, 20.9, 20.8

017 행위 다이어그램의 종류 Ⓑ

필기 23.5, 20.8

유스케이스 다이어그램(Use Case Diagram)

- 사용자의 요구를 분석하는 것으로, 기능 모델링 작업에 사용한다.
- 사용자(Actor)와 사용 사례(Use Case)로 구성된다.

필기 25.8, 23.5, 23.2, 22.3, 21.5, 21.3, 20.8

순차 다이어그램(Sequence Diagram)

상호작용하는 시스템이나 객체들이 주고받는 메시지를 표현한다.

커뮤니케이션 다이어그램(Communication Diagram)

동작에 참여하는 객체들이 주고받는 메시지와 객체들 간의 연관 관계를 표현한다.

필기 25.8, 23.5, 21.3, 20.9

상태 다이어그램(State Diagram)

- 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호작용에 따라 상태가 어떻게 변화하는지를 표현한다.
- 럼바우(Rumbaugh) 객체지향 분석 기법에서 동적 모델링에 활용된다.

필기 25.8, 23.2, 21.5, 20.8

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.

340040

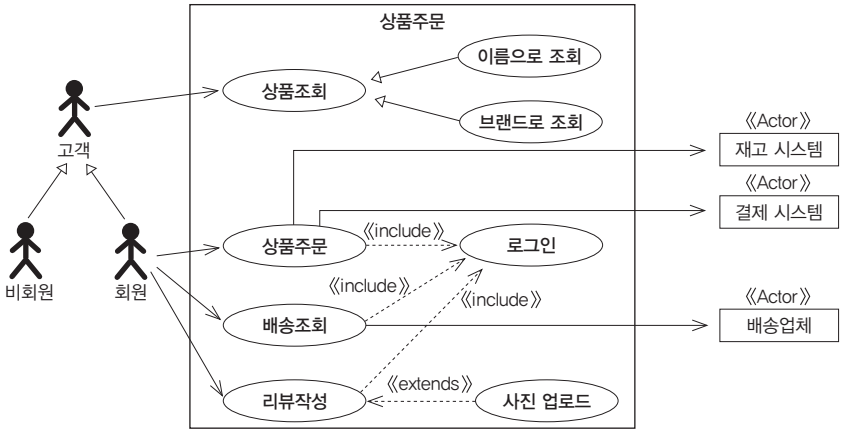
필기 25.2, 24.5, 23.7, 23.5, 22.4, 21.5

018 유스케이스 다이어그램 Ⓑ

- 유스케이스(Use Case) 다이어그램은 사용자와 다른 외부 시스템들이 **개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점에서 표현한 것**이다.
- 유스케이스 다이어그램의 구성 요소

구성 요소	표현 방법	내용
시스템(System) / 시스템 범위(System Scope)	상품주문	시스템 내부의 유스케이스들을 사각형으로 묶어 시스템의 범위를 표현한 것
필기 21.5 액터(Actor)	• 주액터  고객 • 부액터 《Actor》 재고 시스템	• 시스템과 상호작용을 하는 모든 외부 요소 • 주로 사람이나 외부 시스템을 의미함 • **주액터** : 시스템을 사용함으로써 이득을 얻는 대상으로, 주로 사람이 해당됨 • **부액터** : 주액터의 목적 달성을 위해 시스템에 서비스를 제공하는 외부 시스템으로, 조직이나 기관 등이 될 수 있음
유스케이스(Use Case)	상품조회	사용자가 보는 관점에서 시스템이 액터에게 제공하는 서비스나 기능을 표현한 것
관계(Relationship)	• 포함 《include》 • 확장 《extends》 • 일반화	• 유스케이스 다이어그램에서 관계는 액터와 유스케이스, 유스케이스와 유스케이스 사이에서 나타날 수 있음 • **유스케이스에서 나타날 수 있는 관계** : 포함(Include) 관계, 확장(Extends) 관계, 일반화(Generalization) 관계

예제 다음은 회원과 비회원으로 구분되는 회원이 상품주문 과정에서 수행할 수 있는 기능을 표현한 유스케이스 다이어그램이다.



340043

21.10, 필기 21.8

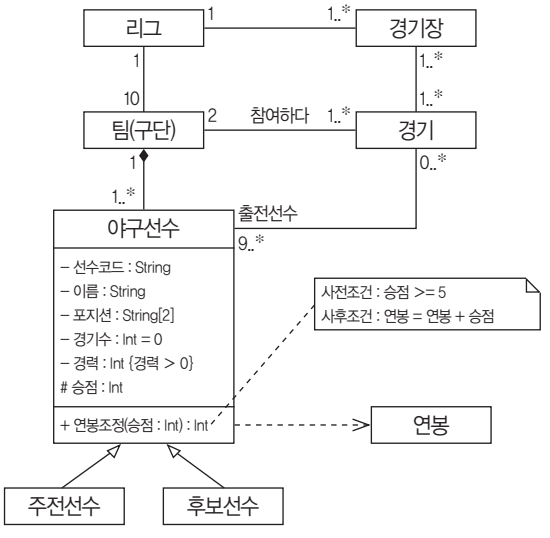
019 클래스 다이어그램

Ⓐ

- 클래스(Class) 다이어그램은 **클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한 것**이다.
- 클래스 다이어그램의 구성 요소

구성 요소	표현 방법	내용
21.10, 필기 21.8 클래스(Class)	클래스명 속성1 속성2 오퍼레이션1 오퍼레이션2	• 각각의 객체들이 갖는 속성과 오퍼레이션(동작)을 표현한 것 • 일반적으로 3개의 구획(Compartment)으로 나눠 클래스의 이름, 속성, 오퍼레이션을 표기함 • **속성(Attribute)** : 클래스의 상태나 정보를 표현함 • **오퍼레이션(Operation)** : 클래스가 수행할 수 있는 동작으로, 함수(메소드, Method)라고도 함
제약조건		• 속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건이 있다면 이를 적음 • 클래스 안에 제약조건을 기술할 때는 중괄호 { }를 이용함
관계(Relationships)		• 관계는 클래스와 클래스 사이의 연관성을 표현함 • 클래스 다이어그램에 표현하는 관계에는 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계가 있음

예제 다음은 프로야구 리그에 필요한 정보의 일부를 표현한 클래스 다이어그램이다.

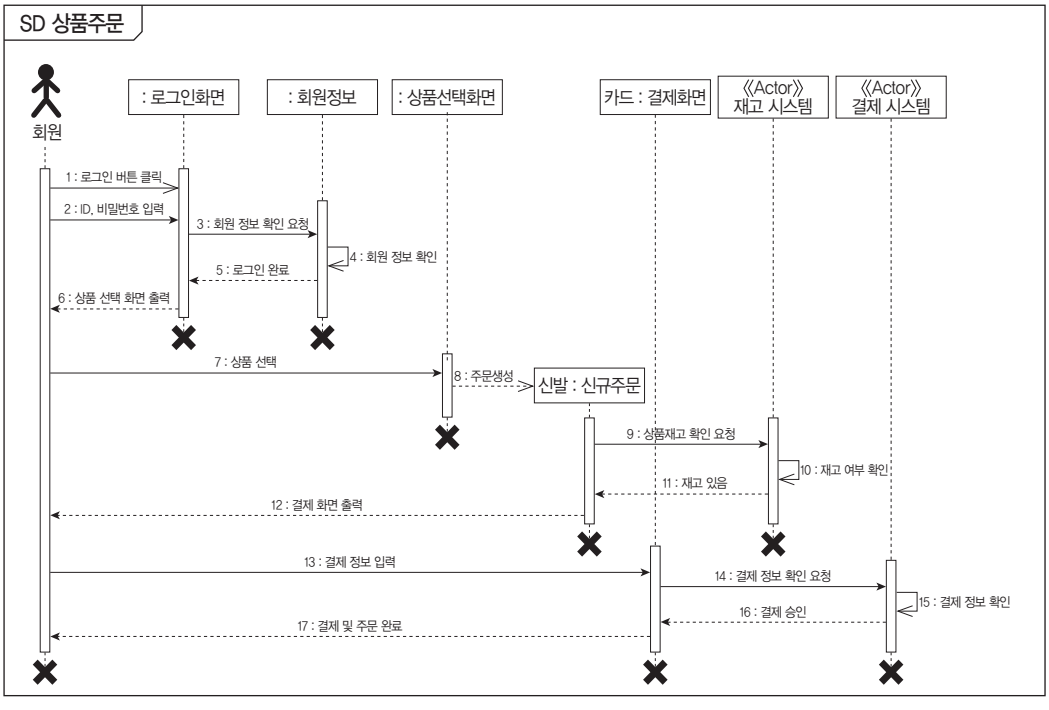


020 순차 다이어그램 B

- 순차(Sequence) 다이어그램은 **시스템이나 객체들이 메시지를 주고받으며 상호작용하는 과정을 그림으로 표현한 것**이다.
- 순차 다이어그램의 구성 요소

구성 요소	표현 방법	의미
액터(Actor)	회원	시스템으로부터 서비스를 요청하는 외부 요소로, 사람이나 외부 시스템을 의미함
객체(Object)	: 로그인화면	메시지를 주고받는 주체
필기 22.7, 20.8 생명선(Lifeline)		• 객체가 메모리에 존재하는 기간으로, 객체 아래쪽에 점선을 그어 표현함 • 객체 소멸(✖)이 표시된 기간까지 존재함
필기 22.7, 20.8 실행 상자 (Active Box, 활성 상자)		객체가 메시지를 주고받으며 구동되고 있음을 표현함
필기 22.7, 20.8 메시지(Message)	1 : 로그인 버튼 클릭	객체가 상호 작용을 위해 주고받는 메시지
객체 소멸	✖	해당 객체가 더 이상 메모리에 존재하지 않음을 표현한 것
프레임(Frame)	SD 상품주문	다이어그램의 전체 또는 일부를 묶어 표현한 것

예제 다음은 회원의 상품 주문 과정에 재고 시스템과 결제 시스템이 관계된 상호 작용 과정을 표현한 순차 다이어그램이다.

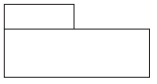


340048

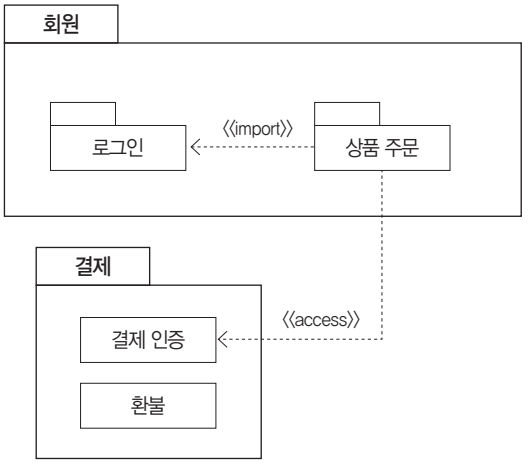
25.11, 23.10, 20.11

021 패키지 다이어그램 Ⓐ

- 패키지(Package) 다이어그램은 유스케이스나 클래스 등의 **요소들을 그룹화한 패키지 간의 의존 관계를 표현한 것**이다.
- 패키지 다이어그램의 구성 요소

구성 요소	표현 방법	의미
패키지(Package)		• 객체들을 그룹화한 것 • **단순 표기법** : 패키지 안에 패키지 이름만 표현 • **확장 표기법** : 패키지 안에 요소까지 표현
객체(Object)		유스케이스, 클래스, 인터페이스, 테이블 등 패키지에 포함될 수 있는 다양한 요소들
의존 관계 (Dependency)		• 패키지와 패키지, 패키지와 객체 간을 점선 화살표로 연결하여 표현함 • 스테레오 타입을 이용해 의존 관계를 구체적으로 표현할 수 있음 • 의존 관계의 표현 형태는 사용자가 임의로 작성할 수 있으며, 대표적으로 import와 access가 사용됨 – 《import》 : 패키지에 포함된 객체들을 직접 가져와서 이용하는 관계 – 《access》 : 인터페이스를 통해 패키지 내의 객체에 접근하여 이용하는 관계

예제 다음은 회원이 상품 주문 시 사용하는 〈회원〉, 〈로그인〉, 〈상품 주문〉, 〈결제〉 패키지들 간의 의존 관계를 표현한 패키지 다이어그램이다.

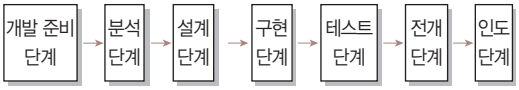


340051

필기 21.3, 20.9

022 컴포넌트 기반 방법론 Ⓒ

- 컴포넌트 기반 방법론(CBD; Component Based Design)은 기존의 시스템이나 소프트웨어를 구성하는 컴포넌트를 조합하여 하나의 **새로운 애플리케이션을 만드는 방법론**이다.
- 컴포넌트의 재사용(Reusability)이 가능하여 시간과 노력을 절감할 수 있다.
- 컴포넌트 기반 방법론의 개발 절차



340054

필기 25.5, 25.2, 23.7, 23.5, 21.3, 20.9, 20.8, 20.6

023 CASE Ⓑ

- 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 것**이다.
- CASE의 주요 기능
  - 소프트웨어 생명 주기 전 단계의 연결
  - 다양한 소프트웨어 개발 모형 지원
  - 그래픽 지원

340057

20.5, 필기 23.2, 22.7, 22.4, 22.3, 21.8, 21.3, 20.6

024 LOC 기법 Ⓐ

- LOC(source Line Of Code, 원시 코드 라인 수) 기법은 소프트웨어 **각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법**이다.
- 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)

340058

필기 21.5, 20.9

025 수학적 산정 기법 Ⓒ

- 수학적 산정 기법은 상향식 비용 산정 기법으로, 경험적 추정 모형, 실험적 추정 모형이라고도 한다.
- 수학적 산정 기법은 개발 비용 산정의 자동화를 목표로 한다.
- 주요 수학적 산정 기법
  - COCOMO 모형
  - Putnam 모형
  - 기능 점수(FP) 모형

440047

필기 25.2, 22.7, 22.4

026 COCOMO 모형 Ⓒ

- COCOMO(COnstructive COst MOdel) 모형은 원시 프로그램의 규모인 **LOC(원시 코드 라인 수)에 의한 비용 산정 기법**이다.
- 개발할 소프트웨어의 규모(LOC)를 예측한 후 이를 소프트웨어 종류에 따라 다르게 책정되는 비용 산정 방정식에 대입하여 비용을 산정한다.
- 비용 산정 결과는 프로젝트를 완성하는 데 필요한 노력(Man-Month)으로 나타난다.
- 보헴(Boehm)이 제안하였다.

340060

필기 25.5, 23.5, 23.2, 22.4, 21.8, 21.5, 21.3, 20.8, 20.6

027 COCOMO의 소프트웨어 개발 유형 Ⓑ

필기 23.5, 23.2, 22.4, 21.8, 21.5, 21.3, 20.8, 20.6

조직형(Organic Mode)

- 기관 내부에서 개발된 중 · 소 규모의 소프트웨어이다.
- 일괄 자료 처리나 과학기술 계산용, 비즈니스 자료 처리용 등의 5만(50KDSI) 라인 이하의 소프트웨어를 개발하는 유형이다.
- 사무 처리용, 업무용, 과학용 응용 소프트웨어 개발에 적합하다.

필기 22.4, 21.5, 21.3, 20.8

반분리형(Semi-Detached Mode)

- 조직형과 내장형의 중간형 소프트웨어이다.
- 트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템 등의 30만(300KDSI) 라인 이하의 소프트웨어를 개발하는 유형이다.
- 컴파일러, 인터프리터와 같은 유틸리티 개발에 적합하다.

필기 25.5, 22.4, 21.5, 21.3, 20.8

내장형(Embedded Mode)

- 초대형 규모의 소프트웨어이다.
- 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.

340061

필기 21.5, 20.6

028 Putnam 모형 Ⓒ

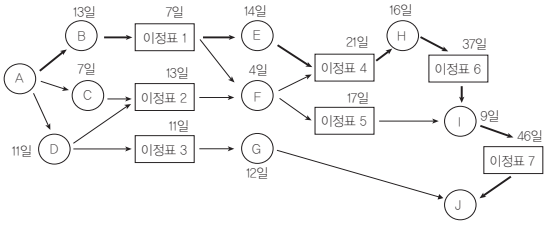
- Putnam 모형은 **소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 예상하는 모형**이다.
- 푸트남(Putnam)이 제안한 것으로, 생명 주기 예측 모형이라고도 한다.
- 시간에 따른 함수로 표현되는 Rayleigh-Norden 곡선의 노력 분포도를 기초로 한다.

340065

필기 25.8, 25.2, 23.7, 23.2, 22.7, 20.8

029 CPM Ⓑ

- CPM(Critical Path Method, 임계 경로 기법))은 프로젝트 완성에 필요한 **작업을 나열하고 작업에 필요한 소요 기간을 예측하는데 사용하는 기법**이다.
- CPM은 노드와 간선으로 구성된 네트워크로 노드는 작업을, 간선은 작업 사이의 전후 의존 관계를 나타낸다.



※ 그림에서 굵은선이 임계 경로, 즉 최장 경로입니다.

340068

필기 25.2, 23.2, 20.9, 20.6

030 CMMI Ⓑ

- CMMI(Capability Maturity Model Integration)는 **소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델**이다.
- CMMI의 소프트웨어 프로세스 성숙도

단계	특징
초기(Initial)	작업자 능력에 따라 성공 여부 결정
필기 23.2, 20.9, 20.6 관리(Managed)	특정한 프로젝트 내의 프로세스 정의 및 수행
필기 23.2, 20.9, 20.6 정의(Defined)	조직의 표준 프로세스를 활용하여 업무 수행
정량적 관리 (Quantitatively Managed)	프로젝트를 정량적으로 관리 및 통제
필기 20.9, 20.6 최적화 (Optimizing)	프로세스 역량 향상을 위해 지속적인 프로세스 개선

340069

필기 20.9, 20.8

031 SPICE Ⓒ

- SPICE(Software Process Improvement and Capability dEtermination)는 정보 시스템 분야에서 **소프트웨어의 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준**이다.
- 공식 명칭은 ISO/IEC 15504이다.

340072

필기 23.7, 21.8, 20.9

032 소프트웨어 개발 프레임워크 Ⓑ

- 소프트웨어 개발 프레임워크(Framework)는 **소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여** 손쉽게 구현할 수 있도록 여러 가지 기능들을 **제공해주는 반제품 형태의 소프트웨어 시스템**이다.
- 선행 사업자의 기술에 의존하지 않는 표준화된 개발 기반으로 인해 사업자 종속성이 해소된다.

340074

필기 22.7, 21.5, 20.9, 20.6

033 소프트웨어 개발 프레임워크의 특성 Ⓒ

필기 22.7, 21.5, 20.9, 20.6

모듈화(Modularity)

- 프레임워크는 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상시킨다.
- 프레임워크는 개발 표준에 의한 모듈화로 인해 유지보수가 용이하다.

필기 22.7, 21.5, 20.9, 20.6

재사용성(Reusability)

프레임워크는 재사용 가능한 모듈들을 제공함으로써 예산 절감, 생산성 향상, 품질 보증이 가능하다.

필기 22.7, 21.5

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.

2장 데이터 입·출력 구현

440063

034 스키마 Ⓐ

23.4, 20.10, 필기 25.5, 21.3, 20.9

스키마(Schema)는 **데이터베이스의 구조와 제약조건에 관한 전반적인 명세를 기술**한 것이다.

종류	내용
23.4 외부 스키마	사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한 것
23.4, 필기 25.5, … 개념 스키마	• 데이터베이스의 전체적인 논리적 구조 • 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스로, 하나만 존재함
23.4, 필기 20.9 내부 스키마	• 물리적 저장장치의 입장에서 본 데이터베이스 구조 • 실제로 저장될 레코드의 형식, 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타냄

340081

035 데이터베이스 설계 순서 Ⓐ

23.7, 20.7

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

340082

036 개념적 설계 Ⓐ

21.4, 필기 23.2, 22.4

- 개념적 설계(정보 모델링, 개념화)는 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 **현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정**이다.
- 개념적 설계에서는 개념 스키마 모델링과 트랜잭션 모델링을 병행 수행한다.
- 개념적 설계에서는 요구 분석에서 나온 결과인 요구조건 명세를 DBMS에 독립적인 E-R 다이어그램으로 작성한다.

340083

037 논리적 설계 Ⓐ

21.4, 필기 24.7, 22.7, 20.6

- 논리적 설계(데이터 모델링)는 **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 과정**이다.
- 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.

340084 21.4, 필기 24.5, 24.2, 22.4, 22.3, 21.8, 21.5, 21.3, 20.9

038 물리적 설계 A

- 물리적 설계(데이터 구조화)는 **논리적 설계에서 논리적 구조로 표현된 데이터를** 디스크 등의 물리적 저장장치에 저장할 수 있는 **물리적 구조의 데이터로 변환하는 과정**이다.
- 물리적 설계에서는 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정한다.
- 저장 레코드의 형식, 순서, 접근 경로, 조회 집중 레코드 등의 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.

340085 21.4, 필기 25.8, 24.5, 23.2, 22.4, 20.9

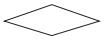
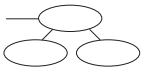
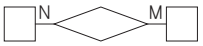
039 데이터 모델 A

- 데이터 모델은 **현실 세계의 정보들을** 컴퓨터에 표현하기 위해서 단순화, 추상화하여 **체계적으로 표현한 개념적 모형**이다.
- 데이터 모델에 표시할 요소

요소	내용
21.4, 필기 24.5, 23.2, 20.9 구조(Structure)	논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질 표현
21.4, 필기 25.8, 24.5, 23.2, … 연산(Operation)	데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구
21.4, 필기 24.5, 23.2, 20.9 제약 조건(Constraint)	데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

340089 필기 25.5, 24.7, 24.5, 23.7, 22.7, 22.3, 21.5, 21.3, 20.9, 20.6

040 E-R 다이어그램 B

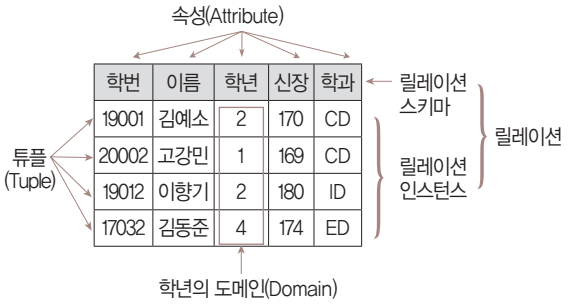
기호	기호 이름	의미
필기 25.5, 24.7, 24.5, 23.7, 22.7, … 	사각형	개체(Entity) 타입
필기 24.7, 24.5, 23.7, 21.5, 21.3, … 	마름모	관계(Relationship) 타입
필기 25.5, 24.7, 24.5, 23.7, 21.5, … 	타원	속성(Attribute)
필기 22.3 	이중 타원	다중값 속성(복합 속성)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 성명은 성과 이름으로 구성
	관계	1:1, 1:N, N:M 등의 개체 간 관계에 대한 대응수를 선 위에 기술함
필기 24.7, 24.5, 21.5, 21.3, 20.9, … 	선, 링크	개체 타입과 속성을 연결

440070 25.11, 23.4, 21.4, 필기 24.7, 23.2, 22.3, 21.3, 20.9, 20.8, 20.6

041 관계형 데이터베이스의 릴레이션 구조 A

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

〈학생〉 릴레이션



- 릴레이션 인스턴스 : 데이터 개체를 구성하고 있는 속성들에 데이터 타입이 정의되어 구체적인 데이터 값을 가진 것을 의미함

440071

25.11, 25.4, 24.7, 23.4, 21.4, 필기 25.5, 23.2, 22.4, 22.3, 21.3, 20.9, 20.8

042 튜플 A

- 튜플(Tuple)은 **릴레이션을 구성하는 각각의 행**을 말한다.
- 튜플은 속성의 모임으로 구성된다.
- 파일 구조에서 레코드와 같은 의미이다.
- 튜플의 수를 카디널리티(Cardinality) 또는 기수, 대응수라고 한다.

340092

25.7, 25.4, 24.7, 21.4, 필기 25.5, 23.2, 22.3, 21.5, 21.3, 20.9, 20.8

043 속성 A

- 속성(Attribute)은 **데이터베이스를 구성하는 가장 작은 논리적 단위**이다.
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당된다.
- 속성은 개체의 특성을 기술한다.
- 속성의 수를 디그리(Degree) 또는 차수라고 한다.

340093

25.4, 필기 25.8, 21.3, 20.6

044 도메인 A

- 도메인(Domain)은 **하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합**이다.
- 도메인은 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는 데에도 이용된다.
  - 예 '성별' 애트리뷰트의 도메인은 "남"과 "여"로, 그 외의 값은 입력될 수 없다.

340094

24.10, 22.5, 필기 25.5, 22.7, 22.4, 20.6

045 후보키 A

- 후보키(Candidate Key)는 릴레이션을 구성하는 **속성들 중에서 튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합**이다.
- 기본키로 사용할 수 있는 속성들을 말한다.
- 후보키는 유일성(Unique)과 최소성(Minimality)을 모두 만족시켜야 한다.

22.5, 필기 20.6 **유일성 (Unique)**	하나의 키 값으로 하나의 튜플만을 유일하게 식별할 수 있어야 함
22.5, 필기 20.6 **최소성 (Minimality)**	키를 구성하는 속성 하나를 제거하면 유일하게 식별할 수 없도록 꼭 필요한 최소의 속성으로 구성되어야 함

440076

24.10, 필기 22.7

046 대체키 A

- 대체키(Alternate Key)는 **후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키**를 의미한다.
- 대체키를 보조키라고도 한다.

340097

24.10, 22.5, 필기 25.2, 24.5, 23.7, 22.7, 21.8, 20.9

047 슈퍼키 A

- 슈퍼키(Super Key)는 한 릴레이션 내에 있는 **속성들의 집합으로 구성된 키**를 말한다.
- 릴레이션을 구성하는 모든 튜플 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않는다.
- 슈퍼키는 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족하지만, 최소성은 만족하지 못한다.

340098

25.4, 24.10, 필기 25.8, 25.2, 24.7, 23.5, 22.7, 22.3, 20.6

048 외래키 Ⓐ

- 외래키(Foreign Key)는 **다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합**을 의미한다.
- 한 릴레이션에 속한 속성 A와 참조 릴레이션의 기본키인 B가 동일한 도메인 상에서 정의되었을 때의 속성 A를 외래키라고 한다.
- 외래키로 지정되면 참조 릴레이션의 기본키에 없는 값은 입력할 수 없다.

440079

25.4, 24.10, 23.10, 필기 25.5, 25.2, 24.2, 23.2, 22.7, 22.4, 21.8, 21.5, 21.3, …

049 무결성 Ⓐ

- 무결성(Integrity)은 **데이터베이스에 저장된 데이터 값과** 그것이 표현하는 **현실 세계의 실제값이 일치하는 정확성**을 의미한다.
- **개체 무결성** : 기본 테이블의 기본키를 구성하는 어떤 속성도 Null 값이나 중복값을 가질 수 없다는 규정
- **참조 무결성** : 외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 함. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정
- **도메인 무결성** : 주어진 속성 값이 정의된 도메인에 속한 값이어야 한다는 규정

340100

필기 25.2, 24.7, 24.5, 24.2, 22.7, 21.8, 20.9

050 관계대수 Ⓑ

- 관계대수는 관계형 데이터베이스에서 **원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어**이다.
- 관계대수는 릴레이션을 처리하기 위해 연산자와 연산규칙을 제공하며, 피연산자와 연산 결과가 모두 릴레이션이다.
- 관계대수는 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.

440081

25.11, 25.7, 23.10, 22.10, 22.7, 20.10, 필기 25.5, 24.5, 23.7, 23.2, 21.8, 21.5, …

051 순수 관계 연산자 Ⓐ

23.10, 필기 25.5, 24.5, 23.7, 23.2, 21.3, 20.8

Select

- 릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만드는 연산이다.
- 릴레이션의 행에 해당하는 튜플(Tuple)을 구하는 것이므로 수평 연산이라고도 한다.
- 기호 : σ(시그마)

25.7, 23.10, 22.10, 22.7, 필기 24.5, 23.7, 23.2, 21.8, 21.5, 20.8

Project

- 주어진 릴레이션에서 속성 리스트(Attribute List)에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 연산이다.
- 릴레이션의 열에 해당하는 속성을 추출하는 것이므로 수직 연산자라고도 한다.
- 기호 : π(파이)

23.10, 22.10, 필기 21.8, 21.5, 20.8

Join

- 공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산이다.
- Join의 결과는 Cartesian Product(교차곱)를 수행한 다음 Select를 수행한 것과 같다.
- 기호 : ⋈

25.11, 23.10, 20.10, 필기 24.5, 21.8, 21.5, 20.8

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.
- 기호 : ÷

440082

22.10, 필기 24.5, 24.2, 23.7, 23.5, 21.8, 21.5

052 일반 집합 연산자 A

22.10

합집합(UNION)

- 두 릴레이션에 존재하는 튜플의 합집합을 구하되, 결과로 생성된 릴레이션에서 중복되는 튜플은 제거되는 연산이다.
- 합집합의 카디널리티는 두 릴레이션 카디널리티의 합보다 크지 않다.
- 기호 : ∪

교집합(INTERSECTION)

- 두 릴레이션에 존재하는 튜플의 교집합을 구하는 연산이다.
- 교집합의 카디널리티는 두 릴레이션 중 카디널리티가 적은 릴레이션의 카디널리티보다 크지 않다.
- 기호 : ∩

22.10

차집합(DIFFERENCE)

- 두 릴레이션에 존재하는 튜플의 차집합을 구하는 연산이다.
- 차집합의 카디널리티는 릴레이션 R의 카디널리티보다 크지 않다.
- 기호 : —

22.10, 필기 24.5, 24.2, 21.8, 21.5

교차곱(CARTESIAN PRODUCT)

- 두 릴레이션에 있는 튜플들의 순서쌍을 구하는 연산이다.
- 교차곱의 디그리(Degree)는 두 릴레이션의 디그리를 더한 것과 같고, 카디널리티(Cardinality)는 두 릴레이션의 카디널리티를 곱한 것과 같다.
- 기호 : ×

440083

22.7, 필기 23.7, 22.7

053 관계해석 A

- 관계해석(Relational Calculus)은 **관계 데이터의 연산을 표현하는 방법**이다.
- 관계 데이터 모델의 제안자인 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.
- 원하는 정보를 정의할 때는 계산 수식을 사용한다.

340104

22.5, 20.11, 필기 25.2, 24.5, 24.2, 23.2, 21.8, 21.5, 21.3, 20.8

054 이상 A

- 이상(Anomaly)이란 데이터베이스 내에 **데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 않게 발생하는 곤란한 현상**을 의미한다.
- 삽입 이상(Insertion Anomaly) : 테이블에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들로 인해 삽입할 수 없게 되는 현상
- 삭제 이상(Deletion Anomaly) : 테이블에서 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는, 즉 연쇄 삭제가 발생하는 현상
- 갱신 이상(Update Anomaly) : 테이블에서 튜플에 있는 속성 값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 불일치성(Inconsistency)이 생기는 현상

440085

22.7, 필기 24.5, 24.2, 21.8

055 함수적 종속 Ⓐ

필기 24.5, 24.2, 21.8

함수적 종속(Functional Dependency)

어떤 테이블 R에서 X와 Y를 각각 R의 속성 집합의 부분 집합이라 하자. 속성 X의 값 각각에 대해 시간에 관계없이 항상 속성 Y의 값이 오직 하나만 연관되어 있을 때 Y는 X에 함수적 종속 또는 X가 Y를 함수적으로 결정한다고 하고, X → Y로 표기한다.

22.7

완전 함수적 종속(Full Functional Dependency)

어떤 테이블 R에서 속성 Y가 다른 속성 집합 X 전체에 대해 함수적 종속이면서 속성 집합 X의 어떠한 진부분 집합 Z(즉, Z ⊂ X)에도 함수적 종속이 아닐 때 속성 Y는 속성 집합 X에 완전 함수적 종속이라고 한다.

22.7

부분 함수적 종속(Partial Functional Dependency)

어떤 테이블 R에서 속성 Y가 다른 속성 집합 X 전체에 대해 함수적 종속이면서 속성 집합 X의 임의의 진부분 집합에 대해 함수적 종속일 때, 속성 Y는 속성 집합 X에 부분 함수적 종속이라고 한다.

22.7, 필기 24.2, 22.3, 20.8, 20.6

이행적 함수적 종속(Transitive Functional Dependency)

X → Y이고 Y → Z일 때 X → Z를 만족하는 관계를 의미한다.

340106

필기 21.8, 20.9

056 정규화 Ⓒ

- 정규화(Normalization)는 **테이블의 속성들이 상호 종속적인 관계를 갖는 특성을 이용하여 테이블을 무손실 분해하는 과정**이다.
- 정규화의 목적은 가능한 한 중복을 제거하여 삽입, 삭제, 갱신 이상의 발생 가능성을 줄이는 것이다.
- 정규형에는 제 1정규형(1NF), 제 2정규형(2NF), 제 3정규형(3NF), BCNF, 제 4정규형(4NF), 제 5정규형(5NF)이 있으며, 순서대로 정규화의 정도가 높아진다.

24.4, 21.7, 필기 25.5, 25.2, 24.7, 24.5, 24.2, 22.7, 22.4, 22.3, 21.8, 21.5, 21.3, 20.9, 20.8, 20.6

057 정규화 과정 Ⓐ

아래의 〈주문목록〉 테이블을 가지고 정규화 과정을 살펴보자. 〈주문목록〉 테이블의 기본키(Primary Key)는 제품번호이다.

〈주문목록〉

제품번호	제품명	재고수량	주문번호	고객번호	주소	주문수량
1001	모니터	2000	A345 D347	100 200	서울 부산	150 300
1007	마우스	9000	A210 A345 B230	300 100 200	광주 서울 부산	600 400 700
1201	키보드	2100	D347	200	부산	300

- 제 1정규형(1NF; First Normal Form)
  - 제 1정규형은 테이블 R에 속한 모든 속성의 도메인(Domain)이 원자 값(Atomic Value)만으로 되어 있는 정규형이다. 즉 테이블의 모든 속성 값이 원자 값으로만 되어 있는 정규형이다.
  - 〈주문목록〉 테이블에서는 하나의 제품에 대해 여러 개의 주문 관련 정보(주문번호, 고객번호, 주소, 주문수량)가 발생하고 있다. 따라서 〈주문목록〉테이블은 제 1정규형이 아니다.

예제 1 〈주문목록〉 테이블에서 반복되는 주문 관련 정보를 분리하여 제 1정규형으로 만드시오.

〈주문목록〉

제품번호	제품명	재고수량	주문번호	고객번호	주소	주문수량
1001	모니터	2000	A345 D347	100 200	서울 부산	150 300
1007	마우스	9000	A210 A345 B230	300 100 200	광주 서울 부산	600 400 700
1201	키보드	2100	D347	200	부산	300

⬇

〈제품〉

제품번호	제품명	재고수량
1001	모니터	2000
1007	마우스	9000
1201	키보드	2100

〈제품주문〉

주문번호	제품번호	고객번호	주소	주문수량
A345	1001	100	서울	150
D347	1001	200	부산	300
A210	1007	300	광주	600
A345	1007	100	서울	400
B230	1007	200	부산	700
D347	1201	200	부산	300

해설 〈주문목록〉 테이블에서 반복되는 주문 관련 정보인 주문번호, 고객번호, 주소, 주문수량을 분리하면 위와 같이 제 1정규형인 〈제품〉 테이블과 〈제품주문〉 테이블이 만들어진다.

– 1차 정규화 과정으로 생성된 〈제품주문〉 테이블의 기본키는 (주문번호, 제품번호)이고, 다음과 같은 함수적 종속이 존재한다.

> 주문번호, 제품번호 → 고객번호, 주소, 주문수량
> 주문번호 → 고객번호, 주소
> 고객번호 → 주소

• 제 2정규형(2NF; Second Normal Form)

– 제 2정규형은 테이블 R이 제 1정규형이고, 기본키가 아닌 모든 속성이 기본키에 대하여 완전 함수적 종속을 만족하는 정규형이다.

– 〈주문목록〉 테이블이 〈제품〉 테이블과 〈제품주문〉 테이블로 무손실 분해되면서 모두 제 1정규형이 되었지만 그 중 〈제품주문〉 테이블에는 기본키인 (주문번호, 제품번호)에 완전 함수적 종속이 되지 않는 속성이 존재한다. 즉 주문수량은 기본키에 대해 완전 함수적 종속이지만 고객번호와 주소는 주문번호에 의해서도 결정될 수 있으므로, 기본키에 대해 완전 함수적 종속이 아니다. 따라서 〈제품주문〉 테이블은 제 2정규형이 아니다.

예제 2 〈제품주문〉 테이블에서 주문번호에 함수적 종속이 되는 속성들을 분리하여 제 2정규형을 만드시오.

〈제품주문〉

주문번호	제품번호	고객번호	주소	주문수량
A345	1001	100	서울	150
D347	1001	200	부산	300
A210	1007	300	광주	600
A345	1007	100	서울	400
B230	1007	200	부산	700
D347	1201	200	부산	300

➡

〈주문목록〉

주문번호	제품번호	주문수량
A345	1001	150
D347	1001	300
A210	1007	600
A345	1007	400
B230	1007	700
D347	1201	300

〈주문〉

주문번호	고객번호	주소
A345	100	서울
D347	200	부산
A210	300	광주
B230	200	부산

해설 〈제품주문〉 테이블에서 주문번호에 함수적 종속이 되는 속성인 고객번호와 주소를 분리(즉 부분 함수적 종속을 제거)해 내면 위와 같이 제 2정규형인 〈주문목록〉 테이블과 〈주문〉 테이블로 무손실 분해된다.

– 제 2정규화 과정을 거쳐 생성된 〈주문〉 테이블의 기본키는 주문번호이다. 그리고 〈주문〉 테이블에는 아직도 다음과 같은 함수적 종속들이 존재한다.

> 주문번호 → 고객번호, 주소
> 고객번호 → 주소

• 제 3정규형(3NF; Third Normal Form)

– 제 3정규형은 테이블 R이 제 2정규형이고 기본키가 아닌 모든 속성이 기본키에 대해 이행적 함수적 종속(Transitive Functional Dependency)을 만족하지 않는 정규형이다.

– 〈제품주문〉 테이블이 〈주문목록〉 테이블과 〈주문〉 테이블로 무손실 분해되면서 모두 제 2정규형이 되었다. 그러나 〈주문〉 테이블에서 고객번호가 주문번호에 함수적 종속이고, 주소가 고객번호에 함수적 종속이므로 주소는 기본키인 주문번호에 대해 이행적 함수적 종속을 만족한다. 즉 주문번호 → 고객번호이고, 고객번호 → 주소이므로 주문번호 → 주소는 이행적 함수적 종속이 된다. 따라서 〈주문〉 테이블은 제 3정규형이 아니다.

예제 3 〈주문〉 테이블에서 이행적 함수적 종속을 제거하여 제 3정규형을 만드시오.

〈주문〉

주문번호	고객번호	주소
A345	100	서울
D347	200	부산
A210	300	광주
B230	200	부산

➡

〈주문〉

주문번호	고객번호
A345	100
D347	200
A210	300
B230	200

〈고객〉

고객번호	주소
100	서울
200	부산
300	광주

해설 〈주문〉 테이블에서 이행적 함수적 종속(즉 주문번호 → 주소)을 제거하여 무손실 분해함으로써 위와 같이 제 3정규형인 〈주문〉 테이블과 〈고객〉 테이블이 생성된다.

- BCNF(Boyce-Codd Normal Form)
  - BCNF는 테이블 R에서 모든 결정자가 후보키(Candidate Key)인 정규형이다.
  - 일반적으로 제 3정규형에 후보키가 여러 개 존재하고, 이러한 후보키들이 서로 중첩되어 나타나는 경우에 적용 가능하다.
  - 아래의 〈수강_교수〉 테이블(제 3정규형)은 함수적 종속{(학번, 과목명) → 담당교수, (학번, 담당교수) → 과목명, 담당교수 → 과목명}을 만족하고 있다. 〈수강_교수〉 테이블의 후보키는 (학번, 과목명)과 (학번, 담당교수)이다.

〈수강_교수〉

학번	과목명	담당교수
211746	데이터베이스	홍길동
211747	네트워크	유관순
211748	인공지능	윤봉길
211749	데이터베이스	홍길동
211747	데이터베이스	이순신
211749	네트워크	유관순

  - 〈수강_교수〉 테이블에서 결정자 중 후보키가 아닌 속성이 존재한다. 즉 함수적 종속 담당교수 → 과목명이 존재하는데, 담당교수가 〈수강_교수〉 테이블에서 후보키가 아니기 때문에 〈수강_교수〉 테이블은 BCNF가 아니다.

예제 4 〈수강_교수〉 테이블에서 결정자가 후보키가 아닌 속성을 분리하여 BCNF를 만드시오.

〈수강〉

학번	담당교수
211746	홍길동
211747	유관순
211748	윤봉길
211749	홍길동
211747	이순신
211749	유관순

〈교수〉

담당교수	과목명
홍길동	데이터베이스
이순신	데이터베이스
윤봉길	인공지능
유관순	네트워크

해설 〈수강_교수〉 테이블에서 BCNF를 만족하지 못하게 하는 속성(즉 담당교수 → 과목명)을 분리해내면 위와 같이 BCNF인 〈수강〉 테이블과 〈교수〉 테이블로 무손실 분해된다.

- 제 4정규형(4NF; Fourth Normal Form)

  제 4정규형은 테이블 R에 다중 값 종속(MVD; Multi Valued Dependency) A →→ B가 존재할 경우 R의 모든 속성이 A에 함수적 종속 관계를 만족하는 정규형이다.

  ※ **다중 값 종속(다치 종속)** : A, B, C 3개의 속성을 가진 테이블 R에서 어떤 복합 속성(A, C)에 대응하는 B 값의 집합이 A 값에만 종속되고 C 값에는 무관하면, B는 A에 다중 값 종속이라 하고, A →→ B로 표기함

- 제 5정규형(5NF; Fifth Normal Form)

  제 5정규형은 테이블 R의 모든 조인 종속(JD; Join Dependency)이 R의 후보키를 통해서만 성립되는 정규형이다.

  ※ **조인 종속** : 어떤 테이블 R의 속성에 대한 부분 집합 X, Y, …, Z가 있다고 하자. 이때 만일 테이블 R이 자신의 프로젝션(Projection) X, Y, …, Z를 모두 조인한 결과와 동일한 경우 테이블 R은 조인 종속 JD(X, Y, …, Z)를 만족한다고 함

340108

필기 23.2, 21.5, 21.3, 20.9, 20.8, 20.6

058 정규화 과정 정리 B

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 두부 가게에 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

340109

24.7, 21.4, 20.5, 필기 25.5, 24.2, 23.7, 20.9

059 반정규화 A

- 반정규화(Denormalization)는 시스템의 성능을 향상하고 개발 및 운영의 편의성 등을 높이기 위해 **정규화된 데이터 모델을 의도적으로 통합, 중복, 분리하여 정규화 원칙을 위배하는 행위**이다.
- 반정규화를 수행하면 시스템의 성능이 향상되고 관리 효율성은 증가하지만 데이터의 일관성 및 정합성이 저하될 수 있다.
- 과도한 반정규화는 오히려 성능을 저하시킬 수 있다.

340111

필기 25.8, 25.5, 24.7, 24.5, 24.2, 23.7, 22.7, 21.5, 21.3

060 시스템 카탈로그 B

- 시스템 카탈로그(System Catalog)는 시스템 그 자체에 관련이 있는 **다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스**이다.
- 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.

340113 21.7, 20.5, 필기 25.8, 25.5, 25.2, 24.7, 24.5, 24.2, 23.7, 23.5, 23.2, 22.7, …

061 트랜잭션의 특성 Ⓐ

특성	의미
21.7, 20.5, 필기 25.2, 24.5, … Atomicity(원자성)	트랜잭션의 연산은 데이터베이스에 모두 반영되도록 완료(Commit)되든지 아니면 전혀 반영되지 않도록 복구(Rollback)되어야 함
20.5, 필기 25.8, 25.5, … Consistency (일관성)	트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환함
20.5, 필기 25.5, 25.2, 23.7, … Isolation (독립성, 격리성, 순차성)	둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없음
20.5, 필기 25.5, 25.2, 24.7, … Durability (영속성, 지속성)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 함

340114 필기 22.7, 20.9

062 CRUD 분석 Ⓒ

- CRUD 분석은 **프로세스와 테이블 간에 CRUD 매트릭스를 만들어서 트랜잭션을 분석하는 것**이다.
- CRUD 분석을 통해 많은 트랜잭션이 몰리는 테이블을 파악할 수 있으므로 디스크 구성 시 유용한 자료로 활용할 수 있다.
- CRUD 매트릭스의 각 셀에는 Create(생성), Read(읽기), Update(갱신), Delete(삭제)의 앞 글자가 들어간다.

340115 필기 24.7, 21.8, 21.3

063 인덱스 Ⓒ

- 인덱스(Index, 색인)는 **데이터 레코드를 빠르게 접근하기 위해 〈키 값, 포인터〉 쌍으로 구성되는 데이터 구조**이다.
- 인덱스는 레코드가 저장된 물리적 구조에 접근하는 방법을 제공한다.
- 인덱스를 통해서 파일의 레코드에 빠르게 액세스 할 수 있다.

340117 필기 25.8, 25.5, 25.2, 24.7, 24.5, 24.2, 23.7, 22.4, 20.9, 20.6

064 뷰 Ⓑ

- 뷰(View)는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 **하나 이상의 기본 테이블로부터 유도된**, 이름을 가지는 **가상 테이블**이다.
- 뷰가 정의된 기본 테이블이나 뷰를 삭제하면 그 테이블이나 뷰를 기초로 정의된 다른 뷰도 자동으로 삭제된다.
- 뷰를 정의할 때는 CREATE문, 제거할 때는 DROP문을 사용한다.

340119 필기 25.8, 25.2, 24.7, 24.5, 24.2, 23.2, 22.7, 21.5, 20.8

065 파티션의 종류 Ⓑ

필기 25.8, 24.7, 24.5, … **범위 분할 (Range Partitioning)**	지정한 열의 값을 기준으로 분할함 예 일별, 월별, 분기별 등
필기 25.8, 25.2, 24.7, … **해시 분할 (Hash Partitioning)**	• 해시 함수를 적용한 결과 값에 따라 데이터를 분할함 • 특정 파티션에 데이터가 집중되는 범위 분할의 단점을 보완한 것으로, 데이터를 고르게 분산할 때 유용함 • 특정 데이터가 어디에 있는지 판단할 수 없음 • 고객번호, 주민번호 등과 같이 데이터가 고른 컬럼에 효과적임
필기 25.8, 24.7, 20.8 **조합 분할 (Composite Partitioning)**	• 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할하는 방식 • 범위 분할한 파티션이 너무 커서 관리가 어려울 때 유용함

340120 필기 25.8, 25.5, 25.2, 24.7, 23.5, 20.8, 20.6

066 분산 데이터베이스의 목표 B

- 위치 투명성(Location Transparency) : 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있음
- 중복 투명성(Replication Transparency) : 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행함
- 병행 투명성(Concurrency Transparency) : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않음
- 장애 투명성(Failure Transparency) : 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

340121 20.7

067 RTO/RPO A

20.7 **RTO(Recovery Time Objective, 목표 복구 시간)**	비상사태 또는 업무 중단 시점으로부터 복구되어 가동될 때까지의 소요 시간을 의미함 예 장애 발생 후 6시간 내 복구 가능
RPO(Recovery Point Objective, 목표 복구 시점)	비상사태 또는 업무 중단 시점으로부터 데이터를 복구할 수 있는 기준점을 의미함 예 장애 발생 전인 지난 주 금요일에 백업시켜 둔 복원 시점으로 복구 가능

340123 25.11, 23.10, 21.4, 필기 23.5, 21.3, 20.9

068 임의 접근통제 A

- 임의 접근통제(DAC; Discretionary Access Control)는 **데이터에 접근하는 사용자의 신원에 따라 접근 권한을 부여하는 방식**이다.
- 데이터 소유자가 접근통제 권한을 지정하고 제어한다.
- 객체를 생성한 사용자가 생성된 객체에 대한 모든 권한을 부여받고, 부여된 권한을 다른 사용자에게 허가할 수도 있다.

340124 25.11, 23.10, 필기 23.5, 21.8, 21.3, 20.9

069 강제 접근통제 A

- 강제 접근통제(MAC; Mandatory Access Control)는 **주체와 객체의 등급을 비교하여 접근 권한을 부여하는 방식**이다.
- 시스템이 접근통제 권한을 지정한다.
- 데이터베이스 객체별로 보안 등급을 부여할 수 있다.
- 사용자별로 인가 등급을 부여할 수 있다.

440103 25.11, 23.10, 필기 24.5, 24.2, 23.5, 22.4, 21.3

070 역할기반 접근통제 A

- 역할기반 접근통제(RBAC; Role Based Access Control)는 **사용자의 역할에 따라 접근 권한을 부여하는 방식**이다.
- 중앙관리자가 접근통제 권한을 지정한다.
- 임의 접근통제와 강제 접근통제의 단점을 보완하였다.
- 다중 프로그래밍 환경에 최적화된 방식이다.

340127 필기 25.8, 24.5, 23.7, 23.5, 23.2, 22.3, 20.9

071 DAS B

- DAS(Direct Attached Storage)는 **서버와 저장장치를 전용 케이블로 직접 연결하는 방식**이다.
- 일반 가정에서 컴퓨터에 외장하드를 연결하는 것이 여기에 해당된다.
- 직접 연결 방식이므로 다른 서버에서 접근할 수 없고 파일을 공유할 수 없다.

340129 필기 25.2, 22.7, 21.5

072 SAN C

- SAN(Storage Area Network)은 DAS의 빠른 처리와 NAS의 파일 공유 장점을 혼합한 방식으로, **서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성하는 방식**이다.
- 파이버 채널(Fibre Channel, 광 채널) 스위치를 이용하여 네트워크를 구성한다.
- 파이버 채널 스위치는 서버와 저장장치를 광케이블로 연결하므로 처리 속도가 빠르다.
- 서버들이 저장장치 및 파일을 공유할 수 있다.

340130 필기 25.5, 24.7, 24.2, 23.7, 22.3, 21.8

073 자료 구조의 분류 B

- **선형 구조** : 배열, 선형 리스트(연속 리스트, 연결 리스트), 스택, 큐, 데크
- **비선형 구조** : 트리, 그래프

340132 필기 25.8, 25.5, 25.2, 23.7, 23.2, 22.4, 22.3, 21.8, 21.3

074 스택 B

- 스택(Stack)은 **리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료구조**이다.
- 후입선출(LIFO; Last In First Out) 방식으로 자료를 처리한다.
- 저장할 기억 공간이 없는 상태에서 데이터가 삽입되면 오버플로(Overflow)가 발생한다.
- 삭제할 데이터가 없는 상태에서 데이터를 삭제하면 언더플로(Underflow)가 발생한다.

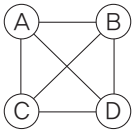
340135 필기 23.2, 20.9

075 방향/무방향 그래프의 최대 간선 수 C

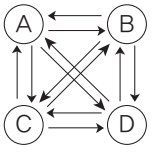
- 방향 그래프의 최대 간선 수 : $n(n-1)$
- 무방향 그래프에서 최대 간선 수 : $n(n-1)/2$

※ n은 정점의 개수이다.

예제 정점이 4개인 경우 무방향 그래프와 방향 그래프의 최대 간선 수를 구하시오.



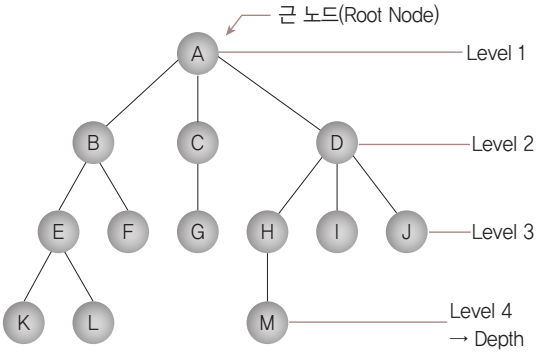
- 무방향 그래프의 최대 간선 수 : $4(4-1)/2 = 6$



- 방향 그래프의 최대 간선 수 : $4(4-1) = 12$

340137 필기 25.8, 24.7, 23.7, 23.2, 20.8, 20.6

076 트리 관련 용어 B



- **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 뻗어나온 가지의 수
  예 A=3, B=2, C=1, D=3

- 단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드, 즉 Degree가 0인 노드
  예 K, L, F, G, M, I, J
- Level : 근 노드의 Level을 1로 가정한 후 어떤 Level이 L이면 자식 노드는 L+1
  예 H의 레벨은 3
- 깊이(Depth, Height) : Tree에서 노드가 가질 수 있는 최대의 레벨
  예 위 트리의 깊이는 4
- 숲(Forest) : 여러 개의 트리가 모여 있는 것
  예 위 트리에서 근 노드 A를 제거하면 B, C, D를 근 노드로 하는 세 개의 트리가 있는 숲이 생긴다.
- 트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수
  예 노드 A나 D가 세 개의 디그리를 가지므로 위 트리의 디그리는 3이다.

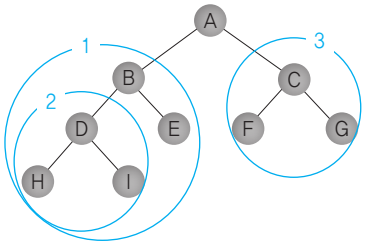
340138

필기 25.2, 24.5, 21.3, 20.8, 20.6

077 Preorder 운행법 Ⓑ

Preorder 운행법은 이진 트리를 **Root → Left → Right 순으로 운행하며 노드들을 찾아가는 방법**이다.

Preorder 운행법의 방문 순서



※ 서브트리를 하나의 노드로 생각할 수 있도록 그림과 같이 서브트리 단위로 묶는다. 다른 운행법 모두 공통으로 사용한다.

❶ Preorder는 Root → Left → Right이므로 A13이 된다.

❷ 1은 B2E이므로 A**B2E**3이 된다.

❸ 2는 DHI이므로 AB**DHIE**3이 된다.

❹ 3은 CFG이므로 ABDHIE**CFG**가 된다.

- 방문 순서 : ABDHIECFG

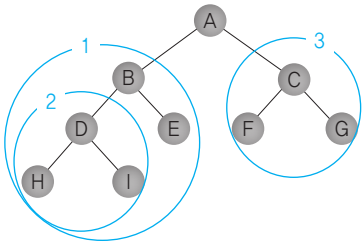
340139

필기 24.2, 23.5, 21.8, 20.9

078 Inorder 운행법 Ⓑ

Inorder 운행법은 이진 트리를 **Left → Root → Right 순으로 운행하며 노드들을 찾아가는 방법**이다.

Inorder 운행법의 방문 순서



❶ Inorder는 Left → Root → Right이므로 1A3이 된다.

❷ 1은 2BE이므로 **2BE**A3이 된다.

❸ 2는 HDI이므로 **HDIBE**A3이 된다.

❹ 3은 FCG이므로 HDIBEA**FCG**가 된다.

- 방문 순서 : HDIBEAFCG

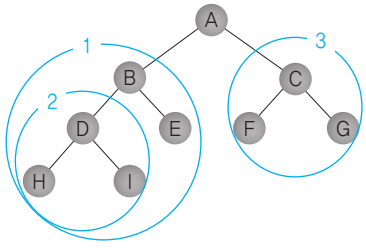
440115

필기 25.8, 25.5, 24.7, 22.7, 22.4

079 Postorder 운행법 Ⓒ

Postorder 운행법은 **이진 트리를 Left → Right → Root 순으로 운행하며 노드들을 찾아가는 방법**이다.

Postorder 운행법의 방문 순서



❶ Postorder는 Left → Right → Root이므로 13A가 된다.

❷ 1은 2EB이므로 **2EB**3A가 된다.

❸ 2는 HID이므로 **HIDEB**3A가 된다.

❹ 3은 FGC이므로 HIDEB**FGC**A가 된다.

- 방문 순서 : HIDEBFGCA

340143

필기 24.7, 24.5, 21.5, 21.3, 20.9

080 Postfix로 표기된 수식을 Infix로 바꾸기 Ⓑ

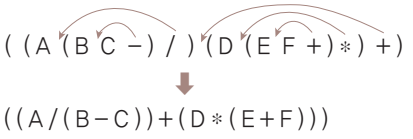
Postfix는 Infix 표기법에서 연산자를 해당 피연산자 두 개의 뒤로 이동한 것이므로 연산자를 다시 해당 피연산자 두 개의 가운데로 옮기면 된다.

ABC-/DEF+*+ ➡ A/(B-C)+D*(E+F)

❶ 먼저 인접한 피연산자 두 개와 오른쪽의 연산자를 괄호로 묶는다.

((A(BC-)/)(D(EF+)*)+)

❷ 연산자를 해당 피연산자의 가운데로 이동시킨다.



((A/(B-C))+(D*(E+F)))

❸ 필요 없는 괄호를 제거한다.

A/(B-C)+D*(E+F)

340144

필기 25.8, 24.2, 20.9

081 삽입 정렬 Ⓒ

- 삽입 정렬(Insertion Sort)은 가장 간단한 정렬 방식으로, **이미 순서화된 파일에 새로운 하나의 레코드를 순서에 맞게 삽입시켜 정렬하는 방식**이다.
- 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.

340145

필기 25.5, 24.7, 24.5, 22.7, 21.3, 20.8

082 선택 정렬 Ⓑ

- 선택 정렬(Selection Sort)은 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 방식**이다.
- 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 :** 8 5 6 2 4 → 8 5 6 2 4 → 2 5 6 8 4

  첫 번째부터 마지막 값 중 최소값 2를 찾아 첫 번째 값 8과 위치를 교환한다.
- **2회전 :** 2 5 6 8 4 → 2 5 6 8 4 → 2 4 6 8 5

  두 번째부터 마지막 값 중 최소값 4를 찾아 두 번째 값 5와 위치를 교환한다.
- **3회전 :** 2 4 6 8 5 → 2 4 6 8 5 → 2 4 5 8 6

  세 번째부터 마지막 값 중 최소값 5를 찾아 세 번째 값 6과 위치를 교환한다.
- **4회전 :** 2 4 5 8 6 → 2 4 5 8 6 → 2 4 5 6 8

  네 번째부터 마지막 값 중 최소값 6을 찾아 네 번째 값 8과 위치를 교환한다.

340146

필기 25.2, 23.2, 22.4, 21.8, 21.5

083 버블 정렬 Ⓑ

- 버블 정렬(Bubble Sort)은 주어진 파일에서 **인접한 두 개의 레코드 키 값을 비교하여 그 크기에 따라 레코드 위치를 서로 교환하는 정렬 방식**이다.
- 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

340147

필기 25.5, 24.2, 23.5, 23.2, 22.3, 21.3

084 퀵 정렬 Ⓑ

- 퀵 정렬(Quick Sort)은 **키를 기준으로 작은 값은 왼쪽, 큰 값은 오른쪽 서브 파일에 분해시키는** 과정을 반복하는 **정렬 방식**이다.
- 레코드의 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어 가면서 정렬한다.
- 평균 수행 시간 복잡도는 $O(n\log_2 n)$이고, 최악의 수행 시간 복잡도는 $O(n^2)$이다.

3장 통합 구현

340153 필기 21.5

085 연계 서버 / 송·수신 시스템 Ⓒ

- **연계 서버** : 데이터를 전송 형식에 맞게 변환하고 송 · 수신을 수행하는 등 송 · 수신과 관련된 모든 처리 수행
- **송신 시스템** : 인터페이스 테이블 또는 파일의 데이터를 전송 형식에 맞도록 변환 및 송신을 수행하는 시스템
- **수신 시스템** : 수신 데이터를 인터페이스 테이블이나 파일로 생성하는 시스템

340156 20.5

086 XML Ⓐ

- XML(eXtensible Markup Language)은 **특수한 목적을 갖는 마크업 언어를 만드는 데 사용되는 다목적 마크업 언어**이다.
- 웹브라우저 간 HTML 문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위하여 개발되었다.
- 사용자가 직접 문서의 태그(Tag)를 정의할 수 있으며, 다른 사용자가 정의한 태그를 사용할 수 있다.
- 트리 구조로 구성되어 있어 상위 태그는 여러 개의 하위 태그를 가질 수 있다.

340157 20.7

087 SOAP Ⓐ

- SOAP(Simple Object Access Protocol)는 컴퓨터 **네트워크 상에서 HTTP/HTTPS, SMTP 등을 이용하여 XML을 교환하기 위한 통신 규약**이다.
- 웹 서비스에서 사용되는 메시지의 형식과 처리 방법을 지정한다.
- 기본적으로 HTTP 기반에서 동작하기 때문에 프록시와 방화벽의 영향 없이 통신할 수 있다.
- 최근에는 무거운 구조의 SOAP 대신 RESTful 프로토콜을 이용하기도 한다.

340158 21.4

088 WSDL Ⓐ

- WSDL(Web Services Description Language)은 **웹 서비스와 관련된 서식이나 프로토콜 등을 표준적인 방법으로 기술하고 게시하기 위한 언어**이다.
- XML로 작성되며, UDDI의 기초가 된다.
- SOAP, XML 스키마와 결합하여 인터넷에서 웹 서비스를 제공하기 위해 사용된다.
- 클라이언트는 WSDL 파일을 읽어 서버에서 어떠한 조작이 가능한지를 파악할 수 있다.

4장 서버 프로그램 구현

340165 20.5

089 모듈화 A

- 모듈화(Modularity)는 소프트웨어의 성능 향상, 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 **시스템의 기능들을 모듈 단위로 나누는 것**을 의미한다.
- 모듈화는 모듈 간 결합도(Coupling)의 최소화와 모듈 내 요소들의 응집도(Cohesion)를 최대화하는 것이 목표이다.

340166 필기 24.7, 21.8

090 추상화 C

- 추상화(Abstraction)는 문제의 **전체적이고 포괄적인 개념을 설계**한 후 차례로 세분화하여 **구체화시켜 나가는 것**이다.
- 추상화의 유형

과정 추상화	자세한 수행 과정을 정의하지 않고, 전반적인 흐름만 파악할 수 있게 설계하는 방법
자료 추상화	데이터의 세부적인 속성이나 용도를 정의하지 않고, 데이터 구조를 대표할 수 있는 표현으로 대체하는 방법
제어 추상화	이벤트 발생의 정확한 절차나 방법을 정의하지 않고, 대표할 수 있는 표현으로 대체하는 방법

340168 필기 25.5, 24.5, 21.8, 21.5

091 정보 은닉 C

- 정보 은닉(Information Hiding)은 **한 모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법**이다.
- 정보 은닉을 통해 모듈을 독립적으로 수행할 수 있다.
- 하나의 모듈이 변경되더라도 다른 모듈에 영향을 주지 않으므로 수정, 시험, 유지보수가 용이하다.

340171 필기 23.2, 20.8

092 협약(Contract)에 의한 설계 C

- 협약에 의한 설계는 컴포넌트를 설계할 때 클래스에 대한 여러 가정을 공유할 수 있도록 명세한 것이다.
- 컴포넌트에 대한 정확한 인터페이스를 명세한다.
- 명세에 포함될 조건

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

340175 필기 25.8, 24.7, 23.7, 23.2, 22.7, 21.8, 21.5, 20.9

093 파이프-필터 패턴 B

- 파이프-필터 패턴(Pipe-Filter Pattern)은 **데이터 스트림 절차의 각 단계를 필터로 캡슐화하여 파이프를 통해 전송**하는 패턴이다.
- 앞 시스템의 처리 결과물을 파이프를 통해 전달받아 처리한 후 그 결과물을 다시 파이프를 통해 다음 시스템으로 넘겨주는 패턴을 반복한다.
- 데이터 변환, 버퍼링, 동기화 등에 주로 사용된다.
- 대표적으로 UNIX의 쉘(Shell)이 있다.

340177 필기 24.5, 23.5, 23.2, 21.8

094 기타 패턴 B

필기 24.5, 23.5, 23.2, 21.8

마스터-슬레이브 패턴(Master-Slave Pattern)

슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받는 방식으로 작업을 수행하는 패턴이다.

예 장애 허용 시스템, 병렬 컴퓨팅 시스템

브로커 패턴(Broker Pattern)

사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커 컴포넌트가 요청에 맞는 컴포넌트와 사용자를 연결해주는 패턴이다.

예 분산 환경 시스템

피어-투-피어 패턴(Peer-To-Peer Pattern)

피어(Peer)라 불리는 하나의 컴포넌트가 클라이언트가 될수도, 서버가 될 수도 있는 패턴이다.

예 파일 공유 네트워크

이벤트-버스 패턴(Event-Bus Pattern)

소스가 특정 채널에 이벤트 메시지를 발행(Publish)하면, 해당 채널을 구독(Subscribe)한 리스너(Listener)들이 메시지를 받아 이벤트를 처리하는 패턴이다.

예 알림 서비스

블랙보드 패턴(Blackboard Pattern)

모든 컴포넌트들이 공유 데이터 저장소와 블랙보드 컴포넌트에 접근이 가능한 패턴이다.

예 음성 인식, 차량 식별, 신호 해석

필기 21.8

인터프리터 패턴(Interpreter Pattern)

프로그램 코드의 각 라인을 수행하는 방법을 지정하고, 기호마다 클래스를 갖도록 구성된 패턴이다.

예 번역기, 컴파일러, 인터프리터

340180

필기 25.8, 25.5, 25.2, 24.2, 23.5, 22.3, 21.8, 21.5, 20.8, 20.6

095 클래스 B

- 클래스(Class)는 **공통된 속성과 연산을 갖는 객체의 집합**이다.
- 각각의 객체들이 갖는 속성과 연산을 정의하고 있는 틀이다.
- 클래스에 속한 각각의 객체를 인스턴스(Instance)라고 한다.

440149

필기 25.2, 23.2, 22.7, 21.5

096 메시지 B

- 메시지(Message)는 객체들 간의 상호작용에 사용되는 수단으로, **객체의 동작이나 연산을 일으키는 외부의 요구 사항**이다.
- 메시지를 받은 객체는 대응하는 연산을 수행하여 예상된 결과를 반환한다.

340182

필기 24.5, 24.2, 23.5, 21.8, 21.3, 20.9, 20.8

097 캡슐화 B

- 캡슐화(Encapsulation)는 **외부에서의 접근을 제한하기 위해 인터페이스를 제외한 세부 내용을 은닉**하는 것이다.
- 캡슐화된 객체는 외부 모듈의 변경으로 인한 파급 효과가 적다.
- 객체들 간에 메시지를 주고받을 때 상대 객체의 세부 내용은 알 필요가 없으므로 인터페이스가 단순해지고, 객체 간의 결합도가 낮아진다.

340186

필기 23.7, 21.8, 21.3

098 객체지향 분석 B

- 객체지향 분석(OOA; Object Oriented Analysis)은 사용자의 요구사항과 관련된 **객체, 속성, 연산, 관계 등을 정의하여 모델링**하는 작업이다.
- 개발을 위한 업무를 객체와 속성, 클래스와 멤버, 전체와 부분 등으로 나누어서 분석한다.
- 클래스를 식별하는 것이 객체지향 분석의 주요 목적이다.

340187 필기 21.3, 20.6

099 객체지향 분석의 방법론 C

- Rumbaugh(럼바우) 방법 : 분석 활동을 객체 모델, 동적 모델, 기능 모델로 나누어 수행함
- Booch(부치) 방법 : 미시적(Micro) 개발 프로세스와 거시적(Macro) 개발 프로세스를 모두 사용하며, 클래스와 객체들을 분석 및 식별하고 클래스의 속성과 연산을 정의함
- Jacobson 방법 : 유스케이스(Use Case)를 강조하여 사용함
- 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 구성함
- Wirfs-Brock 방법 : 분석과 설계 간의 구분이 없고, 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행함

340188 21.7, 필기 25.5, 25.2, 24.7, 24.2, 23.7, 22.7, 22.3, 21.8, 21.5, 20.9, 20.8, 20.6

100 럼바우의 분석 기법 A

- 럼바우(Rumbaugh)의 분석 기법은 **모든 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링**하는 기법이다.
- 객체 모델링 기법(OMT, Object-Modeling Technique)이라고도 한다.
- 분석 활동은 '객체 모델링 → 동적 모델링 → 기능 모델링' 순으로 이루어 진다.
  - 객체 모델링(Object Modeling) : 정보 모델링(Information Modeling)이라고도 하며, 시스템에서 요구되는 객체를 찾아내어 속성과 연산 식별 및 객체들 간의 관계를 규정하여 객체 다이어그램으로 표시하는 모델링
  - 동적 모델링(Dynamic Modeling) : 상태 다이어그램을 이용하여 시간의 흐름에 따른 객체들 간의 제어 흐름, 상호 작용, 동작 순서 등의 동적인 행위를 표현하는 모델링
  - 기능 모델링(Functional Modeling) : 자료 흐름도(DFD)를 이용하여 다수의 프로세스들 간의 자료 흐름을 중심으로 처리 과정을 표현한 모델링

440157 22.7, 필기 25.8, 24.5, 24.2, 23.5, 22.7, 22.3, 20.9, 20.8

101 객체지향 설계 원칙(SOLID 원칙) A

- 단일 책임 원칙(SRP) : 객체는 단 하나의 책임만 가져야 한다는 원칙
- 개방-폐쇄 원칙(OCP) : 기존의 코드를 변경하지 않고 기능을 추가할 수 있도록 설계해야 한다는 원칙
- 리스코프 치환 원칙(LSP) : 자식 클래스는 최소한 부모 클래스의 기능은 수행할 수 있어야 한다는 원칙
- 인터페이스 분리 원칙(ISP) : 자신이 사용하지 않는 인터페이스와 의존 관계를 맺거나 영향을 받지 않아야 한다는 원칙
- 의존 역전 원칙(DIP) : 의존 관계 성립 시 추상성이 높은 클래스와 의존 관계를 맺어야 한다는 원칙

340190 필기 23.7, 23.5, 23.2, 21.8

102 모듈 B

- 모듈(Module)은 **모듈화를 통해 분리된 시스템의 각 기능**으로, 서브루틴, 서브시스템, 소프트웨어 내의 프로그램, 작업 단위 등을 의미한다.
- 모듈의 기능적 독립성은 소프트웨어를 구성하는 각 모듈의 기능이 서로 독립됨을 의미한다.
- 하나 또는 몇 개의 논리적인 기능을 수행하기 위한 명령어들의 집합이라고도 할 수 있다.
- 모듈의 독립성은 결합도(Coupling)와 응집도(Cohesion)에 의해 측정된다.

340191

필기 25.2, 24.5, 23.2, 21.8, 21.5, 21.3, 20.8, 20.6

103 결합도 Ⓑ

- 결합도(Coupling)는 **모듈 간에 상호 의존하는 정도** 또는 두 모듈 사이의 연관 관계이다.
- 결합도가 약할수록 품질이 높고, 강할수록 품질이 낮다.
- 결합도의 종류와 강도

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

결합도 강함 ← → 결합도 약함

340192

25.4, 24.7, 21.10, 21.4, 필기 25.8, 23.7, 23.2, 22.7, 22.4, 20.9, 20.8

104 결합도의 종류 Ⓐ

25.4, 21.4, 필기 25.8, 23.7, 23.2, 22.7, 22.4, 20.9

내용 결합도(Content Coupling)

한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도이다.

25.4, 21.4, 필기 23.7, 23.2, 20.9

공통(공유) 결합도(Common Coupling)

- 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도이다.
- 파라미터가 아닌 모듈 밖에 선언된 전역 변수를 사용하여 전역 변수를 갱신하는 방식으로 상호작용하는 때의 결합도이다.

필기 25.8, 22.7

외부 결합도(External Coupling)

어떤 모듈에서 선언한 데이터(변수)를 외부의 다른 모듈에서 참조할 때의 결합도이다.

24.7, 21.10, 필기 20.8

제어 결합도(Control Coupling)

- 어떤 모듈이 다른 모듈 내부의 논리적인 흐름을 제어하기 위해 제어 신호나 제어 요소를 전달하는 결합도이다.
- 하위 모듈에서 상위 모듈로 제어 신호가 이동하여 하위 모듈이 상위 모듈에게 처리 명령을 내리는 권리 전도 현상이 발생하게 된다.

25.4, 21.4, 필기 25.8, 23.2, 22.7, 22.4, 20.9

스탬프(검인) 결합도(Stamp Coupling)

모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도이다.

필기 23.7, 23.2, 22.7, 20.9

자료 결합도(Data Coupling)

모듈 간의 인터페이스가 자료 요소로만 구성될 때의 결합도이다.

340193

24.4, 필기 25.8, 24.5, 23.5, 23.2, 22.4, 21.5, 21.3, 20.6

105 응집도 Ⓐ

- 응집도(Cohesion)는 **모듈의 내부 요소들이 서로 관련되어 있는 정도**이다.
- 응집도가 강할수록 품질이 높고, 약할수록 품질이 낮다.
- 응집도의 종류와 강도

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

응집도 강함 ← → 응집도 약함

340194

24.7, 21.7, 필기 21.8, 20.9, 20.8

106 응집도의 종류 Ⓐ

21.7

기능적 응집도(Functional Cohesion)

모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도이다.

24.7

순차적 응집도(Sequential Cohesion)

모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도이다.

21.7

교환(통신)적 응집도(Communication Cohesion)

동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도이다.

21.7, 필기 20.8

절차적 응집도(Procedural Cohesion)

모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도이다.

필기 21.8

시간적 응집도(Temporal Cohesion)

특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우의 응집도이다.

논리적 응집도(Logical Cohesion)

유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도이다.

필기 20.9

우연적 응집도(Coincidental Cohesion)

모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우의 응집도이다.

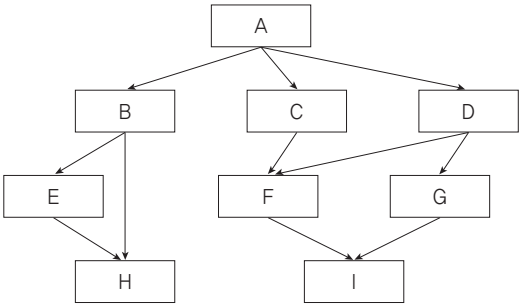
340195

22.7, 20.5, 필기 22.7, 21.3

107 팬인 / 팬아웃 Ⓐ

- 팬인(Fan-In)은 **어떤 모듈을 제어하는 모듈의 수**이다.
- 팬아웃(Fan-Out)은 **어떤 모듈에 의해 제어되는 모듈의 수**를 의미한다.

예제 다음의 시스템 구조도에서 각 모듈의 팬인(Fan-In)과 팬아웃(Fan-Out)을 구하시오.



해설

- 팬인(Fan-In) : A는 0, B · C · D · E · G는 1, F · H · I는 2
- 팬아웃(Fan-Out) : H · I는 0, C · E · F · G는 1, B · D는 2, A는 3

340196

필기 25.5, 24.2, 22.3, 20.9

108 N-S 차트 Ⓒ

- N-S 차트(Nassi-Schneiderman Chart)는 **논리의 기술에 중점을 두고 도형을 이용해 표현**하는 방법이다.
- GOTO나 화살표를 사용하지 않는다.
- 연속, 선택 및 다중 선택, 반복의 3가지 제어 논리 구조로 표현한다.
- 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는 데 적합하다.

340198

21.4

109 IPC Ⓐ

- IPC(Inter-Process Communication)는 **모듈 간 통신 방식을 구현하기 위해 사용되는 대표적인 프로그래밍 인터페이스 집합**이다.
- 복수의 프로세스를 수행하며 이뤄지는 프로세스 간 통신까지 구현이 가능하다.
- IPC의 대표 메소드 5가지
  - 공유 메모리(Shared Memory)
  - 소켓(Socket)
  - 세마포어(Semaphores)
  - 파이프와 네임드 파이프(Pipes & named Pipes)
  - 메시지 큐잉(Message Queueing)

340199

25.11, 21.10, 필기 21.3

110 테스트 케이스 Ⓐ

- 테스트 케이스(Test Case)는 구현된 **소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위한 테스트 항목에 대한 명세서**이다.
- ISO/IEC/IEEE 29119-3 표준에 따른 테스트 케이스의 구성 요소
  - 식별자 : 항목 식별자, 일련번호
  - 테스트 항목 : 테스트 대상(모듈 또는 기능)
  - 입력 명세 : 테스트 데이터 또는 테스트 조건
  - 출력 명세 : 테스트 케이스 수행 시 예상되는 출력 결과
  - 환경 설정 : 필요한 하드웨어나 소프트웨어의 환경
  - 특수 절차 요구 : 테스트 케이스 수행 시 특별히 요구되는 절차
  - 의존성 기술 : 테스트 케이스 간의 의존성

340201

필기 22.4, 21.3, 20.9

111 재사용 Ⓐ

- 재사용(Reuse)은 **이미 개발된 기능들을 새로운 시스템이나 기능 개발에 사용하기 적합하도록 최적화**하는 작업이다.
- 새로 개발하는데 필요한 비용과 시간을 절약할 수 있다.
- 누구나 이해할 수 있고 사용이 가능하도록 사용법을 공개해야 한다.
- 재사용 규모에 따른 분류

필기 20.9 **함수와 객체**	클래스나 메소드 단위의 소스 코드를 재사용함
필기 20.9 **컴포넌트**	컴포넌트 자체에 대한 수정 없이 인터페이스를 통해 통신하는 방식으로 재사용함
필기 20.9 **애플리케이션**	공통된 기능들을 제공하는 애플리케이션을 공유하는 방식으로 재사용함

340203

필기 23.7, 23.2, 20.9, 20.6

112 코드의 종류 Ⓑ

필기 23.7, 23.2, 20.6

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

필기 23.2, 20.9

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

340204

20.11, 필기 25.8, 24.7, 23.5, 22.3, 21.8, 20.9, 20.8

113 디자인 패턴 A

- 디자인 패턴(Design Pattern)은 **모듈 간의 관계 및 인터페이스를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제**를 의미한다.
- 문제 및 배경, 실제 적용된 사례, 재사용이 가능한 샘플 코드 등으로 구성되어 있다.
- GOF의 디자인 패턴은 생성 패턴, 구조 패턴, 행위 패턴으로 구분된다.

340205

24.4, 21.10, 필기 25.8, 25.2, 24.5, 24.2, 23.7, 23.5, 23.2, 22.7, 22.3, 21.8, …

114 생성 패턴 A

생성 패턴(Creational Pattern)은 클래스나 객체의 생성과 참조 과정을 정의하는 패턴이다.

24.4, 필기 24.5, 22.3, 21.3

추상 팩토리(Abstract Factory)

- 구체적인 클래스에 의존하지 않고, 인터페이스를 통해 서로 연관 · 의존하는 객체들의 그룹으로 생성하여 추상적으로 표현하는 패턴이다.
- 연관된 서브 클래스를 묶어 한 번에 교체하는 것이 가능하다.

필기 21.3

빌더(Builder)

- 작게 분리된 인스턴스를 건축 하듯이 조합하여 객체를 생성하는 패턴이다.
- 객체의 생성 과정과 표현 방법을 분리하고 있어, 동일한 객체 생성에서도 서로 다른 결과를 만들어 낼 수 있다.

21.10, 필기 25.5, 23.7, 23.2, 21.5, 20.8

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.

필기 23.2, 21.5, 20.8

프로토타입(Prototype)

- 원본 객체를 복제하는 방법으로 객체를 생성하는 패턴이다.
- 일반적인 방법으로 객체를 생성하며, 비용이 큰 경우 주로 이용한다.

필기 25.2, 24.2, 23.5, 22.7, 21.8, 21.5, 21.3

싱글톤(Singleton)

- 하나의 객체를 생성하면 생성된 객체를 어디서든 참조할 수 있지만, 여러 프로세스가 동시에 참조할 수는 없는 패턴이다.
- 클래스 내에서 인스턴스가 하나뿐임을 보장하며, 불필요한 메모리 낭비를 최소화 할 수 있다.

440174

25.7, 25.4, 23.4, 22.10, 필기 25.8, 25.5, 23.2, 22.4, 21.5

115 구조 패턴 A

구조 패턴(Structural Pattern)은 구조가 복잡한 시스템을 개발하기 쉽도록 클래스나 객체들을 조합하여 더 큰 구조로 만드는 패턴이다.

25.4, 필기 25.8, 25.5, 23.2, 22.4, 21.5

어댑터(Adapter)

- 호환성이 없는 클래스들의 인터페이스를 다른 클래스가 이용할 수 있도록 변환해주는 패턴이다.
- 기존의 클래스를 이용하고 싶지만 인터페이스가 일치하지 않을 때 이용한다.

22.10, 필기 23.2, 22.4, 21.5

브리지(Bridge)

- 구현부에서 추상층을 분리하여 서로가 독립적으로 확장할 수 있도록 구성한 패턴이다.
- 기능과 구현을 두 개의 별도 클래스로 구현한다.

필기 25.8, 23.2

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.

필기 25.5, 23.2

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.

25.7, 23.4, 필기 25.5, 22.4

프록시(Proxy)

- 복잡한 시스템을 개발하기 쉽도록 클래스나 객체들을 조합하는 패턴으로, 대리자라고도 불린다.
- 내부에서는 객체 간의 복잡한 관계를 단순하게 정리해 주고, 외부에서는 객체의 세부적인 내용을 숨겨주는 역할을 수행한다.

340207 24.10, 24.7, 22.10, 21.7, 20.7, 필기 25.5, 24.2, 23.5, 23.2, 21.8, 21.5, 20.8, 20.6

116 행위 패턴 Ⓐ

행위 패턴(Behavioral Pattern)은 클래스나 객체들이 서로 상호작용하는 방법이나 책임 분배 방법을 정의하는 패턴이다.

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.

필기 20.8

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 사한다.

24.7

반복자(Iterator)

- 자료 구조와 같이 접근이 잦은 객체에 대해 동일한 인터페이스를 사용하도록 하는 패턴이다.
- 내부 표현 방법의 노출 없이 순차적인 접근이 가능하다.

필기 23.2, 21.5

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.

22.10, 20.7, 필기 20.8

옵서버(Observer)

- 한 객체의 상태가 변화하면 객체에 상속되어 있는 다른 객체들에게 변화된 상태를 전달하는 패턴이다.
- 일대다의 의존성을 정의한다.
- 주로 분산된 시스템 간에 이벤트를 생성 · 발행(Publish)하고, 이를 수신(Subscribe)해야 할 때 이용한다.

필기 20.8

상태(State)

- 객체의 상태에 따라 동일한 동작을 다르게 처리해야 할 때 사용하는 패턴이다.
- 객체 상태를 캡슐화하고 이를 참조하는 방식으로 처리한다.

필기 24.2, 21.8

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.

필기 23.2

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.

필기 25.5, 20.6

방문자(Visitor)

- 각 클래스들의 데이터 구조에서 처리 기능을 분리하여 별도의 클래스로 구성하는 패턴이다.
- 분리된 처리 기능은 각 클래스를 방문(Visit)하여 수행한다.

5장 인터페이스 구현

340216 필기 25.8, 25.2, 24.7, 24.5, 24.2, 20.8, 20.6

117 요구사항 검증 방법 Ⓑ

필기 25.8, 25.2, 24.7, 24.5, 24.2, 20.8, 20.6

요구사항 검토(Requirements Review)

- 요구사항 명세서의 오류 확인 및 표준 준수 여부 등의 결함 여부를 검토 담당자들이 수작업으로 분석하는 방법이다.
- 동료검토(Peer Review) : 요구사항 명세서 작성자가 명세서 내용을 직접 설명하고 동료들이 이를 들으면서 결함을 발견하는 형태의 검토 방법
- 워크스루(Walk Through) : 검토 회의 전에 요구사항 명세서를 미리 배포하여 사전 검토한 후에 짧은 검토 회의를 통해 결함을 발견하는 형태의 검토 방법
- 인스펙션(Inspection) : 요구사항 명세서 작성자를 제외한 다른 검토 전문가들이 요구사항 명세서를 확인하면서 결함을 발견하는 형태의 검토 방법

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 케이스를 생성하여 요구사항이 현실적으로 테스트 가능한지를 검토한다.

CASE 도구 활용

일관성 분석(Consistency Analysis)을 통해 요구사항 변경사항의 추적, 분석, 관리, 표준 준수 여부를 확인한다.

340222 필기 25.2, 24.7, 24.2, 22.7, 21.8, 21.3, 20.9, 20.8

118 미들웨어 Ⓑ

- 미들웨어(Middleware)는 **운영체제와 응용 프로그램**, 또는 서버와 클라이언트 **사이에서 다양한 서비스를 제공하는 소프트웨어**이다.
- 미들웨어는 표준화된 인터페이스를 제공함으로써 시스템 간의 데이터 교환에 일관성을 보장한다.
- **미들웨어의 종류** : DB, RPC, MOM, TP-Monitor, ORB, WAS

340223 필기 25.8, 23.7, 23.5, 23.2, 22.4, 21.3, 20.6

119 미들웨어의 종류 Ⓑ

필기 25.8, 23.2

DB(DataBase)

- 데이터베이스 벤더에서 제공하는 클라이언트에서 원격의 데이터베이스와 연결하는 미들웨어이다.
- DB를 사용하여 시스템을 구축하는 경우 보통 2-Tier 아키텍처라고 한다.

필기 23.7, 21.3

RPC(Remote Procedure Call, 원격 프로시저 호출)

응용 프로그램의 프로시저를 사용하여 원격 프로시저를 마치 로컬 프로시저처럼 호출하는 미들웨어이다.

필기 25.8, 23.7, 23.2, 22.4

MOM(Message Oriented Middleware, 메시지 지향 미들웨어)

- 메시지 기반의 비동기형 메시지를 전달하는 미들웨어이다.
- 온라인 업무보다는 이기종 분산 데이터 시스템의 데이터 동기를 위해 많이 사용된다.

필기 25.8, 23.5, 20.6

TP-Monitor(Transaction Processing Monitor, 트랜잭션 처리 모니터)

- 온라인 트랜잭션 업무에서 트랜잭션을 처리 및 감시하는 미들웨어이다.

- 항공기나 철도 예약 업무 등 사용자 수가 증가해도 빠른 응답 속도를 유지해야 하는 업무에 주로 사용된다.

필기 25.8, 23.2

ORB(Object Request Broker, 객체 요청 브로커)

- 코바(CORBA) 표준 스펙을 구현한 객체 지향 미들웨어이다.
- 최근에는 TP-Monitor의 장점인 트랜잭션 처리와 모니터링 등을 추가로 구현한 제품도 있다.

필기 25.8, 23.7, 23.2

WAS(Web Application Server)

- 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위한 미들웨어이다.
- 클라이언트/서버 환경보다는 웹 환경을 구현하기 위한 미들웨어이다.
- HTTP 세션 처리를 위한 웹 서버 기능뿐만 아니라 미션-크리티컬한 기업 업무까지 JAVA, EJB 컴포넌트 기반으로 구현이 가능하다.

340225

21.4, 20.10, 필기 23.2, 22.7, 21.8, 20.9

120 EAI A

EAI(Enterprise Application Integration)는 **기업 내 각종 애플리케이션 및 플랫폼 간**의 정보 전달, 연계, 통합 등 **상호 연동이 가능하게 해주는 솔루션**이다.

20.10, 필기 23.2, 22.7, 20.6

Point-to-Point

- 가장 기본적인 애플리케이션 통합 방식이다.
- 애플리케이션을 1:1로 연결한다.
- 변경 및 재사용이 어렵다.

20.10, 필기 23.2, 22.7, 20.6

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.

20.10, 필기 23.2, 22.7, 21.8, 20.6

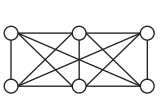
Message Bus(ESB 방식)

- 애플리케이션 사이에 미들웨어를 두어 처리하는 방식이다.
- 확장성이 뛰어나며 대용량 처리가 가능하다.

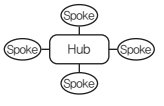
20.10, 필기 20.9

Bus Hybrid

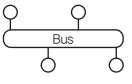
- Hub & Spoke와 Message Bus의 혼합 방식이다.
- 그룹 내에서는 Hub & Spoke 방식을, 그룹 간에는 Message Bus 방식을 사용한다.
- 필요한 경우 한 가지 방식으로 EAI 구현이 가능하다.
- 데이터 병목 현상을 최소화할 수 있다.



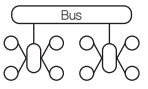
Point-to-Point



Hub & Spoke



Message Bus



Bus Hybrid

340229

20.5, 필기 24.5, 24.2, 23.5, 22.4, 20.6

121 JSON A

- JSON(JavaScript Object Notation)은 웹과 컴퓨터 프로그램에서 용량이 적은 데이터를 교환하기 위해 **데이터 객체를 속성 · 값의 쌍(Attribute-Value Pairs) 형태로 표현하는 개방형 표준 포맷**이다.
- 비동기 처리에 사용되는 AJAX에서 XML을 대체하여 사용되고 있다.

340230

25.7, 23.4, 20.7, 필기 20.8

122 AJAX Ⓐ

- AJAX(Asynchronous JavaScript and XML)는 자바 스크립트(JavaScript)를 사용하여 **클라이언트와 서버 간에 XML 데이터를 주고 받는 비동기 통신 기술**이다.
- 전체 페이지를 새로 고치지 않고도 웹 페이지 일부 영역만을 업데이트할 수 있다.

340231

필기 20.9, 20.8, 20.6

123 인터페이스 보안 기능 적용 Ⓑ

- 인터페이스 보안은 인터페이스의 보안성 향상을 위해 **인터페이스의 보안 취약점을 분석한 후 적절한 보안 기능을 적용**하는 것이다.
- 인터페이스 보안 기능은 일반적으로 네트워크, 애플리케이션, 데이터베이스 영역에 적용한다.

필기 20.9, 20.8, 20.6 **네트워크 영역**	• 인터페이스 송 · 수신 간 스니핑(Sniffing) 등을 이용한 데이터 탈취 및 변조 위협을 방지하기 위해 네트워크 트래픽에 대한 암호화를 설정함 • 암호화는 인터페이스 아키텍처에 따라 IPSec, SSL, S-HTTP 등의 다양한 방식으로 적용함
애플리케이션 영역	소프트웨어 개발 보안 가이드를 참조하여 애플리케이션 코드 상의 보안 취약점을 보완하는 방향으로 애플리케이션 보안 기능을 적용함
데이터베이스 영역	데이터베이스, 스키마, 엔티티의 접근 권한과 프로시저(Procedure), 트리거(Trigger) 등 데이터베이스 동작 객체의 보안 취약점에 보안 기능을 적용함

340232

24.7, 20.7, 필기 25.8, 24.7, 24.2, 22.7, 21.5

124 IPsec / SSL / S-HTTP Ⓐ

- IPsec(IP Security)
  - 네트워크 계층에서 IP 패킷 단위의 데이터 변조 방지 및 은닉 기능을 제공하는 프로토콜이다.
  - 암호화 수행 시 암호화와 복호화가 모두 가능한 양방향 암호화 방식을 사용한다.
  - **구성 요소** : AH, ESP
  - **운영 모드** : 터널 모드, 전송 모드
- SSL(Secure Sockets Layer) : TCP/IP 계층과 애플리케이션 계층 사이에서 인증, 암호화, 무결성을 보장하는 프로토콜
- S-HTTP(Secure Hypertext Transfer Protocol) : 클라이언트와 서버 간에 전송되는 모든 메시지를 암호화하는 프로토콜

340233

필기 24.2, 21.3, 20.6

125 데이터 무결성 검사 도구 Ⓒ

- 데이터 무결성 검사 도구는 인터페이스 보안 취약점을 분석하는데 사용되는 도구이다.
- 데이터 무결성 검사 도구는 시스템 파일의 변경 유무를 확인하고, 파일이 변경되었을 경우 이를 관리자에게 알려준다.
- **종류** : Tripwire, AIDE, Samhain, Claymore, Slipwire, Fcheck 등

340234 22.5, 필기 24.7, 24.5, 24.2, 23.7, 23.2, 22.7, 22.4, 21.5, 20.9, 20.6

126 인터페이스 구현 검증 도구 Ⓐ

인터페이스 구현을 검증하기 위해서는 인터페이스 단위 기능과 시나리오 등을 기반으로 하는 통합 테스트가 필요하며, 통합 테스트를 수행하기 위해 사용하는 테스트 자동화 도구는 다음과 같다.

22.5, 필기 24.7, 24.5, 24.2, 23.7, 23.2, 22.7, 21.5, 20.9

xUnit

- 같은 테스트 코드를 여러 번 작성하지 않게 도와주며, 테스트마다 예상 결과를 기억할 필요가 없게 하는 자동화된 해법을 제공하는 단위 테스트 프레임워크이다.
- Smalltalk에 처음 적용되어 SUnit이라는 이름이었으나, Java용의 JUnit, C++용의 CppUnit, .NET용의 NUnit 등 다양한 언어에 적용되면서 xUnit으로 통칭되고 있다.

필기 24.7, 24.5, 24.2, 23.2, 22.4, 21.5, 20.9, 20.6

STAF

- 서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크이다.
- 크로스 플랫폼이나 분산 소프트웨어에서 테스트 환경을 조성할 수 있도록 지원한다.
- 분산 소프트웨어의 경우 각 분산 환경에 설치된 데몬(Daemon)이 프로그램 테스트에 대한 응답을 대신하며, 테스트가 완료되면 이를 통합하고 자동화하여 프로그램을 완성한다.

필기 24.7, 24.5, 23.7, 23.2

FitNesse

웹 기반 테스트 케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크이다.

필기 24.7, 24.5, 24.2, 23.7, 23.2, 22.4, 20.9

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.

필기 23.7, 21.5

watir

인터프리터 방식의 객체 지향 스크립트 언어인 Ruby를 사용하는 애플리케이션 테스트 프레임워크이다.

6장 화면 설계

340237 21.7, 필기 21.8, 21.5

127 UI / UX A

21.7, 필기 21.8, 21.5

UI(User Interface, 사용자 인터페이스)

- UI는 **사용자와 시스템 간의 상호작용**이 원활하게 **이뤄지도록 도와주는 장치나 소프트웨어**를 의미한다.
- UI의 세 가지 분야
  - 정보 제공과 전달을 위한 물리적 제어에 관한 분야
  - 콘텐츠의 상세적인 표현과 전체적인 구성에 관한 분야
  - 모든 사용자가 편리하고 간편하게 사용하도록 하는 기능에 관한 분야

UX(User Experience, 사용자 경험)

- UX는 **사용자가 시스템이나 서비스를 이용하면서 느끼고 생각하게 되는 총체적인 경험**을 의미한다.
- UI가 사용성, 접근성, 편의성을 중시한다면 UX는 이러한 UI를 통해 사용자가 느끼는 만족이나 감정을 중시한다.

440190 22.5, 21.10, 필기 23.7, 22.7, 22.4, 21.8

128 UI의 구분 A

- CLI(Command Line Interface) : 명령과 출력이 텍스트 형태로 이뤄지는 인터페이스
- GUI(Graphical User Interface) : 아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스
- NUI(Natural User Interface) : 사용자의 말이나 행동 등 자연스러운 움직임을 통해 기기를 조작하는 인터페이스

340239 20.10, 20.7, 필기 22.7, 20.8, 20.6

129 UI의 기본 원칙 A

- **직관성** : 누구나 쉽게 이해하고 사용할 수 있어야 함
- **유효성** : 사용자의 목적을 정확하고 완벽하게 달성해야 함
- **학습성** : 누구나 쉽게 배우고 익힐 수 있어야 함
- **유연성** : 사용자의 요구사항을 최대한 수용하고 실수를 최소화해야 함

340249 필기 24.7, 24.5, 21.8, 21.3, 20.8, 20.6

130 ISO/IEC 9126의 소프트웨어 품질 특성 B

필기 20.6

기능성(Functionality)

- 소프트웨어가 사용자의 요구사항을 정확하게 만족하는 기능을 제공하는지 여부를 나타낸다.
- **하위 특성** : 적절성/적합성, 정밀성/정확성, 상호 운용성, 보안성, 준수성

필기 20.8

신뢰성(Reliability)

- 주어진 시간동안 주어진 기능을 오류 없이 수행할 수 있는 정도를 나타낸다.
- **하위 특성** : 성숙성, 고장 허용성, 회복성

필기 21.3

사용성(Usability)

- 사용자와 컴퓨터 사이에 발생하는 어떠한 행위에 대하여 사용자가 정확하게 이해하고 사용하며, 향후 다시 사용하고 싶은 정도를 나타낸다.
- **하위 특성** : 이해성, 학습성, 운용성, 친밀성

효율성(Efficiency)

- 사용자가 요구하는 기능을 얼마나 빠르게 처리할 수 있는지 정도를 나타낸다.
- **하위 특성** : 시간 효율성, 자원 효율성

유지 보수성(Maintainability)

- 환경의 변화 또는 새로운 요구사항이 발생했을 때 소프트웨어를 개선하거나 확장할 수 있는 정도를 나타낸다.
- **하위 특성** : 분석성, 변경성, 안정성, 시험성

필기 24.7, 24.5, 21.8

이식성(Portability)

- 소프트웨어가 다른 환경에서도 얼마나 쉽게 적용할 수 있는지 정도를 나타낸다.
- **하위 특성** : 적용성, 설치성, 대체성, 공존성

7장 애플리케이션 테스트 관리

340254

20.5, 필기 25.5, 24.7, 24.2, 22.7, 20.6

131 애플리케이션 테스트의 기본 원리 A

- **파레토 법칙(Pareto Principle)** : 애플리케이션의 20%에 해당하는 코드에서 전체 결함의 80%가 발견된다는 법칙
- **살충제 패러독스(Pesticide Paradox)** : 동일한 테스트 케이스로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 현상
- **오류-부재의 궤변(Absence of Errors Fallacy)** : 소프트웨어의 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 말할 수 없는 것

340255

20.7

132 프로그램 실행 여부에 따른 테스트 A

20.7

정적 테스트

- 프로그램을 실행하지 않고 명세서나 소스 코드를 대상으로 분석하는 테스트이다.
- 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도, 남은 결함 등을 발견하기 위해 사용한다.
- **종류** : 워크스루, 인스펙션, 코드 검사 등

동적 테스트

- 프로그램을 실행하여 오류를 찾는 테스트이다.
- 소프트웨어 개발의 모든 단계에서 테스트를 수행한다.
- **종류** : 블랙박스 테스트, 화이트박스 테스트

340260

필기 24.2, 23.7, 22.3, 21.5, 20.8, 20.6

133 화이트박스 테스트의 종류 B

필기 24.2, 23.7, 22.3, 21.5, 20.8, 20.6

기초 경로 검사(Base Path Testing)

- 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법이다.
- 대표적인 화이트박스 테스트 기법이다.

필기 22.3

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

340540

25.11, 25.7, 25.4, 24.10, 24.4, 23.7, 23.4, 21.7, 20.10

134 화이트박스 테스트의 검증 기준 A

24.10, 21.7

문장 검증 기준(Statement Coverage)

소스 코드의 모든 구문이 한 번 이상 수행되도록 테스트 케이스를 설계한다.

25.7, 25.4, 24.10, 23.4, 21.7, 20.10

결정 검증 기준(Decision Coverage)

- 소스 코드의 모든 조건문에 대해 조건식의 결과가 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스를 설계한다.
- 분기 검증 기준(Branch Coverage)이라고도 한다.

25.11, 24.10, 23.7, 21.7

조건 검증 기준(Condition Coverage)

소스 코드의 조건문에 포함된 개별 조건식의 결과가 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스를 설계한다.

조건/결정 검증 기준(Condition/Decision Coverage)

결정 검증 기준과 조건 검증 기준을 모두 만족하는 설계로, 조건문이 True인 경우와 False인 경우에 따라 조건 검증 기준의 입력 데이터를 구분하는 테스트 케이스를 설계한다.

24.4

변경 조건/결정 검증 기준(Modified Condition/Decision Coverage)

조건/결정 검증 기준을 향상시킨 검증 기준으로, 개별 조건식이 다른 개별 조건식의 영향을 받지 않고 전체 조건식의 결과에 독립적으로 영향을 주도록 테스트 케이스를 설계한다.

다중 조건 검증 기준(Multiple Condition Coverage)

소스 코드의 조건문에 포함된 모든 개별 조건식의 모든 조합을 고려하도록 테스트 케이스를 설계한다.

340262 20.10

135 블랙박스 테스트 A

- 블랙박스 테스트(Black Box Test)는 소프트웨어가 수행할 특정 기능을 알기 위해서 **각 기능이 완전히 작동되는 것을 입증하는 테스트**로, 기능 테스트라고도 한다.
- 사용자의 요구사항 명세를 보면서 테스트한다.
- 주로 구현된 기능을 테스트한다.
- 소프트웨어 인터페이스를 통해 실시된다.

340263 23.10, 22.10, 22.5, 21.10, 21.4, 20.11, 필기 25.8, 25.5, 25.2, 24.7, 24.5, 23.5, …

136 블랙박스 테스트의 종류 A

23.10, 22.5, 21.4, 20.11, 필기 25.8, 25.5, 24.7, 24.5, 23.5, 20.9, 20.8

동치 분할 검사(Equivalence Partitioning Testing)

- 프로그램의 입력 조건에 타당한 입력 자료와 타당하지 않은 입력 자료의 개수를 균등하게 하여 테스트 케이스를 정하고, 해당 입력 자료에 맞는 결과가 출력되는지 확인하는 기법이다.
- 동등 분할 기법 또는 동치 클래스 분해라고도 한다.

22.10, 22.5, 21.4, 필기 25.8, 25.5, 25.2, 24.7, 21.3, 20.9, 20.8, 20.6

경계값 분석(Boundary Value Analysis)

입력 조건의 중간값보다 경계값에서 오류가 발생될 확률이 높다는 점을 이용하여 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하는 기법이다.

22.5, 21.10, 필기 20.9

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.

필기 25.5, 24.7, 20.8

오류 예측 검사(Error Guessing)

과거의 경험이나 확인자의 감각으로 테스트하는 기법이다.

비교 검사(Comparison Testing)

여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법이다.

예제 A 애플리케이션에서 평가점수에 따른 성적부여 기준이 다음과 같을 때, 동치 분할 검사와 경계값 분석의 테스트 케이스를 확인하시오.

평가점수	성적
90~100	A
80~89	B
70~79	C
0~69	D

〈동치 분할 검사〉

테스트 케이스	1	2	3	4
입력값	60	75	82	96
예상 결과값	D	C	B	A
실제 결과값	D	C	B	A

해설 동치 분할 검사는 입력 자료에 초점을 맞춰 테스트 케이스를 만들어 검사하므로 평가점수를 입력한 후 점수에 맞는 성적이 출력되는지 확인한다.

〈경계값 분석〉

테스트 케이스	1	2	3	4	5	6	7	8	9	10
입력값	-1	0	69	70	79	80	89	90	100	101
예상 결과값	오류	D	D	C	C	B	B	A	A	오류
실제 결과값	오류	D	D	C	C	B	B	A	A	오류

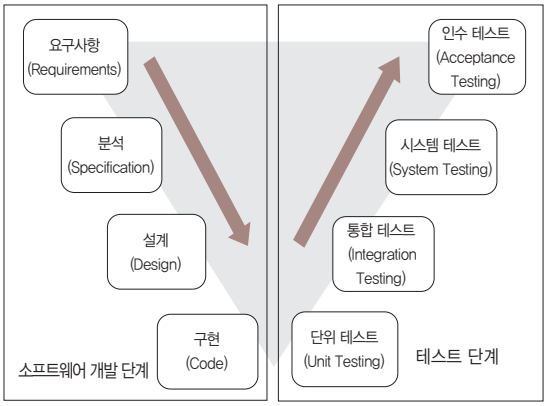
해설 경계값 분석은 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하므로 평가점수의 경계값에 해당하는 점수를 입력한 후 올바른 성적이 출력되는지 확인한다.

340264

22.5, 필기 24.7, 24.5, 22.3

137 개발 단계에 따른 애플리케이션 테스트 A

- 소프트웨어의 개발 단계에 따라 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트로 분류된다. 이렇게 분류된 것을 테스트 레벨이라고 한다.
- 애플리케이션 테스트와 소프트웨어 개발 단계를 연결하여 표현한 것을 V-모델이라고 한다.



소프트웨어 생명 주기의 V-모델

340265

21.4, 필기 25.5, 24.7, 23.5, 22.4, 21.8

138 단위 테스트 A

- 단위 테스트(Unit Test)는 코딩 직후 소프트웨어 설계의 최소 단위인 **모듈이나 컴포넌트에 초점을 맞춰 테스트**하는 것이다.
- 인터페이스, 외부적 I/O, 자료 구조, 독립적 기초 경로, 오류 처리 경로, 경계 조건 등을 검사한다.
- 사용자의 요구사항을 기반으로 한 기능성 테스트를 최우선으로 수행한다.
- 구조 기반 테스트와 명세 기반 테스트로 나뉘지만 주로 구조 기반 테스트를 시행한다.

340266

21.4

139 통합 테스트 A

- 통합 테스트(Integration Test)는 **단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트**를 의미한다.
- 모듈 간 또는 통합된 컴포넌트 간의 상호 작용 오류를 검사한다.
- 비점진적 통합 방식
  - 단계적으로 통합하는 절차 없이 모든 모듈이 미리 결합되어 있는 프로그램 전체를 테스트하는 방법
  - 종류 : 빅뱅 통합 테스트 방식
- 점진적 통합 방식
  - 모듈 단위로 단계적으로 통합하면서 테스트하는 방법
  - 종류 : 하향식 통합 테스트, 상향식 통합 테스트, 혼합식 통합 테스트

440215

22.7, 필기 25.8, 24.7, 24.5, 24.2, 23.7, 23.5, 22.3, 21.3, 20.9, 20.8, 20.6

140 인수 테스트 A

- 인수 테스트(Acceptance Test)는 개발한 소프트웨어가 **사용자의 요구사항을 충족하는지에 중점을 두고 테스트**하는 것이다.
- 인수 테스트는 개발한 소프트웨어를 사용자가 직접 테스트한다.
- 인수 테스트 종류

22.7, 필기 23.7, … **알파 테스트**	• 개발자의 장소에서 사용자가 개발자 앞에서 행하는 테스트 기법 • 테스트는 통제된 환경에서 행해지며, 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인하면서 기록함
22.7, 필기 22.3, … **베타 테스트**	• 선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법 • 실업무를 가지고 사용자가 직접 테스트

340269 21.7, 필기 25.8, 24.5, 24.2, 21.3, 20.8, 20.6

141 하향식 통합 테스트 Ⓐ

- 하향식 통합 테스트(Top Down Integration Test)는 프로그램의 **상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트**하는 기법이다.
- 하향식 통합 테스트 절차
  1. 주요 제어 모듈은 작성된 프로그램을 사용하고, 주요 제어 모듈의 종속 모듈들은 스텁(Stub)으로 대체한다.
  2. 깊이 우선 또는 넓이 우선 등의 통합 방식에 따라 하위 모듈인 스텁들이 한 번에 하나씩 실제 모듈로 교체된다.
  3. 모듈이 통합될 때마다 테스트를 실시한다.
  4. 새로운 오류가 발생하지 않음을 보증하기 위해 회귀 테스트를 실시한다.

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

340270 21.10, 필기 25.5, 22.7, 22.4, 21.8

142 상향식 통합 테스트 Ⓐ

- 상향식 통합 테스트(Bottom Up Integration Test)는 프로그램의 **하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트**하는 기법이다.
- 상향식 통합 테스트 절차
  1. 하위 모듈들을 클러스터(Cluster)로 결합한다.
  2. 상위 모듈에서 데이터의 입 · 출력을 확인하기 위해 더미 모듈인 드라이버(Driver)를 작성한다.
  3. 통합된 클러스터 단위로 테스트한다.
  4. 테스트가 완료되면 클러스터는 프로그램 구조의 상위로 이동하여 결합하고 드라이버는 실제 모듈로 대체된다.

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

440218 22.7

143 회귀 테스트 Ⓐ

- 회귀 테스트(Regression Test)는 **통합 테스트로 인해 변경된 모듈이나 컴포넌트에 새로운 오류가 있는지 확인하는 테스트**이다.
- 이미 테스트된 프로그램의 테스팅을 반복하는 것이다.
- 회귀 테스트는 수정한 모듈이나 컴포넌트가 다른 부분에 영향을 미치는지, 오류가 생기지 않았는지 테스트하여 새로운 오류가 발생하지 않음을 보증하기 위해 반복 테스트한다.

340275 필기 24.2, 23.2, 20.9

144 테스트 오라클 Ⓒ

- 테스트 오라클(Test Oracle)은 **테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참값을 대입하여 비교하는 기법** 및 활동을 말한다.
- 결과를 판단하기 위해 테스트 케이스에 대한 예상 결과를 계산하거나 확인한다.
- 테스트 오라클의 특징 : 제한된 검증, 수학적 기법, 자동화 기능

340276 20.11, 필기 25.2, 22.7

145 테스트 오라클의 종류 Ⓐ

- 참(True) 오라클 : 모든 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하는 오라클로, 발생된 모든 오류를 검출할 수 있음
- 샘플링(Sampling) 오라클 : 특정한 몇몇 테스트 케이스의 입력 값들에 대해서만 기대하는 결과를 제공하는 오라클로 전수 테스트가 불가능한 경우 사용하는 오라클
- 추정(Heuristic) 오라클 : 특정 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하고, 나머지 입력 값들에 대해서는 추정으로 처리하는 오라클
- 일관성 검사(Consistent) 오라클 : 애플리케이션에 변경이 있을 때, 테스트 케이스의 수행 전과 후의 결과 값이 동일한지를 확인하는 오라클

340283 20.5

146 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

340287 필기 25.5, 24.7, 23.7, 22.7, 20.6

147 빅오 표기법으로 표현한 최악의 시간 복잡도 Ⓑ

필기 23.7, 22.7, … O(1)	입력값(n)에 관계 없이 일정하게 문제 해결에 하나의 단계만을 거침 예 스택의 삽입(Push), 삭제(Pop)
$O(\log_2 n)$	문제 해결에 필요한 단계가 입력값(n) 또는 조건에 의해 감소함 예 이진 트리(Binary Tree), 이진 검색(Binary Search)
O(n)	문제 해결에 필요한 단계가 입력값(n)과 1:1의 관계를 가짐 예 for문
필기 25.5, 24.7, … $O(n\log_2 n)$	문제 해결에 필요한 단계가 $n(\log_2 n)$번만큼 수행됨 예 힙 정렬(Heap Sort), 2-Way 합병 정렬(Merge Sort)
$O(n^2)$	문제 해결에 필요한 단계가 입력값(n)의 제곱만큼 수행됨 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 수행됨 예 피보나치 수열(Fibonacci Sequence)

340288 필기 25.5, 24.7, 24.5, 23.2, 20.8

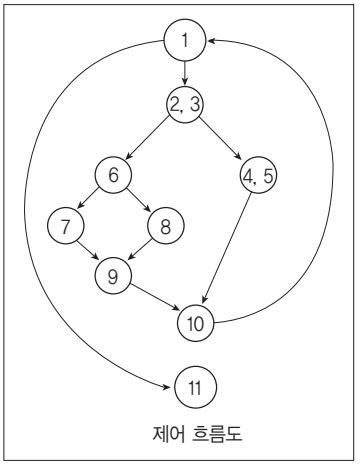
148 순환 복잡도 Ⓒ

- 순환 복잡도(Cyclomatic Complexity)는 한 프로그램의 **논리적인 복잡도를 측정하기 위한 소프트웨어**의 척도이다.
- 맥케이브 순환도(McCabe's Cyclomatic)라고도 한다.
- 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

예제 제어 흐름도가 다음과 같을 때 순환 복잡도(Cyclomatic Complexity)를 계산하시오.

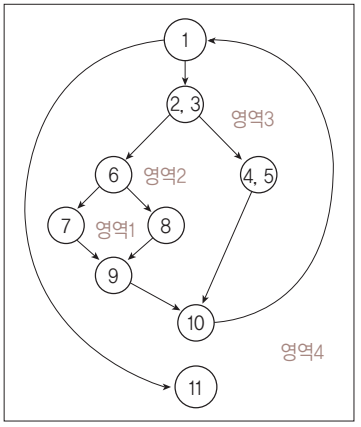


제어 흐름도

해설

순환 복잡도는 다음 두 가지 방법으로 구할 수 있습니다.

방법 1 제어 흐름도에서 화살표로 구분되는 각 영역의 개수를 구하면 4입니다.



방법 2 순환 복잡도 = 화살표의 수 − 노드의 수 + 2 이므로 11 − 9 + 2 = 4입니다.

440227

필기 24.7, 24.5, 22.3, 21.5, 20.6

149 소스 코드 최적화 Ⓑ

- 소스 코드 최적화는 나쁜 코드(Bad Code)를 배제하고, 클린 코드(Clean Code)로 작성하는 것이다.
- 나쁜 코드로 작성된 애플리케이션의 코드를 클린 코드로 수정하면 애플리케이션의 성능이 개선된다.
- **클린 코드(Clean Code)** : 누구나 쉽게 이해하고 수정 및 추가할 수 있는 단순, 명료한 코드, 즉 잘 작성된 코드
- **나쁜 코드(Bad Code)**
  - 프로그램의 로직(Logic)이 복잡하고 이해하기 어려운 코드
  - 스파게티 코드(Spaghetti Code) : 코드의 로직이 서로 복잡하게 얽혀 있는 코드
  - 외계인 코드(Alien Code) : 아주 오래되거나 참고문서 또는 개발자가 없어 유지보수 작업이 어려운 코드

340290

필기 24.2, 22.7, 22.3, 20.9, 20.8

150 클린 코드 작성 원칙 Ⓑ

필기 24.2, 22.7, 22.3

가독성

- 누구든지 코드를 쉽게 읽을 수 있도록 작성한다.
- 코드 작성 시 이해하기 쉬운 용어를 사용하거나 들여쓰기 기능 등을 사용한다.

필기 24.2, 22.7, 20.9, 20.8

단순성

- 코드를 간단하게 작성한다.
- 한 번에 한 가지를 처리하도록 코드를 작성하고 클래스/메소드/함수 등을 최소 단위로 분리한다.

필기 24.2, 22.7, 22.3, 20.8

의존성 배제

- 코드가 다른 모듈에 미치는 영향을 최소화한다.
- 코드 변경 시 다른 부분에 영향이 없도록 작성한다.

필기 24.2, 22.7, 22.3, 20.8

중복성 최소화

- 코드의 중복을 최소화한다.
- 중복된 코드는 삭제하고 공통된 코드를 사용한다.

필기 22.3

추상화

상위 클래스/메소드/함수에서는 간략하게 애플리케이션의 특성을 나타내고, 상세 내용은 하위 클래스/메소드/함수에서 구현한다.

340291

22.5, 필기 25.5, 23.7, 21.8, 20.9, 20.6

151 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 도구이다.

22.5, 필기 25.5, 23.7, 21.8, 20.9, 20.6

정적 분석(Static Analysis) 도구

- 작성한 소스 코드를 실행하지 않고 코딩 표준이나 코딩 스타일, 결함 등을 확인하는 코드 분석 도구이다.
- **종류** : pmd, cppcheck, SonarQube, checkstyle, ccm, cobertura 등

22.5

동적 분석(Dynamic Analysis) 도구

- 작성한 소스 코드를 실행하여 코드에 존재하는 메모리 누수, 스레드 결함 등을 분석하는 도구이다.
- **종류** : Avalanche, Valgrind 등

340292 필기 25.5, 25.2, 24.2, 23.7, 23.2, 21.8, 21.5, 21.3, 20.6

152 DDL B

- DDL(Data Define Language, 데이터 정의어)은 DB 구조, 데이터 형식, 접근 방식 등 **DB를 구축하거나 수정할 목적으로 사용하는 언어**이다.
- DDL의 3가지 유형

명령어	기능
필기 23.7, 23.2, 21.8, 21.5, … CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의함
필기 23.7, 23.2, 21.8, 20.6 ALTER	TABLE에 대한 정의를 변경하는 데 사용함
필기 23.7, 23.2, 21.8, 20.6 DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제함

340296 20.7

153 CREATE INDEX A

CREATE INDEX는 **인덱스를 정의하는 명령문**이다.

표기 형식

```
CREATE [UNIQUE] INDEX 인덱스명
ON 테이블명(속성명 [ASC | DESC] [,속성명 [ASC | DESC]])
[CLUSTER];
```

- UNIQUE
  - 사용된 경우 : 중복 값이 없는 속성으로 인덱스를 생성함
  - 생략된 경우 : 중복 값을 허용하는 속성으로 인덱스를 생성함
- 정렬 여부 지정
  - ASC : 오름차순 정렬
  - DESC : 내림차순 정렬
  - 생략된 경우 : 오름차순으로 정렬됨
- CLUSTER : 사용하면 인덱스가 클러스터드 인덱스로 설정됨

예제 〈고객〉 테이블에서 UNIQUE한 특성을 갖는 '고객번호' 속성에 대해 내림차순으로 정렬하여 '고객번호_idx'라는 이름으로 인덱스를 정의하시오.

```
CREATE UNIQUE INDEX 고객번호_idx
ON 고객(고객번호 DESC);
```

340297 20.10, 필기 21.3, 20.9

154 ALTER TABLE A

ALTER TABLE은 **테이블에 대한 정의를 변경하는 명령문**이다.

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 사용함
- ALTER : 특정 속성의 Default 값을 변경할 때 사용함
- DROP COLUMN : 특정 속성을 삭제할 때 사용함

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;
```

440235

23.7, 필기 25.8, 24.7, 23.2, 22.3, 21.5, 20.6

155 DROP Ⓐ

DROP은 **스키마, 도메인, 기본 테이블, 뷰 테이블, 인덱스, 제약 조건 등을 제거하는 명령문**이다.

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 제거함
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소함

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

```
DROP TABLE 학생 CASCADE;
```

340299

필기 24.7, 23.7, 21.5, 20.8

156 DCL Ⓑ

- DCL(Data Control Language, 데이터 제어어)은 **데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어**이다.
- DCL은 데이터베이스 관리자(DBA)가 데이터 관리를 목적으로 사용한다.

- DCL의 종류

명령어	기능
필기 20.8 COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려줌
필기 21.5, 20.8 ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구함
필기 20.8 GRANT	데이터베이스 사용자에게 사용 권한을 부여함
필기 24.7 REVOKE	데이터베이스 사용자의 사용 권한을 취소함

340300

21.10, 필기 25.8, 24.5, 24.2, 22.4, 20.9

157 GRANT / REVOKE Ⓐ

- 데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.
- GRANT : 권한 부여를 위한 명령어
- REVOKE : 권한 취소를 위한 명령어
- 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 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 1 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 2 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;
```

340301 20.7, 필기 21.5

158 ROLLBACK Ⓐ

- ROLLBACK은 **변경되었으나 아직 COMMIT되지 않은 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어**이다.
- 트랜잭션 전체가 성공적으로 끝나지 못하면 일부 변경된 내용만 데이터베이스에 반영되는 비일관성(Inconsistency) 상태가 될 수 있기 때문에 일부분만 완료된 트랜잭션은 롤백(Rollback)되어야 한다.

340303 필기 24.7, 23.5, 20.8, 20.6

159 DML Ⓑ

- DML(Data Manipulation Language, 데이터 조작어)은 데이터베이스 사용자가 **저장된 데이터를 실질적으로 관리하는데 사용되는 언어**이다.
- DML은 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
- DML의 유형

명령어	기능
필기 20.8, 20.6 SELECT	테이블에서 튜플을 검색함
필기 20.8, 20.6 INSERT	테이블에 새로운 튜플을 삽입함
필기 20.8, 20.6 DELETE	테이블에서 튜플을 삭제함
필기 23.5, 20.8, 20.6 UPDATE	테이블에서 튜플의 내용을 갱신함

※ 다음 테이블을 참조하여 핵심 160~162의 예제 결과를 확인하시오.

〈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

340543 24.7, 23.7, 필기 23.7, 23.5

160 삽입문(INSERT INTO~) Ⓐ

삽입문은 기본 테이블에 새로운 튜플을 삽입할 때 사한다.

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.

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';
```

340305

23.4, 20.10, 필기 24.5, 23.2

161 삭제문(DELETE FROM~) A

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.

예제 1 〈사원〉 테이블에서 "임꺽정"에 대한 튜플을 삭제하시오.

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

예제 2 〈사원〉 테이블에서 "인터넷" 부서에 대한 모든 튜플을 삭제하시오.

```
DELETE FROM 사원 WHERE 부서 = '인터넷';
```

예제 3 〈사원〉 테이블의 모든 레코드를 삭제하시오.

```
DELETE FROM 사원;
```

340306

24.7, 21.7, 필기 24.7, 23.7, 21.5, 20.9

162 갱신문(UPDATE~ SET~) A

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

일반 형식

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

예제 1 〈사원〉 테이블에서 "홍길동"의 '주소'를 "수색동"으로 수정하시오.

```
UPDATE 사원 SET 주소 = '수색동' WHERE 이름 = '홍길동';
```

예제 2 〈사원〉 테이블에서 "황진이"의 '부서'를 "기획부"로 변경하고 '기본급'을 5만원 인상시키시오.

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

340307

22.10, 20.5, 필기 25.5, 23.2, 22.3, 21.5, 20.8, 20.6

163 SELECT A

일반 형식

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속성명, …]
FROM 테이블명[, 테이블명, …]
[WHERE 조건]
[GROUP BY 속성명, 속성명, …]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- SELECT절
  - PREDICATE : 검색할 튜플 수를 제한하는 명령어를 기술함
    - ▶ DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 표시함
  - 속성명 : 검색하여 불러올 속성(열) 또는 속성을 이용한 수식을 지정함
  - AS : 속성이나 연산의 이름을 다른 이름으로 표시하기 위해 사용함
- FROM절 : 검색할 데이터가 들어있는 테이블 이름을 기술함
- WHERE절 : 검색할 조건을 기술함

- GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 사용한다. 일반적으로 GROUP BY절은 그룹 함수와 함께 사용됨
- HAVING절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정함
- ORDER BY절 : 데이터를 정렬하여 검색할 때 사용함
  - 속성명 : 정렬의 기준이 되는 속성명을 기술함
  - [ASC | DESC] : 정렬 방식으로, 'ASC'는 오름차순, 'DESC'는 내림차순임. 생략하면 오름차순으로 지정됨

※ 다음 테이블을 참조하여 핵심 164~167의 예제 결과를 확인하시오.

〈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

340308 24.7, 20.5

164 기본 검색 A

SELECT 절에 원하는 속성을 지정하여 검색한다.

예제 1 〈사원〉 테이블의 모든 튜플을 검색하시오.

```
SELECT * FROM 사원;
```

〈결과〉

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

예제 2 〈사원〉 테이블에서 '주소'만 검색하되 같은 '주소'는 한 번만 출력하시오.

```
SELECT DISTINCT 주소 FROM 사원;
```

〈결과〉

주소
망원동
서교동
합정동
대흥동
상암동
연남동

440245 25.11, 25.4, 22.7, 21.7, 20.7, 필기 25.5, 25.2, 24.7, 23.7, 23.5, 22.3, 21.8

165 조건 지정 검색 A

- WHERE 절에 다음 연산자들을 이용한 조건을 지정하여 조건에 만족하는 튜플만 검색한다.
- 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 사용됨

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

- IN 연산자 : 필드의 값이 IN 연산자의 수로 지정된 값과 같은 레코드만 검색하며, OR 연산을 수행한 결과와 같음

예제 1 〈사원〉 테이블에서 '기획'부의 모든 튜플을 검색하시오.

```
SELECT * FROM 사원 WHERE 부서 = '기획';
```

〈결과〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
성춘향	기획	02/20/64	대흥동	100
일지매	기획	04/29/78	연남동	110

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

예제 3 〈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

예제 4 〈사원〉 테이블에서 '생일'이 '01/01/69'에서 '12/31/73' 사이인 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 생일 BETWEEN #01/01/69# AND #12/31/73#;
```

〈결과〉

이름	부서	생일	주소	기본급
임꺽정	인터넷	01/09/69	서교동	80
김선달	편집	10/22/73	망원동	90

예제 5 〈사원〉 테이블에서 '주소'가 NULL인 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 주소 IS NULL;
```

〈결과〉

이름	부서	생일	주소	기본급
강건달	인터넷	12/11/80		90

340310

22.5, 21.7, 필기 25.2, 23.5, 22.4

166 정렬 검색 Ⓐ

ORDER BY 절에 특정 속성을 지정하여 지정된 속성으로 자료를 정렬하여 검색한다.

예제 〈사원〉 테이블에서 '부서'를 기준으로 오름차순 정렬하고, 같은 '부서'에 대해서는 '이름'을 기준으로 내림차순 정렬시켜서 검색하시오.

```
SELECT *
FROM 사원
ORDER BY 부서 ASC, 이름 DESC;
```

〈결과〉

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

440247

24.10, 24.4, 22.7, 필기 25.8, 24.7, 24.2, 23.5, 22.4, 21.5, 20.9, 20.6

167 하위 질의 A

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

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

예제 2 취미활동을 하지 않는 사원들을 검색하시오.

```
SELECT *
FROM 사원
WHERE 이름 NOT IN (SELECT 이름 FROM 여가활동);
```

〈결과〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
황진이	편집	07/21/75	합정동	100
장길산	편집	03/11/67	상암동	120
강건달	인터넷	12/11/80		90

예제 3 "망원동"에 거주하는 사원들의 '기본급'보다 적은 '기본급'을 받는 사원의 정보를 검색하시오.

```
SELECT 이름, 기본급, 주소
FROM 사원
WHERE 기본급 < ALL (SELECT 기본급 FROM 사원 WHERE
        주소 = "망원동");
```

〈결과〉

이름	기본급	주소
임꺽정	80	서교동

해설

ALL ( )는 하위 질의로 검색된 범위를 기본 질의의 조건으로 사용한다. 즉 〈사원〉 테이블에서 주소가 "망원동"인 사원의 기본급을 모두 추출한 후 추출된 기본급들을 기본 질의의 조건으로 사용한다.

❶ **SELECT 기본급 FROM 사원 WHERE 주소 = "망원동";** : 〈사원〉 테이블에서 '주소'가 "망원동"인 사원들의 '기본급'을 추출한다. 결과는 120, 90이다.

❷ **SELECT 이름, 기본급, 주소 FROM 사원 WHERE 기본급 〈ALL(❶)** : '기본급'이 ❶에서 추출된 '기본급'의 모든(ALL) 범위인 120, 90보다 작은(〈), 즉 범위 안에서 가장 작은 90보다 작은 기본급을 갖는 자료를 대상으로 '이름', '기본급', '주소'를 출력한다. 결과는 "임꺽정", "80", "서교동"이다.

340312

25.11, 23.4, 22.10, 21.4, 필기 24.5, 20.8

168 그룹 함수 A

그룹 함수는 GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 때 사용된다.

명령어	기능
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

440249 24.7, 23.4, 20.11, 20.10, 필기 21.8

169 그룹 지정 검색 A

GROUP BY절에 지정한 속성을 기준으로 자료를 그룹화하여 검색한다.

〈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 1 〈상여금〉 테이블에서 '부서'별 '상여금'의 평균을 구하시오.

```
SELECT 부서, AVG(상여금) AS 평균
FROM 상여금
GROUP BY 부서;
```

〈결과〉

부서	평균
기획	102.5
인터넷	70
편집	66

예제 2 〈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

440250 23.10, 필기 25.5, 24.7, 24.5, 23.5, 23.2, 22.3, 21.5

170 집합 연산자를 이용한 통합 질의 A

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.
- 집합 연산자의 종류(통합 질의의 종류)

집합 연산자	설명	집합 종류
UNION	• 두 SELECT문의 조회 결과를 통합하여 모두 출력함 • 중복된 행은 한 번만 출력함	합집합
UNION ALL	• 두 SELECT문의 조회 결과를 통합하여 모두출력함 • 중복된 행도 그대로 출력함	합집합
INTERSECT	두 SELECT문의 조회 결과 중 공통된 행만 출력함	교집합
EXCEPT	첫 번째 SELECT문의 조회 결과에서 두 번째 SELECT문의 조회 결과를 제외한 행을 출력함	차집합

340315 25.1

171 JOIN Ⓐ

- JOIN은 2개의 릴레이션에서 **연관된 튜플들을 결합하여, 하나의 새로운 릴레이션을 반환**한다.
- JOIN은 일반적으로 FROM절에 기술하지만, 릴레이션이 사용되는 곳 어디에나 사용할 수 있다.
- JOIN은 크게 INNER JOIN과 OUTER JOIN으로 구분된다.
  - INNER JOIN : THETA JOIN, EQUI JOIN, NATURAL JOIN, NON-EQUI JOIN
  - OUTER JOIN : LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN
- 조건이 없는 INNER JOIN을 수행하면 CROSS JOIN과 동일한 결과를 얻을 수 있다.
- CROSS JOIN(교차 조인)
  - 교차 조인은 조인하는 두 테이블에 있는 튜플들의 순서쌍을 결과로 반환한다.
  - 교차 조인의 결과로 반환되는 테이블의 행의 수는 두 테이블의 행 수를 곱한 것과 같다.

340545 25.4, 24.4, 필기 21.5

172 EQUI JOIN(동등 조인) Ⓐ

- JOIN에 참여하는 두 릴레이션의 속성 값을 비교하여 조건을 만족하는 튜플만 반환하는 조인을 THETA JOIN(세타 조인)이라고 하며, 조인에 사용되는 조건에는 =, ≠, <, ≤, >, ≥가 있다.
- EQUI JOIN(동등 조인)은 조인에 사용되는 조건 중 =(equal) 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 방법이다.
- EQUI JOIN의 결과 표시되는 속성(차수)은 두 릴레이션의 속성을 더한 것과 같다.
- EQUI JOIN에서 JOIN 조건이 '='일 때 동일한 속성이 두 번 나타나게 되는데, 이 중 중복된 속성을 제거하여 같은 속성을 한 번만 표기하는 방법을 NATURAL JOIN(자연 조인)이라고 한다.
- EQUI JOIN에서 연결 고리가 되는 공통 속성을 JOIN 속성이라고 한다.

- 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);
```

340317 21.7

173 OUTER JOIN Ⓐ

- LEFT OUTER JOIN : INNER JOIN의 결과를 구한 후, 우측 항 릴레이션의 어떤 튜플과도 맞지 않는 좌측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가함
  - 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1 LEFT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;
```

- RIGHT OUTER JOIN : INNER JOIN의 결과를 구한 후, 좌측 항 릴레이션의 어떤 튜플과도 맞지 않는 우측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가함
  - 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1 RIGHT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;
```

예제 〈학생〉 테이블과 〈학과〉 테이블에서 '학과코드' 값이 같은 튜플을 JOIN하여 '학번', '이름', '학과코드', '학과명'을 출력하는 SQL문을 작성하시오. 이때, '학과코드'가 입력되지 않은 학생도 출력하시오.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생 LEFT OUTER JOIN 학과
ON 학생.학과코드 = 학과.학과코드;
```

해설 INNER JOIN을 하면 '학과코드'가 입력되지 않은 "박치민"은 출력되지 않는다. 그러므로 JOIN 구문을 기준으로 왼쪽 테이블, 즉 〈학생〉의 자료는 모두 출력되는 LEFT JOIN을 사용한 것이다. 다음과 같이 JOIN 구문을 기준으로 테이블의 위치를 교환하여 RIGHT JOIN을 사용해도 결과는 같다.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학과 RIGHT OUTER JOIN 학생
ON 학과.학과코드 = 학생.학과코드;
```

〈결과〉

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

340318

필기 22.7, 20.6

174 트리거 Ⓒ

- 트리거(Trigger)는 데이터베이스 시스템에서 데이터의 삽입(Insert), 갱신(Update), 삭제(Delete) 등의 **이벤트(Event)가 발생할 때 관련 작업이 자동으로 수행되게 하는 절차형 SQL**이다.
- 트리거는 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력 등의 목적으로 사용된다.

9장 소프트웨어 개발 보안 구축

440256 20.11, 필기 24.7, 23.5, 23.2, 22.7, 22.4, 21.3, 20.8, 20.6

175 소프트웨어 개발 보안 요소 Ⓐ

소프트웨어 개발 보안 요소는 소프트웨어 개발에 있어 충족시켜야 할 요소 및 요건을 의미하며, 기밀성, 무결성, 가용성을 보안의 3대 요소라 한다.

필기 23.5, 22.7, 22.4, 21.3, 20.8

기밀성(Confidentiality)

- 시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용된다.
- 정보가 전송 중에 노출되더라도 데이터를 읽을 수 없다.

필기 24.7, 23.5, 23.2, 22.7, 22.4, 21.3, 20.8, 20.6

무결성(Integrity)

시스템 내의 정보는 오직 인가된 사용자만 수정할 수 있다.

20.11, 필기 23.5, 22.7, 22.4, 21.3, 20.8

가용성(Availability)

인가받은 사용자는 시스템 내의 정보와 자원을 언제라도 사용할 수 있다.

인증(Authentication)

- 시스템 내의 정보와 자원을 사용하려는 사용자가 합법적인 사용자인지를 확인하는 모든 행위이다.
- **대표적 방법** : 패스워드, 인증용 카드, 지문 검사 등

부인 방지(NonRepudiation)

데이터를 송 · 수신한 자가 송 · 수신 사실을 부인할 수 없도록 송 · 수신 증거를 제공한다.

340329 20.7, 필기 24.2, 23.7, 21.8

176 SQL 삽입 Ⓐ

- SQL 삽입(Injection)은 웹 응용 프로그램에 **SQL을 삽입하여** 내부 데이터베이스(DB) 서버의 **데이터를 유출 및 변조하고, 관리자 인증을 우회하는 보안 약점**이다.
- 동적 쿼리에 사용되는 입력 데이터에 예약어 및 특수문자가 입력되지 않게 필터링 되도록 설정하여 방지할 수 있다.

340336 필기 24.5, 21.5, 20.6

177 스택 가드 Ⓑ

- 스택 가드(Stack Guard)는 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 막는다.

340337 필기 24.5, 24.2, 20.9, 20.6

178 접근 제어자 Ⓑ

- 접근 제어자는 프로그래밍 언어에서 특정 개체를 선언할 때 **외부로부터의 접근을 제한하기 위해 사용되는 예약어**이다.
- 접근 제어자의 종류(접근 가능 : ○, 접근 불가능 : ×)

접근 제어자	클래스 내부	패키지 내부	하위 클래스	패키지 외부
필기 20.9, 20.6 Public	○	○	○	○
필기 20.6 Protected	○	○	○	×
필기 20.9 Default	○	○	×	×
필기 20.9, 20.6 Private	○	×	×	×

340338 23.7, 필기 23.5, 23.2, 21.3, 20.9, 20.6

179 개인키 암호화 기법 A

- 개인키 암호화(Private Key Encryption) 기법은 **동일한 키로 데이터를 암호화하고 복호화하는 암호화 기법**이다.
- 대칭 암호 기법 또는 단일키 암호화 기법이라고도 한다.
- 암호화와 복호화 속도가 빠르다.
- 관리해야 할 키의 수가 많다.
- 개인키 암호화 기법의 종류에는 스트림 암호화 방식과 블록 암호화 방식이 있다.

스트림 암호화 방식	• 평문과 동일한 길이의 스트림을 생성하여 비트 단위로 암호화 하는 방식 • 종류 : LFSR, RC4, TKIP
블록 암호화 방식	• 한 번에 하나의 데이터 블록을 암호화 하는 방식 • 종류 : DES, SEED, AES, ARIA, IDEA, Skipjack

440267 22.7

180 IDEA A

- IDEA(International Data Encryption Algorithm)는 **스위스의 라이(Lai)와 메시(Messey)가 1990년에 개발한 PES를 개선한 알고리즘**이다.
- 블록 크기는 64비트이고, 키 길이는 128비트이다.

440268 22.7

181 Skipjack A

- Skipjack은 **국가 안전 보장국(NSA)에서 개발한 암호화 알고리즘**이다.
- 클리퍼 칩(Clipper Chip)이라는 IC 칩에 내장되어 있다.
- 블록 크기는 64비트이고, 키 길이는 80비트이다.
- 주로 음성 통신 장비에 삽입되어 음성 데이터를 암호화한다.

340342 21.10

182 DES A

- DES(Data Encryption Standard)는 1975년 미국 NBS에서 발표한 개인키 암호화 알고리즘이다.
- 블록 크기는 64비트, 키 길이는 56비트이며 16회의 라운드를 수행한다.
- DES를 3번 적용하여 보안을 더욱 강화한 3DES(Triple DES)가 있다.

340343 24.7, 21.7, 필기 24.7, 22.7

183 AES A

- AES(Advanced Encryption Standard)는 2001년 미국 표준 기술 연구소(NIST)에서 발표한 개인키 암호화 알고리즘이다.
- DES의 한계를 느낀 NIST에서 공모한 후 발표하였다.
- 블록 크기는 128비트이며, 키 길이에 따라 AES-128, AES-192, AES-256으로 분류된다.

340344 필기 24.7, 23.7, 21.8, 20.8, 20.6

184 RSA B

- RSA(Rivest Shamir Adleman)는 1978년 MIT의 라이베스트(Rivest), 샤미르(Shamir), 애들먼(Adelman)에 의해 제 안된 공개키 암호화 알고리즘이다.
- 큰 숫자를 소인수분해 하기 어렵다는 것에 기반하여 만들어졌다.

340350 22.5

185 TKIP A

- TKIP(Temporal Key Integrity Protocol)는 기존의 무선 랜 보안 프로토콜인 **WEP의 취약성을 보완한 데이터 보안 프로토콜**이다.
- 암호 알고리즘의 입력 키 길이를 128비트로 늘리고 패킷당 키 할당, 키값 재설정 등의 키 관리 방식을 개선하였다.

340345 23.7, 필기 21.5, 21.3

186 해시 A

- 해시(Hash)는 임의의 길이의 **입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것**을 의미한다.
- 해시 알고리즘을 해시 함수라고 부르며, 해시 함수로 변환된 값이나 키를 해시값 또는 해시키라고 부른다.
- **종류** : SHA 시리즈, HAVAL, MD4, MD5, N-NASH, SNEFRU 등

340347 20.5

187 MD5 A

- MD5(Message Digest algorithm 5)는 1991년 R.Rivest가 MD4를 대체하기 위해 고안한 암호화 해시 함수이다.
- 블록 크기가 512비트이며, 키 길이는 128비트이다.

440278 필기 24.7, 24.5, 24.2, 23.7, 23.5, 23.2, 22.4

188 인증 B

- 인증(Authentication)은 다중 사용자 컴퓨터 시스템이나 네트워크 시스템에서 **로그인을 요청한 사용자의 정보를 확인하고 접근 권한을 검증하는 보안 절차**이다.
- 인증의 종류

종류	유형
지식 기반 인증 (Something You Know)	고정된 패스워 드, 패스 프레이즈, 아이핀(i-Pin) 등
소유 기반 인증 (Something You Have)	신분증, 메모리 카드, 스마트 카드, OTP 등
생체 기반 인증 (Something You Are)	지문, 홍채/망막, 얼굴, 음성, 정맥 등
위치 기반 인증 (Somewhere You Are)	GPS, IP, 콜백 등
행위 기반 인증 (Something You Do)	서명, 동작 등

340550 25.11

189 OAuth A

- OAuth(Open Authorization)는 인터넷 애플리케이션에서 사용자 인증에 사용되는 표준 인증 방법으로, 공개 API(OpenAPI)로 구현되었다.
- 인터넷 사용자가 웹사이트나 애플리케이션에 비밀번호를 제공하지 않고 자신에게 접근 권한을 부여하여 사용할 수 있다.
- 2010년 IETF에서 1.0이 공식 표준안으로 발표되었다.

440284 24.10, 22.7, 필기 20.9

190 VPN A

- VPN(Virtual Private Network, 가상 사설 통신망)은 인터넷 등 통신 사업자의 **공중 네트워크와 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안 솔루션**이다.
- SSL VPN : PC에 VPN Client 프로그램을 설치하여 VPN 서버에 접속하는 방식으로, 암호화를 위해 SSL 프로토콜을 사용함
- IPSec VPN : VPN 서버가 설치된 각각의 네트워크를 서로 연결하는 방식으로, 암호화를 위해 IPSec 프로토콜을 사용함

440286 22.10

191 SIEM A

- SIEM(Security Information and Event Management)은 다양한 장비에서 발생하는 **로그 및 보안 이벤트를 통합하여 관리하는** 빅 데이터 기반의 **보안 솔루션**이다.
- 방화벽, IDS, IPS, 웹 방화벽, VPN 등에서 발생한 로그 및 보안 이벤트를 통합하여 관리함으로써 비용 및 자원을 절약할 수 있다.
- 장기간의 로그 및 보안 이벤트를 수집 및 검색할 수 있는 빅데이터 기반의 통합 로그 수집 시스템이다.

440287 25.7, 23.4

192 SSH A

- SSH(Secure SHell, 시큐어 셸)는 **다른 컴퓨터에 원격으로 접속하여 작업을 수행할 수 있도록 다양한 기능을 지원하는 프로토콜** 또는 이를 이용한 응용 프로그램이다.
- 데이터 암호화와 강력한 인증 방법으로 보안성이 낮은 네트워크에서도 안전하게 통신할 수 있다.
- 키(key)를 통한 인증 방법을 사용하려면 사전에 클라이언트의 공개키를 서버에 등록해야 한다.
- 기본적으로는 22번 포트를 사용한다.

340362 21.10

193 AAA(=3A) A

- AAA는 다음 3가지 기능을 기반으로 사용자의 컴퓨터 자원 접근에 대한 처리와 서비스를 제공하는 기반 구조(Infrastructure) 또는 규격을 의미한다.
- 인증(Authentication) : 접근하는 사용자의 신원을 검증하는 기능
- 인가(Authorization) : 신원이 검증된 사용자에게 특정된 권한과 서비스를 허용하는 기능
- 과금(Accounting) : 사용자가 어떤 종류의 서비스를 이용했고, 얼마만큼의 자원을 사용했는지 기록 및 보관하는 기능

340363 22.5

194 정보보호 관리 체계 A

- 정보보호 관리 체계(ISMS; Information Security Management System)는 **정보 자산을 안전하게 보호하기 위한 보호 절차와 대책**을 의미한다.
- 조직에 맞는 정보보호 정책을 수립하고, 위험에 상시 대응하는 여러 보안 대책을 통합 관리 · 운용한다.
- 우리나라에서는 정보보호 관리 체계를 평가하고 인증하는 사업을 한국인터넷진흥원(KISA)에서 운영하고 있다.

340552 25.7

195 SYN Flooding A

- TCP(Transmission Control Protocol)는 신뢰성 있는 전송을 위해 3-wayhandshake를 거친 후에 데이터를 전송하게 되는데, SYN Flooding은 공격자가 가상의 클라이언트로 위장하여 3-way-handshake 과정을 의도적으로 중단시킴으로써 공격 대상지인 서버가 대기 상태에 놓여 정상적인 서비스를 수행하지 못하게 하는 공격 방법이다.
- 3-way-handshake
  - 신뢰성 있는 연결을 위해 송신지와 수신지 간의 통신에 앞서 3단계에 걸친 확인 작업을 수행한 후 통신을 수행한다.
  - 1단계 : 송신지에서 수신지로 'SYN' 패킷을 전송
  - 2단계 : 수신지에서 송신지로 'SYN + ACK' 패킷을 전송
  - 3단계 : 송신지에서 수신지로 'ACK' 패킷을 전송

340366 24.10, 필기 24.2, 23.5, 20.6

196 SMURFING A

- SMURFING(스머핑)은 **IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄으로써 네트워크를 불능 상태로 만드는 공격 방법**이다.
- SMURFING 공격을 무력화하는 방법 중 하나는 각 네트워크 라우터에서 브로드캐스트 주소를 사용할 수 없게 미리 설정해 놓는 것이다.

340369 20.5, 필기 24.2, 23.2

197 LAND Attack A

LAND Attack(Local Area Network Denial Attack)은 **패킷을 전송할 때 송신 IP 주소와 수신 IP 주소를 모두 공격 대상의 IP 주소로 하여** 공격 대상에게 전송하는 것으로, 이 패킷을 받은 공격 대상은 송신 IP 주소가 자신이므로 자신에게 응답을 수행하게 되는데, 이러한 패킷이 계속해서 전송될 경우 **자신에 대해 무한히 응답하게 하는 공격 방법**이다.

340513 25.4

198 스케어웨어 A

스케어웨어(Scareware)는 사용자에게 컴퓨터가 바이러스에 감염되었다고 속이거나 보안 위험이 있다고 거짓 경고 메시지를 보내어 불안감을 조성한 후 문제 해결을 명목으로 불필요한 소프트웨어 구매를 유도하는 악성 소프트웨어이다.

340372 25.4, 21.4, 필기 24.7, 22.7, 21.3

199 세션 하이재킹 A

- 세션 하이재킹(Session Hijacking)은 **상호 인증 과정을 거친 후 접속해 있는 서버와** 서로 접속되어 **클라이언트 사이의 세션 정보를 가로채는 공격 기법**으로, 접속을 위한 인증 정보 없이도 가로챈 세션을 이용해 공격자가 원래의 클라이언트인 것처럼 위장하여 서버의 자원이나 데이터를 무단으로 사용한다.
- TCP 3-Way-Handshake 과정에 끼어듦으로써 클라이언트와 서버 간의 동기화된 시퀀스 번호를 가로채 서버에 무단으로 접근하는 TCP 세션 하이재킹이 대표적인 예이다.

340373 21.10

200 ARP 스푸핑 A

ARP 스푸핑(ARP Spoofing)은 ARP의 취약점을 이용한 공격 기법으로, **자신의 물리적 주소(MAC)를 공격 대상의 것으로 변조하여** 공격 대상에게 도달해야 하는 **데이터 패킷을 가로채거나 방해하는 기법**이다.

※ **ARP(Address Resolution Protocol)** : 호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소(MAC Address)로 변환해주는 프로토콜

440300 22.10

201 사회 공학 A

사회 공학(Social Engineering)은 컴퓨터 보안에 있어서, **인간 상호 작용의 깊은 신뢰를 바탕으로** 사람들을 속여 정상 보안 절차를 깨트리기 위한 비기술적 **시스템 침입 수단**이다.

440301 22.10

202 다크 데이터 A

다크 데이터(Dark Data)는 특정 목적을 가지고 데이터를 수집하였으나, 이후 **활용되지 않고 저장만 되어 있는 대량의 데이터**를 의미한다.

440302 22.10

203 타이포스쿼팅 A

- 타이포스쿼팅(Typosquatting)은 네티즌들이 사이트에 접속할 때 **주소를 잘못 입력**하거나 철자를 빠뜨리는 **실수를 이용하기 위해 이와 유사한 유명 도메인을 미리 등록하는 것**이다.
- URL 하이재킹(Hijacking)이라고도 한다.

340374 20.11, 필기 23.2

204 스니핑 A

스니핑(Sniffing)은 **네트워크의 중간에서 남의 패킷 정보를 도청**하는 해킹 유형의 하나로, 수동적 공격에 해당한다.

340375 22.5

205 워터링 홀 Ⓐ

- 워터링 홀(Watering Hole)은 목표 대상이 자주 방문하는 **웹 사이트를 사전에 감염시켜** 대상이 해당 **사이트에 방문했을 때 악성 코드에 감염되게 하는** 웹 기반 **공격**이다.
- 감염된 PC를 기반으로 대상이 속한 조직의 중요 시스템에 접근하거나 불능으로 만드는 등의 영향력을 행사할 수 있다.

340376 필기 20.6

206 키로거 공격 Ⓒ

키로거 공격(Key Logger Attack)은 **컴퓨터 사용자의 키보드 움직임을 탐지해** ID, 패스워드, 계좌번호, 카드번호 등과 같은 개인의 중요한 **정보를 몰래 빼가는 해킹 공격**이다.

340377 필기 24.7, 21.8, 20.6

207 랜섬웨어 Ⓑ

랜섬웨어(Ransomware)는 인터넷 사용자의 컴퓨터에 잠입해 **내부 문서나 파일 등을 암호화해 사용자가 열지 못하게 하는 프로그램**으로, 암호 해독용 프로그램의 전달을 조건으로 사용자에게 돈을 요구하기도 한다.

340378 필기 23.7, 20.6

208 백도어 Ⓒ

- 백도어(Back Door, Trap Door)는 **시스템 설계자가** 서비스 기술자나 유지 보수 프로그램 작성자의 **액세스 편의를 위해 시스템 보안을 제거하여 만들어놓은 비밀 통로**로, 컴퓨터 범죄에 악용되기도 한다.
- **백도어 탐지 방법** : 무결성 검사, 열린 포트 확인, 로그 분석, SetUID 파일 검사 등

340379 24.4

209 기타 정보 보안 관련 용어 1 Ⓐ

24.4

Rootkit

- 시스템에 침입한 후 침입 사실을 숨긴 채 백도어, 트로이목마를 설치하고, 원격 접근, 내부 사용 흔적 삭제, 관리자 권한 획득 등 주로 불법적인 해킹에 사용되는 기능들을 제공하는 프로그램들의 모음이다.
- 컴퓨터의 운영체제에서 실행 파일과 실행 중인 프로세스를 숨김으로써 운영체제 검사 및 백신 프로그램의 탐지를 피할 수 있다.

24.4

APT(Advanced Persistent Threats, 지능형 지속 위협)

다양한 IT 기술과 방식들을 이용해 조직적으로 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 형태의 공격이다.

Ping of Death(죽음의 핑)

Ping 명령을 전송할 때 패킷의 크기를 인터넷 프로토콜 허용 범위 이상으로 전송하여 공격 대상의 네트워크를 마비시키는 서비스 거부 공격 방법이다.

큐싱(Qshing)

QR코드(Quick Response Code)를 통해 악성 앱의 다운로드를 유도하거나 악성 프로그램을 설치하도록 하는 금융사기 기법의 하나로, QR코드와 개인정보 및 금융정보를 낚는다(Fishing)는 의미의 합성 신조어이다.

스미싱(Smishing)

문자 메시지(SMS)를 이용해 사용자의 개인 신용 정보를 빼내는 수법이다.

스피어 피싱(Spear Phishing)

사회 공학의 한 기법으로, 특정 대상을 선정한 후 그 대상에게 일반적인 이메일로 위장한 메일을 지속적으로 발송하여, 발송 메일의 본문 링크나 첨부된 파일을 클릭하도록 유도해 사용자의 개인 정보를 탈취하는 공격이다.

440309

23.4, 필기 24.5, 22.7, 22.4

210 기타 정보 보안 관련 용어 2 Ⓐ

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.

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.

23.4

트로이 목마(Trojan Horse)

정상적인 기능을 하는 프로그램으로 위장하여 프로그램 내에 숨어 있다가 해당 프로그램이 동작할 때 활성화되어 부작용을 일으키지만 자기 복제 능력은 없다.

CC(Common Criteria) 인증

국가마다 서로 다른 정보보호시스템 평가기준을 연동하고 평가결과를 상호 인증하기 위해 제정된 정보보안 평가기준으로, ISO/IEC 15408에 등록된 국제 표준이다.

멀버타이징(Malvertising)

악성 소프트웨어를 뜻하는 멀웨어(Malware)와 광고(Advertising)의 합성어로, 온라인 광고를 통해 악성코드를 유포시키는 행위를 말한다.

정보공유분석센터(ISAC; Information Sharing & Analysis Center)

취약점 및 침해요인과 그 대응방안에 관한 정보를 제공하며, 침해사고가 발생하는 경우 실시간 경보 · 분석체계를 운영하고, 금융 · 통신 등 분야별 정보통신기반시설을 보호하기 위한 업무를 수행하는 곳이다.

침입 탐지 시스템(IDS; Intrusion DetectionSystem)

컴퓨터 시스템의 비정상적인 사용, 오용, 남용 등을 실시간으로 탐지하는 시스템이다.

데이터 디들링(Data Diddling)

처리할 자료를 다른 자료와 바꿔서 처리하는 것으로, 입력값이나 출력값을 부정한 의도로 수정하여 잘못된 결과가 나오도록 유도하는 방식이다.

비트로커(BitLocker)

Windows 7부터 지원되기 시작한 Windows 전용의 볼륨 암호화 기능으로, TPM(Trusted Platform Module)과 AES-128 알고리즘을 사용한다.

공급망 공격(Supply Chain Attack)

소프트웨어 공급망에 침투하여 악성코드를 배포하는 공격으로, SW 빌드 및 배포 과정에 악성코드를 삽입하여 이용자들을 공격한다.

23.4

바이러스(Virus)

컴퓨터의 정상적인 작동을 방해하기 위해 운영체제나 저장된 데이터에 손상을 입히는 프로그램으로, 자신을 복제할 수 있고, 다른 프로그램을 감염시킬 수 있다.

23.4, 필기 24.5, 22.7, 22.4

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.

10장 프로그래밍 언어 활용

340383

211 헝가리안 표기법

23.7, 20.10 Ⓐ

헝가리안 표기법(Hungarian Notation)이란 **변수명 작성 시** 변수의 자료형을 알 수 있도록 **자료형을 의미하는 문자를 포함하여 작성하는 방법**이다. 예를 들어 정수형 변수라는 것을 알 수 있도록 변수명에 int를 의미하는 i를 덧붙여 i_InputA, i_InputB, i_Result라고 하는 것과 같다.

예 int i_InputA : 정수형 변수

예 double d_Result : 배정도 실수형 변수

340384

212 주요 자료형

필기 25.5, 25.2, 21.3, 20.8 Ⓑ

종류	자료형	크기(C)	크기(Java)
필기 20.8 정수형	int	4Byte	4Byte
필기 25.5, 21.3 문자형	char	1Byte	2Byte
실수형	float	4Byte	4Byte
	double	8Byte	8Byte

340387

213 연산자 우선순위

21.10, 필기 25.2, 24.2 Ⓐ

대분류	중분류	연산자	결합규칙	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

25.7, 25.4, 24.10, 24.7, 24.4, 23.7, 23.4, 22.10, 22.7, 22.5, 20.5, 필기 25.8, 25.5, 24.2, 22.7

214 제어문 - C언어 Ⓐ

예제 다음 C언어로 구현된 프로그램의 실행 결과를 확인하시오.

```
#include <stdio.h>
main() {
❶       int score[] = { 86, 53, 95, 76, 61 };
❷       char grade;
❸       char str[] = "Rank";
❹⓫⓯    for (int i = 0; i < 5; i++) {
❺⓬         switch (score[i] / 10) {
              case 10:      'score[i]/10'이 10일 경우 찾아오는 곳으로, 'A'를 저장하기 위해 그냥 다음 문장으로 이동한다.
              case 9:       'score[i]/10'이 9일 경우 찾아오는 곳이다.
                 grade = 'A';       grade에 'A'를 저장한다.
                 break;             switch문을 벗어나 ❾⓮번으로 이동한다.
❻             case 8:
❼                grade = 'B';
❽                break;
              case 7:       'score[i]/10'이 7일 경우 찾아오는 곳이다.
                 grade = 'C';       grade에 'C'를 저장한다.
                 break;             switch문을 벗어나 ❾⓮번으로 이동한다.
⓭             default: grade = 'F';
              }
❾⓮            if (grade != 'F')
❿                printf("%d is %c %s\n", i + 1, grade, str);
          }
}
```

코드 해설

❶ int score[ ] = { 86, 53, 95, 76, 61 };

5개의 요소를 갖는 정수형 배열 score를 선언한다. 개수를 지정하지 않았으므로, 초기값으로 지정된 수만큼 배열의 요소가 만들어진다.

	[0]	[1]	[2]	[3]	[4]
score	86	53	95	76	61

• int : 배열에 저장할 자료의 형이다.

• score : 사용할 배열의 이름으로, 사용자가 임의로 지정한다.
• [ ] : 배열의 크기를 지정하는 것으로, 개수를 지정하지 않으면, 초기값으로 지정된 수만큼 배열의 요소가 만들어진다.

❷ 문자형 변수 grade를 선언한다.

❸ char str[ ] = "Rank";
• 문자형 배열 str을 선언하고 "Rank"로 초기화한다.
• 문자열을 저장하는 경우 문자열의 끝을 의미하는 널 문자('\0')가 추가로 저장되며, 출력 시 널 문자는 표시되지 않는다.

	[0]	[1]	[2]	[3]	[4]
str	R	a	n	k	\0

❹ for (int i = 0; i 〈 5; i++)
반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 for의 { } 부분을 반복 수행한다. 즉 for의 { } 부분인 ❺~❿번 문장을 5번 반복 수행한다.
• for는 반복문을 의미하는 예약어로 그대로 입력한다.
• i = 0 : 초기값을 지정할 수식이다.
• i 〈 5 : 최종값으로 지정할 수식이다.
• i++ : 증가값으로 사용할 수식이다.

❺ switch (score[i] / 10)
score[i]를 10으로 나눈 결과에 해당하는 숫자를 찾아간다. 현재 i가 0이므로 score[0]인 86을 10으로 나눈 결과는 8.60이지만 C, Java에서 정수형 변수에 저장된 값의 나눗셈은 결과도 정수가 되므로 결과는 8이다. 8에 해당하는 ❻번으로 이동한다.
• switch는 switch문에 사용되는 예약어로, 그대로 입력한다.
• score[i]/10 : case들의 값 중 하나를 도출하는 수식이다. 변수를 적어도 된다.

❻ 'score[i]/10'이 8일 경우 찾아오는 곳이다.

❼ grade에 'B'를 저장한다.

❽ switch문을 벗어나 ❾번으로 이동한다.

❾ if (grade != 'F')
grade가 'F'가 아니면 ❿번을 수행하고, 'F'이면 반복문의 처음인 ⓫번으로 이동한다. 현재 grade는 'B'이므로 ❿번으로 이동한다.
• if는 조건 판단문에 사용되는 예약어이므로 그대로 적는다.
• grade != 'F' : 조건식으로, 조건은 참(1) 또는 거짓(0)이 결과로 나올 수 있는 수식을 ( ) 안에 입력한다.

❿ printf("%d is %c %s\n", i + 1, grade, str);
i+1의 결과인 1을 정수(%d)로 출력한 후, 공백 한 칸과 is를 출력하고, 다시 공백 한 칸을 출력한다. 이어서 grade의 값 B를 문자(%c)로 출력하고 공백 한 칸을 출력한 후 str의 값 Rank를 문자열(%s)로 출력한다.

결과 1 is B Rank

이어서 제어문자 '\n'에 의해 커서가 다음 줄의 처음으로 이동된다.
for문의 반복이 아직 종료되지 않았으므로 제어는 ⓫번으로 이동한다.

⓫ 증가값(i++)에 의해 i는 1이 되고 최종값(i〈5) 조건을 만족하므로 for의 { } 부분을 수행한다.

⓬ 현재 i가 1이므로 score[1]인 53을 10으로 나눈 결과 5에 해당하는 ⓭번으로 이동한다.

⓭ grade에 'F'를 저장한다.

⓮ grade가 'F'이므로 ❿번을 수행하지 않고 ⓯번으로 이동한다.

위와 같은 과정을 for문의 i가 5가 되어 최종값(i〈5) 조건을 만족하지 않을 때까지 3번 더 반복한다.

※ 반복문 실행에 따른 변수들의 변화는 다음과 같습니다.

i	score[i]	score[i]/10	grade	grade != 'F'	출력
0	86	8	B	참	1 is B Rank
1	53	5	F	거짓	
2	95	9	A	참	3 is A Rank
3	76	7	C	참	4 is C Rank
4	61	6	F	거짓	
5					

결과

```
1 is B Rank
3 is A Rank
4 is C Rank
```

440318

25.11, 25.7, 25.4, 24.7, 23.4, 22.10, 22.5, 21.4, 20.11, 20.10, 필기 24.2, 23.7, 22.4, 22.3, 21.8

215 제어문 - Java

A

예제 다음 Java로 구현된 프로그램의 실행 결과를 확인하시오.

```
public class Test {
   public static void main(String[] args) {
❶     String str = "agile";
❷     int x[] = { 1, 2, 3, 4, 5 };
❸     char y[] = new char[5];
❹     int i = 0;
❺     while (i < str.length()) {
❻        y[i] = str.charAt(i);
❼        i++;
      }
❽     for (int p : x) {
❾        i--;
❿        System.out.print(y[i]);
⓫        System.out.print(p + " ");
      }
   }
}
```

코드 해설

❶ 문자열 변수 str을 선언하고 "agile"로 초기화한다.

❷ int x[ ] = { 1, 2, 3, 4, 5 };

5개의 요소를 갖는 정수 배열 x를 선언하고 초기화한다. 개수를 지정하지 않으면, 초기값으로 지정된 수만큼 배열의 요소가 만들어진다.

	[0]	[1]	[2]	[3]	[4]
x	1	2	3	4	5

❸ char y[ ] = new char[5];

5개의 요소를 갖는 문자 배열 y를 선언한다.

	[0]	[1]	[2]	[3]	[4]
y					

- Java에서 배열은 객체로 취급되며, 객체 변수는 'new' 명령을 이용해서 생성해야 한다. 배열을 선언하면서 초기값을 지정하는 경우에는 ❷번과 같이 선언하면서 지정하면 된다.
- char y[ ] : y는 문자형 배열 변수라는 의미이다. Java에서는 'char[ ] y'와 같이 대괄호를 자료형 바로 뒤에 적는 것을 선호하는데, C 언어에서는 이렇게 표기할 수 없다.
- new char[5] : 5개의 요소를 갖는 문자형 배열을 생성한다.

❹ 정수형 변수 i를 선언하고 0으로 초기화한다.

❺ while (i < str.length( )) {

i가 str 변수의 길이보다 작은 동안 ❻~❼번을 반복 수행한다.

- while은 반복문에서 사용되는 예약어로 그대로 입력한다.
- (i < str.length( )) : 조건으로, 참이나 거짓을 결과로 갖는 수식을 조건에 입력한다. 참(1)을 직접 입력할 수도 있다.
  - length( ) 메소드 : 변수의 크기를 반환한다. str에는 5글자의 문자열이 저장되어 있으므로 str.length( )는 5이다.

❻ y[i] = str.charAt(i);

y[i]에 str의 i번째에 있는 문자를 저장한다.

❼ 'i = i + 1;'과 동일하다. i의 값을 1씩 누적시킨다.

❺~❼번 반복문의 수행에 따른 변수들의 변화는 다음과 같다.

i	str	str.length()	str.charAt(i)	y[]
0	agile	5	a	a
1			g	a g
2			i	a g i
3			l	a g i l
4			e	a g i l e
5				

❽ for (int p : x) {

향상된 반복문이다. p는 x 배열의 각 요소의 값을 차례로 받으면서 x 배열의 요소 수만큼 ❾~⓫번을 반복 수행한다. x 배열이 5개의 요소를 가지므로 각 요소를 p에 저장하면서 ❾~⓫번을 5회 수행한다.

- int p : x 배열의 각 요소를 일시적으로 저장하기 위해 선언한 변수이다. x 배열과 형이 같아야 한다. x 배열이 정수면 정수, 문자면 문자여야 한다.
- x : 배열의 이름이다.

❾ 'i = i - 1;'과 동일하다. i의 값을 1씩 감소시킨다.

❿ y[i]의 값을 출력한다.

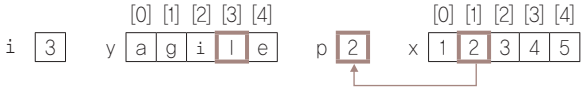
⓫ p의 값을 출력하고, 이어서 공백 한 칸을 출력한다.

변수의 변화는 다음과 같다. ❺번 반복문을 수행한 후 i는 5였다가 ❾번을 수행한 후 i는 4가 되었다.

- **❽~⓫의 첫 번째 수행** : x 배열의 첫 번째 값이 p를 거쳐 y[4]의 값과 함께 출력된다.

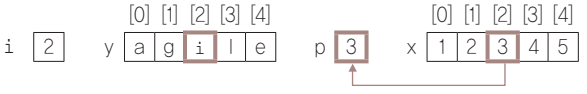
결과 e1

• ❽~⓫의 두 번째 수행 : x 배열의 두 번째 값이 p를 거쳐 y[3]의 값과 함께 출력된다.



결과 e1 l2

• ❽~⓫의 세 번째 수행 : x 배열의 세 번째 값이 p를 거쳐 y[2]의 값과 함께 출력된다.



결과 e1 l2 i3

이런 방식으로 x 배열의 요소 수만큼 2번 더 반복한다.

❽~⓫번 반복문의 수행에 따른 변수들의 변화는 다음과 같다.

p	i	y[i]	출력
1	4	e	e1
2	3	l	e1 l2
3	2	i	e1 l2 i3
4	1	g	e1 l2 i3 g4
5	0	a	e1 l2 i3 g4 a5

440319

25.4, 23.4, 20.10, 필기 24.2, 23.2, 21.3

216 break와 continue

A

• break : 반복문이나 switch문 안에서 break가 나오면 블록을 벗어남

• continue : 반복문에서 continue가 나오면 continue 이후의 문장을 실행하지 않고 제어를 반복문의 처음으로 옮김

25.11, 25.4, 24.10, 24.7, 24.4, 23.4, 22.7, 21.10, 21.7, 20.11, 필기 25.5, 25.2, 23.2, 22.3, 21.8, 21.5

217 포인터 - C언어 Ⓐ

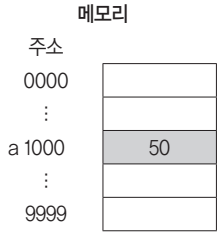
예제 다음 C언어로 구현된 프로그램의 실행 결과를 확인하시오.

```
#include <stdio.h>
main() {
❶    int a = 50;
❷    int *b = &a;
❸    *b = *b + 20;
❹    printf("%d, %d\n", a, *b);
❺    char *s;
❻    s = "gilbut";
❼    for (int i = 0; i < 6; i += 2) {
❽       printf("%c, ", s[i]);
❾       printf("%c, ", *(s + i));
❿       printf("%s\n", s + i);
     }
}
```

코드 해설

❶ 정수형 변수 a를 선언하고 50으로 초기화한다.

※ 주기억장치의 빈 공간 어딘가에 a라는 이름을 붙이고 그곳에 50을 저장합니다.(여기서 지정한 주소는 임의로 정한 것이며, 이해를 돕기 위해 주소를 실제 표현되는 16진수가 아니라 10진수로 표현했습니다.)

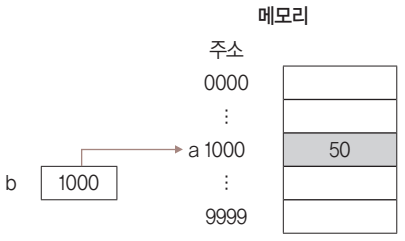


❷ int *b = &a;

정수형 변수가 저장된 곳의 주소를 기억할 포인터 변수 b를 선언하고, a의 주소로 초기화한다. b에는 a의 주소가 저장된다.

- int : 포인터 변수가 가리키는 곳에 저장되는 값의 자료형을 입력한다. 정수형 변수 a가 저장된 곳의 주소를 기억할 것이므로 int를 사용한다.
- *b : 포인터 변수를 선언할 때는 변수명 앞에 *를 붙인다.
- &a : 변수의 주소를 알아낼 때는 변수 앞에 번지 연산자 &를 붙인다.

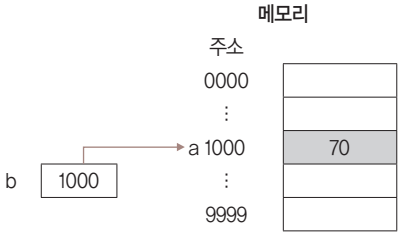
※ 변수 a의 주소가 b에 기억된다는 것은 b가 변수 a의 주소를 가리키고 있다는 의미입니다.



❸ *b = *b + 20;

b가 가리키는 곳의 값(*b)에 20을 더한다. b가 가리키는 곳이 a이므로 결국 a의 값도 바뀌는 것이다. 실행문에서 포인터 변수에 간접 연산자 *를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 말한다.

※ b가 가리키는 곳의 값에 20을 더해 다시 b가 가리키는 곳에 저장합니다. 그곳은 변수 a의 주소이므로 변수 a의 값도 저절로 변경되는 것입니다.



❹ printf("%d, %d\n", a, *b);

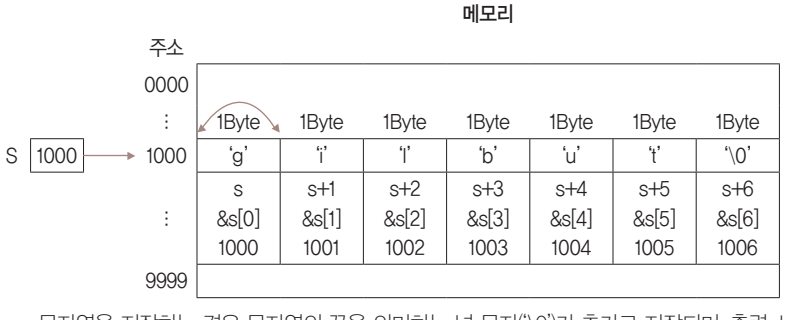
a의 값을 정수로 출력한 후 쉼표(,)와 공백 한 칸을 출력한다. 이어서 b가 가리키는 곳의 값(*b)을 정수로 출력한다.

결과 **70, 70**

❺ 문자형 변수가 저장된 곳의 주소를 기억할 포인터 변수 s를 선언한다.

❻ s = "gilbut";

- s는 주소를 기억하는 포인터 변수이므로 s에 "gilbut"가 기억되는 것이 아니라 "gilbut"라는 문자열이 메모리의 어딘가에 저장된 후 그 저장된 곳의 주소가 s에 기억된다.



- 문자열을 저장하는 경우 문자열의 끝을 의미하는 널 문자('\0')가 추가로 저장되며, 출력 시 널 문자는 표시되지 않는다.

❼ 반복 변수 i가 0부터 2씩 증가하면서 6보다 작은 동안 ❽~❿번을 반복 수행한다. i가 0인 상태에서 ❽번으로 이동한다.

❽ printf("%c, ", s[i]);

s[i], 즉 s[0]의 값을 문자로 출력한 후, 쉼표(,)와 공백 한 칸을 출력한다.

결과 **g,**

❾ printf("%c, ", *(s + i));

(s+i), 즉 (s+0)이 가리키는 곳의 값을 문자로 출력한 후, 쉼표(,)와 공백 한 칸을 출력한다.

결과 **g, g,**

❿ printf("%s\n", s + i);

(s+i), 즉 (s+0)의 위치부터 문자열의 끝('\0')까지의 모든 문자를 하나의 문자열로 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과 **g, g, gilbut**

※ ❼~❿번 반복문의 수행에 따른 변수들의 변화는 다음과 같습니다.

i	s[i]	s+i	*(s+i)	출력
0	g	1000	g	g, g, gilbut
2	l	1002	l	g, g, gilbut l, l, lbut
4	u	1004	u	g, g, gilbut l, l, lbut u, u, ut
6				

440321

25.11, 25.7, 25.4, 24.10, 24.7, 23.10, 22.7, 21.10, 21.4, 필기 24.5, 23.5, 23.2, 22.4

218 구조체 - C언어 Ⓐ

예제 다음 C언어로 구현된 프로그램의 실행 결과를 확인하시오.

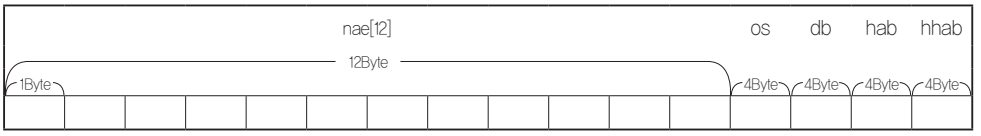
```
#include <stdio.h>
Ⓐ struct jsu {                구조체 jsu를 정의한다.
      char nae[12];                    12개의 요소를 갖는 문자 배열 nae를 선언한다.
      int os, db, hab, hhab;           정수형 변수 os, db, hab, hhab를 선언한다.
};

int main() {
❶    struct jsu st[3] = { {"데이터1", 95, 88}, {"데이터2", 84, 91},
                          {"데이터3", 86, 75} };
❷    struct jsu* p;
❸    p = &st[0];
❹    (p + 1)->hab = (p + 1)->os + (p + 2)->db;
❺    (p + 1)->hhab = (p + 1)->hab + p->os + p->db;
❻    printf("%d", (p + 1)->hab + (p + 1)->hhab);
}
```

코드 해설

Ⓐ 구조체 jsu의 구조

	nae[0]	nae[1]	nae[2]	…	nae[11]
char nae[12]					
int os					
int db					
int hab					
int hhab					

※ 위의 구조체는 다음과 같이 메모리의 연속된 공간에 저장된 후 사용됩니다.



모든 C 프로그램은 반드시 main( ) 함수에서 시작한다.

❶ 구조체 jsu 자료형으로 3개짜리 배열 st를 선언하고 초기화한다.

	char nae[12]	int os	int db	int hab	int hhab
st[0]	st[0].nae[0]~st[0].nae[11]	st[0].os	st[0].db	st[0].hab	st[0].hhab
st[1]	st[1].nae[0]~st[1].nae[11]	st[1].os	st[1].db	st[1].hab	st[1].hhab
st[2]	st[2].nae[0]~st[2].nae[11]	st[2].os	st[2].db	st[2].hab	st[2].hhab

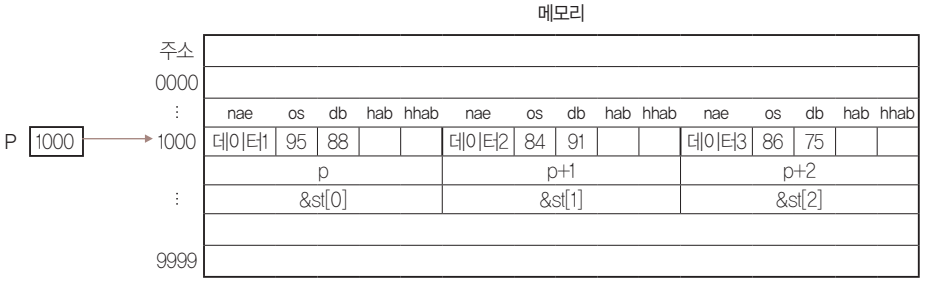
↓

	char nae[12]										int os	int db	int hab	int hhab
st[0]	데	이	터	1	\0						95	88		
st[1]	데	이	터	2	\0						84	91		
st[2]	데	이	터	3	\0						86	75		

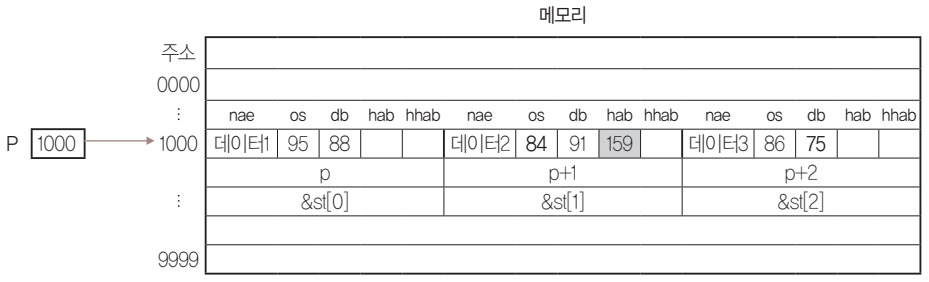
※ 문자열을 저장하는 경우 문자열의 끝을 의미하는 널 문자(\0)가 추가로 저장되며, 출력 시 널 문자는 표시되지 않습니다. 영문, 숫자는 1Byte, 한글은 2Byte를 차지합니다.

❷ 구조체 jsu의 포인터 변수 p를 선언한다.

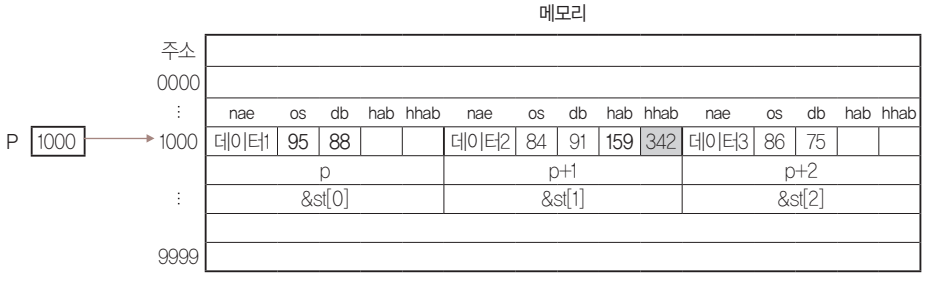
❸ p에 st 배열의 첫 번째 요소의 주소를 저장한다.



❹ p+1이 가리키는 곳의 멤버 hab에 p+1이 가리키는 곳의 멤버 os 값과 p+2가 가리키는 곳의 멤버 db 값을 더한 후 저장한다. p가 st[0]을 가리키므로 p+1은 st[1]을 p+2는 st[2]를 가리킨다. 따라서 st[1]의 os 값 84와 st[2]의 db 값 75를 더한 값 **159**를 st[1]의 hab에 저장한다.



❺ p+10이 가리키는 곳의 멤버 hhab에 p+10이 가리키는 곳의 멤버 hab 값과 p가 가리키는 곳의 멤버 os와 db 값을 모두 더한 후 저장한다. st[1]의 hab 값 159, st[0]의 os와 db 값 95와 88을 모두 더한 값 **342**를 st[1]의 hhab에 저장한다.



❻ p+10이 가리키는 곳의 멤버 hab와 hhab의 값을 더한 후 정수로 출력한다. 159와 342를 더한 **501**이 출력된다.

결과 **501**

▶440322

24.10, 24.7, 24.4, 23.10, 23.7, 23.4, 22.10, 22.7, 22.5, 21.7, 20.10, 20.5, 필기 22.4, 22.3

219 사용자 정의 함수 - C언어 Ⓐ

예제 다음은 재귀 함수를 이용해 팩토리얼(Factorial) 연산을 수행하는 C 프로그램이다. 그 실행 결과를 확인하시오.

```
#include <stdio.h>
int factorial(int n);

main() {
      int (*pf)(int);
      pf = factorial;
      printf("%d", pf(3));
}
```

사용할 사용자 정의 함수를 선언하는 곳이다. main( ) 함수에서 사용할 factorial이란 함수를 이곳에서 정의하는 것이다. 즉 factorial 함수를 프로그램에서 만들어 사용하겠다는 의미이다.
- int : 사용할 함수의 반환값이 정수임을 알려준다. 그대로 적어준다.
- factorial : 함수의 이름이다. main( ) 함수 뒤에서 정의한 이름과 일치해야 한다.
- (int n) : 인수를 저장할 변수이다. 호출하는 곳에서 보내준 인수와 자료형이 일치해야 한다.

```
int factorial(int n) {
      if (n <= 1)
           return 1;
      else
           return n * factorial(n - 1);
}
```

코드 해설

모든 C 프로그램은 반드시 main( ) 함수에서 시작한다.

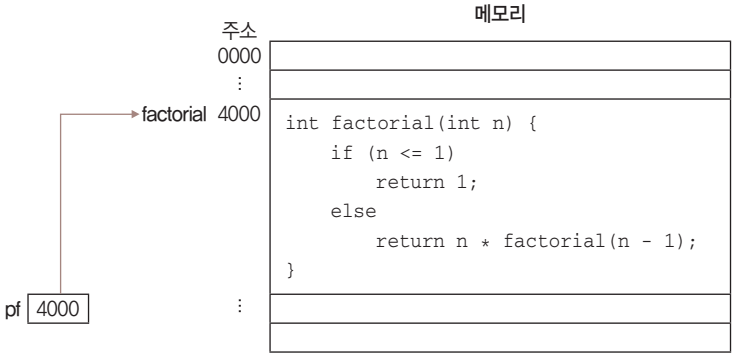
```
main( ) {
❶    int (*pf)(int);
❷    pf = factorial;
❸    printf("%d", pf(3));
}
```

❶ int (*pf)(int);

정수형 인수 한 개를 받는 정수형 함수 포인터 pf를 선언한다.

- int : 함수의 반환값이 정수임을 알려준다.
- (*pf) : 함수가 저장된 곳의 위치를 저장하는 함수 포인터의 이름이다.
- (int) ; 함수가 전달받을 인수의 자료형이다.

❷ factorial 함수의 시작 주소를 함수 포인터 pf에 저장한다. 즉 pf가 factorial( ) 함수의 시작 주소를 가리키고 있다는 것을 의미한다. 다음 그림에서 factorial( ) 함수가 할당된 공간의 주소는 임의로 정한 것이며, 이해를 돕기 위해 10진수로 표현했다.



❸ 3을 인수로 하여 pf( ) 함수를 호출한 다음 돌려받은 값을 정수형으로 출력한다. pf에는 factorial( ) 함수의 시작 주소가 저장되어 있으므로 함수 포인터 pf를 호출하는 것은 factorial( ) 함수를 호출하는 것과 같은 의미이다.

```
❹ int factorial(int n) {
❺     if (n <= 1)
          return 1;
      else
❻         return n * factorial(n – 1);
  }
```

❹ int factorial(int n) {

정수를 반환하는 factorial 함수의 시작점이다. ❸번에서 전달받은 3을 정수형 변수 n이 받는다.

- int : 함수의 반환값이 정수임을 알려준다.
- factorial : 함수의 이름이다. main( ) 함수 앞에서 선언한 이름과 일치해야 한다.
- (int n) : 호출하는 곳에서 보내준 인수를 저장할 변수이다. 호출하는 곳에서 보내준 인수와 자료형이 일치해야 한다.

❺ factorial 함수가 호출될 때 3을 전달받았으므로 n은 3이다. n의 값 3은 1보다 작거나 같지 않으므로 ❻번을 수행한다.

❻ return n * factorial(n – 1);

n * factorial(n–1)을 연산한 후 함수를 호출했던 ❸번으로 결과를 반환한다. ❻번을 수행하기 위해 factorial( ) 함수를 호출하는데, 호출할 때 전달되는 값은 factorial(n–1)이므로 factorial(2)인 상태로 호출된다.

```
❼ int factorial(int n) {
❽       if (n <= 1)
            return 1;
        else
❾           return n * factorial(n – 1);
    }
```

factorial 함수가 호출될 때 2를 전달받았으므로 n은 2이다. ❽번의 조건을 만족하지 않으므로 ❾번을 수행한다. ❾번을 수행하기 위해 factorial( ) 함수를 호출하는데, 호출할 때 전달되는 값은 factorial(n–1)이므로 factorial(1)인 상태로 호출된다.

```
❿ int factorial(int n) {
⓫       if (n <= 1)
⓬           return 1;
        else
            return n * factorial(n – 1);
    }
```

factorial 함수가 호출될 때 1을 전달받았으므로 n은 1이다. ⓫번의 조건을 만족하므로 ⓬번을 수행한다. 'return 1;'이므로 함수의 실행을 종료하고 1을 반환하면서 제어를 factorial(1) 함수를 호출했던 ⓭번으로 옮긴다.

```
❼ int factorial(int n) {
❽       if (n <= 1)
            return 1;
        else
❾⓭        return n * factorial(n – 1);
    }
```

⓭ return n * factorial(n – 1);

⓬번에서 1을 반환하였으므로 2를 반환하면서 제어를 factorial(2) 함수를 호출했던 ⓮번으로 옮긴다.

return n * factorial(n – 1);
ⓐ ⓑ

- ⓐ : 2 ('factorial(n–1)'을 호출할 때 n은 2였으므로)
- ⓑ : 1 (⓬번에서 1을 반환하였으므로)

```
❹ int factorial(int n) {
❺       if (n <= 1)
            return 1;
        else
❻⓮         return n * factorial(n – 1);
    }
```

⓮ return n * factorial(n - 1);

⓭번에서 2를 반환하였으므로 6을 반환하면서 제어를 factorial(3) 함수를 호출했던 ⓯번으로 옮긴다.

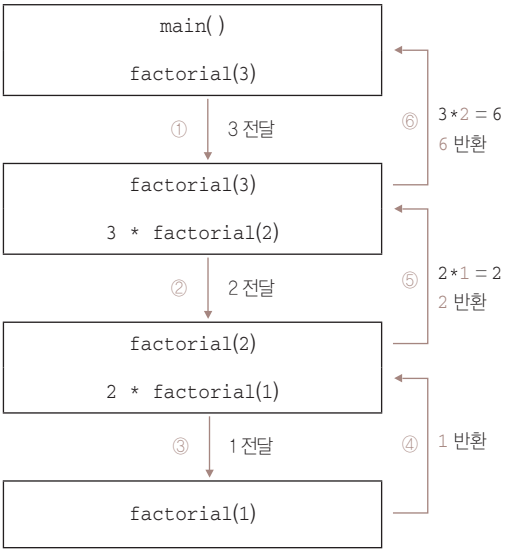
return n * factorial(n - 1);
3 2

```
main( ) {
❶       int (*pf)(int);
❷       pf = factorial;
❸⓯     printf("%d", pf(3));
}
```

⓯ ⓮번에서 6을 반환하였으므로 돌려받은 값 6을 정수형으로 출력하고 프로그램을 종료한다.

결과 6

지금까지의 재귀 함수 과정을 개괄적인 그림을 통해 살펴보자.



25.7, 24.10, 24.7, 24.4, 23.4, 22.10, 22.7, 22.5, 21.7, 20.10, 20.5, 필기 23.5, 23.2

220 Java의 클래스 1 Ⓐ

예제 다음은 Java에서 클래스를 만들고 객체를 생성해서 사용하는 간단한 프로그램이다. 어떤 일을 수행하는지 확인하시오.

```
Ⓐ class ClassA {
Ⓑ     int a = 10;
Ⓒ  ❹ int funcAdd(int x, int y) {
   ❺     return x + y + a;
      }
   }
Ⓓ public class Test {
Ⓔ     public static void main(String[] args) {
   ❶     int x = 3, y = 6, r;
   ❷     ClassA cal = new ClassA();
   ❸❻    r = cal.funcAdd(x, y);
   ❼     System.out.print(r);
      }
   }
```

코드 해설

Ⓐ class ClassA {

ClassA 클래스를 정의한다. Ⓑ~Ⓒ가 클래스의 범위이다.

- class : 클래스를 정의하는 명령어로, 꼭 써야 하는 예약어이다. 같은 파일에서 클래스를 정의할 때는 public을 두 번 사용하지 못한다. 실행 클래스에서 사용하므로 여기서는 사용할 수 없다. 그렇다는 것이다. 외우지는 마라.
- ClassA : 클래스 이름으로, 사용자가 원하는 이름을 임의로 지정할 수 있다. 단 첫 글자는 관례상 대문자로 지정한다.

Ⓑ 정수형 변수 a를 선언하고 10으로 초기화한다. Java에서는 클래스 안에 선언하는 변수를 클래스의 속성이라고 부른다.

Ⓒ int funcAdd(int x, int y) {

정수를 반환하는 funcAdd(int x, int y) 메소드를 정의한다. 호출문으로부터 정수형 인수 2개를 전달받아 각각 x와 y에 저장한다.

- int : 메소드의 반환값이 정수임을 알려준다.
- funcAdd : 메소드의 이름이다.
- (int x, int y) : 호출하는 곳에서 보내준 인수를 저장할 변수이다. 호출하는 곳에서 보내준 인수의 개수와 자료형이 일치해야 한다.

Ⓓ Test 클래스를 정의한다. 실행 클래스의 시작점으로 Java 프로그램은 아무리 작은 프로그램이라도 클래스를 만들어서 클래스 안에 실행문과 메소드(함수)를 만들고 실행해야 한다. 그리고 클래스 중에는 반드시 main( ) 메소드를 담고 있는 실행 클래스가 있어야 한다.

Ⓔ main( ) 메소드의 시작점이다. 여기서부터 실제 프로그램이 시작된다.

❶ 정수형 변수 x, y, r을 선언하고, x와 y를 각각 3과 6으로 초기화한다.

❷ ClassA cal = new ClassA( );

ClassA 클래스의 객체 변수 cal을 선언한다.

- ClassA : 클래스의 이름이다. 앞에서 정의한 클래스의 이름을 그대로 적어준다.
- cal : 객체 변수의 이름이다. 사용자가 원하는 이름을 적으면 된다.
- new : 객체 생성 예약어이다. 그대로 적어준다.
- ClassA( ) : 생성자이다.

❸ x와 y의 값 3과 6을 인수로 cal의 funcAdd( ) 메소드를 호출하여 반환받은 값을 r에 저장한다.
cal은 ClassA 클래스의 객체 변수이므로 ClassA의 funcAdd( ) 메소드인 ❹번이 호출된다.

❹ 정수를 반환하는 funcAdd( ) 메소드의 시작점이다. 호출문으로부터 정수형 인수 2개를 전달받아 각각 x와 y에 저장한다. ❸번에서 호출할 때 3과6을 전달했으므로 x는 3, y는 6이다.

❺ x + y + a를 연산한 후 메소드를 호출했던 ❻번으로 결과를 반환한다. x + y의 값은 9이고, a는 메소드에 없으므로 소속된 클래스에서 찾는다. ClassA의 a의 값이 10이므로 x + y + a(3 + 6 + 10)의 값은 19가 된다.

❻ ❺번에서 19가 반환되었으므로 r에 19를 저장한다.

❼ r의 값 19를 출력한다.

결과 19

340395

25.11, 25.7, 25.4, 24.10, 24.7, 23.4, 22.7, 21.7, 20.10, 20.7, 20.5

221 Java의 클래스 2 Ⓐ

예제 다음 Java 프로그램의 실행 결과를 확인하시오.

```
Ⓐ class ClassA {
Ⓑ  ❺ ClassA() {
     ❻    System.out.print('A');
     ❼    this.prn();
     ❿ }
Ⓒ    void prn() {
        System.out.print('B');
     }
   }
Ⓓ class ClassB extends ClassA {
Ⓔ  ❸ ClassB() {
     ❹    super();
     ⓫    System.out.print('D');
     }
Ⓕ  ❽ void prn() {
     ❾    System.out.print('E');
     }
```

```
Ⓖ  ⓭ void prn(int x) {
   ⓮    System.out.print(x);
      }
   }

   public class Test {
      public static void main(String[] args) {
   ❶    int x = 7;
   ❷    ClassB cal = new ClassB();
   ⓬    cal.prn(x);
   ⓯ }
   }
```

코드 해설

Ⓐ class ClassA {
ClassA 클래스를 정의한다. Ⓑ~Ⓒ가 클래스의 범위이다.

Ⓑ ClassA( ) {
ClassA 클래스에 속한 ClassA( ) 메소드를 정의한다. ClassA( ) 메소드는 클래스와 이름이 동일하다. 이와 같이 클래스와 이름이 동일한 메소드는 해당 클래스의 객체 변수 생성 시 자동으로 실행되는데, 이러한 메소드를 생성자(Constructor)라고 한다.

Ⓒ void prn( ) {
반환값 없는 메소드 prn( )을 정의한다.

Ⓓ class ClassB extends ClassA {
ClassB를 클래스 정의하고 부모 클래스로 ClassA를 지정하면서 ClassA에 속한 변수와 메소드를 상속받는다. ClassB 클래스는 ClassA의 변수와 메소드를 사용할 수 있게 된다. Ⓔ~Ⓖ가 클래스의 범위이다.
• extends [클래스명] : 클래스 정의 시 상속받을 클래스를 추가하는 예약어

Ⓔ ClassB( ) {
ClassB 클래스에 속한 ClassB( ) 메소드를 정의한다. ClassB( ) 메소드도 ClassB 클래스와 이름이 동일하므로 객체 변수 생성 시 자동으로 실행되는 생성자이다.

Ⓕ void prn( ) {
반환값 없는 메소드 prn( )을 정의한다. Ⓓ에서 ClassB 클래스는 ClassA 클래스의 메소드를 사용할 수 있다고 했으므로 ClassB 클래스에는 이름이 같은 메소드(Ⓒ, Ⓕ) prn( )이 두 개 존재하게 된다. 이와 같이 상속으로 인해 동일한 이름의 메소드가 여러 개인 경우, 부모 클래스에서 정의된 prn( ) 메소드(Ⓒ)는 자식 클래스의 prn( ) 메소드(Ⓕ)에 의해 재정의되어 자식 클래스의 prn 메소드(Ⓕ)만 사용되는데, 이를 메소드 오버라이딩 또는 메소드 재정의라고 한다.

Ⓖ void prn(int x) {
반환값 없는 메소드 prn(int x)를 정의한다. 메소드의 이름이 Ⓒ, Ⓕ와 같지만 '인수를 받는 자료형과 개수'가 다르므로 서로 다른 메소드이다. 즉 prn( )과 prn(int x)는 다른 메소드라는 것이다. 이렇게 이름은 같지만 인수를 받는 자료형과 개수를 달리하여 여러 기능을 정의하는 것을 오버로딩(Overloading)이라고 한다.

모든 Java 프로그램의 실행은 반드시 main( ) 메소드에서 시작한다.
❶ 정수형 변수 x를 선언하고 7로 초기화한다.
❷ ClassB cal = new ClassB( );

ClassB 클래스의 객체 변수 cal을 선언하고 ClassB 클래스의 생성자를 호출한다. ClassB 클래스에는 클래스명과 동일한 생성자가 정의되어 있으므로 생성자를 실행하기 위해 ❸번으로 이동한다.

- ClassB : 클래스의 이름이다. 앞에서 정의한 클래스의 이름을 그대로 적어준다.
- cal : 객체 변수의 이름이다. 사용자가 원하는 이름을 적으면 된다.
- new : 객체 생성 예약어다. 그대로 적어준다.
- ClassB( ) : 클래스와 이름이 동일한 메소드로, 생성자이다.

❸ ClassB 클래스의 생성자인 ClassB( ) 메소드의 시작점이다. 지금처럼 클래스 안에 생성자를 정의하는 문장이 있을 경우 객체가 생성될 때 자동으로 호출되어 실행된다.

❹ 부모 클래스의 생성자인 ClassA( ) 메소드를 호출한다. ❺번으로 이동한다.

- super( ) : 부모 클래스의 생성자를 호출한다.

❺ ClassA 클래스의 생성자 ClassA( ) 메소드의 시작점이다.

❻ A를 출력한다.

결과 A

❼ 자신이 속한 ClassA 클래스의 prn( ) 메소드를 호출한다. ClassA 클래스의 prn( ) 메소드는 ClassB 클래스의 prn( ) 메소드에 의해 재정의되었으므로 ❽번으로 이동한다.

❽ 반환값 없는 prn( ) 메소드의 시작점이다.

❾ E를 출력한 후 메소드가 종료되면 호출했던 ❼번의 다음 줄인 ❿번으로 이동한다.

결과 AE

❿ ClassA( ) 메소드가 종료되었으므로 호출했던 ❹번의 다음 줄인 ⓫번으로 이동한다.

⓫ D를 출력한 후 ClassB( ) 메소드가 종료되면 호출했던 ❷번의 다음 줄인 ⓬번으로 이동한다.

결과 AED

⓬ x의 값 7을 인수로 cal의 prn(int x) 메소드를 호출한다. ⓭번으로 이동한다.

⓭ 반환값 없는 prn(int x) 메소드의 시작점이다. ⓬번에서 전달한 7을 x가 받는다.

⓮ x의 값 7을 출력한 후 prn(int x) 메소드가 종료되면 호출했던 ⓬번의 다음 줄인 ⓯번으로 이동하여 프로그램을 종료한다.

결과 AED7

340396 20.10

222 생성자 Ⓐ

- 생성자(Constructor)는 객체 변수 생성에 사용되는 메소드로, **객체 변수를 생성하면서 초기화를 수행**한다.
- 클래스 안에 생성자를 정의하는 문장이 있다면 문장에 적힌 대로 객체 변수를 초기화하면서 생성하지만, 없으면 그냥 객체 변수만 생성하고 제어가 다음 줄로 이동한다.

340397

25.7, 25.4, 23.10, 23.4, 20.11, 20.10, 20.7

223 Java의 활용 Ⓐ

예제 다음 Java 프로그램의 실행 결과를 확인하시오.

```
Ⓐ abstract class Animal {
Ⓑ    String a = " is animal";
Ⓒ    abstract void look();
Ⓓ ❽ void show() {
  ❾    System.out.println("Zoo");
     }
  }
Ⓔ class Chicken extends Animal {
Ⓕ ❷ Chicken() {
  ❸    look();
  ❻ }
Ⓖ ❹ void look() {
  ❺    System.out.println("Chicken" + a);
     }
Ⓗ    void display() {
        System.out.println("two wings");
     }
  }
  public class Test {
     public static void main(String[] args) {
  ❶    Animal a = new Chicken();
  ❼    a.show();
  ❿ }
  }
```

코드 해설

Ⓐ abstract class Animal {

추상 클래스 Animal을 정의한다.

- abstract **[클래스 정의부]** : abstract는 추상 클래스를 정의하는 명령어로, 추상 클래스 정의 시 꼭 써야하는 예약어이다.

※ 추상 클래스는 내부에 실행 코드가 없는 추상 메소드를 포함하기 때문에 객체 변수의 생성자로 사용할 수 없습니다.

예 Animal a = new Animal( ); ← 오류 발생

Ⓑ 문자열 변수 a를 선언하고 " is animal"로 초기화한다.

Ⓒ abstract void look( );

추상 메소드 look( )을 정의한다.

- abstract [메소드 정의부] : abstract는 추상 메소드를 정의하는 명령어로, 추상 메소드 정의 시 꼭 써야하는 예약어이다.

※ 추상 메소드는 선언만 있고 내부에 실행 코드가 없는 메소드로, 이후 상속 관계가 설정된 자식 클래스에서 재정의한 후 사용합니다.

Ⓓ 반환값 없는 메소드 show( )를 정의한다.

Ⓔ 클래스 Chicken을 정의하고 부모 클래스로 Animal을 지정하면서 Animal에 속한 변수와 메소드를 상속받는다.

Ⓕ Chicken 클래스에 속한 Chicken( ) 메소드를 정의한다. Chicken 클래스와 이름이 동일하므로 객체 변수 생성 시 자동으로 실행되는 생성자이다.

Ⓖ 반환값 없는 메소드 look( )을 정의한다. look( ) 메소드는 Ⓒ에서 정의된 추상 메소드(Ⓒ)를 재정의하는 것이다.

Ⓗ 반환값 없는 메소드 display( )를 정의한다.

모든 Java 프로그램은 반드시 main( ) 메소드에서 시작한다.

❶ Animal a = new Chicken( );

Chicken 클래스의 생성자를 이용하여 Animal 클래스의 객체 변수 a를 선언한다.

- [부모클래스명] [객체변수명] = new [자식클래스생성자( )] : 부모 클래스의 객체 변수를 선언하면서 자식 클래스의 생성자를 사용하면 형 변환이 발생한다.
- 이렇게 형 변환이 발생했을 때 부모 클래스와 자식 클래스에 동일한 속성이나 메소드가 있으면 자식 클래스의 속성이나 메소드로 재정의된다.

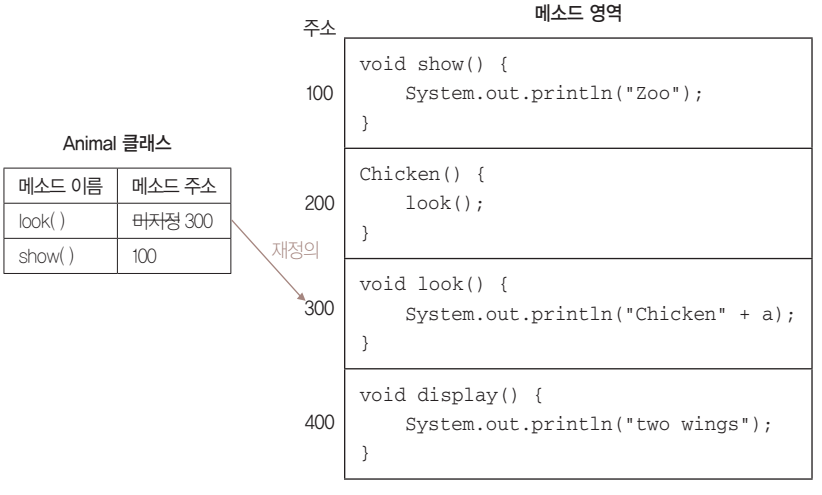
※ 객체 변수 a는 Animal 클래스의 객체 변수입니다. Animal 클래스는 실행 코드가 없는 추상 메소드 look( )으로 인해 객체 변수의 생성이 불가능해야 하지만, 형 변환으로 인해 look( ) 메소드가 Chicken 클래스에서 재정의되었으므로 객체 변수의 생성이 가능해진 것입니다.

❷ Chicken 클래스의 생성자인 Chicken( ) 메소드의 시작점이다. 클래스 안에 생성자를 정의하는 문장이 있으므로 Chicken 클래스의 객체가 생성될 때 자동으로 호출되어 실행된다.

❸ look( ) 메소드를 호출한다. ❹번으로 이동한다.

look( ) 메소드는 a 객체의 자료형이 Animal이므로 Animal 클래스의 look( )이라고 생각할 수 있지만 ❶번에서 클래스 형 변환이 발생하였고, look( ) 메소드가 자식 클래스에서 재정의되었으므로 Chicken 클래스의 look( ) 메소드가 수행된다.

※ Animal 클래스의 look( ) 메소드는 실행 코드가 정의되지 않은 추상 메소드로, 코드를 실행할 때 실행 코드가 저장된 메모리 위치를 참조할 수 없는데, 재정의가 이루어지면 다음 그림과 같이 Chicken 클래스의 look( ) 메소드를 가리키게 됩니다.



❹ 반환값 없는 look( ) 메소드의 시작점이다.

❺ Chicken과 a의 값 is animal을 출력한 후 커서를 다음 줄의 처음으로 옮긴다. 이어서 메소드가 종료되면 호출했던 ❸번의 다음 줄인 ❻번으로 이동한다.

결과 Chicken is animal

※ Ⓔ에서 Chicken 클래스가 Animal 클래스의 변수와 메소드를 상속받았으므로 Animal 클래스의 a 변수의 값을 사용할 수 있습니다.

❻ Chicken( ) 메소드가 종료되었으므로 호출했던 ❶번의 다음 줄인 ❼번으로 이동한다.

❼ a의 show( ) 메소드를 호출한다. a는 Animal 클래스의 객체 변수이므로 Animal의 show( ) 메소드인 ❽이 호출된다.

❽ 반환값 없는 show( ) 메소드의 시작점이다.

❾ Zoo를 출력한 후 메소드가 종료되면 호출했던 ❼번의 다음 줄인 ❿번으로 이동하여 프로그램을 종료한다.

결과
```
Chicken is animal
Zoo
```

340554

25.7, 25.4, 24.10, 24.7, 24.4, 23.10, 23.7, 21.10, 21.4, 20.11, 필기 25.5, 25.2, 24.7, 24.5, 24.2, 21.8

224 Python의 활용 1 Ⓐ

예제 다음 Python으로 구현된 프로그램에 "xyz321–opq654"를 입력했을 때 그 실행 결과를 확인하시오.

```
❶ x, y = input('입력 :').split('-')
❷ a = [ 'abc123', 'def456', 'ghi789' ]
❸ a.append(x)
❹ a.append(y)
❺ a.remove('def456')
❻ print(a[1][-3:], a[2][:-3], sep = ',')
❼ for i in range(3, 6):
❽    print(i, end = ' ')
```

코드 해설

❶ x, y = input('입력 :'). split('–')

화면에 **입력** : 이 출력되고 그 뒤에서 커서가 깜박거리며 입력을 기다린다. 키보드로 값을 입력하고 엔터를 누르면 입력된 값이 split로 지정된 분리문자인 '–'를 기준으로 **xyz321**은 변수 x에 저장되고, **opq654**는 변수 y에 저장된다.

- x, y : 입력되는 값이 저장될 변수이다.
- input( ).split('–') : 분리문자 '–'를 기준으로 입력된 값을 변수 x와 y에 나눠서 저장한다.
- '**입력** :' : 입력 시 화면에 출력되는 문자로, 생략이 가능하다.

❷ a = ['abc123', 'def456', 'ghi789']

리스트 a를 선언하면서 초기값을 지정한다. 초기값으로 지정된 수만큼 리스트의 요소가 만들어진다.

	[0]	[1]	[2]
a	'abc123'	'def456'	'ghi789'

❸ a.append(x)

a 리스트의 마지막에 x의 값 "xyz321"을 추가한다.

	[0]	[1]	[2]	[3]
a	'abc123'	'def456'	'ghi789'	'xyz321'

❹ a.append(y)

a 리스트의 마지막에 y의 값 'opq654'를 추가한다.

	[0]	[1]	[2]	[3]	[4]
a	'abc123'	'def456'	'ghi789'	'xyz321'	'opq654'

❺ a.remove('def456')

a 리스트에서 "def456"을 찾아 삭제하고, 이후의 요소들을 하나씩 앞으로 이동시킨다.

	[0]	[1]	[2]	[3]
a	'abc123'	'ghi789'	'xyz321'	'opq654'

❻ print(a[1][–3:], a[2][:–3], sep = ',')

a[1]과 a[2]의 요소들을 슬라이스(slice)하고, 쉼표(,)로 구분하여 출력한다. 이어서 커서를 다음 줄의 처음으로 옮긴다.

- a[1][–3:] : a[1]에 저장된 문자열 "ghi789"의 –3 위치부터 마지막 요소까지 추출한다.

a[1]	g	h	i	7	8	9
	–6	–5	–4	–3	–2	–1

※ 최종위치가 생략된 경우 초기위치로 지정된 위치부터 마지막 요소까지의 모든 값이 반환됩니다.

- a[2][:–3] : a[2]에 저장된 문자열 "xyz321"의 맨 처음 요소부터 –4까지의 요소들을 추출한다.

	0	1	2	3	4	5
a[2]	x	y	z	3	2	1
	–6	–5	–4	–3	–2	–1

※ 초기위치가 생략된 경우 요소의 맨 처음 요소부터 최종위치(–3)에서 1을 뺀 위치(–4)까지의 모든 값이 반환됩니다.

- sep = ',' : 분리문자로 쉼표(,)를 지정한다. 출력할 값들을 쉼표로 구분하여 출력한다.

결과
```
789,xyz
```

❼ for i in range(3, 6):

3에서 5(6–1)까지 순서대로 i에 저장하며 ❽번을 반복 수행한다.

- i : 반복 변수로 range에서 생성되는 값을 순서대로 저장한다.
- in : for와 함께 사용되는 예약어로 그대로 입력한다.
- range(3, 6) : 3부터 5(6–1)까지의 연속적인 숫자를 생성한다.
- **콜론(:)** : for문이 반복할 코드가 다음 줄부터 시작한다는 의미이다. 반드시 입력해야 한다.

❽ print(i, end = ' ')

i의 값을 출력하고 종료문자로 공백 한 칸이 출력된다. ❼번에서 i는 3에서 5까지 순서대로 저장한다고 하였으므로, 3 4 5가 출력된다.

결과
```
789,xyz
3 4 5
```

440328

25.11, 25.7, 25.4, 23.4, 22.7, 20.7, 필기 22.3

225 Python의 활용 2 Ⓐ

예제 다음 Python 프로그램의 실행 결과를 확인하시오.

```
❶ a = {'apple', 'lemon', 'banana'}
❷ a.update( {'kiwi', 'banana'} )
❸ a.remove('lemon')
❹ a.add('apple')
❺ for i in a:
❻       print("과일명 : %s" % i)
```

코드 해설

세트는 수학에서 배우는 집합(set)과 동일한 역할을 하는 Python의 자료형으로, 중괄호{ }를 이용하여 리스트와 같이 다양한 요소들을 저장할 수 있다. 세트는 순서가 정해져 있지 않으며(unordered), 중복된 요소는 저장되지 않는다는 특징이 있다.

❶ a = {'apple', 'lemon', 'banana'}

세트 a를 선언하면서 초기값을 지정한다. 초기값으로 지정된 수만큼 세트의 요소가 만들어진다.

a	'apple'	'lemon'	'banana'

❷ a.update({'kiwi', 'banana'})

a 세트에 새로운 세트를 추가하여 확장한다. 새로운 세트 {'kiwi', 'banana'}가 추가되어야 하지만 'banana'는 이미 a 세트에 있으므로 'kiwi'만 추가된다.

a	'apple'	'lemon'	'banana'	'kiwi'

❸ a.remove('lemon')

a 세트에서 'lemon'을 제거한다.

a	'apple'	'banana'	'kiwi'

❹ a.add('apple')

a 세트에 'apple'을 추가한다. a 세트에는 이미 'apple'이 존재하므로 a 세트의 요소는 변하지 않는다.

❺ for i in a:

a 세트의 각 요소의 값을 차례로 i에 저장하면서 a 세트의 요소 수만큼 ❻번을 반복 수행한다.

- i : a 세트의 각 요소가 일시적으로 저장될 변수이다.
- a : 세트의 이름이다.

❻ **과일명 :** 과 i의 값을 출력한 후 커서를 다음 줄의 처음으로 옮긴다. for문에 의한 변수의 변화는 다음과 같다.

- **❺~❻번의 첫 번째 수행** : a 세트의 첫 번째 값이 i를 거쳐 출력된다.

결과
```
과일명 : apple
```

- **❺~❻번의 두 번째 수행** : a 세트의 두 번째 값이 i를 거쳐 출력된다.

결과
```
과일명 : apple
과일명 : banana
```

- **❺~❻번의 세 번째 수행** : a 세트의 세 번째 값이 i를 거쳐 출력된다.

결과
```
과일명 : apple
과일명 : banana
과일명 : kiwi
```

440329 22.10

226 람다 식 A

- 어떤 문제를 해결하기 위한 과정을 수학식으로 표현한 것을 람다 식(Lambda Expression)이라고 한다.
- 프로그래밍 언어에서 람다 식은 수학적 연산을 수행하는 함수나 메소드를 간소화할 때 사용한다.
- **형식**

```
lambda 변수명 : 수학식
```

  - 변수명 : 인수로 전달받은 값을 저장할 변수의 이름을 지정
  - 수학식 : 수행할 연산을 하나의 식으로 풀어 입력
- 예를 들어 다음과 같은 코드가 있다고 가정한다.

```
def func(x):
    return x * x - 3
print(func(10))
```

결과 97

- 위의 func( ) 함수는 다음과 같이 람다 식으로 수정하여 사용할 수 있다.

```
func = lambda x : x * x - 3
print(func(10))
```

결과 97

340400 25.4, 필기 21.5

227 Python의 클래스 A

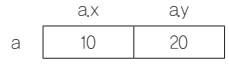
예제 다음은 두 수를 교환하는 프로그램을 Python으로 구현한 것이다.

```
class Cls:          Cls 클래스 정의부의 시작점이다. 여기서부터 ❼번까지가 클래스 정의부에 해당한다.
    x, y = 10, 20       Cls 클래스의 변수(속성) x와 y를 선언하고, 각각 10과 20으로 초기화한다.
❹   def chg(self):
❺       temp = self.x
❻       self.x = self.y
❼       self.y = temp
❶ a = Cls()
❷ print(a.x, a.y)
❸ a.chg()
❽ print(a.x, a.y)
```

코드 해설

❶ Cls 클래스의 객체 a를 생성한다. 객체 a는 Cls의 속성 x, y와 메소드 chg( )를 갖는다.

- a : 사용자 정의 변수다. 사용자가 임의로 지정한다.
- Cls( ) : 클래스의 이름이다. 괄호( )를 붙여 그대로 적는다.



❷ a 객체의 속성 x와 y를 출력한다.

- 객체와 속성은 .(마침표)로 연결한다.

결과
```
10 20
```

❸ a 객체의 메소드 chg를 호출한다. ❹번으로 이동한다.

- 객체와 메소드는 .(마침표)로 연결한 후 괄호( )를 붙여 적는다.

❹ a 객체의 메소드 chg의 시작점이다. 별도로 사용되는 인수가 없으므로 괄호( )에는 self만 적는다.

❺ a 객체의 속성 x의 값을 temp에 저장한다.

- self : 메소드 안에서 사용되는 self는 자신이 속한 클래스를 의미한다.
- self.x : a.x와 동일하다.

❻ a 객체의 속성 y의 값을 a 객체의 속성 x에 저장한다.

❼ temp의 값을 a 객체의 속성 y에 저장한다. 메소드 chg가 종료되었으므로 메소드를 호출한 다음 문장인 ❽번으로 제어를 옮긴다.

❽ a 객체의 속성 x와 y를 출력한다.

결과
```
10 20
20 10
```

340401

24.10, 24.7, 22.5, 필기 23.2, 21.8

228 Python의 클래스 없는 메소드 사용 A

예제 다음 프로그램의 실행 결과를 확인하시오.

```
❸ def calc(x, y):
❹       x *= 3
❺       y /= 3
❻       print(x, y)
❼       return x

❶ a, b = 3, 12
❷ a = calc(a, b)
❽ print(a, b)
```

코드 해설

❶ 변수 a와 b에 3과 12를 저장한다.
❷ a와 b의 값, 즉 3과 12를 인수로 하여 calc 메소드를 호출한 결과를 a에 저장한다. ❸번으로 이동한다.
❸ 메소드 calc의 시작점이다. ❷번에서 calc(a, b)라고 했으므로 x는 a의 값 3을 받고, y는 b의 값 12를 받는다.
❹ x = x * 3이므로 x는 9가 된다.
❺ y = y / 3이므로 y는 4가 된다.
❻ 결과
```
9 4.0
```
❼ x의 값을 반환한다. x의 값 9를 ❷번의 a에 저장한 후 제어를 ❽번으로 옮긴다.
❽ 결과
```
9 4.0
9 12
```

340402

25.4, 21.7

229 Python - Range A

- Range는 연속된 숫자를 생성하는 것으로, 리스트나 반복문에서 많이 사용된다.
- **형식**

range(최종값)	0에서 '최종값'-1까지 연속된 숫자를 생성한다.
range(초기값, 최종값)	'초기값'에서 '최종값'-1까지 연속된 숫자를 생성한다.
range(초기값, 최종값, 증가값)	• '초기값'에서 '최종값'-1까지 '증가값'만큼 증가하면서 숫자를 생성한다. • '증가값'이 음수인 경우 '초기값'에서 '최종값'+1까지 '증가값'만큼 감소하면서 숫자를 생성한다.

340403

필기 20.9, 20.8

230 Python – 슬라이스 Ⓒ

- 슬라이스(slice)는 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.

340514

25.7, 22.5

231 Python - 주요 메소드 Ⓐ

형식	내용
22.5 pop(위치)	객체의 '위치'에 있는 값을 출력하고 해당 요소를 삭제함 예 [10, 11, 12].pop(1) → 11 출력 → [10, 12]
25.7 len( )	객체 요소의 개수를 반환함 예 len({2, 7, 6}) → 3
22.5 extend(리스트)	리스트의 끝에 새로운 '리스트'를 추가하여 확장함 예 ['a', 'b'].extend(['c', 'd']) → ['a', 'b', 'c', 'd']
22.5 reverse( )	리스트의 순서를 역순으로 뒤집음 예 [1, 2, 3].reverse( ) → [3, 2, 1]
25.7 set( )	• 반복 가능한 개체를 세트 자료형으로 변환하거나 빈 세트를 생성함 • 중복된 값은 제거됨 예 a = [1, 2, 3, 4, 2, 1] → {1, 2, 3, 4} b = set(a)
25.7 add(값)	세트에 '값'을 추가함 예 a = {2, 7, 6} → {2, 7, 6, 9} a.add(9)
25.7 values( )	딕셔너리의 모든 값을 반환함 예 a = {'name': 'kim','age': 30} → ['kim', 30] a.values( )

340515 필기 25.8, 25.5, 25.2, 23.5, 23.2, 21.5

232 C언어의 대표적인 표준 라이브러리 Ⓑ

필기 23.5, 23.2

stdio.h

- 데이터의 입 · 출력에 사용되는 기능들을 제공한다.
- **주요 함수** : printf, scanf, fprintf, fscanf, fclose, fopen 등

필기 25.2, 23.2

string.h

- 문자열 처리에 사용되는 기능들을 제공한다.
- **주요 함수** : strlen, strcpy, strcmp 등

필기 25.5, 21.5

stdlib.h

- 자료형 변환, 난수 발생, 메모리 할당에 사용되는 기능들을 제공한다.
- **주요 함수** : atoi, atof, srand, rand, malloc, free 등

25.7, 25.4, 필기 25.8, 24.5, 22.7

malloc( ) 함수

- 바이트 단위로 메모리 공간을 동적으로 할당한다.
- 메모리 할당이 불가능할 경우 NULL이 반환된다.

25.7, 25.4, 필기 24.5, 22.7

free( ) 함수

malloc( ) 함수에 의해 동적으로 할당된 메모리를 해제한다.

401802 25.7, 25.4, 24.10, 22.5, 필기 25.5, 24.2, 23.7, 23.2

233 JAVA의 예외 처리 Ⓐ

프로그램의 정상적인 실행을 방해하는 조건이나 상태를 예외(Exception)라고 하며, 이러한 **예외가 발생했을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴**을 수행하도록 하는 것을 예외 처리(Exception Handling)라고 한다.

- JAVA는 잘못된 동작이나 결과에 영향을 줄 수 있는 예외를 객체로 취급하며, 예외와 관련된 클래스를 java.lang 패키지에서 제공한다.
- JAVA에서는 try ~ catch 문을 이용해 예외를 처리한다.
- try 블록 코드를 수행하다 예외가 발생하면 예외를 처리하는 catch 블록으로 이동하여 예외 처리 코드를 수행하므로 예외가 발생한 이후의 코드는 실행되지 않는다.
- **기본 형식**

```
try {
    예외가 발생할 가능성이 있는 코드;
}
catch ( 예외객체1 매개변수 ) {
    예외객체1에 해당하는 예외 발생 시 처리 코드;
}
catch ( 예외객체2 매개변수 ) {
    예외객체2에 해당하는 예외 발생 시 처리 코드;
}
catch ( 예외객체n 매개변수) {
    예외객체n에 해당하는 예외 발생 시 처리 코드;
}
catch (Exception 매개변수) {
    예외객체1~n에 해당하지 않는 예외 발생 시 처리 코드;
}
finally {
    예외의 발생 여부와 관계없이 무조건 처리되는 코드;
}
```

※ 일반적으로 예외가 발생한 경우에는 'try문 → 해당 예외 catch문 → finally문' 순으로 진행되며, 예외가 발생하지 않은 경우에는 'try문 → finally문' 순으로 진행됩니다. finally문은 예외 발생과 관계없이 무조건 수행되는 블록으로 생략이 가능합니다.

401803

25.4, 24.10, 필기 23.2, 22.3

234 JAVA의 주요 예외 객체 Ⓐ

예외 객체	발생 원인
ClassNotFoundException	클래스를 찾지 못한 경우
NoSuchMethodException	메소드를 찾지 못한 경우
FileNotFoundException	파일을 찾지 못한 경우
InterruptedIOException	입 · 출력 처리가 중단된 경우
ArithmeticException	0으로 나누는 등의 산술 연산에 대한 예외가 발생한 경우
IllegalArgumentException	잘못된 인자를 전달한 경우
NumberFormatException	숫자 형식으로 변환할 수 없는 문자열을 숫자 형식으로 변환한 경우
ArrayIndexOutOfBoundsException	배열의 범위를 벗어난 접근을 시도한 경우
NegativeArraySizeException	0보다 작은 값으로 배열의 크기를 지정한 경우
NullPointerException	존재하지 않는 객체를 참조한 경우

11장 응용 SW 기초 기술 활용

340411 필기 20.8

235 운영체제 C

- 운영체제(OS; Operating System)는 **컴퓨터 시스템의 자원들을 효율적으로 관리**하며, 사용자가 컴퓨터를 **편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임**이다.
- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로 동작하는 시스템 소프트웨어의 일종이다.
- 프로세서, 기억장치, 입 · 출력장치, 파일 및 정보 등의 자원을 관리한다.

340413 20.11

236 UNIX A

- UNIX는 1960년대 **AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제**이다.
- 시분할 시스템(Time Sharing System)을 위해 설계된 대화식 운영체제이다.
- 대부분 C 언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.
- 트리(Tree) 구조의 파일 시스템을 갖는다.

340414 필기 20.9, 20.6

237 UNIX 시스템의 구성 C

필기 20.9

커널(Kernel)

- UNIX의 가장 핵심적인 부분으로 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 여러가지 기능을 수행한다.

필기 20.6

쉘(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다.
- 시스템과 사용자 간의 인터페이스를 담당한다.

340416 20.7

238 Android A

- Android(안드로이드)는 **구글(Google) 사에서 개발한 리눅스 커널 기반의 개방형 모바일 운영체제**이다.
- 모든 코드가 공개된 개방형 소프트웨어이다.
- 자바와 코틀린으로 애플리케이션을 작성한다.

340418 필기 25.8, 25.5, 25.2, 24.7, 24.5, 23.7, 23.2, 22.3, 21.3, 20.8

239 기억장치 관리 - 배치 전략 B

- 배치(Placement) 전략은 새로 반입되는 **프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략**이다.
- **최초 적합(First Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
- **최적 적합(Best Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법
- **최악 적합(Worst Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

예제 기억장치 상태가 다음 표와 같다. 기억장치 관리 전략으로 First Fit, Best Fit, Worst Fit 방법을 사용하려 할 때, 각 방법에 대하여 10K의 프로그램이 할당받게 되는 영역의 번호는?

영역 번호	영역 크기	상태
1	5K	공백
2	14K	공백
3	10K	사용 중
4	12K	공백
5	16K	공백

① 먼저 10K가 적재될 수 있는지 각 영역의 크기를 확인한다.

② First Fit : 빈 영역 중에서 10K의 프로그램이 들어갈 수 있는 첫 번째 영역은 2번이다.
③ Best Fit : 빈 영역 중에서 10K 프로그램이 들어가고 단편화를 가장 작게 남기는 영역은 4번이다.
④ Worst Fit : 빈 영역 중에서 10K 프로그램이 들어가고 단편화를 가장 많이 남기는 영역은 5번이다.

340423 필기 23.2, 21.3

240 가상기억장치 구현 기법 C

필기 21.3

페이징(Paging) 기법

- 가상기억장치에 보관되어 있는 **프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후** 나눠진 프로그램을 동일하게 나눠진 **주기억장치의 영역에 적재시켜 실행하는 기법**이다.
- 프로그램을 일정한 크기로 나눈 단위를 페이지(Page)라고 하고, 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임(Page Frame)이라고 한다.

필기 21.3

세그먼테이션(Segmentation) 기법

- 가상기억장치에 보관되어 있는 **프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법**이다.
- 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트(Segment)라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다.

340424 필기 23.2, 21.8

241 페이지 교체 알고리즘 C

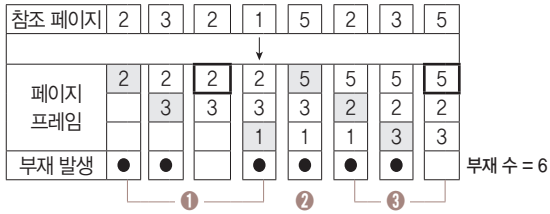
- **페이지 부재(Page Fault)가 발생하면** 가상기억장치에서 필요한 페이지를 찾아 주기억장치에 적재해야 하는데, 이때 주기억장치의 모든 페이지 프레임이 사용중이면 **어떤 페이지 프레임을 선택하여 교체할 것인지를 결정하는 기법**이 페이지 교체 알고리즘이다.
- **종류** : OPT, FIFO, LRU, LFU, NUR, SCR 등

340426 필기 25.8, 25.2, 24.7, 24.5, 24.2, 23.7, 23.2, 22.7, 22.3, 20.9, 20.6

242 FIFO B

FIFO(First In First Out)는 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 **가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법**이다.

예제 다음의 참조 페이지를 세 개의 페이지 프레임을 가진 기억장치에서 FIFO 알고리즘을 사용하여 교체했을 때 페이지 부재의 수는? (단, 초기 페이지 프레임은 모두 비어 있는 상태이다.)



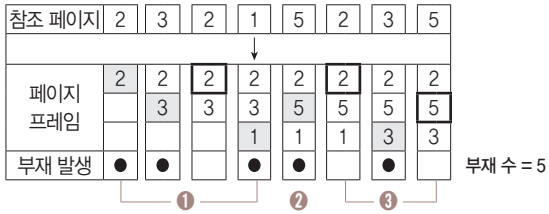
❶ 참조 페이지를 각 페이지 프레임에 차례로 적재시키되 이미 적재된 페이지는 해당 위치의 페이지 프레임을 사용한다.
❷ 사용할 페이지 프레임이 없을 경우 가장 먼저 들어와서 오래 있었던 페이지 2를 제거한 후 5를 적재한다.
❸ 그 다음에 적재된 페이지 3을 제거한 후 2를 적재하며, 같은 방법으로 나머지 참조 페이지를 수행한다.

440354 24.10, 24.4, 필기 22.4

243 LRU A

LRU(Least Recently Used)는 **최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법**이다.

예제 다음의 참조 페이지를 세 개의 페이지 프레임을 가진 기억장치에서 LRU 알고리즘을 사용하여 교체했을 때 페이지 부재의 수는? (단, 초기 페이지 프레임은 모두 비어 있는 상태이다.)



❶ 참조 페이지를 각 페이지 프레임에 차례로 적재시키되 이미 적재된 페이지는 해당 위치의 페이지 프레임을 사용한다.
❷ 사용할 페이지 프레임이 없을 경우 현재 시점에서 가장 오랫동안 사용되지 않은 페이지 3을 제거한 후 5를 적재한다.
❸ 같은 방법으로 나머지 참조 페이지를 수행한다.

340555 24.4

244 LFU A

- LFU(Least Frequently Used)는 **사용 빈도가 가장 적은 페이지를 교체하는 기법**이다.
- 활발하게 사용되는 페이지는 사용 횟수가 많아 교체되지 않고 사용된다.

예제 다음의 참조 페이지를 네 개의 페이지 프레임을 가진 기억장치에서 LFU 알고리즘을 사용하여 교체했을 때 페이지 부재의 수는? (단, 초기 페이지 프레임은 모두 비어 있는 상태이다.)

참조 페이지	2	3	1	3	1	2	4	5
페이지 프레임	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3
			1	1	1	1	1	1
							4	5
부재 발생	●	●	●				●	●

부재 수 = 5

❶ 참조 페이지를 각 페이지 프레임에 차례로 적재시키되 이미 적재된 페이지는 해당 위치의 페이지 프레임을 사용한다.
❷ 사용할 페이지 프레임이 없을 경우 사용 빈도가 가장 적은 페이지 4를 제거한 후 5를 적재한다.

340431 필기 21.5

245 Locality C

- Locality(국부성, 지역성, 구역성, 국소성)는 **프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질**이 있다는 이론이다.
- 스래싱을 방지하기 위한 워킹 셋 이론의 기반이 되었다.
- **시간 구역성(Temporal Locality)** : 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상
- **공간 구역성(Spatial Locality)** : 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상

340432 필기 21.3

246 워킹 셋 C

- 워킹 셋(Working Set)은 **프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합**이다.
- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정된다.

340433 필기 25.2, 21.5

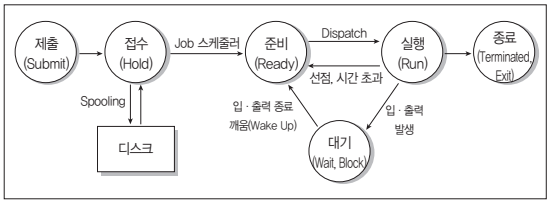
247 스래싱 C

- 스래싱(Thrashing)은 **프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상**이다.
- 다중 프로그래밍 시스템이나 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정 중에 자주 페이지 부재(Page Fault)가 발생함으로써 나타나며, 전체 시스템의 성능이 저하된다.

340436 20.11, 필기 23.5, 20.6

248 프로세스 상태 전이 A

프로세스 상태 전이는 프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것을 의미한다.



340437

필기 25.8, 21.8

249 프로세스 상태 전이 관련 용어 Ⓒ

필기 21.8

Dispatch(디스패치)

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(스풀링)

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.

340438

필기 25.8, 20.6

250 스레드 Ⓒ

- 스레드(Thread)는 **시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위** 또는 프로세스 내에서의 작업 단위로 사용된다.
- 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.

340443

25.7, 22.10, 필기 20.9

251 SJF Ⓐ

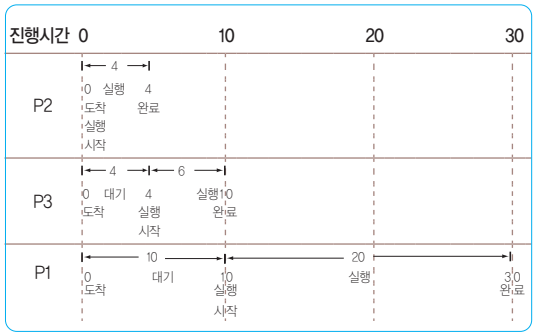
- SJF(Shortest Job First, 단기 작업 우선)는 **준비상태 큐에서** 기다리고 있는 프로세스들 중에서 **실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법**이다.
- 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘이다.

예제 다음과 같은 프로세스들이 차례로 준비상태 큐에 들어왔다고 가정할 때, SJF 기법을 이용하여 평균 실행 시간, 평균 대기 시간, 평균 반환 시간을 구하시오(제출 시간이 없을 경우).

프로세스 번호	P1	P2	P3
실행 시간	20	4	6

① 아래와 같이 실행 시간이 짧은 프로세스를 먼저 처리하도록 이동시킨 후 각 프로세스의 대기 시간과 반환 시간을 구한다.

② 실행시간, 대기 시간, 반환 시간, 각 시간의 평균은 FCFS의 예제와 동일한 방법으로 구한다.



- 평균 실행 시간 : (4+6+20)/3 = 10
- 평균 대기 기간 : (0+4+10)/3 = 4.6
- 평균 반환 시간 : (4+10+30)/3 = 14.6

340444

20.5, 필기 25.8, 25.5, 24.7, 23.5, 23.2, 20.8, 20.6

252 HRN Ⓐ

- HRN(Highest Response-ratio Next)은 **대기 시간과 서비스(실행) 시간을 이용하는 기법**이다.
- 우선순위를 계산하여 그 숫자가 가장 높은 것부터 낮은 순으로 우선순위가 부여된다.
- 우선순위 계산식

$$\text{우선순위 계산식} = \frac{\text{대기 시간} + \text{서비스 시간}}{\text{서비스 시간}}$$

예제 다음과 같은 프로세스가 HRN 기법으로 스케줄링 될 때 우선순위를 계산하시오.

프로세스 번호	P1	P2	P3
실행 시간	20	4	6
대기 시간	10	20	10
우선순위 계산	(20+10)/20=1.5	(4+20)/4=6	(6+10)/6=2.6
우선순위	P2 → P3 → P1		

440370 25.7, 22.10

253 RR Ⓐ

- RR(Round Robin)은 각 프로세스를 **시간 할당량(Time Slice, Quantum) 동안만 실행한 후** 실행이 완료되지 않으면 **다음 프로세스에게 CPU를 넘겨주는 기법**이다.
- 시분할 시스템(Time Sharing System)을 위해 고안된 방식으로, 할당되는 시간의 크기가 작으면 작은 프로세스들에게 유리하다.
- 할당되는 시간이 클 경우 FCFS 기법과 같아지고, 할당되는 시간이 작을 경우 문맥 교환 및 오버헤드가 자주 발생되어 요청된 작업을 신속히 처리할 수 없다.

440371 25.7, 24.7, 22.10

254 SRT Ⓐ

- SRT(Shortest Remaining Time)는 **현재 실행중인 프로세스의 남은 시간과** 준비상태 큐에 **새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스에게** CPU를 **할당하는 기법**이다.
- 시분할 시스템에 유용하며, 준비상태 큐에 있는 각 프로세스의 실행 시간을 추적하여 보유하고 있어야 하므로 오버헤드가 증가한다.

예제 다음과 같은 프로세스들이 차례로 준비상태 큐에 들어왔다고 가정할 때, SRT 기법을 이용하여 평균 대기 시간, 평균 반환 시간을 구하시오.

프로세스 번호	A	B	C	D
도착 시간	0	1	2	3
실행 시간	8	4	9	5

① 아래와 같이 현재 실행중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스를 먼저 처리하도록 이동시킨 후 대기 시간과 반환 시간을 구한다.

② 실행이 마무리되지 못한 경우 준비상태 큐에 재배치하여 차례를 기다리므로 다음과 같이 표시할 수 있다.

진행 시간 →	0	1	❺	❿	⓱	㉖
프로세스	A	B	D	A	C	
실행 시간	0	1	5	7	9	
남은 시간	7	0	0	0	0	

※ 색 동그라미는 프로세스가 완료됨을 표시한 것이다(A → 17초, B → 5초, C → 26초, D → 10초).

③ **반환 시간** : '완료 시간 – 대기 시간'으로 구한다.

④ **대기 시간** : '완료 시간 – 도착 시간 – 실행 시간'으로 구한다.

프로세스 번호	A	B	C	D	평균
반환 시간	17	4	24	7	52/4 = 13
대기 시간	9	0	15	2	26/4 = 6.5

340445 필기 20.9

255 환경 변수 Ⓒ

- 환경 변수(Environment Variable)란 **시스템 소프트웨어의 동작에 영향을 미치는 동적인 값들의 모임**을 의미한다.
- 환경 변수는 변수명과 값으로 구성된다.
- Windows에서 set을 입력하면 모든 환경 변수와 값을 출력한다.
- UINIX나 LINUX에서는 set, env, printenv, setenv 중 하나를 입력하면 모든 환경 변수와 값을 표시한다.

340447

25.11, 23.10, 20.7, 필기 25.8, 24.7, 24.5, 23.5, 23.2, 22.7, 20.8

256 UNIX/LINUX 기본 명령어 A

cat	파일 내용을 화면에 표시함
25.11 cd	디렉터리의 위치를 변경함
23.10, 20.7 chown	파일 소유자와 그룹을 변경함
25.11 cp	파일을 복사함
rm	파일을 삭제함
find	파일을 찾음
kill	PID(프로세스 고유 번호)를 이용하여 프로세스를 종료함
필기 23.5, 23.2, … fork	새로운 프로세스를 생성함
25.11 ls	현재 디렉터리의 파일 목록을 표시함
필기 24.7, 24.5 uname	시스템의 이름과 버전, 네트워크 호스트명 등의 시스템 정보를 표시함
mv	파일을 이동함
ps	현재 실행중인 프로세스를 표시함
25.11 pwd	현재 작업중인 디렉터리 경로를 화면에 표시함
who	현재 시스템에 접속해 있는 사용자를 표시함
필기 25.8, 24.7, 22.3 umask	• 파일이나 디렉터리의 초기 권한을 설정할 때 사용하는 값 • 파일의 경우 666에서 umask를 뺀 값을, 디렉터리의 경우 777에서 umask를 뺀 값을 초기 접근 권한으로 갖음

340448

23.10, 20.7, 필기 23.2

257 chmod A

- chmod는 **파일의 보호 모드를 설정하여 파일의 사용 허가를 지정하는 UNIX 명령어**이다.
- 8진법 숫자를 이용한 방법으로도 파일의 보호 모드를 설정할 수 있다.

예제 UNIX 기반 시스템에서 'batch.sh' 파일에 대해 소유자와 그룹에게는 전체 권한, 기타 사용자에게는 읽기와 실행 권한만 부여하는 명령문을 8진법 숫자를 이용하여 작성하시오.

해설

- UNIX에서는 파일의 권한(permission)을 10자리로 표현하는데 1번째 자리는 디렉터리(d) 또는 파일(–)을, 2~4번째 자리는 소유자(Owner) 권한을, 5~7번째 자리는 그룹(Group) 권한을, 8~10번째 자리는 기타 사용자(Other) 권한을 의미합니다.
- 각 자리는 r(읽기), w(쓰기), x(실행), –(권한없음)으로 표시합니다.

예 – rwx rwx r–x

  - 파일 구분(–) : 파일을 의미
  - 소유자(rwx) : 읽기, 쓰기, 실행 가능
  - 그룹(rwx) : 읽기, 쓰기, 실행 가능
  - 기타 사용자(r–x) : 읽기, 실행만 가능
- 권한을 변경하는 chmod 명령어는 위의 권한 표현 방식을 8진수로 변경하여 사용할 수 있습니다.
- 변경 방법은 파일 구분을 제외한 각 권한을 권한있음(1)과 권한없음(0)으로 바꾼 뒤 8진수로 변환하여 chmod 명령어의 매개 변수로 사용하면 됩니다.

예 rwx rwx r–x

↓ ('–'는 0, 나머지는 1로 바꾸어 준다.)

111 111 101

↓ (3자리 2진수를 8진수로 변환한다. 111 = 7, 101 = 5)

7 7 5

↓ (chmod 명령문을 완성한다.)

chmod 775 batch.sh

340449 21.4, 20.11, 필기 25.5, 24.7, 24.2, 23.7, 23.2, 21.8, 21.3, 20.8, 20.6

258 IP 주소 A

IP 주소(Internet Protocol Address)는 **인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소**이다.

21.4, 필기 21.8

IPv4(Internet Protocol version 4)

- 8비트씩 4부분, 총 32비트로 구성되어 있다.
- 네트워크 부분의 길이에 따라 A 클래스에서 E 클래스까지 총 5단계로 구성되어 있다.

21.4, 20.11, 필기 24.7, 24.2, 23.7, 23.2, 21.3, 20.8, 20.6

IPv6(Internet Protocol version 6)

- IPv6은 현재 사용하고 있는 IP 주소 체계인 IPv4의 주소 부족 문제를 해결하기 위해 개발되었다.
- 16비트씩 8부분, 총 128비트로 구성되어 있다.
- 각 부분을 16진수로 표현하고, 콜론(:)으로 구분한다.
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제를 해결할 수 있다.

340450 필기 20.6

259 IPv6의 주소 체계 C

- **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)

340556 25.7, 25.4, 22.10, 22.7, 필기 25.8, 24.2, 23.5, 23.2, 21.8, 21.5, 20.8

260 서브네팅(Subnetting)의 예 A

예제 192.168.1.0/24 네트워크를 FLSM 방식을 이용하여 3개의 Subnet으로 나누시오. (단 IP Subnet-Zero를 적용했다.)

192.168.1.0/24 네트워크의 서브넷 마스크는 1의 개수가 24개, 즉 C 클래스에 속하는 네트워크이다.

11111111	11111111	11111111	00000000
255	255	255	0

서브넷 마스크를 Subnet으로 나눌 때는 서브넷 마스크가 0인 부분, 즉 마지막 8비트를 이용하면 된다. Subnet으로 나눌 때 "3개의 Subnet으로 나눈다"는 것처럼 네트워크가 기준일 때는 왼쪽을 기준으로 나눌 네트워크 수에 필요한 비트를 할당하고 나머지 비트로 호스트를 구성하면 된다. 3개의 Subnet으로 구성하라 했으니 8비트 중 3을 표현하는데 필요한 2비트를 제외하고 나머지 6비트를 호스트로 구성하면 된다.

			네트워크 ID	호스트 ID
11111111	11111111	11111111	00	000000
255	255	255	192	

호스트 ID가 6Bit로 설정되었고, 문제에서 FLSM(Fixed Length Subnet Mask), 즉 고정된 크기로 주소를 할당하라고 했으므로 3개의 네트워크에 64개(2^6 = 64)씩 고정된 크기로 할당하면 다음과 같다.

네트워크(ID)	호스트 수	IP 주소 범위
1(00)	64	192.168.1.0(00000000) ~ 63(00111111)
2(01)	64	192.168.1.64(01000000) ~ 127(01111111)
3(10)	64	192.168.1.128(10000000) ~ 191(10111111)

340453 21.10, 20.5, 필기 25.8, 25.5, 24.7, 24.2, 23.7, 23.5, 21.5, 21.3, 20.9, 20.8, 20.6

261 OSI 참조 모델 A

OSI 참조 모델은 다른 시스템 간의 원활한 통신을 위해 **ISO(국제표준화기구)에서 제안한 통신 규약(Protocol)**이다.

20.5, 필기 23.5

물리 계층(Physical Layer)

전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의한다.

21.10, 필기 25.8, 24.2, 23.7, 21.3, 20.8

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.

21.10, 필기 25.5, 21.5

네트워크 계층(Network Layer, 망 계층)

개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 한다.

필기 24.7, 20.9, 20.6

전송 계층(Transport Layer)

논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 한다.

필기 25.8

세션 계층(Session Layer)

송 · 수신 측 간의 관련성을 유지하고 대화 제어를 담당한다.

21.10

표현 계층(Presentation Layer)

서로 다른 데이터 표현 형태를 갖는 시스템 간의 상호 접속을 위해 필요한 계층으로, 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색 등의 기능을 수행한다.

응용 계층(Application Layer)

사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 서비스를 제공한다.

340454

262 네트워크 관련 장비 Ⓑ

필기 25.2, 24.7, 21.5

필기 25.2, 24.7, 21.5

라우터(Router)

- 브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택하는 기능이 추가된 장치이다.
- 서로 다른 LAN 또는 LAN과 WAN을 연결하는 기능도 한다.

리피터(Repeater)

거리가 증가할수록 감쇠하는 디지털 신호의 장거리 전송을 위해 수신한 신호를 재생시키거나 출력 전압을 높여 전송하는 장치이다.

허브(Hub)

- 한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치로, 각각의 회선을 통합하여 관리한다.
- 신호 증폭 기능을 하는 리피터의 역할을 포함한다.

필기 25.2, 24.7

브리지(Bridge)

- LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹을 연결하는 장치이다.
- 브리지를 이용한 서브넷(Subnet) 구성 시 전송 가능한 회선 수는 브리지가 n개일 때, n(n-1)/2개이다.

게이트웨이(Gateway)

- OSI 전 계층의 프로토콜 구조가 다른 네트워크를 연결하는 장치이다.
- LAN에서 다른 네트워크에 데이터를 보내거나 다른 네트워크로부터 데이터를 받아들이는 출입구 역할을 한다.

340455

263 프로토콜 Ⓐ

20.10

- 프로토콜(Protocol)은 서로 다른 기기들 간의 **데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약**이다.
- 1965년 톰 마릴(Tom Marill)이 컴퓨터가 메시지를 전달하고, 메시지가 제대로 도착했는지 확인하며, 도착하지 않았을 경우 메시지를 재전송하는 일련의 방법을 '기술적 은어'란 뜻의 프로토콜로 정의한 바 있다.

340456

264 프로토콜의 기본 요소 Ⓐ

20.5

- 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정함
- 의미(Semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정함
- 시간(Timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정함

340457

24.7, 23.4, 21.7

265 패킷 교환 방식 Ⓐ

패킷 교환 방식(Packet Switching)은 **메시지를 일정한 길이의 패킷으로 잘라서 전송하는 방식**으로, 가상 회선 방식과 데이터그램 방식이 있다.

24.7, 21.7

가상 회선 방식

- 단말기 상호 간에 논리적인 가상 통신 회선을 미리 설정하여 송신지와 수신지 사이의 연결을 확립한 후에 설정된 경로를 따라 패킷들을 순서적으로 운반하는 방식이다.
- 정보 전송 전에 제어 패킷에 의해 경로가 설정된다.
- 모든 패킷은 같은 경로로 발생 순서대로 전송된다. 즉 패킷의 송 · 수신 순서가 같다.

24.7, 21.7

데이터그램 방식

- 연결 경로를 설정하지 않고 인접한 노드들의 트래픽(전송량) 상황을 감안하여 각각의 패킷들을 순서에 상관없이 독립적으로 운반하는 방식이다.
- 패킷마다 전송 경로가 다르므로, 패킷은 목적지의 완전한 주소를 가져야 한다.
- 순서에 상관없이 여러 경로를 통해 도착한 패킷들은 수신 측에서 순서를 재정리한다.

340458

23.4, 필기 24.7, 24.5, 23.7, 23.2, 21.5, 21.3, 20.8, 20.6

266 TCP/IP Ⓐ

TCP/IP(Transmission Control Protocol/Internet Protocol)는 인터넷에 연결된 **서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜**이다.

필기 24.7, 24.5, 23.7, 23.2, 21.5, 21.3, 20.8, 20.6

TCP(Transmission Control Protocol)

- OSI 7계층의 전송 계층에 해당한다.
- 가상 회선 방식을 기반으로 하는 양방향 연결 서비스를 제공한다.
- 패킷의 다중화, 순서 제어, 오류 제어, 흐름 제어 기능을 제공한다.

IP(Internet Protocol)

- OSI 7계층의 네트워크 계층에 해당한다.
- 데이터그램 방식을 기반으로 하는 비연결형 서비스를 제공한다.
- 패킷의 분해/조립, 주소 지정, 경로 선택 기능을 제공한다.

340460

필기 24.2, 21.3, 20.9

267 UDP Ⓑ

- UDP(User Datagram Protocol)는 **데이터 전송 전에 연결을 설정하지 않는 비연결형 서비스를 제공하는 프로토콜**이다.
- TCP에 비해 상대적으로 단순한 헤더 구조를 가지므로, 오버헤드가 적고, 흐름 제어나 순서 제어가 없어 전송 속도가 빠르다.
- 실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에서 사용된다.

440388

23.4

268 L2TP Ⓐ

- L2TP(Layer 2 Tunneling Protocol)는 **PPTP와 L2F의 기술적 장점들을 결합하여 만들어진 터널링 프로토콜**이다.
- 데이터 링크 계층에서 구현되는 터널링 프로토콜이다.
- 자체적으로 암호화 및 인증 기능을 제공하지 않아 다른 보안 프로토콜과 함께 사용되는 경우가 많다.
- PPTP(Point to Point Tunneling Protocol) : PPP 패킷을 IP 패킷에 캡슐화하여 통과시키기 위한 터널링 프로토콜
- L2F(Layer 2 Forwarding) : 인터넷을 통한 VPN(가상 사설망) 연결을 위해 개발된 터널링 프로토콜

340462

23.4, 20.10, 필기 22.3

269 ICMP Ⓐ

- ICMP(Internet Control Message Protocol, 인터넷 제어 메시지 프로토콜)는 **IP와 조합하여 통신중에 발생하는** 오류의 처리와 전송 경로 변경 등을 위한 **제어 메시지를 관리하는 역할**을 한다.
- 헤더는 8Byte로 구성된다.

440390

25.4, 21.4, 필기 25.5, 24.5, 24.2, 23.7, 22.7, 20.9, 20.6

270 ARP / RARP Ⓐ

필기 24.5, 24.2, 23.7, 20.9 , 20.6

ARP(Address Resolution Protocol, 주소 분석 프로토콜)

호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소(MAC Address)로 변환하는 기능을 수행하는 프로토콜이다.

21.4

RARP(Reverse Address Resolution Protocol)

ARP와 반대로 물리적 주소를 IP 주소로 변환하는 기능을 수행하는 프로토콜로, 역순 주소 결정 프로토콜이라 불린다.

440391

25.11, 22.10, 21.7, 20.7, 필기 25.5, 25.2, 24.10, 24.7, 23.7, 23.2, 22.7, 22.4, …

271 네트워크 관련 신기술 Ⓐ

필기 24.2, 23.7, 22.7, 22.4, 20.8

메시 네트워크(Mesh Network)

- 차세대 이동통신, 홈네트워킹, 공공 안전 등 특수 목적을 위한 새로운 방식의 네트워크 기술이다.
- 대규모 디바이스의 네트워크 생성에 최적화되어 있다.

필기 20.6

피코넷(PICONET)

여러 개의 독립된 통신장치가 블루투스 기술이나 UWB 통신 기술을 사용하여 통신망을 형성하는 무선 네트워크 기술이다.

24.10, 21.7

애드 혹 네트워크(Ad-hoc Network)

- 재난 현장과 같이 별도의 고정된 유선망을 구축할 수 없는 장소에서 모바일 호스트(Mobile Host)만을 이용하여 구성한 네트워크이다.
- 망을 구성한 후 단기간 사용되는 경우나 유선망을 구성하기 어려운 경우에 적합하다.

필기 20.9

파장 분할 다중화(WDM; Wavelength Division Multiplexing)

광섬유를 이용한 통신기술의 하나로, 파장이 서로 다른 복수의 신호를 보냄으로써 여러 대의 단말기가 동시에 통신 회선을 사용할 수 있도록 하는 기술이다.

필기 20.9

소프트웨어 정의 데이터센터(SDDC; Software Defined Data Center)

- 데이터 센터의 모든 자원을 가상화하여 인력의 개입 없이 소프트웨어 조작만으로 관리 및 제어되는 데이터 센터를 의미한다.
- 컴퓨팅, 네트워킹, 스토리지, 관리 등을 모두 소프트웨어로 정의한다.

20.7

개방형 링크드 데이터(LOD, Linked Open Data)

- Linked Data와 Open Data의 합성어로, 누구나 사용할 수 있도록 웹상에 공개된 연계 데이터를 의미한다.
- 웹상에 존재하는 데이터를 개별 URI(인터넷 식별자)로 식별하고, 각 URI에 링크 정보를 부여함으로써 상호 연결된 웹을 지향하는 모형이다.

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.

25.5, 25.2

클라우드 컴퓨팅(Cloud Computing)

각종 컴퓨팅 자원을 중앙 컴퓨터에 두고 인터넷 기능을 갖는 단말기로 언제 어디서나 인터넷을 통해 컴퓨터 작업을 수행할 수 있는 가상화된 환경을 의미한다.

USN(Ubiquitous Sensor Network)

- 각종 센서로 수집한 정보를 무선으로 수집할 수 있도록 구성한 네트워크이다.
- 필요한 모든 것에 RFID 태그를 부착하고, 이를 통하여 사물의 인식정보는 물론 주변의 환경정보까지 탐지하여 이를 네트워크에 연결하여 정보를 관리한다.

25.11, 22.10, 필기 24.7, 21.8

SSO(Single Sign On)

- 한 번의 로그인으로 개인이 가입한 모든 사이트를 이용할 수 있게 해주는 시스템이다.
- 개인정보를 각 사이트마다 일일이 기록해야 하던 불편함을 해소할 수 있다.
- 기업에서는 회원에 대한 통합관리가 가능해 마케팅을 극대화시킬 수 있다.

340465

필기 23.2, 21.3, 20.8

272 네트워크 구축 Ⓑ

네트워크는 **두 대 이상의 컴퓨터를** 전화선이나 케이블 등으로 **연결하여 자원을 공유하는 것**을 말한다.

성형(Star, 중앙 집중형)

중앙에 중앙 컴퓨터가 있고, 이를 중심으로 단말장치들이 연결되는 중앙 집중식의 네트워크 구성 형태이다.

링형(Ring, 루프형)

컴퓨터와 단말장치들을 서로 이웃하는 것끼리 연결시킨 포인트 투 포인트(Point-to-Point) 방식의 구성 형태이다.

필기 23.2, 21.3, 20.8

버스형(Bus)

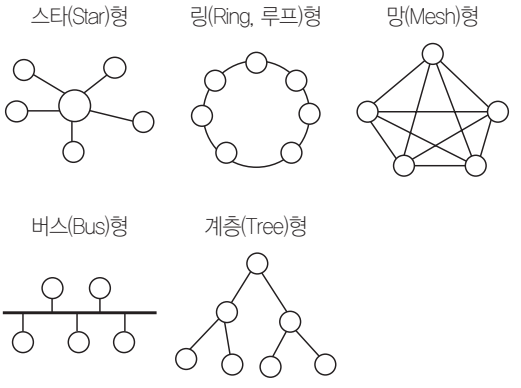
한 개의 통신 회선에 여러 대의 단말장치가 연결되어 있는 형태이다.

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.
- 모든 노드를 망형으로 연결하려면 노드의 수가 n개일 때, n(n-1)/2개의 회선이 필요하고 노드당 n-1개의 포트가 필요하다.



340467

필기 25.2, 21.3

273 IEEE 802의 주요 표준 규격 Ⓒ

IEEE 802는 LAN의 표준안이다.

802.1	전체의 구성, OSI 참조 모델과의 관계, 통신망 관리 등에 관한 규약
802.2	논리 링크 제어(LLC) 계층에 관한 규약
필기 21.3 802.3	CSMA/CD 방식의 매체 접근 제어 계층에 관한 규약
802.4	토큰 버스 방식의 매체 접근 제어 계층에 관한 규약
802.5	토큰 링 방식의 매체 접근 제어 계층에 관한 규약
802.6	도시형 통신망(MAN)에 관한 규약
802.9	종합 음성/데이터 네트워크에 관한 규약
802.11	무선 LAN에 관한 규약

340468 필기 20.6

274 IEEE 802의 주요 표준 규격 Ⓒ

802.11	2.4GHz 대역 전파와 CSMA/CA 기술을 사용해 최고 2Mbps까지의 전송 속도를 지원함
802.11a	5GHz 대역의 전파를 사용하며, OFDM 기술을 사용해 최고 54Mbps까지의 전송 속도를 지원함
802.11b	802.11 초기 버전의 개선안으로 등장하였으며, 초기 버전의 대역 전파와 기술을 사용해 최고 11Mbps의 전송 속도로 기존에 비해 5배 이상 빠르게 개선되었음
필기 20.6 802.11e	802.11의 부가 기능 표준으로, QoS 기능이 지원되도록 하기 위해 매체 접근 제어(MAC) 계층에 해당하는 부분을 수정하였음
802.11g	2.4GHz 대역의 전파를 사용하지만 5GHz 대역의 전파를 사용하는 802.11a와 동일한 최고 54Mbps까지의 전송 속도를 지원함
802.11n	2.4GHz 대역과 5GHz 대역을 사용하는 규격으로, 최고 600Mbps까지의 전송 속도를 지원함

340469 23.10, 20.11

275 NAT Ⓐ

- NAT(Network Address Translation, 네트워크 주소 변환)은 **한 개의 정식 IP 주소에 대량의 가상 사설 IP 주소를 할당 및 연결하는 기능**이다.
- 한 개의 IP 주소를 사용해서 외부에 접속할 수 있는 노드는 어느 시점에서 한 개로 제한되는 문제가 있지만 IP 마스커레이드(Masquerade)를 이용하여 해결할 수 있다.

440398 24.7, 24.4, 23.10, 22.7, 20.10, 필기 25.5, 23.2, 21.5, 20.9, 20.6

276 IGP Ⓐ

IGP(Interior Gateway Protocol, 내부 게이트웨이 프로토콜)는 **하나의 자율 시스템(AS) 내의 라우팅에 사용되는 프로토콜**이다.

24.7, 필기 25.5, 20.9, 20.6

RIP(Routing Information Protocol)

- 현재 가장 널리 사용되는 라우팅 프로토콜로 거리 벡터 라우팅 프로토콜이라고도 불린다.
- 최단 경로 탐색에 Bellman-Ford 알고리즘을 사용한다.
- 소규모 동종의 네트워크(자율 시스템, AS) 내에서 효율적인 방법이다.

24.4, 20.10, 필기 23.2, 21.5

OSPF(Open Shortest Path First protocol)

- RIP의 단점을 해결하여 새로운 기능을 지원하는 인터넷 프로토콜로, 대규모 네트워크에서 많이 사용된다.
- 최단 경로 탐색에 다익스트라(Dijkstra) 알고리즘을 사용한다.

440399 22.7

277 EGP / BGP Ⓐ

EGP(Exterior Gateway Protocol)

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 만들어진 프로토콜이다.
- 초기에 BGP 라우터들이 연결될 때에는 전체 경로 제어표(라우팅 테이블)를 교환하고, 이후에는 변화된 정보만을 교환한다.

340474 필기 20.9

278 흐름 제어 Ⓒ

흐름 제어(Flow Control)란 네트워크 내의 원활한 흐름을 위해 **송 · 수신 측 사이에 전송되는 패킷의 양이나 속도를 규제하는 기능**이다.

필기 20.9

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 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.

340475 20.11, 필기 25.8, 25.5, 24.5, 24.2, 20.9, 20.8

279 SW 관련 신기술 Ⓐ

20.11

블록체인(Blockchain)

P2P(Peer-to-Peer) 네트워크를 이용하여 온라인 금융 거래 정보를 온라인 네트워크 참여자(Peer)의 디지털 장비에 분산 저장하는 기술이다.

필기 25.5, 24.2, 20.8

매시업(Mashup)

웹에서 제공하는 정보 및 서비스를 이용하여 새로운 소프트웨어나 서비스, 데이터베이스 등을 만드는 기술로, 다수의 정보원이 제공하는 콘텐츠를 조합하여 하나의 서비스로 제공하는 웹 사이트 또는 애플리케이션을 말한다.

필기 20.9

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), 업무 프로세스(Biz-Process), 서비스 중간(Service Intermediary), 애플리케이션(Application), 데이터 저장(Persistency) 계층

필기 24.5, 20.8

디지털 트윈(Digital Twin)

- 현실속의 사물을 소프트웨어로 가상화한 모델이다.
- 실제 물리적인 자산을 소프트웨어로 가상화함으로써 실제 자산의 특성에 대한 정확한 정보를 얻을 수 있다.

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.

양자 암호키 분배(QKD; Quantum Key Distribution)

- 양자 통신을 위해 비밀키를 분배하여 관리하는 기술이다.
- 두 시스템이 암호 알고리즘 동작을 위한 비밀키를 안전하게 공유하기 위해 양자 암호키 분배 시스템을 설치하여 운용하는 방식으로 활용된다.

서비스형 소프트웨어(SaaS; Software as a Service)

소프트웨어의 여러 기능 중에서 사용자가 필요로 하는 서비스만 이용할 수 있도록 한 소프트웨어이다.

필기 25.8, 24.5, 23.7

증발품(Vaporware)

판매 계획 또는 배포 계획은 발표되었으나 실제로 고객에게 판매되거나 배포되지 않고 있는 소프트웨어이다.

필기 25.5, 22.3

도커(Docker)

- 컨테이너 기술을 자동화하여 쉽게 사용할 수 있게 하는 오픈소스 프로젝트이다.
- 소프트웨어 컨테이너 안에 응용 프로그램들을 배치시키는 일을 자동화해 주는 역할을 수행한다.

440402 22.10, 필기 25.8, 24.7, 23.7, 22.3, 21.5

280 HW 관련 신기술 A

필기 21.5

앤 스크린(N-Screen)

N개의 서로 다른 단말기에서 동일한 콘텐츠를 자유롭게 이용할 수 있는 서비스이다.

신 클라이언트 PC(Thin Client PC)

- 하드디스크나 주변장치 없이 기본적인 메모리만 갖추고 서버와 네트워크로 운용되는 개인용 컴퓨터이다.
- 서버 기반 컴퓨팅과 관계가 깊다.

필기 25.8, 24.7, 23.7, 22.3

고가용성 솔루션(HACMP; High Availability Clustering Multi Processing)

- 긴 시간동안 안정적인 서비스 운영을 위해 장애 발생 시 즉시 다른 시스템으로 대체 가능한 환경을 구축하는 메커니즘을 의미한다.
- 2개의 서버를 연결하는 이중화를 통해 서버의 안정성을 높일 수 있다.

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.

멤리스터(Memristor)

메모리(Memory)와 레지스터(Resister)의 합성어로, 전류의 방향과 양 등 기존의 경험을 모두 기억하는 특별한 소자이다.

22.10

트러스트존 기술(TrustZone Technology)

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.

340477 22.5

281 RAID A

RAID(Redundant Array of Independent Disk)는 2개 이상의 하드디스크로 디스크 배열을 구성하고, 파일을 구성하는 데이터 블록들을 서로 다른 디스크에 분산 저장하거나 다중화하는 저장 기술로, 구현된 기술에 따라 다음과 같이 레벨(Level)로 구분한다.

22.5

RAID 0

- 스트라이핑(Striping)이라고 불린다.
- 디스크를 병렬로 연결하여 디스크의 개수만큼 용량과 속도가 배로 증가한다.
- 하나의 디스크만 손상되어도 전체 데이터가 파손된다.

RAID 1

- 미러링(Mirroring)이라고 불린다.
- 같은 데이터를 다른 디스크에 동일하게 복사하는 방식이다.

RAID 2~4

- 하나의 디스크에 오류 정정 부호를 비트(RAID 2)/바이트(RAID 3)/워드(RAID 4) 단위로 저장하고, 나머지 디스크는 RAID 0과 같이 활용하여 안정성을 높인 모드이다.
- 하나의 디스크가 손상되어도 정상 가동이 가능하며 최소 3개의 디스크가 필요하다.

RAID 5

- 오류 정정 부호를 블록 단위로 여러 디스크에 분산 저장한 방식이다.
- 하나의 디스크가 손상되어도 정상 가동이 가능하며 최소 3개의 디스크가 필요하다.

RAID 6

- RAID 5와 원리는 같으나 오류 정정 부호 2개를 저장하는 방식이다.
- 두 개의 디스크가 손상되어도 정상 가동이 가능하며 최소 4개의 디스크가 필요하다.

340478

필기 21.5, 20.9

282 Secure OS Ⓒ

- Secure OS는 기존의 운영체제(OS)에 내재된 보안 취약점을 해소하기 위해 **보안 기능을 갖춘 커널(Kernel)을 이식하여 외부의 침입으로부터 시스템 자원을 보호하는 운영체제**를 의미한다.
- 보안 커널은 보안 기능을 갖춘 커널을 의미하며, TCB(Trusted Computing Base)를 기반으로 참조 모니터의 개념을 구현하고 집행한다.
- **Secure OS의 보안 기능** : 식별 및 인증, 임의적/강제적 접근통제, 객체 재사용 보호, 완전한 조정, 신뢰 경로, 감사 및 감사기록 축소 등

340479

20.11, 20.5, 필기 25.5, 24.5, 23.7, 23.5, 23.2, 20.9, 20.8, 20.6

283 DB 관련 신기술 Ⓐ

20.11, 필기 25.5, 23.2, 20.6

하둡(Hadoop)

- 오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼이다.
- 더그 커팅과 마이크 캐퍼렐라가 개발했으며, 구글의 맵리듀스(MapReduce) 엔진을 사용하고 있다.
- 일반 PC급 컴퓨터들로 가상화된 대형 스토리지를 형성하고 그 안에 보관된 거대한 데이터 세트를 병렬로 처리할 수 있도록 개발된 자바 소프트웨어 프레임워크이다.

필기 24.5, 23.5, 20.9

맵리듀스(MapReduce)

- 대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델이다.
- 흩어져 있는 데이터를 연관성 있는 데이터 분류로 묶는 Map 작업을 수행한 후 중복 데이터를 제거하고 원하는 데이터를 추출하는 Reduce 직업을 수행한다.

20.5, 필기 24.5, 23.7, 23.2, 20.8

데이터 마이닝(Data Mining)

대량의 데이터를 분석하여 데이터에 내재된 변수 사이의 상호관계를 규명하여 일정한 패턴을 찾아내는 기법이다.

필기 20.9

타조(Tajo)

오픈 소스 기반 분산 컴퓨팅 플랫폼인 아파치 하둡(Apache Hadoop) 기반의 분산 데이터 웨어하우스 프로젝트이다.

필기 20.9

OLAP(Online Analytical Processing)

- 다차원으로 이루어진 데이터로부터 통계적인 요약 정보를 분석하여 의사결정에 활용하는 방식이다.
- OLAP 연산 : Roll-up, Drill-down, Drill-through, Drillacross, Pivoting, Slicing, Dicing

브로드 데이터(Broad Data)

다양한 채널에서 소비자와 상호 작용을 통해 생성된 것으로, 기업 마케팅에 있어 효율적이고 다양한 데이터이며, 이전에 사용하지 않거나 알지 못했던 새로운 데이터나 기존 데이터에 새로운 가치가 더해진 데이터이다.

340480

필기 25.2, 23.5, 21.3

284 회복 Ⓑ

- 회복(Recovery)은 트랜잭션들을 수행하는 도중 장애가 발생하여 **데이터베이스가 손상되었을 때 손상되기 이전의 정상 상태로 복구하는 작업**이다.
- **회복 기법의 종류** : 연기 갱신 기법, 즉각 갱신 기법, 그림자 페이지 대체 기법, 검사점 기법

340481

22.5

285 REDO / UNDO Ⓐ

22.5

REDO

데이터베이스가 비정상적으로 종료되었을 때, 디스크에 저장된 로그를 분석하여 트랜잭션의 시작(start)과 완료(commit)에 대한 기록이 있는 트랜잭션들의 작업을 재작업한다. 즉 로그를 이용하여 해당 데이터 항목에 대해 이전 값을 이후 값으로 변경하는 연산이다.

22.5

UNDO

데이터베이스가 비정상적으로 종료되었을 때, 디스크에 저장된 로그를 분석하여 트랜잭션의 시작(start)에 대한 기록은 있지만 완료(commit) 기록은 없는 트랜잭션들이 작업한 변경 내용들을 모두 취소한다. 즉 로그를 이용하여 해당 데이터 항목에 대해 이후 값을 이전 값으로 변경하는 연산이다.

340483 20.11, 필기 24.7, 20.8

286 즉각 갱신 기법 A

- 즉각 갱신 기법(Immediate Update)은 트랜잭션이 데이터를 갱신하면 **트랜잭션이 부분 완료되기 전이라도 즉시 실제 데이터베이스에 반영하는 방법**이다.
- 장애가 발생하여 회복 작업할 경우를 대비하여 갱신된 내용들은 Log에 보관시킨다.
- Redo(재시도)와 Undo(취소) 모두 사용 가능하다.

340485 21.7, 필기 20.9, 20.8

287 로킹 A

- 로킹(Locking)은 트랜잭션들이 어떤 로킹 단위를 **액세스하기 전에 Lock(잠금)을 요청해서 Lock이 허락되어야만 그 로킹 단위를 액세스할 수 있도록 하는 기법**이다.
- 주요 데이터의 액세스를 상호 배타적으로 한다.

440411 필기 21.8

288 타임 스탬프 순서 C

- 타임 스탬프 순서(Time Stamp Ordering)는 트랜잭션과 트랜잭션이 읽거나 갱신한 데이터에 대해 트랜잭션이 실행을 시작하기전에 **시간표(Time Stamp)를 부여하여 부여된 시간에 따라 트랜잭션 작업을 수행하는 기법**이다.
- 직렬성 순서를 결정하기 위해 트랜잭션 간의 처리 순서를 미리 선택하는 기법들 중에서 가장 보편적인 방법이다.

340487 필기 25.5, 24.7, 24.5, 24.2, 23.2, 21.8, 21.3, 20.9, 20.8, 20.6

289 로킹 단위 B

- 로킹 단위(Locking Granularity)는 병행제어에서 **한꺼번에 로킹할 수 있는 객체의 크기**를 의미한다.
- 데이터베이스, 파일, 레코드, 필드 등이 로킹 단위가 될 수 있다.
- 로킹 단위가 크면 로크 수가 작아 관리하기 쉽지만 병행성 수준이 낮아진다.
- 로킹 단위가 작으면 로크 수가 많아 관리하기 복잡해 오버헤드가 증가하지만 병행성 수준이 높아진다.

340488 필기 25.2, 22.7, 21.3, 20.6

290 교착상태 B

- 교착상태(Dead Lock)는 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 **서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상**을 의미한다.
- 교착상태 발생의 필요 충분 조건

필기 21.3, 20.6 **상호 배제 (Mutual Exclusion)**	한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함
필기 21.3, 20.6 **점유와 대기 (Hold and Wait)**	최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용되고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 함
필기 21.3, 20.6 **비선점 (Non-preemption)**	다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 함
필기 21.3, 20.6 **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 함

340489

필기 24.7, 24.2, 23.2, 21.5, 20.6

291 교착상태의 해결 방법 Ⓑ

예방 기법 (Prevention)	• 교착상태가 발생하지 않도록 사전에 시스템을 제어하는 방법 • 교착상태 발생의 네 가지 조건 중에서 어느 하나를 제거함으로써 수행됨
필기 23.2, 21.5, 20.6 **회피 기법 (Avoidance)**	• 교착상태가 발생할 가능성을 배제하지 않고 교착상태가 발생하면 적절히 피해 나가는 방법 • 주로 은행원 알고리즘(Banker's Algorithm)이 사용됨
발견 기법 (Detection)	시스템에 교착상태가 발생했는지 점검하여 교착상태에 있는 프로세스와 자원을 발견하는 것
회복 기법 (Recovery)	교착상태를 일으킨 프로세스를 종료하거나 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것

12장 제품 소프트웨어 패키징

340491 20.5

292 릴리즈 노트 작성 항목 Ⓐ

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

- Header(머릿말) : 릴리즈 노트 이름, 소프트웨어 이름, 릴리즈 버전, 릴리즈 날짜, 릴리즈 노트 날짜, 릴리즈 노트 버전 등
- 개요 : 소프트웨어 및 변경사항 전체에 대한 간략한 내용
- 목적 : 해당 릴리즈 버전에서의 새로운 기능이나 수정된 기능의 목록과 릴리즈 노트의 목적에 대한 간략한 개요
- 문제 요약 : 수정된 버그에 대한 간략한 설명 또는 릴리즈 추가 항목에 대한 요약
- 재현 항목 : 버그 발견에 대한 과정 설명
- 수정/개선 내용 : 버그를 수정/개선한 내용을 간단히 설명
- 사용자 영향도 : 사용자가 다른 기능들을 사용하는데 있어 해당 릴리즈 버전에서의 기능 변화가 미칠 수 있는 영향에 대한 설명
- SW 지원 영향도 : 해당 릴리즈 버전에서의 기능 변화가 다른 응용 프로그램들을 지원하는 프로세스에 미칠 수 있는 영향에 대한 설명
- 노트 : SW/HW 설치 항목, 업그레이드, 소프트웨어 문서화에 대한 참고 항목
- 면책 조항 : 회사 및 소프트웨어와 관련하여 참조할 사항 예 프리웨어, 불법 복제 금지 등
- 연락처 : 사용자 지원 및 문의 응대를 위한 연락처 정보

340494 필기 21.8, 21.5, 20.9

293 디지털 저작권 관리의 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 데이터(Meta Data)와 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

340495 필기 25.2, 24.5, 24.2, 23.7, 23.2, 22.7, 21.3, 20.9, 20.8, 20.6

294 디지털 저작권 관리의 기술 요소 Ⓑ

- 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 생성Packager) : 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

340496 필기 20.9

295 소프트웨어 설치 매뉴얼 C

- 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.

440422 22.10, 20.7, 필기 25.8, 25.2, 23.7, 20.9, 20.6

296 소프트웨어 패키징의 형상 관리 A

- 형상 관리(SCM; Software Configuration Management)는 **개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동**이다.
- 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.
- 대표적인 형상 관리 도구에는 Git, SVN, CVS 등이 있다.

340499 20.10

297 형상 관리 기능 A

- **형상 식별** : 형상 관리 대상에 이름과 관리 번호를 부여하고, 계층(Tree) 구조로 구분하여 수정 및 추적이 용이하도록 하는 작업
- **버전 제어** : 소프트웨어 업그레이드나 유지 보수 과정에서 생성된 다른 버전의 형상 항목을 관리하고, 이를 위해 특정 절차와 도구(Tool)를 결합시키는 작업
- **형상 통제** : 식별된 형상 항목에 대한 변경 요구를 검토하여 현재의 기준선(Base Line)이 잘 반영될 수 있도록 조정하는 작업
- **형상 감사** : 기준선의 무결성을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인하는 작업
- **형상 기록** : 형상의 식별, 통제, 감사 작업의 결과를 기록 · 관리하고 보고서를 작성하는 작업

340500 필기 24.7, 24.2, 23.2, 21.5, 20.8

298 소프트웨어의 버전 등록 관련 주요 기능 B

- **저장소(Repository)** : 최신 버전의 파일들과 변경 내역에 대한 정보들이 저장되어 있는 곳
- **가져오기(Import)** : 버전 관리가 되고 있지 않은 아무것도 없는 저장소(Repository)에 처음으로 파일을 복사함
- **체크아웃(Check-Out)** : 프로그램을 수정하기 위해 저장소(Repository)에서 파일을 받아옴
- **체크인(Check-In)** : 체크아웃 한 파일의 수정을 완료한 후 저장소(Repository)의 파일을 새로운 버전으로 갱신함
- **커밋(Commit)** : 체크인을 수행할 때 이전에 갱신된 내용이 있는 경우에는 충돌(Conflict)을 알리고 diff 도구를 이용해 수정한 후 갱신을 완료함
- **동기화(Update)** : 저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화함

340503 필기 25.8, 21.5

299 소프트웨어 버전 관리 도구 - 분산 저장소 방식 C

- 분산 저장소 방식은 버전 관리 자료가 하나의 **원격 저장소와 분산된 개발자 PC의 지역 저장소에 함께 저장되어 관리되는 방식**이다.
- 지역 저장소에서 버전 관리가 가능하므로 원격 저장소에 문제가 생겨도 지역 저장소의 자료를 이용하여 작업할 수 있다.
- **종류** : Git, GNU arch, DCVS, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM 등

340508

필기 25.2, 23.2, 22.4, 20.9

300 빌드 자동화 도구 Ⓑ

- 빌드 자동화 도구는 **빌드를 포함하여 테스트 및 배포를 자동화하는 도구**이다.
- 애자일(Agile)과 같은 지속적인 통합(Continuous Integration) 개발 환경에서 유용하게 활용된다.
- **빌드 자동화 도구 종류** : Jenkins, Gradle, Ant, Maven, Make 등

340510

필기 25.2, 20.9

301 Gradle Ⓒ

- Gradle은 Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구이다.
- 안드로이드 앱 개발 환경에서 사용된다.
- 안드로이드뿐만 아니라 플러그인을 설정하면, JAVA, C/C++, Python 등의 언어도 빌드할 수 있다.
- Groovy를 사용해서 만든 DSL(Domain Specific Language)을 스크립트 언어로 사용한다.