



[정처리] 서술형 대비 요약정리

과목	Aa 단어	내용
1	<u>소프트웨어 아키텍처 패턴</u>	소프트웨어 설계 시 참조 가능한 솔루션 (일반적으로 발생하는 문제점들에 대해 일반화되고 재사용 가능한 솔루션)
1	<u>요구공학 관리 단계 구성 (CMM Level 2)</u>	협기변화 1. 협상 : 구현 가능한 기능 협상 2. 기준선 설정 : 기준선(베이스라인) 설정 3. 변경관리 : 형상통제 위원회를 운영하여 변경 관리 *CCB: 형상 관리의 방침을 정하고 산출물을 검토하는 조직 *베이스라인: 개발 과정의 산출물의 변화를 통제하는 시점 4. 확인 및 검증 : 요구사항에 부합하는지 확인
1	<u>요구사항 확인 기법</u>	1. 정형 기술 검토(TCR) - 동료 검토 (Peer Review) : 작성자가 설명하고, 이해 관계자들이 설명을 들으며 결함 발견 - 워크 스루 (Walk Through) : 검토 자료 사전 배포 후, 짧은 회의 진행 - 인스펙션 (Inspection) : 저작자가 아닌 다른 전문가가 검토 2. 프로토타이핑 활용 3. 테스트 케이스를 통한 확인 4. CASE 도구 활용 5. 베이스라인 검증 6. 요구사항 추적표 (RTM; Requirement Traceability Matirx) 통해 검증 : 요구사항 정의서 기준으로 개발단계별 최종 산출물이 어떻게 변경되었는지 확인 가능한 문서
1	<u>정형 기술 검토 (TCR) 기법</u>	- 관리 리뷰 (Management Review): 프로젝트 진행 상황을 전반적으로 검토 - 기술 리뷰 (Technical Review) : 명세를 준수하고 있는지 검토 - 인스펙션 (Inspection) : 저작자가 아닌 다른 전문가가 검토. ⇒ 참가자 구성: 주재자 (Moderator, 참가자를 선정하고 계획 및 주재) & 작성자, 낭독자, 기록자, 검토자 - 워크스루 (Walk Throughs) : 회의 전 검토자료 배포하여 사전 검토 후, 짧은 시간 동안 회의 진행 - 감사 (Audit) : 제품이 표준이나 가이드라인을 준수하는지 검토. 제품 제공자, 소비자, 제3기관이 수행
1	<u>요구사항 명세 원리 및 검증 항목</u>	명완검 일수 개추 1. 명확성 : 각 명세 내용은 하나의 의미만 부여 2. 완전성 : 모든 요구사항이 포함되어야 함 3. 검증 가능성 : 달성 정도를 확인할 수 있어야 함 4. 일관성 : 모순이 없어야 함 5. 수정 용이성 : 쉽게 수정할 수 있어야 함 6. 개발 후 이용성 : 운영 및 유지보수에 이용이 가능해야 함 7. 추적 가능성 : 추적이 가능해야 함
1	<u>요구사항 명세 기법</u>	1. 비정형 명세 기법 : 자연어 기반 서술 2. 정형 명세 기법 : 수학적 표기법으로 서술
1	<u>요구사항 분석 단계</u>	분개할협분 1. 요구사항 분류 : 기능적 요구사항(시스템이 제공해야 할 기능) vs. 비기능적 요구사항(시스템이 준수해야 할 제약사항) 2. 개념 모델링 생성 : 주로 UML 사용. 요구사항을 쉽게 이해할 수 있도록 개념적 표현 3. 요구사항 할당 : 요구사항 만족을 위한 아키텍처 구성요소 식별 4. 요구사항 협상 : 충돌되는 경우 합의, 우선순위 부여 5. 정형 분석 : 정형화된 언어를 통해 수학적 기호로 표현
1	<u>요구사항 도출 기법</u>	- 인터뷰 : 직접 대화 - 브레인스토밍 : 말하기 쉬운 분위기 속에서 비판없이 의견을 수용 - 델파이 기법 : 전문가 경험 활용 - 롤 플레잉 : 각자 맡은 역할을 연기 - 워크숍 : 단기간 집중하여 정보 획득 후 공유 (사전 준비 필요) - 설문 조사
1	<u>요구공학 개발 단계 구성 (CMM Level 3)</u>	도분명확 1. 도출 : 이해관계자 식별, 고객 분석 2. 분석 : 분개할협분 분류 → 개념 모델링 생성 → 할당 → 협상 → 분석 3. 명세 : 정형화된 형태로 명세 작성 4. 확인 : 요구사항 이해를 확인하고 문서가 완전한지 검증

▼ 과 목	Aa 단어	≡ 내용
1	<u>요구공학 프로세스</u>	1. 개발 단계 (CMM Level 3) : 도분명확 요구사항을 분석하자!! 2. 관리 단계 (CMM Level 2) : 협기변화 설계-개발-테스트를 거치는 동안 요구사항 잘 만족하고 있는지 볼까~
1	<u>요구사항 분석 기법</u>	- 자료 흐름 지향 분석 : 데이터 흐름도(DFD)와 자료 사전(DD)을 통해 분석 - 객체지향 분석 : 시스템 기능과 데이터를 함께 분석해 UML로 표준화
1	<u>요구공학</u>	요구사항을 도출, 분석, 명세, 확인하는 구조화된 활동
1	<u>미들웨어</u>	컴퓨터와 컴퓨터 간 연결 및 연결 관리를 돕는 소프트웨어
1	<u>DBMS</u>	데이터베이스를 관리할 수 있는 응용 프로그램
1	<u>DBMS 분석 시 고려 사항</u>	가성호기구 1. 성능 측면 (가용성, 성능, 상호 호환성) 2. 지원 측면 (기술 지원, 구축 비용)
1	<u>네트워크</u>	원하는 정보를 수신자에게 정확하게 전달하기 위한 인프라
1	<u>OSI 7계층</u>	네트워크 통신에서 충돌 문제를 최소화하고자, 국제표준화기구(ISO)에서 제시한 네트워크 통신 규약
1	<u>OS 현행 시스템 분석</u>	신성기주구 1. 품질 측면 (신뢰도, 성능) 2. 지원 측면 (기술 지원, 주변 기기, 구축 비용)
1	<u>OS</u>	컴퓨터의 하드웨어를 사용자가 쉽게 사용할 수 있도록 인터페이스를 담당하는 소프트웨어
1	<u>메멘토 (Memento).</u>	특정 시점의 객체 내부 상태를 객체화하여, 해당 시점으로 되돌리는 기능을 제공
1	<u>책임 연쇄 (Chain of Responsibility).</u>	한 객체가 요청을 처리하지 못하면, 연결된 객체로 넘어가 처리
1	<u>전략 (Strategy).</u>	동일한 계열의 알고리즘을 캡슐화하고, 전략을 선택해 사용
1	<u>상태 (State).</u>	객체의 상태를 캡슐화하고, 이를 참조해 동작을 다르게 처리
1	<u>반복자 (Iterator).</u>	접근이 잦은 객체에 대해 동일한 인터페이스를 사용하도록 하는 패턴 (내부 노출 없이 순차적 접근 가능)
1	<u>비지터 (Visitor).</u>	처리 기능을 별도로 분리한 패턴 (분리된 처리 기능은 클래스를 방문하여 수행)
1	<u>커맨드 (Command).</u>	요청을 객체로 캡슐화하여, 각 요청(명령)이 들어오면 그에 맞는 서브 클래스 실행
1	<u>옵서버 (Observer).</u>	객체를 지켜보고 있다가, 객체의 상태가 변화면 그 객체에 의존하는 다른 객체들에게 변화된 상태를 전달
1	<u>템플릿 메소드 (Template Method).</u>	상위 클래스에서 기능을 정의하고, 하위 클래스에서 세부 처리 방법을 구체화하는 패턴 cf. 팩토리 메소드 - 상위 클래스에서 인터페이스 정의 후 하위 클래스에서 실제 생성
1	<u>인터프리터 (Interpreter).</u>	여러 언어 구문을 해석할 수 있게 해주는 패턴
1	<u>중재자 (Mediator).</u>	객체 사이에 중재자를 두어 의존성을 줄이는 패턴

▼ 과 목	Aa 단어	≡ 내용
1	<u>데코레이터</u> (Decorator).	객체 결합을 통해 기능을 확장
1	<u>어댑터</u> (Adapter).	호환성 없는 클래스의 인터페이스를 이용할 수 있게 변환
1	<u>디자인 패턴 - 행위 패턴</u>	미인템옵비커이체스스메 1. 미디에이터 2. 인터프리터 3. 템플릿 메소드 4. 옵서버 5. 비지터 6. 커맨드 7. 이터레이터 8. 체인 오브 리스폰서빌리티 9. 스테이트 10. 스트레이트지 11. 메멘토
1	<u>플라이웨이트</u> (Flyweight).	객체가 필요할 때 생성하는 대신 공유하여 메모리 절약
1	<u>컴포지트</u> (Composite).	객체 관계를 파일 트리 구조로 구성하여, 복합 객체와 단일 객체를 동일하게 취급
1	<u>브리지</u> (Bridge).	구현부에서 추상층을 분리하여 결합도를 낮춘 패턴
1	<u>퍼사드</u> (Facade).	복잡한 시스템에 단순한 인터페이스를 제공해 접근성을 높인 패턴
1	<u>프로토타입</u> (Prototype).	원형 객체를 복사하여 생성 (객체 생성 시 갖춰야할 기본 형태가 있을 때 사용)
1	<u>빌더</u> (Builder).	객체를 조립하여 생성. 생성 방법과 구현 방법을 구분하여, 동일한 객체 생성이여도 다른 결과가 나올 수 있음
1	<u>추상 팩토리</u> (Abstract Factory).	구체적인 클래스에 의존하지 않고, 연관된 객체들의 그룹으로 생성 (객체 간 결합이 느슨해짐)
1	<u>싱글톤</u> (Singleton).	클래스 내 객체가 하나 뿐임을 보장. 하나의 객체를 생성해 어디든 참조할 수 있으나 동시 참조 불가
1	<u>팩토리 메소드</u> (Factory Method).	상위 클래스에서 인터페이스 정의, 서브 클래스가 실제 생성
1	<u>디자인 패턴 - 구조 패턴</u>	퍼플컴프브어데 1. 퍼사드 2. 플라이웨이트 3. 컴포지트 4. 프록시 5. 브리지 6. 어댑터 7. 데코레이터
1	<u>디자인 패턴 - 생성 패턴</u>	팩프빌싱추 1. 팩토리 메소드 2. 프로토타입 3. 빌더 4. 싱글톤 5. 추상 팩토리
1	<u>디자인 패턴 유형</u>	생구행 1. 생성 (5) 2. 구조 (7) 3. 행위 (11)
1	<u>디자인 패턴 구성요 소</u>	패문솔 사결샘 1. 패턴 이름 2. 문제 및 배경 3. 솔루션 4. 사례 5. 결과 6. 샘플코드 cf. 라이브러리의 구성은 도설샘 (도움말, 설치파일, 샘플코드)
1	<u>디자인 패턴</u>	소프트웨어 설계 시 자주 쓰이는 방법을 정리한 패턴으로 참고 시 개발 효율성이 높아진다
1	<u>소프트웨어 아키텍처 비용 평가 모델</u>	싸카 (SACAA) 1. SAAM : 변경 용이성과 기능성에 집중. 경험없어도 쉽게 사용 가능 2. ATAM : SAAM을 계승. 아키텍처 품질 속성을 만족하는지도 평가 2. CBAM : ATAM에 경제성 평가 보장 3. ADR : 아키텍처 구성요소 간 응집도 평가 4. ARID : ATAM+ADR. 전체가 아닌 특정 부분에 대한 비용 평가
1	<u>모델-뷰-컨트롤러</u> (MVC) 패턴	- 3개의 서브시스템으로 구조화한 패턴 1) 모델 : 핵심 기능과 데이터 보관 2) 뷰 : 사용자에게 정보 표시 3) 컨트롤러 : 사용자의 입력 처리 - 하나의 모델에 여러 개의 뷰를 필요로 하는 대화형 애플리케이션에 적합

▼ 과 목	Aa 단어	≡ 내용
1	<u>브로커 패턴</u>	- 사용자가 요청하면, 브로커가 적합한 컴포넌트를 연결하는 방식 - 원격 서비스 호출에 응답하는 컴포넌트가 여럿일 때 적합
1	<u>파이프-필터 패턴</u>	- 데이터 스트림을 처리하는 시스템에서 사용 ex) Unix의 Shell - 하나의 서브시스템이 데이터를 받아 처리하고, 결과를 다음 서브 시스템에게 넘겨줌
1	<u>계층화 패턴</u>	- 시스템을 계층으로 구분 ex) OSI 7계층 - 서로 마주보는 계층에서만 상호작용 발생
1	<u>클라이언트-서버 패턴</u>	- 하나의 서버 + 다수의 클라이언트 - 사용자는 클라이언트와만 상호작용
1	<u>소프트웨어 아키텍처 패턴 종류</u>	게를서 파필 브모 1. 계층화 패턴 2. 클라이언트-서버 패턴 3. 파이프-필터 패턴 4. 브로커 패턴 5. 모델-뷰-컨트롤러 패턴
1	<u>구조 뷰</u>	소프트웨어 모듈의 구성을 보여주는 뷰
1	<u>논리 뷰</u>	기능적인 요구사항이 어떻게 제공되는지 표현한 뷰
1	<u>배치 뷰</u>	컴포넌트가 물리적인 아키텍처에 어떻게 배치되는가를 보여주는 뷰
1	<u>유스케이스 뷰</u>	유스케이스를 도출하고 다른 뷰를 검증하는 뷰
1	<u>프로세스 뷰</u>	비기능적인 속성으로 자원 사용 등을 표현한 뷰
1	<u>소프트웨어 4+1 뷰</u>	요구사항을 4개의 관점에서 바라보는 방법. 4개 구조가 충돌되지 않는지, 요구사항을 충족하는지 증명하기 위해 유스케이스 사용
1	<u>현행 시스템 파악</u>	구기인 아소 하네 1) 구성 현황 / 기능 현황 / 인터페이스 파악 2) 아키텍처, 소프트웨어 구성 파악 3) 하드웨어, 네트워크 구성 파악
1	<u>소프트웨어 아키텍처</u>	소프트웨어의 구성요소와, 구성요소의 특성, 구성요소 간 관계를 표현하는 구조
1	<u>PERT</u>	낙관치·중관치·비관치의 3점 추정방식으로 일정 관리
1	<u>CCPM (중요 연쇄 공정법)</u>	주 공정법의 연쇄법으로, 자원 제약사항을 고려해 계산
1	<u>FP (기능점수) 모형</u>	요구 기능별로 가중치를 부여해 총 점수를 계산해 비용 산정
1	<u>CPM (주 공정법)</u>	여러 작업의 수행 순서가 얹힌 프로젝트에서 일정을 계산하는 기법 - 임계 경로 (Critical Path) 계산법 ⇒ 가장 긴 경로 계산!
1	<u>일정 관리 모델</u>	씨씨필 1. CPM 2. CCPM 3. PERT
1	<u>푸트남 (Putnam) 모형</u>	생명주기 단계별 인력분포를 예측하는 방식 (시간에 따른 함수로 표현되는 Rayleigh-Norden 곡선 분포도 기초)
1	<u>Man Month 모형</u>	한 사람이 1개월 간 할 수 있는 일의 양을 기준으로 비용 산정 $(\text{Man Month}) = (\text{LoC}) / \text{개발자의 월간 생산성} \quad (\text{프로젝트 기간}) = (\text{Man Month}) / \text{프로젝트 인력}$
1	<u>COCOMO 모형</u>	프로그램 규모에 따라 비용을 산정 조반임 1) 조직형/단순형 (Organic) : 소규모. 5만 라인(50KSDI) 이하 2) 반 분리형 (Semi-Detached) : 중간형. 30만 라인 (30KSDI) 이하 3) 임베디드형 (Embedded) : 초대형.

▼ 과 목	Aa 단어	≡ 내용
1	<u>비용산정 모델</u>	소프트웨어 개발 계획을 수립하기 위해, 투입될 자원이나 시간을 산정하는 방식 - 하향식 : 전문가가 산정 ex. 델파이 기법 - 상향식 : 요구사항과 기능에 따라 산정 ex. LoC, Man Month, COCOMO, 푸트남, FP
1	<u>LoC (Lines of Codes) 모델</u>	코드 라인 수의 예측치를 구하여 비용 산정 - 산정방법 낙중비46 (낙관치 + 중관치 *4 + 비관치) / 6 ex. (낙관100+중관150*4+비관200) / 6 = 150
1	<u>린의 7가지 원칙</u>	낭비지확인사전 1. 낭비제거 2. 품질 내재화 3. 지식 창출 4. 늦은 확정 5. 빠른 인도 6. 사람 존중 7. 전체 최적화
1	<u>스크럼 용어</u>	- 백로그 : 제품에 대한 요구사항 - 스프린트 : 짧은 기간 내 반복적으로 으쌔으쌔 - 데일리(스크럼) 미팅 : 매일 To-Do List 계획수립. 번다운 차트 작성  - 스크럼 마스터 : 프로젝트 리더 - 스프린트 회고 : 각자 반성하고 개선점 확인  - 번 다 운 차트 : 남아있는 백로그 대비 시간을 시각적으로 표현 (백로그를 수직, 시간을 수평)
1	<u>스크럼</u>	매일 정해진 시간/장소에서 짧은 시간의 개발을 위한 애자일 방법론
1	<u>린</u>	낭비 요소를 제거해 품질을 향상시키는 애자일 방법론
1	<u>XP의 12가지 기본 원리</u>	1. 짝 프로그래밍 (Pair Programming) : 다른 사람과 페어로 개발하여 공동 책임 을 지님 2. 공동 코드 소유 (Collective Ownership) : 시스템에 있는 코드는 누구 나 언제든지 수정 가능 3. 지속적인 통합 (CI; Continuous Integration) : 여러 번 소 프트웨어를 통합하고 빌드해야 함 4. 계획 세우기 (Planning Process) : 고객이 원하는 가치를 정의하고, 개발에 필요한 건 무엇이며, 어떤 곳에서 지연이 될 수 있는지 알려줘야 함 5. 작은 릴리즈 (Small Release) : 작은 시스템을 먼저 만들 고, 짧은 단위로 업데이트 6. 메타포어 (Metaphor) : 공통 이름 체계를 통해 의사 소통을 원활히 7. 간단한 디자인 (Simple Design) : 요구사항에 적합한 단순한 시스템을 설계 8. 테스트 기반 개발 (TDD; Test Drive Develop) : 테스트를 먼저 수행하고, 통과할 수 있는 코드를 작성 9. 리팩토링 (Refactoring) : 기능을 바꾸 지 않으면서 중복제거, 단순화 등을 위해 코드를 재구성 10. 40시간 작업 (40- Hour Work) : 피곤으로 인한 실수가 없도록 주 40시간만 일함시다 11. 고객 상주 (On Site Customer) : 개발자들의 질문에 즉각 대답해줄 수 있는 고객이 풀타임 상주해야 함 12. 코드 표준 (Coding Standard) : 코딩 표준을 두고 효과적으로 개발
1	<u>XP</u>	1~3주의 반복(Iteration) 주기를 갖는 애자일 방법론
1	<u>XP의 5가지 가치</u>	용단의피존 1. 용기 (용기를 갖고 빠르게 개발!) 2. 단순성 (필요한 것만 하자!) 3. 의 사소통 (개발자-관리자-고객 간 원활하게 소통!) 4. 피드백 (의사소통에 대한 빠른 피드백!) 5. 존중 (팀원간 상호 존중!)
1	<u>컴포넌트 기반 방법 론</u>	컴포넌트를 조립해 작성하는 방법론
1	<u>객체지향 방법론</u>	객체라는 단위로 시스템을 설계하는 방법론
1	<u>애자일 방법론</u>	절차보다 사람이 우선되는, 변화에 유연한 경량 개발 방법론
1	<u>제품 계열 방법론</u>	제품에 적용할 공통 기능을 정의하여 개발하는 방법론 (임베디드 소프트웨어 작 성에 유용)

▼ 과 목	Aa 단어	≡ 내용
1	<u>정보공학 방법론</u>	정보 시스템 개발에 필요한 절차를 체계화한 방법론 (대형 프로젝트)
1	<u>구조적 방법론</u>	- 전체 시스템을 나눠 개발하고 통합하는 분할-정복 방식의 방법론 - 나찌-슈나이더만 차트 사용
1	<u>반복적 모델</u> (Iteration Model)	병렬적으로 개발 후 통합하거나, 반복적으로 개발해 점차 완성시켜나가는 모델
1	<u>소프트웨어 개발방법론</u>	소프트웨어의 개발 시작부터 전 개발 과정을 형상화한 방법론 종류: 구정 객컴 애제
1	<u>프로토타이핑 모델</u> (Prototyping Model)	- 주요 기능을 프로토타입으로 구현하고, 피드백을 반영해 만들어나가는 모델
1	<u>나선형 모델</u> (Spiral Model)	위험을 최소화하기 위해, 점진적으로 개발해나가는 모델 계위개고 - 절차: 계획 및 정의 → 위험 분석 → 개발 → 고객 평가
1	<u>폭포수 모델</u> (Waterfall Model)	- =선형 순차적 모델, =고전적 생명주기 모델 - 각 개발 단계를 마무리 지은 후 넘어가는 모델 - 가장 오래됐고, 성공사례가 많으며, 단계별 산출물이 명확하고, 요구사항 변경이 어려움
1	<u>SDLC 모델 프로세스</u>	요설구태유 1. 요구사항 분석 : 요구사항을 분석하고, 제약조건·목표 등을 정의 2. 설계 : 수행 방법을 논리적으로 결정 ex. 시스템 구조 설계, 사용자 인터페이스 설계 3. 구현 : 프로그래밍 언어를 사용해 실제로 코드를 작성 ex. 인터페이스 개발, 자료 구조 개발, 오류 처리 4. 테스트 ex. 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트 5. 유지보수
1	<u>SDLC 모델 종류</u>	폭프나반 1. 폭포수 모델 2. 프로토타이핑 모델 3. 나선형 모델 4. 반복적 모델
1	<u>SDLC (소프트웨어 생명주기)</u>	시스템의 요구분석 ~ 유지보수까지 전 과정을 모델링한 것
2	<u>UI 설계 도구</u>	화면 설계 도구 - 파워 목업 (파워포인트+목업 기능) - 발사믹 목업 (스케치 St.) - 카카오 오븐 프로토타이핑 도구 - UX핀 - 액슈어 (디스크립션까지 작성 가능) - 네이버 프로토나우 UI 디자인 산출물로 작업하는 프로토타이핑 도구 - 인비전 (시안 업로드 후 인터랙션 적용) - 픽사이이트 (구글이 인수, 모바일 앱 최적화) - 프레이어 (커피스크립트 기반)
2	<u>UI 시나리오 문서 작성 요건</u>	완일이가 추수 1. 완전성 : 누락 없이 최대한 상세하게! 2. 일관성 : 요구사항은 일관적! UI도 일관적으로! 3. 이해성 : 이해하기 쉬워야 함! 4. 가독성 : 쉽게 읽혀야 함! 5. 추적 용이성 : 변경사항을 알아보기 쉬워야 함! 6. 수정 용이성 : 쉽게 수정할 수 있어야 함!
2	<u>상태 다이어그램</u>	객체의 상태와 상태 변화를 표현한 다이어그램 - ● : 시작점 - ● : 종료점 - 선이, 선이 조건, 이벤트
2	<u>활동 다이어그램</u>	시스템의 처리 활동을 순서대로 표현한 다이어그램 - ● : 시작점 - ● : 종료점 - 조건 노드, 병합 노드, 포크 노드
2	<u>커뮤니케이션 다이어그램</u>	객체들이 주고받는 메시지와, 상호작용(객체 간 연관)까지 표현한 다이어그램 - 객체 : 사각형 (객체명:클래스명 으로 표현) - 메시지 : → 로 표현
2	<u>유스케이스 다이어그램</u>	사용자 시점에서 표현한 다이어그램 - 유스케이스 : 찌그러진 원형 - 액터 : ○<-<

과 목	Aa 단어	≡ 내용
2	시퀀스 다이어그램	"시간적 개념" 중심으로 메시지 흐름을 표현한 다이어그램 - 생명선 : - - - - - 실행 : - - [] - - (함수 실행시간)
2	패키지 다이어그램	패키지 관계를 표현한 다이어그램 - 패키지 : 폴더 형태
2	컴포넌트 다이어그램	컴포넌트와 컴포넌트 간 관계를 표현한 다이어그램 - 컴포넌트 : 탭 2개가 달린 직사각형 + 이름
2	클래스 간 관계	연의집 포일실 1. 연관(Association) : 관련되어 있어 // <- -> (양방향은 화살표 생략) 2. 의존(Dependency) : 클래스가 다른 클래스 사용 // < > 3. 집합(Aggregation) : 포함하지만 독립적 [부분-◇전체] // ◇- -◇ 4. 포함(Composition) : 포함하고 생명주기를 함께 해 // ◆- -◆ 5. 일반화(Generalization) : 일반적인지 구체적인지 [구체-▷일반] // ◁- -▷ 6. 실체화(Realization) : 기능으로 묶인 관계야 // ◁ ▷
2	클래스의 접근 제어자	1. public + : 외부 모든 클래스에서 접근 가능 2. protected # : 동일 패키지 (하위 클래스 포함)일 때 접근 가능 3. default - : 자바 전용. 접근 제어자 명시가 없을 때, 동일 패키지(하위 클래스 포함) 또는 파생 클래스에서 접근 가능 4. private : 같은 클래스 내에서만 접근 가능
2	클래스 다이어그램	클래스의 속성(변수), 연산(메서드), 클래스 간 관계를 표현한 다이어그램 - 속성 : 클래스의 구조적 특성 (인스턴스가 보유 가능한 값의 범위) - 접근 제어자 : 접근 가능한 정도
2	UML 스테레오 타입	UML 기본 요소 + 새로운 요소를 더한 확장 매커니즘 << >> (길러멧) 기호 사용 <<include>> : 어떤 시점에 반드시 다른 유스케이스를 실행함 <<extend>> : 어떤 시점에 다른 유스케이스를 실행할 수도 있고 아닐 수도 있음 <<abstract>> : 추상 클래스 (인스턴스 생성X, 공통 특징만 정의) <<interface>> : 모든 메서드와 상수가 추상인 클래스 <<entity>> : 정보 또는 행위를 표현하는 클래스 <<boundary>> : 상호작용을 담당하는 클래스 <<control>> : 로직 및 제어를 담당하는 클래스
2	동적 다이어그램 종류	시유 커할 상타 1. 시퀀스 다이어그램 : "시간적 개념" 중심으로 메시지 표현 2. 유스 케이스 다이어그램 : 사용자 관점에서 표현 3. 커뮤니케이션 다이어그램 : 객체들이 주고 받는 메시지와, 상호작용(객체 간 연관)까지 표현 4. 활동 다이어그램 : 시스템이 수행하는 활동을 표현 5. 상태 다이어그램 : 객체의 상태와 상태 변화를 표현 6. 타이밍 다이어그램 : 객체의 상태 변화와 시간 제약을 표현
2	정적 다이어그램 종류	클객 컴배 복패 1. 클래스 다이어그램 : 클래스 간 관계를 표현 2. 객체 다이어그램 : 객체 간 관계 표현 3. 컴포넌트 다이어그램 : 컴포넌트 간 관계를 표현 *구현 단계에서 사용 4. 배치 다이어그램 : 물리적 요소의 위치 표현 *구현 단계에서 사용 5. 복합체 구조 다이어그램 : 복합 구조인 경우 그 내부 표현 6. 패키지 다이어그램 : 패키지 간 관계 표현
2	UML의 구성요소	사관다 1. 사물 2. 관계 3. 다이어그램
2	UML 다이어그램	- 정적(구조적) 다이어그램 : 클객 컴배 복패 - 동적(행위적) 다이어그램 : 시유 커할 상타
2	UML	표준화된 범용 모델링 언어
2	UML의 특징	가구명문 1. 가시화 언어 : 원활한 의사소통을 위해 가시화 2. 구축 언어 : UML → 소스코드 변환 가능 3. 명세화 언어 4. 문서화 언어

과 목	Aa 단어	≡ 내용
2	<u>스토리보드</u>	정책, 와이어프레임 등 구축하려는 서비스를 위한 정보가 수록된 문서
2	<u>UI 화면 설계 구분</u>	와스프 - 와이어프레임 : 화면 단위의 레이아웃 설계 - 스토리보드 : 와이어프레임 + 설명 - 프로토타입 : 와이어프레임(or 스토리보드) + 동적 효과
2	<u>사용자 요구사항 도출</u>	페르소나 1. 페르소나 정의 2. 개념 모델 정의 3. 요구사항 정의 4. UI 컨셉션
2	<u>UI 개발을 위한 주요 기법</u>	- 3C 분석 : Customer(고객), 자사(Company), 경쟁사(Competitor) 분석 - SWOT 분석 : 강점, 약점, 기회, 위협 요인을 분석 - 시나리오 플래닝 : 다양한 시나리오를 설계해 불확실성 제거 - 사용설 테스트 : 사용자가 직접 제품을 사용하며 과제 수행 - 워크숍 : 집단이 모여 지식, 아이디어를 교환하고 검토하는 연구회
2	<u>UI 표준</u>	액정 스펙트 1. UX 원칙 정의 2. 정책 및 철학 설정 3. 스타일 가이드 정의 4. UI 패턴 모델 정의 5. 조직 구성
2	<u>UI 품질 요구사항 (ISO/IEC 9126 기반)</u>	이신사유효기 1. 이식성 : 다른 환경에도 잘 적응하는가? 이적설대 - 적응성: 다른 환경에도 잘 적응되고 - 설치성: 잘 설치되며 - 대체성: 다른 SW를 대체할 수 있는지 2. 신뢰성 : 오류가 없거나 있더라도 괜찮은가? 신성고회 - 성숙성: 오류가 없고 - 고장허용성: 오류가 있어도 성능을 유지할 수 있고 - 회복성: 오류를 금방 회복할 수 있는지 3. 사용성 : 쓰기 편한가? 사이학운 - 이해성: 이해하기 쉽고 - 학습성: 배우기 쉽고 - 운용성: 다루기 쉬운지 4. 유지보수성 : 개선 및 확장이 쉬운가? 유본 변안시 - 분석성: 결함이나 고장을 발견하기 쉽고 - 변경성: 수정하기 쉽고 - 안정성: 수정하더라도 안정적이고 - 시험성: 변경된 내용을 검증할 수 있는지 5. 효율성 : 한정된 자원을 효율적으로 쓰는가? 효시자 - 시간반응성: 처리 속도가 빠르고 - 자원효율성: 적절한 자원을 제공하는지 6. 기능성 : 요구사항을 정확하게 만족하며 기능하는가? 기적정상보호 - 적절성: 적절하고 - 정밀성: 정확하고 - 상호운용성: 상호 운용되고 - 보안성: 보안성 있고 - 호환성: 표준 잘 지키는지
2	<u>UI 설계 지침</u>	사일단결 명표오접가 1. 사용자 중심 : 사용자가 이해하기 쉽도록 2. 일관성 : 조작법을 빨리 이해할 수 있도록 일관적으로 설계해야 함 3. 단순성 4. 결과 예측 가능 5. 명확성 : 개념적으로 인지하기 쉬어야 함 6. 표준화 : 디자인 표준으로 선행학습 이후 쉽게 사용할 수 있어야 함 7. 오류 발생 해결 : 오류 상황을 인지할 수 있어야 함 8. 접근성 : 연령, 성별 등 다양한 계층을 수용해야 함 9. 가시성 : 주요 기능은 메인 화면에 노출해야 함
2	<u>UI 설계 원칙</u>	직유학유 1. 직관성 : 누구나 쉽게 이해하고 쉽게 사용! 2. 유효성: 사용자 목표가 달성될 수 있어야 함! 3. 학습성: 쉽게 배울 수 있다! 4. 유연성: 인터랙션을 최대한 포용
2	<u>UI</u>	사용자와 시스템 사이의 매개체
2	<u>UI 유형</u>	CGNO 1. CLI : 텍스트(명령어) 기반 2. GUI : 그래픽 기반 (마우스, 펜) 3. NUI : 신체 부위 이용 (터치, 음성) 4. OUI : 유기적 상호작용 기반 인터페이스 (모든 사물이 입출력장치로 변화)
3	<u>데크</u>	양쪽 끝에서 PUSH(삽입) & POP(삭제)
3	<u>큐</u>	끝(Rear)에서는 Enqueue(삽입) & 앞(Front)에서는 Dequeue(삭제)
3	<u>스택</u>	한 방향으로만 PUSH(삽입) & POP(삭제)

과 목	Aa 단어	≡ 내용
3	<u>ERD (E-R 다이어그램)</u>	현실의 정보를 사람이 이해할 수 있는 형태로 표현해, 개체와 개체의 속성 개체 간 관계를 도식화한 다이어그램
3	<u>웹 마이닝</u>	웹으로부터 얻는 방대한 정보 속에서 의미있는 정보를 찾아내는 기법 (데이터 마이닝 기술 응용)
3	<u>텍스트 마이닝</u>	대량의 텍스트 속에서 의미있는 정보를 찾아내는 기법 (자연어, 문서 처리기술 적용)
3	<u>NoSQL의 유형</u>	키컬도그 (DMBS 유형에서 'DBMS'라는 단어 제외) 1. Key-Value 2. Column Family Data Store 3. Document Store 4. Graph
3	<u>데이터 마이닝</u>	대규모 데이터 속에서 의미있는 정보를 파악해 의사결정에 활용하는 기법
3	<u>데이터 마이닝 기법</u>	분연 연대 1. 분류 규칙 (Classification) : 과거 데이터로부터 분류 모델을 만들어 이를 토대로 새로운 결과 값을 예측 2. 연관 규칙 (Association) : 데이터 항목 간 종속 관계를 찾아냄 ex. 넥타이 구매자는 셔츠도 같이 구매함 3. 연속 규칙 (Sequence) : 연관 규칙에 '시간' 관련 정보가 포함된 기법 4. 데이터 군집화 (Clustering) : 유사한 특성을 지닌 그룹으로 분류하되 정보가 없는 상태에서 분류
3	<u>NoSQL의 특징</u>	BASE 1. Basically Available : 언제든지 접근 가능해야 함 2. Soft-State : 노드의 상태는 외부 정보로 결정됨 3. Eventually Consistency : 일정 시간이 지나면 데이터 일관성이 유지됨
3	<u>NoSQL</u>	=Not Only SQL. 전통적인 RDBMS가 아닌 DBMS를 지칭 (조인 연산 불가, 수평적으로 확장 가능)
3	<u>빅데이터 분석 및 처리 기술</u>	1. 빅데이터 분석 - 데이터 가공 ex) 피그(Pig), 하이브(Hive) - 데이터 마이닝 ex) 머하웃(Mahout) 2. 빅데이터 실시간 처리 - 실시간 SQL 처리 ex) 임팔라 (Impala) - 요청 작업의 워크플로우 관리 ex) 우지(Oozie) 3. 분산 코디네이션 - 분산 처리 기술 ex) 주키퍼(Zookeeper) 4. 분석 및 시각화 - 시각화 기술 ex) R
3	<u>빅데이터 수집 및 처리 기술</u>	1. 비정형 데이터 수집 ex) 척화(Chuckwa), 플럼(Flume), 스크라이브(Scribe) 2. 정형 데이터 수집 ex) 스쿱(Squoop), 하이호(Hiho), ETL, FTP 3. 분산 데이터 저장/처리 ex) HDFS(하둡 분산 파일 시스템), 맵리듀스(구글이 발표) 4. 분산 데이터 베이스 ex) HBase
3	<u>빅데이터 특성</u>	V3 1. Volume (양) : PB 수준의 대규모 데이터 2. Variety (다양성) : 유형이 로그, 소셜, 위치 등 다양해짐 3. Velocity (속도) : 정보가 빠르게 증가하고 수집됨
3	<u>DBMS의 특징</u>	일회무효보 1. 데이터 일관성 : 조작 후에도 데이터는 변함없음 2. 데이터 회복성 : 장애 발생 시 복구되어야 함 3. 데이터 무결성 : 동일한 내용에 서로 다른 데이터가 저장되지 않아야 함 4. 데이터 효율성 : 응답 시간, 저장 공간 등을 최적화해야 함 5. 데이터 보안성 : 불법적인 노출, 변경으로부터 보호해야 함
3	<u>온톨로지</u>	실세계에 존재하는 개념 정보를 컴퓨터가 이해할 수 있도록 서술한 지식베이스
3	<u>Graph DBMS</u>	- 시멘틱 웹과 온톨로지 분야에서 활용되는 그래프로 데이터를 표현 ex) Neo4j, AllegroGraph
3	<u>Document Store DBMS</u>	- 값의 데이터 타입이 문서 타입(XML, JSON)인 DBMS ex) MongoDB, Couchbase
3	<u>Key-Value DBMS</u>	- Unique한 키에 하나의 값을 갖는 형태 ex) Redis, DynamoDB

과 목	Aa 단어	≡ 내용
3	<u>Column Family Data Store DBMS</u>	- Key안에 (Column, Value) 조합의 필드를 갖는 DBMS - 구글의 Bigtable 기반 ex) HBase, Cassandra
3	<u>DBMS의 유형</u>	키컬도그 1. Key-Value DBMS 2. Column Family Data Store DBMS 3. Document Store DBMS 4. Graph DBMS
3	<u>DBMS</u>	데이터의 추가, 변경, 삭제 등의 관리 기능을 제공하는 소프트웨어
3	<u>데이터베이스의 종류</u>	1. 파일 시스템 개념 : DB 전 단계의 데이터 관리 방식 2. 관계형 DBMS (=RDMBS) : 관계형 데이터 모델을 기반으로 하는 DB 관리 시스템 3. 계층형 DBMS (=HDMBS) : 데이터를 상하 종속 관계로 계층화한 모델 4. 네트워크형 DBMS (=NDBMS) : 데이터를 망상 형태로 표현한 모델
3	<u>데이터베이스 특성</u>	점변동내 1. 실시간 접근성 : 쿼리에 실시간으로 응답 2. 지속적인 변화 : 삽입/삭제/갱신으로 최신 데이터 유지 3. 동시 공유 : 다수 사용자가 이용 4. 내용 참조 : 사용자가 요구하는 내용으로 참조
3	<u>데이터베이스</u>	다수의 인원과 시스템이 사용할 목적으로 관리하는 데이터 집합 통제운공 1. 통합된 데이터 2. 저장된 데이터 3. 운영 데이터 4. 공유 데이터
3	<u>반 정규화 기법</u>	테병분중 컬중 관중 1. 테이블 병합 2. 테이블 분할 3. 중복 테이블 추가 : 집진특 집계 테이블, 진행 테이블, 특정 부분만을 포함하는 테이블 4. 중복 컬럼 허용 5. 중복 관계 허용
3	<u>반 정규화 (De-Nomralization)</u>	정규화된 개체/속성/관계를 단순화하는 기법 - 장점 : 성능 & 관리 효율성 ↑ - 단점 : 일관성 & 무관성 ↓
3	<u>5차 정규화</u>	조인 종속 제거 (릴레이션의 모든 조인 종속이 후보키를 통해서만 설립)
3	<u>2차 정규화</u>	부분 함수 종속성 제거 (완전 함수적 종속 관계)
3	<u>보이스-코드 정규화</u>	결정자는 모두 후보키
3	<u>3차 정규화</u>	이행 함수 종속성 제거 ($A \rightarrow B, B \rightarrow C, A \rightarrow C$ 일 때 이행 함수 종속 관계)
3	<u>4차 정규화</u>	다치(다중값) 종속 제거
3	<u>정규화 단계</u>	도부이결다조 1NF, 2NF, 3NF, 보이스-코드(BC)NF, 4NF, 5NF
3	<u>1차 정규화</u>	도메인은 원자값으로만
3	<u>이상 현상 (Anomaly)</u>	데이터 중복으로 인해 릴레이션 조작 시 발생하는 비합리적 현상 - 삽입, 삭제, 갱신 이상 삽삭갱
3	<u>E-R 다이어그램 구성요소</u>	- 개체: □ - 속성: ○ - 다중 값 속성: ◎ - 관계: ◇ - 관계-속성: —
3	<u>정규화 (Nomralization)</u>	데이터의 중복성을 제거하여 이상 현상을 방지하는 과정
3	<u>관계 해석</u>	원하는 정보가 무엇인가에 대한 비절차적 언어 (프레디킷 해석 기반)

과 목	Aa 단어	≡ 내용
3	순수 관계 연산자	릴레이션이 하나만 나오면 σ 또는 π 이고, 릴레이션이 둘 나오면 \bowtie 또는 \div 이다! 1. 선택 σ : σ 조건 (R) (R에서 조건을 만족하는 튜플 반환) \Rightarrow 가로 2. 프로젝트 π : π 속성리스트 (R) (R에서 주어진 속성들로만 구성된 튜플 반환) \Rightarrow 세로 3. 조인 \bowtie : $R \bowtie S$ (공통 속성으로 R과 S의 튜플 연결하여 반환) 4. 디비전 \div : $R \div S$ (S의 모든 튜플과 관련된 R의 튜플 반환)
3	관계 대수	원하는 정보를 어떻게 유도하는가에 대한 절차적 정형 언어 - 일반 집합 연산자 합교차카 - 순수 관계 연산자 셀프조디
3	일반 집합 연산자	1. 합집합(Union): $R \cup S$ (전체) 2. 교집합(Intersection): $R \cap S$ (공통) 3. 차집합(Difference): $R - S$ (R에만 존재하고 S에는 없음) 4. 카티션 프로덕트 (CARTESIAN Product): $R \times S$ (R과 S에 속한 모든 튜플 연결)
3	계층 데이터 모델	- 트리 형태 - 상하 관계 존재 - 1:N 관계만 허용
3	네트워크 데이터 모델	- 그래프 형태 - CODASYL DBTG 모델이라고도 함 - 상위-하위 레코드 간 N:M 관계
3	관계 데이터 모델	- 2차원 테이블 형태 - Codd 박사가 제안 - 1:1, 1:N, N:M 자유롭게 표현
3	물리적 데이터 모델	- 객체 생성 (테이블, 뷰, 인덱스 등) - 반 정규화 수행
3	논리적 데이터 모델	- 논리적 데이터베이스 구조로 매핑 - 목표 DBMS 설정, 스키마 설계 - 정규화 수행 - 모델링 종류: 관계내 ① 관계 데이터 모델 ② 계층 데이터 모델 ③ 네트워크 데이터 모델
3	개념적 데이터 모델	- 현실 세계의 정보를 추상적, 개념적으로 표현 - DB 종류에 무관 - 주요 산출물 : E-R 다이어그램
3	데이터 모델의 절차	개논물 개념적 \rightarrow 논리적 \rightarrow 물리적
3	데이터 모델의 구성 요소	연구제 ① 연산 ② 구조 ③ 제약조건
3	데이터 모델	현실 세계의 정보를 컴퓨터가 이해할 수 있도록 표현한 모델
4	UDDI	WSDL의 등록·검색을 위한 저장소 (공개적으로 접근 가능한 레지스트리)
4	WSDL	Web Service Description Language 웹 서비스명, 제공위치 등 웹 서비스의 정보가 기술된 XML 형식의 언어
4	SOAP	HTTP, HTTPS 등을 사용해 XML 기반의 메시지를 교환하는 프로토콜
4	웹 서비스 방식	숲 웹서디행 우디 1. SOAP 2. WSDL 3. UDDI
4	EAI vs. ESB	EAI - 기업 내부 이기종 통합 - 토폴로지 : 포허메하 - 핵심 기술 : 어댑터, 브로커, 메시지 큐 ESB - 기업 간 서비스 통합 - 토폴로지 : 버스 방식의 분산형 - 핵심 기술 : 웹 서비스, 지능형 라우터, 포맷 변환, 개방형 표준
4	Hybrid	그룹 내 = Hub & Spoke 그룹 간 = Message Bus
4	Hub & Spoke	허브 시스템을 통한 중앙 집중 방식
4	Message Bus	애플리케이션 사이에 미들웨어(버스)를 두어 연계

과 목	Aa 단어	≡ 내용
4	<u>Point-to-point</u>	가장 기초적인 1:1 단순 통합방법
4	<u>EAI의 유형</u>	포히메하 1. Point-to-point 2. Hub & Spoke 3. Message Bus 4. Hybrid
4	<u>EAI 구성요소</u>	1. EAI 플랫폼 2. 어댑터 : EAI의 핵심장치. 애플리케이션을 연결하는 데이터 입출력 도구 3. 브로커 : 데이터 전송 시 포맷과 코드를 변환해줌 4. 메시지 큐 : 비동기 메시지를 사용하는 프로그램 사이에서 송수신해주는 기술 5. 비즈니스 워크플로우 : 미리 정의된 워크플로우에 따라 업무 처리
4	<u>연계 매커니즘 수행 절차</u>	1. 연계 데이터 추출 및 생성 2. 코드 매핑 (데이터 변환) 3. 연계 테이블 또는 연계 파일 생성 4. 로그 기록 5. 연계 서버 또는 송수신 어댑터 6. 전송 7. 수신된 데이터 DB에 반영
4	<u>ESB</u>	기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션을 하나의 시스템으로 관리할 수 있게 하는 아키텍처로, 미들웨어(버스)를 중심으로 애플리케이션 통합을 "느슨한 결합" 방식으로 지원
4	<u>Socket</u>	프로세스 간 통신의 접속점 (IP주소+포트번호) 클라이언트 요청을 서버와 연결해줌
4	<u>EAI</u>	기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션 간 연계를 돕는 솔루션
4	<u>간접 연계 기술</u>	이에아 이스비 소켓 1. ESI 2. ESB 3. Socket
4	<u>연계 매커니즘 (간접 연계)</u>	간접 연계 - 장: 서로 상이한 네트워크, 프로토콜 연계 가능 - 단: 성능 저하, 개발 비용 높음
4	<u>하이퍼링크</u>	현재 페이지에서 다른 부분 또는 다른 페이지로 이동하게 해주는 속성
4	<u>JDBC</u>	JDBC 드라이버를 이용해 송신 시스템의 DB와 연결하는 방식 * JDBC : 자바에서 데이터베이스에 접속할 수 있게 해주는 API * JDBC 드라이버의 구성 : Java Application - JDBC API - JDBC Driver Manager - JDBC Driver
4	<u>API</u>	데이터를 주고 받을 때 어떤 방식으로 요청하고 제공받을 수 있는지, 규격을 정해놓은 인터페이스
4	<u>DB Connection</u>	DB Connection Pool을 생성하고 해당 풀 명을 이용하여 연결하는 방식 * 커넥션 풀: DB와 연결된 커넥션을 미리 만들어 풀 속에 저장하고, 필요할 때마다 쓰고 반환하는 기법
4	<u>DB Link</u>	수신 시스템에서 DB 링크를 생성하고, 송신 시스템에서 해당 링크를 참조하는 방식
4	<u>직접 연계 기술</u>	링크연제하 1. DB Link 2. DB Connection 3. API 4. JDBC 5. 하이퍼링크
4	<u>연계 매커니즘 (직접 연계)</u>	직접 연계 - 장: 구현이 쉽고 개발 기간 짧음 - 단: 결합도가 높음
4	<u>연계 데이터 표준화</u>	공개중 1. 인터페이스 데이터 공통부 : 표준 항목 2. 인터페이스 데이터 개별부 : 개별 데이터 3. 인터페이스 데이터 종료부 : 전송데이터의 끝을 알림
4	<u>연계 시스템의 구성</u>	송수중 1. 송신 시스템 : 연계할 데이터를 송신 2. 수신 시스템 : 수신한 데이터를 변환해 저장하고 활용하는 시스템 3. 중계 시스템 : 송신-수신 시스템 사이에서 송수신하고 모니터링하는 시스템

과 목	Aa 단어	≡ 내용
4	<u>인터페이스(연계) 명세서</u>	연계에 필요한 항목을 명세화한 문서 구성요소 : 인최크시데 1. 인터페이스 ID 2. 최대 처리 횟수 3. 데이터 크기 (평균/최대) 4. 시스템 정보 (송수신 각각) : 시스템 명, 업무명, 연계방식 등 5. 데이터 정보 (송수신 각각) : 번호, 필드, 데이터 타입 등
4	<u>개체 정의서</u>	개념 데이터 모델링 단계에서 도출된 개체와 관련된 정보를 명세화한 문서
4	<u>연계 요구사항 분석 시 참고 문서</u>	코테용시 1. 코드 정의서 2. 테이블 정의서 3. 응용 프로그램 구성도 (화면 설계서, 사용자 인터페이스 정의서 등) 4. 시스템 구성도 (소프트웨어 구성도, 하드웨어 구성도, 네트워크 구성도 등)
5	<u>인터페이스 감시 도구</u>	1. 스카우터 2. 제니퍼
5	<u>제니퍼</u>	개발부터 운영에 이르기까지 전 생애주기 동안 모니터링 가능한 감시 도구
5	<u>스카우터</u>	애플리케이션 및 DB 모니터링 가능한 감시 도구
5	<u>오류 처리 방법</u>	화로테 1. 화면에서 오류를 인지하도록 구현 2. 오류 로그 생성 3. 관련 테이블에 오류 사항 기록
5	<u>Watir</u>	<u>루비 기반</u> 의 웹 애플리케이션 테스트 프레임워크
5	<u>NTAF</u>	STAF(재사용 및 확장성) + FitNesse(협업 기능) 장점을 통합한 NHN의 프레임워크
5	<u>Selenium</u>	<u>다양한 브라우저</u> 와 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크
5	<u>STAF</u>	각 테스트 대상 <u>분산 환경</u> 에 '데몬'을 사용하여 테스트를 수행
5	<u>FitNesse</u>	웹 기반 <u>테스트 케이스</u> 를 지원
5	<u>xUnit</u>	Java, C++ 등 <u>다양한 언어</u> 를 지원하는 단위 테스트 프레임워크
5	<u>S-HTTP</u>	클라/서버 간 메시지를 암호화하는 보안 기술 (HTTP를 사용한 경우에만 가능)
5	<u>인터페이스 구현 검증 도구</u>	엑스피 엔셀웨 1. xUnit 2. STAF 3. FitNesse 4. NTAF 5. Selenium 6. Watir
5	<u>SSL/TLS</u>	전송계층(4계층)과 응용계층(7계층) 사이에서 안전한 데이터 전송을 보장하는 보안 프로토콜 구성요소 카호르 CAHHR 1. Change Cipher Spec Protocol : 협상된 Cipher Spec을 상대방에게 알리는 프로토콜 2. Alert Protocol : 경고 메시지 전달 3. Heartbeat Protocol : 클라/서버가 정상 상태인지 확인 4. Handshake Protocol : 클라/서버가 서로 인증하고 암호화 키를 협상 5. Record Protocol : 협상된 Cipher Spec
5	<u>IPSec</u>	네트워크 계층(3계층)에서 사용하는 보안 프로토콜. 인증 헤더(AH) + 암호화(ESP)를 이용 - AH(인증) 프로토콜: MAC를 통해 인증 제공 - ESP(암호화) 프로토콜 : MAC+암호화를 통해 인증+기밀성 제공 - IKE(키관리) 프로토콜 : Key를 주고받는 알고리즘

과 목	Aa 단어	≡ 내용
5	<u>DB 암호화 기법</u>	애플티하 1. API 방식 : 애플리케이션 서버에 암호 모듈 적용 (애플리케이션 서버에 부하 발생) 2. 플러그인 방식 : DB 서버에 암호 모듈 적용 (DB서버에 부하 발생) 3. TDE 방식 : DBMS 커널이 자체적으로 암호화 기능 수행 (Transparent Data Encryption) 4. 하이브리드 방식 : API+플러그인 (부하 분산)
5	<u>인터페이스 구현 방식</u>	① 데이터 통신 사용 : 인터페이스 객체 생성 후 데이터 통신으로 전송 ② 인터페이스 개체(Entity) 사용 : 송신 시스템/수신 시스템에서 인터페이스 테이블 구현
5	<u>REST의 구성</u>	리메메 1. 리소스 2. 메서드 3. 메시지
5	<u>REST 메서드의 종류</u>	1. POST [C] 2. GET [R] 3. PUT [D] 4. DELETE [D]
5	<u>REST</u>	HTTP URI로 자원을 표시하고, HTTP 메서드를 통해 해당 자원에 대한 삽입, 삭제, 갱신 등의 명령을 적용하는 아키텍처
5	<u>AJAX의 동작 원리</u>	1. 요청 이벤트 발생 2. 자바스크립트 호출 3. 자바스크립트가 XMLHttpRequest 객체를 사용해 서버에 요청 (비동기이므로 그동안 다른 일 처리) 4. 서버는 XMLHttpRequest 객체를 가지고 AJAX 요청 처리 5. 전달받은 데이터를 사용해 웹 페이지 일부분을 갱신하는 자바스크립트 호출
5	<u>AJAX의 주요기술</u>	- XMLHttpRequest : 비동기 통신을 담당하는 자바스크립트 객체 - XML : HTML의 단점을 보완하여, 특수한 목적을 갖는 마크업 언어 - DOM : XML 문서를 트리 구조 형태로 접근하게 해주는 API - XSLT (eXtensible Stylesheet Language Transformations) : XML 문서를 다른 XML 문서로 변환하는 데 사용하는 언어 - HTML : 웹 문서를 표현하는 마크업 언어 - CSS : 마크업 언어가 표시되는 방법을 기술하는 언어
5	<u>AJAX</u>	서버-클라 간 비동기적으로 데이터를 교환하기 위한 기술
5	<u>XML</u>	HTML의 단점을 보완하여, 특수한 목적을 갖는 마크업 언어
5	<u>XML의 특징</u>	- 트리 구조이며, 모든 태그에는 종료 태그가 필수 - 속성값은 큰 따옴표(")로 묶고 대소문자를 구분함
5	<u>JSON의 표현 자료형</u>	1. 숫자 2. 문자열 (항상 "" 따옴표 사용!) 3. 배열 (대괄호 [] 표시) 4. 객체 (중괄호 { } 로 표시하며, 이름은 문자열을 쓴다) `` { "이름": "에쉬레 버터", "가격": 15000, "판매처": ["쿠팡", "이마트", "롯데마트"]} ``
5	<u>JSON</u>	'키-값' 쌍으로 이루어진 데이터를 전달하기 위해, 인간이 읽을 수 있는 텍스트를 사용한 포맷 (AJAX에서 많이 사용)
5	<u>인터페이스 기능 구현</u>	선생 전전수파 검후 결반 1. 대상자 선택 (대상이 될 데이터를 SQL로 선택) 2. 인터페이스 데이터 생성 (JSON 등 형식에 맞게 가공) 3. 인터페이스 데이터 전송 요청 4. 인터페이스 데이터 전송 5. 인터페이스 데이터 수신 6. 인터페이스 데이터 파싱 7. 인터페이스 데이터 검증 (포맷 및 제약조건 오류 검사) 8. 후속 기능 수행 (수신 시스템에 정의된 후속 기능 진행) 9. 수신 결과 생성 및 전송 10. 수신 측의 처리 결과 반환
5	<u>컴포넌트 명세서</u>	컴포넌트의 개요, 내부 클래스 동작과 같은 내용을 정의한 것 * 컴포넌트 : 특정 기능을 수행하기 위해 독립적으로 개발되고, 다른 부품과 조립되는 소프트웨어 프로그램
5	<u>인터페이스 명세서</u>	컴포넌트 명세서에 명시된 인터페이스 클래스의 세부 내용을 정의한 것

▼ 과 목	Aa 단어	≡ 내용
5	<u>인터페이스 전송 시 데이터 표준 확인 절차</u>	의미표준식별 1. 입출력 의미 파악 2. 데이터 표준 확인 3. 데이터 항목 식별 4. 데이터 표준 최종확인
5	<u>ESB</u>	기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션을 하나의 시스템으로 관리할 수 있도록 하는 아키텍처로, 미들웨어(버스)를 중심으로 애플리케이션 통합을 느슨한 결합 방식으로 지원
5	<u>EAI 구축 유형</u>	포히메하 1. Point-to-point : 가장 기초적인 1:1 단순 통합 방법 2. Hub & Spoke : 허브 시스템을 통한 중앙 집중 방식 3. Message Bus : 애플리케이션 사이에 미들웨어(버스)를 두어 연계 4. Hybrid : 그룹 내는 허브&스포크, 그룹 간은 메시지 버스
5	<u>EAI 구성요소</u>	1. EAI 플랫폼 2. 어댑터 : 핵심장치. 데이터 입출력 도구 3. 브로커 : 포맷과 코드를 변환해주는 솔루션 4. 메시지 큐 : 비동기 메시지를 사용할 때 송수신하는 기술 5. 비즈니스 워크플로우 : 정의된 워크플로우에 따라 업무를 처리하는 기술
5	<u>EAI</u>	기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션 간 연계를 돕는 솔루션
5	<u>인터페이스를 위한 모듈 연계 방식</u>	1. EAI 2. ESB
5	<u>상세 기능별 인터페이스 명세서 주요항목</u>	아명오개 전후파반 1. 인터페이스 ID 2. 인터페이스명 3. 오퍼레이션 이름 4. 오퍼레이션 개요 5. 사전 조건 6. 사후 조건 7. 파라미터 8. 반환값
5	<u>인터페이스 산출물</u>	(인터페이스) 요구사항 → (인터페이스) 정의서 → (인터페이스) 명세서 → (인터페이스) 설계서
5	<u>시스템 인터페이스 설계서</u>	인터페이스 목록, 각 인터페이스 세부 정보를 정의한 문서
5	<u>인터페이스 종류</u>	사용자 인터페이스 or 시스템 인터페이스
6	<u>클래스 정의</u>	[C++] private: 아래서부터 쪽~ 해당 접근 제어자가 적용됨 [자바] 자바는 변수/메서드 하나하나 접근제어자 지정 필요 [파이썬] 파이썬은 <u>매개변수 앞에 self 키워드 필요</u> (입력받는 값이 없을 때 self를 쓰기 위함)
6	<u>List</u>	[파이썬] 의 자료형. 순서 중요 append(값) , insert(인덱스, 값) , remove(값)
6	<u>Dictionary</u>	[파이썬] 의 자료형. 키-값 쌍 딕셔너리이름[키] = 값 , del 딕셔너리이름[키]
6	<u>Set</u>	[파이썬] 의 자료형. 중복값 무시, 순서X add(값) , remove(값)
6	<u>HashMap</u>	[자바] 의 자료형. 키-값 쌍 put(키, 값) , remove(키) , get(키)
6	<u>ArrayList</u>	[자바] 의 자료형. 순서 중요 add(값) , add(인덱스, 값) , remove(인덱스) , get(인덱스)
6	<u>HashSet</u>	[자바] 의 자료형. 중복값 무시, 순서X add(값) , remove(값)
6	<u>라이브러리의 패키지</u>	모듈을 구조화한 라이브러리
6	<u>라이브러리 구성</u>	도설샘 1. 도움말 2. 설치 파일 3. 샘플 코드
6	<u>라이브러리의 모듈</u>	변수, 함수 등을 모아둔 파일

과 목	Aa 단어	≡ 내용
6	<u>객체지향 프로그래밍의 구성요소</u>	객체 1. 객체 : 개체+속성+메서드로 이루어진 인스턴스 개속메 2. 클래스 : 객체를 표현하는 추상 데이터 타입 3. 메서드 : 객체 간 통신
6	<u>라이브러리</u>	효율적인 개발을 위해 필요한 프로그램들을 모은 집합체 (모듈과 패키지를 총칭)
6	<u>선언형 언어</u>	1. 함수형 언어 - LISP : 수학 표기법을 나타내기 위한 목적 - 하스켈 : 패턴 맞춤, 커링, 조건제시법 등의 기능 2. 논리형 언어 - Prolog : 논리식을 기반으로 인공지능, 자연언어 처리 분야에서 사용 3. 특수분야 언어 - SQL : DBMS 관리를 위한 질의어
6	<u>스크립트 언어</u>	피피피자스 1) PHP : 동적 웹페이지를 위한 언어 / 자체 인터프리터 제공 2) Perl : 실용성 모토 / 인터프리터 3) Python 4) JavaScript
6	<u>객체지향 프로그래밍</u>	씨씨자델 1) C++ : C언어에 객체지향 프로그래밍 개념 추가 / 컴파일 2) C# : MS에서 개발. 불안전 코드(Unsafe Code) 같은 기술을 통해 상호 운용성 확보 3) 자바 : 웹 애플리케이션 개발에 가장 많이 사용되는 언어 4) 델파이 : 파스칼 문법 + 여러 기능. Windows 아래에서 모든 부분 프로그래밍 가능
6	<u>절차적 프로그래밍</u>	1) 포트란 (FORTRAN) : 과학 기술 전문 언어로, 고급 수학 함수들 사용 가능 / 컴파일 2) C 언어 : 유닉스 운영 체제에서 사용하기 위한 언어 / 컴파일 3) 알골 : 알고리즘 연구개발 목적 4) 베이직 : 교육용으로 개발된 언어 / 인터프리터
6	<u>프로그래밍 구현 기법</u>	1) 컴파일 방식: 고급 언어 → 기계어로 번역하는 방식 ex. FORTRAN , PASCAL , C , C++ 2) 인터프리터 방식: 명령문을 하나씩 번역하고 실행하는 방식 ex. BASIC , LISP , PHP , Perl , 프롤로그 3) 혼합형 방식: 고급 언어를 컴파일하여 중간 언어로 변환 후, 인터프리터가 번역을 실행하는 방식 ex. 자바
6	<u>인터페이스</u>	자바에서 다형성을 극대화하기 위한 문법으로, 추상 메서드와 상수만을 멤버로 갖는다 - interface 로 정의하고 implements 로 가져다 쓴다
6	<u>프로그래밍 언어 분류 (실행방식)</u>	1. 명령형 언어 (=절차형 언어): 명령들이 순차적으로 실행되는 방식 포코파씨알베 ex. FORTRAN , COBOL , PASCAL , C , ALGOL , BASIC 2. 객체지향 언어: 객체 간 메시지 통신을 이용하는 방식 ex. 자바 , C++ 3. 함수형 언어: 수학수식과 같은 함수들로 프로그램을 구성하여 호출하는 방식 리하 ex. 리스프(LISP) , 하스켈(Haskell) 4. 논리형 언어: 논리 문장을 이용하여 표현하는 방식 ex. 프롤로그(Prolog)
6	<u>추상 클래스 구현</u>	[C++] : 메서드 뒤에 =0 붙임 [자바] : 클래스명과 메서드명 앞에 abstract 붙임 [파이썬] : 메서드 내부에 pass 키워드를 사용
6	<u>상위 클래스 접근</u>	[C++] : 부모클래스: 부모클래스() [자바] : super.메서드명() //super()로 상위 클래스의 생성자에 접근 [파이썬] : super().메서드명()
6	<u>추상 클래스</u>	자식클래스에서 상위클래스의 미구현 추상 메서드를 구현하는 기능
6	<u>오버로딩</u>	같은 이름의 메서드를 매개변수만 다르게 하여 여러 개 정의하는 것
6	<u>오버라이딩</u>	하위 클래스에서 상위 클래스 메서드를 재정의할 수 있는 기능 [C++] 은 virtual 키워드가 꼭! 필요하다 (부모/자식 중 하나라도 있으면 오버라이딩 되지만, 둘 다 키워드가 없으면 오버라이딩 되지 않으니 주의)
6	<u>상속 문법</u>	[C++] class 자식클래스 : public 부모클래스 { } [자바] class 자식클래스 extends 부모클래스 { } [파이썬] class 자식클래스 (부모클래스):
6	<u>상속</u>	어떤 객체의 변수와 메서드를 다른 객체가 물려받는 것

과 목	Aa 단어	≡ 내용
6	<u>생성자</u>	해당 클래스의 객체가 생성될 때 자동으로 호출되는 메서드 - [C++] [자바] : 클래스명과 동일하면 생성자 - [파이썬] : <code>__init__</code> 라는 메서드명을 사용하면 생성자
6	<u>소멸자</u>	객체의 수명이 끝났을 때 객체를 제거하기 위한 메서드 - [C++] : 클래스명과 동일하고, <code>~</code> 기호를 사용하면 소멸자 - [자바] : <code>finalize</code> 라는 메서드명을 사용하면 소멸자 - [파이썬] : <code>__del__</code> 라는 메서드명을 사용하면 소멸자
6	<u>자신 클래스 참조</u>	[C++] this -> 변수명; this -> 함수명(매개변수); [자바] this.변수명; this.함수명(매개변수); [파이썬] self.변수명 self.함수명(매개변수)
6	<u>접근 제어자</u>	1. public <code>+</code> : 외부 모든 클래스에서 접근 가능 2. protected <code>#</code> : 같은 패키지 내부, 하위 클래스(상속)일 때 접근 가능 3. default <code>~</code> : 자바 전용. 접근 제어자 명시가 없을 때, 같은 패키지 내부에서 접근 가능 4. private <code>-</code> : 같은 클래스 내에서만 접근 가능
6	<u>구조체</u>	- 기본 타입을 갖고 내 입맛대로 새롭게 정의하는 자료형 - [C] [C++] 에서 <code>struct</code> 로 선언 후 사용
6	<u>열거체</u>	- 정수형 상수에 이름을 붙여 쓰기 편하게 만드는 것 - [C] [C++] 은 <code>enum</code> 으로 선언하며 초기값이 없으면 0부터 차례대로 할당
6	<u>루프 제어 명령문</u>	- <code>break</code> : 반복문 중지 - <code>continue</code> : 아래 구문 실행하지 않고 다음 반복문으로 넘어감
6	<u>for문</u>	[C] [C++] [자바] <code>for</code> (초기값; 최종값; 증감 값) { 명령문; } [파이썬] <code>for</code> 변수 in range (시작값, 끝값) : 명령문 - (시작값)부터 (끝값-1)까지 1씩 증가 <code>for</code> 변수 in range (반복횟수) : 명령문 - 0부터 (반복횟수-1)까지 1씩 증가
6	<u>switch문</u>	1) 해당하는 case가 있다 - break가 있으면 switch 문 종료 - break가 없으면 break를 만날 때까지 switch 문 내 다른 문장을 실행 2) 해당하는 case가 없다 - default 실행 - 조건에 해당하는 case도, default도 없다면 아무것도 실행되지 않는다
6	<u>if문</u>	[C] [C++] [자바] <code>if()</code> {} <code>else</code> {} [파이썬] <code>if:</code> <code>elif:</code> <code>else:</code>
6	<u>입력 함수</u>	[C] <code>scanf("%c %d", &a, &b);</code> // 문자열 입력이 아닐 때 → 변수명 앞에 & 붙임 <code>scanf("%s", c);</code> // 문자열 입력일 때 → &는 안 붙임 [C++] <code>std::cin >> a;</code> [자바] <code>Scanner c = new Scanner(System.in); s = c.nextLine();</code> //문자열 <code>i = c.nextInt();</code> //정수형 <code>f = c.nextFloat();</code> //실수형 [파이썬] <code>s = input()</code> <code>s = eval(s)</code> # 문자열인 경우 <code>eval()</code> 함수를 써서 숫자로 변환해 줘야 한다
6	<u>포맷 스트링</u>	<code>%c</code> : 문자 <code>%s</code> : 문자열 <code>%d</code> : 10진수 <code>%x</code> : 16진수 <code>%o</code> : 8진수 <code>%f</code> : 실수 (6자리까지) <code>%[전체자리수].[소수점자리수]f</code> : 해당 자리수만큼 실수 ex. <code>float a = 1.2</code> 일 때 <code>("%.2f", a)</code> 는 1.20 이고, <code>("%.3f\n", a)</code> 는 1.200 이다
6	<u>출력 함수</u>	[C] <code>printf("Hello World");</code> <code>printf("Hello\nWorld");</code> <code>printf("%d", a);</code> [C++] <code>std::cout << "Hello" << endl << "World";</code> [자바] <code>System.out.print("Hello\nWorld");</code> <code>System.out.println("Nana");</code> [파이썬] <code>print('Hello', end='')</code> # 개행 없음 <code>print('World')</code> # 개행 있음 # 아무것도 없이 <code>print()</code> 만 하면 개행처리와 마찬가지로
6	<u>증감 연산자</u>	<code>++x</code> / <code>--x</code> (증감시킨 후 변수 사용) <code>x++</code> / <code>x--</code> (변수 사용 후 증감)

과 목	Aa 단어	≡ 내용
6	<u>비트 연산자</u>	& (같은 비트 값이 모두 1이면 1) (같은 비트 값중 하나라도 1이면 1) ^ (같은 비트값이 서로 다르면 1=XOR) ~ (모든 비트값을 반대로) ex. 5&6 (답:4) - 5는 101, 6은 110 - 1&1=1, 0&1=0, 1&0=0 이므로 100 - 100은 4 ex. 30&15 (답:14) - 30은 11110, 15는 1111 - 둘이 자릿수가 다르면 11110&01111로 계산 - 01110은 십진수로 14
6	<u>2진수 → 10진수</u>	뒤에서부터 순서대로 1,2,4,8,16,32... 작성. 이후 1인 자릿수만 총 더해서 계산 ex) 10100 = 16,8,4,2,1 = 16+4= 20
6	<u>시프트 연산자</u>	<< (우측의 값만큼 왼쪽으로 비트 이동) >> (우측의 값만큼 오른쪽으로 비트 이동) ex. 6<<1 (답:12) - 6을 2진수로 바꾸면 110 - 왼쪽으로 1 이동이므로 1100 - 1100을 10진수로 바꾸면 12
6	<u>10진수 → 2진수</u>	2로 나누고, 나머지를 우측에 작성. 더 이상 2로 나눌 수 없게 되면 아래서부터 위로 읽어나감 ex) 5/2=2(나머지1) → 2/2=1(나머지0) → 답:101
6	<u>헝가리안 표기법</u>	식별자 앞에 자료형을 붙이는 표기법
6	<u>스네이크 표기법</u>	식별자에 여러 단어가 이어질 때 각 단어 사이에 언더바를 넣는 표기법
6	<u>카멜 표기법</u>	식별자에 여러 단어가 이어질 때 첫 단어의 시작만 소문자로, 각 단어의 첫 글자는 대문자로 쓰는 표기법
6	<u>파스칼 표기법</u>	식별자에 여러 단어가 이어질 때 각 단어의 첫 글자는 대문자로 쓰는 표기법
6	<u>리스트 슬라이싱</u>	1) 시작 인덱스: 생략 시 처음부터 2) 종료 인덱스: 종료 인덱스 '전' 까지만 슬라이싱하며 생략 시 마지막까지 3) 스텝: 생략 시 1 <code>a = [7, 6, 3, 4, 0]</code> <code>print(a[:4:2])</code> # 출력: [7,3] <code>a = "Hello Python"</code> <code>b = a[0:3]</code> <code>c = a[-4:-1]</code> <code>print(b+c)</code> # 출력: Heltho
6	<u>리스트 인덱스</u>	[0] [1] ... [n-2] [n-1] [-n] [-(n-1)] ... [-2] [-1]
6	<u>포인터 변수</u>	- 변수 선언 시 데이터 타입 뒤에 * 를 붙이면 포인터 변수가 된다 - 변수명 앞에 & 를 붙이면 해당 변수의 주솟값을 가리킨다. - 주솟값에 들어있는 값은 * 로 얻을 수 있다. // 변수 a의 값이 배열일 때, <code>a[0]</code> 은 0번지 값이고 <code>&a[0]</code> 은 0번지 주소를 가리킴 // <code>printf("%s")</code> 으로 문자열 출력 시, 무조건 NULL 값 바로 앞까지 출력한다 Q. char b[16] = "hello world"; printf("%s", &b[6]); A: <code>world</code>
6	<u>자료형</u>	Java : HashSet, ArrayList, LinkedList, HashMap Python : Set, List, Tuple, Dictionary
6	<u>문자 vs. 문자열</u>	[C] [C++] [자바] 에서 문자 하나는 작은 따옴표('), 문자열은 큰 따옴표("")로 표기 (따옴표는 문자이나 문자열이나를 구분하기 위한 것이지 실제 출력 결과가 아니므로, 출력 결과에 따옴표를 쓰는 건 오답이니 주의!)
7	<u>주요 옵티마이저 힌트</u>	◦ <code>/*+ RULE */</code> : 규칙 기반 접근방식 사용 ◦ <code>/*+ CHOOSE */</code> : 오라클 옵티마이저 디폴트값을 따름 ◦ <code>/*+ INDEX(테이블명 인덱스명) */</code> : 해당 인덱스 강제 사용 ◦ <code>/*+ USE_NL(테이블명) */</code> : 조인 방식을 Nested Loop Join으로 설정 ◦ <code>/*+ USE_MERGE(테이블명) */</code> : 조인 방식을 Sort Merge Join으로 설정 ◦ <code>/*+ USE_HASH(테이블명) */</code> : 조인 방식을 Hash Join으로 설정
7	<u>옵티마이저의 역할</u>	1. 쿼리 변환 : SQL을 표준화된 형태로 변환 2. 비용 산정 : 카디널리티, 비용 등을 계산 3. 계획 생성

과 목	Aa 단어	≡ 내용
7	<u>힌트</u>	힌트를 통해 옵티마이저의 실행 계획을 변경 가능함
7	<u>CBO</u>	- 모든 접근 경로를 고려해 실행 계획을 선택함 (비용=수행시간 기반) - 이해도가 낮아도 성능보장 가능
7	<u>RBO</u>	- 사전에 등록된 규칙에 따라 실행 계획을 선택함 (규칙=우선순위 기반) - 사용자가 원하는 처리 경로로 유도하기 쉬움
7	<u>옵티마이저</u>	SQL을 수행할 최적의 처리경로를 찾는 DBMS의 핵심엔진
7	<u>실행 계획</u> (Execution Plan)	옵티마이저가 생성한 처리경로
7	<u>옵티마이저 유형</u>	1. RBO - 규칙 기반 옵티마이저 2. CBO - 비용 기반 옵티마이저
7	<u>튜닝 개선 절차</u>	1. 문제 있는 SQL 식별 (APM 모니터링 도구 사용) 2. 옵티마이저 통계 확인 3. SQL문 재구성 4. 인덱스 재구성 5. 실행계획 유지관리
7	<u>튜닝 (쿼리 성능 개선)</u>	SQL 실행 계획을 수정해 프로시저 성능을 개선하는 것
7	<u>트리거 문법</u>	트리거 실행 순서 [BEFORE AFTER] • BEFORE : 이벤트 실행 전 • AFTER : 이벤트 실행 후 [FOR EACH ROW] - 매번 변경되는 데이터의 행의 수만큼 명령어 실행 [OLD NEW] - 변경 전/후 데이터 중 어떤 걸 쓸지 선택
7	<u>트리거 구성</u>	디비컨SE 1. 선언부 (DECLARE) 2. 이벤트부 (EVENT) 3. 시작/종료부 (BEGIN/END) 4. 제어부 (CONTROL) 5. SQL 6. 예외부 (EXCEPTION)
7	<u>트리거</u>	DB 시스템에서 삽입, 삭제, 갱신 등의 이벤트 발생 시 관련 작업이 자동으로 수행되는 절차형 SQL (EVENT 명령어로 실행시점을 인지하고, 외부 변수 IN/OUT 이 없고, TCL 사용 불가)
7	<u>사용자 정의함수 구성</u>	디비컨SER 1. 선언부 (DECLARE) 2. 시작/종료부 (BEGIN/END) 3. 제어부 (CONTROL) 4. SQL 5. 예외부 (EXCEPTION) 6. 반환부 (RETURN)
7	<u>사용자 정의함수</u>	일련의 SQL 처리 후, 결과를 단일값으로 반환하는 절차형 SQL
7	<u>프로시저 문법</u>	변수의 입출력 구분 [IN OUT INOUT] • IN : 운영체제 → 프로시저로 값 전달 • OUT : 프로시저의 결과 → 운영체제로 전달 • IN과 OUT 둘 다 수행
7	<u>프로시저 구성</u>	디비컨SET 1. 선언부 (DECLARE) 2. 시작/종료부 (BEGIN/END) 3. 제어부 (CONTROL) 4. SQL 5. 예외부 (EXCEPTION) 6. 실행부 (TRANSACTION)
7	<u>프로시저</u>	일련의 쿼리들을 하나의 함수처럼 실행하기 위한 쿼리의 집합
7	<u>절차형 SQL 제어부</u>	1. IF문 IF ~ THEN ELSIF ~ THEN ELSE ~ END IF; 2. Case문 CASE ~ WHEN ~ THEN ELSE ~ END CASE; 3. Loop문 LOOP ~ EXIT WHEN ~ END LOOP; 4. While문 WHILE ~ LOOP EXIT WHEN ~ END LOOP; 5. For Loop문 FOR ~ IN ~ LOOP ~ END LOOP ~
7	<u>절차형 SQL의 종류</u>	프함트 1. 프로시저 (일련의 쿼리들을 하나의 함수처럼 실행하기 위한 쿼리의 집합) 2. 사용자 정의 함수 (일련의 SQL 처리 후, 그 결과를 단일값으로 반환) 3. 트리거 (DB 시스템에서 삽입, 삭제, 갱신 등의 이벤트 발생 시 관련 작업이 자동으로 수행)

과목	Aa 단어	내용
7	<u>DBMS_OUTPUT</u>	버퍼로부터 메시지를 읽어오는 인터페이스 패키지 <code>DBMS_OUTPUT.PUT(문자열);</code> // 문자열 출력하는 프로시저 <code>DBMS_OUTPUT.PUT_LINE(문자열);</code> // 문자열 출력 후 개행하는 프로시저
7	<u>절차형 SQL</u>	SQL 안에서도 절차 지향적인 프로그래밍이 가능하도록 하는 트랜잭션 언어
7	<u>그룹 내 비율 함수</u>	라투리퍼랭 1. RATIO_TO_REPORT - 그룹의 합을 기준으로 각 로우의 상대적 비율 (로우값/총계) 2. PERCENT_RANK - 그룹에서 먼저 나오는 게 0, 늦게 나오는 게 1로 하여 순서별 백분율 (순위-1)/(총 로우-1) ex. // RATIO_TO_REPORT() 는 합계를 계산할 로우를 매개변수로 지정해주되, 순서를 지정하지 않아도 됨 // PERCENT_RANK() 는 0~1 값이 나오므로 매개변수 불필요, 순서 지정 SELECT 이름, 나이, RATIO_TO_REPORT(레벨) OVER () A, PERCENT_RANK() OVER (ORDER BY 레벨 DESC) B FROM 학생;
7	<u>행 순서 함수</u>	퍼라락리 1. FIRST_VALUE - 윈도우에서 가장 먼저 나오는 값 2. LAST_VALUE - 윈도우에서 가장 늦게 나오는 값 3. LAG - 윈도우에서 이전 로우의 값 4. LEAD - 윈도우에서 다음 로우의 값 ex. // 기준이 될 로우를 OVER내 매개변수로 넣어줘야 함 SELECT 직업, 레벨, FIRST_VALUE(직업) OVER (ORDER BY 레벨 DESC) A FROM 캐릭터;
7	<u>순위 함수</u>	랭덴로 1. RANK (2245) - 동일순위 스킵 2. DENSE_RANK (2234) - 동일순위 그대로 3. ROW_NUMBER (2345) - 그냥 차례대로 ex. // 순위를 보여주는 함수 이므로 OVER내 매개변수 없음 SELECT 직업, 레벨, RANK() OVER (ORDER BY 레벨 DESC) A FROM 캐릭터;
7	<u>데이터 분석 함수 요약</u>	
7	<u>윈도 함수</u>	=OLAP 함수. 종류: 순행비 " SELECT " 절에 사용하며, 함수(로우) OVER(ORDER BY 로우 DESC) 별칭 형태로 사용한다 (OVER 빼먹지 않게 주의!) (별칭 지정할 때 AS 없어도 됨!)
7	<u>GROUPING SETS 함수</u>	컬럼별 개별 집계 산출 SELECT 기숙사, 학년, SUM(점수) FROM 점수 GROUP BY GROUPING SETS(기숙사, 학년, ());
7	<u>CUBE 함수</u>	다차원 집계 산출 SELECT 기숙사, 학년, SUM(점수) FROM 점수 GROUP BY CUBE(기숙사, 학년);
7	<u>ROLLUP 함수</u>	소계 산출 SELECT 기숙사, 학년, SUM(점수) FROM 점수 GROUP BY ROLLUP(기숙사, 학년);
7	<u>그룹 함수</u>	그룹화 후 그룹별로 결과를 출력, " GROUP BY " 절에 사용 롤큐그셋 1. ROLLUP : 소계 산출 2. CUBE : 모든 값에 대한 다차원 집계 산출 3. GROUPING SETS : 컬럼별 개별 집계 (순서 무관)
7	<u>집계 함수</u>	단일값 리턴, " SELECT " 절에 사용 NULL은 없는 데이터로 취급함 1. COUNT : 해당하는 튜플의 수 ex. COUNT(DISTINCT 컬럼) = 중복 제거한 컬럼 수 ex. COUNT(학과) = 해당 학과에 해당하는 튜플의 수 (학생 등) 2. SUM : 합계 3. AVG : 평균 4. MAX : 최댓 5. MIN : 최솟값 6. STDDEV : 표준편차 7. VARIAN : 분산
7	<u>GRANT 명령어</u>	그온투 GRANT 권한 ON 테이블 TO 사용자 [WITH GRANT OPTION];
7	<u>REVOKE 명령어</u>	리온프 REVOKE 권한 ON 테이블 FROM 사용자 [CASCADE CONSTRAINT];
7	<u>DCL 명령어</u>	

과목	Aa 단어	≡ 내용
7	<u>DELETE 명령어</u>	덤프웨 DELETE FROM 테이블명 WHERE 조건절; ex. DELETE FROM 학생 WHERE 이름 = '론 위즐리';
7	<u>UPDATE 명령어</u>	업셋웨 UPDATE 테이블명 SET 컬럼명 = 데이터 WHERE 조건절; ex. UPDATE 학생 SET 기숙사 = '슬리데린' WHERE 이름 = '드레이코 말포이';
7	<u>INSERT 연산자</u>	교집합. 겹치는 데이터만 추출
7	<u>MINUS 연산자</u>	차집합. 첫 번째 쿼리에만 있고, 두 번째 쿼리에만 없는 결과만 추출
7	<u>UNION 연산자</u>	합집합. 중복을 제거 하고 모두 포함
7	<u>집합 연산자</u>	2개 이상의 테이블의 질의의 결과를 합치는 연산자 (검색을 위해 합치는 조인과 다름) 유형: 유유인마
7	<u>UNION ALL 연산자</u>	완전 합집합. 중복까지 포함 함
7	<u>서브 쿼리 유형</u>	1. SELECT - 반드시 단일 행을 리턴, 집계 함수 주로 사용 ex. SELECT (SELECT MAX(가격) FROM 도서 A WHERE A.책번호 = B.책번호 AND 책명 = '마법약') FROM 도서가격 B ; 2. FROM - 동적으로 생성된 테이블처럼 사용 ex. SELECT MAX(가격) FROM 도서가격 A, (SELECT 책번호 FROM 도서 WHERE 책명 = '마법약') B WHERE A.책번호 = B.책번호 ; 3. WHERE ex. WHERE 책번호 IN (SELECT 책번호 FROM 도서 WHERE 책명 = '마법약');
7	<u>교차 조인</u>	조인 조건이 없는 모든 데이터 조합 추출 CROSS JOIN
7	<u>물리적 조인</u>	네소해 1. Nested-Loop Join (중첩 반복 조인) 2. Sort-Merge Join (정렬 합병 조인) 3. Hash Join (해시 조인)
7	<u>셀프 조인</u>	자기 자신에게 별칭을 지정한 후 조인
7	<u>내부 조인</u>	양 테이블의 공통 컬럼의 값이 같을 때 사용 INNER JOIN
7	<u>외부 조인</u>	- 왼쪽 외부 조인 : 왼쪽 테이블 전체 + 오른쪽 테이블 동일 데이터 LEFT OUTER JOIN - 오른쪽 외부 조인 : 오른쪽 테이블 전체 + 왼쪽 테이블 동일 데이터 RIGHT OUTER JOIN - 완전 외부 조인 : 양쪽 모든 데이터 추출 FULL OUTER JOIN
7	<u>논리적 조인</u>	내외교셀 1. 내부 조인 2. 외부 조인 (왼쪽 외부, 오른쪽 외부, 완전 외부) 3. 교차 조인 4. 셀프 조인
7	<u>조인 명령어</u>	프조온웨 ex. SELECT A.번호, B.가격 FROM 상품목록 A JOIN 가격목록 B ON A.번호 = B.번호 WHERE A.번호 IS NOT NULL; // 조인 테이블의 별칭을 FROM에서 지정
7	<u>조인</u>	두 개 이상의 테이블 연결해 데이터를 검색하는 방법
7	<u>INSERT 명령어</u>	인인투벨 INSERT INTO 테이블명(컬럼1, 컬럼2...) VALUES (데이터1, 데이터2...) ex. INSERT INTO 학생(기숙사, 학년, 성명) VALUES ('그리핀도르', 3, '해리 포터');
7	<u>LIKE 패턴</u>	• % (0개 이상의 문자열과 일치) • [] (1개의 문자와 일치) • [^] (1개의 문자와 불일치) • _ (특정 위치의 1개의 문자와 일치) ex) 이름 LIKE '_나%' : 바나나, 가나, 소나기, 김나나나나

과 목	Aa 단어	≡ 내용
7	<u>WHERE 조건절</u>	- 컬럼 BETWEEN 값1 AND 값2 (= 컬럼 ≥ 값1 && 컬럼 ≤ 값2) - 컬럼 IN (값1, 값2...) - 컬럼 NOT IN (값1, 값2...) - 컬럼 LIKE '패턴' - IS NULL - IS NOT NULL - AND / OR / NOT ex) 나이가 20살~29살이면서 성별이 여자 >> WHERE 나이 BETWEEN 20 AND 29 AND 성별 = '여자';
7	<u>SELECT 명령어</u>	셀프웨구해오 1. SELECT [ALL DISTINCT] 컬럼명 [AS 별칭] 2. FROM 테이블 3. WHERE 조건 4. GROUP BY 그룹명 5. HAVING 그룹조건 5. ORDER BY 컬럼명 [ASC DESC];
7	<u>DML 명령어 요약</u>	
7	<u>참조 무결성</u>	외래키 값은 항상 참조되는 테이블의 기본키여야 함
7	<u>참조 무결성을 위한 옵션</u>	1. RESTRICT : 참조 무결성 위배 시 연산 거절 2. CASCADE : 해당 튜플을 참조하는 튜플도 함께 삭제 3. SET NULL : 해당 튜플을 참조하는 튜플의 외래키에 NULL 값 삽입 (단, NOT NULL 제약조건 시 연산 거절)
7	<u>인덱스 삭제</u>	DROP INDEX 인덱스명;
7	<u>인덱스 변경</u>	ALTER INDEX 인덱스명 ON 테이블(컬럼);
7	<u>인덱스 생성</u>	크인인 온테커 CREATE INDEX 인덱스명 ON 테이블(컬럼);
7	<u>뷰 삭제</u>	DROP VIEW 뷰이름;
7	<u>뷰 생성</u>	크뷰뷰 예즈 CREATE [OR REPLACE] VIEW 뷰이름 AS SELECT 이름 FROM 학생 WHERE 성별 = 'F';
7	<u>테이블 삭제</u>	드롭[캐스 리스] 트케 DROP TABLE 테이블명 [CASCADE RESTRICT]; TRUNCATE TABLE 테이블명;
7	<u>테이블 변경</u>	알 에모드 ALTER TABLE 테이블명 ADD 컬럼명 데이터타입 [제약조건]; ALTER TABLE 테이블명 MODIFY 컬럼명 데이터타입 [제약조건]; ALTER TABLE 테이블명 DROP 컬럼명;
7	<u>테이블 생성 제약조건</u>	- PRIMARY KEY - FOREIGN KEY REFERENCES 테이블(컬럼) - NOT NULL - UNIQUE - CHECK (조건 OR 조건) - DEFAULT
7	<u>테이블 생성</u>	크테테 컬타제 CREATE TABLE 테이블명 (컬럼명 데이터타입 [제약조건];)
7	<u>DDL 명령어 요약</u>	
7	<u>파티셔닝 장점</u>	성가백합 1. 성능 향상 (액세스 범위가 줄어드니까) 2. 가용성 향상 (데이터 훼손 가능성이 적으니까) 3. 백업 가능 4. 경합 감소
7	<u>리스트 파티셔닝</u>	값 목록을 기준으로 파티셔닝
7	<u>해시 파티셔닝</u>	해시 함수의 값을 기준으로 파티셔닝 (균등 분할 가능)
7	<u>컴포지트 파티셔닝</u>	2개 이상의 파티셔닝을 결합
7	<u>레인지 파티셔닝</u>	숫자나 날짜와 같은 범위 기준으로 파티셔닝
7	<u>파티셔닝</u>	테이블을 논리적 단위로 쪼개는 것

과 목	Aa 단어	≡ 내용
7	<u>파티셔닝 유형</u>	레해리캠 1. 레인지 파티셔닝 2. 해시 파티셔닝 3. 리스트 파티셔닝 4. 컴포지트 파티셔닝
7	<u>클러스터링</u>	검색 속도 향상을 위해 물리적으로 저장하는 것 - 분포도가 넓을 수록(=좋지 않을 수록) 클러스터링이 적합
7	<u>인덱스 단일 스캔</u> (Index Unique Scan)	↓ ↓ 수직 탐색으로만 스캔
7	<u>인덱스 생략 스캔</u> (Index Skip Scan)	선두 컬럼이 조건 절에 없더라도 인덱스를 활용하는 스캔 (필요없는 부분은 과감히 스킵스킵!)
7	<u>인덱스 전체 스캔</u> (Index Full Scan)	→ → 리프 블록을 처음부터 끝까지 수평 탐색
7	<u>인덱스 범위 스캔</u> (Index Range Scan)	↓ → 루트 블록에서 리프 블록까지 수직 탐색 후, 리프 블록을 수평 탐색
7	<u>인덱스 스캔 방식</u>	범전단생 1. 인덱스 범위 스캔 2. 인덱스 전체 스캔 3. 인덱스 단일 스캔 4. 인덱스 생략 스캔
7	<u>인덱스 종류</u>	비단순함 해결 클 1. 비트맵 인덱스 : 컬럼 개수가 적고, 수정이 적을 수록 좋은 인덱스 (생년월일, 상품번호 등) 2. 단일 인덱스 : 하나의 컬럼으로만 구성된 인덱스 3. 순서 인덱스 : 데이터가 정렬된 순서로 생성되는 인덱스 4. 함수기반 인덱스 : 함수를 적용해 만든 인덱스 5. 해시 인덱스 : 해시 함수를 통해 키 값으로 데이터에 접근하는 인덱스 (튜플 양에 무관하게 접근 비용 동일함) 6. 결합 인덱스 : 두 개 이상의 컬럼으로 구성된 인덱스 7. 클러스터드 인덱스 : PK 기준으로 레코드를 묶어 데이터의 물리적 순서에 따라 생성된 인덱스
7	<u>인덱스 컬럼 선정</u>	- 적정 분포도: 10~15% (분포도: 특정 컬럼 값이 테이블에 평균적으로 분포된 정도) - 수정이 빈번하지 않은 컬럼 - 분포도가 좋은 컬럼 ⇒ 단일 인덱스 - 자주 결합되는 컬럼 ⇒ 결합 인덱스
7	<u>인덱스</u>	DB 시스템에서 빠른 검색을 위한 데이터 구조 - 인덱스가 없으면 Table Full Scan을 하지만, 인덱스가 있으면 Index Range Scan을 하므로 검색 속도가 빠름
7	<u>뷰 속성</u>	- REPLACE : 기존 존재 시 재생성 - FORCE : 기본 테이블 관계 없이 뷰 생성 - NO FORCE : 기본 테이블이 있을 때만 뷰 생성 - WITH CHECK OPTION : 서브 쿼리 내 조건을 만족하는 행만 변경 - WITH READ ONLY : DML(조작어) 불가
7	<u>디그리</u>	애트리뷰트의 개수
7	<u>뷰</u>	데이터의 독립성을 보장하고 조작 연산을 간소화할 수 있는 논리 테이블 - ALTER로 변경 불가 - 자체 인덱스 불가
7	<u>애트리뷰트</u>	=Column.
7	<u>테이블</u>	데이터를 저장하는 공간. (=릴레이션, =엔터티)
7	<u>카디널리티</u>	튜플의 개수

▼ 과 목	Aa 단어	≡ 내용
7	튜플	=Row. =레코드. 한 릴레이션에서 중복되는 튜플은 존재 불가
7	외부 스키마	사용자(개발자) 관점에서의 구조. 사용자 뷰를 나타냄
7	개념 스키마	제약조건, 권한, 보안 등 전체적인 논리 구조
7	내부 스키마	물리적 저장장치 관점에서의 구조. 레코드 형식, 물리적 순서 등
7	스키마	DB의 구조, 제약조건 등의 정보를 담고 있는 구조
7	스키마 3계층	외개내 1. 외부 스키마 2. 개념 스키마 3. 내부 스키마
7	도메인	하나의 속성이 가질 수 있는 원자값의 집합
7	DDL의 대상	도스테뷰인 1. 도메인 2. 스키마 3. 테이블 4. 뷰 5. 인덱스
7	SQL 문법	1. DDL (정의어) 크알드트 2. DML (조작어) 세인업데 3. DCL (제어어) 그리
7	회복 기법	회로체크 1. 로그 기반 회복 기법 1-1) 지연 갱신 회복 기법 : 트랜잭션 완료 전에는 로그에만 기록. 장애 발생 시 로그를 폐기 1-2) 즉각 갱신 회복 기법 : 트랜잭션 갱신 결과를 바로 DB에 반영. 장애 발생 시 로그를 참고하여 되돌림 2. 체크포인트 회복 기법 : 체크포인트 이전으로 복원 3. 그림자 페이징 회복 기법 : 트랜잭션 수행 시 복제본을 생성해 이를 이용해 복구
7	고립화 수준 종류	언커리시 1. Read Uncommitted : 연산 중인 데이터를 다른 트랜잭션이 읽는 것을 허용 2. Read Committed : 연산이 완료되기 전까지 다른 트랜잭션이 읽는 것을 제한 3. Repeatable Read : 선행 트랜잭션이 특정 데이터를 읽을 때, 해당 데이터의 갱신/삭제를 제한 4. Serializable Read : 선행 트랜잭션이 특정 데이터 영역을 읽을 때, 해당 영역 전체의 접근을 제한
7	고립화	무결성을 해치지 않기 위해 잠금을 설정하는 정도
7	다중 버전 동시성 제어(MVCC).	트랜잭션의 타임스탬프와 접근하려는 데이터의 타임스탬프를 비교해, 적절한 버전을 선택하여 접근하도록 하는 기법
7	타임스탬프 순서(Time Stamp Ordering).	트랜잭션이나 데이터에 타임 스탬프를 부여하여, 그 시간에 따라 작업을 수행
7	낙관적 검증	일단 검증 없이 진행 후, 종료한 다음 검증을 수행해 반영
7	로킹	트랜잭션의 순차적 진행을 보장하는 기법 - 로킹 단위 : 한번에 로킹할 수 있는 객체 크기 - 로킹 단위가 작을수록 DB공유도는 증가하지만 로킹 오버헤드 증가
7	병행 제어	다수 사용자 환경에서 일관성 유지를 위해 제어하는 기법
7	트랜잭션의 상태변환	활부완실절 1. 활동상태 (Active) : 트랜잭션 실행 중 2. 부분 완료 상태 (Partially Committed) : 마지막 명령문 실행 상태 3. 완료 상태 (Committed) : 성공적으로 완료 4. 실패 상태 (Failed) : 정상적 실행 불가 5. 철회 상태 (Aborted) : 트랜잭션 취소

과 목	Aa 단어	≡ 내용
7	<u>병행제어 기법</u>	로 낙타다 1. 로킹(Locking) 2. 낙관적 검증 기법(Validation) 3. 타임 스탬프 순서 (Time Stamp Ordering) 4. 다중버전 동시성 제어(MVCC; Multi Version Concurrency Control)
7	<u>병행제어 미보장 문제점</u>	갱신모연: 갱신손실(Lost Update), 현황파악오류(Dirty Read), 모순성 (Inconsistency), 연쇄복귀(Cascading Rollback)
7	<u>TCL</u>	트랜잭션 제어 언어 - 트랜잭션 결과를 허용하거나 취소하기 위한 제어 언어 커를 체 COMMIT, ROLLBACK, CHECKPOINT
7	<u>영속성 (Durability).</u>	성공적으로 완료된 트랜잭션 결과는 DB에 영속적으로 저장되어야 한다 (기법: 회복기법)
7	<u>격리성 (Isolation).</u>	트랜잭션 실행 중 중간 연산 결과에 다른 트랜잭션이 접근할 수 없다 (기법: 고립화 수준)
7	<u>원자성 (Atomicity).</u>	All or Nothing. 연산 전체는 성공 또는 실패여야 하며, 하나라도 실패할 경우 전체가 취소되어야 하는 특성
7	<u>일관성 (Consistency).</u>	트랜잭션이 성공적으로 실행되면, DB는 일관된 상태를 유지해야 하는 특성 (기법: 병행(동시성) 제어)
7	<u>트랜잭션</u>	DB 시스템에서 하나의 논리적 기능을 정상적으로 수행하기 위한 작업 단위 (특성: ACID)
8	<u>쿼츠 크론 표현식</u>	스케줄러에서 배치 수행시간을 설정하기 위한 표현식
8	<u>크론 표현식 예시</u>	0 0 12 * * ? : 매일 12시에 실행 0 * 14 * * ? : 매일 14시부터 15시까지 매 분마다 실행 0 0/5 14,20 * * ? : 매일 14:00~14:55까지 5분마다 실행, 매일 20:00~20:55까지 5분마다 실행 0 0 20 ? * MON-FRI : 매주 월요일부터 금요일 20시에 실행 0 15 10 L * ? : 매달 마지막 날 10시 15분에 실행 0 15 10 ? * 6L 2020-2021 : 2020년부터 2021년까지 매달 마지막 금요일 10시 15분에 실행 0 11 11 1 1 ? : 1월 1일 11시 11분마다 실행 0 0 9 * * 15w : 매달 15일 9시에 실행 (단, 15일이 일요일이면 16일 실행, 토요일이면 14일 실행)
8	<u>배치 스케줄러 종류</u>	스쿼 1) 스프링 배치 : 스프링 프레임워크에서 사용하는 스케줄러 2) 쿼츠 스케줄러 : Job 과 Trigger 를 분리하는 오픈 소스 스케줄러
8	<u>온디맨드 배치</u>	사용자의 요구가 있을 때 실행
8	<u>정기 배치</u>	정해진 시점에 실행
8	<u>이벤트 배치</u>	사전에 정의된 조건이 충족될 때 실행
8	<u>DAO</u>	Data Access Object. 실질적으로 DB에 접근하는 객체
8	<u>MyBatis</u>	자바의 DB 프로그래밍을 돕는 프레임워크
8	<u>DTO</u>	Data Transfer Object. 화면에서 전달받은 데이터를 전송하는 객체
8	<u>VO</u>	Value Object. 간단한 객체
8	<u>자료 결합도</u>	Data. 인터페이스로 전달되는 파라미터로만 상호작용할 때
8	<u>스탬프 결합도</u>	Stamp. 인터페이스로 배열, 객체 등이 전달될 때

▼ 과 목	Aa 단어	≡ 내용
8	<u>제어 결합도</u>	Control. 다른 모듈의 내부 논리를 제어할 때
8	<u>외부 결합도</u>	External. 외부의 데이터 포맷, 인터페이스, 프로토콜을 공유할 때
8	<u>공통 결합도</u>	Common. 공통 데이터(전역 변수)를 공유할 때
8	<u>내부 결합도</u>	Content. 다른 모듈의 내부 변수나 기능을 사용할 때
8	<u>통신적 응집도</u>	Communational. 동일한 입출력을 사용해 각자 다른 기능을 수행할 때
8	<u>기능적 응집도</u>	Functional. 내부 모든 기능이 하나의 목적을 위해 수행할 때
8	<u>순차적 응집도</u>	Sequential. 출력 결과를 다른 요소가 입력 데이터로 사용할 때
8	<u>절차적 응집도</u>	Procedural. 모듈의 기능이 여러 개일 때, 구성요소들이 그 기능을 순차적으로 수행할 때
8	<u>시간적 응집도</u>	Temporal. 특정 시간에 함께 처리되는 사이일 때
8	<u>논리적 응집도</u>	Logical. 유사하거나 특정 형태로 분류되는 요소들이 함께 있는 정도
8	<u>우연적 응집도</u>	Concidental. 서로 다른 상위 모듈에 의해 호출되어 연관이 없는 정도
8	<u>메인 루틴</u>	전체의 개략적인 동작 절차를 표시하며 서브 루틴을 호출하는 루틴
8	<u>서브 루틴</u>	메인 루틴에 의해 호출되는 루틴
8	<u>형상관리 명령어</u>	1. 생성 <code>git init</code> <code>svn import</code> 2. 복제 <code>git clone</code> <code>svn checkout</code> 3. 커밋 <code>git/svn commit</code> 4. 변경내용 확인 <code>git/svn diff</code> 5. 추가 <code>git/svn add</code> 6. 이동 <code>git/svn mv</code> 7. 삭제 <code>git/svn rm</code> 8. 브랜치 생성 <code>git branch</code> <code>svn copy</code> 9. 병합 <code>git/svn merge</code> 10. 원격 저장소 반영 <code>git push</code> <code>svn commit</code> 11. 설정 확인 <code>git config</code> <code>svn info</code>
8	<u>형상관리 도구별 특징</u>	1. RCS: 공유폴더 방식 (잠금 처리 후 1명만 수정) 2. CVS: 클라/서버 방식 (동시 접근 가능) 3. SVN: 클라/서버 방식 4. Clear Case: 복수 클라/복수 서버 (필요시 서버 증설 가능) 5. Bitkeeper: 분산 저장소 방식 6. Git: 분산 저장소 방식
8	<u>베이스라인</u>	개발 산출물 변화를 통제하는 시점
8	<u>형상관리 통제 위원회(CCB)</u>	형상관리 방침을 정하고 관리하는 조직
8	<u>웹 애플리케이션 서버</u>	=WAS. 동적 콘텐츠(JSP, 서블릿)를 처리하는 서버 (ex. Tomcat, Weblogic 등)
8	<u>클라이언트 프로그램</u>	사용자가 설치해서 커뮤니케이션하는 프로그램
8	<u>웹서버</u>	정적 콘텐츠를 처리하는 서버 (ex. 아파치 웹 서버, 구글 웹 서버 등)
8	<u>크론 표현식</u>	초분시일월요일 1=SUN, 7=SAT * : 모든 수 ? : 미사용 - : 기간 설정 , : 특정 기간 설정 / : 시작시간과 반복간격 L : 마지막 기간 W : 가까운 평일 # : 몇 번째 주, 요일
8	<u>배치 스케줄러</u>	배치(일괄 처리)를 위해 반복적인 작업을 지원하는 도구

▼ 과 목	Aa 단어	≡ 내용
8	<u>배치 프로그램 유형</u>	이온정 1) 이벤트 배치 2) 온디맨드 배치 3) 정기 배치
8	<u>배치 프로그램</u>	사용자와 상호작용 없이, 일련의 작업들을 묶어 일괄적으로 처리하는 방법
8	<u>서버 프로그램 구현 방식</u>	디스다씨클 DTO/VO ⇒ SQL ⇒ DAO ⇒ Service Class ⇒ Controller Class
8	<u>응집도</u>	모듈 내부의 구성요소 간 연관 정도
8	<u>응집도 유형</u>	우논시절통순기 1) 우연적 2) 논리적 3) 시간적 4) 절차적 5) 통신적 6) 순차적 7) 기능적
8	<u>루틴</u>	특정 동작을 수행하는 일련의 코드
8	<u>팬아웃</u>	어떤 모듈에 의해 제어(호출)되는 모듈의 수
8	<u>팬인</u>	어떤 모듈을 제어(호출)하는 모듈의 수
8	<u>결합도</u>	모듈과 다른 모듈 간 상호 의존도
8	<u>결합도 유형</u>	내공외제스자 1) 내부(내용) 2) 공통 3) 외부 4) 제어 5) 스탬프 6) 자료
8	<u>모듈</u>	그 자체로 하나의 완전한 기능을 수행할 수 있는 독립된 실체 (기능 단위로 분해, 추상화되어 재사용 가능한 단위)
8	<u>모듈화</u>	성능 향상, 유지 관리 등을 위해 기능 단위 모듈로 분해하는 기법 (모듈 단위로 설계하는 기법)
8	<u>형상관리 도구 유형</u>	공클분 1. 공유 폴더 방식 (개발이 완료된 파일을 공유 폴더에 복사) 2. 클라이언트/서버 방식 (중앙에 버전관리 시스템이 동작) 3. 분산 저장소 방식 (로컬 저장소/원격 저장소 분산)
8	<u>형상관리 절차</u>	식통감기 1. 형상 식별 (관리할 대상 식별 후 번호 부여) 2. 형상 통제 (형상통제위원회 운영, 베이스라인 관리) 3. 형상 감사 (무결성 평가) 4. 형상 기록
8	<u>개발 도구 분류</u>	빌구테형 : 빌드 도구, 구현 도구, 테스트 도구, 형상관리 도구
8	<u>형상관리(CM)</u>	소프트웨어 개발 과정에서 발생하는 변경사항을 관리하는 것
8	<u>DBMS</u>	사용자-DB 사이에서 데이터 관리를 하는 소프트웨어 ex) Oracle, MySQL
8	<u>미들웨어</u>	컴퓨터-컴퓨터 간 연결 및 연결 관리를 돕는 소프트웨어 ex) Weblogic, Jeus, Tomcat
8	<u>운영체제</u>	하드웨어를 사용자가 편리하게 사용하기 위한 소프트웨어 ex) Windows, Unix, Linux
9	<u>Stacheldraht</u>	DDos의 에이전트 역할을 하는 도구
9	<u>TFN</u>	UDP 플러딩, SYN 플러딩, 스머프 등등 여러 DDos 공격이 가능한 도구
9	<u>Trinoo</u>	UDP 플러딩 공격 도구

과 목	Aa 단어	≡ 내용
9	<u>DDos 공격 구성요소</u>	1) 공격자 : 해커 컴퓨터 2) 마스터 (Master) : 공격자의 명령을 받고 에이전트를 관리하는 시스템 3) 핸들러 (Handler) : 마스터 시스템의 프로그램 4) 에이전트 (Agent) : 직접 공격하는 시스템 5) 데몬 (Daemon) : 에이전트 시스템의 프로그램
9	<u>CVSS (Common Vulnerability Scoring System)</u>	👤 공통 취약점(CV)에 등급(S)을 매긴 시스템(S). 위험도 계산 가능
9	<u>CVE (Common Vulnerabilities and Exposures)</u>	👤 소프트웨어의 공통 취약점(CV)을 식별화(E)한 것 (CVE-연도-순서)
9	<u>CWE (Common Weakness Enumeration)</u>	👤 소프트웨어의 공통 약점(CW)을 식별화(E)한 것
9	<u>C-TAS (사이버 위협 정보 분석 공유 시스템)</u>	👤 사이버 위협정보를 체계적으로 수집해 관계 기관과 자동화된 정보공유를 할 수 있는 시스템 (KISA 주관)
9	<u>CC (Common Criteria)</u>	👤 컴퓨터 보안을 위한 국제 평가 기준
9	<u>워터 마킹 (Water Marking)</u>	👤 디지털 콘텐츠에 저작자 정보를 삽입해, 불법 복제 시 원소유자를 증명하는 기술
9	<u>CPTED</u>	👤 범죄 예방 환경 설계 (Crime Privent Through Environment Design) 학문 간 연계를 통해 범죄를 최소화할 수 있는 환경을 설계하는 전략
9	<u>핑거프린팅 (Finger Printing)</u>	👤 저작권 정보와 구매자 정보를 콘텐츠에 삽입해 불법 배포자를 추적할 수 있는 기술
9	<u>OWASP Top 10</u>	👤 웹 애플리케이션의 10가지 보안 취약점에 대한 방안을 제공하는 가이드
9	<u>허니팟 (HoneyPot)</u>	👤 일부러 허술하게 만들어 해커에게 노출하는 유인시스템
9	<u>스펙터 (Specter)</u>	🐱 실패한 분기 예측으로 메모리 영역을 훑쳐보는 취약점
9	<u>멜트다운 (Meltdown)</u>	🐱 인텔 아키텍처의 버그를 이용해 시스템 메모리에 접근하는 취약점
9	<u>윈드토키 (WindTalker)</u>	🐱 터치, 타이핑 등의 패턴을 스니핑하여 해킹
9	<u>토르 네트워크 (Tor Network)</u>	🐱 암호화 기법으로 데이터를 전송해 익명으로 사용 가능한 네트워크
9	<u>크리덴셜 스템핑 (Credential Stuffing)</u>	🐱 다른 곳에서 유출된 로그인 정보를 다른 곳에 무작위 대입하는 공격
9	<u>스턱스넷 공격 (Stuxnet)</u>	🐱 독일 지멘스사의 SCADA 시스템을 목표로 제작된 악성코드 (주요 산업 기반 시설의 제어 시스템에 침투하는 공격)
9	<u>익스플로잇 (Exploit)</u>	🐱 SW/HW의 버그나 취약점을 악용해 공격하는 행위

과 목	Aa 단어	≡ 내용
9	<u>리버스 셸 공격 (Reverse Shell)</u>	🐱 타깃 서버(피해자)가 클라이언트로 접속하게 하고, 클라이언트에서 서버의 셸을 획득하는 공격
9	<u>디렉토리 리스팅 취약점 (Directory Listing)</u>	🐱 웹 서버의 인덱싱 기능이 활성화된 경우, 서버 내 모든 디렉토리를 볼 수 있는 취약점
9	<u>포트 스캐닝 (Port Scanning)</u>	🐱 침입 전 어떤 포트가 활성화되어 있는지 확인하는 기법
9	<u>DNS 스푸핑</u>	🐱 DNS 서버 캐시를 조작해 의도치 않은 주소로 접속하게 하는 공격 (=DNS 캐시 포이즈닝)
9	<u>MIMT (Man in the Middle)</u>	🐱 통신 연결 중간에 침입해 통신 내용을 도청하는 공격
9	<u>스캠 공격 (SCAM)</u>	🐱 기업 이메일을 도용해 거래 대금을 가로채는 공격
9	<u>크라이웨어 (Crimeware)</u>	🐱 금융·인증 정보를 탈취해 금전적 이익을 취하는 악성 코드
9	<u>IoT-SSDP</u>	🐱 SSDP(단순 서비스 검색 프로토콜)의 특성을 이용해, IoT 디바이스를 좀비 PC로 이용해 DDoS 공격
9	<u>하트 블리드 (Heart Bleed)</u>	🐱 하트비트(암호화 라이브러리)의 확장 모듈 취약점을 이용해 데이터를 탈취하는 공격
9	<u>워터링 홀 (Watering Hole)</u>	🐱 특정인을 표적으로 삼아, 특정인이 자주 방문하는 웹사이트에 악성코드를 심어 공격
9	<u>드라이브 바이 다운 로드 (Drive By Download)</u>	🐱 악성 스크립트를 웹 서버에 설치 후, 사용자를 멀웨어 서버로 연결해 감염시키는 공격
9	<u>부 채널 공격 (Side Channel Attack)</u>	🐱 전력 소비와 같은 물리적 특성을 측정해 비밀 정보를 알아내는 공격
9	<u>Cold Site</u>	데이터만 원격지에 보관하고, 재해 시 이를 근간으로 복구 (RTO는 수주~수개월)
9	<u>Warm Site</u>	중요성이 높은 자원만 주 센터와 동일한 수준으로 보유 (RTO는 수일~수주)
9	<u>Hot Site</u>	주 센터와 동일한 수준의 자원을 대기 상태로 보유하며 데이터를 최신 상태로 유지 (RTO는 4시간 이내)
9	<u>Mirror Site</u>	주 센터 & 복구센터 모두 운영 상태. (RTO는 0)
9	<u>DRP</u>	재해 복구 계획 (Disaster Recovery Plan) - 재해로 장기간 운영이 불가한 경우를 대비한 계획
9	<u>DRS의 유형</u>	1) Mirror Site 2) Hot Site 3) Warm Site 4) Cold Site
9	<u>DRS</u>	재해 복구 시스템 (Disaster Recovery System) - DRP를 위한 관리체제
9	<u>RPO</u>	복구 지점 목표 (Recovery Point Object) - 재해 시 업무중단 시점 ~ 정상가동까지 허용하는 손실

과 목	Aa 단어	≡ 내용
9	<u>RTO</u>	복구 시간 목표 (Recovery Time Object) - 재해 시 업무중단 시점 ~ 업무복구 시점까지 걸린 시간
9	<u>BIA</u>	비즈니스 영향 평가 (Business Impact Analysis) - 장애나 재해에 따른 영향도 조사
9	<u>BCP</u>	비즈니스 연속성 계획 (Business Continuity Plan) - 위기관리를 기반으로 비상 시 비즈니스 연속성을 보장하는 체계 - BIA가 선행되어야 함
9	<u>DRM</u>	Digital Right Management; 디지털 저작물에 암호를 걸어, 권한이 없는 사용자의 사용을 막는 솔루션
9	<u>보안 취약점 분석 절차</u>	자진제진결 1. 자산 조사 2. 진단 대상 선정 (전수조사 vs. 샘플링) 3. 제약사항 확인 4. 진단 수행 (기술 진단, 인터뷰, 내부 실사 등) 5. 결과 보고서 작성
9	<u>DLP</u>	Data Loss Prevention; 주요 자료가 외부로 유출되는 것을 차단하는 솔루션
9	<u>Anti-Spam Solution</u>	메일 서버 앞단에 바이러스 검사, 정보 유출 방지 등의 기능을 제공하는 솔루션
9	<u>Secure OS</u>	OS 커널에 보안 기능을 추가한 솔루션
9	<u>무선 침입 방지 시스템 (WIPS)</u>	비인가 무선 단말기의 접속을 차단하는 시스템
9	<u>침입 방지 시스템 (IPS)</u>	공격 및 침입을 실시간으로 차단하는 시스템
9	<u>통합 보안 시스템 (UTM)</u>	방화벽, IDS, IPS, VPN 등 다양한 보안 장비 기능을 통합 제공하는 시스템
9	<u>가상사설망 (VPN)</u>	공중망을 사용할 때 마치 전용망을 사용하는 것과 같은 보안 효과를 주는 솔루션
9	<u>침입 탐지 시스템 (IDS)</u>	비인가 사용자의 침입을 실시간으로 탐지하는 시스템
9	<u>방화벽 (FireWall)</u>	내부·외부 트래픽을 모니터링하여 접근을 허용/차단하는 시스템
9	<u>웹 방화벽 (WAF)</u>	웹 애플리케이션에 특화되어, XSS나 SQL Injection 등을 탐지하고 차단하는 시스템
9	<u>네트워크 접근 제어 (NAC)</u>	내부 네트워크에 접속을 시도할 때 통제하는 솔루션
9	<u>messages</u>	운영에 대한 전반적 메시지 명령어: (그냥 텍스트 파일임)
9	<u>secure</u>	보안 및 인증과 관련된 로그 명령어: (그냥 텍스트 파일임)
9	<u>sulog</u>	su(switch user) 명령어 결과 정보 명령어: (그냥 텍스트 파일임)
9	<u>xferlog</u>	FTP 전송 기록 명령어: (그냥 텍스트 파일임)
9	<u>btmpt(x)</u>	로그인에 실패한 정보 명령어: lastb
9	<u>lastlog</u>	사용자별 최근 로그인 시간 명령어: lastlog
9	<u>acct/pacct</u>	사용자별 실행한 모든 명령어 명령어: lastcomm

과목	Aa 단어	내용
9	<u>utmp(x)</u>	현재 로그인한 사용자 정보 명령어: <code>who</code> , <code>w</code> , <code>users</code> , <code>finger</code>
9	<u>wtmp(x)</u>	로그인, 로그아웃, shutdown, reboot 정보 명령어: <code>last</code>
9	<u>API 오용 취약점</u>	DNS Lookup에 의존하지 않기 (DNS 엔트리를 속일 수 있음) 보안에 취약한 함수 사용하지 않기 널 매개변수 검사하기 (자바에서 매개변수가 NULL이면 반환 오류 있을 수 있음)
9	<u>캡슐화 취약점</u>	디버그 코드는 꼭 제거 민감한 데이터를 가진 클래스 사용에 주의
9	<u>코드 오류 취약점</u>	NULL이 될 수 있는 레퍼런스는 참조 전 NULL 값인지 검사 정수를 문자로 변환할 때, 잘려나가지 않도록 크기 확인 자원 사용 후에는 반드시 해제 변수 선언 시 초기화 (이전에 사용한 내용이 남지 않도록)
9	<u>시간 및 상태 취약점</u>	세션 통제 취약점 (세션 정보에 읽고 쓰기가 가능한 변수 미사용) 병렬 시스템 (공유 자원의 접근 직렬화, 블록문 내에서만 재귀함수 호출)
9	<u>에러 처리 취약점</u>	취약한 패스워드 조건 미사용 오류 메시지에 정보 노출 하지 않음 예외 처리 구문 작성
9	<u>Blind SQL Injection</u>	쿼리 결과의 참/거짓을 통해 공격
9	<u>Error-Based SQL Injection</u>	에러값을 기반으로 한 단계씩 점진적으로 정보를 캐내는 공격
9	<u>Mass SQL Injection</u>	한 번의 공격으로 대량의 DB값을 변조하는 공격
9	<u>Stored Procedure SQL Injection</u>	저장 프로시저를 이용해 공격
9	<u>Form SQL Injection</u>	HTML Form 기반 인증의 취약점을 이용한 공격
9	<u>Union SQL Injection</u>	UNION 연산자를 이용해 쿼리 결과를 결합해 공격
9	<u>SQL Injection</u>	악의적인 SQL 구문을 삽입·실행시켜 DB 정보를 탈취하거나 조작하는 공격
9	<u>CSRF</u>	=Cross Site Request Forgery (사이트 간 요청 위조) 사용자가 자신의 의지와 무관하게, 공격자가 의도한 행위를 요청하게 하는 공격
9	<u>Reflected XSS</u>	악성 URL 클릭 시 공격 스크립트가 반사되는 기법
9	<u>DOM XSS</u>	DOM 기반 XSS 취약점이 있는 브라우저를 대상으로 한 기법
9	<u>Stored XSS</u>	악성 스크립트가 포함된 웹페이지를 읽을 때 감염되는 기법
9	<u>XSS</u>	검증되지 않은 입력데이터가 포함된 웹페이지를 열람할 때, 웹페이지에 포함된 부적절한 스크립트가 실행되는 공격
9	<u>시큐어 코딩 가이드</u>	입보시에코캡아 1. 입력데이터 검증 및 표현 2. 보안 기능 3. 시간 및 상태 4. 에러 처리 5. 코드 오류 6. 캡슐화 7. API 오용
9	<u>OWASP CLASP</u>	개역평구취 개념 관점, 역할 관점, 평가 관점, 구현 관점, 취약성 관점 등의 프로세스로 구성된 프레임워크. 이미 운영 중인 시스템에 적용이 쉬움. OWASP; Open Web Application Security Project CLASP; Comprehensive Lightweight Application Security Process

과목	Aa 단어	내용
9	<u>MS SDL</u>	=MS Secure Development Lifecycle. 마소가 자사 SW개발에 의무 적용시킨 프레임워크. 동일제품을 pre-SDL, post-SDL 버전으로 나눠 테스트
9	<u>Seven TouchPoints</u>	검증된 모범 보안 사례를 SDLC에 통합한 방법론
9	<u>Open SAMM</u>	확대가 가능한 개방형 프레임워크. 설계 리뷰, 코드 리뷰, 보안 테스트 3개를 주요 검증 활동으로 함 * 비용평가모델인 SAAM과는 다르다! (SAMM=소프트웨어 보증 성숙도 모델)
9	<u>BSIMM</u>	보안 활동의 성숙도 수준을 측정하는 개발 프레임워크
9	<u>개인정보보호 관련 법령</u>	개방신 1. 개인정보보호법 (주여운외 분도유변해) - 주민등록번호, 여권번호, 운전면허번호, 외국인등록번호와 같은 고유 식별 번호는 - 분실, 도난, 유출, 변조, 훼손에 주의해야 함 2. 정보통신망법 3. 신용정보법
9	<u>SSL/TLS</u>	전송계층(4계층)과 응용계층(7계층) 사이에서 안전한 데이터 전송을 보장하는 보안 프로토콜 구성요소 카호르 CAHHR 1. Change Cipher Spec Protocol : 협상된 Cipher Spec을 상대방에게 알리는 프로토콜 2. Alert Protocol : 경고 메시지 전달 3. Heartbeat Protocol : 클라/서버가 정상 상태인지 확인 4. Handshake Protocol : 클라/서버가 서로 인증하고 암호화 키를 협상 5. Record Protocol : 협상된 Cipher Spec
9	<u>S-HTTP</u>	클라/서버 간 메시지를 암호화하는 보안 기술 (HTTP를 사용한 경우에만 가능)
9	<u>IPSec</u>	네트워크 계층(3계층)에서 사용하는 보안 프로토콜. 인증 헤더(AH) + 암호화(ESP)를 이용 - AH(인증) 프로토콜: MAC를 통해 인증 제공 - ESP(암호화) 프로토콜 : MAC+암호화를 통해 인증+기밀성 제공 - IKE(키관리) 프로토콜 : Key를 주고받는 알고리즘
9	<u>HAS-160</u>	해시 암호화 국내 표준 디지털 서명 알고리즘(KCDSA)를 위해 개발된 알고리즘 (MD5의 장점 + SHA-1의 장점)
9	<u>SHA-256/384/512</u>	해시 암호화 256비트의 해시값을 생성하는 함수 (128+128=256, 256+128=384, 384+128=512)로 출력 길이를 늘린 해시 알고리즘
9	<u>SHA-1</u>	해시 암호화 NSA에서 미 정부 표준으로 지정. DSA(디지털 서명 알고리즘)에서 사용
9	<u>MD5</u>	해시 암호화 MD4를 개선한 알고리즘
9	<u>ElGamal</u>	비대칭키 이산대수의 어려움을 근거로 함. 전자서명에 사용 가능
9	<u>ECC</u>	비대칭키 타원 곡선 암호. RSA의 대안
9	<u>RSA</u>	비대칭키 수학교수 3명의 앞글자를 뺐음 (리베스트, 샤미르, 아들만). 소인수 분해 문제의 어려움을 근거로 함
9	<u>디피-헬만</u>	비대칭키 최초의 비밀키 교환 프로토콜. 이산대수 계산의 어려움을 근거로 함
9	<u>LFSR</u>	대칭키(스트림) 선형 되먹임 시프트 레지스터
9	<u>RC4</u>	대칭키(스트림) 셔플링 기법을 이용해 평문과 XOR 연산해 암호화
9	<u>IDEA</u>	대칭키(블록) DES를 개체하기 위해 개발

과목	Aa 단어	내용
9	<u>ARIA</u>	대칭키(블록) 학계(Academy) + 연구기관(Research Institute) + 정부(Agency) 국정원과 산학연구협회가 개발함
9	<u>SEED</u>	대칭키(블록) KISA가 개발. 16라운드를 거쳐 128비트 블록으로 암호화
9	<u>AES</u>	대칭키(블록) =Advanced Encryption Standard. 3DES의 문제점을 극복하기 위해 개발. 라운드 수는 10, 12, 14로 분류되고, 한 라운드는 빼고(SubBytes)+이동하고(ShiftRows)+섞고(MixColumns)+더하는(AddRoundkey) 4계층으로 구성됨
9	<u>DES</u>	대칭키(블록) 미 연방표준국(NIST)에서 발표. 키 길이는 56bit, 블록크기는 64bit
9	<u>MDC</u>	Modification Detection Code 키를 사용하지 않는 변경 감지 코드로 무결성을 보장하는 암호 알고리즘
9	<u>MAC</u>	Message Authentication Code 키를 사용하는 메시지 인증 코드로 무결성을 보장하는 암호 알고리즘
9	<u>스트림 암호</u>	매우 긴 주기의 난수열을 발생시켜 암호화하는 대칭 키 암호 방식
9	<u>일방향 암호</u>	임의의 길이의 정보를 입력받아 고정된 길이의 암호문을 출력하는 암호 방식 (복호화 불가)
9	<u>블록 암호</u>	고정 길이의 블록을 암호화하는 대칭 키 암호 방식
9	<u>RBAC</u>	- 접근 결정 : 역할 - 권한 부여 : 중앙관리자
9	<u>MAC</u>	- 접근 결정 : 권한 (등급) - 권한 부여 : 시스템
9	<u>DAC</u>	- 접근 결정 : 신분 - 권한 부여 : 데이터 소유자 * ACL (Access Control List) 로 자원에 대한 권한 부여
9	<u>서버 접근 통제 유형</u>	대책알백 1. DAC (임의적 접근 통제) 2. MAC (강제적 접근 통제) 3. RBAC (역할 기반 접근 통제)
9	<u>인증 기술의 유형</u>	지소생특 1. 지식기반 ex. ID/PW 2. 소지기반 ex. 공인인증서 3. 생체기반 ex. 지문, 홍채 4. 특징기반 ex. 서명, 발걸음
9	<u>접근 통제 유형</u>	식인인책 1. 식별 : "내가 바로 나나다!" (주체가 객체에게 정보 제공) 2. 인증 : "당신이 나나로군요" (객체가 주체의 신원을 인정) 3. 인가 : "들어가십시오~" (인증된 주체에게 접근을 허용) 4. 책임추적성 : "흠.. 나나님이 이런 걸 보고 있군요" (주체의 접근과 행동을 추적 및 기록)
9	<u>Ping</u>	 접속하려는 원격 호스트가 정상 운영 중인지 확인하는 명령어
9	<u>난독화</u>	 코드의 가독성을 낮춰 역공학에 대비
9	<u>Tripwire</u>	 백도어가 생기거나 설정 파일 변화가 있을 때 이를 감지할 수 있게 돕는 도구
9	<u>사이버 킬체인</u>	 공격형 방위 시스템. APT 공격의 방어 모델
9	<u>Tcpdump</u>	 스니핑 도구 (패킷 내용을 출력하는 프로그램)
9	<u>랜섬웨어</u>	 암호화 후 복호화를 위해 돈을 요구하는 악성 소프트웨어

과 목	Aa 단어	≡ 내용
9	<u>이블 트윈 공격</u>	🐱 합법적인 Wifi 제공자처럼 행세하며 연결된 사용자 정보를 탈취하는 공격
9	<u>제로데이 공격</u>	🐱 보안 취약점이 공표되기 전 신속히 이뤄지는 공격
9	<u>웜</u>	🐱 스스로를 복제하여 전파하는 악성 프로그램
9	<u>악성 봇</u>	🐱 해커의 명령에 의해 원격으로 제어되는 프로그램 (DDos 등에 악용)
9	<u>봇넷</u>	🐱 악성 프로그램에 감염된 컴퓨터들이 네트워크로 연결된 형태
9	<u>APT 공격</u>	🐱 특정 대상을 목표로 한 지능적/지속적인 공격
9	<u>큐싱</u>	🐱 QR코드+피싱. QR을 이용한 피싱 공격
9	<u>공급망 공격</u>	🐱 SW 개발사의 코드를 수정하거나 배포 서버에 접근해 파일을 변경하는 공격
9	<u>스미싱</u>	🐱 SMS+피싱. 문자메시지를 통해 개인정보를 탈취하거나 소셜결제를 유도하는 공격
9	<u>스피어피싱</u>	🐱 특정 대상 선정 후, 일반적인 메일로 위장한 메일을 발송해 개인정보를 탈취하는 공격
9	<u>힙 버퍼 오버플로우 공격</u>	힙 영역(사용자가 직접 관리 가능한 메모리 영역)의 버퍼에 오버플로우를 일으켜서 데이터를 오염시키는 공격 기법
9	<u>스택 버퍼 오버플로우 공격</u>	스택 영역(지역변수, 매개변수가 저장되는 영역)의 버퍼에 오버플로우를 일으켜서 복귀주소를 바꾸는 공격 기법
9	<u>레인보우 테이블 공격</u>	크래킹하려는 해시값을 테이블에서 검색하는 공격
9	<u>패스워드 하이브리드 공격</u>	사전+무차별을 결합하여 공격
9	<u>무차별 크래킹</u>	PW로 사용될 수 있는 문자를 무작위로 대입하는 공격
9	<u>사전 크래킹</u>	ID/PW가 될 가능성이 있는 단어를 대입하는 공격
9	<u>대칭 키 암호화 종류</u>	- 블록 암호 (DES, 3DES, AES, SEED, ARIA, IDEA) - 스트림 암호 (LFSR, RC4)
9	<u>비대칭 키 암호화 종류</u>	디피-헬만, RSA, ECC, Elgamal
9	<u>암호화 알고리즘 방식</u>	대비해 1. 대칭 키 암호 방식 2. 비대칭 키 암호 방식 3. 해시 암호 방식
9	<u>레이스 컨디션 공격</u>	🐱 프로세스가 임시 파일을 만들 때, 실행 중에 끼어들어 임시파일을 심볼릭 링크하는 공격
9	<u>백도어</u>	🐱 정상적인 인증절차를 우회하는 기법
9	<u>루트킷</u>	🐱 불법 해킹에 사용되는 기능을 제공하는 프로그램 모음
9	<u>ASLR 활용</u>	👤 Address Space Layout Randomization; 주소 공간 배치 난수화 주소 공간 배치를 난수화하여, 실행 시마다 메모리 주소를 변경시키는 것

과 목	Aa 단어	≡ 내용
9	<u>키로거 공격</u>	🐱 키보드 움직임을 탐지해 주요정보를 빼가는 공격
9	<u>포맷 스트링 공격</u>	🐱 포맷 스트링을 인자로 하는 함수의 취약점을 이용한 공격
9	<u>스택실드 활용</u>	👤 함수 시작 시 복귀주소를 특수 스택에 저장해 두고, 함수 종료시 스택 값을 비교해 다를 경우 오버플로우로 간주하고 중단
9	<u>스택가드 활용</u>	👤 카나리(무결성 체크용 값)를 미리 삽입해두고, 버퍼 오버플로우 발생 시 카나리값을 체크해 변한 경우 복귀 주소 호출하지 않음
9	<u>버퍼 오버플로우 공격 대응</u>	1) 스택가드 2) 스택실드 3) ASLR 4) 안전한 함수 사용 : 버퍼오버플로우에 취약한 scanf() 대신 fscanf() 등을 사용 5) 실행 제한 : 스택에서의 쓰기 권한 제한 등
9	<u>버퍼 오버플로우 공격</u>	메모리의 버퍼 크기를 초과하는 데이터를 입력해 프로세스 흐름을 변경시키는 공격 기법
9	<u>트로이 목마</u>	겉으로는 정상적이나 실행하면 악성코드가 실행되는 프로그램
9	<u>ARP 스누핑</u>	MAC 주소를 위장하여 패킷을 스니핑하는 공격
9	<u>IP 스누핑</u>	IP를 위조하여 인증된 시스템인 것처럼 IP를 위조하여 목표 시스템의 정보를 빼내는 공격
9	<u>ICMP Redirect 공격</u>	ICMP Redirect 메시지를 공격자가 원하는 형태로 위조해 패킷을 스니핑하는 공격
9	<u>네트워크 스캐너 / 스니퍼</u>	공격자가 취약점을 탐색하는 도구
9	<u>패스워드 크래킹</u>	사무패레 1) 사전 크래킹 Dictionary Cracking 2) 무차별 크래킹 Brute Force Cracking 3) 패스워드 하이브리드 공격 Password Hybrid Attack 4) 레인보우 테이블 공격 Rainbow Table Attack
9	<u>스니핑</u>	공격대상에 직접 공격을 가하지 않고, 몰래 정보를 들여다보는 수동적인 공격 기법
9	<u>DoS 공격</u>	시스템의 자원을 고갈시켜 서비스 거부를 유발하는 공격
9	<u>무결성</u>	정당한 방법으로만 데이터를 변경할 수 있으며, 데이터의 정확성을 보장하는 특성
9	<u>기밀성</u>	인가되지 않은 접근에 정보 노출을 차단하는 특성
9	<u>SW개발 보안 용어</u>	자위취위 자산, 위협, 취약점, 위험
9	<u>가용성</u>	권한을 가지고 있으면 서비스를 지속해서 사용할 수 있도록 하는 특성
9	<u>Hash DoS</u>	웹 서버의 해시 테이블에 해시 충돌을 일으켜 자원을 소모시키는 공격
9	<u>Hulk DoS</u>	공격자가 공격대상의 URL을 계속 변경하면서(=차단 정책 우회) 다량의 GET 요청을 보내는 공격
9	<u>Slow HTTP Post DoS</u>	=RUDY. 헤더의 Content-Length를 아주 크게 만들고, 데이터를 아주 소량으로 보내 연결을 유지하게 하는 공격
9	<u>HTTP GET 플러딩</u>	과도한 GET 메시지를 통해 과부하시키는 공격 (캐싱서버가 아닌 웹 서버가 직접 처리하도록 유도)

과 목	Aa 단어	≡ 내용
9	<u>Slow HTTP Header DoS</u>	=Slowloris. 헤더 정보를 조작하여, 웹 서버가 온전한 헤더정보가 올 때까지 기다리게 하는 공격
9	<u>Slow HTTP Read Dos</u>	TCP 윈도우 크기와 데이터 처리율을 감소시킨 뒤, 다량의 HTTP 요청을 보내는 공격
9	<u>퐁크/보잉크</u>	IP 패킷의 재전송, 재조합 과정에서 오류를 발생시키는 공격 (퐁크: 같은 시퀀스 번호 / 보잉크: 시퀀스 번호에 빈 공간)
9	<u>랜드 어택</u>	출발지 IP와 도착지 IP를 같은 주소로 만들어, 자기 자신에게 응답을 보내도록 하는 공격
9	<u>티어 드롭</u>	조작된 IP 패킷 조각을 보내 재조립 과정에서 오류를 발생시키는 공격
9	<u>스머프/스머핑</u>	출발지 IP를 공격 대상의 IP로 변조하여, 브로드캐스팅으로 ICMP Echo 패킷 요청을 보내 과부하시키는 공격
9	<u>죽음의 핑</u>	ICMP 패킷(핑)을 아주 크게 만들어 과부하시키는 공격
9	<u>DDos 공격</u>	여러 대의 공격자를 분산 배치 후 동시에 동작시키는 DoS 공격 (대역폭 소진 공격 & 서비스 마비 공격)
9	<u>SYN 플러딩</u>	ACK를 발송하지 않고 SYN 패킷만 보내 점유하여 자원을 고갈시키는 공격
9	<u>UDP 플러딩</u>	대량의 UDP 패키지를 임의의 포트번호로 전송하지만, 응답 메시지는 공격자에 전달되지 않아 자원을 고갈시키는 공격
9	<u>DRDos 공격</u>	공격대상이 반사 서버로부터 다량의 응답을 받도록 하는 공격
9	<u>SW개발 보안 3대 요소</u>	기밀성, 무결성, 가용성
10	<u>소프트웨어 결함</u>	- Error/오류: 사람에 의해 생성됨, Defect(결함)의 원인 - Defect / Bug : Error로 인해 포함된 결함. 제거되지 않으면 Failure나 Problem이 됨 - Failure / Problem : Defect가 실행될 때 발생하는 현상
10	<u>테스트 커버리지</u>	테스트의 수행 정도를 나타내는 값 기라코 1. 기능 기반 커버리지 : 전체 기능을 모수로 측정 2. 라인 커버리지 : 소스 코드 라인 수를 모수로 측정 3. 코드 커버리지 : 구문(코드) 자체가 얼마나 테스트됐는가를 측정 * 커버리지가 함은 대부분 코드 커버리지를 가리킴
10	<u>리팩토링</u>	기능을 변경하지 않고 내부 코드를 보완해 유지보수성을 향상시키는 것 (목적: 유지보수성 향상, 품질 향상, 생산성 향상, 유연한 시스템)
10	<u>클린 코드</u>	잘 작성되어 가독성 높고 단순한 코드
10	<u>클린 코드 작성 원칙</u>	가단의 중추 1. 가독성 2. 단순성 3. 의존성 최소화 4. 중복 제거 5. 추상화
10	<u>코드 품질 분석 도구</u>	정적 분석 도구 - pmd, cppcheck, SonarQube, checkstyle, cobertuna 동적 분석 도구 - Avalanche, Valgrind
10	<u>스파게티 코드</u>	동작은 하지만 내부 코드가 복잡하여 파악이 어려운 코드
10	<u>베드 코드 유형</u>	오문이 결집 1. 오염 : 비즈니스 기능을 수행하지 못하는 컴포넌트가 많음 2. 문서 부족 : 코드와 문서 불일치 3. 의미없는 이름 4. 높은 결함도 5. 아키텍처 침식 : 아키텍처가 변형되어 품질 저하

과 목	Aa 단어	≡ 내용
10	<u>외계인 코드</u>	아주 오래되거나 참고할 개발자/문서가 없어 유지보수가 힘든 코드
10	<u>베드 코드</u>	다른 개발자가 로직을 이해하기 어려운 코드
10	<u>성능 테스트 수행 방법</u>	도환시성 1. 성능 테스트 도구 설치 2. 환경설정 3. 시나리오 작성 4. 성능 테스트 실행 및 모니터링
10	<u>성능 저하 원인 - DB 관련</u>	락페 릭사커 1. DB Lock : Lock 해제 시까지 타임 아웃 2. 불필요한 DB Fetch : 대량의 데이터 요청으로 응답 시간 저하 / 결과 세트에서 마지막 위치로 커서 옮기는 작업 빈번 시 응답 저하 3. 연결 누수 Connection Leak : JDBC 객체 사용 후 미종료 4. 부적절한 DB Connection Pool Size : 커넥션 풀이 너무 작거나 큼 5. 커밋 관련 : 트랜잭션이 커밋되지 않고 반환되거나 불필요한 커밋이 잦음
10	<u>성능 분석</u>	처용경사 1. 처리량 : 주어진 시간에 처리 가능한 트랜잭션(웹 페이지) 수 2. 응답 시간 : 사용자 입력 → 응답 출력 개시까지 걸린 시간 3. 경과시간 : 사용자 입력 → 트랜잭션 처리 → 결과 출력까지 걸린 시간 4. 자원 사용률 : 트랜잭션 처리 동안 사용하는 자원 사용량
10	<u>결함 우선순위</u>	크+하미로 1. Critical : 이 결함으로 전체 기능 미동작 2. High : 이 결함으로 다른 기능 미동작 3. Medium 4. Low
10	<u>결함 심각도</u>	크메노마심 치주보경단 1. Critical (치명적) : 테스트 불가할 정도 2. Major (주요) : 기능이 기대와 다르게 동작 3. Normal (보통) : 사소한 기능 오작동 4. Minor (경미) : UI 오류 5. Simple (단순) : 미관성 해침
10	<u>결함 분류</u>	시기지문 1. 시스템 결함 (비정상적인 종료, DB 에러) 2. 기능 결함 (요구사항 불일치, 스크립트 에러) 3. GUI 결함 (부정확한 메시지) 4. 문서 결함 (매뉴얼 불일치)
10	<u>결함 에이징 분석</u>	특정 결함의 지속 시간을 측정
10	<u>결함 추세 분석</u>	테스트 시간 흐름에 따른 결함 수를 측정
10	<u>결함 생명주기</u>	Open → Reviewed → Assigned → Resolved → Verified → Closed → Deferred → Reopen → Closed
10	<u>결함 분석 방법</u>	고구일 1. 고립화: 입력값, 절차, 환경 중 무엇이 영향을 미치는지 파악 2. 구체화: 입력값, 절차, 환경을 정확히 파악 3. 일반화: 결함에 영향을 주는 요소를 최대한 일반화
10	<u>결함 분포 분석</u>	특정 속성에 해당하는 결함 수를 측정
10	<u>결함 추이 분석</u>	결함 지표를 분석해 추후 발생할 결함을 추정하는 작업 분추에
10	<u>결함 관리 프로세스</u>	계기검수 재추최 1. 결함 관리 계획 2. 결함 관리 DB에 기록 3. 결함 검토 4. 결함 수정 5. 결함 재검토 6. 결함 추적 및 모니터링 6. 최종 결함 분석 및 보고서 작성
10	<u>테스트 리포팅</u>	정요품 결실 1. 테스트 결과 정리 2. 테스트 요약문서 작성 3. 품질 상태 파악 4. 결과서 작성 5. 테스트 실행 절차 리뷰 및 평가
10	<u>샌드위치 테스트</u>	상위 모듈은 하향식, 하위 모듈은 상향식 테스트를 수행
10	<u>테스트 자동화 도구 유형</u>	정실성통 1. 정적 분석 도구 : 실행하지 않고 소스 코드를 테스트 2. 테스트 실행 도구 : 작성된 스크립트를 실행해 테스트 (데이터 주도 접근 방식 vs. 키워드 주도 접근 방식) 3. 성능 테스트 도구 : 처용경사 테스트 4. 테스트 통제 도구 : 형상 관리 도구, 결함 추적 도구 등

▼ 과 목	Aa 단어	≡ 내용
10	<u>상향식 테스트</u>	최하위 모듈부터 위 방향으로 통합하며 테스트 1. 하위 모듈을 클러스터로 결합 (클러스터링) 2. 드라이버 개발 3. 통합된 클러스터 테스트 4. 드라이버를 실제 모듈로 대체
10	<u>빅뱅 테스트</u>	모든 컴포넌트를 통합해 한꺼번에 테스트 (비점증적 방식)
10	<u>하향식 테스트</u>	메인 제어 모듈로부터 아래 방향으로 통합하며 테스트 1. 초기에 시스템 구조 파악 2. 스텝 개발 3. 깊이-우선 또는 너비-우선 방식에 따라 스텝을 실제 모듈로 대체
10	<u>테스트 스파이</u>	테스트 대상 클래스와 협력 클래스로 가는 출력을 검증하는 데 사용
10	<u>가짜 객체</u>	협력 클래스 기능을 대체하기 위해 사용
10	<u>테스트 드라이버</u>	하위 모듈을 호출하거나 파라미터를 전달하는 모듈 (상향식 테스트에서 사용)
10	<u>목 객체</u>	사전에 조건을 입력하면 그 상황에 예정된 행위를 수행하는 가짜 객체 목 객체 유형 : 테스트 스텝, 테스트 드라이버, 테스트 스파이, 가짜 객체
10	<u>테스트 스텝</u>	특정 값을 리턴하거나 메시지를 출력하는 등 단순 기능을 수행하는 더미 모듈 (하향식 테스트에서 사용)
10	<u>테스트 하네스</u>	단위 테스트를 지원하기 위한 코드와 데이터 목스드 슈스케
10	<u>인수 테스트</u>	계약상 요구사항을 만족하는지 확인 (알파 테스트-통제된 환경에서 선택된 사용자가 개발자랑 vs. 베타 테스트-실제 환경에서)
10	<u>시스템 테스트</u>	통합된 단위시스템 검증 (기능적 요구사항 테스트, 비기능적 요구사항 테스트)
10	<u>테스트 레벨</u>	개발 단계에 따라 테스트를 분류한 것 단통시인
10	<u>단위 테스트</u>	설계 최소 단위인 모듈, 컴포넌트, 서브 루틴 등을 테스트
10	<u>통합 테스트</u>	단위 테스트를 통과한 모듈 간 인터페이스, 컴포넌트 상호작용 등을 테스트 (빅뱅 테스트, 상/하향식 테스트, 샌드위치 테스트)
10	<u>일관성 검사 오라클</u> <u>Consistent</u>	변경이 있을 때, 수행 전후 결괏값이 동일한지 확인
10	<u>참 오라클</u>	모든 입력값의 기대결과를 생성 (오류를 모두 검출 가능)
10	<u>휴리스틱 오라클</u> <u>Heuristic</u>	몇 개 입력값에 대해서는 정확한 결과를 제공하고, 나머지는 휴리스틱으로 처리 (샘플링 오라클 개선)
10	<u>샘플링 오라클</u>	특정 몇 개 입력값의 기대결과만 생성
10	<u>테스트 오라클</u>	테스트 결과가 참/거짓인지 판단하기 위해, 사전에 정의된 참값을 입력해 비교하는 기법 참샘휴일
10	<u>테스트 케이스 구성 요소</u>	입출항 특한 의식 1. 입력 명세 (입력할 데이터 및 조건) 2. 출력 명세 (기대되는 결괏값) 3. 테스트 항목 4. 특수절차요구 5. 환경설정 (수행에 필요할 HW/SW 환경) 6. 의존성 기술 (TC 간 의존성) 7. 식별자
10	<u>체크리스트</u>	테스트 내용을 목록화하여 테스트 (재사용 목적)

과 목	Aa 단어	≡ 내용
10	<u>경험 기반 테스트</u>	테스터의 경험과 직관을 기반으로 한 테스트 ▶ 탐오체득
10	<u>오류 추정</u>	개발자가 범하기 쉬운 실수를 추정하고 이에 맞춰 테스트
10	<u>탐색적 테스트</u>	TC 명세화 없이, 경험에 의존해 탐색적으로 테스트 수행 (테스터의 휴리스틱한 능력 필요)
10	<u>특성테스트</u>	ISO/IEC 9126 등 표준 품질 특성을 염두에 두고 테스트 수행
10	<u>결정 테이블 테스트 (Decision Table Testing).</u>	요구사항을 테이블로 구성해, 원인(조건)-결과(행위)를 조합해 테스트
10	<u>페어와이즈 테스트 (Pairwise Testing).</u>	테스트 데이터값을 최소 한 번씩 조합
10	<u>유스케이스 테스트 (Use Case Testing).</u>	유스케이스로 모델링 되어있을 때, 프로세스 흐름 기반으로 테스트 수행
10	<u>비교 테스트 (Comparison Testing).</u>	여러 버전의 프로그램에 같은 입력값을 넣어 결과를 비교
10	<u>분류 트리 테스트 (Classification Tree Method Testing).</u>	트리 구조로 분석 및 표현하여 테스트 (항목 선정 → 분류 트리 구축 → 분류 클래스 조합해 TC 생성)
10	<u>동등 분할 테스트 (Equivalence Partitioning Testing).</u>	입력 데이터 영역을 유효값/무효값으로 그룹핑해 대푯값 TC를 도출해 테스트
10	<u>경계값 분석 테스트 (Boundary Value Analysis Testing).</u>	최솟값을 바로 위나 아래와 같이 입력 데이터의 극한 한계를 테스트 (2-Value, 3-Value)
10	<u>원인-결과 그래프 테스트 (Cause-Effect Graphing Testing).</u>	그래프를 통해 입력값 간의 관계 및 출력에 미치는 영향을 분석해, 효용성이 높은 TC를 선택해 테스트
10	<u>상태 전이 테스트 (State Transition Testing).</u>	이벤트에 의해 객체 상태가 전이되는 경우의 수를 측정 (상태 전이도 모델링 → 전이 트리 도출 → 전이 경로 TC → 비정상 전이 TC)
10	<u>블랙박스 테스트 유형</u>	▶ 상원동경비분폐유결 1. 상태 전이 테스트 2. 원인-결과 그래프 테스트 3. 동치 분할 테스트 4. 경계값 분석 테스트 5. 비교 테스트 6. 분류 트리 테스트 7. 페어와이즈 테스트 8. 유스케이스 테스트 9. 결과 테이블 테스트
10	<u>블랙박스 테스트</u>	요구사항 명세를 보며 기능 위주 테스트 (명세 기반 테스트, 동적 테스트)
10	<u>제어 흐름 테스트 (Control Flow Testing).</u>	제어 구조를 그래프로 나타내어 테스트

과 목	Aa 단어	내용
10	데이터 흐름 테스트 (Data Flow Testing)	제어 흐름 그래프에 데이터 사용 현황을 추가한 그래프를 통해 테스트
10	기본 경로 커버리지 (Base Path Coverage)	맥케이브의 순환복잡도를 기반으로, 수행 가능한 모든 경로를 테스트 (순환복잡도 계산: $V=E-N+2$ 또는 $V=P+1$)
10	변경 조건/결정 커버리지 (MC/DC)	결정 포인트 내 개별 조건식이 다른 개별 조건식의 영향을 받지 않고, 전체 조건식에 독립적으로 영향을 주도록 수행
10	다중 조건 커버리지 (Multiple Condition Coverage)	모든 개별 조건식의 가능한 조합을 100% 보장 (개별조건식의 수가 N이라고 했을 때, 2^N 의 결과가 나옴. 즉, 개별조건식이 2개면 $2^2=4$, 3개면 $2^3=8$)
10	조건/결정 커버리지 (Condition/Decision Coverage)	결정 포인트 내 전체 조건식도 참/거짓을 한 번씩, 개별 조건식도 참/거짓을 한 번씩 수행
10	조건 커버리지 (Condition Coverage)	결정 포인트 내 개별 조건식이 참/거짓을 한 번씩 수행 (전체 조건식에 주는 영향은 고려X)
10	결정 커버리지 (Decision Coverage)	결정 포인트 내 전체 조건식이 참/거짓을 한 번씩 수행 (=선택 커버리지, 분기 커버리지)
10	화이트박스 테스트 유형	구결조조변다기제테 1. 구문 커버리지 2. 결정 커버리지 (=선택 커버리지, 분기 커버리지) 3. 조건 커버리지 4. 조건/결정 커버리지 5. 변경 조건/결정 커버리지 6. 다중 조건 커버리지 7. 기본 경로 커버리지 8. 제어 흐름 테스트 9. 데이터 흐름 테스트
10	구문 커버리지 (Statement Coverage)	모든 명령문을 한 번 이상 수행. 결과에 무관하게 구문 실행 개수로 측정
10	화이트박스 테스트	모듈 내부 구조, 논리 경로를 테스트 (구조 기반 테스트, 동적 테스트)
10	정적분석	분석 도구의 도움을 받아 수행 (정적=소스코드 실행X) - 코딩 표준 - 소스코드 복잡도 (맥케이브 순환복잡도 지표) * $V(G) = E - N + 2$ 또는 $V(G) = P + 1$ - 자료 흐름 분석
10	리뷰	SW의 결함을 검출하거나 진행 상황을 파악하는 활동으로 전문가가 수행
10	리뷰의 유형	1. 관리 리뷰 2. 기술 리뷰(=코드 리뷰) 3. 인스펙션: 저작자가 아닌 다른 전문가가 검토 (=동료 리뷰) 4. 워크스루: 사전에 자료 배포 후 짧은 회의 5. 감사: 표준 준수 확인, 제3기관 수행
10	정적 테스트 종류	리뷰 / 정적 분석
10	성능 테스트 유형	부스스내 1. 부하 테스트 (Load) : 부하를 점점 늘려 임계점을 찾는 테스트 2. 스트레스 테스트 (Stress) : 임계점 이상의 부하를 가해 비정상적인 상황에서의 성능을 측정 3. 스파크 테스트 (Spark) : 짧은 시간 내 많은 사용자가 몰릴 때 반응 테스트 4. 내구성 테스트 (Endurance) : 오랜 시간 동안 높은 부하를 가해 성능 테스트

과 목	Aa 단어	≡ 내용
10	<u>테스트 종류에 따른 분류</u>	구명경 1. 구조 기반 (내부 논리 흐름에 따라 테스트 ⇒ 화이트박스) 2. 명세 기반 (명세서 기반으로 테스트 ⇒ 블랙박스) 3. 경험 기반 (유사 평가 경험을 기반으로 테스트 ⇒ 탐오체특 : 탐색적 테스트, 오류 추정, 체크리스트, 특성 테스트)
10	<u>테스트 목적에 따른 분류</u>	회안성 구회병 1. 회복 테스트 (Recovery) : 고의로 실패를 유도한 뒤, 정상 복귀를 테스트 2. 안전 테스트 (Security) : 보안 결함 점검 3. 성능 테스트 (Performance) : 요구에 대한 반응속도를 측정 (유형: 부스스내) 4. 구조 테스트 (Structure) : 논리 경로, 소스코드 복잡도 측정 5. 회귀 테스트 (Regression) : 오류 제거 후, 수정에 의해 유입된 오류가 없는지 테스트 6. 병행 테스트 (Parallel) : 변경된 시스템과 기존 시스템에 동일 데이터 입력 후 결괏값 비교
10	<u>테스트 시각에 따른 분류</u>	검증 (Verification): 개발자 시점, 규격(명세기능)을 만족하는가 확인 (Validation): 사용자 시점, 올바른 SW
10	<u>테스트 케이스 Test Case</u>	요구사항을 만족하는지 확인하기 위해 설계된 입력값, 실행조건, 기대결과로 구성된 명세서
10	<u>테스트 스위트 Test Suites</u>	실행환경에 따라 구분해 놓은 TC의 집합
10	<u>테스트 시나리오 Test Scenario</u>	테스트할 기능, 상황을 정리해 절차를 명세화한 문서
10	<u>테스트 스크립트 Test Script</u>	TC를 실행순서를 작성한 문서 (=테스트 스텝, 테스트 프로시저)
10	<u>SW 테스트의 필요성</u>	발예향 1) 오류 발견 관점: 잠재된 오류를 발견 2) 오류 예방 관점: 사전에 오류를 발견하고 예방 3) 품질 향상 관점: 요구사항과 기대 수준을 만족하도록 품질 향상
10	<u>SW 테스트의 기본 원칙</u>	완살초 정오 집결 1) 완벽한 테스트는 불가: 무한입력값, 무한경로 2) 살충제 패러독스: 같은 TC 반복 → 새 오류 검출 불가 3) 초기에 테스트 시작: 후반에 갈수록 비용 증가 (요르돈의 법칙, 눈덩이 법칙) 4) 정황에 의존: SW 성격, 환경에 맞춰 테스트 5) 오류-부재의 궤변: 오류(결함)이 없더라도, 요구사항을 만족하지 못하면 품질이 높다고 할 수 없음 6) 결함집중: 결함 80%는 전체 모듈 20% 내에서 발견 (파레토 법칙) 7) 결함이 존재: 테스트는 결함이 존재함을 밝히는 활동
10	<u>SW 테스트</u>	개발된 소프트웨어가 요구사항을 만족하는지 확인하고 결함을 검출하는 활동
11	<u>클라우드 기반 개발 환경 인프라</u>	1) 컴퓨팅 환경 : 프로그램을 설치하고 하드웨어 세팅 2) 스토리지 : 대규모 데이터 저장을 장치 세팅 3) 데이터베이스 : 실데이터를 저장하고 관리하기 위한 세팅, 멀티미디어 데이터 처리를 위한 세팅 4) 네트워킹 전송 : 서비스, 프로그램, 콘텐츠 전달을 위한 환경 세팅 5) 개발자 도구 : 개발을 위한 환경 구축 6) 보안 환경 구축 : 시스템과 데이터 보호를 위한 액세스, 암호화 관리 등 7) 응용 기술 세팅 : AR, VR, 머신러닝, 딥러닝 등 8) 생산성 향상 : 볼륨 자동 확장 환경, 스트리밍 서비스 환경 등 구축
11	<u>TCP</u>	
11	<u>다단계 피드백 큐</u>	여러 개의 큐에 프로세스 특성에 따라 서로 다른 시간 할당량을 부여. (FIFO+RR) ⇒ 우선 도착한 프로세스를 실행하되, 완료가 안 된 프로세스는 하위 큐로 보내고 마지막 큐에서도 안 끝나면 RR 방식
11	<u>다단계 큐</u>	여러 개의 큐를 이용해 각각 독립적인 스케줄링을 가짐

▼ 과 목	Aa 단어	≡ 내용
11	RR (라운드 로빈)	시간 할당량을 정해놓고, 시간 내 처리되지 못할 경우 대기 큐 가장 뒤로 보냄
11	SRT	=Shortest Remaining Time First. 대기 큐에 수행(남은) 시간이 짧은 프로세스가 생기면 언제라도 선점
11	SJF	Shortest Job First. 대기 큐 중 수행시간이 가장 짧은 프로세스에게 할당
11	FIFO	대기 큐에 먼저 도착한 순서에 따라 선택
11	기한부	정해진 시간 내 프로세스가 완료되도록 계획
11	HRN	응답률이 높은 프로세스를 선택해 할당
11	우선순위	프로세스별 우선순위에 따라 CPU 할당
11	개발 인프라 구축 방식	온클하 1. 온프레미스 방식: 외부 인터넷망이 차단된 상태에서, 인트라넷망을 활용해 개발 환경 구축 2. 클라우드 방식: 서비스를 임대하여 개발환경 구축 3. 하이브리드 방식
11	개발 지원 도구	1. 요구사항 관리: JFeature 2. 설계: DBdesigner 3. 구현: Eclipse, CodeBlock 4. 테스트: JUnit 5. 빌드: Ant, Jenkins 6. 형상관리: SubVersion 7. 품질관리: jDepend, Mylyn 8. 이슈관리: Mentis, Git 9. 프로젝트 관리: Redmine, OpenProj
11	프로그래밍 언어별 특징	객체지향 언어: JAVA, C#, VB.NET, ABAP 순차적 언어: C++, PHP, Python, Perl, COBOL 정적 언어: JAVA, C#, VB.NET, ABAP, C++, COBOL 동적 언어: PHP, Python, Perl, SQL
11	패킷 스위칭 vs. 서킷 스위칭	패킷 스위칭 : 헤더의 주소 정보에 따라 전송 (이메일 등에 적합) 서킷 스위칭 : 데이터 일부를 송수신 해 경로를 파악 후 전송 (영상 등에 적합)
11	리눅스 계열 OS	데비안 계열 : Debian GNU, Linux(개발자 최적화), Ubuntu(가장 광범위) 레드햇 계열: Fedora(스마트 설정과 업데이트), CentOS(프로그래밍 최적화)
11	서킷 스위칭	서킷이라는 특정 연결을 만들어 독점적으로 사용해 통신하는 방식 (전송 보장)
11	패킷 스위칭	패킷으로 데이터를 전송하며, 전송하는 동안만 자원을 사용하는 통신 방식 1. X.25 : 고정된 대역폭 사용, 낮은 성능 2. 프레임 릴레이 : 유연한 대역폭 사용, 가격 저렴 3. ATM : 광대역 전송에 쓰이는 스위칭 기법
11	Telnet	네트워크 연결에 사용하는 응용계층의 프로토콜
11	SMTP	이메일을 보내기 위한 프로토콜
11	IMAP	이메일을 가져오기 위한 프로토콜 (⇒ 메일 서버에서 불러옴)
11	HTTP	인터넷에서 데이터를 주고받기 위한 텍스트 기반의 프로토콜
11	FTP	서버-클라이언트 간 파일 전송을 위한 프로토콜
11	POP3	이메일을 가져오기 위한 프로토콜 (⇒ 로컬 PC에 저장 후 불러옴)

▼ 과 목	Aa 단어	≡ 내용
11	<u>응용 계층</u> (Application).	사용자가 OSI 환경에 접근할 수 있도록 서비스(인터페이스)를 제공하는 계층 ① 단위: 데이터 ② 프로토콜: 헛프릿프 에셈팍아맵 텔넷 HTTP, FTP, SMTP, POP3, IMAP, Telnet
11	<u>MPEG</u>	멀티미디어를 위한 표준 규격
11	<u>JPEG</u>	이미지를 위한 표준 규격
11	<u>SSH</u>	보안 셸. 원격 호스트에 접근하기 위한 프로토콜
11	<u>SSL/TLS</u>	안전한 데이터 전송을 위한 보안 프로토콜 (4계층(응용) - 7계층(전송) 사이에서 안전한 데이터 전송 보장)
11	<u>NetBIOS</u>	응용계층의 애플리케이션에 API 제공
11	<u>RPC</u>	원격 프로시저 호출. 다른 주소 공간에 있는 프로세스 실행 가능
11	<u>표현 계층</u> (Presentation).	응용프로그램의 데이터를 통신에 알맞은 형태로 만들거나, 하위 계층의 데이터를 사용자가 이해할 수 있는 형태로 만드는 계층 ① 단위: 데이터 ② 프로토콜: JPEG, MPEG
11	<u>세션 계층 (Session).</u>	송수신간 연결을 제어 ① 단위: 데이터 ② 프로토콜: 알넷스스 RPC, NetBIOS, SSH, SSL/TLS
11	<u>UDP 헤더 구조</u>	소데 렝체다 1. Source Port Number 2. Destination Port Number 3. UDP Length 4. UDP Checksum 5. Data
11	<u>TCP 특징</u>	신연흐흔 1. 신뢰성 보장 : 패킷 손실, 중복이 없도록 보장 (IP 계층 보완) 2. 연결지향적 : 연결 회선을 통해 통신이 이뤄짐 3. 흐름 제어 : 송신-수신 속도 일치시킴 4. 혼잡 제어 : 네트워크 혼잡도에 따라 송신을 제어
11	<u>L4 스위치</u>	OSI 4계층에서 네트워크 단위를 연결하는 장비. TCP/UDP 등 스위칭 수행
11	<u>UDP 특성</u>	비비실해 1. 비신뢰성 : 메시지의 도착을 보장하지 않음 2. 비순서화 : 수신된 메시지 순서 맞추지 않음 3. 실시간 응용 및 멀티캐스팅 가능 4. 단순 헤더 (고정 크기 헤더)
11	<u>TCP 헤더 구조</u>	소데씨 엑해플원 체어옴패 1. Source Port Number 2. Destination Port Number 3. Sequence Number : 신뢰성과 흐름 담당 4. Acknowledgement Number : 승인 번호 (수신을 기대하는 다음 번호) 5. HLEN : 헤더 길이 6. Flag Bit : 값 유효 여부 등을 표시하는 플래그 7. Window Size 8. Checksum : 에러 확인 9. Urgent Pointer : 시퀀스 번호로부터의 옴셋 10. Options and Padding
11	<u>QoS (Quality of Service).</u>	데이터의 중요도에 따라 우선순위를 부여해, 데이터 전송 성능을 보장하는 것
11	<u>전송계층</u> (Transport).	종단 간 신뢰성 있고 효율적으로 데이터 전송 ① 단위: 세그먼트 ② 장비: L4 스위치 ③ 프로토콜 : TCP, UDP
11	<u>거리-벡터 알고리즘</u>	인접 라우터와 정보를 교환하여 경로 도출
11	<u>라우팅 알고리즘 유형</u>	거백링상 1. 거리-벡터 알고리즘 2. 링크-상태 알고리즘
11	<u>링크-상태 알고리즘</u>	링크 상태 정보를 모든 라우터에게 전달해 경로 도출 (범위 넓고 복잡함)

▼ 과 목	Aa 단어	≡ 내용
11	<u>BGP</u>	- 동적&외부 라우팅 프로토콜 - AS간 경로 정보 교환 - 경로-벡터 알고리즘 사용 - ISP 사업자간 주로 사용
11	<u>OSPF</u>	- 동적&내부 라우팅 프로토콜 - 다익스트라 알고리즘 사용 (링크-상태 알고리즘 기초) 다익싱상 - 홉카운트 무제한 - RIP의 단점 개선 - AS(자치시스템) 분할 (지역 별로 라우팅 관리)
11	<u>RIP</u>	- 동적&내부 라우팅 프로토콜 - 벨만-포드 알고리즘 사용 (거리-벡터 알고리즘 기초) 벨포거백 - 홉 카운트 15 - UDP 포트 520 사용 - 30초마다 정보 공유
11	<u>멀티캐스트</u>	같은 데이터를 여러 명의 그룹 수신자들에게 동시에 전송하는 프로토콜
11	<u>터널링</u>	인접한 IPv4 망에 터널을 만들고 캡슐화하여 전송
11	<u>브로드캐스트</u>	같은 서브 네트워크의 모든 수신자에게 데이터를 전송하는 프로토콜
11	<u>애니캐스트</u>	잠재적인 수신자 그룹 안에서 가장 가까운 노드에게 연결해 전송하는 프로토콜
11	<u>라우팅 프로토콜 종류</u>	립 오스프 비집 1. RIP 2. OSPF 3. BGP
11	<u>유니캐스트</u>	식별된 고유 주소의 목적지에 1:1로 데이터를 전송하는 프로토콜
11	<u>듀얼 스택</u>	IP 계층에 IPv4, IPv6 프로토콜을 모두 탑재하여 전송 상대에 따라 선택
11	<u>IPv4 → IPv6 전환 방법</u>	듀터주 1. 듀얼 스택 2. 터널링 3. 주소변환 : 게이트웨이(주소변환기)로 패킷 변환
11	<u>IPv6의 특징</u>	확인실때 플플이단해 1. IP주소 확장 2. 인증 및 보안 기능 3. 실시간 패킷 추적 가능 4. Plug & Play 지원 (실시간 멀티미디어 처리) 5. 이식성 (물리적 위치의 제한 X) 6. 단순 헤더
11	<u>IPv6</u>	128 Bit (=16Byte) 주소 체계를 갖는 인터넷 프로토콜 (16비트 * 8부분 = 128) 멀티캐스트, 유니캐스트, 애니캐스트 전송방식 멀유애
11	<u>서브넷 마스크</u>	IP 주소에서 Network와 Host를 구분하는 것
11	<u>IPv4 구성</u>	찌릿삼구오 (127, 191, 223, 239, 255) - D클래스 : 멀티캐스트 용도 - E클래스 : 연구용
11	<u>IPv4</u>	32Bit (=4Byte) 주소 체계를 갖는 인터넷 프로토콜 (8비트 * 4부분 = 32) 멀티캐스트, 유니캐스트, 브로드캐스트 전송방식 멀유브
11	<u>망(백본) 스위칭 허브</u>	광역 네트워크를 커버하는 스위칭 허브
11	<u>IP</u>	패킷 단위의 네트워크 통신 프로토콜
11	<u>라우팅 프로토콜</u>	최적의 데이터 전송 경로를 설정하는 프로토콜 립 오스프 비집
11	<u>IGMP</u>	멀티캐스트 실시간 전송을 위해 사용하는 프로토콜 (화상회의 등)
11	<u>ARP</u>	IP주소(3계층)을 MAC주소(2계층)으로 변환하는 프로토콜
11	<u>RARP</u>	MAC 주소는 알지만 IP 주소는 모를 때 사용하는 프로토콜

▼ 과 목	Aa 단어	≡ 내용
11	<u>L3 스위치</u>	3계층에서 동작하는 스위치 (L2 스위치 기능 + L3 라우터 기능을 모두 갖춘 장비)
11	<u>라우터</u>	최적의 경로를 지정하고 경로에 따라 전송시키는 장비
11	<u>인터넷 공유기</u>	하나의 인터넷 라인을 여러 컴퓨터가 공유할 수 있게 해주는 장비
11	<u>ICMP</u>	IP 패킷 처리 시의 문제를 알려주는 프로토콜
11	<u>게이트웨이</u>	다른 통신망에 접속할 수 있게 해주는 장비
11	<u>네트워크 계층</u>	데이터 전송을 위한 최적의 경로를 설정 ① 단위: 패킷 ② 장비: 스라게공망 L3 스위치, 라우터, 게이트웨이, 인터넷 공유기, 망(백본) 스위칭 허브 ③ 프로토콜: 아라라씨지 IP, 라우팅 프로토콜, ARP, RARP, ICMP, IGMP
11	<u>ATM</u>	고정 크기 단위로 전송하는 비동기식 전송 기술
11	<u>PPP</u>	통신 노드 간 연결을 위한 프로토콜
11	<u>스위칭 허브</u>	스위치 기능을 가진 허브
11	<u>Frame Relay</u>	프레임 간 중계기능, 다중화 기능을 통해 빠른 데이터 전송이 가능한 고속 전송 기술
11	<u>HDLC</u>	점대점, 다중점 통신에 사용하는 프로토콜
11	<u>브릿지</u>	LAN과 LAN을 연결하는 장비
11	<u>NIC</u>	=네트워크 인터페이스 카드. 외부 네트워크와 빠른 통신을 위해 컴퓨터 내에 설치되는 장비
11	<u>L2 스위치</u>	목적지 MAC 주소를 기반으로 빠르게 데이터를 전송하는 장비 스포 컷스 프프 1. Store and Forwarding : 데이터를 전부 받은 후 처리 2. Cut Through : 목적지 주소만 확인 후 바로 전송 3. Fragment Frame : 앞 64비트만 읽어 에러 처리 후 전송
11	<u>리피터</u>	디지털 신호를 증폭시키는 장비
11	<u>데이터링크 계층 (Data Link)</u>	노드 간 오류 제어, 흐름 제어, 회선 제어 ① 단위: 프레임 ② 장비: 투브앤스 L2 스위치, 브릿지, NIC, 스위칭 허브 ③ 프로토콜: HPFA HDLC, PPP, Frame Relay, ATM
11	<u>허브</u>	여러 대의 컴퓨터를 연결해 네트워크로 보내거나, 하나의 네트워크로 수신된 정보를 여러 대의 컴퓨터로 보내는 장비
11	<u>물리 계층 (Physical)</u>	데이터를 전기적인 신호로 변환시켜 통신 ① 단위: 비트 ② 프로토콜: RS-232C ③ 장비: 허브, 리피터
11	<u>OSI 7계층</u>	네트워크 충돌 문제를 최소화하고자, ISO(국제표준화기구)에서 제시한 통신 규약 [이미지] 계층 : 아파서티내다피 단위 : 비프페세데
11	<u>네트워크 프로토콜</u>	컴퓨터(통신 장비) 간 메시지 교환을 위한 규약 - 단편화 : 전송이 가능한 단위로 나누는 기법 - 재조립 : 단편화된 조각을 복원하는 기법 - 캡슐화 : 상위 계층의 데이터에 정보를 추가해 하위 계층으로 보내는 기법 (송신측) - 동기화 : 송신-수신 시점 맞추는 기법

과 목	Aa 단어	≡ 내용
11	<u>프로토콜의 기본 3요소</u>	구의타 1. 구문 Syntax : 정보 전송을 위한 데이터 형식 2. 의미 Semantic : 정보 전송을 위한 제어 정보 3. 타이밍 Timing : 정보 전송을 위한 순서와 속도 조절
11	<u>프로토콜</u>	서로 다른 시스템 간 데이터 교환을 위한 규약
11	<u>PaaS</u>	플랫폼형 서비스. 인프라를 관리하는 복잡함 없이, 애플리케이션 개발에 필요한 플랫폼을 클라우드로 제공
11	<u>네트워크</u>	원하는 정보를 정확하게 전달하기 위한 인프라 1. LAN : 근거리 네트워크 2. WAN : 광대역 네트워크
11	<u>SaaS</u>	소프트웨어형 서비스. 사용자에게 제공하는 소프트웨어와 관련 데이터를 클라우드로 제공
11	<u>공용 클라우드</u>	제공 업체의 서비스를 이용한 클라우드. 확장성 높음
11	<u>하이브리드 클라우드</u>	사설+공용 클라우드 동시 사용
11	<u>IaaS</u>	인프라형 서비스. 서버, 스토리지 같은 자원을 클라우드로 제공
11	<u>클라우드 컴퓨팅</u>	자신의 컴퓨터가 아닌 클라우드에 연결된 컴퓨터로 처리하는 기술 분류 : 사공하 유형 : 인플소(IPS)
11	<u>사설 클라우드</u>	조직 내부의 컴퓨팅 자원을 이용해 내부적으로 구축한 클라우드. 보안성 높고, 직접 제어 가능
11	<u>NFV</u>	=Network Function Virtualization 라우터, 로드밸런서 등 하드웨어에 가상화 기술을 적용하여 네트워크 서비스를 가상화하는 기술
11	<u>SDN</u>	=Software Defined Network 컨트롤 플레인(트래픽 경로 지정) / 데이터 플레인(트래픽 전송)으로 분리하여 네트워크를 관리하는 기술
11	<u>네트워크 가상화 기술</u>	물리적으로 떨어진 장비들을 연결하는 기술 ex. SDN, NFV
11	<u>분산 처리 기술</u>	✓ 여러 대의 컴퓨터의 계산 능력을 이용해 데이터를 처리하는 기술
11	<u>컨테이너</u>	✓ 컨테이너화된 애플리케이션들이 단일 운영체제상에서 실행되도록 하는 기술 ex. 도커
11	<u>I/O 가상화</u>	I/O와 서버 사이에 계층을 추가해 자원을 효율적으로 활용 ex. VNIC(가상 네트워크 인터페이스 카드)
11	<u>컴퓨팅 가상화</u>	컴퓨터 리소스를 가상화하여 논리적 단위로 활용 ex. 하이퍼바이저
11	<u>스토리지 가상화</u>	스토리지와 서버 사이에 계층을 추가해 논리적 단위로 활용 ex. 분산 파일 시스템
11	<u>플랫폼 가상화</u>	✓ 하드웨어 플랫폼 위에서 실행되는 호스트 프로그램이 게스트 프로그램을 만들어 독립된 환경을 구축한 것처럼 보여주는 기술
11	<u>리소스 가상화</u>	✓ 소프트웨어가 독립된 하드웨어에서 실행된 것처럼 보여주는 기술
11	<u>가상화</u>	물리적인 리소스를 하나로 보이게 하거나, 하나의 물리적인 리소스를 여러 개처럼 보이게 하는 기술 - 가상화의 종류 : 플랫폼 가상화, 리소스 가상화
11	<u>선점형 스케줄링</u>	우선순위가 높은 프로세스가 오면 기존 프로세스를 중단하고 CPU를 점유 알알다

▼ 과 목	Aa 단어	≡ 내용
11	<u>비선점형 스케줄링</u>	CPU 할당받으면 반환 시까지 다른 프로세스가 점유 불가 우기훈FJ
11	<u>응답률 계산식</u>	$(\text{대기시간} + \text{서비스시간}) / \text{서비스시간} = \text{응답률}$ 응답률이 높을수록 우선순위가 높다 (SJF의 기아현상을 해결하기 위해 HRN 기법에서 사용)
11	<u>프로세스 스케줄링</u>	CPU를 사용하려는 프로세스 간 우선순위를 관리하는 작업 반종도 대반서 * 반환시간(응답시간) = 종료 시간 - 도착 시간 * 대기시간 = 반환시간 - 서비스 시간 0부터 시작
11	<u>PCB</u>	=Process Control Block. OS가 프로세스를 관리를 위해 필요한 내용을 담고 있는 자료 구조
11	<u>프로세스 상태 전이</u>	디타블웨 1. 디스패치 : Ready List에서 프로세스를 선정하고 CPU 할당 (준비 → 실행) 2. 타이머 런 아웃 : 할당 시간 초과, PCB에 저장 후 준비 상태로 전이 (실행 → 준비) 3. 블록 : 입출력 발생 (실행 → 대기) 4. 웨이크업 : 입출력 종료 (대기 → 준비)
11	<u>프로세스 상태</u>	[이미지] 생존실대완 1. 생성 (fork) 2. 준비 (Ready List에서 할당을 대기) 3. 실행 4. 대기 (입출력 발생) 5. 완료
11	<u>프로세스</u>	CPU에 의해 관리되는 현재 실행중인 프로그램 (=Job, Task)
11	<u>메모리 단편화</u>	메모리 할당/반납 과정에서 낭비되는 공간이 생기는 것 내슬외버공통압 1. 내부 단편화 (적재하고 남는 공간 발생; 페이지징) ⇒ Slab Allocator, 통합, 압축 2. 외부 단편화 (작아서 못쓰는 공간 발생; 세그멘테이션) ⇒ 버디 메모리 할당, 통합, 압축
11	<u>할당 기법</u>	주기억장치에 어떤 방법으로 프로세스를 할당할 것인지 결정하는 기법 연단다분페 세 1) 연속 할당 : 연속해서 배치 ex. 단일 분할, 다중 분할 2) 분산 할당 : 프로세스를 조각내어 배치 ex. 페이지징(같은 크기로), 세그멘테이션(가변적 크기로)
11	<u>교체 기법</u>	주기억장치에 있는 프로세스 중 어떤 것을 제거할지 결정하는 기법 피포/엘루/엘푸/옴트/누르/스크르 1. FIFO : 가장 먼저 들어온 페이지 교체 2. LRU : 가장 오랫동안 사용되지 않은 페이지 교체 3. LFU : 참조 횟수가 적은 페이지 교체 4. OPT : 앞으로 사용될 가능성이 적은 페이지 교체 5. NUR : 최근까지 사용되지 않은 페이지 교체 (LRU 오버헤드 감소) 6. SCR : 세컨드 찬스! 가장 먼저 들어왔지만 자주 사용되는 페이지의 교체를 막음 (FIFO 보완)
11	<u>배치 기법</u>	주기억장치의 어떤 위치에 프로세스를 할당할 것인지 결정하는 기법 1) 최초 배치 : 첫 번째 분할에 배치 2) 최적 배치 : 비슷한 공간에 배치 3) 최악 배치 : 가장 큰 공간에 배치
11	<u>메모리 관리 기법</u>	반배할교 1. 반입 기법 - "언제" 요예 2. 배치 기법 - "어디에" 초적악 3. 할당 기법 - "어떻게" 연단다분페세 4. 교체 기법 - "누구를" 피포/엘루/엘푸/옴트/누르/스크르
11	<u>반입 기법</u>	주기억장치에 다음 프로세스를 언제 할당할 것인지 결정하는 기법 요예 1) 요구 반입 : 요구가 있을 시 반입 2) 예상 반입 : 예측하여 반입
11	<u>메모리 관리</u>	필요할 때마다 프로세스에게 기억장치를 할당하고, 사용이 끝나면 회수하는 것

과 목	Aa 단어	≡ 내용
11	<u>Unix/Linux 기본 명령어</u>	uname -a : 모든 시스템 정보 표시 uname -r : OS 배포버전 표시 cat : 파일 내용 출력 uptime : 가동시간 확인 id : 사용자 로그인명, id 확인 last : 모든 로그인/로그아웃 정보 확인 (wtmp(x) 파일) who : 현재 접속한 사용자 정보 확인 (utmp(x) 파일) ls : 현재 경로의 파일 및 폴더 출력 pwd : 현재 디렉토리의 절대 경로 출력 (print working direcotry) cd : 디렉토리 이동 rm : 파일 삭제 cp : 파일 복제 mv : 파일 이동 rsync : 복사 후 동기화 chmod : 퍼미션 변경 chown : 소유자, 소유그룹 변경 tar : 압축 해제 gzip : 압축 find : 파일 검색 grep : 문자열 검색 df : 남은 디스크 용량 확인 (disk free) du : 파일 사이즈 확인 (disk usage) ps : 프로세스 목록 확인 pmap : 프로세스ID 기준 메모리맵 정보 표시 kill : 프로세스 종료 fork : 프로세스 생성 ifconfig : 네트워크 인터페이스 확인 host : host는 아는데 IP 주소 모를 때 사용
11	<u>접근제어</u>	user / group / other / all 유그아을 r (4) / w (2) / x (1) ex. chmod o-w test.txt = other 사용자의 쓰기 권한 제거 chmod 664 test.txt = user, group의 rx 권한 설정, other의 r권한 설정
11	<u>Android</u>	Linux 기반의 휴대용 장치를 위한 운영체제
11	<u>Windows 기본 명령어</u>	ATTRIB : 파일 속성 표시 CALL : 일괄 프로그램에서 다른 일괄 프로그램 호출 CD : 디렉토리 표시 CHKDSK : 디스크 검사 CLS : 화면 지움 CMD : 명령 프롬프트 실행 COMP : 두 개 이상 파일 비교 DISKPART : 파티션 구성 표시 ECHO : 메시지 표시 ERASE : 파일 삭제 EXIT : 인터프리터 종료
11	<u>Mac OS</u>	Unix 기반의 GUI 운영 체제
11	<u>Linux</u>	Unix 기반의 오픈 소스 운영 체제
11	<u>Unix의 특징</u>	대다사이게 1. 대화식 운영체제 기능 : 명령어 입력 시 해당 명령어 수행 2. 다중 작업 기능 3. 다중 사용자 기능 4. 이식성 제공 5. 계층 트리 구조 파일 시스템 기능 제공
11	<u>Windows의 특징</u>	지선자오 1. GUI 제공 2. 선점형 멀티태스킹 방식 제공 : 동시에 여러 프로그램을 실행하면서 자원 사용을 제어 3. 자동 감지 기능 제공 : HW 설치 시 환경 구성 4. OLE 기능 제공 : 문서에 개체 삽입/연결
11	<u>OS의 커널</u>	하드웨어와 관련된 핵심 처리 기능 담당
11	<u>OS의 셸</u>	사용자 명령에 대한 처리를 담당 (사용자 명령을 입력받아 기계어 형태로 변환해 커널에 전달)
11	<u>OS의 특징</u>	편인스자제 1. 사용자 편리성 제공 2. 인터페이스 기능 제공 3. 스케줄링 담당 : 자원 분배 4. 자원 관리 : CPU, 메모리 관리 5. 제어 기능 : 입출력 장치 등
11	<u>OS (운영체제)</u>	사용자가 하드웨어를 쉽게 사용할 수 있도록 인터페이스 기능을 제공하는 소프트웨어
12	<u>제품 소프트웨어 패키징</u>	개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위해 포장하는 과정
12	<u>증분 백업</u>	정해진 시점 기준으로 그 이후의 변경된 데이터 백업
12	<u>차등 백업</u>	마지막 전체 백업 이후 변경된 모든 데이터 백업

▼ 과 목	Aa 단어	≡ 내용
12	<u>백업의 유형</u>	전차중 1. 전체 백업 Full 2. 차등 백업 Differential 3. 증분 백업 Incremental
12	<u>배포용 미디어 제작 프로세스</u>	선관설 검증인 1. 미디어 형태 선정 (CD? 온라인?) 2. 관리 체계 확인 (시리얼 넘버 등) 3. 설치 파일 및 매뉴얼 확인 4. 배포본 검증 5. 배포용 미디어 정보 확인 7. 최종 인증 확인 및 배포
12	<u>제품 소프트웨어 배포본</u>	사용자가 쓰기 편하도록 개발된 컴포넌트 또는 패키지가 제품화된 형태
12	<u>설치 매뉴얼 구성요소</u>	목이주구 1. 목차 및 개요 2. 이력 정보 3. 매뉴얼 주석 4. 설치 도구 구성
12	<u>제품 소프트웨어 사용자 매뉴얼</u>	사용자가 소프트웨어 사용에 필요한 내용을 기록한 문서
12	<u>사용자 매뉴얼 작성 프로세스</u>	작구구검 1. 작성 지침 정의 2. 구성요소 정의 3. 구성요소별 내용 작성 4. 검토
12	<u>코드 난독화</u>	역공학을 방지하기 위해 소스코드를 알아보기 힘들게 바꾸는 기술
12	<u>CMS</u>	콘텐츠 생산~폐기까지 전 과정을 관리하는 기술
12	<u>Secure DB</u>	DB의 파일을 암호화는 보안 강화 기술
12	<u>제품 소프트웨어 매뉴얼</u>	제품 소프트웨어를 설치 및 사용하는 데 필요한 내용을 기록한 문서
12	<u>MPEG-21</u>	멀티미디어 프레임워크를 위한 표준 규격
12	<u>XML</u>	특수한 목적을 갖는 마크업 언어를 만드는 데 쓰는 언어
12	<u>SSO</u>	한 번의 인증을 통해 여러 시스템에 재인증할 필요 없이 로그인하는 기술
12	<u>On-the-fly Packaging</u>	사용자가 요청한 시점에 콘텐츠를 암호화하는 방법
12	<u>Pre-packaging</u>	콘텐츠를 등록하자마자 암호화하는 방법
12	<u>XrML</u>	권리 조건을 표현한 XML 기반의 마크업 언어
12	<u>DOI</u>	Digital Object Identifier. 디지털 저작물에 번호를 부여하는 시스템
12	<u>URI</u>	인터넷 자원을 식별하는 고유 주소
12	<u>전자서명</u>	서명자와 서명사실을 나타내는 전자적 정보 cf. 디지털 서명 알고리즘(DSA) ⇒ SHA-1, 국내 표준 디지털 서명 아로리즘(KCDSA) ⇒ HAS-160
12	<u>PKI</u>	공개키 기반 구조(Public Key Infrastructure). 인증서를 발급받아 안전하게 통신
12	<u>패키징 도구</u>	암키식 저 인정파크 1. 암호화 : PKI, 전자서명 2. 키 관리 3. 식별기술: DOI, URI 4. 저작권 표현: XrML, MPEG-21 5. 인증: SSO 6. 정책 관리: XML, CMS 7. 암호화 파일 생성 : Pre-packaging, On-the-fly Packaging 8. 크랙 방지 : 코드 난독화, SecureDB

▼ 과 목	Aa 단어	≡ 내용
12	<u>DRM 구성요소</u>	제콘패 클소 컨보 1. 제공자 (저작권자) 2. DRM 콘텐츠 : 암호화된 콘텐츠와 콘텐츠의 메타 데이터 3. 패키저 : 콘텐츠 배포를 위해 묶는 도구 4. 클리어링 하우스 : 라이선싱 중개/발급 및 저작권료 정산/분배 5. 콘텐츠 소비자 6. DRM 컨트롤러 : 콘텐츠의 이용 권한 통제 7. 보안 컨테이너 : 콘텐츠의 유통을 위한 보안장치
12	<u>저작권</u>	저작물에 대한 배타적 독점적 권리
12	<u>제품 소프트웨어 패키징 도구</u>	디지털 콘텐츠의 저작권을 보호하고 안전한 유통과 배포를 보장하는 도구
12	<u>패키징 도구 활용시 고려사항</u>	압이복최사 1. 암호화/보안 고려 2. 이기종 연동 고려 3. 복잡성 및 비효율성 고려 4. 최적화 암호화 알고리즘 적용 5. 사용자 편의성 고려
12	<u>릴리즈 노트 작성항목</u>	헤 개목이 재수사 소노 면연 1. 헤더: 문서 이름, 제품 이름, 버전 번호 등 2. 개요 3. 목적 : 새로운 기능 목록 4. 이슈 요약 : 버그 설명이나 추가항목 등 5. 재현 항목 : 버그 재현 단계 6. 수정/개선 내용 7. 사용자 영향도 8. 소프트웨어 지원 영향도 9. 노트 : SW/HW 설치 항목, 업그레이드 항목 10. 면책조항 11. 연락처
12	<u>릴리즈 노트</u>	제품에 대한 정보와, 수정/변경 작업에 대한 정보를 제공하는 문서
12	<u>모듈화</u>	모듈(기능 단위로 분해하고 추상화되어 재사용 가능한 단위) 중심의 소프트웨어 설계 기법
	<u>Untitled</u>	