

소프트웨어 개발

1장 / 데이터 입 · 출력 구현

2장 / 통합 구현

3장 / 제품 소프트웨어 패키징

4장 / 애플리케이션 테스트 관리

5장 / 인터페이스 구현



1 장

데이터 입·출력 구현

034 자료 구조 **A** 등급

035 데이터저장소 / 데이터베이스 / DBMS **A** 등급

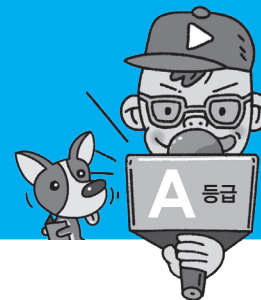
036 데이터 입·출력 **B** 등급

037 절차형 SQL **B** 등급



이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 배열 2. 선형 리스트 3. 연결 리스트 4. 스택 5. 트리 6. 데이터베이스 7. DBMS 8. SQL 9. ORM
10. 트랜잭션



전문가의 조언

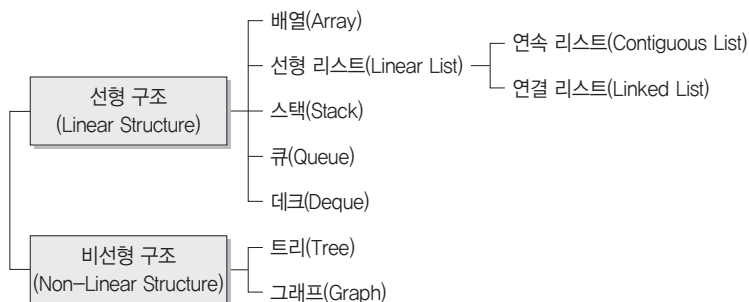
자료 구조를 선형 구조와 비선형 구조로 구분할 수 있도록 확실히 기억하고, 각 자료 구조의 특징을 학습하세요.

1 자료 구조의 정의

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

- 자료 구조는 자료의 표현과 그것과 관련된 연산이다.
- 자료 구조는 일련의 자료들을 조직하고 구조화하는 것이다.
- 어떠한 자료 구조에서도 필요한 모든 연산들을 처리할 수 있다.
- 자료 구조에 따라 프로그램 실행시간이 달라진다.

2 자료 구조의 분류



3 배열(Array)

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

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

예 1 크기가 n인 1차원 배열 a

a[0]	a[1]	a[2]	a[3]	...	a[n-1]
------	------	------	------	-----	--------

a는 변수의 이름이고, '[숫자]'는 첨자에 해당한다. 하나의 사각형은 하나의 기억장소를 가리키며, a[0]부터 a[n-1]까지 총 n개의 기억장소가 존재한다.

예 2 크기가 n×n인 2차원 배열 a

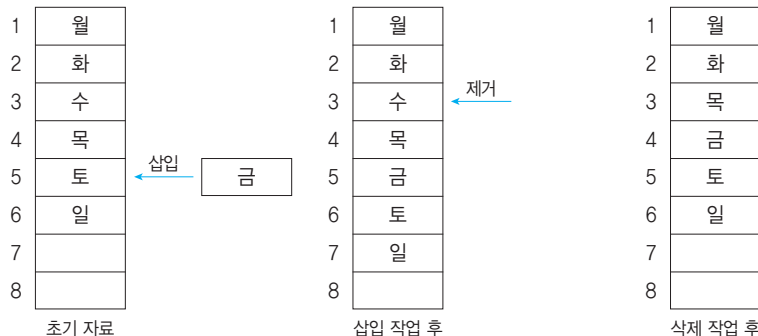
a[0][0]	a[0][1]	a[0][2]	...	a[0][n-1]
a[1][0]	a[1][1]	a[1][2]	...	a[1][n-1]
a[2][0]	a[2][1]	a[2][2]	...	a[2][n-1]
...
a[n-1][0]	a[n-1][1]	a[n-1][2]	...	a[n-1][n-1]

2개의 첨자가 존재하는 2차원 배열로 a[0][0]부터 a[n-1][n-1]까지 총 n×n개의 기억장소가 존재한다.

4 선형 리스트(Linear List)

선형 리스트는 일정한 순서에 의해 나열된 자료 구조이다.

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



- 연결 리스트(Linked List)
 - 연결 리스트는 자료들을 반드시 연속적으로 배열시키지는 않고 임의의 기억공간에 기억시키되, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료 구조이다.



전문가의 조언

선형 리스트는 빈 공간 없이 차례 차례 데이터가 저장된다는 것을 염두에 두고 특징을 정리하세요.

포인터

포인터는 현재의 위치에서 다음 노드의 위치를 알려주는 요소입니다.

- 프론트 포인터(F, Front Pointer) : 리스트를 구성하는 최초의 노드 위치를 가리키는 요소
- 널 포인터(Null Pointer, Nil Pointer) : 다음 노드가 없음을 나타내는 포인터로, 일반적으로 마지막 노드의 링크 부분에 0, ^, \0 등의 기호를 입력하여 표시합니다.

밀도가 1

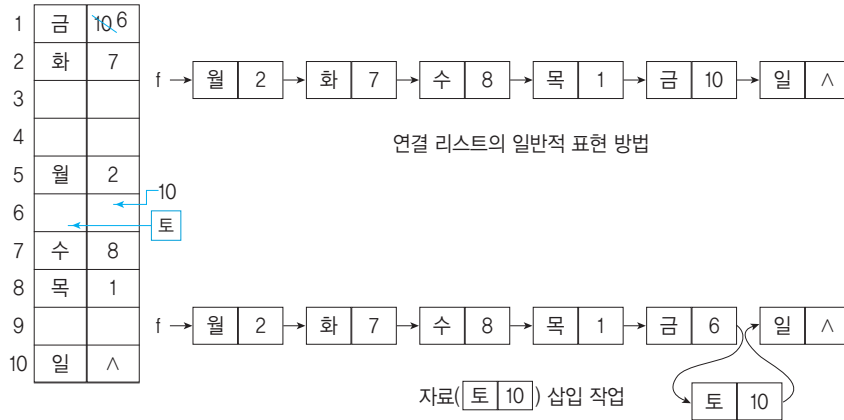
밀도란 일정한 면적에 무엇이 뻥뻥이 들어 있는 정도를 말하는 것입니다. 연속 리스트의 기억장소 이용 효율을 '밀도가 1'이라고 표현한 것은 연속 리스트는 기억장소를 연속적으로 배정받아 데이터를 기억하므로 배정된 기억장소를 빈 공간없이 꼭 차게 사용한다는 의미입니다.

노드(Node)

Data 부분 Link 부분

노드는 자료를 저장하는 데이터 부분과 다음 노드를 가리키는 포인터인 링크 부분으로 구성된 기억 공간입니다.

- 연결 리스트는 노드*의 삽입·삭제 작업이 용이하다.
- 기억 공간이 연속적으로 놓여 있지 않아도 저장할 수 있다.
- 연결 리스트는 연결을 위한 링크(포인터) 부분이 필요하기 때문에 순차 리스트에 비해 기억 공간의 이용 효율이 좋지 않다.
- 연결 리스트는 연결을 위한 포인터를 찾는 시간이 필요하기 때문에 접근 속도가 느리다.
- 연결 리스트는 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘들다.



연결 리스트 기억장치 내에서의 표현 방법



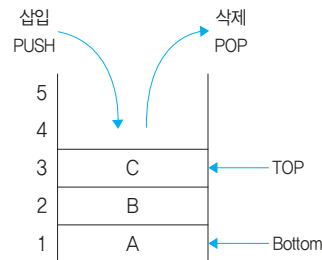
전문가의 조언

중요해요! 한쪽으로만 입·출력이 가능한 스택의 개념을 숙지하세요.

5 스택(Stack)

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

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

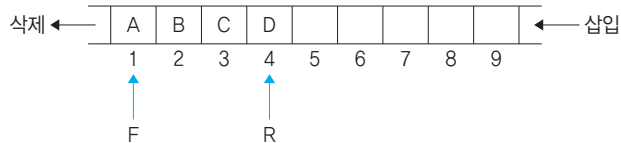


- TOP : 스택으로 할당된 기억 공간에 가장 마지막으로 삽입된 자료가 기억된 위치를 가리키는 요소이다.
- Bottom : 스택의 가장 밑바닥이다.

6 큐(Queue)

큐는 리스트의 한쪽에서는 삽입 작업이 이루어지고 다른 한쪽에서는 삭제 작업이 이루어지도록 구성된 자료 구조이다.

- 큐는 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO; First In First Out) 방식으로 처리한다.
- 큐는 시작과 끝을 표시하는 두 개의 포인터가 있다.

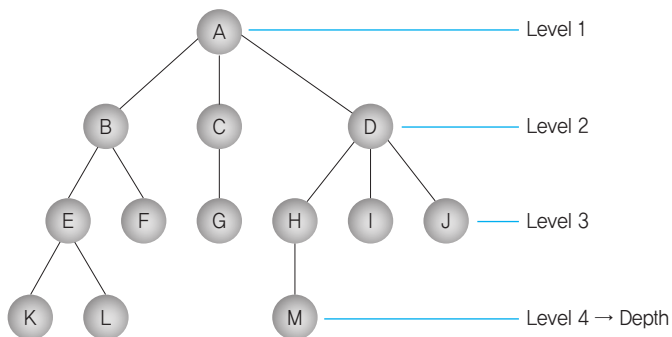


- **프런트(F, Front) 포인터** : 가장 먼저 삽입된 자료의 기억 공간을 가리키는 포인터로, 삭제 작업을 할 때 사용한다.
- **리어(R, Rear) 포인터** : 가장 마지막에 삽입된 자료가 위치한 기억 공간을 가리키는 포인터로, 삽입 작업을 할 때 사용한다.
- 큐는 운영체제의 작업 스케줄링에 사용한다.

7 트리(Tree)

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

- 트리는 하나의 기억 공간을 노드(Node)라고 하며, 노드와 노드를 연결하는 선을 링크(Link)라고 한다.
- 트리는 가족의 계보(족보), 조직도 등을 표현하기에 적합하다.
- 트리 관련 용어



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

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

- **근 노드(Root Node)** : 트리의 맨 위에 있는 노드

예 A



전문가의 조언

중요해요! 한쪽으로는 입력만, 다른 한쪽으로는 출력만 가능한 큐의 개념을 숙지하세요.



기출문제 따라잡기

Section 034

출제예상

5. 배열(Array)에 대한 설명으로 거리가 먼 것은?

- ① 동일한 자료형의 데이터들이 같은 크기로 나열되어 순서를 가지는 집합이다.
- ② 기억장소의 추가가 어려우며, 데이터 삭제 시 메모리의 낭비가 발생한다.
- ③ 데이터마다 동일한 이름의 변수를 사용하여 처리가 간편하다.
- ④ 연결을 위한 링크(포인터) 부분이 필요하다.

위의 문제에서 배열은 무엇을 이용하여 데이터에 접근한다고 했죠?

이전기출

6. 연속 리스트의 특징이 아닌 것은?

- ① 일정한 순서에 의해 나열된 구조이다.
- ② 배열과 같이 연속되는 기억장소에 저장되는 리스트를 말한다.
- ③ 기억장소의 효율을 나타내는 메모리 밀도가 1이다.
- ④ 데이터 항목을 추가, 삭제하는 것이 용이하다.

연속 리스트는 연속되는 기억 장소에 차례대로 데이터가 저장되는 구조이기 때문에 중간에 데이터를 추가하거나 삭제하려면 해당 위치 이후의 모든 데이터를 이동시켜야 한다는 단점이 있습니다.

이전기출

7. 연결 리스트(Linked List)에 대한 설명으로 거리가 먼 것은?

- ① 노드의 삽입이나 삭제가 쉽다.
- ② 노드들이 포인터로 연결되어 검색이 빠르다.
- ③ 연결을 해주는 포인터(Pointer)를 위한 추가 공간이 필요하다.
- ④ 연결 리스트 중에서 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘들다.

포인터로 연결되어 있어 포인터를 찾아가는 시간이 필요하므로 선형 리스트에 비해 접근 속도가 느립니다.

이전기출

8. 포인터를 사용하여 리스트를 나타냈을 때의 설명 중 옳지 않은 것은?

- ① 새로운 노드의 삽입이 쉽다.
- ② 기억 공간이 많이 소요된다.
- ③ 한 리스트를 여러 개의 리스트로 분리하기 쉽다.
- ④ 노드를 리스트에서 삭제하기 어렵다.

포인터와 링크가 같은 말이라는 것은 알고 있죠? 링크는 다음 노드의 위치를 가리키는 주소로서 리스트에서 특정 노드를 삭제할 때는 그 노드와 연결된 링크를 끊으면(지우면) 간단하게 삭제됩니다.

이전기출

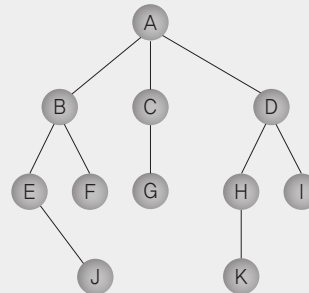
9. 리스트 내의 데이터 삽입, 삭제가 한쪽 끝에서 이루어지는 데이터 구조를 무엇이라 하는가?

- ① 스택(Stack) ② 큐(Queue)
- ③ 데크(Deque) ④ 원형 큐(Circular Queue)

스택의 가장 큰 특징은 삽입과 삭제가 리스트의 한쪽 끝에서 이루어진다는 것입니다.

이전기출

10. 다음 Tree의 Degree와 터미널 노드의 수는?



- ① Degree : 2 터미널 노드 : 4
- ② Degree : 3 터미널 노드 : 5
- ③ Degree : 4 터미널 노드 : 2
- ④ Degree : 4 터미널 노드 : 10

터미널 노드(단말 노드)는 자식이 하나도 없는 노드입니다. 그리고 트리의 디그리는 가장 차수가 많은 노드의 디그리입니다.

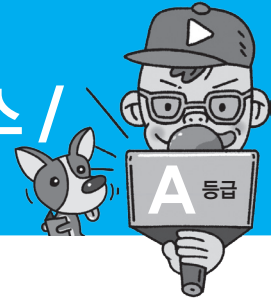
이전기출

11. 트리 구조에 대한 용어 설명 중 옳지 않은 것은?

- ① 어떤 노드의 서브트리 수를 그 노드의 차수라고 한다.
- ② 차수가 0인 노드를 단말 노드라고 한다.
- ③ 같은 부모 노드를 가지는 노드를 형제 노드라고 한다.
- ④ 모든 노드는 하나의 부모 노드를 가진다.

트리 관련 용어는 외우지 않아도 기계의 구성원을 생각하면 쉽게 이해 할 수 있습니다. 모든 노드가 하나의 부모 노드를 가진다는 것은 모든 사람의 부모가 같다는 말이니 틀린 말이죠.

▶ 정답 : 1. ④ 2. ② 3. ④ 4. ① 5. ④ 6. ④ 7. ② 8. ④ 9. ① 10. ② 11. ④



전문가의 조언

데이터베이스와 관련된 자세한 내용은 3과목에서 다루고 있습니다. 여기서는 제시된 용어들의 정의와 간단한 특징들만 정리하고 넘어가세요.



전문가의 조언

데이터베이스 구축 및 설계에 대한 자세한 내용은 Section 070을 참조하세요.



전문가의 조언

데이터베이스의 정의는 여러 사람에 의해 **공동**으로 사용될 데이터를 중복을 배제하여 **통합**하고, 쉽게 접근하여 처리할 수 있도록 저장장치에 **저장**하여 항상 사용할 수 있도록 운영하는 **운영** 데이터라고 생각하면 쉽습니다.

운영 데이터

단순한 입·출력 자료나 작업 처리상 일시적으로 필요한 임시 자료는 운영 자료로 취급되지 않습니다.



전문가의 조언

MS-Office 프로그램에 포함되어 있는 액세스 프로그램도 DBMS의 한 종류입니다. DBMS의 정의와 필수 기능 세 가지를 꼭 알고 넘어가세요.

1 데이터저장소

데이터저장소는 소프트웨어 개발 과정에서 다루어야 할 데이터들을 논리적인 구조로 조직화하거나, 물리적인 공간에 구축한 것을 의미한다.

- 데이터저장소는 논리 데이터저장소와 물리 데이터저장소로 구분된다.
- 논리 데이터저장소는 데이터 및 데이터 간의 연관성, 제약조건을 식별하여 논리적인 구조로 조직화한 것을 의미한다.
- 물리 데이터저장소는 논리 데이터저장소에 저장된 데이터와 구조들을 소프트웨어가 운용될 환경의 물리적 특성을 고려하여 하드웨어적인 저장장치에 저장한 것을 의미한다.
- 논리 데이터저장소를 거쳐 물리 데이터저장소를 구축하는 과정은 데이터베이스를 구축하는 과정과 동일하다.

2 데이터베이스

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

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

3 DBMS(DataBase Management System; 데이터베이스 관리 시스템)

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

- DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해 준다.
- DBMS는 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.
- DBMS의 필수 기능에는 정의(Definition), 조작(Manipulation), 제어(Control) 기능이 있다.

- 정의 기능 : 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 형(Type)과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 기능이다.
- 조작 기능 : 데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능이다.
- 제어 기능*
 - ▶ 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 한다.
 - ▶ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 권한(Authority)을 검사할 수 있어야 한다.
 - ▶ 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 병행 제어(Concurrency Control)를 할 수 있어야 한다.

4 DBMS의 장 · 단점

장점	단점
<ul style="list-style-type: none"> • 데이터의 논리적, 물리적 독립성이 보장된다. • 데이터의 중복을 피할 수 있어 기억 공간이 절약된다. • 저장된 자료를 공동으로 이용할 수 있다. • 데이터의 일관성을 유지할 수 있다. • 데이터의 무결성을 유지할 수 있다. • 보안을 유지할 수 있다. • 데이터를 표준화할 수 있다. • 데이터를 통합하여 관리할 수 있다. • 항상 최신의 데이터를 유지한다. • 데이터의 실시간 처리가 가능하다. 	<ul style="list-style-type: none"> • 데이터베이스의 전문가가 부족하다. • 전산화 비용이 증가한다. • 대용량 디스크로의 집중적인 Access로 과부하(Overhead)가 발생한다. • 파일의 예비(Backup*)와 회복(Recovery)이 어렵다. • 시스템이 복잡하다.



데이터의 독립성

데이터의 독립성은 종속성에 대비되는 말로 DBMS의 궁극적 목표이기도 합니다. 데이터의 독립성에는 논리적 독립성과 물리적 독립성이 있습니다.

- **논리적 독립성** : 응용 프로그램과 데이터베이스를 독립시킴으로써, 데이터의 논리적 구조를 변경시키더라도 응용 프로그램은 변경되지 않습니다.
- **물리적 독립성** : 응용 프로그램과 보조기억장치 같은 물리적 장치를 독립시킴으로써, 데이터베이스 시스템의 성능 향상을 위해 새로운 디스크를 도입하더라도 응용 프로그램에는 영향을 주지 않고 데이터의 물리적 구조만을 변경합니다.

DBMS의 제어 기능

무결성, 권한 검사, 병행 제어



전문가의 조언

DBMS의 장 · 단점은 무작정 암기하려 하지 말고 데이터베이스의 정의나 특징을 유지하면서 기존 파일 시스템의 문제점을 해결한 시스템이라는 것을 염두에 두고 이해하세요.

백업(Backup)

백업은 장비 고장 등의 비상사태에도 데이터베이스가 보존되도록 복사하는 작업을 말합니다.



출제예상

1. 다음 중 데이터저장소에 대한 설명으로 옳지 않은 것은?

- ① 논리 데이터저장소는 데이터들을 논리적인 구조로 조직화한 것이다.
- ② 물리 데이터저장소는 논리 데이터저장소의 데이터와 구조를 하드웨어 저장장치에 저장한 것이다.
- ③ 물리 데이터저장소를 구축할 때는 소프트웨어가 운용될 환경의 물리적 특성을 고려해야 한다.
- ④ 데이터저장소의 구축 과정과 데이터베이스의 구축 과정은 상이하다.

논리 데이터저장소를 거쳐 물리 데이터저장소를 구축하는 과정은 데이터베이스를 구축하는 과정과 동일합니다.

이전기술

2. 데이터베이스의 정의 중 '데이터베이스는 어떤 조직의 고유 기능을 수행하기 위해 반드시 필요한 데이터를 의미한다.'에 해당되는 것은?

- ① 통합된 데이터(Integrated Data)
- ② 저장 데이터(Stored Data)
- ③ 운영 데이터(Operational Data)
- ④ 공용 데이터(Shared Data)

조직의 고유한 업무 또는 기능을 수행하기 위한 데이터는 운영 데이터를 가리킵니다. 운영 데이터뿐만 아니라 나머지 3가지 정의의 의미도 알고 있어야 합니다.

이전기술

3. DBMS의 필수 기능 중 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 타입과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 것은?

- ① Manipulation 기능 ② Definition 기능
- ③ Control 기능 ④ Procedure 기능

문제를 잘 읽어보세요. 문제에 답이 들어있습니다.

이전기술

4. DBMS의 필수 기능 중 데이터베이스를 접근하여 데이터의 검색, 삽입, 삭제, 갱신 등의 연산 작업을 위한 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능은?

- ① 정의 기능 ② 조작 기능
- ③ 제어기능 ④ 절차기능

삽입, 삭제, 갱신 하면 조작 기능이라는 것 잊지 않았죠?

이전기술

5. 데이터베이스 관리 시스템(DBMS)의 주요 필수 기능과 거리가 먼 것은?

- ① 데이터베이스 구조를 정의할 수 있는 정의 기능
- ② 데이터 사용자의 통제 및 보안 기능
- ③ 데이터베이스 내용의 정확성과 안정성을 유지할 수 있는 제어 기능
- ④ 데이터 조작어로 데이터베이스를 조작할 수 있는 조작 기능

반드시 암기할 사항이죠. 정의, 조작, 제어!

이전기술

6. 데이터베이스 관리 시스템(DBMS)의 필수 기능 중 제어 기능에 대한 설명으로 거리가 먼 것은?

- ① 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 한다.
- ② 데이터의 논리적 구조와 물리적 구조 사이에 변환이 가능하도록, 두 구조 사이의 사상(Mapping)을 명시하여야 한다.
- ③ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 권한(Authorit)을 검사할 수 있어야 한다.
- ④ 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 병행 제어(Concurrency Control)를 할 수 있어야 한다.

②번은 데이터베이스를 생성하기 위한 정의 기능에 해당됩니다. 제어 기능의 핵심은 무결성, 보안, 권한, 병행 제어입니다.

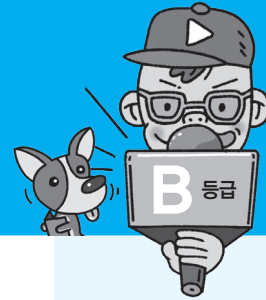
이전기술

7. 데이터베이스 구성의 장점이 아닌 것은?

- ① 데이터 중복 최소화
- ② 여러 사용자에 의한 데이터 공유
- ③ 데이터 간의 종속성 유지
- ④ 데이터 내용의 일관성 유지

DBMS의 발전 배경 두 가지, 아직 잊지 않았죠? 종속성과 중복성의 배제 중요함니다. 꼭 기억하세요.

▶ 정답 : 1. ④ 2. ③ 3. ② 4. ② 5. ② 6. ② 7. ③



1 데이터 입 · 출력의 개요

데이터 입 · 출력은 소프트웨어의 기능 구현을 위해 데이터베이스에 데이터를 입력하거나 데이터베이스의 데이터를 출력하는 작업을 의미한다.

- 데이터 입 · 출력은 단순 입력과 출력뿐만 아니라 데이터를 조작하는 모든 행위를 의미하며, 이와 같은 작업을 위해 SQL(Structured Query Language)을 사용한다.
- 데이터 입 · 출력을 소프트웨어에 구현하기 위해 개발 코드 내에 SQL 코드를 삽입하거나, 객체와 데이터를 연결하는 것을 데이터 접속(Data Mapping)이라고 한다.
- SQL을 통한 데이터베이스의 조작을 수행할 때 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 트랜잭션(Transaction)이라고 한다.

2 SQL(Structured Query Language)

SQL은 1974년 IBM 연구소에서 개발한 SEQUEL에서 유래한다. 국제표준 데이터베이스 언어로, 많은 회사에서 관계형 데이터베이스(RDB)*를 지원하는 언어로 채택하고 있다.

- 관계대수*와 관계해석*을 기초로 한 혼합 데이터 언어이다.
- 질의어*지만 질의 기능만 있는 것이 아니라 데이터 구조의 정의, 데이터 조작, 데이터 제어 기능을 모두 갖추고 있다.
- SQL은 데이터 정의어(DDL), 데이터 조작어(DML), 데이터 제어어(DCL)로 구분된다.
 - 데이터 정의어(DDL; Data Define Language) : SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
 - 데이터 조작어(DML; Data Manipulation Language) : 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다.
 - 데이터 제어어(DCL; Data Control Language) : 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다.

전문가의 조언

데이터 입 · 출력에서는 앞에서 구축한 데이터베이스를 이용하는 기술과 개념을 설명합니다. 자세한 절차와 방법은 3과목에서 다루고 있으니 용어들의 개념만 확실히 파악하고 넘어가세요.

관계형 데이터베이스(RDB, Relational DataBase)

2차원적인 표(Table)를 이용해서 데이터 상호 관계를 정의하는 데이터베이스입니다. 자세한 내용은 Section 078을 참조하세요.

관계대수 / 관계해석

관계대수는 관계형 데이터베이스에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어이고, 관계해석은 관계 데이터의 연산을 표현하는 방법으로, 원하는 정보를 정의할 때는 계산 수식을 사용합니다. 자세한 내용은 Section 081을 참조하세요.

질의어(Query Language)

질의어는 데이터베이스 파일과 범용 프로그래밍 언어를 정확히 알지 못하는 단말 사용자들이 단말기를 통해서 대화식으로 쉽게 DB를 이용할 수 있도록 되어 있는 비절차어의 일종입니다.



3 데이터 접속(Data Mapping)

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

4 트랜잭션(Transaction)

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

- ROLLBACK : 하나의 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성이 깨졌을 때 트랜잭션이 행한 모든 변경 작업을 취소하고 이전 상태로 되돌리는 역사

전문가의 조언

따라잡기

A cartoon illustration of a bald man with a determined expression, wearing a grey tank top and black pants, running to the right. He is chasing a small white dog with black spots. Above them are three balloons with the letters 'A', 'L', and 'U' on them. The background is a simple blue sky with white clouds.

기출문제 따라잡기

Section 036

출제예상

1. 데이터 접속(Data Mapping)에 대한 설명 중 보기에 해당하는 기술은?

- 객체지향 프로그래밍의 객체(Object)와 관계형(Relational) 데이터베이스의 데이터를 연결(Mapping)하는 기술이다.
- 부수적인 코드가 생략되고 SQL 코드를 직접 입력할 필요가 없어 간단하고 직관적인 코드로 데이터를 조작할 수 있다.
- 관련 프레임워크에는 JPA, Hibernate, Django 등이 있다.

- ① ORM ② SQL Mapping
③ JDBC ④ ODBC

객체(Object)와 관계형(Relational)의 데이터를 연결(Mapping)하는 기술은 무엇일까요?

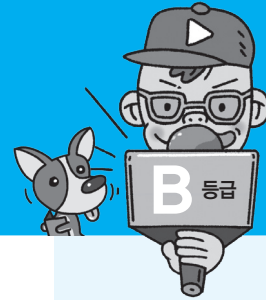
출제예상

2. 트랜잭션(Transaction)에 대한 설명으로 옳지 않은 것은?

- ① 트랜잭션은 작업의 논리적 단위이다.
- ② 트랜잭션을 제어하기 위한 명령어를 TCL이라고 한다.
- ③ 하나의 트랜잭션은 Commit되거나 Rollback되어야 한다.
- ④ Savepoint는 트랜잭션당 한 번만 지정할 수 있다.

Savepoint는 회복 시 참조하는 지점으로, 한 번만 지정할 수 있게 고정되어 있는 것은 아닙니다. 하나의 트랜잭션이 큰 경우 여러 개의 Savepoint를 지정할 수 있습니다.

▶ 정답: 1. ① 2. ④



1 절차형 SQL의 개요

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

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

2 절차형 SQL의 테스트와 디버깅*

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

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

전문가의 조언

절차형 SQL은 데이터베이스 전용의 간단한 프로그래밍이라고 할 수 있습니다. 절차형 SQL의 종류에 대한 세부적인 내용이나 코드들은 3과목에서 다루고 있으니 여기서는 절차형 SQL의 개념과 디버깅, 최적화 등이 어떤 방식으로 수행되는지를 개략적으로 파악해 두세요.

트랜잭션 언어

트랜잭션 언어는 데이터베이스를 조작하고 트랜잭션을 처리하는 언어로, SQL과 TCL이 트랜잭션 언어에 속합니다.

이벤트(Event)

이벤트는 시스템에 어떤 일이 발생한 것을 말하며, 트리거에서 이벤트는 데이터의 입력, 갱신, 삭제와 같이 데이터 조작 작업이 발생했음을 의미합니다.

디버깅(Debugging)

디버깅은 오류를 잡기 위해 소스 코드를 한 줄 한 줄 추적해 가며 변수 값의 변화를 감시하는 작업

구문 오류(Syntax Error)

구문 오류란 잘못된 문법으로 작성된 SQL문을 실행하면 출력되는 오류를 말합니다.

주석(Comment)

주석은 설명을 위해 입력한 부분을 의미합니다. 즉 주석은 사람만 알아볼 수 있으며, 컴파일 되지도 않습니다. 때문에 잠시 사용하지 않을 SQL 코드를 주석으로 처리해 두면 지우지 않고도 해당 코드를 무시하고 SQL 문을 수행할 수 있습니다.



전문가의 조언

SQL이 구조화 질의 언어라는 것을 잊지 않았죠? SQL을 통해 데이터베이스에 정보를 요청하는 것을 질의 또는 쿼리라고 하며, 이것이 좀 더 효율적으로 수행되도록 수정하는 작업을 쿼리 성능 최적화라고 합니다.

APM(Application Performance Management/Monitoring)

APM은 애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구입니다.

옵티마이저(Optimizer)

옵티마이저는 DBMS에 내장되어 작성된 SQL이 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈입니다.



기출문제 따라잡기

Section 037

출제예상

1. 절차형 SQL에 대한 설명으로 옳지 않은 것은?

- ① 절차형 SQL의 종류에는 프로시저, 트리거, 사용자 정의 함수가 있다.
- ② 프로시저는 특정 기능을 수행하는 트랜잭션 언어로, 처리 결과를 단일값으로 반환한다.
- ③ 트리거는 데이터베이스에 이벤트가 발생할 때 수행되는 작업이다.
- ④ 사용자 정의 함수는 프로시저와 유사하며, 예약어 RETURN을 사용하는 것이 특징이다.

프로시저는 처리 결과를 반환하지 않거나 한 개 이상의 값을 반환합니다.

출제예상

2. 데이터베이스에 입력, 갱신, 삭제 등의 데이터 조작과 관련된 이벤트가 발생했을 때 수행되는 절차형 프로시저는?

- ① 프로시저(Procedure)
- ② 사용자 정의 함수(User-Defined Function)
- ③ 트리거(Trigger)
- ④ 예외 처리(Exception Handling)

1번 문제를 풀었다면 이 문제는 쉽게 맞힐 수 있겠죠?

출제예상

3. 절차형 SQL의 생성부터 최적화까지의 과정에 대한 설명으로 거리가 먼 것은?

- ① 절차형 SQL을 생성할 때 오류가 발생했다면 SHOW 명령을 통해 오류 내용을 확인한다.
- ② 절차형 SQL을 실행하기 전에 디버깅을 통해 로직을 검증한다.
- ③ 디버깅 시 데이터베이스의 데이터들이 변경되지 않도록 관련 코드들을 주석으로 처리한다.
- ④ 절차형 SQL의 성능이 느리다면 사용된 SQL 코드 중 가장 긴 SQL 코드의 최적화를 수행한다.

SQL 코드가 길다고 무조건 비효율적인 쿼리라고 볼 수 없습니다. 최적화는 성능 측정 도구인 APM을 사용하여 각 쿼리의 성능을 확인한 후 성능이 떨어지는 쿼리를 대상으로 최적화를 수행합니다.

▶ 정답 : 1. ② 2. ③ 3. ④



1. Which of the following is a linear list in that elements are accessed, created and deleted in a last-in-first-out order?

- ① Queue ② Graph
③ Stack ④ Tree

2. 리스트에서 FIFO(First In First Out)의 특성을 지닌 추상적 자료형으로서, 시작과 끝을 표시하는 두 개의 포인터를 갖는 자료 구조는?

- ① 스택(Stack) ② 그래프(Graph)
③ 큐(Queue) ④ 트리(Tree)

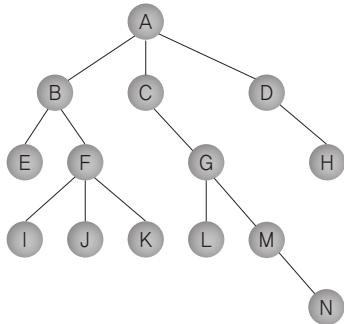
3. 노드의 삽입 작업은 선형 리스트의 한쪽 끝에서, 제거 작업은 다른 쪽 끝에서 수행되는 자료 구조는?

- ① 스택 ② 큐
③ 트리 ④ 그래프

4. 큐(Queue)에 대한 설명으로 옳지 않은 것은?

- ① 입력은 리스트의 한끝에서, 출력은 그 상대방 끝에서 일어난다.
② 운영체제의 작업 스케줄링에 사용된다.
③ 오버플로는 발생될 수 있어도 언더플로는 발생되지 않는다.
④ 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 FIFO 방식으로 처리된다.

5. 다음과 같은 트리(Tree) 구조에서 기본 용어의 설명이 옳은 것은?



- ① Node는 10이다.
② Node의 차수(Degree of Node)는 4이다.
③ 레벨(Level)은 5이다.
④ 근(Root) Node는 N이다.

6. 다음 중 논리 데이터저장소에 대한 설명으로 가장 올바른 것은?

- ① 데이터 및 데이터 간의 연관성, 제약조건을 식별하여 논리적 구조로 조직화한 것이다.
② 소프트웨어가 운용될 환경의 특성을 고려하여 하드웨어 저장장치에 저장한 것이다.
③ 현실 세계에 대한 인식을 추상적 개념으로 표현한 것이다.
④ 파일 구조상의 데이터 항목 또는 데이터 필드들을 의미한다.

7. DBMS(Database Management System)에 대한 설명으로 거리가 먼 것은?

- ① 사용자가 데이터베이스를 용이하게 관리할 수 있도록 지원하는 소프트웨어이다.
② 파일 시스템이 갖는 한계를 극복하기 위해 제안되었다.
③ 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.
④ 데이터베이스의 안정성을 위해 응용 프로그램이 데이터베이스를 공유하는 것을 제한한다.

8. DBMS의 기능에 대한 설명으로 잘못된 것은?

- ① DBMS의 주요 기능은 크게 정의, 조작, 제어 기능으로 분류할 수 있다.
② 정의 기능은 자료형, 이용 방식, 구조 정의, 제약 조건 등을 명시하는 기능이다.
③ 조작 기능에는 데이터를 조작할 때 처리 결과의 정확성을 유지하는 병행 제어 기능이 포함된다.
④ 제어 기능에는 무결성, 보안 유지, 권한 검사 등이 포함된다.

9. 데이터베이스의 정의로 가장 적합한 것은?

- ① 공용 데이터(Shared Data), 통합 데이터(Integrated Data), 통신 데이터(Communication Data), 운영 데이터(Operational Data)
② 공용 데이터(Shared Data), 색인 데이터(Indexed Data), 통신 데이터(Communication Data), 운영 데이터(Operational Data)
③ 공용 데이터(Shared Data), 색인 데이터(Indexed Data), 저장 데이터(Stored Data), 운영 데이터(Operational Data)
④ 공용 데이터(Shared Data), 통합 데이터(Integrated Data), 저장 데이터(Stored Data), 운영 데이터(Operational Data)

▶ 정답 : 1. ③ 2. ③ 3. ② 4. ③ 5. ③ 6. ① 7. ④ 8. ③ 9. ④



10. 데이터베이스는 어느 한 조직의 여러 응용 시스템들이 공유할 수 있도록 통합되고, 저장된 운영 데이터의 집합이라고 정의할 수 있다. 이 정의가 함축하고 있는 의미 중 효율성 증진을 위하여 불가피하게 최소의 중복(Minimal Redundancy) 또는 통제된 중복(Controlled Redundancy)을 허용하는 것으로 설명되는 항목은?

- ① 저장된 데이터(Stored Data)
- ② 공유되는 데이터(Shared Data)
- ③ 통합된 데이터(Integrated Data)
- ④ 운영 데이터(Operational Data)

11. DBMS의 필수 기능 중 정의 기능이 갖추어야 할 요건에 해당 하는 것은?

- ① 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되게 해야 한다.
- ② 데이터와 데이터의 관계를 명확하게 명세할 수 있어야 하며, 원하는 데이터 연산은 무엇이든 명세할 수 있어야 한다.
- ③ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안을 유지하여야 한다.
- ④ 여러 사용자가 데이터베이스를 동시에 접근하여 처리할 때 데이터베이스와 처리 결과가 항상 정확성을 유지하도록 병행 제어를 할 수 있어야 한다.

12. 데이터 입 · 출력에 대한 설명으로 거리가 먼 것은?

- ① 응용 프로그램이 데이터베이스로부터 데이터를 입 · 출력하는 작업을 의미한다.
- ② 데이터를 조작하는 행위를 제외한 소프트웨어와 데이터베이스 간의 데이터 전송만을 의미한다.
- ③ 데이터 입 · 출력에 필요한 일련의 연산들이 포함된 하나의 작업 단위를 트랜잭션이라고 한다.
- ④ 데이터 입 · 출력을 위해 응용 프로그램의 객체와 데이터베이스의 데이터를 연결하는 것을 데이터 매핑이라고 한다.

13. 트랜잭션을 제어하기 위해 사용하는 명령어인 TCL의 종류에 속하지 않는 것은?

- ① COMMIT ② RETURN
- ③ ROLLBACK ④ SAVEPOINT

14. 소프트웨어 개발에서 데이터 입 · 출력에 대한 설명으로 옳지 않은 것은?

- ① 소프트웨어의 기능 구현을 위해 데이터베이스로부터 데이터를 입 · 출력하는 것이다.
- ② 데이터베이스의 데이터를 입 · 출력하기 위해 TCL을 사용한다.
- ③ SQL을 통해 데이터베이스에서 처리되는 하나의 논리적 작업 단위를 트랜잭션이라고 한다.
- ④ 소프트웨어가 데이터베이스에 있는 데이터를 조작하기 위해서는 SQL 매핑 또는 ORM 기술을 사용해야 한다.

15. 다음 SQL(Structured Query Language)에 대한 설명으로 옳지 않은 것은?

- ① SQL은 데이터 구조의 정의, 조작, 제어 기능을 갖춘 혼합 데이터 언어로, DDL, DML, DCL로 구성되어 있다.
- ② DDL은 데이터베이스에 문제가 발생했을 때 복원하는 작업을 수행한다.
- ③ DML은 튜플에 대한 조회, 삽입, 삭제 등의 작업을 수행한다.
- ④ DCL은 보안, 무결성, 병행 제어 등의 작업을 수행한다.

16. 다음 중 절차형 SQL에 대한 설명으로 거리가 먼 것은?

- ① C, Java와 같이 반복, 분기 등의 제어가 포함된 연속적인 작업이 가능하다.
- ② BEGIN~END 형식의 블록 구조로 구성된다.
- ③ DBMS에 저장되어 실행되기 때문에 입 · 출력 패킷이 적다.
- ④ 일반적인 프로그래밍 언어와 다르게 사용자가 직접 함수를 정의하여 사용하는 것은 불가능하다.

17. 절차형 SQL의 테스트에 대한 설명으로 잘못된 것은?

- ① 디버깅과 실행을 통한 결과 검증으로 테스트를 수행한다.
- ② 디버깅 시 데이터베이스에 변화를 주는 코드들은 모두 삭제한 후 변경 내역만을 점검한다.
- ③ 구문 오류나 참조 오류는 생성 시 존재 여부를 확인할 수 있다.
- ④ 생성 및 실행 중에 발생한 오류(Error) 및 경고(Warning)는 SHOW 명령어를 통해 확인할 수 있다.

**1. Section 034**

요소들이 후입선출(Last-In-First-Out)의 순서로 접근, 생성 및 삭제되는 순차 리스트는 스택(Stack)이다.

2. Section 034

- 스택(Stack) : 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조
- 트리(Tree) : 정점(Node, 노드)과 선분(Branch, 가지)을 이용하여 사이클을 이루지 않도록 구성한 그래프(Graph)의 특수한 형태

4. Section 034

오버플로(Overflow)는 큐가 꽉 채워져 있는 상태로 더 이상 자료를 삽입할 수 없는 상태이고, 언더플로(Underflow)는 자료가 없어서 자료를 제거할 수 없는 상태를 말하는 것으로 큐는 오버플로와 언더플로가 모두 발생할 수 있다.

5. Section 034

- 노드는 A, B, C, D, E, F, G, H, I, J, K, L, M, N으로 노드의 수는 14이다.
- 차수(Degree)는 각 노드에서 뻗어 나온 가지의 수를 의미하고, 특정한 노드를 지정하지 않았으므로 트리의 차수로 생각하면 차수는 3이다.
- 근 노드(Root Node)는 트리의 맨 위에 있는 노드인 A이다.

6. Section 035

- 논리 데이터저장소 : 데이터 및 데이터 간의 연관성, 제약 조건을 식별하여 논리적인 구조로 조직화한 것
- 물리 데이터저장소 : 논리 데이터저장소에 저장된 데이터와 구조들을 소프트웨어가 운용될 환경의 물리적 특성을 고려하여 하드웨어적인 저장장치에 저장한 것

7. Section 035

DBMS는 데이터베이스를 이용하고자 하는 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해주는 시스템이다.

8. Section 035

조작 기능은 데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능이다.

9. Section 035**데이터베이스의 정의**

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

10. Section 035

데이터베이스에서 불가피하게 최소의 중복(Minimal Redundancy) 또는 통제된 중복(Controlled Redundancy)을 허용하게 되는 이유는 효율성 증진을 위하여 데이터를 하나로 통합하기 때문에 발생한다. 즉 데이터베이스는 통합된 데이터를 만들기 위해 발생하는 최소의 통제된 중복이다.

11. Section 035

①번은 조작 기능, ③, ④번은 제어 기능에 대한 설명이다.

12. Section 036

데이터 입·출력은 소프트웨어와 데이터베이스 간에 데이터 입·출력뿐만 아니라 SQL의 DML을 이용한 데이터의 조작도 포함된다.

13. Section 036

TCL의 종류에는 변경 내용을 반영하는 COMMIT, 이전 상태로 되돌리는 ROLLBACK, 중간 저장점을 지정하는 SAVEPOINT가 있다.

14. Section 036

데이터베이스의 데이터를 조작하는데 사용하는 언어는 SQL이고, TCL은 트랜잭션을 제어하는데 사용하는 명령어다.

15. Section 036

DDL(데이터 정의어)은 SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다. ②번의 내용은 DCL(데이터 제어어)의 기능이다.



16. Section 037

절차형 SQL 중에는 사용자가 직접 함수를 정의하여 사용하는 사용자 정의 함수가 있다.

17. Section 037

디버깅 시 데이터베이스에 변화를 주는 코드들은 삭제한 후 점검하는 것이 아니라, 주석 처리를 하여 실행되지 않게 한 후 점검이 끝나면 주석 처리만을 해제하여 사용하는 것이 일반적인 방법이다.



2 장

통합 구현

038 단위 모듈 구현 **C** 등급

039 단위 모듈 테스트 **B** 등급

040 개발 자원 도구 **B** 등급



이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 단위 기능 명세서 2. IPC 3. 테스트 케이스 4. 단위 모듈 테스트 5. 테스트 프로세스 6. IDE 7. 빌드 도구
8. 단위 모듈 9. Ant 10. Maven



전문가의 조언

단위 모듈과 단위 기능의 개념을 이해하고 구현 방법과 과정을 확실히 파악하고 넘어가세요.

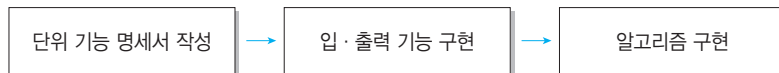
정보 은닉

정보 은닉은 한 모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 것입니다.

1 단위 모듈(Unit Module)의 개요

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

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



2 단위 기능 명세서 작성

단위 기능 명세서는 설계 과정에서 작성하는 기능 및 코드 명세서나 설계 지침과 같이 단위 기능을 명세화한 문서들을 의미한다.

- 단위 기능 명세서를 작성하는 단계에서는 복잡한 시스템을 단순하게 구현하기 위한 추상화 작업이 필요하다.
- 단위 기능 명세서를 작성하는 단계에서는 대형 시스템을 분해하여 단위 기능별로 구분하고, 각 기능들을 계층적으로 구성하는 구조화 과정을 거친다.
- 단위 기능 명세서 작성 시 모듈의 독립적인 운용과 한 모듈 내의 정보가 다른 모듈에 영향을 주지 않도록 정보 은닉*의 원리를 고려한다.

3 입·출력 기능 구현

입·출력 기능 구현 단계에서는 단위 기능 명세서에서 정의한 데이터 형식에 따라 입·출력 기능을 위한 알고리즘 및 데이터를 구현한다.

- 입·출력 기능 구현 단계에서는 단위 모듈 간의 연동 또는 통신을 위한 입·출력 데이터를 구현한다.

- 입·출력 기능 구현 시 사용자 인터페이스인 CLI*, GUI*와의 연동을 고려한다.
- 입·출력 기능 구현 시 네트워크나 외부 장치와의 입·출력은 무료로 공개되어 있는 Open Source* API를 이용하면 간편하게 구현할 수 있다.

잠깐만요



IPC(Inter-Process Communication)

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

- IPC의 대표 메소드 5가지

Shared Memory	다수의 프로세스가 공유 가능한 메모리를 구성하여 프로세스 간 통신을 수행한다.
Socket	네트워크 소켓을 이용하여 네트워크를 경유하는 프로세스들 간 통신을 수행한다.
Semaphores	공유 자원에 대한 접근 제어를 통해 프로세스 간 통신을 수행한다.
Pipes&named Pipes	<ul style="list-style-type: none"> • 'Pipe'라고 불리는 선입선출 형태로 구성된 메모리를 여러 프로세스가 공유하여 통신을 수행한다. • 하나의 프로세스가 Pipe를 이용 중이라면 다른 프로세스는 접근할 수 없다.
Message Queueing	메시지가 발생하면 이를 전달하는 형태로 프로세스 간 통신을 수행한다.

4 알고리즘 구현

알고리즘 구현 단계에서는 입·출력 데이터를 바탕으로 단위 기능별 요구 사항들을 구현 가능한 언어를 이용하여 모듈로 구현한다.

- 알고리즘 구현 단계에서는 구현된 단위 기능들이 사용자의 요구와 일치하는지 확인하는 과정이 필요하다.
- 구현되는 모듈은 단위 기능의 종류에 따라 디바이스 드라이버 모듈, 네트워크 모듈, 파일 모듈, 메모리 모듈, 프로세스 모듈 등으로 구분된다.

디바이스 드라이버 모듈	하드웨어 주변 장치의 동작을 구현한 모듈
네트워크 모듈	네트워크 장비 및 데이터 통신을 위한 기능을 구현한 모듈
파일 모듈	컴퓨터 내부의 데이터 구조 영역에 접근하는 방법을 구현한 모듈
메모리 모듈	파일을 프로세스의 가상 메모리에 매핑/해제하는 방법, 프로세스 사이의 통신 기능을 구현한 모듈
프로세스 모듈	하나의 프로세스 안에서 다른 프로세스를 생성하는 방법을 구현한 모듈

A

B

C

D

CLI(Command Line Interface)

CLI는 Telnet이나 DOS와 같이 키보드를 통해 명령어를 입력받는 사용자 인터페이스(UI)입니다.

GUI(Graphical User Interface)

GUI는 윈도우나 MacOS와 같이 키보드뿐만 아니라 마우스 등의 도구를 통해 화면의 아이콘, 메뉴 등의 다양한 그래픽적 요소 명령을 입력받는 사용자 인터페이스(UI)입니다.

Open Source

Open Source는 일정한 조건을 준수하면 누구나 무료로 사용·수정·재배포가 허가되는 소스 코드입니다.



출제예상

1. 소프트웨어 구현을 위해 필요한 여러 동작 중 한 가지 동작을 수행하는 작은 기능을 모듈로 구현한 것은?

- ① 통합 모듈
- ② 단위 모듈
- ③ 컴포넌트
- ④ 인터페이스

한 가지 동작을 수행하는 작은 기능은 단위 기능을 의미하며, 단위 기능을 구현한 모듈을 단위 모듈이라고 합니다.

출제예상

2. 단위 모듈에 대한 설명으로 옳지 않은 것은?

- ① 처리문, 명령문, 데이터 구조 등이 포함되어 있다.
- ② 사용자나 다른 모듈로부터 값을 제공받아 시작되는 작은 프로그램이라고 할 수 있다.
- ③ 하나의 기능을 구현하므로, 두 개의 모듈을 통합하는 경우 두 개의 기능을 구현할 수 있다.
- ④ 독립적인 컴파일이 불가능하여, 모듈 통합이 이루어진 후에야 컴파일이 가능하다.

단위 모듈은 독립적인 컴파일이 가능하며, 다른 모듈에 호출되거나 삽입될 수도 있습니다.

출제예상

3. 다음 중 단위 모듈을 구현하는 과정에 속하지 않는 것은?

- ① 단위 기능 명세서 작성
- ② 입·출력 기능 구현
- ③ 알고리즘 구현
- ④ 모듈 통합

단위 모듈의 구현 과정은 '단위 기능 명세서 작성' → 입·출력 기능 구현 → 알고리즘 구현 순으로 진행됩니다. 모듈 통합은 각 단위 모듈들이 모두 완성된 후 수행하는 절차입니다.

출제예상

4. 단위 모듈의 구현 과정 중 입·출력 기능 구현에 관한 설명으로 가장 거리가 먼 것은?

- ① 각 장치와의 입·출력은 Open Source API를 통해 간편히 구현할 수 있다.
- ② 완성된 모듈 간의 통신이 원활히 이루어지는지 확인해야 한다.
- ③ 단위 모듈 간의 연동 또는 외부와의 통신을 위한 입·출력 데이터를 구현하는 단계다.
- ④ 사용자 인터페이스가 CLI를 사용하는지, GUI를 사용하는지를 고려해야 한다.

완성된 모듈을 테스트 하려면 먼저 모듈이 완성되어야겠죠. 모듈이 완성되면 알고리즘 구현까지 완료되어야 합니다.

출제예상

5. 단위 모듈의 구현 과정에서 각 모듈과 모듈에 대한 설명 중 틀린 것은?

- ① 디바이스 드라이버 모듈은 하드웨어 주변 장치의 동작을 구현한다.
- ② 파일 모듈은 파일을 프로세스의 가상 메모리에 매핑 또는 해제하는 방법을 구현한다.
- ③ 프로세스 모듈은 하나의 프로세스 안에서 다른 프로세스를 생성하는 방법을 구현한다.
- ④ 네트워크 모듈은 네트워크 장비 및 데이터 통신을 위한 기능을 구현한다.

파일 모듈은 파일을 사용하는 방법을 구현해야 하고, 메모리 모듈은 메모리를 사용하는 방법을 구현해야 합니다. 아닌 것을 찾아보세요!

출제예상

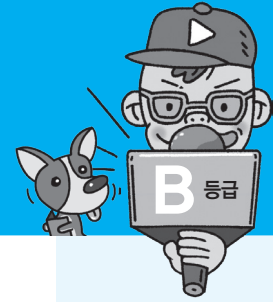
6. 단위 모듈의 데이터 입·출력을 구현하는 과정 중 다음 설명이 의미하는 것은?

- 모듈 간 통신 방식을 구현하기 위해 사용되는 대표적인 프로그래밍 인터페이스 집합이다.
- 복수의 프로세스를 수행하며 이뤄지는 프로세스 간 통신까지 구현이 가능하다.
- 대표적인 메소드로 Shared Memory, Socket, Semaphores 등이 있다.

- ① IPC(Inter-Process Communication)
- ② API(Application Interface)
- ③ Spring
- ④ ORM(Object-Relational Mapping)

모듈 간 통신 또는 프로세스(Process) 간 통신(Communication)을 위한 인터페이스 집합을 의미하는 용어는 무엇일까요?

▶ 정답: 1. ② 2. ④ 3. ④ 4. ② 5. ② 6. ①



1 단위 모듈 테스트의 개요

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

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

2 테스트 케이스(Test Case)

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

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

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



전문가의 조언

테스트 케이스의 개념을 이해하고 테스트 수행 과정을 확실히 파악하고 넘어가세요.

화이트박스/블랙박스 테스트

화이트박스 테스트는 모듈의 소스 코드를 오픈시킨 상태에서 소스 코드의 모든 논리적인 경로를 테스트하는 방법이며, 블랙박스 테스트는 소프트웨어가 수행할 특정 기능이 완전히 작동되는 것을 입증하는 테스트입니다. 자세한 내용은 Section 051을 참조하세요.



전문가의 조언

테스트 케이스는 모듈이 올바르게 작성되었는지 확인하기 위해 모듈에 입력될 수 있는 여러 값들과 예상 결과들을 나열하여 목록을 만드는 과정입니다.

명세 기반 테스트

명세 기반 테스트는 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 구현하고 있는지 확인하는 것으로, 테스트의 수행 증거로도 활용됩니다.

3 테스트 프로세스

테스트 프로세스는 테스트를 위해 수행하는 모든 작업들이 테스트의 목적과 조건을 달성할 수 있도록 도와주는 과정이다.

테스트 프로세스 5단계

- ❶ **계획 및 제어 단계** : 테스트 목표를 달성하기 위한 계획을 수립하고, 계획대로 진행 되도록 제어하는 단계
- ❷ **분석 및 설계 단계** : 테스트 목표를 구체화하여 테스트 시나리오*와 테스트 케이스를 작성하는 단계
- ❸ **구현 및 실행 단계**
 - 효율적인 테스트 수행을 위해 테스트 케이스들을 조합하여 테스트 프로시저*에 명세하는 단계이다.
 - 모듈의 환경에 적합한 단위 테스트 도구를 이용하여 테스트를 수행하는 단계이다.
- ❹ **평가 단계** : 테스트가 계획과 목표에 맞게 수행되었는지 평가하고 기록하는 단계
- ❺ **완료 단계** : 이후의 테스트를 위한 참고 자료 및 테스트 수행에 대한 증거 자료로 활용하기 위해 수행 과정과 산출물을 기록 및 저장하는 단계

테스트 시나리오(Test Scenario)

테스트 시나리오는 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스들을 묶은 집합으로, 테스트 케이스들을 적용하는 구체적인 절차를 명세한 문서를 말합니다.

테스트 프로시저(Test Procedure)

테스트 프로시저는 테스트 케이스의 실행 순서를 의미하며, 테스트 스크립트(Test Script)라고도 불립니다.



기출문제 따라잡기

Section 039

출제예상

1. 단위 모듈 테스트에 대한 설명 중 가장 옳지 않은 것은?

- ❶ 블랙박스 테스트 기법 외에는 사용이 불가능하다.
- ❷ 모듈 통합 이후에는 찾기 어려운 에러들을 간단히 찾을 수 있도록 해준다.
- ❸ 모듈 통합 이후에 나타날 수 있는 논리적 오류들을 찾아 내기는 어렵다.
- ❹ 테스트 케이스를 활용하여 수행할 수 있다.

단위 모듈 테스트는 화이트박스 테스트나 블랙박스 테스트 기법을 사용할 수 있습니다.

출제예상

2. 다음 중 단위 테스트(Unit Test)에 대한 설명으로 옳은 것은?

- ❶ 특별한 환경과 데이터를 갖추지 않아도 테스트가 가능하다.
- ❷ 시스템 수준의 오류를 찾아내는데 적합한 테스트다.
- ❸ 모듈이 의도한 기능을 정확히 수행하는지 확인하기 위한 테스트다.
- ❹ 서브시스템 단위의 테스트를 의미한다.

단위 테스트는 테스트 케이스를 작성하여 모듈 수준의 오류를 찾아내는 테스트 기법입니다.



기출문제 따라잡기

Section 039

출제예상

3. 단위 모듈 테스트를 위한 테스트 케이스(Test Case)에 대한 설명으로 가장 거리가 먼 것은?

- ① 테스트 케이스 작성에 대한 표준은 ISO/IEC/IEEE 29119에 정의되어 있다.
- ② 테스트에 필요한 입력 데이터, 조건, 예상 결과 등을 문서화시킨 것이다.
- ③ 테스트 케이스 작성 없이 테스트를 수행하는 경우 특정 요소에 대한 검증이 누락될 수 있다.
- ④ 테스트 케이스는 프로젝트를 장기화하는 요인에 해당하기 때문에 적절히 수행해야 한다.

테스트는 나중에 발생할 수 있는 오류나 오작동을 막기 위함이지, 오류나 오작동이 발생했을 때 수리하는 비용과 시간을 고려하고 문제를 풀어보세요.

출제예상

4. ISO/IEC/IEEE 29119-3 표준에 따른 테스트 케이스의 구성 요소에 해당하지 않는 것은?

- ① Test Item
- ② Input Specification
- ③ Output Specification
- ④ Expected Result Value

Expected Result Value는 예상 결과값이라는 의미로, 테스트 케이스 수행 시 예상 결과값은 Output Specification에 기입합니다.

출제예상

5. 단위 모듈의 테스트 과정 중 계획/제어 단계에 대한 설명으로 가장 적합한 것은?

- ① 테스트 시나리오와 케이스를 작성하는 단계
- ② 효율적인 테스트 수행을 위해 테스트 케이스를 조합하고, 프로시저를 명세하는 단계
- ③ 테스트 수행 과정 중의 산출물을 기록 및 저장하는 단계
- ④ 테스트에 대한 계획을 수립하고, 테스트가 계획대로 진행될 수 있도록 제어하는 단계

쉬운 문제입니다. 계획과 제어에 관련된 보기를 찾아보세요.

출제예상

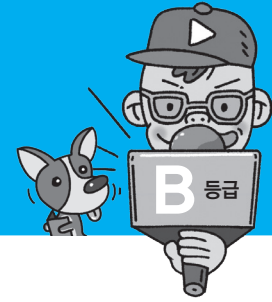
6. 단위 모듈의 테스트를 위한 테스트 프로세스 5단계 중 다음 설명에 해당하는 것은?

- 효율적인 테스트 수행을 위해 테스트 케이스들을 조합한다.
- 조합된 테스트 케이스들을 테스트 프로시저에 명세하여 테스트 수행을 준비한다.
- JUnit, CUnit, NUnit 등의 단위 테스트 도구를 사용하여 테스트를 수행한다.

- ① 계획/제어
- ② 분석/설계
- ③ 구현/실행
- ④ 평가

도구를 사용하여 테스트를 실제로 수행하는 단계를 찾아보세요.

▶ 정답 : 1. ① 2. ③ 3. ④ 4. ④ 5. ④ 6. ③



전문가의 조언

개발을 돕는 도구들에는 어떤 것들이 있는지, 각 도구들의 용도가 무엇인지 정리해 두세요.

전문가의 조언

통합 개발 환경은 공구함과 비교하면 이해가 쉽습니다. 무언가 만 들고, 수리하는데 필요한 망치, 못, 본드, 드라이버 등 모든 공구들을 모아둔 상자와 같죠.

컴파일

컴파일은 개발자가 작성한 고급 언어로 된 프로그램을 컴퓨터가 이해할 수 있는 목적 프로그램으로 번역하여 컴퓨터에서 실행 가능한 형태로 변환하는 작업입니다.

IDE의 외부 서비스 연동

제품의 배포나 버전을 관리하는 서비스나 오픈소스 커뮤니티 등과 연동하여 외부 기능들을 편리하게 사용할 수 있게 합니다.

크로스 플랫폼(Cross Platform)

크로스 플랫폼은 여러 종류의 시스템에서 공통으로 사용될 수 있는 소프트웨어로, 멀티 플랫폼(Multiple Platform)이라고도 불립니다.

전처리(Preprocessing)

전처리는 컴파일에 앞서 코드에 삽입된 주석을 제거하거나 매크로들을 처리하는 과정을 말합니다.

1 통합 개발 환경(IDE)

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

- 통합 개발 환경 도구는 통합 개발 환경을 제공하는 소프트웨어를 의미한다.
- 통합 개발 환경 도구는 코드의 자동 생성 및 컴파일*이 가능하고 추가 기능을 위한 도구들을 다운로드하여 추가할 수 있다.
- 통합 개발 환경 도구는 코드를 실행하거나 테스트할 때 오류가 발생한 부분을 시각화하므로 수정이 용이하다.
- 통합 개발 환경 도구는 외부의 다양한 서비스와 연동*하여 개발에 편의를 제공하고 필요한 정보를 공유할 수 있다.
- 통합 개발 환경을 지원하는 도구는 플랫폼, 운영체제, 언어별로 다양하게 존재하며, 대표적인 도구는 다음과 같다.

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

2 빌드 도구

빌드는 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물을 말한다.

- 빌드 도구는 소스 코드를 소프트웨어로 변환하는 과정에 필요한 전처리(Preprocessing)*, 컴파일(Compile) 등의 작업들을 수행하는 소프트웨어를 말한다.

- 대표적인 도구로는 Ant, Maven, Gradle 등이 있다.

Ant (Another Neat Tool)	<ul style="list-style-type: none"> • 아파치 소프트웨어 재단(Apache Software Foundation)에서 개발한 소프트웨어로, 자바 프로젝트의 공식적인 빌드 도구로 사용되고 있다. • XML* 기반의 빌드 스크립트를 사용하며, 자유도와 유연성이 높아 복잡한 빌드 환경에도 대처가 가능하다. • 정해진 규칙이나 표준이 없어 개발자가 모든 것을 정의하며, 스크립트의 재사용이 어렵다.
Maven	<ul style="list-style-type: none"> • Ant와 동일한 아파치 소프트웨어 재단에서 개발된 것으로, Ant의 대안으로 개발되었다. • 규칙이나 표준이 존재하여 예외 사항만 기록하면 되며, 컴파일과 빌드를 동시에 수행할 수 있다. • 의존성(Dependency)*을 설정하여 라이브러리*를 관리한다.
Gradle	<ul style="list-style-type: none"> • 기존의 Ant와 Maven을 보완하여 개발된 빌드 도구이다. • 한스 도커(Hans Dockter) 외 6인의 개발자가 모여 공동 개발하였다. • 안드로이드 스튜디오의 공식 빌드 도구로 채택된 소프트웨어이다. • Maven과 동일하게 의존성을 활용하며, 그루비(Groovy)* 기반의 빌드 스크립트를 사용한다.

3 기타 협업 도구

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

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

프로젝트 및 일정 관리	<ul style="list-style-type: none"> • 전체 프로젝트와 개별 업무들의 진행 상태, 일정 등을 공유하는 기능을 제공한다. • 종류 : 구글 캘린더(Google Calendar), 분더리스트(Wunderlist), 트렐로(Trello), 지라(Jira), 플로우(Flow) 등
정보 공유 및 커뮤니케이션	<ul style="list-style-type: none"> • 주제별로 구성원들을 지목하여 방을 개설한 후 정보를 공유하고 대화하는 것이 가능하다. • 파일 관리가 간편하고, 의사소통이 자유로운 것이 특징이다. • 종류 : 슬랙(Slack), 잔디(Jandi), 태스크월드(Taskworld) 등
디자인	<ul style="list-style-type: none"> • 디자이너가 설계한 UI나 이미지의 정보들을 코드화하여 개발자에게 전달하는 기능을 제공한다. • 종류 : 스케치(Sketch), 제플린(Zeplin) 등
기타	<ul style="list-style-type: none"> • 아이디어 공유에 사용되는 에버노트(Evernote) • API를 문서화하여 개발자들 간 협업을 도와주는 스웨거(Swagger) • 깃(Git)의 웹호스팅 서비스인 깃허브(GitHub)

XML

XML은 W3C(World Wide Web Consortium)가 채택한 인터넷 표준 언어로, 인터넷 환경에 적합하도록 구성된 메타 언어입니다.

※ 메타 언어 : 프로그램 언어의 규칙을 기술하는데 사용하는 언어

의존성(Dependency)

Maven이나 Gradle에서 라이브러리를 관리할 때 사용하는 명령어로, 빌드 스크립트 안에 사용하고자 하는 라이브러리를 <dependency> 예약어로 등록하면, 빌드 수행 시 인터넷상의 라이브러리 저장소에서 해당 라이브러리를 찾아 코드에 추가해 줍니다.

라이브러리(Library)

라이브러리는 개발 편의를 위해 자주 사용되는 코드, API, 클래스, 값, 자료형 등의 다양한 자원들을 모아놓은 것을 의미합니다.

그루비(Groovy)

그루비는 자바를 기반으로 여러 프로그래밍 언어들의 장점을 모아 만들어진 동적 객체지향 프로그래밍 언어입니다.

출제예상

1. 다음 설명에 해당하는 것은?

편집기, 컴파일러, 디버거 등 개발에 필요한 다양한 툴을 하나의 인터페이스로 통합하여 제공하는 소프트웨어 또는 서비스를 의미한다. 코드를 자동으로 생성해줄 뿐만 아니라 컴파일과정까지 자동으로 수행해주며, 그 밖의 여러 기능들도 다룬로드하여 추가하는 것이 가능하다.

- ① 통합 개발 환경(IDE)
- ② 그룹웨어(Groupware)
- ③ 형상 관리(Configuration Management)
- ④ 빌드 도구(Build Tool)

통합 개발 환경(IDE, Integrated Development Environment)을 벌써 잊었으면
곤란합니다. 틀렸으면 다시 한번 읽어보세요.

출제예상

2. 다음 중 소프트웨어 빌드(Build) 도구에 대한 설명으로 가장 적합한 것은?

- ① 개발자에게 편집기, 컴파일러, 디버거 등의 다양한 기능들을 제공한다.
- ② 개발사 사이트에서 원하는 기능을 다운로드 받아 프로그램에 추가할 수 있다.
- ③ Ant, Maven, Gradle이 대표적인 통합 개발 환경 도구들이다.
- ④ 소스 코드 파일들을 실제 실행할 수 있는 파일로 변환해주는 소프트웨어를 말한다.

빌드는 소스 코드 파일들을 제품 소프트웨어로 변환하는 과정 또는 결과물을 말합니다.

출제예상

3. 다음 중 소프트웨어 빌드(Build) 도구에 대한 설명으로 가장 적합한 것은?

- ① 컴파일 중 오류가 발생하면 이를 시각화하여 수정을 용이하게 한다.
- ② 전처리(preprocessing), 컴파일(compile) 등의 작업을 수행해주는 소프트웨어다.
- ③ 여러 사용자가 서로 다른 작업 환경에서 프로젝트를 원활히 작업할 수 있도록 도와주는 도구이다.
- ④ 제품 개발의 모든 과정 동안 소프트웨어의 버전을 관리해 준다.

소스 코드를 실제 제품 소프트웨어로 만들기 위해서는 전처리, 컴파일 등의 과정이 필요합니다.

출제예상

4. 개발 지원 도구 중 다음 설명에 해당하는 소프트웨어는?

- 안드로이드 스튜디오의 공식 빌드 도구이다.
- 의존성(Dependency)을 활용하여 라이브러리를 관리한다.
- 동적 객체지향 프로그래밍 언어 Groovy를 빌드 스크립트로 사용한다.

- ① Ant ② Maven
③ Zeplin ④ Gradle

Groovy와는 앞에 두 글자가 같은 빌드 도구가 있습니다. 정답을 찾아보세요.

출제예상

5. 여러 사용자가 하나의 프로젝트를 서로 다른 작업 환경에서 원활히 작업할 수 있도록 도와주는 소프트웨어를 의미하는 용어는?

- ① Malware
- ② Groupware
- ③ Ant
- ④ IDE

여러 사용자가 공동으로 작업할 수 있도록 도와주는 소프트웨어는 협업 도구를 뜻합니다. 보기에서 협업 도구를 의미하는 용어를 찾아보세요.

출제예상

6. 다음 중 협업 도구에 대한 설명으로 가장 거리가 먼 것은?

- ① 구성원들 간의 원활한 협업을 위해 사용되는 다양한 소프트웨어를 의미한다.
- ② 외부 라이브러리를 이용할 수 있도록 정의해 놓은 것이다.
- ③ 웹 사이트, PC 프로그램, 스마트폰 앱 등 다양한 형태의 도구들이 존재한다.
- ④ 협업 도구에 대한 조직 구성원들의 학습이 충분하지 않으면 오히려 방해 요소가 될 수 있다.

협력과 관련 없는 보기를 찾아보세요.

▶ 정답: 1. ① 2. ④ 3. ② 4. ④ 5. ② 6. ②



1. 다음 설명이 의미하는 것은?

- 소프트웨어 구현에 필요한 여러 동작 중 한 가지 동작을 수행하는 모듈
- 값을 전달 받아 시작되는 작은 프로그램
- 처리문, 명령문, 데이터 구조 등으로 구성됨

- ① Component
- ② Interface
- ③ Library
- ④ Unit Module

2. 모듈 간 통신 방식을 구현을 위한 인터페이스 집합인 IPC에 속한 메소드가 아닌 것은?

- ① Shared Memory
- ② Qshing
- ③ Semaphores
- ④ Socket

3. IPC(Inter-Process Communication)의 메소드 중 세마포어(Semaphores)의 기능에 대한 설명으로 옳은 것은?

- ① 공유 가능한 메모리를 구성하여 다수의 프로세스 간 통신을 지원한다.
- ② 네트워크 소켓을 이용한 프로세스 간 통신을 지원한다.
- ③ 자원에 대한 접근 제어(access control)를 통해 프로세스 간 통신을 지원한다.
- ④ 메시지 전달 방식을 사용하여 프로세스 간 통신을 지원한다.

4. 파일을 가상 메모리에 매핑 및 해제하거나, 프로세스 사이에서 통신 기능이 원활히 이루어질 수 있도록 구현한 모듈은 단위 기능의 종류에 따라 구분했을 때 어떤 모듈에 속하는가?

- ① 메모리 모듈
- ② 파일 모듈
- ③ 네트워크 모듈
- ④ 디바이스 드라이버 모듈

5. 단위 테스트(Unit Test)의 특징으로 볼 수 없는 것은?

- ① 화이트박스 테스트와 블랙박스 테스트 기법을 사용한다.
- ② 사전 준비 없이 필요할 때 언제든지 테스트 할 수 있다.
- ③ 시스템 수준의 오류는 검사할 수 없다.
- ④ 전체 프로젝트 시간을 단축하는데 도움이 된다.

6. 테스트 케이스(Test Case)의 각 구성 요소에 대한 설명으로 잘못된 것은?

- ① 식별자(Identifier)는 각 항목의 일련번호 등의 구분을 위한 코드를 의미한다.
- ② 입력 명세(Input Specification)에는 입력 데이터 또는 테스트 조건 등을 기입한다.
- ③ 출력 명세(Output Specification)에는 테스트 케이스 수행 후 도출된 결과를 기입한다.
- ④ 환경 설정(Environmental Needs)에는 필요한 하드웨어 및 소프트웨어의 환경을 기입한다.

7. 테스트 시나리오(Test Scenario)에 대한 설명으로 옳은 것은?

- ① 테스트에 필요한 입력 데이터, 테스트 조건, 예상 결과 등을 기입한 문서이다.
- ② 테스트 수행에 필요한 환경이나 테스트 케이스 간의 의존성을 기입한 문서이다.
- ③ 테스트 스크립트(Test Script)라고도 불리며, 테스트 케이스의 실행 순서를 의미한다.
- ④ 여러 개의 테스트 케이스를 묶은 것으로, 테스트 케이스 수행에 대한 절차를 명세한 문서이다.

8. 테스트 프로세스 5단계 중 각 단계에 대한 설명으로 잘못된 것은?

- ① '계획 및 제어'에서는 테스트 목표를 달성하기 위해 계획을 수립하거나 진행을 제어한다.
- ② '분석 및 설계'에서는 테스트 프로시저와 테스트 케이스를 작성한다.
- ③ '평가'에서는 테스트가 계획과 목표에 맞게 수행되었는지 평가하고 기록한다.
- ④ '완료'에서는 수행 과정과 산출물을 기록 및 저장한다.

9. 다음 설명이 의미하는 것은?

Microsoft에서 개발한 통합 개발 환경(IDE) 도구로, Win32 및 Win64 플랫폼을 지원한다. Windows 운영체제를 기반으로 실행되며, Basic, C, C++, C#, 닷넷(.net) 등 다양한 언어를 지원하는 소프트웨어다.

- ① 이클립스(Eclipse)
- ② 비주얼 스튜디오(Visual Studio)
- ③ 엑스 코드(Xcode)
- ④ IDEA

▶ 정답 : 1. ④ 2. ② 3. ③ 4. ① 5. ② 6. ③ 7. ④ 8. ② 9. ②



10. 대표적인 빌드 도구의 하나인 Ant(Another Neat Tool)에 대한 설명으로 잘못된 것은?

- ① 아파치 소프트웨어 재단에서 개발한 소프트웨어로 자바 프로젝트의 공식적인 빌드 도구다.
- ② XML 기반의 빌드 스크립트를 사용한다.
- ③ 정해진 규칙이나 표준이 없어 자유롭게 모든 것을 정의하는 것이 가능하다.
- ④ Dependency를 설정하여 라이브러리를 관리한다.

11. 다음 중 협업 도구(Groupware)에 대한 설명으로 잘못된 것은?

- ① 개발에 참여하는 사람들이 서로 다른 작업 환경에서 원활히 프로젝트를 수행할 수 있도록 도와준다.
- ② 일정 관리, 업무흐름 관리, 정보 공유, 커뮤니케이션 등의 업무를 보조하는 도구들을 말한다.
- ③ PC뿐만 아니라 스마트폰이나 키오스크 등 다양한 플랫폼에서 사용할 수 있도록 제공된다.
- ④ 협업 도구에 익숙하지 않거나 이용할 의지가 없더라도 사용을 강제하는 것이 업무에 도움이 된다.



1. Section 038

- 컴포넌트(Component) : 컴포넌트는 독립적인 업무 또는 기능을 수행하는 단위이며, 실행 코드 기반으로 작성된 모듈
- 인터페이스(Interface) : 서로 다른 두 시스템이나 소프트웨어 등을 서로 이어주는 부분 또는 접속 장치
- 라이브러리(Library) : 개발 편의를 위해 자주 사용되는 코드, API, 클래스, 값, 자료형 등의 다양한 자원들을 모아놓은 것

2. Section 038

IPC(Inter-Process Communication)의 대표적인 메소드에는 Shared Memory, Socket, Semaphores, Pipes&named Pipes, Message Queueing 등이 있다.

3. Section 038

①번은 공유 메모리(Shared Memory), ②번은 소켓(Socket), ④번은 메시지 큐잉(Message Queueing)에 대한 내용이다.

- Shared memory : 다수의 프로세스가 공유 가능한 메모리를 구성하여 프로세스 간 통신을 수행
- Socket : 네트워크 소켓을 이용하여 네트워크를 경유하는 프로세스들 간 통신을 수행
- Message queueing : 메시지가 발생하면 이를 전달하는 형태로 프로세스 간 통신을 수행

4. Section 038

기능별 구현 모듈

- 디바이스 드라이버 모듈 : 하드웨어 주변 장치의 동작을 구현한 모듈
- 네트워크 모듈 : 네트워크 장비 및 데이터 통신을 위한 기능을 구현한 모듈
- 파일 모듈 : 컴퓨터 내부의 데이터 구조 영역에 접근하는 방법을 구현한 모듈
- 메모리 모듈 : 파일을 프로세스의 가상 메모리에 매핑/해제하는 방법과 프로세스 사이의 통신 기능을 구현한 모듈
- 프로세스 모듈 : 하나의 프로세스 안에서 다른 프로세스를 생성하는 방법을 구현한 모듈

5. Section 039

단위 테스트를 수행하기 위해서는 테스트 수행 전에 단독 실행을 위한 환경과 테스트 데이터를 준비해야 한다.

6. Section 039

출력 명세는 예상되는 출력 결과를 입력하는 구성 요소이다.

7. Section 039

- ①, ②번은 테스트 케이스(Test Case), ③번은 테스트 프로시저(Test Procedure)에 대한 내용이다.
- 테스트 케이스 : 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서로, 명세 기반 테스트의 설계 산출물
- 테스트 프로시저 : 테스트 케이스의 실행 순서를 의미하며, 테스트 스크립트(Test Script)라고도 불림

8. Section 039

테스트 프로세스 5단계

- 계획 및 제어 단계 : 테스트 목표를 달성하기 위한 계획을 수립하고, 계획대로 진행되도록 제어하는 단계
- 분석 및 설계 단계 : 테스트 목표를 구체화하여 테스트 시나리오와 테스트 케이스를 작성하는 단계
- 구현 및 실행 단계 : 효율적인 테스트 수행을 위해 테스트 케이스들을 조합하여 테스트 프로시저에 명세하는 단계
- 평가 단계 : 테스트가 계획과 목표에 맞게 수행되었는지 평가하고 기록하는 단계
- 완료 단계 : 이후의 테스트를 위한 참고 자료 및 테스트 수행에 대한 증거 자료로 활용하기 위해 수행 과정과 산출물을 기록 및 저장하는 단계

10. Section 040

Ant는 의존성(Dependency)을 사용하지 않고 직접 라이브러리를 다운받거나 연결시켜서 사용한다.

11. Section 040

협업 도구에 익숙하지 않거나 이용할 의지가 없다면 협업 도구는 오히려 협업의 방해 요소가 될 수 있다.



m·e·m·o

A series of horizontal dotted lines for writing, spanning the width of the page.



3 장

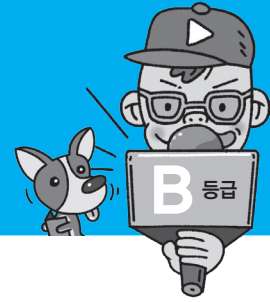
제품 소프트웨어 패키징

- 041 소프트웨어 패키징 **B** 등급
- 042 릴리즈 노트 작성 **B** 등급
- 043 디지털 저작권 관리(DRM) **A** 등급
- 044 소프트웨어 설치 매뉴얼 작성 **C** 등급
- 045 소프트웨어 사용자 매뉴얼 작성 **C** 등급
- 046 소프트웨어 버전 등록 **B** 등급
- 047 소프트웨어 버전 관리 도구 **A** 등급
- 048 빌드 자동화 도구 **B** 등급



이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 소프트웨어 패키징
2. 릴리즈 노트
3. DRM
4. 소프트웨어 설치 매뉴얼
5. 소프트웨어 사용자 매뉴얼
6. 형상 관리
7. Subversion
8. Git
9. Jenkins
10. Gradle



전문의가의 조언

일반적으로 패키징(Packaging)이란 관련된 것들을 하나로 묶는 것을 말하며, 소프트웨어 패키징이란 기능별로 생성한 실행 파일들을 묶어 배포용 설치 파일을 만드는 것을 의미합니다. 소프트웨어를 패키징 할 때는 웹, 모바일, PC 등 소프트웨어가 사용될 단말기의 종류, 윈도우, 유닉스, 안드로이드 등 소프트웨어가 실행될 운영체제의 종류, CPU, RAM 등 하드웨어의 최소 사양 등을 정의하여 패키징을 수행합니다. 이번 섹션에서는 패키징 과정에서 고려할 사항과 패키징 작업 순서를 학습합니다.

모듈화(Modularity)

모듈화란 소프트웨어의 성능을 향상시키거나 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 시스템을 각 기능별로 나누는 것을 말합니다.

UI(User Interface)

여기서의 UI는 사용자가 소프트웨어를 사용하면서 접하게 되는 다양한 화면을 의미합니다.

Managed Service

Managed Service는 고객이 사용 중인 소프트웨어를 24시간 모니터링하면서 문제 발생 시 현장에 바로 출동하여 필요한 점검을 수행하는 등의 체계적인 운영 관리와 유지 보수를 수행하는 서비스입니다.

전문의가의 조언

최근에는 Eclipse, Visual Studio, Xcode, Android Studio 등의 IDE 도구가 프로그램 코딩부터 배포까지 대부분의 과정을 지원하며, 별도의 버전 관리 프로그램을 연동하면 버전 관리 작업까지도 지원하기 때문에 별도의 패키징 도구를 사용하지 않습니다. 패키징된 소프트웨어는 사용자가 직접 다운로드 설치할 수 있도록 웹 사이트나 앱 스토어 등에 등록되어 배포됩니다.

1 소프트웨어 패키징의 개요

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

- 개발자가 아니라 사용자를 중심으로 진행한다.
- 소스 코드는 향후 관리를 고려하여 모듈화*하여 패키징한다.
- 사용자가 소프트웨어를 사용하게 될 환경을 이해하여, 다양한 환경에서 소프트웨어를 손쉽게 사용할 수 있도록 일반적인 배포 형태로 패키징한다.
- 사용자를 중심으로 진행되는 작업이므로 사용자의 편의성 및 실행 환경을 우선적으로 고려해야 한다.

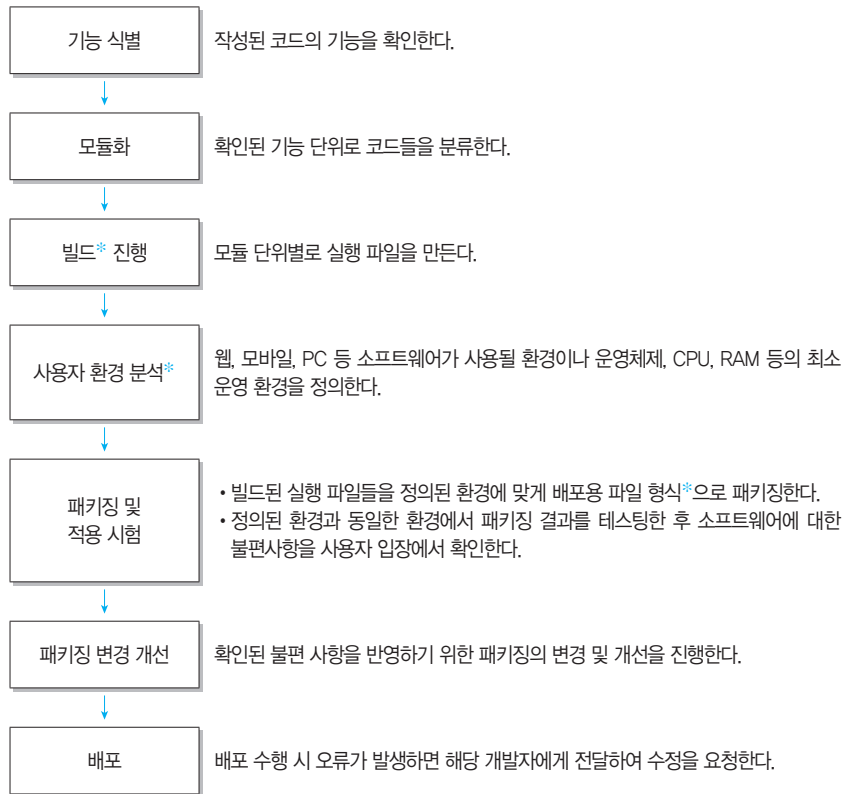
2 패키징 시 고려사항

- 사용자의 시스템 환경, 즉 운영체제(OS), CPU, 메모리 등에 필요한 최소 환경을 정의한다.
- UI(User Interface)*는 사용자가 눈으로 직접 확인할 수 있도록 시각적인 자료와 함께 제공하고 매뉴얼과 일치시켜 패키징한다.
- 소프트웨어는 단순히 패키징하여 배포하는 것으로 끝나는 것이 아니라 하드웨어와 함께 관리될 수 있도록 Managed Service* 형태로 제공하는 것이 좋다.
- 고객의 편의성을 고려한 안정적인 배포가 중요하다.
- 다양한 사용자의 요구사항을 반영할 수 있도록 패키징의 변경 및 개선에 대한 관심을 항상 고려한다.

3 패키징 작업 순서

패키징 주기는 소프트웨어 개발 기법에 따라 달라지는데, 짧은 개발 주기를 반복하는 애자일 기법인 경우에는 보통 2 ~ 4주 내에서 지정하며, 각 주기가 끝날 때마다 패키징을 수행한다.

- 프로젝트 개발 과정에서 주기별로 패키징한 결과물은 테스트 서버에 배포한다.
- 마지막 개발 과정을 거쳐 최종 패키징한 결과물은 고객이 사용할 수 있도록 온라인 또는 오프라인으로 배포한다.
 - 온라인 배포 : 별도로 마련한 운영 서버에 설치 및 사용 매뉴얼과 함께 배포 파일을 등록하여 고객이 직접 다운로드 사용할 수 있도록 한다.
 - 오프라인 배포 : CD-ROM이나 DVD, USB 등에 설치 및 사용 매뉴얼과 함께 배포 파일을 담는다.

**빌드(Build)**

빌드는 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물을 말합니다.

사용자 환경 분석

사용자 실행 환경은 운영체제(OS), 시스템 사양, 사용 방법 등을 최대한 자세하게 구분하여 미리 정의해 놓아야 하며, 실행 환경이 다양한 경우에는 각 환경별로 배포본을 만들기 위해 여러 번의 패키징을 수행해야 합니다. 예를 들면, Windows 10 운영체제는 32비트와 64비트를 선택하여 설치할 수 있도록 Win10_32bit.msi와 Win10_64bit.msi 두 가지 버전으로 패키징하여 배포됩니다.

주요 배포용 파일 형식

- msi : Windows용 패키지 형식
- dmg : Mac OS용 패키지 형식
- jar : java 응용 소프트웨어나 라이브러리를 배포하기 위한 패키지 형식
- war : java Servlet, java Class, xml 및 웹 애플리케이션 서비스를 제공하기 위한 패키지 형식
- ear : jar와 war를 묶어 하나의 애플리케이션 서비스를 제공할 수 있는 패키지 형식
- apk : 안드로이드용 앱 패키지 형식
- ipa : iOS용 앱 패키지 형식

**기출문제 따라잡기****Section 041**

출제예상

1. 다음 중 소프트웨어 패키징에 대한 설명으로 잘못된 것은?

- ① 개발 코드는 향후 관리를 고려하여 모듈화하여 패키징한다.
- ② 패키징은 개발자를 중심으로 진행한다.
- ③ 다양한 사용자의 요구사항을 반영할 수 있도록 패키징의 변경 및 개선에 대한 관리를 항상 고려한다.
- ④ 사용자가 다양한 환경에서 소프트웨어를 손쉽게 사용할 수 있도록 일반적인 배포 형태로 패키징한다.

소프트웨어를 설계하거나 개발할 때 그리고 개발된 소프트웨어를 패키징 할 때 까지도 모든 과정에서 가장 먼저 고려되어야 할 대상은 소프트웨어를 사용할 사용자입니다.

출제예상

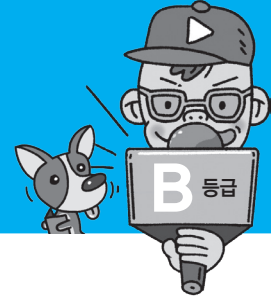
2. 다음에 제시된 패키징 작업들을 순서대로 올바르게 나열한 것은?

- ㄱ. 기능 식별
- ㄴ. 빌드 진행
- ㄷ. 패키징 및 적용 시험
- ㄹ. 사용자 환경 분석
- ㅁ. 모듈화
- ㅂ. 패키징 변경 개선

- ① ㄱ → ㄴ → ㄷ → ㄹ → ㅁ → ㅂ
- ② ㄴ → ㅁ → ㄱ → ㄹ → ㅁ → ㅂ
- ③ ㄹ → ㄱ → ㅁ → ㄴ → ㄷ → ㅂ
- ④ ㄱ → ㅁ → ㄴ → ㄹ → ㄷ → ㅂ

개발 과정을 진행한 후 사용자 환경을 분석하고 패키징을 수행합니다.

▶ 정답 : 1. ② 2. ④



전문가의 조언

앱 스토어나 플레이 스토어에서 스마트폰에 설치할 앱을 고를 때 신경 써 살펴보면 해당 앱에 대한 기능이나 서비스, 사용 환경 혹은 지속적인 업데이트 등에 대한 정보를 확인할 수 있는데, 이와 같이 사용자에게 제공하는 소프트웨어에 대한 정보를 릴리즈 노트라고 합니다. 이번 섹션에서는 릴리즈 노트의 개념과 특징, 중요성, 그리고 릴리즈 노트 작성 시 고려할 사항에 대해 학습합니다. 릴리즈 노트에는 소프트웨어와 관련하여 사용자에게 알려야 할 정보가 기록된다는 것을 염두에 두고 학습할 내용을 읽어보면 어렵지 않게 이해할 수 있습니다. 차분히 읽으면서 정리하세요.

릴리즈(Release)

Release는 '풀어놓다'라는 의미로, 개발이 완성된 소프트웨어를 출시, 즉 배포하는 것을 말합니다.

빌드(Build)

빌드는 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물을 말합니다.

1 릴리즈 노트(Release Note)의 개요

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

- 릴리즈 노트를 통해 테스트 진행 방법에 대한 결과와 소프트웨어 사양에 대한 개발팀의 정확한 준수 여부를 확인할 수 있다.
- 소프트웨어에 포함된 전체 기능, 서비스의 내용, 개선 사항 등을 사용자와 공유할 수 있다.
- 릴리즈 노트를 이용해 소프트웨어의 버전 관리나 릴리즈 정보를 체계적으로 관리할 수 있다.
- 릴리즈 노트는 소프트웨어의 초기 배포 시 또는 출시 후 개선 사항을 적용한 추가 배포 시에 제공한다.
- 소프트웨어의 초기 배포 시 제공되는 릴리즈 노트에서는 소프트웨어에 포함된 기능이나 사용 환경에 대한 내용을 확인할 수 있다.
- 소프트웨어 출시 후 개선된 작업이 있을 때마다 관련 내용을 릴리즈 노트에 담아 제공한다.
- 릴리즈 노트에 정리된 정보들은 철저한 테스트를 거친 것이며, 개발팀에서 제공하는 소프트웨어 사양에 대한 최종 승인을 얻은 후 문서화 되어 제공된다.

2 릴리즈 노트 초기 버전 작성 시 고려사항

릴리즈 노트의 초기 버전은 다음의 사항을 고려하여 작성한다.

- 릴리즈 노트는 정확하고 완전한 정보를 기반으로 개발팀에서 직접 현재 시제로 작성해야 한다.
- 신규 소스, 빌드* 등의 이력이 정확하게 관리되어 변경 또는 개선된 항목에 대한 이력 정보들도 작성되어야 한다.
- 릴리즈 노트 작성에 대한 표준 형식은 없지만 일반적으로 다음과 같은 항목이 포함된다.

항목	내용
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 버그*에 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

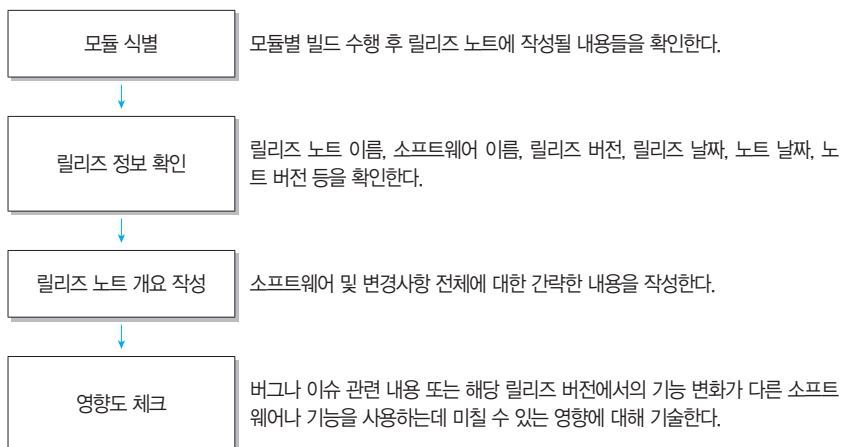
3 릴리즈 노트 추가 버전 작성 시 고려사항

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

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

4 릴리즈 노트 작성 순서

릴리즈 노트는 일반적으로 다음과 같은 순서로 작성한다.

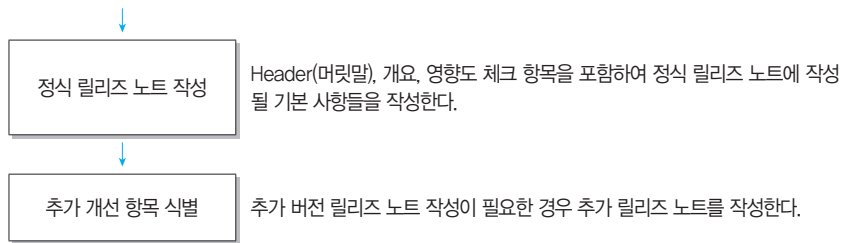


버그(Bug)

버그란 소프트웨어나 하드웨어의 오류 또는 잘못된 동작을 말하며, 이러한 버그를 수정하는 작업을 디버깅(Debugging)이라고 합니다.

베타 버전(Beta Version)

베타 버전은 소프트웨어를 정식으로 출시하기 전에 테스트 할 목적으로 지정된 일부 사용자들에게만 배포하는 시험용 소프트웨어입니다.



기출문제 따라잡기

Section 042

출제예상

1. 다음이 설명하는 것은 무엇인가?

개발 과정에서 소프트웨어가 얼마나 개선되었는지를 정리한 정보를 사용자와 공유하기 위해 작성하는 문서로, 이를 통해 사용자는 소프트웨어에 포함된 서비스나 사용 환경 등을 확인할 수 있다.

- ① 요구사항 명세서(Requirement Specification)
- ② 릴리즈 노트(Release Note)
- ③ 소프트웨어 매뉴얼(Software Manual)
- ④ 소프트웨어 개발 계획서(Software Development Plan)

출시(Release)된 소프트웨어에 대한 정보를 사용자가 확인할 수 있도록 제공하는 문서를 말합니다.

출제예상

2. 다음 중 릴리즈 노트를 통해 확인하거나 수행할 수 있는 내용이 아닌 것은?

- ① 테스트 진행 방법에 대한 결과를 확인할 수 있다.
- ② 소프트웨어 사양에 대한 개발팀의 정확한 준수 여부를 확인할 수 있다.
- ③ 소프트웨어에 포함된 전체 기능, 서비스의 내용, 개선 사항 등을 확인할 수 있다.
- ④ 사용자의 소프트웨어 구매 성향이나 소프트웨어 구매 시 고려사항 등을 확인할 수 있다.

릴리즈 노트는 개발 과정에서 정리된 정보를 사용자와 공유하기 위해 작성한 문서입니다. 개발 과정에서 확인할 수 있는 내용이 아닌 것을 찾아보세요.

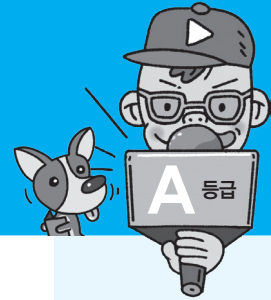
출제예상

3. 다음 중 릴리즈 노트에 대한 설명으로 잘못된 것은?

- ① 릴리즈 노트는 개발팀에서 제공하는 사양에 대한 최종 승인까지 얻은 후 문서화 되어 사용자에게 제공한다.
- ② 릴리즈 노트는 정확하고 완전한 정보를 기반으로 개발팀에서 직접 현재 시제로 작성해야 한다.
- ③ 중대한 오류가 발생하여 이를 긴급하게 수정하는 경우에는 별도로 패키징을 수행해서 재배포를 수행하므로 이와 관련된 릴리즈 노트는 작성하지 않아도 된다.
- ④ 자체적으로 소프트웨어에 대한 기능 업그레이드를 완료한 경우 정식으로 릴리즈 버전을 추가하고 이에 따른 릴리즈 노트를 작성한다.

릴리즈 노트를 추가로 작성할 수 있는 경우를 생각해 보세요. 베타 버전 출시, 긴급한 버그 수정, 업그레이드와 같은 자체 기능 향상, 사용자 요청 등의 특수한 상황이 발생한 경우 릴리즈 노트를 추가로 작성합니다.

▶ 정답 : 1. ② 2. ④ 3. ③



1 저작권의 개요

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

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

2 디지털 저작권 관리의 개요

디지털 저작권 관리(DRM; Digital Right Management)는 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 디지털 콘텐츠의 생성, 유통, 이용까지의 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술이다.

- 원본 콘텐츠가 아날로그인 경우에는 디지털로 변환한 후 패키지(Packager)에 의해 DRM 패키징을 수행한다.
- 콘텐츠의 크기에 따라 음원이나 문서와 같이 크기가 작은 경우에는 사용자가 콘텐츠를 요청하는 시점에서 실시간으로 패키징을 수행하고, 크기가 큰 경우에는 미리 패키징을 수행한 후 배포한다.
- 패키징을 수행하면 콘텐츠에는 암호화된 저작권자의 전자서명이 포함되고 저작권자가 설정한 라이선스 정보가 클리어링 하우스(Clearing House)*에 등록된다.
- 사용자가 콘텐츠를 사용하기 위해서는 클리어링 하우스에 등록된 라이선스 정보를 통해 사용자 인증과 콘텐츠 사용 권한 소유 여부를 확인받아야 한다.
- 종량제 방식*을 적용한 소프트웨어의 경우 클리어링 하우스를 통해 서비스의 실제 사용량을 측정하여 이용한 만큼의 요금을 부과한다.



전문가의 조언

소프트웨어는 고객이 쉽게 설치하고 편리하게 사용할 수 있도록 제공하는 것도 중요하지만 소프트웨어에 대한 불법 사용이나 복제 등을 방지하기 위한 암호화 및 보안 기능을 부가하여 제공해야 합니다. 이번 섹션에서는 소프트웨어를 안전하게 유통할 수 있도록 도와주는 디지털 저작권 관리(DRM) 기술에 대해 학습합니다. 저작권의 개념과 디지털 저작권 관리(DRM)에서 사용되는 용어들을 확실히 정리하고 넘어가세요.

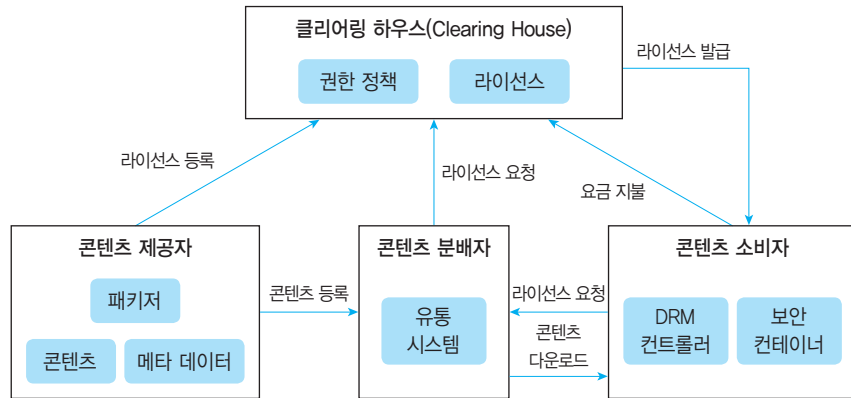
Clearing House

- 디지털 저작권 라이선스의 중개 및 발급을 수행하는 곳으로, 디지털 저작물의 이용 내역을 근거로 저작권료의 정산 및 분배가 수행됩니다.
- 'Clearing'에는 '결제', '청산'이라는 의미도 있으므로 클리어링 하우스란 '결제가 이루어지는 곳'으로 해석할 수도 있습니다.

종량제 방식

종량제 방식이란 실제 사용한 양에 따라 요금을 차등 적용하는 방식을 말합니다.

3 디지털 저작권 관리의 흐름도



- 클리어링 하우스(Clearing House) : 저작권에 대한 사용 권한, 라이선스 발급, 사용량에 따른 결제 관리 등을 수행하는 곳
- 콘텐츠 제공자(Contents Provider) : 콘텐츠를 제공하는 저작권자
- 패키지(Packager) : 콘텐츠를 메타 데이터*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

메타 데이터(Meta Data)

메타 데이터는 데이터에 대한 데이터, 즉 데이터에 대한 속성 정보 등을 설명하기 위한 데이터입니다.

전자 서명(Digital Signature)

전자 서명이란 전자 문서의 변경 여부를 확인할 수 있도록 작성자의 고유 정보를 암호화하여 문서에 포함하는 기술을 말합니다.

크랙(Crack)

크랙이란 '깨다', '부수다'라는 의미 그대로 불법적인 방법으로 소프트웨어에 적용된 저작권 보호 기술을 해제하여 무단으로 사용할 수 있도록 하는 기술이나 도구를 말합니다.

4 디지털 저작권 관리의 기술 요소

디지털 저작권 관리를 위해 사용되는 기술은 다음과 같다.

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



기출문제 따라잡기

Section 043

출제예상

1. 다음이 설명하는 것은 무엇인가?

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

- ① 저작권 ② 사용권
- ③ 저작권 ④ 재산권

저작물에 대해 창작자에게 주어지는 권리? 내용 중에 답이 있죠?

출제예상

2. 디지털 저작권 관리(DRM)에서 사용되는 용어들에 대한 설명으로 틀린 것은?

- ① 패키저(Packager) : 저작권에 대한 사용 권한, 라이선스 발급, 사용량에 따른 결제 관리
- ② DRM 컨트롤러(DRM Controller) : 배포된 콘텐츠의 이용 권한을 통제
- ③ 보안 컨테이너(Security Container) : 콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치
- ④ 콘텐츠 제공자(Contents Provider) : 콘텐츠를 제공하는 저작권자

라이선스 발급이나 결제 관리는 분명하고 명확해야 합니다. Clearing이라는 영문에 결제라는 의미도 있다는 것이 힌트가 될 수 있겠네요.

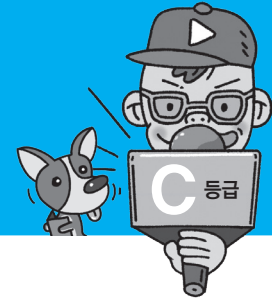
출제예상

3. 다음 중 디지털 저작권 관리(DRM)의 기술 요소가 아닌 것은?

- ① 암호화(Encryption)
- ② 식별 기술(Identification)
- ③ 방화벽(Firewall)
- ④ 정책 관리(Policy Management)

디지털 저작권 관리 기술은 저작물을 보호하기 위해 저작물에 적용하는 기술로 암호화, 키 관리, 암호화 파일 생성, 식별 기술, 정책 관리, 인증 등이 있습니다. 방화벽은 외부의 불법적인 침입으로부터 내부 네트워크를 보호하기 위한 기술입니다.

▶ 정답 : 1. ③ 2. ① 3. ③



전문가의 조언

이번 섹션에서는 사용자가 소프트웨어를 구매한 후 설치하는 과정에서 사용할 매뉴얼 작성 방법에 대해 학습합니다. 설치 매뉴얼에는 설치 과정에 대한 모든 내용이 순서대로 빠짐없이 수록되어야 한다는 것을 염두에 두고 읽어본다면 어렵지 않게 이해할 수 있는 내용들입니다. 가볍게 읽으면서 정리하세요.

1 소프트웨어 설치 매뉴얼의 개요

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

- 설치 매뉴얼은 사용자 기준으로 작성한다.
- 설치 시작부터 완료할 때까지의 전 과정을 빠짐없이 순서대로 설명한다.
- 설치 과정에서 표시될 수 있는 오류 메시지 및 예외 상황에 관한 내용을 별도로 분류하여 설명한다.
- 소프트웨어 설치 매뉴얼에는 목차 및 개요, 서문, 기본 사항 등이 기본적으로 포함되어야 한다.
- 소프트웨어 설치 매뉴얼의 목차에는 전체 설치 과정을 순서대로 요약한 후 관련 내용의 시작 페이지를 함께 기술한다.
- 소프트웨어 설치 매뉴얼의 개요에는 설치 매뉴얼의 주요 특징, 구성과 설치 방법, 순서 등의 내용을 기술한다.

2 서문

서문에는 문서 이력, 설치 매뉴얼의 주석, 설치 도구의 구성, 설치 환경 체크 항목을 기술한다.

- 문서 이력

버전	작성자	작성일	검토자	일시	검수인
v0.1	박개발	2019-04-12	황종근	2019-04-15	박인식
변경 내용	최초 작성				
v1.1	홍진수	2019-04-25	황종근	2019-04-26	박인식
변경 내용	설치 초기 화면과 설치 완료 화면에 사용될 회사 로고 변경				

- **설치 매뉴얼의 주석** : 주의 사항과 참고 사항을 기술한다.
 - 주의 사항 : 소프트웨어를 설치할 때 사용자가 반드시 알고 있어야 하는 중요한 내용을 기술한다.
 - 참고 사항 : 설치에 영향을 미칠 수 있는 사용자의 환경이나 상황에 대한 내용을 기술한다.

- 설치 도구의 구성
 - exe*, dll*, ini*, chm* 등의 설치 관련 파일에 대해 설명한다.
 - 폴더 및 설치 프로그램 실행 파일에 대해 설명한다.
 - 설치 과정 및 결과가 기록되는 log* 폴더에 대해 설명한다.
- 설치 환경 체크 항목

항목	내용
사용자 환경	CPU, Memory, OS(운영체제) 등
응용 프로그램	설치 전 다른 응용 프로그램 종료
업그레이드 버전	업그레이드 이전 버전에 대한 존재 유무 확인
백업 폴더 확인	데이터 저장 폴더를 확인하여 설치 시 폴더를 동기화시킴

3 기본 사항

소프트웨어와 관련하여 기본적으로 설명되어야 할 항목들은 다음과 같다.

항목	설명
소프트웨어 개요	<ul style="list-style-type: none"> • 소프트웨어의 주요 기능 및 UI* 설명 • UI 및 화면 상의 버튼, 프레임 등을 그림으로 설명
설치 관련 파일	<ul style="list-style-type: none"> • 소프트웨어 설치에 필요한 파일 설명 • exe, ini, log 등의 파일 설명
설치 아이콘(Installation)	설치 아이콘 설명
프로그램 삭제	설치된 소프트웨어의 삭제 방법 설명
관련 추가 정보	<ul style="list-style-type: none"> • 소프트웨어 이외의 관련 설치 프로그램 정보 • 소프트웨어 제작사 등의 추가 정보 기술

4 설치 매뉴얼 작성 방법

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

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

설치 화면 및 UI

- 설치 실행과 메인 화면 및 안내창에 대한 내용을 기술한다.
- **설치 실행** : exe 등의 설치(Install) 파일을 실행할 수 있도록 관련 실행 화면에 대한 이미지를 첨부하여 설명한다.
- **메인 화면 및 안내창** : 설치 시 나타나는 메인 화면과 각 과정에서의 안내창에 대한 이미지를 첨부하여 설명한다.

- **exe** : 실행 가능한(executable) 파일의 확장자
- **dll** : 장치의 드라이버 등 프로그램 설치 과정에서 필요한 경우 호출해서 사용하는 동적 링크 라이브러리(dynamic link library) 파일의 확장자
- **ini** : Windows 기반 컴퓨터의 기본 구성 값을 변경해야 하는 경우 사용되는 설정 초기화(initialization) 파일의 확장자
- **chm** : HTML(Microsoft Compiled HTML)로 구성된 도움말 파일의 확장자
- **log** : 프로그램이 실행되는 과정에서 발생하는 오류나 작업 결과 등이 기록된 파일로, 향후 문제 발생 시 이를 진단하기 위한 자료로 사용됨

UI(User Interface)

여기서의 UI는 사용자가 소프트웨어를 사용하면서 접하게 되는 다양한 화면을 의미합니다.

Serial

Serial은 소프트웨어를 구별하기 위해 할당된 일련의 고유한 번호로 숫자 또는 숫자와 문자가 혼합되어 구성됩니다.

설치 이상 메시지 설명

- 설치 방법이나 설치 환경이 잘못된 경우 표시될 수 있는 메시지에 대해 설명한다.

예 시스템 이상이나 xx 파일 오류 등으로 인해 해당 항목에 대한 설치를 진행할 수 없습니다.

1. 해당 원인 및 메시지에 대한 설명과 사용자의 이해를 돕기 위한 오류 코드표를 첨부한다.
2. 설치 이상 메시지 처리 과정에 대한 화면 이미지를 첨부한다.

- 설치 과정별로 참고할 사항이나 주의할 사항에 대한 메모를 추가한다.

설치 완료 및 결과

설치 완료 화면을 수록하여 설치가 정상적으로 마무리되었음을 사용자에게 최종적으로 알려준다.

FAQ

설치 과정에서 사용자가 직면할 수 있는 문제 상황에 대비할 수 있도록, 설치 시 발생할 수 있는 다양한 상황을 FAQ로 정리하여 수록한다.

설치 시 점검 사항

- 설치 전 사용자의 설치 환경에 따라 점검해야 할 사항들이 무엇인지 설명한다.
- 설치에 필요한 사용자 계정 및 설치 권한에 대해 확인할 수 있도록 설명한다.
- 설치 과정에서 오류가 발생할 경우 점검할 수 있는 사항들에 대해 설명한다.

Network 환경 및 보안

- 네트워크 오류로 인해 설치 시 문제가 발생하지 않도록 사전에 필요한 네트워크 연결 상태를 점검하도록 안내한다.
- 보안이나 방화벽으로 인해 설치 시 문제가 발생하지 않도록 관련된 내용을 안내한다.

고객 지원 방법(Customer Support)

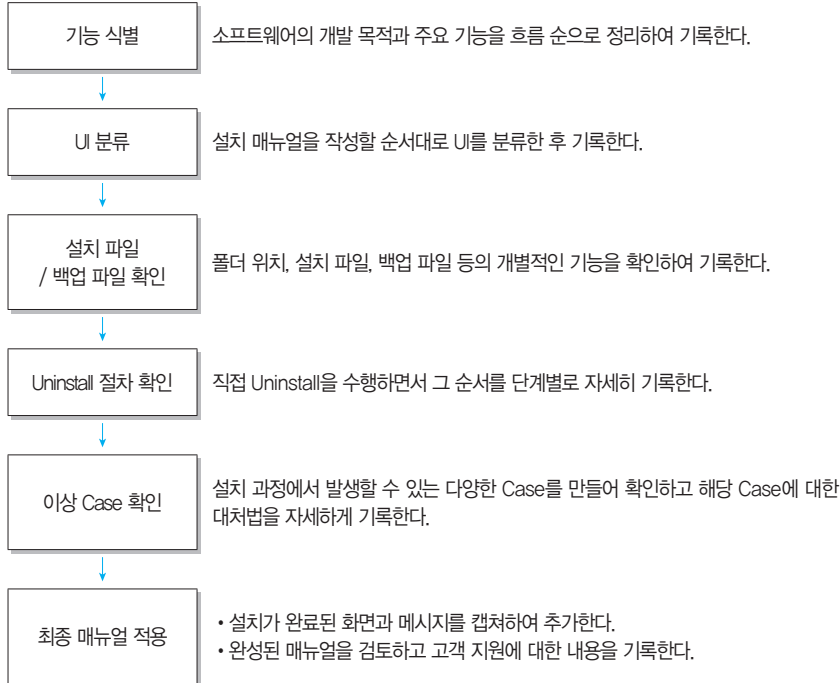
설치와 관련하여 기술적인 지원이나 소프트웨어에 대한 서비스를 원할 경우 국가, 웹사이트, 전화번호, 이메일 등 문의할 수 있는 연락처를 안내한다.

준수 정보 & 제한 보증(Compliance Information & Limited Warranty)

- Serial* 보존, 불법 등록 사용 금지 등에 대한 준수 사항을 안내한다.
- 저작권자 소유권 정보, SW 허가권 정보, 통신 규격, 개발 언어, 연동 프로그램, 문서 효력, 지적 소유권 정보 등과 관련된 내용을 안내한다.

5 설치 매뉴얼 작성 순서

소프트웨어 설치 매뉴얼은 다음과 같은 순서로 작성한다.



기출문제 따라잡기

Section 044

출제예상

1. 소프트웨어 설치 매뉴얼에 대한 설명으로 잘못된 것은?

- ① 설치를 처음 시작할 때부터 완료할 때까지의 과정을 빠짐없이 순서대로 진행한다.
- ② 소프트웨어 설치 매뉴얼은 매뉴얼을 작성하는 개발자의 기준에서 작성한다.
- ③ 설치 과정에서 표시될 수 있는 오류 메시지 및 예외 상황에 관한 내용을 분류하여 설명한다.
- ④ 설치 과정별로 설치 화면과 표시되는 메시지를 순서대로 모두 캡처하여 설명한다.

소프트웨어 설치 매뉴얼은 설치해서 사용하는 사용자의 기준에서 작성합니다.

출제예상

2. 다음에 제시된 소프트웨어 설치 매뉴얼 내용 작성 방법 중 잘못된 것은?

- ① 소프트웨어의 설치 과정을 순서대로 상세히 설명하되, 사용자가 이해하기 쉽도록 설치 화면을 지속적으로 캡처하면서 구성한다.
- ② 소프트웨어의 설치가 정상적으로 완료되면 표시되는 최종 설치 완료 화면을 수록한다.
- ③ 네트워크 환경이나 보안 설정은 사용자의 개인적인 점검 부분들이므로 매뉴얼에 포함하지 않는다.
- ④ 설치 방법이나 설치 환경이 잘못된 경우 표시될 수 있는 메시지에 대해 설명한다.

네트워크 환경, 보안이나 방화벽으로 인해 설치 시 문제가 발생하지 않도록 이와 관련해서 점검할 사항을 매뉴얼에 포함해야 합니다.

▶ 정답 : 1. ② 2. ③



전문가의 조언

이번 섹션에서는 사용자가 소프트웨어를 설치한 후 사용하는 과정에서 필요한 사용자 매뉴얼 작성 방법에 대해 학습합니다. 사용자 매뉴얼에는 사용자가 소프트웨어를 사용하면서 필요한 제반 사항이 모두 포함되도록 작성되어야 한다는 것을 염두에 두고 읽어본다면 어렵지 않게 이해할 수 있는 내용들입니다. 이번 섹션도 가볍게 읽으면서 정리하세요.

패치(Patch)

패치는 이미 제작하여 배포된 프로그램의 오류 수정이나 성능 향상을 위해 프로그램의 일부 파일을 변경하는 것을 말합니다.

컴포넌트(Component)

컴포넌트는 독립적인 업무 또는 기능을 수행하는 단위이며, 실행 코드 기반으로 작성된 모듈입니다.

컴포넌트 명세서

컴포넌트 명세서는 컴포넌트의 개요 및 내부 클래스의 동작, 외부와의 통신 명세 등을 정의한 문서입니다.

컴포넌트 설계서

컴포넌트 설계서는 컴포넌트 구현에 필요한 컴포넌트 구조도, 컴포넌트 목록, 컴포넌트 명세, 인터페이스 명세로 구성된 설계서입니다.

1 소프트웨어 사용자 매뉴얼의 개요

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

- 사용자 매뉴얼은 사용자가 소프트웨어 사용에 필요한 절차, 환경 등의 제반 사항이 모두 포함되도록 작성한다.
- 소프트웨어 배포 후 발생할 수 있는 오류에 대한 패치*나 기능에 대한 업그레이드를 위해 매뉴얼의 버전을 관리한다.
- 개별적으로 동작이 가능한 컴포넌트* 단위로 매뉴얼을 작성한다.
- 사용자 매뉴얼은 컴포넌트 명세서*와 컴포넌트 구현 설계서*를 토대로 작성한다.
- 사용자 매뉴얼에는 목차 및 개요, 서문, 기본 사항 등이 기본적으로 포함되어야 한다.
- 사용자 매뉴얼의 목차에는 매뉴얼 전체 내용을 순서대로 요약한 후 관련 내용의 시작 페이지를 함께 기술한다.
- 사용자 매뉴얼의 개요에는 소프트웨어의 주요 특징, 매뉴얼의 구성과 실행 방법, 사용법, 항목별 점검 기준, 항목별 설정 방법 등에 대한 내용을 기술한다.

2 서문

서문에는 문서 이력, 사용자 매뉴얼의 주석, 기록 보관을 위해 필요한 내용을 기술한다.

- 문서 이력

버전	작성자	작성일	검토자	일시	검수인
v0.1	박정호	2019-04-12	황종근	2019-04-15	박인식
변경 내용	최초 작성				
v1.1	신동석	2019-04-20	황종근	2019-04-21	박인식
변경 내용	제품 등록 방법 변경				

- 사용자 매뉴얼의 주석 : 주의 사항과 참고 사항을 기술한다.
 - 주의 사항 : 소프트웨어를 사용할 때 사용자가 반드시 알고 있어야 하는 중요한 내용을 기술한다.
 - 참고 사항 : 특별한 사용자의 환경이나 상황에 대한 내용을 기술한다.
- 기록 보관 내용
 - 소프트웨어를 사용하면서 필요한 기술 지원이나 추가 정보를 얻기 위한 소프트웨어 등록 정보를 기술한다.

- 소프트웨어 등록 시 필요한 정보는 소프트웨어 명칭, 모델명, 문서 번호*, 제품 번호*, 구입 날짜 등이다.

3 기본 사항

소프트웨어와 관련하여 기본적으로 설명되어야 할 항목들은 다음과 같다.

항목	설명
소프트웨어 개요	<ul style="list-style-type: none"> • 소프트웨어의 주요 기능 및 UI* 설명 • UI 및 화면 상의 버튼, 프레임 등을 그림으로 설명
소프트웨어 사용 환경	<ul style="list-style-type: none"> • 소프트웨어 사용을 위한 최소 환경 설명 • CPU, 메모리 등의 PC 사양, 운영체제(OS) 버전 설명 • 최초 구동에 대한 설명 • 소프트웨어 사용 시 발생할 수 있는 프로그램 충돌이나 개인정보, 보안 등에 관한 주의사항을 설명한다.
소프트웨어 관리	소프트웨어의 사용 종료 및 관리 등에 관한 내용 설명
모델, 버전별 특징	모델 및 버전별로 UI 및 기능의 차이점을 간략하게 요약한다.
기능, 인터페이스의 특징	제품의 기능과 인터페이스의 특징을 간략하게 요약한다.
소프트웨어 구동 환경	<ul style="list-style-type: none"> • 개발에 사용한 언어 및 호환 가능한 운영체제(OS)에 대해 설명한다. • 설치 후 구동하기까지의 과정을 운영체제(OS)별로 설명한다.

4 사용자 매뉴얼 작성 방법

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

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

사용자 화면 및 UI

- 주의 사항과 참고 사항을 기술한다.
- **주의 사항** : 소프트웨어를 사용할 때 사용자가 반드시 알고 있어야 하는 중요한 내용을 설명한다.
- **참고 사항** : 특별한 사용자의 환경이나 상황에 대한 내용을 설명한다.

주요 기능 분류

- 기능이 실행되는 화면을 순서대로 캡처하여 기능에 대한 사용법을 설명한다.
- 기능이 구현되는 과정에서 참고할 사항이나 주의할 사항에 대한 메모를 추가한다.

응용 프로그램(Program) 및 설정(Setting)

- 소프트웨어 구동 시 함께 실행해도 되는 응용 프로그램, 또는 함께 실행되면 안 되는 응용 프로그램에 대해 설명한다.

• **문서 번호** : 릴리즈 번호(Rev), 버전 날짜 등 고유한 문서번호를 기재함

• **제품 번호** : 소프트웨어의 고유한 시리얼 넘버(Serial Number)를 기재함

※ **릴리즈(Release)** : Release는 '풀어놓다'라는 의미로, 개발이 완성된 소프트웨어를 출시, 즉 배포하는 것을 말함

※ **Serial** : 소프트웨어를 구별하기 위해 할당된 일련의 고유한 번호로, 숫자 또는 숫자와 문자가 혼합되어 구성됨

UI(User Interface)

여기서의 UI는 사용자가 소프트웨어를 사용하면서 접하게 되는 다양한 화면을 의미합니다.

- 소프트웨어가 구동될 때 먼저 실행되어야 할 응용 프로그램이 있다면 설명한다.
- 소프트웨어가 정상적으로 구동되기 위한 설정(Setting)이나 기본값에 대해 설명한다.

장치 연동

소프트웨어가 특정 장치(Device)에 내장되는 경우 연동되는 장치(Device)에 대해 설명한다.

Network 환경

Network에 접속되어 사용되는 소프트웨어인 경우 정상적인 연결을 위한 설정값 등을 설명한다.

Profile 안내

- Profile은 소프트웨어의 구동 환경을 점검하는 파일로, 사용자가 Profile의 경로를 변경하거나 위치를 이동하지 않도록 안내한다.
- Profile과 같이 소프트웨어 구동에 필수적인 파일에 대해 설명한다.

고객 지원 방법(Customer Support)

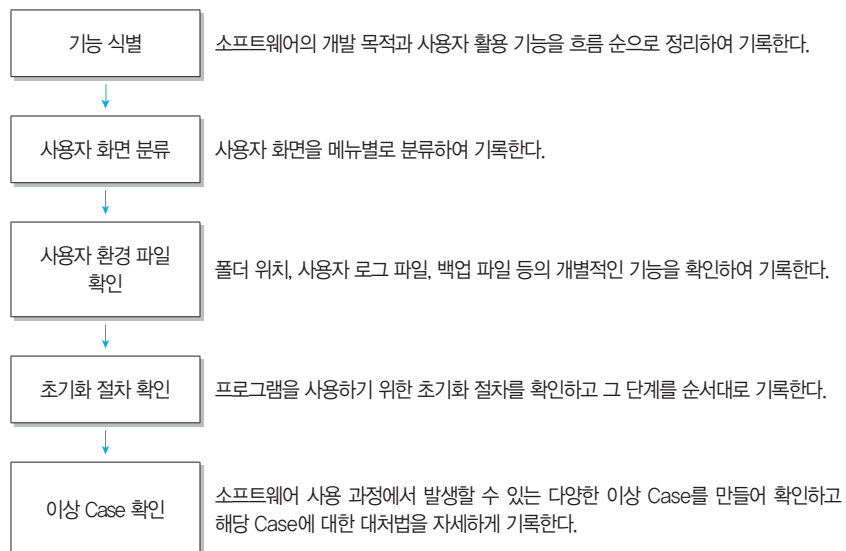
사용과 관련하여 기술적인 지원이나 소프트웨어에 대한 서비스를 원할 경우 국가, 웹사이트, 전화번호, 이메일 등 문의할 수 있는 연락처를 안내한다.

준수 정보 & 제한 보증(Compliance Information & Limited Warranty)

- Serial 보존, 불법 등록 사용 금지 등에 대한 준수 사항을 안내한다.
- 저작권자 소유권 정보, SW 허가권 정보, 통신 규격, 개발 언어, 연동 프로그램, 문서 효력, 지적 소유권 정보 등과 관련된 정보를 안내한다.

5 사용자 매뉴얼 작성 순서

소프트웨어 사용자 매뉴얼은 다음과 같은 순서로 작성한다.



최종 매뉴얼 적용

- 사용과 관련된 문의 답변(FAQ)을 정리하여 기록한다.
- 완성된 매뉴얼을 검토하고 고객 지원에 대한 내용을 기록한다.



기출문제 따라잡기

Section 045

출제예상

1. 소프트웨어 사용자 매뉴얼에 대한 설명으로 잘못된 것은?

- ① 사용자 매뉴얼은 사용자가 설치와 사용에 필요한 절차, 환경 등의 제반 사항 모두가 포함되도록 작성한다.
- ② 사용자가 기술 지원을 받기 위해 소프트웨어를 등록할 때 소프트웨어명, 소프트웨어 모델명, 제품 번호, 구입 날짜 등을 기재할 수 있도록 관련 내용을 사용자 매뉴얼에 포함한다.
- ③ 개별적으로 동작이 가능한 컴포넌트 단위로 매뉴얼이 작성되어야 한다.
- ④ 소프트웨어 구동 환경에 대한 내용은 해당 소프트웨어에 가장 최적화된 운영체제만을 대상으로 설명한다.

소프트웨어 사용자의 환경은 다양하므로 매뉴얼에는 다양한 환경들에 대한 정보가 모두 포함되도록 작성되어야 합니다.

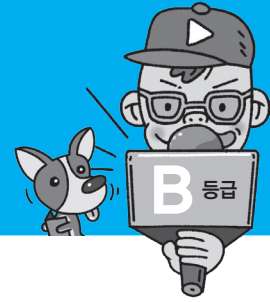
출제예상

2. 다음에 제시된 소프트웨어 사용자 매뉴얼 내용 작성 방법 중 잘못된 것은?

- ① 사용자가 사용 방법을 이해하기 쉽도록 상황별로 누락 없이 캡처하여 순서대로 상세히 설명한다.
- ② 소프트웨어 구동 시 함께 실행해도 되는 응용 프로그램, 또는 함께 실행되면 안 되는 응용 프로그램에 대해 설명한다.
- ③ 소프트웨어가 특정 장치(Device)에 내장되는 경우 연동되는 장치(Device)에 대해 설명한다.
- ④ Profile은 소프트웨어를 사용하는 사용자 정보를 담고 있는 파일로, 사용자가 Profile의 경로를 변경하거나 위치를 이동하지 않도록 안내한다.

Profile은 사용자 정보를 담고 있는 파일이 아니라 소프트웨어 구동에 필요한 환경을 점검하는 파일입니다.

▶ 정답 : 1. ④ 2. ④



전문의가의 조언

고객으로부터 소프트웨어 대한 오류가 접수되면, 개발자는 해당 오류가 어느 단계에서 어떻게 적용되었는지를 확인해야 문제의 실마리를 찾을 수 있습니다. 이와 같이 오류 수정이나 제품의 지속적인 기능 향상을 위해서는 소프트웨어의 변경 내역이 개발 단계에서부터 지속적으로 관리되어야 하는데, 이를 형상 관리 또는 버전 관리라고 합니다. 이번 섹션에서는 형상 관리에 대해 학습합니다. 먼저 형상 관리의 개념을 명확히 잡고 형상 관리의 중요성과 기능 그리고 소프트웨어 버전 등록 시 사용되는 주요 용어들을 정리하세요.

형상

형상이란 소프트웨어 개발 단계의 각 과정에서 만들어지는 프로그램, 프로그램을 설명하는 문서, 데이터 등을 통칭하는 말입니다.

가시성(Visibility)

일반적으로 가시성이란 대상을 확인할 수 있는 정도를 의미합니다.

기준선(Base Line, 변경 통제 사점)

기준선은 정식으로 검토되고 합의된 명세서나 제품으로, 소프트웨어 개발 시 소프트웨어 변경을 적절히 제어할 수 있도록 도와줍니다.

무결성(無缺性)

무결성은 결점이 없다는 것으로, 정해진 기준에 어긋나지 않고 조건을 충실히 만족하는 정도라고 이해할 수 있습니다.

1 소프트웨어 패키징의 형상 관리

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

- 소프트웨어 변경의 원인을 알아내고 제어하며, 적절히 변경되고 있는지 확인하여 해당 담당자에게 통보한다.
- 형상 관리는 소프트웨어 개발의 전 단계에 적용되는 활동이며, 유지보수 단계에서도 수행된다.
- 형상 관리는 소프트웨어 개발의 전체 비용을 줄이고, 개발 과정의 여러 방해 요인이 최소화되도록 보증하는 것을 목적으로 한다.

2 형상 관리의 중요성

- 지속적인 소프트웨어의 변경 사항을 체계적으로 추적하고 통제할 수 있다.
- 제품 소프트웨어에 대한 무절제한 변경을 방지할 수 있다.
- 제품 소프트웨어에서 발견된 버그나 수정 사항을 추적할 수 있다.
- 소프트웨어는 형태가 없어 가시성*이 결핍되므로 진행 정도를 확인하기 위한 기준으로 사용될 수 있다.

3 형상 관리 기능

형상 관리는 품질 보증을 위한 중요한 요소로서 다음과 같은 기능을 수행한다.

- **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(변경 관리)** : 식별된 형상 항목에 대한 변경 요구를 검토하여 현재의 기준선(Base Line)*이 잘 반영될 수 있도록 조정하는 작업
- **형상 감사** : 기준선의 무결성*을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인하는 작업
- **형상 기록(상태 보고)** : 형상의 식별, 통제, 감사 작업의 결과를 기록·관리하고 보고서를 작성하는 작업

4 소프트웨어의 버전 등록 관련 주요 용어

소프트웨어 개발 과정에서 코드와 라이브러리, 관련 문서 등의 버전을 관리하기 위해 자료를 등록하고 갱신하는 과정에서 사용되는 주요 용어와 의미는 다음과 같다.

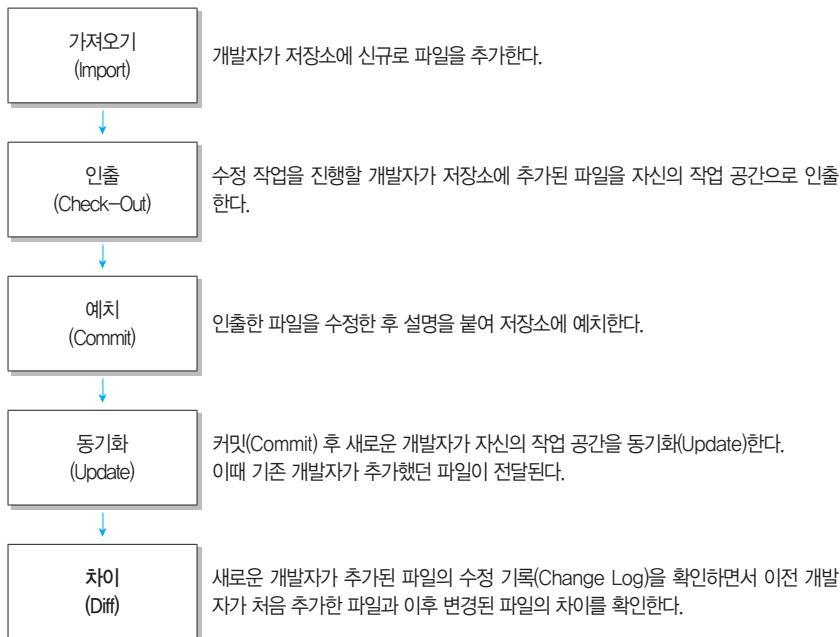
항목	설명
저장소(Repository)	최신 버전의 파일들과 변경 내역에 대한 정보들이 저장되어 있는 곳이다.
가져오기(Import)	버전 관리가 되고 있지 않은 아무것도 없는 저장소(Repository)에 처음으로 파일을 복사한다.
체크아웃(Check-Out)	<ul style="list-style-type: none"> • 프로그램을 수정하기 위해 저장소(Repository)에서 파일을 받아온다. • 소스 파일과 함께 버전 관리를 위한 파일들도 받아온다.
체크인(Check-In)	체크아웃 한 파일의 수정을 완료한 후 저장소(Repository)의 파일을 새로운 버전으로 갱신한다.
커밋(Commit)	체크인을 수행할 때 이전에 갱신된 내용이 있는 경우에는 충돌(Conflict)을 알리고 diff 도구*를 이용해 수정한 후 갱신을 완료한다.
동기화(Update)	저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화한다.

diff 도구

diff 도구는 비교 대상이 되는 파일들의 내용(소스 코드)을 비교하며 서로 다른 부분을 찾아 표시해 주는 도구입니다.

5 소프트웨어 버전 등록 과정

소프트웨어의 버전 등록은 다음과 같은 순서로 진행된다.



전문가의 조언

버전 관리 프로그램에 따라 방법은 다를 수 있지만, diff (commit) <commit2>와 같이 지정하면, 지정한 두 커밋(Commit) 사이의 수정 내역을 확인할 수 있습니다. 이와 같이 이전 개발자들의 수정 내역을 확인하고 싶을 때 diff 명령을 사용합니다.



기출문제 따라잡기

Section 046

출제예상

1. 소프트웨어 형상 관리(Configuration Management)란?

- ① 소프트웨어 구성 항목을 관리하는 것
- ② 개발 과정의 변화되는 사항을 관리하는 것
- ③ 테스트 과정에서 소프트웨어를 통합하는 것
- ④ 개발 인력을 관리하는 것

형상 관리는 변화되는 여러 형상들을 관리하는 것입니다.

출제예상

2. 형상 관리(Configuration Management)의 관리 항목으로 거리가 먼 것은?

- ① 정의 단계의 문서
- ② 개발 단계의 문서와 프로그램
- ③ 유지보수 단계의 변경 사항
- ④ 소프트웨어 개발 비용

형상과 형상 관리의 의미만 알면 풀 수 있는 문제네요. 형상은 소프트웨어 개발 단계의 각 과정에서 만들어지는 프로그램, 프로그램을 설명하는 문서, 데이터 등을 통칭하는 용어이고, 형상 관리는 이러한 형상들의 변경에 대한 관리를 의미합니다.

출제예상

3. 다음 중 형상 관리의 중요성이라고 할 수 없는 것은?

- ① 지속적으로 변경되는 제품 소프트웨어에 대한 개발 통제가 필요하다.
- ② 소프트웨어에 대한 변경은 필요한 경우 제한 없이 언제든지 할 수 있어야 한다.
- ③ 소프트웨어에서 발견된 버그나 수정 사항에 대한 추적이 필요하다.
- ④ 소프트웨어는 형태가 없어 가시성이 결핍되므로 진행을 감시하기 위한 대상이 필요하다.

형상 관리가 되지 않으면 제품 소프트웨어에 대한 무절제한 변경이 난무할 수 있습니다.

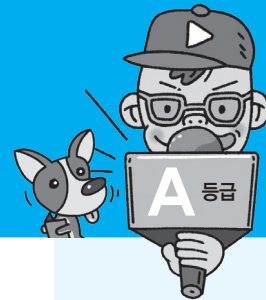
출제예상

4. 다음은 소프트웨어의 버전 등록을 위해 사용하는 주요 용어에 대한 설명이다. 잘못된 것은?

- ① 저장소(Repository) : 최신 버전의 파일들과 변경 내역 정보들이 저장되어 있는 곳
- ② 커밋(Commit) : 버전 관리가 되고 있지 않은 아무것도 없는 저장소(Repository)에 처음으로 파일을 복사함
- ③ 체크아웃(Check-Out) : 저장소(Repository)에서 파일을 받아옴
- ④ 동기화(Update) : 저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화함

커밋(Commit)은 갱신을 완료하는 작업입니다. ②번은 가져오기(Import)에 대한 설명입니다.

▶ 정답 : 1. ② 2. ④ 3. ② 4. ②



1 공유 폴더 방식

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

- 개발자들은 개발이 완료된 파일을 약속된 공유 폴더에 매일 복사한다.
- 담당자는 공유 폴더의 파일을 자기 PC로 복사한 후 컴파일 하여 이상 유무를 확인한다.
- 이상 유무 확인 과정에서 파일의 오류가 확인되면, 해당 파일을 등록한 개발자에게 수정을 의뢰한다.
- 파일에 이상이 없다면 다음날 각 개발자들이 동작 여부를 다시 확인한다.
- 파일을 잘못 복사하거나 다른 위치로 복사하는 것에 대비하기 위해 파일의 변경 사항을 데이터베이스에 기록하여 관리한다.
- 종류에는 SCCS, RCS, PVCS, QVCS 등이 있다.

2 클라이언트/서버 방식

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

- 서버의 자료를 개발자별로 자신의 PC(클라이언트)로 복사하여 작업한 후 변경된 내용을 서버에 반영한다.
- 모든 버전 관리는 서버에서 수행된다.
- 하나의 파일을 서로 다른 개발자가 작업할 경우 경고 메시지를 출력한다.
- 서버에 문제가 생기면, 서버가 복구되기 전까지 다른 개발자와의 협업 및 버전 관리 작업은 중단된다.
- 종류에는 CVS, SVN(Subversion), CVSNT, Clear Case, CMVC, Perforce 등이 있다.

3 분산 저장소 방식

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

- 개발자별로 원격 저장소의 자료를 자신의 로컬 저장소로 복사하여 작업한 후 변경된 내용을 로컬 저장소에서 우선 반영(버전 관리)한 다음 이를 원격 저장소에 반영한다.



전문가의 조언

소프트웨어 버전 관리 도구는 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 공유 폴더 방식, 서버에 저장되어 관리되는 클라이언트/서버 방식, 그리고 하나의 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 분산 저장소 방식으로 분류할 수 있습니다. 먼저 버전 관리 도구의 유형별 특징을 정리하세요. 그리고 버전 관리 도구 중 현업에서 많이 사용되고 있는 Subversion과 Git은 특징뿐만 아니라 주요 명령어의 기능도 정리해 두세요.

CVS(Concurrent Version System)

CVS는 공동 개발을 편리하게 작업할 수 있도록 각종 소스의 버전의 관리를 도와주는 시스템입니다.

trunk

trunk는 '몸통', '줄기'라는 의미로, 개발 과정에서 가장 중심이 되는 디렉터리입니다. trunk 디렉터리 안에 소스 파일과 추가 작업을 위한 서브 디렉터리인 branches 디렉터리가 있습니다.

branches

branch는 '가지', '부문'이라는 의미로, 메인 개발 과정과는 별도로 새로운 기능의 테스트와 같이 추가적인 작업을 수행하기 위한 디렉터리입니다. branches 디렉터리 하위에 작업별로 디렉터를 만들어 그 안에서 개발을 진행합니다. 이후 별도의 디렉터리에서 진행된 개발 결과를 trunk와 병합할 수 있습니다.

리비전(Revision)

리비전은 커밋의 버전으로, 처음 저장소를 만들면 리비전은 001 됩니다. 이후 커밋이 수행될 때마다 리비전이 1씩 증가합니다.

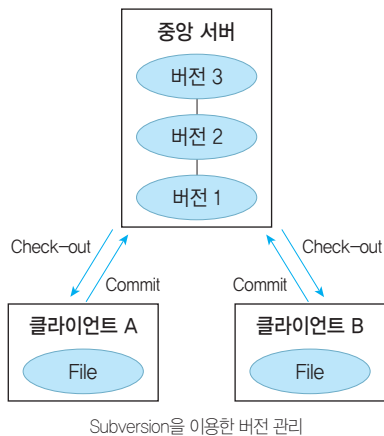
- 로컬 저장소에서 버전 관리가 가능하므로 원격 저장소에 문제가 생겨도 로컬 저장소의 자료를 이용하여 작업할 수 있다.
- 종류에는 Git, GNU arch, DCVS, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM 등이 있다.

4 Subversion(서브버전, SVN)

Subversion은 CVS*를 개선한 것으로, 아파치 소프트웨어 재단에서 2000년에 발표하였다.

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

명령어	의미
add	<ul style="list-style-type: none"> • 새로운 파일이나 디렉터를 버전 관리 대상으로 등록한다. • add로 등록되지 않은 대상은 commit이 적용되지 않는다.
commit	버전 관리 대상으로 등록된 클라이언트의 소스 파일을 서버의 소스 파일에 적용한다.
update	<ul style="list-style-type: none"> • 서버의 최신 commit 이력을 클라이언트의 소스 파일에 적용한다. • commit 전에는 매번 update를 수행하여 클라이언트에 적용되지 않은 서버의 변경 내역을 클라이언트에 적용한다.
checkout	버전 관리 정보와 소스 파일을 서버에서 클라이언트로 받아온다.
lock/unlock	서버의 소스 파일이나 디렉터를 잠그거나 해제한다.
import	아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장하는 명령으로, 한 번 사용하면 다시 사용하지 않는다.
export	버전 관리에 대한 정보를 제외한 순수한 소스 파일만을 서버에서 받아온다.
info	지정된 파일에 대한 위치나 마지막 수정 일자 등에 대한 정보를 표시한다.
diff	지정된 파일이나 경로에 대해 이전 리비전과의 차이를 표시한다.
merge	다른 디렉터리에서 작업된 버전 관리 내역을 기본 개발 작업과 병합한다.



5 Git(깃)

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

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

명령어	의미
add	<ul style="list-style-type: none"> • 작업 내역을 지역 저장소에 저장하기 위해 스테이징 영역(Staging Area)*에 추가한다. • '-all' 옵션으로 작업 디렉터리의 모든 파일을 스테이징 영역에 추가할 수 있다.
commit	작업 내역을 지역 저장소에 저장한다.
branch	<ul style="list-style-type: none"> • 새로운 브랜치를 생성한다. • 최초로 commit을 하면 마스터(master) 브랜치가 생성된다. • commit 할 때마다 해당 브랜치는 가장 최근의 commit한 내용을 가리키게 된다. • '-d' 옵션으로 브랜치를 삭제할 수 있다.



전문가의 조언

Subversion을 이용해 버전 관리 작업을 시작할 때는 먼저 'import' 명령으로 모든 소스 파일을 서버에 등록합니다. 이후 버전 관리는 'checkout → 작업 → add → update → commit' 과정으로 진행합니다. 나머지 명령은 작업이나 자료 송수신 과정에서 필요에 의해 수행합니다.

원격 저장소

원격 저장소는 주로 웹 서버를 빌려 사용하는데, Git 사용자들이 가장 많이 사용하는 웹 호스팅 서비스는 깃 허브(Github.com)입니다. 깃 허브는 오픈소스 프로젝트에 대해서는 무료로 제공하지만 소스를 비공개로 하는 프로젝트에 대해서는 비용을 받습니다.

브랜치(Branch)

Git에서는 저장소가 처음 만들어지면 마스터(Master) 브랜치가 생성되고 이 브랜치에서 기본적인 버전 관리가 진행됩니다. 새로운 기능을 추가하는 작업은 새로운 브랜치를 만들어 작업을 수행하며, 작업이 정상적으로 마무리되면 작업 내역을 마스터 브랜치에 병합합니다. 이렇게 마스터 브랜치와 별도로 생성하는 브랜치를 토픽(Topic) 브랜치 또는 피쳐(Feature) 브랜치라고 합니다. 각각의 브랜치는 다른 브랜치에 영향을 주지 않으므로 독립적인 여러 작업을 동시에 진행할 수 있습니다.

스냅샷(Snapshot)

스냅샷은 영문자와 숫자가 혼합된 40자리 문자열로 표시됩니다.

포인터(Pointer)

포인터는 접근하고자 하는 데이터가 기억되어 있는 위치에 대한 정보를 의미합니다.

스테이징(Staging) 영역

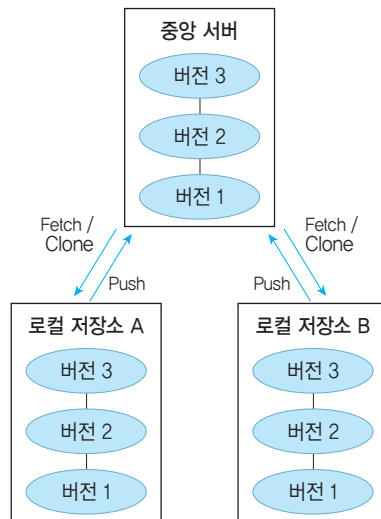
작업 내역을 바로 commit해 지역 저장소에 저장하지 않고 스테이징 영역에 저장했다가 commit을 하는 이유는 스테이징 영역에서 작업 내용을 한 번 더 확인하여 선별적으로 지역 저장소에 반영하기 위함입니다. 이렇게 하면 스테이징 영역을 사용하지 않을 때보다 시간은 더 소요되지만 좀 더 안정된 버전 관리 작업이 가능합니다.



전문가의 조언

Git을 이용해 버전 관리 작업을 시작할 때는 먼저 'init' 명령으로 지역 저장소를 만들고, 'remote add' 명령으로 원격 저장소에 연결한 후 'add -all → commit → push'를 합니다. 이후 버전 관리는 'fetch → 작업 → add → commit → push' 과정으로 진행합니다. 나머지 명령은 작업 과정이나 자료 송수신 과정에서 필요에 의해 수행합니다.

checkout	<ul style="list-style-type: none"> • 지정한 브랜치로 이동한다. • 현재 작업 중인 브랜치는 HEAD 포인터가 가리키는데, checkout 명령을 통해 HEAD 포인터를 지정한 브랜치로 이동시킨다.
merge	지정한 브랜치의 변경 내역을 현재 HEAD 포인터가 가리키는 브랜치에 반영함으로써 두 브랜치를 병합한다.
init	지역 저장소를 생성한다.
remote add	원격 저장소에 연결한다.
push	로컬 저장소의 변경 내역을 원격 저장소에 반영한다.
fetch	원격 저장소의 변경 이력만을 지역 저장소로 가져와 반영한다.
clone	원격 저장소의 전체 내용을 지역 저장소로 복제한다.
fork	지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제한다.



Git을 이용한 버전 관리



기출문제 따라잡기

Section 047

출제예상

1. 다음의 버전 관리 도구에 대한 설명 중 가장 옳지 않은 것은?

- ① 버전 관리 도구는 버전 관리 자료의 저장 위치에 따라 공유 폴더 방식, 클라이언트/서버 방식, 분산 저장소 방식으로 구분할 수 있다.
- ② 클라이언트/서버 방식은 중앙 시스템에 문제가 생겨도 클라이언트의 자료를 이용해 버전 관리 작업을 계속적으로 수행할 수 있다.
- ③ 분산 저장소 방식은 버전 관리 자료가 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리된다.
- ④ 분산 저장소 방식의 대표적인 버전 관리 도구는 Git이고 클라이언트/서버 방식의 대표적인 버전 관리 도구는 SVN(Subversion)이다.

클라이언트/서버 방식은 모든 버전 관리가 중앙 시스템(서버)에서 수행됩니다.

출제예상

2. 다음은 버전 관리 도구인 Subversion에 대한 설명이다. 잘못된 것은?

- ① 클라이언트/서버 구조로, 서버에는 최신 버전과 버전의 변화를 저장한다.
- ② 클라이언트에서는 서버의 자료를 복사해와 작업한 후 변경된 내용을 서버에 반영(Commit)한다.
- ③ 모든 개발 작업은 trunk 디렉터리에서 수행되며, 부가적인 추가 작업은 branches 디렉터리 안에 별도의 디렉터리를 만들어 작업을 완료한 후 trunk 디렉터리의 작업과 병합한다.
- ④ 커밋(Commit)할 때마다 커밋의 버전이라고 할 수 있는 스냅샷(Snapshot)이 일정하게 증가한다.

Subversion과 Git은 커밋이 완료되었을 때 이전 파일과의 차이를 표현되는 방법이 다릅니다. Subversion은 번호를 달리하여 표시하고 Git은 영문자와 숫자가 혼합된 40자리 문자열로 표시합니다.

출제예상

3. 다음은 버전 관리 도구인 Git에 대한 설명이다. 잘못된 것은?

- ① 리누스 토발즈(Linus Torvalds)가 2005년 리눅스 커널 개발에 사용할 관리 도구로 개발한 이후 주니오 하마노(Junio Hamano)에 의해 유지 보수되고 있다.
- ② 분산 버전 관리 시스템으로 지역 저장소와 원격 저장소가 존재한다.
- ③ 원격 저장소나 네트워크에 문제가 있는 경우에도 지역 저장소에서 버전 관리 작업이 가능하므로 장애나 장소에 구애받지 않고 협업이 가능하다.

- ④ 대부분의 버전 관리를 원격 저장소에서 수행할 수 있어 처리 속도가 빠르다.

개발자의 로컬 PC에 있는 지역 저장소와 네트워크를 통해 연결되는 원격 저장소 중 빠른 작업 속도를 제공하는 곳이 어디인지 생각해 보세요.

출제예상

4. 다음은 버전 관리 도구인 Subversion에서 사용하는 명령어들에 대한 설명이다. 잘못된 것은?

- ① add : commit을 수행할 버전 관리 대상을 등록한다.
- ② update : 최신 commit 이력을 소스 파일에 적용한다.
- ③ export : 아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장한다.
- ④ checkout : 서버에서 클라이언트로 버전 관리를 위한 내용과 소스 파일을 받아온다.

export는 서버에서 버전 관리를 위한 내용을 제외한 순수한 소스 파일만 받아오는 명령어고, 아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장하는 명령어는 import입니다.

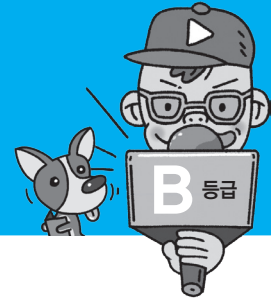
출제예상

5. 다음은 버전 관리 도구인 Git에서 사용하는 명령어들에 대한 설명이다. 잘못된 것은?

- ① branch : 새로운 브랜치를 생성하거나 삭제한다.
- ② push : 원격 저장소의 전체 내용을 지역 저장소로 보낸다.
- ③ merge : 지정한 브랜치의 변경 내역을 현재 HEAD 포인터가 가리키는 브랜치에 반영한다.
- ④ fork : 지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제한다.

push는 로컬 저장소의 변경 내역을 원격 저장소에 반영하는 명령어고, 원격 저장소의 전체 내용을 지역 저장소로 복제하는 명령어는 clone입니다.

▶ 정답 : 1. ② 2. ④ 3. ④ 4. ③ 5. ②



전문가의 조언

빌드 자동화 도구의 개념을 기억하고 대표적인 빌드 자동화 도구인 Jenkins와 Gradle의 특징을 서로 구분할 수 있을 정도로 잘 정리해 두세요.

서블릿 컨테이너

서블릿 컨테이너는 클라이언트의 요청을 처리해 주기 위해 서버 측에서 실행되는 작은 프로그램(Server Side Applet)인 서블릿을 실행하고 서블릿의 생명주기를 관리하는 역할을 합니다.

1 빌드 자동화 도구의 개념

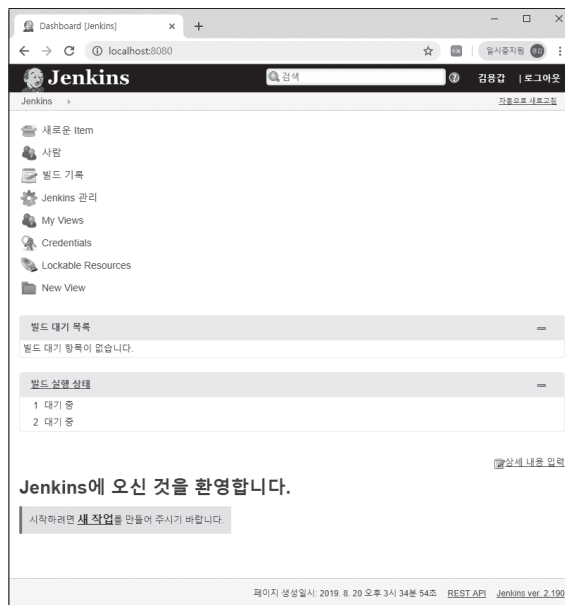
빌드란 소스 코드 파일들을 컴파일한 후 여러 개의 모듈을 묶어 실행 파일로 만드는 과정이며, 이러한 빌드를 포함하여 테스트 및 배포를 자동화하는 도구를 빌드 자동화 도구라고 한다.

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

2 Jenkins

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

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



Jenkins 초기 화면

3 Gradle

Gradle은 Groovy*를 기반으로 한 오픈 소스 형태의 자동화 도구로, 안드로이드 앱 개발 환경에서 사용된다.

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

```

C:\Wstudy\Eclipse-luna\workspace\gradle.java>gradle build
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:jar
:assemble
:compileTestJava UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:check UP-TO-DATE
:build
BUILD SUCCESSFUL

Total time: 12.898 secs
C:\Wstudy\Eclipse-luna\workspace\gradle.java>
  
```

Windows의 명령 프롬프트를 이용한 Gradle 빌드 화면

Groovy

Groovy는 자바에 Python, Ruby, Smalltalk 등의 장점을 결합한 동적 객체 지향 프로그래밍 언어입니다.

DSL(Domain Specific Language)

DSL이란 웹페이지 영역에 특화된어 사용되는 HTML과 같이 특정한도메인, 즉 영역이나 용도에 맞게 기능을 구성한 언어를 말합니다.

스크립트 언어(Script Language)

스크립트 언어는 HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로, 기계어로 컴파일되지 않고 별도의 번역기가 소스를 분석하여 동작하게 하는 언어입니다.



기출문제 따라잡기

Section 048

출제예상

1. 다음 중 빌드 자동화 도구에 대한 설명으로 잘못된 것은?

- ① 빌드 자동화 도구는 지속적인 통합 개발 환경에서 유용하게 활용된다.
- ② 빌드 자동화 도구에는 Ant, Make, Maven, Gradle, Jenkins 등이 있다.
- ③ Jenkins는 Groovy 기반으로 한 오픈 소스로, 안드로이드 앱 개발 환경에서 사용되는 빌드 자동화 도구이다.
- ④ Gradle은 빌드 캐시 기능을 지원하므로 빌드의 속도를 향상시킬 수 있다.

대표적인 빌드 자동화 도구인 Jenkins와 Gradle이 어떤 언어를 기반으로 하는지를 떠올려 보세요.

출제예상

2. 대표적인 빌드 자동화 도구인 Jenkins와 Gradle에 대한 설명으로 잘못된 것은?

- ① 빌드, 테스트, 배포 과정을 자동화하는 도구이다.
- ② Jenkins는 친숙한 Web GUI 제공으로 사용이 쉽다.
- ③ Gradle은 Groovy를 사용해서 만든 DSL을 스크립트 언어로 사용한다.
- ④ Jenkins는 실행할 처리 명령들을 모아 태스크(Task)로 만든 후 태스크 단위로 실행한다.

태스크의 재사용이나 다른 시스템의 태스크를 공유할 수 있는 빌드 캐시 기능이 지원되는 빌드 자동화 도구가 무엇이었죠?

▶ 정답 : 1. ③ 2. ④

**1. 다음 중 패키징 시 고려할 사항이 아닌 것은?**

- ① 사용자의 시스템 환경, 즉 운영체제(OS), CPU, 메모리 등에 필요한 최소 환경을 정의한다.
- ② 초기 반영된 패키징 내용이 변경 및 개선되지 않도록 이에 대한 관리를 항상 고려한다.
- ③ 소프트웨어는 단순히 패키징하여 배포하는 것으로 끝나는 것이 아니라 하드웨어와 함께 관리될 수 있도록 Managed Service 형태로 제공하는 것이 좋다.
- ④ UI(User Interface)는 사용자가 눈으로 직접 확인할 수 있도록 시각적인 자료와 함께 제공하고 매뉴얼과 일치시켜 패키징한다.

2. 다음 중 패키징 작업 과정에 대한 설명으로 잘못된 것은?

- ① 짧은 개발 주기를 반복하는 애자일 기법인 경우 패키징 주기는 보통 2 ~ 4주 내에서 지정하며, 모든 주기가 완료된 후에 최종적으로 패키징을 수행한다.
- ② 패키징한 결과물을 온라인으로 배포할 때는 별도로 마련한 운영 서버에 설치 및 사용 매뉴얼과 함께 배포 파일을 등록하여 고객이 직접 다운받아 사용할 수 있도록 한다.
- ③ 패키징한 결과물을 오프라인으로 배포할 때는 CD-ROM이나 DVD, USB 등에 설치 및 사용 매뉴얼과 함께 배포 파일을 담는다.
- ④ 프로젝트 개발 과정에서 패키징한 결과물은 테스트 서버에 배포한다.

3. 다음 중 패키징 과정에서 수행하는 작업에 대한 설명으로 잘못된 것은?

- ① 모듈화 : 모듈 단위별로 실행 파일을 만든다.
- ② 기능 식별 : 작성된 코드의 기능을 확인한다.
- ③ 적용 시험 : 정의된 환경과 동일한 환경에서 패키징 결과를 테스트한다.
- ④ 배포 : 배포 수행 시 오류가 발생하면 해당 개발자에게 전달하여 수정을 요청한다.

4. 다음은 릴리즈 노트 작성의 일반적인 과정이다. 순서에 맞게 나열한 것은?

- ㉠ 릴리즈 정보 확인
- ㉡ 정식 릴리즈 노트 작성
- ㉢ 모듈 식별
- ㉣ 영향도 체크
- ㉤ 추가 개선 항목 식별
- ㉥ 릴리즈 노트 개요 작성

① ㉠ → ㉡ → ㉢ → ㉣ → ㉤ → ㉥

② ㉡ → ㉠ → ㉢ → ㉣ → ㉥ → ㉤

③ ㉢ → ㉠ → ㉥ → ㉣ → ㉡ → ㉤

④ ㉥ → ㉢ → ㉠ → ㉡ → ㉣ → ㉤

5. 디지털 저작권 관리에 대한 설명으로 잘못된 것은?

- ① 디지털 저작권 관리란 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 관리 및 보호하는 기술이다.
- ② 원본 콘텐츠가 디지털인 경우에는 아날로그로 변환한 후 패키저(Packager)에 의해 DRM 패키징을 수행한다.
- ③ 콘텐츠의 크기가 작은 경우에는 사용자가 콘텐츠를 요청하는 시점에서 실시간으로 패키징을 수행한다.
- ④ 패키징을 수행하면 콘텐츠에는 암호화된 저작권자의 전자서명이 포함되고 저작권자가 설정한 라이선스 정보가 Clearing House에 등록된다.

6. 디지털 저작권 관리(DRM)에서 사용되는 용어들에 대한 설명으로 틀린 것은?

- ① 콘텐츠 분배자(Contents Distributor) : 암호화된 콘텐츠를 유통하는 곳이나 사람
- ② 콘텐츠 소비자(Customer) : 콘텐츠를 구매해서 사용하는 주체
- ③ 패키저(Packager) : 콘텐츠를 메타 데이터와 함께 배포 가능한 형태로 묶어 암호화하는 프로그램
- ④ 클리어링 하우스(Clearing House) : 원본을 안전하게 유통하기 위한 전자적 보안 장치

7. 소프트웨어 설치 매뉴얼 작성에 대한 설명으로 잘못된 것은?

- ① 소프트웨어 설치 매뉴얼은 개발 초기에서부터 적용된 기준이나 사용자가 소프트웨어를 설치하는 과정에 필요한 내용을 기록한 설명서와 안내서이다.
- ② 소프트웨어 설치 매뉴얼에는 목차 및 개요, 서문, 기본 사항 등이 기본적으로 포함되어야 한다.
- ③ 일반적으로 서문에는 문서 이력, 설치 매뉴얼의 주석, 설치 도구의 구성, 설치 환경 체크 항목을 기술한다.
- ④ 설치 과정에서 표시될 수 있는 오류 메시지 및 예외 상황에 관한 내용을 별도로 분류하지 않고 관련 내용에 포함하여 설명한다.



8. 다음 중 소프트웨어 사용자 매뉴얼 작성 과정에서 수행하는 작업에 대한 설명으로 잘못된 것은?

- ① 기능 식별 : 소프트웨어의 개발 목적과 사용자 활용 기능을 흐름 순으로 정리하여 기록한다.
- ② 사용자 환경 파일 확인 : 폴더 위치, 사용자 로그 파일, 백업 파일 등의 개별적인 기능을 확인하여 기록한다.
- ③ 이상 Case 확인 : 소프트웨어 사용 과정에서 발생할 수 있는 다양한 이상 Case를 만들어 확인하고 해당 Case에 대한 대처법을 자세하게 기록한다.
- ④ 최종 매뉴얼 적용 : 프로그램을 사용하기 위한 초기화 절차를 확인하고 그 단계를 순서대로 기록한다.

9. 다음은 소프트웨어의 버전 등록을 위해 사용하는 주요 용어에 대한 설명이다. 잘못된 것은?

- ① 저장소(Repository) : 최신 버전의 파일들과 변경 내역 정보들이 저장되어 있는 곳
- ② 커밋(Commit) : 버전 관리가 되고 있지 않은 아무것도 없는 저장소(Repository)에 처음으로 파일을 복사함
- ③ 체크아웃(Check-Out) : 저장소(Repository)에서 파일을 받아옴
- ④ 동기화(Update) : 저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화함

10. 다음에 제시된 소프트웨어의 일반적인 버전 등록 과정을 순서대로 옳게 나열한 것은?

- ㉠ 가져오기(Import)
- ㉡ 동기화(Update)
- ㉢ 인출(Check-Out)
- ㉣ 차이(Diff)
- ㉤ 예치(Commit)

- ① ㉠ → ㉡ → ㉢ → ㉣ → ㉤
- ② ㉠ → ㉢ → ㉤ → ㉡ → ㉣
- ③ ㉠ → ㉣ → ㉢ → ㉡ → ㉤
- ④ ㉠ → ㉤ → ㉡ → ㉣ → ㉢

11. 다음에서 설명하는 소프트웨어 버전 관리 도구는 무엇인가?

- 버전 관리 자료가 하나의 원격 저장소와 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 방식이다.
- 개발자별로 원격 저장소의 자료를 자신의 로컬 저장소로 복사하여 작업한 후 로컬 저장소에서 우선 버전을 관리할 수 행한 다음 이를 원격 저장소에 반영한다.
- 로컬 저장소에서 버전 관리가 가능하므로 원격 저장소에 문제가 생겨도 로컬 저장소의 자료를 이용해 작업 진행이 가능하다.
- 종류에는 Git, GNU arch, DCVS, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM 등이 있다.

- ① 공유 폴더 방식
- ② 클라이언트/서버 방식
- ③ 분산 저장소 방식
- ④ 하이브리드 방식

12. 소프트웨어 버전 관리 도구인 Git에서 사용하는 명령들에 대한 설명으로 가장 옳지 않은 것은?

- ① add는 작업 내역을 지역 저장소에 저장하기 위해 스테이징 영역에 추가하는 명령어이다.
- ② init는 지정한 브랜치로 이동하는 명령어이다.
- ③ fetch는 원격 저장소의 변경 이력만을 지역 저장소로 가져와 반영하는 명령어이다.
- ④ commit은 작업 내역을 지역 저장소에 저장하는 명령어이다.

**1. Section 041**

패키징의 내용이 변경 및 개선되지 않도록 관리를 고려하는 것이 아니라 다양한 사용자의 요구사항을 반영할 수 있도록 패키징의 변경 및 개선에 대한 관리를 항상 고려해야 한다.

2. Section 041

패키징은 모든 주기가 완료된 후에 최종적으로 수행하는 것이 아니라 각 주기가 끝날 때마다 수행한다.

3. Section 041

- 모듈화 과정은 확인된 기능 단위로 코드를 분류하는 것이다.
- 모듈 단위별로 실행 파일을 만드는 과정은 빌드 진행 과정이다.

5. Section 043

디지털 저작권 관리는 디지털 콘텐츠를 관리 및 보호하는 기술로 원본 콘텐츠가 디지털이 아닌 아날로그인 경우 디지털로 변환한 후 패키지(Packager)에 의해 DRM 패키징을 수행한다.

6. Section 043

- 클리어링 하우스(Clearing House) : 저작권에 대한 사용 권한, 라이선스 발급, 사용량에 따른 결제 관리 등을 수행하는 곳
- 보안 컨테이너(Security Container) : 원본을 안전하게 유통하기 위한 전자적 보안 장치

7. Section 044

설치 과정에서 표시될 수 있는 오류 메시지 및 예외 상황에 관한 내용은 별도로 분류하여 설명한다.

8. Section 045

- ④번의 내용은 초기화 절차 확인 과정에 대한 설명이다.
- 최종 매뉴얼 적용 단계에서는 사용과 관련된 문의 답변(FAQ)을 정리하여 기록하고, 완성된 매뉴얼을 검토한 후 고객 지원에 대한 내용을 기록한다.

9. Section 046

- 커밋(Commit)은 갱신을 완료하는 작업이다.
- ②번의 내용은 가져오기(Import)에 대한 설명이다.

11. Section 047**공유 폴더 방식**

- 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 방식이다.
- 개발자들은 개발이 완료된 파일을 약속된 공유 폴더에 매일 복사한다.
- 담당자는 공유 폴더의 파일을 자기 PC로 복사한 후 컴파일 하여 이상 유무를 확인한다.
- 이상 유무 확인 과정에서 오류가 확인되면, 해당 파일을 등록된 개발자에게 수정을 의뢰한다.
- 작동에 이상이 없다면 다음날 각 개발자들이 동작 여부를 다시 확인한다.
- 파일을 잘못 복사하거나 다른 위치로 복사하는 것에 대비하기 위해 파일의 변경 사항을 데이터베이스에 기록하여 관리한다.
- 종류에는 SCCS, RCS, PVCS, QVCS 등이 있다.

클라이언트/서버 방식

- 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식이다.
- 서버의 자료를 개발자별로 자신의 PC(클라이언트)로 복사해와 작업한 후 변경된 내용을 서버에 반영한다.
- 모든 버전 관리는 서버에서 수행된다.
- 서버에 문제가 생기면, 서버가 복구되기 전까지 다른 개발자와의 협업 및 버전 관리 작업은 중단된다.
- 종류에는 CVS, SVN(Subversion), CVSNT, Clear Case, CMVC, Perforce 등이 있다.

12. Section 047

init는 지역 저장소를 생성하는 명령어고, 지정한 브랜치로 이동하는 명령어는 checkout이다.

4 장

애플리케이션 테스트 관리

049 애플리케이션 테스트 B 등급

050 애플리케이션 테스트의 분류 B 등급

051 테스트 기법에 따른 애플리케이션 테스트 A 등급

052 개발 단계에 따른 애플리케이션 테스트 A 등급

053 통합 테스트 A 등급

054 애플리케이션 테스트 프로세스 B 등급

055 테스트 케이스 / 테스트 시나리오 /
테스트 오라클 B 등급

056 테스트 자동화 도구 B 등급

057 결함 관리 B 등급

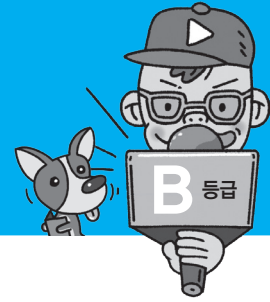
058 애플리케이션 성능 분석 C 등급

059 애플리케이션 성능 개선 C 등급



이 장에서 꼭 알아야 할 키워드 Best 10

1. 시각에 따른 테스트 2. 화이트박스 테스트 3. 블랙박스 테스트 4. 단위 테스트 5. 통합 테스트 6. 하향식 통합 테스트 7. 상향식 통합 테스트 8. 테스트 케이스 9. 테스트 시나리오 10. 테스트 오라클



전문의가의 조언

애플리케이션 테스트는 개발한 애플리케이션이 사용자의 요구를 만족시키는지, 기능이 정상적으로 작동하는지 등을 테스트하는 것입니다. 애플리케이션 테스트의 개념을 기반으로 애플리케이션 테스트의 필요성 및 기본 원리를 숙지해 두세요.

Validation

Validation은 사용자의 입장에서 개발한 소프트웨어가 고객의 요구 사항에 맞게 구현되었는지를 확인하는 것입니다.

Verification

Verification은 개발자의 입장에서 개발한 소프트웨어가 명세서에 맞게 만들어졌는지를 점검하는 것입니다.

전문의가의 조언

'미래창조과학부'는 국내외 시장상황을 반영하여 국내외 주요 상용 소프트웨어를 산업 범용 소프트웨어와 산업 특화 소프트웨어로 구분하고 이를 각각 대분류(16개), 중분류(46개), 소분류(126개)로 분류한 '글로벌 상용 소프트웨어' 분류체계를 발표하였습니다.

1 애플리케이션 테스트의 개념

애플리케이션 테스트는 애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차이다.

- 애플리케이션 테스트는 개발된 소프트웨어가 고객의 요구사항을 만족시키는지 확인(Validation*)하고 소프트웨어가 기능을 정확히 수행하는지 검증(Verification*)한다.
- 애플리케이션 테스트를 실행하기 전에 개발한 소프트웨어의 유형을 분류하고 특성을 정리해서 중점적으로 테스트할 사항을 정리해야 한다.

예 소프트웨어 유형별 특성

소프트웨어명	제공 유형	기능 유형	사용 환경	개발 유형	중점 사항
A. xx오픈DB 구축	서비스 제공 소프트웨어	산업 특화	Web	신규 개발	기능 구현 시 사용자 요구사항의 누락 여부
B. xx통합서비스 구현	서비스 제공 소프트웨어	산업 특화	Web	시스템 통합	기존 시스템과 신규 시스템의 데이터 손실 및 정합성 여부
C. xx오피스	상용 소프트웨어	산업 범용	C/S	신규 개발	다양한 OS환경 지원 여부

참만요



소프트웨어의 분류

소프트웨어(SoftWare)는 하드웨어를 동작시켜 사용자가 작업을 편리하게 수행하도록 하는 프로그램과 자료 구조 등을 총칭하는 것으로, 상용 소프트웨어와 서비스 제공 소프트웨어로 구분됩니다.

- **상용 소프트웨어** : 보통의 사용자들이 공통적으로 필요로 하는 기능을 제공하는 소프트웨어로, 산업의 특성에 따라 산업 범용 소프트웨어와 산업 특화 소프트웨어로 구분됩니다.

산업 범용 소프트웨어	<ul style="list-style-type: none"> • 시스템 소프트웨어 : 하드웨어 전체를 제어하고 운영하는 소프트웨어로, 운영체제, 데이터 관리, 스토리지 소프트웨어, 소프트웨어 공학 도구, 가상화 소프트웨어, 시스템 보안 소프트웨어로 구분됩니다. • 미들웨어 : 응용 프로그램과 운영체제 사이 또는 종류가 다른 두 응용 프로그램 사이에서 보완해 주고 연결해 주는 소프트웨어로, 분산 시스템 소프트웨어, IT 자원 관리, 서비스 플랫폼, 네트워크 보안 소프트웨어로 구분됩니다. • 응용 소프트웨어 : 특정 업무를 처리하기 위한 소프트웨어로 영상 처리, CG/VR, 콘텐츠 배포, 자연어 처리, 음성 처리, 기업용 소프트웨어로 구분됩니다.
산업 특화 소프트웨어	특정 분야에서 요구하는 기능만을 구현한 소프트웨어로, 자동차, 항공, 조선, 건설, 패션 의류, 농업, 의료, 국방, 공공 분야 등을 지원하는 소프트웨어가 있습니다.

- 서비스 제공 소프트웨어 : 소프트웨어를 개발하여 판매하려는 것이 아니라 특정 사용자가 필요로 하는 기능만을 구현해서 제공하는 소프트웨어입니다.

신규 개발 소프트웨어	새로운 서비스를 제공하기 위해 개발된 소프트웨어
기능 개선 소프트웨어	사용자 편의성, 응답 속도, 화면 UI(User Interface), 업무 프로세스 등 기존 서비스 기능을 개선하기 위해 개발된 소프트웨어
추가 개발 소프트웨어	업무나 산업 환경의 변화, 법이나 제도의 개정 등으로 인해 기존 시스템에 새로운 기능을 추가하기 위해 개발된 소프트웨어
시스템 통합 소프트웨어	시스템별로 서비스되던 것을 원스톱(One-Stop) 서비스로 제공하기 위해 업무 기능이나 데이터 등을 통합하여 개발한 소프트웨어

2 애플리케이션 테스트의 필요성

- 애플리케이션 테스트를 통해 프로그램 실행 전에 오류를 발견하여 예방할 수 있다.
- 애플리케이션 테스트는 프로그램이 사용자의 요구사항이나 기대 수준 등을 만족시키는지 반복적으로 테스트하므로 제품의 신뢰도를 향상시킨다.
- 애플리케이션의 개발 초기부터 애플리케이션 테스트를 계획하고 시작하면 단순한 오류 발견뿐만 아니라 새로운 오류의 유입도 예방할 수 있다.
- 애플리케이션 테스트를 효과적으로 수행하면 최소한의 시간과 노력으로 많은 결함을 찾을 수 있다.

3 애플리케이션 테스트의 기본 원리

- 애플리케이션 테스트는 소프트웨어의 잠재적인 결함을 줄일 수 있지만 소프트웨어에 결함이 없다고 증명할 수는 없다. 즉 완벽한 소프트웨어 테스트는 불가능하다.
- 애플리케이션의 결함은 대부분 개발자의 특성이나 애플리케이션의 기능적 특징 때문에 특정 모듈에 집중*되어 있다. 애플리케이션의 20%에 해당하는 코드에서 전체 80%의 결함이 발견된다고 하여 파레토 법칙*을 적용하기도 한다.
- 애플리케이션 테스트에서는 동일한 테스트 케이스*로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 ‘살충제 패러독스(Pesticide Paradox)*’ 현상이 발생한다. 살충제 패러독스를 방지하기 위해서 테스트 케이스를 지속적으로 보완 및 개선해야 한다.
- 애플리케이션 테스트는 소프트웨어 특징, 테스트 환경, 테스터 역량 등 정황(Context)에 따라 테스트 결과가 달라질 수 있으므로, 정황에 따라 테스트를 다르게 수행해야 한다.
- 소프트웨어의 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 말할 수 없다. 이것을 오류-부재의 궤변(Absence of Errors Fallacy)이라고 한다.

특정 모듈 집중

대부분의 결함이 소수의 특정 모듈에 집중해서 발생하는 것을 결함 집중(Defect Clustering)이라고 합니다.

파레토 법칙(Pareto Principle)

파레토의 법칙은 상위 20% 사람들이 전체 부의 80%를 가지고 있다거나, 상위 20% 고객이 매출의 80%를 창출한다는 의미로, 이 법칙이 애플리케이션 테스트에도 적용된다는 것입니다. 즉 ‘테스트로 발견된 80%의 오류는 20%의 모듈에서 발견되므로 20%의 모듈을 집중적으로 테스트하여 효율적으로 오류를 찾자는 것입니다.

테스트 케이스(Test Case)

테스트 케이스는 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서입니다.

살충제 패러독스(Pesticide Paradox)

살충제 패러독스는 살충제를 지속적으로 뿌리면 벌레가 내성이 생겨서 죽지 않는 현상을 의미합니다.

- 테스트와 위험은 반비례한다. 테스트를 많이 하면 할수록 미래에 발생할 위험을 줄일 수 있다.
- 테스트는 작은 부분에서 시작하여 점점 확대하며 진행해야 한다.
- 테스트는 개발자와 관계없는 별도의 팀에서 수행해야 한다.



기출문제 따라잡기

Section 049

출제예상

1. 다음 중 프로그램의 검증(Verification)에 대한 설명으로 옳은 것은?

- ① 개발한 프로그램이 명세서에 맞게 만들어졌는지 점검한다.
- ② 개발한 프로그램이 고객의 요구사항에 맞게 구현되었는지를 점검한다.
- ③ 개발한 프로그램이 생산성이 높게 만들어졌는지 점검한다.
- ④ 사용자의 입장에서 하는 점검이다.

개발자의 입장에서 명세서에 맞게 만들어졌는지를 점검하는 것은 Verification, 사용자의 입장에서 고객의 요구사항에 맞게 구현되었는지를 점검하는 것은 Validation입니다.

출제예상

2. 다음 중 애플리케이션 테스트에 대한 설명으로 틀린 것은?

- ① 애플리케이션 테스트는 프로그램 실행 전에 코드 리뷰, 인스펙션 등을 통해 사전에 오류를 발견하여 예방할 수 있다.
- ② 애플리케이션 테스트를 반복적으로 실행하여 제품의 신뢰도를 향상시킬 수 있다.
- ③ 테스트는 프로그램 개발이 완료된 후 체계적으로 계획하여 실행해야 한다.
- ④ 성공적인 테스트는 아직 발견되지 않은 오류를 찾아내는 것이다.

테스팅은 프로그램 개발 초기부터 계획하고 시작해야 오류 발견뿐만 아니라 새로운 오류의 유입도 예방할 수 있습니다.

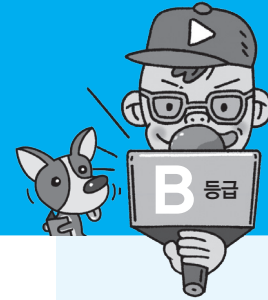
출제예상

3. 소프트웨어에 잠재되어 있는 결함을 찾기 위해 테스트를 진행하려고 한다. 다음 설명 중 틀린 것은?

- ① 테스트는 소프트웨어의 잠재적인 결함을 줄일 수는 있지만 완벽하게 결함이 없다고 증명할 수는 없다.
- ② 결함은 개발자나 기능에 따라서 특정한 모듈에 집중되어 있다.
- ③ 사용자의 요구사항을 만족시키지 못한다면, 오류를 모두 제거해도 해당 소프트웨어의 품질이 높다고 말할 수 없다.
- ④ 테스트는 큰 부분에서 시작하여 점점 축소하여 세부적으로 테스트를 진행해야 한다.

테스트는 단위 테스트에서 시작하여 통합 테스트로 진행해야 합니다. 즉, 작은 부분에서 시작하여 점점 확대하며 진행해야 합니다.

▶ 정답 : 1. ① 2. ③ 3. ④



1 프로그램 실행 여부에 따른 테스트

애플리케이션을 테스트 할 때 프로그램의 실행 여부에 따라 정적 테스트와 동적 테스트로 나뉜다.

정적 테스트	<ul style="list-style-type: none"> • 프로그램을 실행하지 않고 명세서나 소스 코드를 대상으로 분석하는 테스트이다. • 소프트웨어 개발 초기에 결함을 발견할 수 있어 소프트웨어의 개발 비용을 낮추는데 도움이 된다. • 종류 : 워크스루*, 인스펙션*, 코드 검사 등
동적 테스트	<ul style="list-style-type: none"> • 프로그램을 실행하여 오류를 찾는 테스트로, 소프트웨어 개발의 모든 단계에서 테스트를 수행할 수 있다. • 종류 : 블랙박스 테스트, 화이트박스 테스트

2 테스트 기반(Test Bases)에 따른 테스트

애플리케이션을 테스트 할 때 무엇을 기반으로 수행하느냐에 따라 명세 기반, 구조 기반, 경험 기반 테스트로 나뉜다.

명세 기반 테스트	<ul style="list-style-type: none"> • 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스*로 만들어 구현하고 있는지 확인하는 테스트이다. • 종류 : 동등 분할, 경계 값 분석 등
구조 기반 테스트	<ul style="list-style-type: none"> • 소프트웨어 내부의 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트이다. • 종류 : 구문 기반, 결정 기반, 조건 기반, 결정 기반 등
경험 기반 테스트	<ul style="list-style-type: none"> • 유사 소프트웨어나 기술 등에 대한 테스트의 경험을 기반으로 수행하는 테스트이다. • 경험 기반 테스트는 사용자의 요구사항에 대한 명세가 불충분하거나 테스트 시간에 제약이 있는 경우 수행하면 효과적이다. • 종류 : 에러 추정, 체크 리스트, 탐색적 테스트

3 시각에 따른 테스트

애플리케이션을 테스트 할 때 누구를 기준으로 하느냐에 따라 검증(Verification) 테스트와 확인(Validation) 테스트로 나뉜다.

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

전문가의 조언

애플리케이션 테스트는 테스트 시 프로그램의 실행 여부 또는 진행 목적 등에 따라 다양하게 분류됩니다. 각각에 해당하는 테스트 종류를 서로 구분할 수 있을 정도로 정리해 두세요.

워크스루(Walkthrough, 검토 회의)

- 워크스루는 소프트웨어 개발자의 작업 내역을 개발자가 모집한 전문가들이 검토하는 것을 말합니다.
- 소프트웨어 검토를 위해 미리 준비된 자료를 바탕으로 정해진 절차에 따라 평가합니다.
- 오류의 조기 검출을 목적으로 하며 발견된 오류는 문서화합니다.

인스펙션(inspection)

인스펙션은 워크스루를 발전시킨 형태로, 소프트웨어 개발 단계에서 산출된 결과물의 품질을 평가하며 이를 개선하기 위한 방법 등을 제시합니다.

테스트 케이스(Test Case)

테스트 케이스는 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서입니다.

4 목적에 따른 테스트

애플리케이션을 테스트 할 때 무엇을 목적으로 테스트를 진행하느냐에 따라 회복(Recovery), 안전(Security), 강도(Stress), 성능(Performance), 구조(Structure), 회귀(Regression), 병행(Parallel) 테스트로 나뉜다.

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



기출문제 따라잡기

Section 050

출제예상

1. 다음 중 정적 테스트와 동적 테스트에 대한 설명으로 틀린 것은?

- ① 정적 테스트는 개발한 프로그램을 실행하지 않고 테스트한다.
- ② 동적 테스트는 개발한 프로그램을 직접 실행하면서 오류를 찾는 테스트이다.
- ③ 동적 테스트에는 워크스루, 인스펙션, 코드 검사 등이 있다.
- ④ 정적 테스트는 개발 초기에 결함을 발견함으로써 개발 비용을 낮추는데 도움이 된다.

워크스루, 인스펙션, 코드 검사는 정적 테스트입니다.

출제예상

2. 다음 중 확인(Validation) 테스트에 대한 설명으로 옳은 것은?

- ① 개발자의 시각에서 테스트를 진행한다.
- ② 제품이 올바르게 생산되고 있는가를 확인한다.
- ③ 소프트웨어가 명세서대로 만들어졌는지를 중점을 두고 테스트한다.
- ④ 소프트웨어가 사용자의 요구사항을 충족시키는데 중점을 두고 테스트한다.

①, ②, ③번은 검증(Verification) 테스트에 대한 설명입니다.

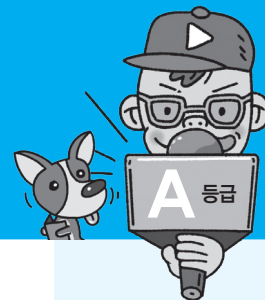
출제예상

3. 다음 중 애플리케이션 테스트에 대한 설명으로 틀린 것은?

- ① 회귀 테스트는 소프트웨어에 여러 가지 결함을 주어 실패하도록 한 후 올바르게 복구되는지를 확인하는 테스트이다.
- ② 강도 테스트는 소프트웨어에 과도한 정보량이나 빈도 등을 부과하여 과부하 시에도 소프트웨어가 정상적으로 실행되는지를 확인하는 테스트이다.
- ③ 안전 테스트는 시스템 내에 설치된 보호 도구가 불법적인 침입으로부터 시스템을 보호할 수 있는지를 확인하는 테스트이다.
- ④ 구조 테스트는 소프트웨어 내부의 논리적인 경로, 소스 코드의 복잡도 등을 평가하는 테스트이다.

회귀 테스트는 소프트웨어 변경 또는 수정된 코드에 새로운 결함이 없음을 테스트합니다. ①번은 회복 테스트에 대한 설명입니다.

▶ 정답: 1. ③ 2. ④ 3. ①



1 화이트박스 테스트(White Box Test)

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

- 화이트박스 테스트는 설계된 절차에 초점을 둔 구조적 테스트로 프로시저 설계의 제어 구조를 사용하여 테스트 케이스를 설계하며, 테스트 과정의 초기에 적용된다.
- 모듈 안의 작동을 직접 관찰한다.
- 원시 코드(모듈)의 모든 문장을 한 번 이상 실행함으로써 수행된다.
- 프로그램의 제어 구조에 따라 선택, 반복 등의 분기점 부분들을 수행함으로써 논리적 경로를 제어한다.

2 화이트박스 테스트의 종류

화이트 박스 테스트의 종류에는 기초 경로 검사, 제어 구조 검사 등이 있다.

기초 경로 검사	<ul style="list-style-type: none"> • 대표적인 화이트박스 테스트 기법이다. • 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법으로, 테스트 측정 결과는 실행 경로의 기초를 정의하는 데 지침으로 사용된다.
제어 구조 검사	<ul style="list-style-type: none"> • 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

3 화이트박스 테스트의 검증 기준

화이트박스 테스트의 검증 기준은 테스트 케이스들이 테스트에 얼마나 적절한지를 판단하는 기준으로, 문장 검증 기준, 분기 검증 기준, 조건 검증 기준, 분기/조건 기준이 있다.

문장 검증 기준 (Statement Coverage)	소스 코드의 모든 구문이 한 번 이상 수행되도록 테스트 케이스 설계
분기 검증 기준 (Branch Coverage)	소스 코드의 모든 조건문이 한 번 이상 수행되도록 테스트 케이스 설계



전문가의 조언

- 애플리케이션 테스트는 소프트웨어 내부 구조의 참조 여부에 따라 블랙박스 테스트와 화이트박스 테스트로 나뉩니다. 블랙박스 테스트와 화이트박스 테스트는 중요합니다. 두 테스트의 개념, 차이점, 종류 등을 모두 숙지해 두세요.
- 화이트박스 테스트의 의미는 '논리'라는 단어를 중심으로 알아두세요. 화이트박스 테스트는 투명한 박스라는 의미로 모듈 안의 내용을 볼 수 있어서 내부의 논리적인 경로를 테스트한다고 생각하면 됩니다.

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

잠깐만요



검증 기준(Coverage)의 종류

검증 기준의 종류에는 크게 기능 기반 커버리지, 라인 커버리지, 코드 커버리지가 있으며, 화이트박스 테스트에서 사용되는 문장 검증 기준, 분기 검증 기준 등은 모두 코드 커버리지에 해당합니다.

- 기능 기반 커버리지: 실제 테스트가 수행된 기능의 수 / 전체 기능의 수
- 라인 커버리지(Line Coverage): 테스트 시나리오가 수행한 소스 코드의 라인 수 / 전체 소스 코드의 라인 수
- 코드 커버리지(Code Coverage): 소스 코드의 구문, 분기, 조건 등의 구조 코드 자체가 얼마나 테스트되었는지를 측정하는 방법



전문가의 조언

- 블랙박스는 박스 안을 들여다 볼 수 없는 검은 상자입니다. 즉 블랙박스 안에서 어떤 일이 일어나는지 알 수는 없지만 입력된 데이터가 블랙박스를 통과하여 출력될 때 그 결과물이 정확한지를 검사하는 것입니다. 이런 블랙박스 테스트의 개념을 염두에 두고 개별적인 검사 기법을 잘 이해해 두세요. 블랙박스 테스트의 종류도 기억해야 합니다.
- Section 0500에서 학습한 명세 기반 테스트, 경험 기반 테스트는 블랙박스 테스트, 구조 기반 테스트는 화이트박스 테스트에 해당합니다.

4 블랙박스 테스트(Black Box Test)

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

- 사용자의 요구사항 명세를 보면서 테스트하는 것으로, 주로 구현된 기능을 테스트한다.
- 소프트웨어 인터페이스에서 실시되는 테스트이다.
- 부정확하거나 누락된 기능, 인터페이스 오류, 자료 구조나 외부 데이터베이스 접근에 따른 오류, 행위나 성능 오류, 초기화와 종료 오류 등을 발견하기 위해 사용되며, 테스트 과정의 후반부에 적용된다.
- 블랙박스 테스트의 종류에는 동치 분할 검사, 경계값 분석, 원인-효과 그래프 검사, 오류 예측 검사, 비교 검사 등이 있다.

5 블랙박스 테스트의 종류

동치 분할 검사 (Equivalence Partitioning Testing)	<ul style="list-style-type: none"> • 입력 자료에 초점을 맞춰 테스트 케이스를 만들고 검사하는 방법으로 동등 분할 기법이라고도 한다. • 프로그램의 입력 조건에 타당한 입력 자료와 타당하지 않은 입력 자료의 개수를 균등하게 하여 테스트 케이스를 정하고, 해당 입력 자료에 맞는 결과가 출력되는지 확인하는 기법이다.
경계값 분석 (Boundary Value Analysis)	<ul style="list-style-type: none"> • 입력 자료에만 치중한 동치 분할 기법을 보완하기 위한 기법이다. • 입력 조건의 중간값보다 경계값에서 오류가 발생될 확률이 높다는 점을 이용하여 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하는 기법이다.

원인-효과 그래프 검사 (Cause-Effect Graphing Testing)	입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법이다.
오류 예측 검사 (Error Guessing)	<ul style="list-style-type: none"> • 과거의 경험이나 확인자의 감각으로 테스트하는 기법이다. • 다른 블랙 박스 테스트 기법으로는 찾아낼 수 없는 오류를 찾아내는 일련의 보충적 검사 기법이며, 데이터 확인 검사라고도 한다.
비교 검사 (Comparison Testing)	여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법이다.



기출문제 따라잡기

Section 051

이전기출

1. 다음 설명에 해당하는 테스트 기법은?

- 모듈 안의 작동을 자세히 관찰할 수 있다.
- 프로그램 원시 코드의 논리적인 구조를 커버(Cover)하도록 테스트 케이스를 설계하는 프로그램 테스트 방법이다.

- ① 블랙박스 테스트
- ② 화이트박스 테스트
- ③ 알파 테스트
- ④ 베타 테스트

박스가 투명하여 모듈 안의 내용을 볼 수 있는 것이 무엇인가요? '모듈 내부의 논리적인~' 이런 말이 나오면 투명한 화이트박스 테스트 기법이란 것! 기억해 두세요.

이전기출

2. White Box Testing의 설명으로 옳지 않은 것은?

- ① 기초 경로 테스트, 경계값 분석이 대표적인 기법이다.
- ② 소스 코드의 모든 문장을 한 번 이상 수행함으로써 진행된다.
- ③ 모듈 안의 작동을 직접 관찰할 수 있다.
- ④ 산출물의 각 기능별로 적절한 프로그램의 제어 구조에 따라 선택, 반복 등의 부분들을 수행함으로써 논리적 경로를 점검한다.

경계값 분석(Boundary Value Analysis)은 블랙박스 테스트 기법입니다.

이전기출

3. 소프트웨어 인터페이스에서 실시되는 테스트로, 소프트웨어의 기능이 의도대로 작동하고 있는지 테스트 하는 기법은?

- ① 화이트박스 테스트
- ② 블랙박스 테스트
- ③ 레드박스 테스트
- ④ 블루박스 테스트

논리적인 경로나 데이터 흐름과 관련된 기법은 화이트박스 테스트 기법이고, 기능과 관련된 기법은 블랙박스 테스트 기법입니다.

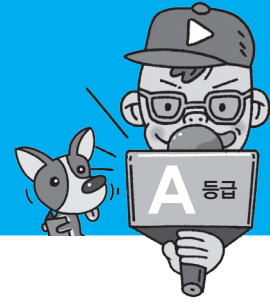
출제예상

4. 다음 중 블랙박스 테스트 기법에 해당하지 않는 것은?

- ① 동치 분할 테스트
- ② 경계값 분석
- ③ 원인 효과 그래픽 기법
- ④ 데이터 흐름 테스트

논리적인 경로나 데이터 흐름과 관련된 기법은 화이트박스 테스트 기법입니다.

▶ 정답: 1. ② 2. ① 3. ② 3. ④



전문가의 조언

애플리케이션 테스트는 소프트웨어의 개발 과정과 함께 지속적으로 진행됩니다. 모듈을 개발하면 모듈에 대한 단위 테스트를 실행하고, 여러 개의 모듈을 결합하여 시스템으로 완성시키는 과정에서는 통합 테스트를, 그리고 설계된 소프트웨어가 시스템에서 정상적으로 수행되는지를 확인하기 위해서 시스템 테스트를 수행합니다. 최종적으로 소프트웨어가 완성되면 사용자에게 인도하기 전에 인수 테스트를 수행합니다. 이러한 과정을 염두에 두고 테스트 진행 순서와 각 테스트의 특징을 정리하세요.

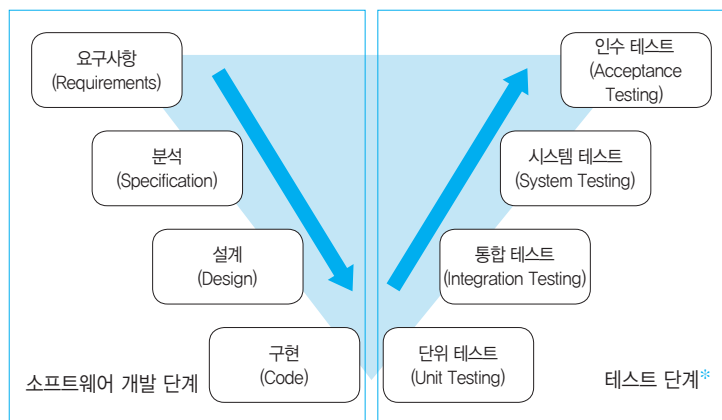
개발 단계에 따른 테스트들을 검증과 확인 테스트로 구분하면 다음과 같습니다.

- **검증(Verification) 테스트** : 개발자 기준의 테스트로, 단위 테스트, 통합 테스트, 시스템 테스트가 해당됨
- **확인(Validation) 테스트** : 사용자 기준의 테스트로, 인수 테스트가 해당됨

1 개발 단계에 따른 애플리케이션 테스트

애플리케이션 테스트는 소프트웨어의 개발 단계에 따라 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트로 분류된다. 이렇게 분류된 것을 테스트 레벨이라고 한다.

- 애플리케이션 테스트는 소프트웨어의 개발 단계에서부터 테스트를 수행하므로 단순히 소프트웨어에 포함된 코드 상의 오류뿐만 아니라 요구 분석의 오류, 설계 인터페이스 오류 등도 발견할 수 있다.
- 애플리케이션 테스트와 소프트웨어 개발 단계를 연결하여 표현한 것을 V-모델이라 한다.



소프트웨어 생명 주기의 V-모델

2 단위 테스트(Unit Test)

단위 테스트는 코딩 직후 소프트웨어 설계의 최소 단위인 모듈이나 컴포넌트에 초점을 맞춰 테스트하는 것이다.

- 단위 테스트에서는 인터페이스, 외부적 I/O, 자료 구조, 독립적 기초 경로, 오류 처리 경로, 경계 조건 등을 검사한다.
- 단위 테스트는 사용자의 요구사항을 기반으로 한 기능성 테스트를 최우선으로 수행한다.
- 단위 테스트는 구조 기반 테스트와 명세 기반 테스트로 나뉘지만 주로 구조 기반 테스트를 시행한다.

테스트 방법	테스트 내용	테스트 목적
구조 기반 테스트	프로그램 내부 구조 및 복잡도를 검증하는 화이트박스(White Box) 테스트 시행	제어 흐름, 조건 결정
명세 기반 테스트	목적 및 실행 코드 기반의 블랙박스(Black Box) 테스트 시행	동등 분할, 경계 값 분석

3 통합 테스트(Integration Test)

통합 테스트는 단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트를 의미한다.

- 통합 테스트는 모듈 간 또는 통합된 컴포넌트 간의 상호 작용 오류를 검사한다.

4 시스템 테스트(System Test)

시스템 테스트는 개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행되는가를 점검하는 테스트이다.

- 환경적인 장애 리스크*를 최소화하기 위해서는 실제 사용 환경과 유사하게 만든 테스트 환경에서 테스트를 수행해야 한다.
- 시스템 테스트는 기능적 요구사항과 비기능적 요구사항으로 구분하여 각각을 만족하는지 테스트한다.

테스트 방법	테스트 내용
기능적 요구사항	요구사항 명세서, 비즈니스 절차, 유스케이스 등 명세서 기반의 블랙박스(Black Box) 테스트 시행
비기능적 요구사항	성능 테스트, 회복 테스트, 보안 테스트, 내부 시스템의 메뉴 구조, 웹 페이지의 네비게이션 등 구조적 요소에 대한 화이트박스(White Box) 테스트 시행

5 인수 테스트(Acceptance Test)

인수 테스트는 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트하는 방법이다.

- 인수 테스트는 개발한 소프트웨어를 사용자가 직접 테스트한다.
- 인수 테스트에 문제가 없으면 사용자는 소프트웨어를 인수하게 되고, 프로젝트는 종료된다.
- 인수 테스트는 다음과 같이 6가지의 종류로 구분해서 테스트한다.

테스트 종류	설명
사용자 인수 테스트	사용자가 시스템 사용의 적절성 여부를 확인한다.
운영상의 인수 테스트	시스템 관리자가 시스템 인수 시 수행하는 테스트 기법으로, 백업/복원 시스템, 재난 복구, 사용자 관리, 정기 점검 등을 확인한다.



전문가의 조언

통합 테스트는 다음 섹션에서 자세히 공부하니 여기에서는 통합 테스트가 무엇인지 정도만 알아두세요.

환경적인 장애 리스크

환경적인 장애 리스크는 OS, DBMS, 시스템 운영 장비 등 테스트 시 사용할 물리적, 논리적 테스트 환경과 실제 소프트웨어를 사용할 환경이 달라서 발생할 수 있는 바람직하지 못한 결과를 의미합니다.

계약 인수 테스트	계약상의 인수/검수 조건을 준수하는지 여부를 확인한다.
규정 인수 테스트	소프트웨어가 정부 지침, 법규, 규정 등 규정에 맞게 개발되었는지 확인한다.
알파 테스트	<ul style="list-style-type: none"> 개발자의 장소에서 사용자가 개발자 앞에서 행하는 테스트 기법이다. 테스트는 통제된 환경에서 행해지며, 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인하면서 기록한다.
베타 테스트	<ul style="list-style-type: none"> 선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법이다. 실업무를 가지고 사용자가 직접 테스트하는 것으로, 개발자에 의해 제어되지 않은 상태에서 테스트가 행해지며, 발견된 오류와 사용상의 문제점을 기록하고 개발자에게 주기적으로 보고한다.



기출문제 따라잡기

Section 052

출제예상

1. 소프트웨어 개발 단계에 따른 테스트 순서로 적절하게 이루어진 것은?

- ① 단위 테스트 → 통합 테스트 → 시스템 테스트 → 인수 테스트
- ② 인수 테스트 → 단위 테스트 → 통합 테스트 → 시스템 테스트
- ③ 단위 테스트 → 통합 테스트 → 인수 테스트 → 시스템 테스트
- ④ 인수 테스트 → 시스템 테스트 → 단위 테스트 → 통합 테스트

개발 관련자가 작은 단위에서 큰 단위로 테스트한 후 시스템 전체를 테스트하여 오류가 없음을 확인하고 나면 사용자가 인수하기 전에 마지막으로 인수 테스트를 수행합니다.

출제예상

2. 소프트웨어 설계의 최소 단위인 모듈에 초점을 맞춰 테스트 하는 단계는?

- ① 단위 테스트
- ② 통합 테스트
- ③ 인수 테스트
- ④ 시스템 테스트

테스트 순서상 맨 처음에 수행하는 테스트가 바로 최소 단위로 테스트하는 단계입니다. 답을 찾아보세요.

출제예상

3. 알파 테스트와 베타 테스트에 대한 설명으로 가장 옳지 않은 것은?

- ① 개발자 관점에서 오류를 찾기 위한 테스트 기법들이다.
- ② 알파 테스트는 선택된 사용자가 개발자 앞에서 검사한다.
- ③ 베타 테스트는 선정된 최종 사용자가 여러 명의 사용자 앞에서 검사한다.
- ④ 알파 테스트는 통제된 환경에서, 베타 테스트는 개발자에 의해 제어되지 않은 상태에서 검사한다.

알파 테스트와 베타 테스트는 사용자가 개발된 소프트웨어를 인수하기 위해 사용자 관점에서 테스트하는 기법입니다.

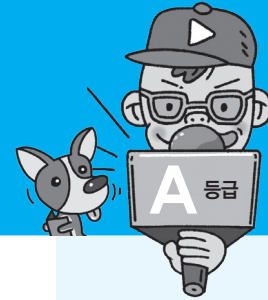
출제예상

4. 다음 테스트 중에서 알파 테스트, 베타 테스트와 가장 밀접한 관계가 있는 테스트는?

- ① Unit Test
- ② Integration Test
- ③ System Test
- ④ Acceptance Test

정답을 모르겠다면 3번 문제 해설을 다시 한 번 읽어보세요.

▶ 정답 : 1. ① 2. ① 3. ① 3. ④



1 통합 테스트(Integration Test)

통합 테스트는 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생하는 오류 및 결함을 찾는 테스트 기법이다.

- 통합 테스트 방법에는 비점진적 통합 방식과 점진적 통합 방식이 있다.

비점진적 통합 방식	<ul style="list-style-type: none"> 단계적으로 통합하는 절차 없이 모든 모듈이 미리 결합되어 있는 프로그램 전체를 테스트하는 방법으로, 빅뱅 통합 테스트* 방식이 있다. 규모가 작은 소프트웨어에 유리하며 단시간 내에 테스트가 가능하다. 전체 프로그램을 대상으로 하기 때문에 오류 발견 및 장애 위치 파악 및 수정이 어렵다.
점진적 통합 방식	<ul style="list-style-type: none"> 모듈 단위로 단계적으로 통합하면서 테스트하는 방법으로, 하향식, 상향식, 혼합식 통합 방식이 있다. 오류 수정이 용이하고, 인터페이스와 연관된 오류를 완전히 테스트할 가능성이 높다.

2 하향식 통합 테스트(Top Down Integration Test)

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

- 주요 제어 모듈을 기준으로 하여 아래 단계로 이동하면서 통합하는데, 이때 깊이 우선 통합법*이나 넓이 우선 통합법*을 사용한다.
- 테스트 초기부터 사용자에게 시스템 구조를 보여줄 수 있다.
- 상위 모듈에서는 테스트 케이스를 사용하기 어렵다.
- 하향식 통합 방법은 다음과 같은 절차로 수행된다.
 - 주요 제어 모듈은 작성된 프로그램을 사용하고, 주요 제어 모듈의 종속 모듈들은 스텝(Stub)*으로 대체한다.
 - 깊이 우선 또는 넓이 우선 등의 통합 방식에 따라 하위 모듈인 스텝들이 한 번에 하나씩 실제 모듈로 교체된다.
 - 모듈이 통합될 때마다 테스트를 실시한다.
 - 새로운 오류가 발생하지 않음을 보증하기 위해 회귀 테스트*를 실시한다.

3 상향식 통합 테스트(Bottom Up Integration Test)

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

- 가장 하위 단계의 모듈부터 통합 및 테스트가 수행되므로 스텝(Stub)은 필요하지 않지만, 하나의 주요 제어 모듈과 관련된 종속 모듈의 그룹인 클러스터(Cluster)가 필요하다.

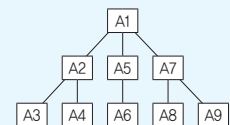
전문가의 조언

통합 테스트에서는 상황식 테스트와 하향식 테스트가 중요하다. 어떤 테스트를 말하는지 구분할 수 있도록 각각의 특징을 잘 알아두세요.

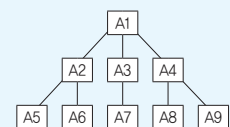
빅뱅 통합 테스트

모듈 간의 상호 인터페이스를 고려하지 않고 단위 테스트가 끝난 모듈을 한꺼번에 결합시켜 테스트하는 방법입니다. 주로 소규모 프로그램이나 프로그램의 일부만을 대상으로 테스트 할 때 사용됩니다.

- **깊이 우선 통합법** : 주요 제어 모듈을 중심으로 해당 모듈에 종속된 모든 모듈을 통합하는 것으로, 다음 그림에 대한 통합 순서는 A1, A2, A3, A4, A5, A6, A7, A8, A9 순입니다.



- **넓이 우선 통합법** : 구조의 수평을 중심으로 해당하는 모듈을 통합하는 것으로, 다음 그림에 대한 통합 순서는 A1, A2, A3, A4, A5, A6, A7, A8, A9 순입니다.



테스트 스텝(Test Stub)

제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로, 일시적으로 필요한 조건만을 가지고 있는 시험용 모듈입니다.

회귀 테스트

이미 테스트된 프로그램의 테스트를 반복하는 것으로, 통합 테스트로 인해 변경된 모듈이나 컴포넌트에 새로운 오류가 있는지 확인하는 테스트입니다.

테스트 드라이버(Test Driver)

테스트 드라이버는 테스트 대상의 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 도구입니다.

- 상향식 통합 방법은 다음과 같은 절차로 수행된다.
 - 하위 모듈들을 클러스터(Cluster)로 결합한다.
 - 상위 모듈에서 데이터의 입·출력을 확인하기 위해 더미 모듈인 드라이버(Driver)*를 작성한다.
 - 통합된 클러스터 단위로 테스트한다.
 - 테스트가 완료되면 클러스터는 프로그램 구조의 상위로 이동하여 결합하고 드라이버는 실제 모듈로 대체된다.



테스트 드라이버와 테스트 스텝의 차이점

구분	드라이버	스텝
필요 시기	상위 모듈 없이 하위 모듈이 있는 경우 하위 모듈 구동	상위 모듈은 있지만 하위 모듈이 없는 경우 하위 모듈 대체
테스트 방식	상향식(Bottom Up) 테스트	하향식(Top-Down) 테스트
개념도		
공통점	소프트웨어 개발과 테스트를 병행할 경우 이용	
차이점	<ul style="list-style-type: none"> 이미 존재하는 하위 모듈과 존재하지 않는 상위 모듈 간의 인터페이스 역할을 합니다. 소프트웨어 개발이 완료되면 드라이버는 본래의 모듈로 교체됩니다. 	<ul style="list-style-type: none"> 일시적으로 필요한 조건만을 가지고 임시로 제공되는 가짜 모듈의 역할을 합니다. 시험용 모듈이기 때문에 일반적으로 드라이버보다 작성하기 쉽습니다.

4 혼합식 통합 테스트

혼합식 통합 테스트는 하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하여 최적의 테스트를 지원하는 방식으로, 샌드위치(Sandwich)식 통합 테스트 방법이라고도 한다.

5 회귀 테스트(Regression Testing)

회귀 테스트는 이미 테스트된 프로그램의 테스트를 반복하는 것으로, 통합 테스트로 인해 변경된 모듈이나 컴포넌트에 새로운 오류가 있는지 확인하는 테스트이다.

- 회귀 테스트는 수정한 모듈이나 컴포넌트가 다른 부분에 영향을 미치는지, 오류가 생기지 않았는지 테스트하여 새로운 오류가 발생하지 않음을 보증하기 위해 반복 테스트한다.

- 회귀 테스트는 모든 테스트 케이스를 이용해 테스트하는 것이 가장 좋지만 시간과 비용이 많이 필요하므로 기존 테스트 케이스 중 변경된 부분을 테스트할 수 있는 테스트 케이스만을 선정하여 수행한다.
- 회귀 테스트의 테스트 케이스 선정 방법
 - 모든 애플리케이션의 기능을 수행할 수 있는 대표적인 테스트 케이스를 선정한다.
 - 애플리케이션 기능 변경에 의한 파급 효과를 분석하여 파급 효과가 높은 부분이 포함된 테스트 케이스를 선정한다.
 - 실제 수정이 발생한 모듈 또는 컴포넌트에서 시행하는 테스트 케이스를 선정한다.



기출문제 따라잡기

Section 053

이전기출

1. 다음 중 상향식 통합 테스트에 대한 설명으로 옳지 않은 것은?

- ① 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트한다.
- ② 테스트를 위해 드라이버를 생성한다.
- ③ 하위 모듈들을 클러스터로 결합한다.
- ④ 깊이 우선 통합법 또는 넓이 우선 통합법에 따라 스텝을 실제 모듈로 대치한다.

드라이버는 상향식 테스트, 스텝은 하향식 통합 테스트에서 사용됩니다.

이전기출

2. 하향식 통합 검사(Test)에 대한 설명으로 가장 옳지 않은 것은?

- ① 시스템 구조의 위층에 있는 모듈부터 아래층의 모듈로 내려오면서 통합한다.
- ② 일반적으로 스텝(Stub)을 드라이버(Driver)보다 쉽게 작성할 수 있다.
- ③ 테스트 초기에는 시스템의 구조를 사용자에게 보여줄 수 없다.
- ④ 상위층에서 테스트 케이스를 쓰기가 어렵다.

높은 곳에 있으면 아래가 한 눈에 보이듯, 상위 모듈에서 테스트를 시작하는 하향식 통합 테스트에서는 테스트 초기부터 사용자에게 시스템 구조를 보여줄 수 있습니다.

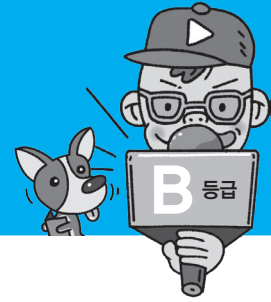
출제예상

3. 통합 테스트 방법은 비점진적 통합 방식과 점진적 통합 방식으로 나뉜다. 다음 중 점진적 통합 방식이 아닌 것은?

- ① 하향식 통합
- ② 상향식 통합
- ③ 빅뱅 통합
- ④ 샌드위치 통합

우주의 대폭발을 의미하는 빅뱅, 단계적으로 폭발 했을까요? 빅뱅 통합도 마찬가지입니다. 단계적인 절차없이 전체 모듈을 한꺼번에 결합시켜 테스트하는 방법으로 비점진적 통합 방식입니다.

▶ 정답 : 1. ④ 2. ③ 3. ③

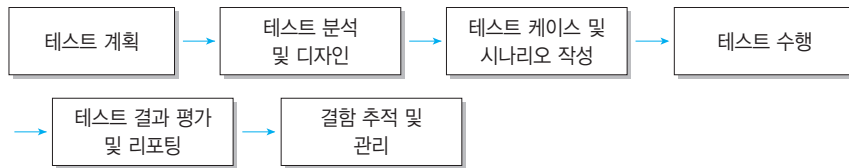


전문가의 조언

애플리케이션 테스트는 개발된 소프트웨어가 제대로 작동하는지 확인하는 것이고, 애플리케이션 테스트 프로세스는 테스트를 효과적으로 수행하기 위해 일정한 절차를 따르는 것을 말합니다. 애플리케이션 테스트 절차를 기억하고 각 단계에서 수행하는 기능에 대해 알아두세요.

1 애플리케이션 테스트 프로세스

애플리케이션 테스트 프로세스는 개발된 소프트웨어가 사용자의 요구대로 만들어졌는지, 결함은 없는지 등을 테스트하는 절차로, 다음과 같은 순서로 진행된다.



- 애플리케이션 테스트를 마치면 테스트 계획서, 테스트 케이스, 테스트 시나리오, 테스트 결과서가 산출된다.
 - 테스트 계획서 : 테스트 목적, 범위, 일정, 수행 절차, 대상 시스템 구조, 조직의 역할 및 책임 등 테스트 수행을 계획한 문서
 - 테스트 케이스 : 사용자의 요구사항을 얼마나 준수하는지 확인하기 위한 입력 값, 실행 조건, 기대 결과 등으로 만들어진 테스트 항목의 명세서
 - 테스트 시나리오 : 테스트를 수행할 여러 개의 테스트 케이스의 동작 순서를 기술한 문서
 - 테스트 결과서 : 테스트 결과를 비교·분석한 내용을 정리한 문서

2 테스트 계획

테스트 계획 단계에서는 프로젝트 계획서, 요구 명세서 등을 기반으로 테스트 목표를 정의하고 테스트 대상 및 범위를 결정한다.

- 테스트 대상 시스템의 구조를 파악한다.
- 테스트에 투입되는 조직 및 비용을 산정한다.
- 테스트 시작 및 종료 조건을 정의한다.
 - 테스트 시작 조건 : 테스트 계획, 일정, 환경 구축, 사용자 요구사항에 대한 테스트 명세서, 투입 조직 및 참여 인력의 역할과 책임 등이 완료되면 테스트가 시작되도록 조건을 정의할 수 있으며, 모든 조건을 만족하지 않아도 테스트를 시작하도록 지정할 수 있다.
 - 테스트 종료 조건 : 정상적으로 테스트를 완료한 경우, 테스트 일정이 만료된 경우, 테스트 비용이 모두 소진된 경우 등 업무 기능의 중요도에 따라 테스트 종료 조건을 다르게 지정할 수 있다.
- 테스트 계획서를 작성한다.

3 테스트 분석 및 디자인

테스트 분석 및 디자인 단계에서는 테스트의 목적과 원칙을 검토하고 사용자의 요구 사항을 분석한다.

- 테스트에 대한 리스크 분석 및 우선순위를 결정한다.
- 테스트 데이터, 테스트 환경, 테스트 도구 등을 준비한다.

잠깐만요



테스트 데이터

테스트 데이터는 시스템의 기능이나 적합성 등을 테스트하기 위해 만든 데이터 집합으로, 소프트웨어의 기능을 차례대로 테스트할 수 있도록 만든 데이터입니다.

- 잘못된 데이터는 잘못된 결과를 도출하기 때문에 효율적인 테스트를 위해서는 올바른 테스트 데이터를 준비해야 합니다.
- 테스트 데이터 종류
 - 실제 데이터 : 선행된 연산에 의해 만들거나 실제 운영되는 데이터를 복제한 데이터
 - 가상 데이터 : 스크립트를 통해서 인위적으로 만든 데이터

4 테스트 케이스 및 시나리오 작성

테스트 케이스 및 시나리오 작성 단계에서는 테스트 케이스의 설계 기법에 따라 테스트 케이스를 작성하고 검토 및 확인한 후 테스트 시나리오를 작성한다.

- 테스트용 스크립트*를 작성한다.

5 테스트 수행

테스트 수행 단계에서는 테스트 환경을 구축한 후 테스트를 수행한다.

- 테스트의 실행 결과를 측정하여 기록한다.

잠깐만요



테스트 환경 구축

테스트 환경 구축은 개발된 소프트웨어가 실제 시스템에서 정상적으로 작동하는지 테스트하기 위해 실제 시스템과 동일하거나 유사한 사양의 하드웨어, 소프트웨어, 네트워크 등의 시설을 구축하는 것입니다.

- 하드웨어 : 서버, 클라이언트, 네트워크 등의 관련 장비 설치
- 소프트웨어 : 구축된 하드웨어 환경에 테스트할 소프트웨어 설치
- 가상 시스템 : 독립된 테스트 환경을 구축하기 힘든 경우, 가상 머신(Virtual Machine)* 기반의 서버 또는 클라우드 환경*을 구축하고 네트워크는 VLAN*과 같은 기법을 이용하여 논리적인 분할 환경을 구축합니다.



전문가의 조언

테스트 케이스는 테스트에 필요한 입력 값, 실행 조건, 기대 결과 등으로 만들어진 테스트 항목 명세서이고, 테스트 시나리오는 테스트 케이스의 동작 순서를 기술한 문서입니다. 테스트 케이스와 테스트 시나리오는 다음 섹션에서 자세히 공부하니 여기서는 개념 정도만 알아두세요.

테스트 스크립트(Test Script)

테스트 스크립트는 테스트 실행 절차나 수행 방법 등을 스크립트 언어로 작성한 파일입니다.

※ 스크립트 언어 : 소스 코드를 컴파일하지 않고도 내장된 번역기에 의해 번역되어 바로 실행할 수 있는 언어

가상 머신(Virtual Machine)

가상 머신은 하드웨어 환경을 소프트웨어로 구현한 것으로, 시스템에 설치된 운영체제와 다른 운영체제를 사용해야 하거나 독립된 작업 공간이 필요한 경우에 사용됩니다.

클라우드 환경

클라우드 환경은 서로 다른 물리적인 위치에 존재하는 컴퓨팅 자원을 가상화 기술로 통합하고 인터넷상의 서버를 통하여 네트워크, 데이터 저장, 콘텐츠 사용 등의 서비스를 한 번에 사용할 수 있는 환경을 의미합니다.

VLAN(Virtual Local Area Network)

VLAN은 LAN을 물리적인 배치와는 상관없이 논리적으로 분리하는 기술입니다.

6 테스트 결과 평가 및 리포팅

테스트 결과 평가 및 리포팅 단계에서는 테스트 결과를 비교 분석하여 테스트 결과서를 작성한다.

- 테스트 결과서는 결함 내용 및 결함 재현 순서 등 결함을 중점적으로 기록한다.
- 테스트가 종료되면 테스트 실행 절차의 리뷰 및 결과에 대한 평가를 수행하고, 그 결과에 따라 실행 절차를 최적화하여 다음 테스트에 적용한다.

7 결함 추적 및 관리

결함 추적 및 관리 단계에서는 테스트를 수행한 후 결함이 어디에서 발생했는지, 어떤 종류의 결함인지 등 결함을 추적하고 관리한다.

- 결함 추적 및 관리를 통해 동일한 결함 발견 시 처리 시간 단축 및 결함의 재발 등을 방지할 수 있다.
- 결함 관리 프로세스
 - ❶ 에러 발견 : 에러가 발견되면 테스트 전문가와 프로젝트팀이 논의한다.
 - ❷ 에러 등록 : 발견된 에러를 결함 관리 대장에 등록한다.

예 결함 관리 대장 작성 예

단계	대상 산출물명	결함 ID	결함 내용	결함 유형	심각도	우선 순위	결함 발생일	결함 조치 대상	조치 내용	조치자	확인자	처리 완료일	상태
테스트 검토	통합 테스트 결과 -xxx	DID_x_1	excel과 불일치	DB	Normal	Low	2019. 05.05	프로그램 수정	목록과 excel 처리로직, SQL 확인	박선호	전숙희	2019. 05.07	완료
테스트 검토	통합 테스트 결과 -xxx	DID_x_2	삭제 요청 승인시 오류 발생	기능	Minor	Medium	2019. 05.09	프로그램 수정	쿼리 조건절 수정	이정민	박희경	2019. 05.12	완료
테스트 검토	통합 테스트 결과 -xxx	DID_x_3	반복 처리 DB 트랜잭션 요청	GUI	Normal	Low	2019. 05.15	프로그램 수정	Excute Batch로 변경	박선호	전숙희	2019. 05.20	완료

- ❸ 에러 분석 : 등록된 에러가 실제 결함인지 아닌지를 분석한다.
- ❹ 결함 확정 : 등록된 에러가 실제 결함이면 결함 확정 상태로 설정한다.
- ❺ 결함 할당 : 결함을 해결할 담당자에게 결함을 할당하고 결함 할당 상태로 설정한다.
- ❻ 결함 조치 : 결함을 수정하고, 수정이 완료되면 결함 조치 상태로 설정한다.
- ❼ 결함 조치 검토 및 승인 : 수정이 완료된 결함에 대해 확인 테스트를 수행하고, 이상이 없으면 결함 조치 완료 상태로 설정한다.

잠깐만요



결함 관리 용어

- **에러(Error)/오류** : 결함(Defect)의 원인이 되는 것으로, 일반적으로 소프트웨어 개발자, 분석가 등 사람에게 의해 발생한 실수를 의미합니다.
- **결함/결점/버그(Bug)** : 에러/오류로 인해 소프트웨어 제품에 발생한 결함을 의미하며, 결함을 제거하지 않으면 소프트웨어 제품에 문제(Problem)가 발생할 수 있습니다.

따라잡기



기출문제 따라잡기

Section 054

출제예상

1. 테스트를 진행한다고 할 때 언제 오류를 발견하는 것이 가장 좋은가?

- ① 요구사항 분석 단계
- ② UI 설계 단계
- ③ UI 구현 단계
- ④ 결함 추적 및 관리 단계

오류는 될 수 있으면 빨리 발견하는 게 좋습니다. 소프트웨어 개발 단계 중 가장 먼저 수행되는 단계를 골라보세요.

출제예상

2. 다음 중 애플리케이션 테스트의 과정으로 옳게 나열된 것은?

- ㉠ 테스트 분석
- ㉡ 테스트 계획
- ㉢ 테스트 실행
- ㉣ 테스트 케이스 설계
- ㉤ 테스트 결과 분석

- ① ㉠ → ㉡ → ㉣ → ㉢ → ㉤
- ② ㉠ → ㉡ → ㉢ → ㉣ → ㉤
- ③ ㉡ → ㉠ → ㉢ → ㉣ → ㉤
- ④ ㉡ → ㉠ → ㉣ → ㉢ → ㉤

어렵지 않은 내용이죠! 혼동된다면 다시 공부하세요.

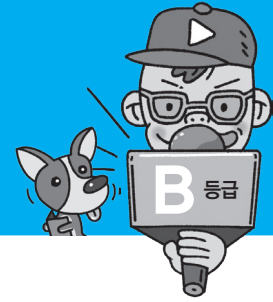
출제예상

3. 다음 중 테스트를 마치면 산출되는 문서가 아닌 것은?

- ① 테스트 계획서
- ② 테스트 케이스
- ③ 테스트 다이어그램
- ④ 테스트 결과서

이번 섹션에서 한 번도 나오지 않은 것이 정답입니다.

▶ 정답 : 1. ① 2. ③ 3. ③



전문가의 조언

테스트 케이스는 개발한 소프트웨어가 제대로 작동하는지를 확인하기 위한 데이터나 실행 조건 등의 집합이고, 테스트 시나리오는 여러 개의 테스트 케이스가 있을 때 이를 적용하는 순서입니다. 그리고 테스트 오라클은 테스트한 결과가 옳은지를 확인하는 도구입니다. 이 세 가지의 개념과 특징을 명확히 구분하여 알아두세요.

명세 기반 테스트

명세 기반 테스트란 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 구현하고 있는지를 확인하는 것입니다.

1 테스트 케이스(Test Case)

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

- 테스트 케이스를 미리 설계하면 테스트 오류를 방지할 수 있고 테스트 수행에 필요한 인력, 시간 등의 낭비를 줄일 수 있다.
- 가장 이상적인 테스트 케이스를 설계하려면 시스템 설계 시 작성해야 한다.

2 테스트 케이스 작성 순서

테스트 케이스는 테스트 전략이나 테스트 계획서 등을 기반으로 하여 다음과 같은 순서로 작성된다.

1. 테스트 계획 검토 및 자료 확보	<ul style="list-style-type: none"> • 테스트 계획서를 재검토하여 테스트 대상 범위 및 접근 방법 등을 이해한다. • 시스템 요구사항과 기능 명세서를 검토하고 테스트 대상 시스템의 정보를 확보한다.
2. 위험 평가 및 우선순위 결정	결함의 위험 정도에 따른 우선순위를 결정하고, 어느 부분에 초점을 맞춰 테스트할지를 결정한다.
3. 테스트 요구사항 정의	시스템에 대한 사용자 요구사항이나 테스트 대상을 재검토하고, 테스트 특성, 조건, 기능 등을 분석한다.
4. 테스트 구조 설계 및 테스트 방법 결정	<ul style="list-style-type: none"> • 테스트 케이스의 형식과 분류 방법을 결정한다. • 테스트 절차, 장비, 도구, 테스트 문서화 방법을 결정한다.
5. 테스트 케이스 정의	요구사항에 따라 테스트 케이스를 작성하고, 입력 값, 실행 조건, 예상 결과 등을 기술한다.
6. 테스트 케이스 타당성 확인 및 유지 보수	<ul style="list-style-type: none"> • 소프트웨어의 기능 또는 환경 변화에 따라 테스트 케이스를 갱신한다. • 테스트 케이스의 유용성을 검토한다.

테스트 케이스		프로젝트명 : 웹사이트 구축	
		대상 시스템명 : web시스템	
단계명 : 단위 테스트		작성자 : 이숙경	작성일 : 2019-09-20
		버전 : 1.0	
테스트 ID	TC-xxx	테스트 일자	2019-10-15
테스트 목적	회원 가입 테스트		
테스트 기능	사용자의 ID 입력 검증		
입력 데이터	사용자 ID		
테스트 단계	케이스 설명		예상 출력 중요도 확인 비고
	사용자가 ID를 4글자 이하로 입력하고 <가입하기> 클릭		오류 메시지
	사용자가 ID를 숫자로만 입력하고 <가입하기> 클릭		오류 메시지
	사용자가 이미 존재하는 ID를 입력하고 <가입하기> 클릭		오류 메시지
	사용자가 올바른 ID를 입력하고 <가입하기> 클릭		사용 가능 메시지
테스트 환경	개발 환경의 테스트 서버, 형상 관리 서버		
전제 조건			
성공/실패 기준	기대 결과가 정상적으로 출력되면 성공		
기타 테스트 의견 사항			

공통 작성 항목

개별 작성 항목

단위 테스트 케이스 작성 예

3 테스트 시나리오(Test Scenario)

테스트 시나리오는 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스들을 묶은 집합으로, 테스트 케이스들을 적용하는 구체적인 절차를 명세한 문서이다.

- 테스트 시나리오에는 테스트 순서에 대한 구체적인 절차, 사전 조건, 입력 데이터 등이 설정되어 있다.
- 테스트 시나리오를 통해 테스트 순서를 미리 정함으로써 테스트 항목을 빠짐없이 수행할 수 있다.

유스케이스(Use Case)

유스케이스는 사용자 측면에서의 요구사항으로, 사용자가 원하는 목표를 달성하기 위해 수행할 내용을 기술합니다.

통합 테스트 시나리오

프로젝트명 : 로그인 다양화

대상 시스템명 : xx시스템

문서번호 : LOGxxx-T095

작성자 : 이숙경

작성일 : 2019-09-20

버전 : 1.0

테스트 목적 로그인 방법

테스트 ID	테스트명	번호	메뉴 경로	테스트 케이스	예상 결과	확인
WEB_LT34	로그인	1	로그인	- ID/PW 입력 후 <로그인> 클릭 - 로그인 상태 유지 선택	- 정상적으로 로그인 된다. - 로그인 후 로그인 상태가 유지된다.	
		2	로그인 → 공인인증서	- 공인인증서 로그인 여부 확인 - 공인인증서 선택하고 PW 입력 후 <확인> 클릭	- 정상적으로 로그인 된다. - 로그인 후 로그인 상태가 유지된다.	
		3	로그인 → 네이버	- 네이버 ID/PW 입력 후 <로그인> 클릭 - 로그인 상태 유지 선택 - 일회용 로그인 클릭	- 정상적으로 로그인 된다. - 로그인 후 로그인 상태가 유지된다. - 일회용 로그인 번호 입력 창이 실행된다.	
		4	로그인 → 카카오	- 카카오 ID/PW 입력 후 <로그인> 클릭 - 로그인 상태 유지 선택	- 정상적으로 로그인 된다. - 로그인 후 로그인 상태가 유지된다.	

통합 테스트 시나리오 작성 예

4 테스트 시나리오 작성 시 유의 사항

- 테스트 시나리오는 시스템별, 모듈별, 항목별 등과 같이 여러 개의 시나리오로 분리하여 작성해야 한다.
- 테스트 시나리오는 사용자의 요구사항과 설계 문서 등을 토대로 작성해야 한다.
- 각각의 테스트 항목은 식별자 번호, 순서 번호, 테스트 데이터, 테스트 케이스, 예상 결과, 확인 등을 포함해서 작성해야 한다.
- 테스트 시나리오는 유스케이스(Use Case)* 간 업무 흐름이 정상적인지를 테스트 할 수 있도록 작성해야 한다.
- 테스트 시나리오는 개발된 모듈 또는 프로그램 간의 연계가 정상적으로 동작하는 지 테스트할 수 있도록 작성해야 한다.

5 테스트 오라클(Test Oracle)

테스트 오라클은 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법 및 활동을 말한다.

- 테스트 오라클은 결과를 판단하기 위해 테스트 케이스에 대한 예상 결과를 계산하거나 확인한다.
- 테스트 오라클의 특징
 - 제한된 검증 : 테스트 오라클을 모든 테스트 케이스에 적용할 수 없다.
 - 수학적 기법 : 테스트 오라클의 값을 수학적 기법을 이용하여 구할 수 있다.
 - 자동화 기능 : 테스트 대상 프로그램의 실행, 결과 비교, 커버리지 측정 등을 자동화 할 수 있다.
- 오라클의 종류에는 참(True) 오라클, 샘플링(Sampling) 오라클, 추정(Heuristic) 오라클, 일관성 검사(Consistent) 오라클 등이 있다.

6 테스트 오라클의 종류

참 오라클은 주로 항공기, 은행, 발전소 소프트웨어 등 미션 크리티컬*한 업무에 사용되고, 샘플링 오라클과 추정 오라클은 일반적인 업무, 게임, 오락 등에 사용된다.

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



전문가의 조언

테스트 케이스를 적용한 결과가 맞았는지, 틀렸는지를 판단하려면 기준 값이 있어야 합니다. 그 값을 계산하거나 확인하는 기법을 테스트 오라클이라고 합니다. 테스트 오라클이 무엇인지 기억해 두세요.

미션 크리티컬(Mission Critical)

미션 크리티컬은 단 한 번이라도 다운되면 시스템 전체에 치명적인 영향을 주므로 절대 다운되면 안 되는 시스템으로, 항공기 운행, 은행의 온라인 시스템 등이 해당됩니다.

기대하는 결과

예를 들면, 퇴직금을 계산하는 프로그램에서 근무기간을 5로 넣어 테스트 케이스를 실행하였을 경우 예상되는 퇴직금이 기대 결과가 되는 것입니다.



기출문제 따라잡기

Section 055

출제예상

1. 테스트 케이스의 예상 결과로, 테스트 결과가 올바른지 판단하기 위한 근거가 되는 것은 무엇인가?

- ① 테스트 케이스
- ② 테스트 하네스
- ③ 테스트 오라클
- ④ 테스트 시나리오

테스트 하네스는 애플리케이션의 컴포넌트 및 모듈을 테스트하는 환경의 일부로서, 테스트를 지원하기 위해 생성된 코드와 데이터를 의미합니다. 테스트 하네스는 다음 섹션에서 배웁니다.

출제예상

2. 다음 중 테스트 케이스에 대한 설명으로 가장 먼 것은?

- ① 테스트를 위한 설계 산출물이다.
- ② 소프트웨어가 사용자의 요구사항을 얼마나 준수했는지 확인하기 위해 입력 값, 실행 조건, 기대 결과 등으로 만들어진 테스트 항목의 명세서이다.
- ③ 테스트 케이스를 미리 설계하면 테스트 오류를 방지할 수 있다.
- ④ 테스트 케이스는 시스템 구현 시 설계하는 것이 가장 이상적이다.

결함은 될 수 있으면 빨리 발견하는 것이 좋습니다. 결함을 빨리 발견하려면 어떻게 해야 할까요?

출제예상

3. 다음 중 테스트 시나리오(Test Scenario)에 대한 설명으로 가장 옳지 않은 것은?

- ① 테스트 시나리오는 테스트 수행을 위한 여러 개의 테스트 케이스들의 집합이다.
- ② 테스트 시나리오는 어떤 기능을 어떤 순서대로 테스트 할 것인지 절차를 기술한다.
- ③ 테스트 시나리오는 전체를 하나의 시나리오로 작성해야 한다.
- ④ 테스트 시나리오를 통해 테스트 순서를 미리 정함으로써 테스트 항목을 빠짐없이 수행할 수 있다.

테스트 시나리오는 시스템별, 모듈별, 항목별과 같이 여러 개의 시나리오로 분리하여 작성해야 합니다.

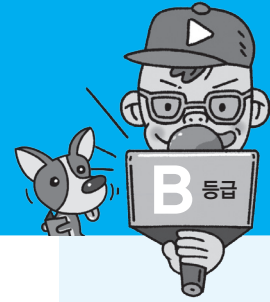
출제예상

4. 다음 중 테스트 오라클의 종류가 아닌 것은?

- ① 참(True) 오라클
- ② 거짓(False) 오라클
- ③ 샘플링(Sampling) 오라클
- ④ 추정(Heuristic) 오라클

바로 답을 찾을 수 있죠? 모르겠다면 다시 공부하세요.

▶ 정답 : 1. ③ 2. ④ 3. ③ 4. ②



1 테스트 자동화의 개념

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

- 테스트 자동화 도구를 사용함으로써 휴먼 에러(Human Error)*를 줄이고 테스트의 정확성을 유지하면서 테스트의 품질을 향상시킬 수 있다.

2 테스트 자동화 도구의 장점 / 단점

장점	<ul style="list-style-type: none"> • 테스트 데이터의 재입력, 재구성 같은 반복적인 작업을 자동화함으로써 인력 및 시간을 줄일 수 있다. • 다중 플랫폼 호환성, 소프트웨어 구성, 기본 테스트 등 향상된 테스트 품질을 보장한다. • 사용자의 요구사항 등을 일관성 있게 검증할 수 있다. • 테스트 결과에 대한 객관적인 평가 기준을 제공한다. • 테스트 결과를 그래프 등 다양한 표시 형태로 제공한다. • UI가 없는 서비스도 정밀 테스트가 가능하다.
단점	<ul style="list-style-type: none"> • 테스트 자동화 도구의 사용 방법에 대한 교육 및 학습이 필요하다. • 자동화 도구를 프로세스 단계별로 적용하기 위한 시간, 비용, 노력이 필요하다. • 비공개 상용 도구*의 경우 고가의 추가 비용이 필요하다.

3 테스트 자동화 수행 시 고려사항

- 테스트 절차를 고려하여 재사용 및 측정이 불가능한 테스트 프로그램은 제외한다.
- 모든 테스트 과정을 자동화 할 수 있는 도구는 없으므로 용도에 맞는 적절한 도구를 선택해서 사용한다.
- 자동화 도구의 환경 설정 및 습득 기간을 고려해서 프로젝트 일정을 계획해야 한다.
- 테스트 엔지니어의 투입 시기가 늦어지면 프로젝트의 이해 부족으로 인해 불완전한 테스트를 초래할 수 있으므로 반드시 프로젝트 초기에 테스트 엔지니어의 투입 시기를 계획해야 한다.

4 테스트 자동화 도구의 유형

테스트 자동화 도구는 테스트를 수행하는 유형에 따라 다음과 같이 분류된다.

전문가의 조언

테스트 자동화 도구는 말 그대로 테스트를 자동화 할 수 있도록 도와주는 도구입니다. 왜 테스트를 자동화하는지를 생각하며 테스트 자동화 도구의 장/단점, 자동화 수행 시 고려사항 등을 정리해 두세요.

테스트 스크립트(Test Script)

테스트 스크립트는 테스트 실행 절차나 수행 방법 등을 스크립트 언어로 작성한 파일입니다.

※ 스크립트 언어 : 소스 코드를 컴파일하지 않고도 내장된 번역기에 의해 번역되어 바로 실행할 수 있는 언어

휴먼 에러(Human Error)

휴먼 에러는 사람의 판단 실수나 조작 실수 등으로 인해 발생하는 에러입니다.

비공개 상용 도구

비공개 상용 도구는 특정 기업체 전용으로 개발되어 독점 공급되는 소프트웨어를 의미합니다.

형상 관리 도구

형상 관리 도구는 테스트 수행에 필요한 다양한 도구 및 데이터를 관리하는 도구입니다.

테스트 스위트와 테스트 시나리오의 차이

테스트 스위트와 테스트 시나리오는 둘 다 테스트 케이스의 묶음입니다. 테스트 스위트가 여러 개의 테스트 케이스의 단순한 묶음이라면 테스트 시나리오는 테스트 케이스의 동작 순서에 따른 묶음입니다.

정적 분석 도구 (Static Analysis Tools)	<ul style="list-style-type: none"> • 프로그램을 실행하지 않고 분석하는 도구로, 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도 및 남은 결함 등을 발견하기 위해 사용된다. • 테스트를 수행하는 사람이 작성된 소스 코드를 이해하고 있어야만 분석이 가능하다.
테스트 실행 도구 (Test Execution Tools)	<ul style="list-style-type: none"> • 스크립트 언어를 사용하여 테스트를 실행하는 방법으로, 테스트 데이터와 테스트 수행 방법 등이 포함된 스크립트를 작성한 후 실행한다. • 데이터 주도 접근 방식 <ul style="list-style-type: none"> - 스프레드시트에 테스트 데이터를 저장하고, 이를 읽어 실행하는 방식이다. - 다양한 테스트 데이터를 동일한 테스트 케이스로 반복하여 실행할 수 있다. - 스크립트에 익숙하지 않은 사용자도 미리 작성된 스크립트에 테스트 데이터만 추가하여 테스트할 수 있다. • 키워드 주도 접근 방식 <ul style="list-style-type: none"> - 스프레드시트에 테스트를 수행할 동작을 나타내는 키워드와 테스트 데이터를 저장하여 실행하는 방식이다. - 키워드를 이용하여 테스트를 정의할 수 있다.
성능 테스트 도구 (Performance Test Tools)	애플리케이션의 처리량, 응답 시간, 경과 시간, 자원 사용률 등을 인위적으로 적용한 가상의 사용자를 만들어 테스트를 수행함으로써 성능의 목표 달성 여부를 확인한다.
테스트 통제 도구 (Test Control Tools)	테스트 계획 및 관리, 테스트 수행, 결함 관리 등을 수행하는 도구로, 종류에는 형상 관리 도구*, 결함 추적/관리 도구 등이 있다.
테스트 하네스 도구 (Test Harness Tools)	<ul style="list-style-type: none"> • 테스트 하네스는 애플리케이션의 컴포넌트 및 모듈을 테스트하는 환경의 일부분으로, 테스트를 지원하기 위해 생성된 코드와 데이터를 의미한다. • 테스트 하네스 도구는 테스트가 실행될 환경을 시뮬레이션 하여 컴포넌트 및 모듈이 정상적으로 테스트되도록 한다.

잠깐만요



테스트 하네스(Test Harness)의 구성 요소

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

5 테스트 수행 단계별 테스트 자동화 도구

테스트 수행 단계를 테스트 계획, 테스트 분석/설계, 테스트 수행, 테스트 관리로 분류하였을 경우 각 단계에 해당하는 테스트 자동화 도구는 다음과 같다.

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



기출문제 따라잡기

Section 056

출제예상

1. 다음 중 테스트 자동화에 대한 설명으로 틀린 것은?

- ① 테스트 자동화는 사람이 하던 반복적인 테스트 절차를 자동화하는 것이다.
- ② 테스트 자동화를 수행하면 휴먼 에러(Human Error)를 줄일 수 있다.
- ③ 자동화를 수행하면 테스트 인력 및 시간을 단축할 수 있다.
- ④ 테스트 자동화 도구는 모두 무료이므로 추가 비용이 들지 않는다.

테스트 자동화 도구 중 일부는 상용 도구로, 고가이며 유지 관리 비용이 높아 추가 투자 비용이 많이 듭니다.

출제예상

2. 다음 중 테스트 자동화 수행 시 고려할 사항으로 가장 거리가 먼 것은?

- ① 자동화 도구는 테스트 과정 전체를 자동화 할 수 있으므로 하나의 도구를 선택해서 사용하면 된다.
- ② 테스트 자동화 시 측정이 불가능한 테스트 프로그램은 제외해야 한다.
- ③ 테스트 엔지니어의 투입 시기는 프로젝트 초기에 계획해야 한다.
- ④ 자동화 도구의 환경 설정 및 습득 기간을 고려해서 프로젝트 일정을 계획해야 한다.

모든 테스트 과정을 자동화 할 수 있는 도구는 없으므로 용도에 맞는 적절한 도구를 그때그때 선택해서 사용해야 합니다.

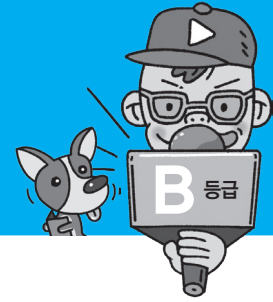
출제예상

3. 다음 중 테스트 하네스(Test Harness)의 구성 요소에 대한 설명으로 틀린 것은?

- ① 테스트 스텝은 테스트 대상 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출한다.
- ② 테스트 케이스는 입력 값, 실행 조건, 기대 결과 등의 집합이다.
- ③ 테스트 스크립트는 자동화된 테스트 실행 절차에 대한 명세를 말한다.
- ④ 목 오브젝트는 사전에 사용자의 행위를 조건부로 입력해 두면, 그 상황에 맞는 예정된 행위를 수행하는 객체를 말한다.

테스트 드라이버와 테스트 스텝을 혼동하지 마세요. ①번은 테스트 드라이버에 대한 설명입니다.

▶ 정답 : 1. ④ 2. ① 3. ①



전문가의 조언

결함을 발견하는 것만큼이나 발견된 결함을 체계적으로 관리하는 것도 중요합니다. 테스트에서 발견된 결함을 기록하고 결함의 원인을 분석하여 해결한 후 결함의 재발생을 방지하기 위한 활동 등이 모두 결함 관리에 해당합니다. 결함 관리의 개념을 바탕으로 결함 관리 및 추적 순서 등을 정리해 두세요.

결함 관리 프로세스

'Section 054 애플리케이션 테스트 프로세스'의 마지막 단계에도 '결함 관리 프로세스'가 있었는데 이번 섹션에도 '결함 관리 프로세스'가 있는데요, 두 '결함 관리 프로세스' 모두 결함을 관리하는 것은 동일한데, 결함을 관리함에 있어 초점을 어디에 두느냐에 따라 순서가 조금 다르게 표현된 것뿐입니다. 'Section 054 애플리케이션 테스트 프로세스'의 '결함 관리 프로세스'는 단계별 테스트 중 발생한 에러에 대해 이것이 결함인지 아닌지 판별하는 것에 초점을 뒀다면 이번 Section의 '결함 관리 프로세스'는 발견된 결함의 처리 과정에 초점을 둔 것입니다. 두 경우를 구분하여 알아두세요.

- **프로그램 리더** : 소프트웨어 설계, 구현 등 소프트웨어의 기술 분야를 책임지는 사람
- **품질 관리(QA) 담당자** : 제품에 대한 고객만족을 목표로 제품의 생산부터 판매, 폐기에 이르는 전 과정을 관리하는 사람

대시보드

대시보드는 다양한 데이터를 쉽게 모니터링 할 수 있도록 만든 일종의 상황판을 말합니다.

1 결함(Fault)의 정의

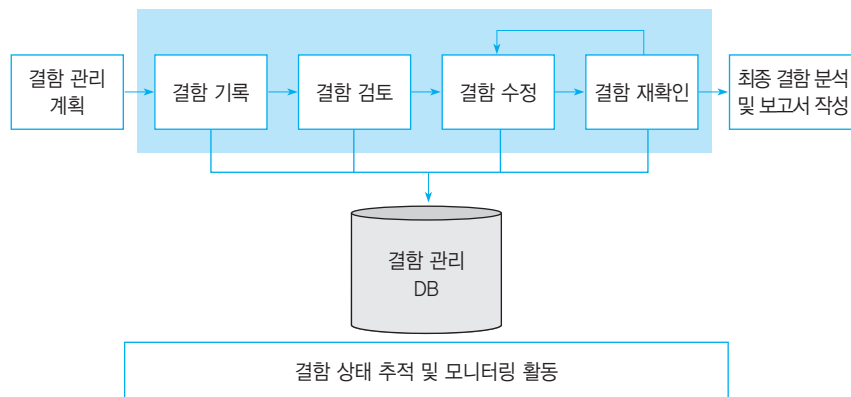
결함(Fault)은 오류 발생, 작동 실패 등과 같이 소프트웨어가 개발자가 설계한 것과 다르게 동작하거나 다른 결과가 발생하는 것을 의미한다.

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

2 결함 관리 프로세스*

결함 관리 프로세스는 애플리케이션 테스트에서 발견된 결함을 처리하는 것으로, 처리 순서는 다음과 같다.

- ① **결함 관리 계획** : 전체 프로세스에 대한 결함 관리 일정, 인력, 업무 프로세스 등을 확보하여 계획을 수립하는 단계이다.
- ② **결함 기록** : 테스터는 발견된 결함을 결함 관리 DB에 등록한다.
- ③ **결함 검토** : 테스터, 프로그램 리더*, 품질 관리(QA) 담당자* 등은 등록된 결함을 검토하고 결함을 수정할 개발자에게 전달한다.
- ④ **결함 수정** : 개발자는 전달받은 결함을 수정한다.
- ⑤ **결함 재확인** : 테스터는 개발자가 수정한 내용을 확인하고 다시 테스트를 수행한다.
- ⑥ **결함 상태 추적 및 모니터링 활동** : 결함 관리 DB를 이용하여 프로젝트별 결함 유형, 발생률 등을 한눈에 볼 수 있는 대시보드* 또는 게시판 형태의 서비스를 제공한다.
- ⑦ **최종 결함 분석 및 보고서 작성** : 발견된 결함에 대한 정보와 이해관계자들의 의견이 반영된 보고서를 작성하고 결함 관리를 종료한다.



결함 관리 프로세스 흐름도

3 결함 상태 추적

테스트에서 발견된 결함은 지속적으로 상태 변화를 추적하고 관리해야 한다.

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

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

4 결함 추적 순서

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

- ❶ **결함 등록(Open)** : 테스터와 품질 관리(QA) 담당자에 의해 발견된 결함이 등록된 상태
- ❷ **결함 검토(Reviewed)** : 등록된 결함을 테스터, 품질 관리(QA) 담당자, 프로그램 리더, 담당 모듈 개발자에 의해 검토된 상태
- ❸ **결함 할당(Assigned)** : 결함을 수정하기 위해 개발자와 문제 해결 담당자에게 결함이 할당된 상태
- ❹ **결함 수정(Resolved)** : 개발자가 결함 수정을 완료한 상태
- ❺ **결함 조치 보류(Deferred)** : 결함의 수정이 불가능해 연기된 상태로, 우선순위, 일정 등에 따라 재오픈을 준비중인 상태
- ❻ **결함 종료(Closed)** : 결함이 해결되어 테스터와 품질 관리(QA) 담당자가 종료를 승인한 상태
- ❼ **결함 해제(Clarified)** : 테스터, 프로그램 리더, 품질 관리(QA) 담당자가 종료 승인한 결함을 검토하여 결함이 아니라고 판명한 상태

5 결함 분류

테스트에서 발견되는 결함을 유형별로 분류하면 다음과 같다.

시스템 결함	시스템 다운, 애플리케이션의 작동 정지, 종료, 응답 시간 지연, 데이터베이스 에러 등 주로 애플리케이션 환경이나 데이터베이스 처리에서 발생된 결함
기능 결함	사용자의 요구사항 미반영/불일치, 부정확한 비즈니스 프로세스, 스크립트 오류, 타 시스템 연동 시 오류 등 애플리케이션의 기획, 설계, 업무 시나리오 등의 단계에서 유입된 결함
GUI 결함	UI 비일관성, 데이터 타입의 표시 오류, 부정확한 커서/메시지 오류 등 사용자 화면 설계에서 발생된 결함

문서 결함

사용자의 요구사항과 기능 요구사항의 불일치로 인한 불안정한 상태의 문서, 사용자의 온라인/오프라인 매뉴얼의 불일치 등 기획자, 사용자, 개발자 간의 의사소통 및 기록이 원활하지 않아 발생한 결함

잠깐만요



테스트 단계별 유입 결함

- 기획 시 유입되는 결함 : 사용자 요구사항의 표준 미준수로 인한 테스트 불가능, 요구사항 불명확/불완전/불일치 결함 등
- 설계 시 유입되는 결함 : 설계 표준 미준수로 인한 테스트 불가능, 기능 설계 불명확/불완전/불일치 결함 등
- 코딩 시 유입되는 결함 : 코딩 표준 미준수로 인한 기능의 불일치/불완전, 데이터 결함, 인터페이스 결함 등
- 테스트 부족으로 유입되는 결함 : 테스트 수행 시 테스트 완료 기준의 미준수, 테스트팀과 개발팀의 의사소통 부족, 개발자의 코딩 실수로 인한 결함 등

6 결함 심각도

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

- 결함 심각도를 우선순위에 따라 분류*하면 다음과 같다.

High	핵심 요구사항 미구현, 장시간 시스템 응답 지연, 시스템 다운 등과 같이 더 이상 프로세스를 진행할 수 없도록 만드는 결함
Medium	부정확한 기능이나 데이터베이스 에러 등과 같이 시스템 흐름에 영향을 미치는 결함
Low	부정확한 GUI 및 메시지, 에러 시 메시지 미출력, 화면상의 문법/철자 오류 등과 같이 시스템 흐름에는 영향을 미치지 않는 결함

7 결함 우선순위

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

- 일반적으로 결함의 심각도가 높으면 우선순위도 높지만 애플리케이션의 특성에 따라 우선순위가 결정될 수도 있기 때문에 심각도가 높다고 반드시 우선순위가 높은 것은 아니다.
- 결함 우선순위는 결정적(Critical), 높음(High), 보통(Medium), 낮음(Low) 또는 즉시 해결, 주의 요망, 대기, 개선 권고 등으로 분류된다.

결함 심각도 분류

결함 심각도는 High, Medium, Low 외의 다른 기준으로도 분류할 수 있습니다.

예 치명적(Critical), 주요(Major), 보통(Normal), 경미(Minor), 단순(Simple)

8 결함 관리 도구

결함 관리 도구는 소프트웨어에 발생한 결함을 체계적으로 관리할 수 있도록 도와주는 도구로, 다음과 같은 것들이 있다.

Mantis	결함 및 이슈 관리 도구로, 소프트웨어 설계 시 단위별 작업 내용을 기록할 수 있어 결함 추적도 가능한 도구
Trac	결함 추적은 물론 결함을 통합하여 관리할 수 있는 도구
Redmine	프로젝트 관리 및 결함 추적이 가능한 도구
Bugzilla	결함 신고, 확인, 처리 등 결함을 지속적으로 관리할 수 있는 도구로, 결함의 심각도와 우선 순위를 지정할 수도 있음



기출문제 따라잡기

Section 057

출제예상

1. 다음 중 결함에 대한 설명으로 틀린 것은?

- ① 소프트웨어가 사용자의 기대치를 만족시키지 못해 변경이 필요한 것은 모두 결함이다.
- ② 발견된 결함은 지속적으로 상태 변화를 추적하고 관리해야 한다.
- ③ 결함의 심각도는 애플리케이션에 발생한 결함이 전체 시스템에 얼마나 치명적인지를 나타내는 척도이다.
- ④ 결함의 심각도가 높으면 반드시 결함의 우선순위가 높다.

애플리케이션의 특성에 따라 우선순위가 결정될 수도 있기 때문에 결함의 심각도가 높다고 해서 반드시 우선순위가 높은 것은 아닙니다.

출제예상

2. 다음 중 결함에 관한 설명으로 틀린 것은?

- ① 결함은 사용자의 기대 결과와 실제 소프트웨어를 실행했을 때의 결과 간의 차이를 의미한다.
- ② 결함 심각도는 우선순위에 따라 High, Medium, Low로 분류하기도 한다.
- ③ 결함 에이징은 테스트 진행 시간에 따른 결함 수를 측정한다.
- ④ 결함의 우선순위는 결함의 중요도와 심각도에 따라 결정된다.

결함 에이징은 특정 결함 상태로 지속되는 시간을 측정합니다. 테스트 진행 시간에 따른 결함 수를 측정하는 것은 결함 추세를 분석하기 위함입니다.

▶ 정답 : 1. ④ 2. ③



전문가의 조언

애플리케이션 성능 분석 도구로 애플리케이션의 성능을 분석하면 성능 저하 요인을 찾고 이를 해결할 수 있습니다. 애플리케이션의 성능 분석 도구의 종류 및 기능을 파악하고 성능 저하 요인에는 어떤 것이 있는지 알아두세요.

- 부하(Load) 테스트 : 애플리케이션에 일정 시간 동안 부하를 가하면서 반응을 측정하는 테스트
- 스트레스(Stress) 테스트 : 부하 테스트를 확장한 테스트로, 애플리케이션이 과부하 상태에서 어떻게 작동하는지 테스트함

1 애플리케이션 성능

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

- 애플리케이션 성능 측정 지표

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

- 애플리케이션의 성능 분석 도구는 애플리케이션의 성능을 테스트하는 도구와 시스템을 모니터링하는 도구로 분류된다.

2 성능 테스트 도구

성능 테스트 도구는 애플리케이션의 성능을 테스트하기 위해 애플리케이션에 부하*나 스트레스*를 가하면서 애플리케이션의 성능 측정 지표를 점검하는 도구이다.

- 종류

도구명	도구 설명	지원 환경
JMeter	HTTP, FTP 등 다양한 프로토콜을 지원하는 부하 테스트 도구	Cross-Platform
LoadUI	<ul style="list-style-type: none"> • 서버 모니터링, Drag&Drop 등 사용자의 편리성이 강화된 부하 테스트 도구 • HTTP, JDBC 등 다양한 프로토콜 지원 	Cross-Platform
OpenSTA	HTTP, HTTPS 프로토콜에 대한 부하 테스트 및 생산품 모니터링 도구	Windows

3 시스템 모니터링(Monitoring) 도구

시스템 모니터링 도구는 애플리케이션이 실행되었을 때 시스템 자원의 사용량을 확인하고 분석하는 도구이다.

- 시스템 모니터링 도구는 성능 저하의 원인 분석, 시스템 부하량 분석, 사용자 분석 등 시스템을 안정적으로 운영할 수 있는 기능을 제공한다.
- 종류

도구명	도구 설명	지원 환경
Scouter	<ul style="list-style-type: none"> • 단일 뷰 통합/실시간 모니터링, 튜닝에 최적화된 인프라 통합 모니터링 도구 • 애플리케이션의 성능을 모니터링/통제하는 도구 	Cross-Platform
Zabbix	웹기반 서버, 서비스, 애플리케이션 등의 모니터링 도구	Cross-Platform

4 애플리케이션 성능 저하 원인 분석

애플리케이션의 성능 저하 현상은 애플리케이션을 DB에 연결하기 위해 Connection 객체*를 생성하거나 쿼리를 실행하는 애플리케이션 로직에서 많이 발생한다.

- 다음은 애플리케이션의 성능 저하 현상을 발생시키는 주요 요인이다.
 - DB에 필요 이상의 많은 데이터를 요청한 경우
 - 데이터베이스의 락(DB Lock)*이 해제되기를 기다리면서 애플리케이션이 대기하거나 타임아웃된 경우
 - 커넥션 풀(Connection Pool)*의 크기를 너무 작거나 크게 설정한 경우
 - JDBC*나 ODBC* 같은 미들웨어*를 사용한 후 종료하지 않아 연결 누수(Connection Leak)*가 발생한 경우
 - 트랜잭션*이 확정(Commit)*되지 않고 커넥션 풀에 반환되거나, 잘못 작성된 코드로 인해 불필요한 Commit이 자주 발생하는 경우
 - 인터넷 접속 불량으로 인해 서버 소켓(Server Socket)*에 쓰기는 지속되나, 클라이언트에서 정상적인 읽기가 수행되지 않는 경우
 - 대량의 파일을 업로드하거나 다운로드하여 처리 시간이 길어진 경우
 - 트랜잭션 처리 중 외부 호출이 장시간 수행되거나 타임아웃된 경우
 - 네트워크 관련 장비 간 데이터 전송이 실패하거나 전송 지연으로 인해 데이터 손실이 발생한 경우

- Connection 객체 : 특정 데이터 원본과 연결할 수 있게 해주는 객체로, 클라이언트/서버 데이터베이스 시스템의 경우 서버에 대한 실제 네트워크 연결과 동일함
- 데이터베이스 락(DB Lock) : 데이터베이스 락은 어떤 사람이 데이터베이스를 사용하고 있을 때 다른 사람들이 데이터베이스를 사용할 수 없도록 잠그는 것을 말한다. 예를 들어, 영화표를 예매할 때 한 개의 좌석에 대해서는 한 명만 예약할 수 있기 때문에 한 사람이 예약을 시작하면 결재를 마칠때까지 다른 사람들은 해당 좌석을 예약할 수 없도록 락을 설정할 수 있음
- 커넥션 풀(Connection Pool) : 데이터베이스와 연결된 커넥션을 풀(pool)에 미리 만들어 놓고 커넥션이 필요할 때 풀에서 꺼내 사용하고 다시 반환하는 방법
- JDBC(Java DataBase Connectivity) : 자바 언어로 데이터베이스 연결, SQL문 실행 등을 처리하기 위한 자바 표준 인터페이스
- ODBC(Open DataBase Connectivity) : 개방형 데이터베이스 표준 접속 규격으로, 공통적인 인터페이스를 통해 서로 다른 데이터베이스 파일을 연결하여 사용할 수 있음
- 미들웨어(MiddleWare) : 운영체제와 해당 운영체제에 의해 실행되는 응용 프로그램 사이에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공하는 소프트웨어
- 연결 누수(Connection Leak) : JDBC 등이 DB에 접근하기 위해 커넥션 풀에 저장된 커넥션을 사용하는데, 사용 후에 커넥션이 반납되지 않아 커넥션 풀이 지속적으로 줄어드는 현상을 의미함
- 트랜잭션(Transaction) : 컴퓨터가 처리해야 할 단위 작업
- 확정(Commit) : 명령에 의해 수행된 결과를 실제 물리적 디스크에 저장하고 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려주는 것
- 서버 소켓(Server Socket) : 소켓은 프로세스 사이의 대화를 가능하게 하는 쌍방향 통신 방식으로, 연결을 요청하는 소켓을 클라이언트 소켓, 요청을 받아들이는 소켓을 서버 소켓이라고 함



기출문제 따라잡기

Section 058

출제예상

1. 다음 중 애플리케이션의 성능을 측정하기 위한 지표가 아닌 것은?

- ① 신뢰도(Reliability)
- ② 처리량(Throughput)
- ③ 경과 시간(Turn Around Time)
- ④ 응답 시간(Response Time)

보기 중에서 성능과 가장 거리가 먼 것을 찾아보세요.

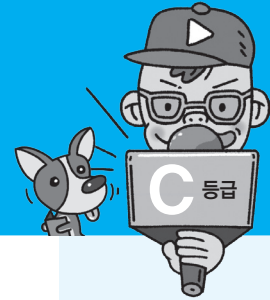
출제예상

2. 다음 중 애플리케이션의 성능을 저하하는 원인이 아닌 것은?

- ① DB에 필요 이상의 많은 데이터를 요청하면 애플리케이션의 성능 저하 현상이 발생할 수 있다.
- ② 애플리케이션의 성능 저하 현상은 커넥션 풀의 크기를 크게 하면 발생하지 않지만 작게 하면 발생할 수 있다.
- ③ 대량의 파일을 업로드하거나 다운로드하여 처리 시간이 길어진 경우 애플리케이션의 성능 저하 현상이 발생할 수 있다.
- ④ 애플리케이션에 연결된 데이터베이스의 락(DB Lock)으로 인해 애플리케이션의 성능 저하 현상이 발생할 수 있다.

커넥션 풀(Connection Pool)의 크기를 너무 작거나 크게 설정하면 애플리케이션의 성능 저하 현상이 발생할 수 있습니다.

▶ 정답 : 1. ① 2. ②



1 소스 코드 최적화

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

- **클린 코드(Clean Code)** : 누구나 쉽게 이해하고 수정 및 추가할 수 있는 단순, 명료한 코드, 즉 잘 작성된 코드를 의미한다.
- **나쁜 코드(Bad Code)** : 프로그램의 로직(Logic)이 복잡하고 이해하기 어려운 코드로, 코드의 로직이 서로 얽혀 있는 스파게티 코드, 동일한 처리 로직이 중복되게 작성된 코드 등이 여기에 해당한다.
- 나쁜 코드로 작성된 애플리케이션의 코드를 클린 코드로 수정하면 애플리케이션의 성능이 개선된다.
- **클린 코드 작성 원칙**

가독성	<ul style="list-style-type: none"> • 누구든지 코드를 쉽게 읽을 수 있도록 작성한다. • 코드 작성 시 이해하기 쉬운 용어를 사용하거나 들여쓰기 기능 등을 사용한다.
단순성	<ul style="list-style-type: none"> • 코드를 간단하게 작성한다. • 한 번에 한 가지를 처리하도록 코드를 작성하고 클래스/메소드/함수 등을 최소 단위로 분리한다.
의존성 배제	<ul style="list-style-type: none"> • 코드가 다른 모듈에 미치는 영향을 최소화한다. • 코드 변경 시 다른 부분에 영향이 없도록 작성한다.
중복성 최소화	<ul style="list-style-type: none"> • 코드의 중복을 최소화한다. • 중복된 코드는 삭제하고 공통된 코드를 사용한다.
추상화	상위 클래스/메소드/함수에서는 간략하게 애플리케이션의 특성을 나타내고, 상세 내용은 하위 클래스/메소드/함수에서 구현한다.

2 소스 코드 최적화 유형

- **클래스 분할 배치** : 하나의 클래스는 하나의 역할만 수행하도록 응집도*를 높이고, 크기를 작게 작성한다.
- **느슨한 결합(Loosely Coupled)** : 인터페이스 클래스*를 이용하여 추상화*된 자료 구조와 메소드를 구현함으로써 클래스 간의 의존성을 최소화한다.
- **코딩 형식 준수** : 코드 작성 시 다음의 형식을 준수한다.
 - 줄 바꿈 사용
 - 개념적 유사성이 높은 종속 함수 사용
 - 호출하는 함수는 선배치, 호출되는 함수는 후배치
 - 지역 변수는 각 함수의 맨 처음에 선언

전문가의 조언

애플리케이션의 성능을 개선하려면 소스 코드 최적화를 수행하여 나쁜 코드를 클린 코드로 수정하고, 소스 코드 품질 분석 도구를 이용하여 소스 코드의 스타일이나 코딩 표준 등을 분석해야 합니다. 이번 섹션에서는 소스 코드 최적화 기법과 소스 코드 품질 분석 도구의 개념 및 방법 등을 정리해 두세요.

응집도(Cohesion)

응집도는 명령어나 호출문 등 모듈의 내부 요소들이 서로 관련되어 있는 정도, 즉 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미합니다.

인터페이스 클래스

인터페이스 클래스는 클래스나 객체의 사용 방법을 정의한 것으로, 개발 코드와 클래스 사이에서 통신 역할을 합니다. 개발 코드가 클래스의 메소드를 직접 호출하지 않고 중간 매체인 인터페이스 클래스를 사용하는 이유는 개발 코드를 수정하지 않고 실행 내용이 나 리턴값을 다양하게 변경할 수 있기 때문입니다. 이럴 경우 클래스를 직접 사용하지 않으므로 클래스 간의 의존성이 줄어듭니다.

추상화(Abstraction)

추상화는 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화하는 것입니다.

스레드(Thread)

스레드는 프로세스 내에서의 작업 단위로써 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위를 의미합니다.

- **좋은 이름 사용** : 변수나 함수 등의 이름은 기억하기 좋은 이름, 발음이 쉬운 용어, 접두어 사용 등 기본적인 이름 명명 규칙(Naming Rule)을 정의하고 규칙에 맞는 이름을 사용한다.
- **적절한 주석문 사용** : 소스 코드 작성 시 앞으로 해야 할 일을 기록하거나 중요한 코드를 강조할 때 주석문을 사용한다.

3 소스 코드 품질 분석 도구

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

- 정적 분석 도구
 - 작성한 소스 코드를 실행하지 않고 코딩 표준이나 코딩 스타일, 결함 등을 확인하는 코드 분석 도구이다.
 - 비교적 애플리케이션 개발 초기의 결함을 찾는데 사용되고, 개발 완료 시점에서는 개발된 소스 코드의 품질을 검증하는 차원에서 사용된다.
 - 동적 분석 도구로는 발견하기 어려운 결함을 찾아내고, 소스 코드에서 코딩의 복잡도, 모델 의존성, 불일치성 등을 분석할 수 있다.
 - 종류 : pmd, cppcheck, SonarQube, checkstyle, ccm, cobertura 등
- 동적 분석 도구
 - 작성한 소스 코드를 실행하여 코드에 존재하는 메모리 누수, 스레드 결함 등을 분석하는 도구이다.
 - 종류 : Avalanche, Valgrind 등

4 소스 코드 품질 분석 도구의 종류

도구	설명	지원 환경
pmd	소스 코드에 대한 미사용 변수, 최적화되지 않은 코드 등 결함을 유발할 수 있는 코드를 검사한다.	Linux, Windows
cppcheck	C/C++ 코드에 대한 메모리 누수, 오버플로우 등 분석	Windows
SonarQube	중복 코드, 복잡도, 코딩 설계 등을 분석하는 소스 분석 통합 플랫폼	Cross-Platform
checkstyle	• 자바 코드에 대해 소스 코드 표준을 따르고 있는지 검사한다. • 다양한 개발 도구에 통합하여 사용 가능하다.	Cross-Platform
ccm	다양한 언어의 코드 복잡도를 분석한다.	Cross-Platform
cobertura	자바 언어의 소스 코드 복잡도 분석 및 테스트 커버리지를 측정한다.	Cross-Platform
Avalanche	• Valgrind 프레임워크 및 STP 기반으로 구현된다. • 프로그램에 대한 결함 및 취약점 등을 분석한다.	Linux, Android
Valgrind	프로그램 내에 존재하는 메모리 및 스레드 결함 등을 분석한다.	Cross-Platform



기출문제 따라잡기

Section 059

출제예상

1. 다음 중 클린 코드 작성 원칙에 대한 설명으로 틀린 것은?

- ① 중복이 최대화된 코드를 작성한다.
- ② 누구든지 쉽게 이해하는 코드를 작성한다.
- ③ 다른 모듈에 미치는 영향이 최소화된 코드를 작성한다.
- ④ 공통된 코드를 사용한다.

클린 코드로 작성하려면 코드 중복을 최소화해야 합니다.

출제예상

2. 다음 중 소스 코드 품질 분석 도구에 대한 설명으로 틀린 것은?

- ① 소스 코드 품질 분석 도구에는 정적 분석 도구와 동적 분석 도구가 있다.
- ② 정적 분석 도구는 작성한 코드를 실행하지 않고 코딩 표준이나 코딩 스타일, 결함 등을 확인한다.
- ③ 동적 분석 도구는 작성한 소스 코드를 실행하여 코드에 존재하는 메모리 누수, 스레드 결함 등을 분석한다.
- ④ 코딩의 복잡도, 모델 의존성, 불일치성 등을 분석하는 도구는 동적 분석 도구이다.

코딩의 복잡도, 모델 의존성, 불일치성 등은 정적 분석 도구를 이용하여 분석할 수 있습니다.

▶ 정답 : 1. ① 2. ④

**1. 다음 중 애플리케이션 테스트에 대한 설명으로 틀린 것은?**

- ① 애플리케이션 테스트는 소프트웨어에 잠재되어 있는 결함을 찾아내는 일련의 행위이다.
- ② 테스트는 고객의 요구사항을 만족했는지 Verification 해야 한다.
- ③ 테스트는 오류 검출뿐만 아니라 새로운 오류의 유입도 방지할 수 있다.
- ④ 테스트를 효과적으로 실행하면 최소한의 시간과 노력으로 많은 결함을 찾을 수 있다.

2. 개발된 소프트웨어가 사용자의 요구사항 및 기대 수준 등을 만족하는지 점검하기 위해 애플리케이션 테스트를 진행하려고 할 때, 이에 대한 설명으로 틀린 것은?

- ① 완벽한 테스트는 불가능하다.
- ② 테스트는 정황(Context)에 의존한다.
- ③ 파레토 법칙을 적용할 수 있다.
- ④ 테스트와 위험은 정비례한다.

3. 소프트웨어 테스트에서 오류의 80%는 전체 모듈의 20% 내에서 발견된다는 법칙은?

- ① Brooks의 법칙 ② Boehm의 법칙
- ③ Pareto의 법칙 ④ Jackson의 법칙

4. 다음 중 애플리케이션 테스트에 대한 설명으로 틀린 것은?

- ① 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 만들어 구현하고 있는지 확인하는 테스트는 명세 기반 테스트이다.
- ② 테스트의 이전 경험과 기술을 기반으로 수행하는 테스트는 경험 기반 테스트이다.
- ③ 소프트웨어 내부의 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트는 구조 기반 테스트이다.
- ④ 동등 분할, 경계 값 분석, 구분 기반, 결정 기반 등은 명세 기반 테스트이다.

5. 다음 중 워크스루(Walkthrough)와 인스펙션(Inspection)에 대한 설명으로 가장 옳지 않은 것은?

- ① 워크스루는 전문가들에 의해 개발자의 작업 내역이 검토된다.
- ② 워크스루는 제품 개발자가 주체가 된다.
- ③ 워크스루는 오류 발견과 발견된 오류의 문제 해결에 중점을 둔다.
- ④ 인스펙션은 워크스루를 발전시킨 형태이다.

6. 블랙박스 테스트를 수행할 경우 발견하기 어려운 오류는?

- ① 인터페이스 오류 ② 성능 오류
- ③ 부정확한 기능 ④ 논리 구조상의 오류

7. 다음 중 블랙박스 테스트에 대한 설명으로 옳지 않은 것은?

- ① 블랙박스 테스트는 기능 테스트라고도 한다.
- ② 각 기능이 완전히 작동되는 것을 입증하는 테스트이다.
- ③ 소프트웨어 인터페이스에서 실시되는 테스트이다.
- ④ 조건 검사, 루프 검사, 데이터 흐름 검사 등의 유형이 있다.

8. 다음 중 화이트박스 테스트에 대한 설명으로 옳지 않은 것은?

- ① 제품의 내부 요소들이 명세서에 따라 수행되고 충분히 실행되는가를 테스트한다.
- ② 모듈 안의 작동을 직접 관찰한다.
- ③ 프로그램 원시 코드의 논리적인 구조를 커버하도록 테스트 케이스를 설계한다.
- ④ 설계된 모든 기능들이 정상적으로 수행되는지 확인한다.

9. 모듈의 논리적 구조를 체계적으로 점검하는 구조 테스트로, 이 방식의 종류에는 기초 경로 검사, 조건 검사, 데이터 흐름 검사, 루프 검사 등이 있는 것은?

- ① 화이트박스 테스트 ② 블랙박스 테스트
- ③ 레드박스 테스트 ④ 블루박스 테스트

10. 다음 중 개발 단계에 따른 소프트웨어 테스트 종류가 아닌 것은?

- ① 단위 테스트 ② 시스템 테스트
- ③ 화이트박스 테스트 ④ 통합 테스트

11. 사용자가 개발자의 장소, 그리고 개발자 앞에서 시행하는 테스트 기법으로, 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인하면서 테스트하는 기법은?

- ① 경계값 분석 ② 베타 테스트
- ③ 알파 테스트 ④ 동치 분할 테스트

12. 다음 중 단위 테스트에 대한 설명으로 틀린 것은?

- ① 소프트웨어의 최소 단위인 모듈에 초점을 맞춰 테스트한다.
- ② 소프트웨어 개발자가 코딩에 대해 테스트하는 것이다.
- ③ 주로 블랙박스 테스트를 실행한다.
- ④ 사용자의 요구사항을 기반으로 한 기능성 테스트를 최우선으로 수행한다.



13. 상향식 통합 테스트(Bottom-up Integration Test)의 과정이 옳게 나열된 것은?

- ㉠ 드라이버라는 제어 프로그램의 작성
㉡ 낮은 수준의 모듈들을 클러스터로 결합
㉢ 클러스터의 검사
㉣ 클러스터를 상위로 결합

- ① ㉠ → ㉡ → ㉢ → ㉣ ② ㉡ → ㉠ → ㉢ → ㉣
③ ㉡ → ㉢ → ㉠ → ㉣ ④ ㉠ → ㉡ → ㉣ → ㉢

14. 하향식 통합에 있어서 모듈 간의 통합 시험을 하기 위해 일시적으로 필요한 조건만을 가지고 임시로 제공되는 시험용 모듈을 무엇이라고 하는가?

- ① Driver ② Stub
③ Sub-Program ④ Dummy-Program

15. 하향식 통합에 대한 설명으로 가장 적합하지 않은 것은?

- ① 프로그램의 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트하는 기법이다.
② 마지막까지 독립된 프로그램 형태를 갖지 못한다.
③ 주요 제어 모듈의 종속 모듈들을 스티브로 대체한다.
④ 깊이 우선 방식이나 넓이 우선 방식에 의해 통합한다.

16. 다음 보기에 제시된 내용을 결합 관리 순서로 나열할 경우 옳바른 것은?

- ㉠ 에러 발견 ㉡ 에러 등록
㉢ 에러 분석 ㉣ 결합 확정
㉤ 결합 할당 ㉥ 결합 조치

- ① ㉠ → ㉡ → ㉢ → ㉣ → ㉤ → ㉥
② ㉠ → ㉡ → ㉤ → ㉣ → ㉢ → ㉥
③ ㉠ → ㉡ → ㉤ → ㉢ → ㉥ → ㉣
④ ㉠ → ㉡ → ㉢ → ㉤ → ㉥ → ㉣

17. 다음 중 소프트웨어의 테스트 결과가 옳바른지 판단하기 위해 사전에 정의된 참 값을 나타내는 것은 무엇인가?

- ① 테스트 하네스 ② 테스트 오라클
③ 테스트 적합성 ④ 테스트 케이스

18. 다음 중 테스트 시나리오와 테스트 케이스에 대한 설명으로 가장 옳지 않은 것은?

- ① 테스트 시나리오는 테스트 케이스의 동작 순서를 기술한 문서이다.

- ② 테스트 케이스는 테스트 절차를 명세한 문서이다.

- ③ 테스트 시나리오는 개발된 모듈 또는 프로그램 간의 연계가 정상적으로 동작하는지 테스트할 수 있도록 작성해야 한다.

- ④ 테스트 케이스는 테스트할 시스템이 수행해야 할 액션들로 구성된 일련의 단계이다.

19. 다음 중 테스트 자동화 도구의 장점에 대한 설명으로 가장 옳지 않은 것은?

- ① 테스트 자동화 도구는 테스트를 쉽고 효율적으로 수행할 수 있도록 도와준다.

- ② 테스트 자동화 도구는 테스트의 정확성을 유지하면서 테스트의 품질을 향상시킬 수 있도록 도와준다.

- ③ 테스트 자동화 도구의 사용 방법에 대한 교육 및 학습이 필요하다.

- ④ 테스트 결과에 대한 객관적인 평가 기준을 제공한다.

20. 다음 중 테스트 자동화에 대한 설명으로 가장 옳지 않은 것은?

- ① 테스트 자동화는 테스트 준비, 구현, 수행, 분석 등을 스크립트 형태로 구현한다.

- ② 테스트 자동화 도구를 이용하면 통계 작업과 그래프 등 다양한 표시 형태로 테스트 결과를 표시할 수 있다.

- ③ 테스트 엔지니어는 프로젝트를 완전히 이해한 후 테스트를 수행해야 하므로 프로젝트가 완료된 후 투입해야 한다.

- ④ 테스트 자동화 도구는 다중 플랫폼 호환성, 소프트웨어 구성, 기본 테스트 등 향상된 테스트 품질을 보장한다.

21. 다음 중 애플리케이션의 성능을 측정하는 지표들에 대한 설명으로 틀린 것은?

- ① 처리량(Throughput) : 일정 시간 내에 애플리케이션이 처리하는 일의 양

- ② 응답 시간(Response Time) : 애플리케이션에 요청을 전달한 시간부터 응답이 도착할 때까지 걸린 시간

- ③ 경과 시간(Turn Around Time) : 애플리케이션이 작업을 처리하기 시작한 시간부터 처리가 완료될 때까지 걸린 시간

- ④ 자원 사용률(Resource Usage) : 애플리케이션이 의뢰한 작업을 처리하는 동안의 CPU, 메모리, 네트워크 등의 사용량

**1. Section 049**

- Verification : 개발자의 입장에서 명세서에 맞게 만들어졌는지를 점검하는 것
- Validation : 사용자의 입장에서 고객의 요구사항에 맞게 구현되었는지 점검하는 것

2. Section 049

테스트와 위험은 반비례한다. 테스트를 많이 하면 할수록 미래에 발생할 위험을 줄일 수 있다.

3. Section 049

상위 20% 사람들이 전체 부의 80%를 가지고 있거나, 상위 20% 고객이 매출의 80%를 창출한다는 의미로 흔히 사용되는 '80:20 법칙'이 바로 파레토(Pareto)의 법칙이다.

4. Section 050

구분 기반, 결정 기반 등은 구조 기반 테스트이다.

5. Section 050

워크스루의 목적은 오류 해결이 아니라 오류의 조기 검출이다.

6. Section 051

기능과 관련된 기법은 블랙박스 테스트 기법이고, 논리적인 경로나 데이터 흐름과 관련된 기법은 화이트박스 테스트 기법이다.

7. Section 051

조건 검사, 루프 검사, 데이터 흐름 검사 등은 화이트박스 테스트이다.

8. Section 051

설계된 모든 기능들이 정상적으로 수행되는지 확인하는 것은 블랙박스 테스트이다.

9. Section 051

블랙박스 테스트(Black Box Test)

- 소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 테스트로서, 기능 테스트라고도 한다.
- 종류 : 동치 분할 검사, 경계값 분석, 원인-효과 그래프 검사, 오류 예측 검사, 비교 검사 등

10. Section 052

- 개발 단계에 따른 테스트 종류 : 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트

- 화이트박스 테스트는 모듈의 원시 코드를 오픈시킨 상태에서 원시 코드의 논리적인 모든 경로를 테스트하는 것으로, 테스트 기법에 따른 소프트웨어 테스트의 한 종류이다.

11. Section 052

- 경계값 분석 : 입력 자료에만 치중한 동치 분할 기법을 보완하기 위한 기법
- 베타 테스트 : 선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법
- 동치 분할 테스트 : 입력 자료에 초점을 맞춰 검사 사례를 만들고 검사하는 기법으로, 동등 분할 기법이라고도 함

12. Section 052

단위 테스트는 주로 구조적 테스트인 화이트박스 테스트를 실행한다.

14. Section 053

- 테스트 드라이버(Test Driver) : 테스트 대상의 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 도구
- 테스트 스텝(Test Stub) : 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로 일시적으로 필요한 조건만을 가지고 있는 시험용 모듈

15. Section 053

하향식 통합 기법은 처음부터 독립된 프로그램 구조를 갖춘다.

18. Section 055

테스트 절차를 명세한 문서는 테스트 시나리오이다.

19. Section 056

테스트 자동화 도구의 사용 방법에 대한 교육 및 학습이 필요한 것은 테스트 자동화 도구의 장점이 아니라 단점이다.

20. Section 056

테스트 엔지니어의 투입 시기가 늦어지면 프로젝트의 이해 부족으로 인해 불완전한 테스트를 초래할 수 있으므로 반드시 프로젝트 초기에 테스트 엔지니어의 투입 시기를 계획하여 적당한 시기에 투입해야 한다.

21. Section 058

경과 시간(Turn Around Time)

애플리케이션에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간

5 장

인터페이스 구현

060 모듈 간 공통 기능 및 데이터 인터페이스 확인 **C** 등급

061 모듈 연계를 위한 인터페이스 기능 식별 **A** 등급

062 모듈 간 인터페이스 데이터 표준 확인 **C** 등급

063 인터페이스 기능 구현 정의 **B** 등급

064 인터페이스 구현 **C** 등급

065 인터페이스 예외 처리 **B** 등급

066 인터페이스 보안 **B** 등급

067 연계 테스트 **B** 등급

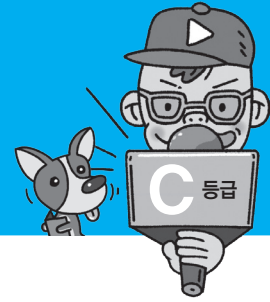
068 인터페이스 구현 검증 **A** 등급

069 인터페이스 오류 확인 및 처리 보고서 작성 **C** 등급



이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 인터페이스 설계서 2. EAI 3. ESB 4. 모듈 세부 설계서 5. JSON 6. XML 7. 인터페이스 보안 8. xUnit
9. NTAF 10. APM



전문의가의 조언

모듈 간 공통 기능 및 데이터 인터페이스를 확인하는 것은 쉽게 말해 모듈 간 연계를 위해 주고받아야 할 데이터가 무엇인지를 찾는 것을 말합니다. 이를 위해 인터페이스 설계서에서 정의한 모듈을 기반으로 공통적으로 제공되는 기능을 식별하고, 이를 바탕으로 각 데이터의 인터페이스를 확인합니다. 인터페이스 설계서를 서로 구분할 수 있도록 각각의 특징을 정리하고, 모듈 간 공통 기능과 각 데이터의 인터페이스를 확인하는 방법을 알아두세요.

※ **모듈 간 연계** : 내부 모듈과 외부 모듈 또는 내부 모듈 간 데이터 교환을 위해 관계를 설정하는 것

파라미터(Parameter, 매개변수)

파라미터는 함수를 통해 어떠한 작업을 수행 할 때 해당 함수 수행에 필요한 값을 전달하기 위해 사용되는 변수를 의미합니다.

인터페이스 설계서

인터페이스 설계서를 인터페이스 정의서라고도 합니다.

시스템 인터페이스 설계서에 대한 자세한 내용은 Section 032를 참조하세요.

1 모듈 간 공통 기능 및 데이터 인터페이스의 개요

- 공통 기능은 모듈의 기능 중에서 공통적으로 제공되는 기능을 의미한다.
- 데이터 인터페이스는 모듈 간 교환되는 데이터가 저장될 파라미터*를 의미한다.
- 모듈 간 공통 기능 및 데이터 인터페이스는 인터페이스 설계서에서 정의한 모듈의 기능을 기반으로 확인한다.
- 확인된 공통 기능 및 데이터 인터페이스는 모듈 간 연계가 필요한 인터페이스의 기능을 식별하는데 사용된다.
- 모듈 간 공통 기능 및 데이터 인터페이스 확인 순서
 - ① 인터페이스 설계서를 통해 모듈별 기능을 확인한다.
 - ② 외부 및 내부 모듈을 기반으로 공통적으로 제공되는 기능과 각 데이터의 인터페이스를 확인한다.

2 인터페이스 설계서*

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

- 인터페이스 설계서는 일반적인 형태의 설계서와 정적·동적 모형을 통한 설계서로 구분된다.

일반적인 인터페이스 설계서

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

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

정적·동적 모형을 통한 인터페이스 설계서

정적·동적 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 이용하여 만든 문서이다.

- 시스템을 구성하는 주요 구성 요소 간의 트랜잭션을 통해 해당 인터페이스가 시스템의 어느 부분에 속하고, 해당 인터페이스를 통해 상호 교환되는 트랜잭션의 종류를 확인할 수 있다.

3 인터페이스 설계서별 모듈 기능 확인

인터페이스 설계서에서 정의한 모듈을 기반으로 각 모듈의 기능을 확인한다.

- 시스템 인터페이스 목록에서 송신 및 전달 부분은 외부 모듈, 수신 부분은 내부 모듈에 해당된다.
- 시스템 인터페이스 설계서에서 데이터 송신 시스템 부분은 외부 모듈, 데이터 수신 시스템 부분은 내부 모듈에 해당된다.
- 상세 기능 인터페이스 명세서에서 오퍼레이션과 사전 조건은 외부 모듈, 사후 조건은 내부 모듈에 해당된다.
- 정적·동적 모형을 통한 인터페이스 설계에서 인터페이스 영역은 내부 모듈, 나머지 부분은 외부 모듈에 해당된다.

4 모듈 간 공통 기능 및 데이터 인터페이스 확인

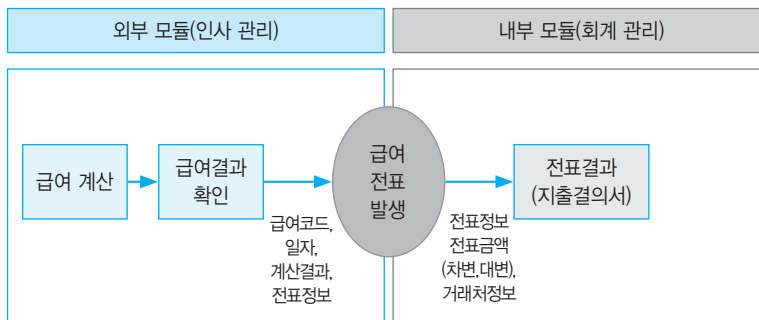
- 내·외부 모듈 기능을 통해 공통적으로 제공되는 기능을 확인한다.

예 내·외부 모듈 기능을 통한 공통 기능

외부 모듈(인사)	내부 모듈(회계)	공통 기능
인사 발령	전표 발생	전표 발생
전표 발생	지출 결의서 확인	
급여 계산	지출 처리	
급여 결과 확인		

- 내·외부 모듈 기능과 공통 기능을 기반으로 필요한 데이터 인터페이스 항목을 확인한다.

예 내·외부 모듈과 공통 기능을 통해 확인된 인터페이스* 및 데이터 인터페이스*



전문가의 조언

인터페이스 설계서에서 확인된 내·외부 모듈의 기능들을 기반으로 모듈 간 공통 기능 및 데이터 인터페이스를 확인합니다.



전문가의 조언

왼쪽 그림에서 인터페이스는 '급여전표발생'이고, 데이터 인터페이스는 '급여코드, 일자, 계산결과, 전표정보, 전표금액, 거래처정보'입니다.



기출문제 따라잡기

Section 060

출제예상

1. 다음 중 인터페이스 설계서에 대한 내용으로 가장 옳지 않은 것은?

- ① 인터페이스 설계서는 각 시스템의 교환 데이터 및 업무, 송·수신 주체 등이 정의되어 있다.
- ② 인터페이스 설계서를 통하여 인터페이스와 통신하는 외부 및 내부 모듈의 기능을 확인할 수 있다.
- ③ 인터페이스 설계서는 일반적인 내용이 포함된 인터페이스 설계서와 다양한 다이어그램 및 데이터 포맷을 포함한 형태의 인터페이스 설계서가 있다.
- ④ 인터페이스 설계서는 하나의 독립적인 기능을 수행하는 모듈의 구성 요소와 세부적인 동작 등이 정의되어 있다.

모듈에 대한 내용을 정의한 설계서는 모듈 세부 설계서입니다.

출제예상

2. 다음이 설명하는 것은?

각 시스템의 구성 요소를 표현한 다이어그램을 통해 만든 설계서로, 시스템을 구성하는 주요 구성 요소 간 트랜잭션을 보여 주고 이를 통해 시스템에서 인터페이스는 어디에 속하고 어떤 트랜잭션이 인터페이스를 통해 상호 교환되는지 확인할 수 있다.

- ① 정적·동적 모형을 통한 인터페이스 설계서
- ② 시스템 인터페이스 설계서
- ③ 데이터 정의를 통한 인터페이스 설계서
- ④ 상세 기능별 인터페이스 명세서

'정적·동적 모형 = 다이어그램'이라는 것 잊지마세요.

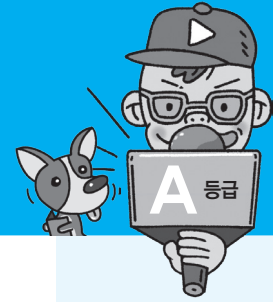
출제예상

3. 다음 중 내부, 외부 모듈 간 공통 기능 및 데이터 인터페이스 확인에 대한 설명으로 옳지 않은 것은?

- ① 인터페이스 설계서에서 정의한 내·외부 모듈을 기반으로 각 모듈의 기능을 확인한다.
- ② 상세 기능 인터페이스 명세서의 오퍼레이션 및 사전 조건은 내부 모듈, 사후 조건은 외부 모듈에 포함된다.
- ③ 인터페이스 목록의 송신 및 전달 영역까지는 외부 모듈, 수신 측 영역은 내부 모듈에 포함된다.
- ④ 인터페이스 설계서의 데이터 수신 시스템 부분은 내부 모듈, 데이터 송신 시스템 부분은 외부 모듈에 포함된다.

상세 기능 인터페이스 명세서의 오퍼레이션 및 사전 조건은 외부 모듈, 사후 조건은 내부 모듈에 포함됩니다.

▶ 정답 : 1. ④ 2. ① 3. ②

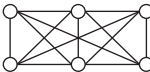
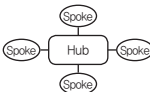
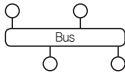
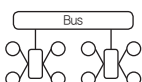


1 모듈 연계의 개요

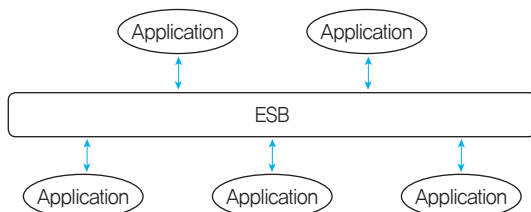
모듈 연계는 내부 모듈과 외부 모듈 또는 내부 모듈 간 데이터의 교환을 위해 관계를 설정하는 것으로, 대표적인 모듈 연계 방법에는 EAI와 ESB 방식이 있다.

EAI(Enterprise Application Integration)

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

유형	기능
Point-to-Point	<ul style="list-style-type: none"> • 가장 기본적인 애플리케이션 통합 방식으로, 애플리케이션을 1:1로 연결한다. • 변경 및 재사용이 어렵다. 
Hub & Spoke	<ul style="list-style-type: none"> • 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식이다. • 확장 및 유지 보수가 용이하다. • 허브 장애 발생 시 시스템 전체에 영향을 미친다. 
Message Bus (ESB 방식)	<ul style="list-style-type: none"> • 애플리케이션 사이에 미들웨어*를 두어 처리하는 방식이다. • 확장성이 뛰어나며 대용량 처리가 가능하다. 
Hybrid	<ul style="list-style-type: none"> • 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. 

ESB(Enterprise Service Bus)



- ESB는 애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스를 제공하는 솔루션이다.



전문가의 조언

모듈 연계 방법의 유형별 기능을 구분할 수 있도록 정리하고, 개발하고자 하는 응용 소프트웨어와 관련된 모듈 간 연계가 필요한 인터페이스의 기능을 식별하는 방법을 알아두세요.

미들웨어(Middle Ware)

미들웨어는 운영체제와 해당 운영 체제에 의해 실행되는 응용 프로그램 사이에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공하는 소프트웨어를 말합니다.

결합도(Coupling)

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

모듈 간 공통 기능 및 데이터 인터페이스 확인에 대한 자세한 내용은 Section 060을 참조하세요.

- ESB는 애플리케이션 통합 측면에서 EAI와 유사하지만 애플리케이션 보다는 서비스 중심의 통합을 지향한다.
- ESB는 특정 서비스에 국한되지 않고 범용적으로 사용하기 위하여 애플리케이션과의 결합도(Coupling)*를 약하게(Loosely) 유지한다.
- 관리 및 보안 유지가 쉽고, 높은 수준의 품질 지원이 가능하다.

2 모듈 간 연계 기능 식별

- 모듈 간 공통 기능 및 데이터 인터페이스*를 기반으로 모듈과 연계된 기능을 시나리오 형태로 구체화하여 식별한다.
- 식별된 연계 기능은 인터페이스 기능을 식별하는데 사용된다.

예 모듈 간 연계 기능 식별

구분	주요 기능	시나리오
외부 모듈	급여 계산	• 사전 조건 : 급여일자가 확정되어야 한다. • 기능 동작 시나리오 : 급여가 급여 정보에 따라 계산한다. • 사후 조건 : 전표 발생용 기본 정보가 생성된다.
	급여 결과 확인	• 사전 조건 : 급여가 계산된다. • 기능 동작 시나리오 : 개인별, 조직별, 직급별 급여 명세서가 조회된다. • 사후 조건 : 회계 전표 보고서 기준으로 전표가 발생된다.
내부 모듈	급여 전표 발생	• 사전 조건 : 회계 전표 발생에 필요한 정보를 계산한다. • 기능 동작 시나리오 : 급여 결과를 회계 정보에 맞게 변환하여 전표를 작성한다. • 사후 조건 : 입력값과 결과값의 정합성이 맞는지 체크한다.

3 모듈 간 인터페이스 기능 식별

- 식별된 모듈 간 관련 기능을 검토하여 인터페이스 동작에 필요한 기능을 식별한다.
- 인터페이스 동작은 대부분 외부 모듈의 결과 또는 요청에 의해 수행되므로 외부 및 인터페이스 모듈 간 동작하는 기능을 통해 인터페이스 기능을 식별한다.
- 내부 모듈의 동작은 외부 모듈에서 호출된 인터페이스에 의해 수행되고 결과를 나타내는 것이므로 해당 업무에 대한 시나리오를 통해 내부 모듈과 관련된 인터페이스 기능을 식별한다.
- 식별된 인터페이스 기능 중에서 실제로 필요한 인터페이스 기능을 최종적으로 선별한다.
- 식별된 인터페이스 기능은 인터페이스 기능 구현을 정의하는데 사용된다.



기출문제 따라잡기

Section 061

출제예상

1. 내·외부 모듈 연계 방법 중 ESB(Enterprise Service Bus)에 대한 설명으로 가장 옳지 않은 것은?

- ① ESB는 애플리케이션과의 결합도(Coupling)를 약하게(Loosely) 유지한다.
- ② ESB는 크게 Point-to-Point형, Hub & Spoke 방식, Hybrid 형태의 구성으로 분류될 수 있다.
- ③ 높은 수준의 품질 지원이 가능하다.
- ④ 관리 및 보안이 쉽다.

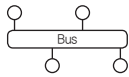
Point-to-Point, Hub & Spoke, Hybrid는 EAI의 구축 유형입니다.

출제예상

2. EAI 구축 유형 중 Message Bus에 대한 설명으로 옳은 것은?

- ① 애플리케이션 사이에 미들웨어(버스)를 두어 처리하는 방식이다.
- ② 단일 접점이 허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식이다.
- ③ 각 애플리케이션들이 트리 형태로 연결된 구조이다.
- ④ 중간에 미들웨어를 두지 않고 각 애플리케이션 간 Point to Point 형태로 연결하는 방식이다.

보기 중 아래의 Message Bus의 개념도와 일치하는 내용을 찾아보세요.



출제예상

3. 다음 중 내·외부 모듈 간 연계를 위한 인터페이스 기능의 식별에 대한 설명으로 옳지 않은 것은?

- ① 인터페이스가 포함된 내·외부 모듈을 통해 연계된 기능을 시나리오 형태로 구체화하여 기능을 식별한다.
- ② 외부 모듈과 인터페이스 모듈 간의 동작 기능에 기반하여 인터페이스의 기능을 식별한다.
- ③ 내부 모듈 관련 인터페이스 기능은 외부 모듈 관련 인터페이스 기능을 식별하는 방법과 상반된 방법으로 식별한다.
- ④ 외부, 공통 및 내부 모듈 기능 분석을 통한 인터페이스 기능을 통해 필요한 인터페이스 기능을 종합적으로 식별한다.

내부 모듈의 동작은 외부 모듈에서 호출된 인터페이스에 의해 수행됩니다. 그럼 내·외부 모듈 관련 인터페이스 기능을 식별하는 방법이 비슷할까요? 다를까요?

▶ 정답 : 1. ② 2. ① 3. ③



전문가의 조언

모듈 간 인터페이스 데이터 표준 확인은 쉽게 말해 모듈 간 원활한 데이터 교환을 위해 인터페이스에 사용되는 데이터 요소의 명칭, 정의, 규칙 등에 대한 원칙을 만드는 것입니다. 인터페이스 데이터 표준의 개념과 확인 방법에 대해 정리하세요.

데이터 인터페이스에 대한 자세한 내용은 Section 060을, 인터페이스 기능에 대한 자세한 내용은 Section 061을 참조하세요.

1 인터페이스 데이터 표준의 개요

인터페이스 데이터 표준은 모듈 간 인터페이스에 사용되는 데이터의 형식을 표준화하는 것이다.

- 인터페이스 데이터 표준은 기존의 데이터 중에서 공통 영역을 추출하거나 어느 한 쪽의 데이터를 변환하여 정의한다.
- 확인된 인터페이스 데이터 표준은 인터페이스 기능 구현을 정의하는데 사용된다.
- 모듈 간 인터페이스 데이터 표준 확인 순서
 - ① 데이터 인터페이스*를 통해 인터페이스 데이터 표준을 확인한다.
 - ② 인터페이스 기능*을 통해 인터페이스 표준을 확인한다.
 - ③ 데이터 인터페이스와 인터페이스 기능을 통해 확인된 인터페이스 표준을 검토하여 최종적인 인터페이스 데이터 표준을 확인한다.

2 데이터 인터페이스 확인

- 데이터 표준을 위해 식별된 데이터 인터페이스에서 입·출력값의 의미와 데이터의 특성 등을 구체적으로 확인한다.
- 확인된 데이터 인터페이스의 각 항목을 통해 데이터 표준을 확인한다.

예 데이터 인터페이스의 데이터 표준

구분	파라미터	의미	데이터 표준
입력값	급여 코드	급여 작업의 단위를 나타내는 Key 값	<ul style="list-style-type: none"> • 급여 지급 연월의 숫자 6자리 • 정규직 R, 계약직 T • 3자리 숫자로, 급여코드 증가 시 숫자 증가
	급여 계산 결과	<ul style="list-style-type: none"> • 각 직원별 급여 계산 결과 • 직원정보, 총액, 공제액 	항목 : 사번, 소속, 직급, 근무일수, 급여, 상여, 비과세급여, 총지급액, 국민연금, 건강보험, 고용보험, 소득세, 주민세, 실지급액 등
출력 정보	전표 정보	<ul style="list-style-type: none"> • 발생 시기 • 전표의 매입/매출 • 세금 신고 여부 	<ul style="list-style-type: none"> • 발생 시기 : YYYYMMDD 형식의 8자리 • 전표 구분 : 매입 AP, 매출 AR
	차·대변	<ul style="list-style-type: none"> • 각 계정별 마스터 정보 발생 • 디테일 정보 발생 	<ul style="list-style-type: none"> • 각 계정별 급여, 상여 세금, 사내 공제 항목 등의 마스터 정보 발생 • 각 부서별 급여합계 등의 디테일 정보 발생

3 인터페이스 기능 확인

- 데이터 표준을 위해 식별된 인터페이스 기능을 기반으로 인터페이스 기능 구현을 위해 필요한 데이터 항목을 확인한다.
- 확인된 데이터 항목과 데이터 인터페이스에서 확인된 데이터 표준에서 수정 · 추가 · 삭제될 항목이 있는지 확인한다.

예 인터페이스 기능 구현을 위해 필요한 데이터 항목

인터페이스 기능	필요 데이터 항목	데이터 인터페이스 항목
외부 모듈로부터 전표 발생을 위한 필수 입력값을 수신한다.	<ul style="list-style-type: none"> • 전표 일자, 계정, 금액 • 발의 부서, 귀속 부서 • 거래처, 지급 여부 	<ul style="list-style-type: none"> • 급여 코드 • 급여 일자 • 급여 계산 결과 전표 • 계정 정보
회계 시스템에서 발생한 전표 발생 작업의 결과를 수신한다.	<ul style="list-style-type: none"> • 전표 발생 여부 • 차, 대변 검증 여부 • 부서별 금액, 계정별 금액 	<ul style="list-style-type: none"> • 전표 금액 • 차, 대변 금액 • 차, 대변 검증 결과

4 인터페이스 데이터 표준 확인

- 데이터 인터페이스에서 확인된 데이터 표준과 인터페이스 기능을 통해 확인된 데이터 항목들을 검토하여 최종적으로 데이터 표준을 확인한다.
- 확인된 데이터 표준은 항목별로 데이터 인터페이스와 인터페이스 기능 중 출처를 구분하여 기록한다.

예 확인된 인터페이스 데이터 표준

구분	파라미터	데이터 표준	출처
입력값	급여 코드	<ul style="list-style-type: none"> • 급여 지급 연월을 숫자 6자리 • 정규직 R, 계약직 T • 3자리 숫자로 급여코드 증가 시 숫자 증가 	데이터 인터페이스
	급여 계산 결과	항목 : 사번, 소속, 직급, 근무일수, 급여, 상여, 비과세급여, 총지급액, 국민연금, 건강보험, 고용보험, 소득세, 주민세, 실지급여액 등	데이터 인터페이스
출력정보	전표 정보	<ul style="list-style-type: none"> • 발생 시기 : YYYYMMDD 형식의 8자리 • 전표 구분 : 매입 AP, 매출 AR 	데이터 인터페이스
	차 · 대변 검증	부서별, 계정별 항목 수와 총금액 검증	인터페이스 기능



기출문제 따라잡기

Section 062

출제예상

1. 다음 설명이 의미하는 것은?

인터페이스를 위해 인터페이스가 되어야 할 범위의 데이터들의 형식과 표준을 정의하는 것으로, 기존에 있던 데이터 중 공통의 영역을 추출하여 정의하는 경우도 있고 인터페이스를 위해 한쪽의 데이터를 변환하는 경우도 있다.

- ① 인터페이스 요구사항 정의서
- ② 프로그램 명세서
- ③ 인터페이스 데이터 표준
- ④ 인터페이스 목록

인터페이스가 되어야 할 범위의 데이터들의 형식과 표준을 정의하는 것은 무엇 일까요?

출제예상

2. 다음 중 내·외부 모듈 간 인터페이스 데이터 표준 확인에 대한 설명으로 옳지 않은 것은?

- ① 외부 및 내부 모듈 간 데이터를 교환하고 상호 호환이 되게 하기 위해서 인터페이스 데이터 표준을 정의하고 이를 관리하여야 한다.
- ② 인터페이스 데이터의 형태가 동일한 경우에만 인터페이스 데이터 표준을 정의할 수 있다.
- ③ 식별된 데이터 인터페이스를 통해 인터페이스 데이터 표준을 확인한다.
- ④ 식별된 인터페이스 기능을 통해 인터페이스 표준을 확인한다.

인터페이스를 위해 한쪽의 데이터를 변환하는 경우도 있습니다. 즉 인터페이스 데이터 표준은 인터페이스 데이터 형태의 동일 여부와 관계없이 정의할 수 있습니다.

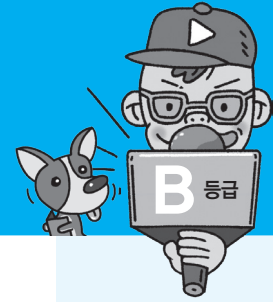
출제예상

3. 다음 중 내·외부 모듈 간 인터페이스 데이터 표준을 확인하는데 사용되는 정보로만 짝지어진 것은?

- ① 인터페이스 목록, 인터페이스 명세
- ② 인터페이스 명세, 데이터 인터페이스서
- ③ 인터페이스 기능, 인터페이스 목록
- ④ 데이터 인터페이스, 인터페이스 기능

인터페이스 데이터 표준하면 '데이터 인터페이스와 인터페이스 기능'이라는 것을 기억해 두세요.

▶ 정답: 1. ③ 2. ② 3. ④



1 인터페이스 기능 구현의 정의에 대한 개요

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

- 인터페이스 기능 구현 정의 순서
 - ① 컴포넌트 명세서를 확인한다.
 - ② 인터페이스 명세서를 확인한다.
 - ③ 일관된 인터페이스 기능 구현을 정의한다.
 - ④ 정의된 인터페이스 기능 구현을 정형화한다.

2 모듈 세부 설계서

모듈 세부 설계서는 모듈의 구성 요소와 세부적인 동작 등을 정의한 설계서이다.

- 대표적인 모듈 세부 설계서에는 컴포넌트 명세서와 인터페이스 명세서가 있다.

컴포넌트 명세서

컴포넌트 명세서는 컴포넌트의 개요 및 내부 클래스의 동작, 인터페이스를 통해 외부와 통신하는 명세 등을 정의한 것이다.

예 컴포넌트 명세서

컴포넌트ID	HR-COM-001	컴포넌트명	인사발령
컴포넌트 개요	각 기업의 인사발령을 수행하고 관계 기업과 필수정보를 공유하는 컴포넌트이다.		
내부 클래스			
ID	클래스명	설명	
HR-CLASS-01	발령이력 관리	개인 이력관리에 발령 형태에 따른 발령이력을 등록한다.	
HR-CLASS-02	인터페이스 호출	관계 기업과 인터페이스를 통해 발령사항을 공유한다.	
인터페이스 클래스			
ID	인터페이스명	오퍼레이션명	구분 [*]
IF-HR-01	인사정보 전송 인터페이스	대상 선정	전달 대상
		정보 전송	전달 행위
		결과 확인	전달 결과

전문가의 조언

인터페이스 기능 구현을 정의하는 것은 실제로 인터페이스 기능을 구현하기 위해 어떤 기능을 어떻게 구현할지를 기능별로 정의하는 것입니다. 인터페이스 기능 구현을 정의하는데 필요한 모듈 세부 설계서를 먼저 정리한 다음 인터페이스 기능 구현을 정의하는 방법에 대해 숙지하세요.

구분

구분은 인터페이스의 오퍼레이션에 대한 서비스를 요청 또는 제공하는 오퍼레이션을 구분하여 입력합니다.

사전조건 / 사후조건

오퍼레이션이 작동하기 전/후에 항상 참이어야 하는 조건을 입력합니다.

파라미터

오퍼레이션이 갖는 데이터를 입력합니다.

반환값

오퍼레이션 작동 후 반환값이 있는 경우 해당 값을 입력합니다.



전문가의 조언

인터페이스 기능 구현을 정의하기 위해서는 모듈 세부 설계서에 정의되어 있는 인터페이스의 주요 기능 및 세부 기능들을 확인하여 내·외부 모듈 간 식별된 인터페이스 기능 및 데이터 표준과 일치하는지 확인해야 합니다.

인터페이스의 기능에 대한 자세한 내용은 Section 061을, 인터페이스 데이터 표준에 대한 자세한 내용은 Section 062를 참조하세요.

인터페이스 기능의 일관성

인터페이스 기능이 일관성을 갖기 위해서는 인터페이스의 기능, 인터페이스 데이터 표준, 모듈 세부 설계서의 정의와 일치해야 합니다.

유스케이스 다이어그램 (Use Case Diagram)

사용자의 요구를 분석하는 것으로 기능 모델링 작업에 사용되는 다이어그램입니다.

인터페이스 명세서

인터페이스 명세서는 컴포넌트 명세서의 항목 중 인터페이스 클래스의 세부 조건 및 기능 등을 정의한 것이다.

예 인터페이스 명세서

인터페이스ID	IF-HR-01	인터페이스명	인사정보 전송 인터페이스
오퍼레이션명	인터페이스 대상 선정		
오퍼레이션 개요	관계 기업과 인터페이스 할 대상을 선택한다.		
사전조건*	과장 이상 정규직만 선택한다.		
사후조건*	데이터 전송 이후 상대 시스템의 결과값을 업데이트 한다.		
파라미터*	발령구분(입사, 이동, 승진), 발령정보(발령일, 소속, 직급)		
반환값*	SUCCESS/FAIL		

3 모듈 세부 설계서 확인

- 각 모듈의 컴포넌트 명세서와 인터페이스 명세서를 기반으로 인터페이스에 필요한 기능을 확인한다.
- 컴포넌트 명세서의 컴포넌트의 개요, 내부 클래스의 클래스명과 설명 등을 통해 컴포넌트가 가지고 있는 주요 기능을 확인한다.
- 컴포넌트 명세서의 인터페이스 클래스를 통해 인터페이스에 필요한 주요 기능을 확인한다.
- 인터페이스 명세서를 통해 컴포넌트 명세서의 인터페이스 클래스에 명시된 인터페이스의 세부 조건 및 기능을 확인한다.

4 인터페이스 기능 구현 정의

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



기출문제 따라잡기

Section 063

출제예상

1. 다음 중 인터페이스 기능 구현 정의에 대한 설명으로 가장 옳지 않은 것은?

- ① 다각적으로 분석된 모듈 설계 명세서, 테이블 정의서, 코드 정의서를 통하여 인터페이스의 기능 구현을 정의한다.
- ② 컴포넌트 명세서에 정의된 기능을 확인하여 컴포넌트가 가지고 있는 주요 기능을 확인한다.
- ③ 인터페이스 명세서를 통해서 컴포넌트 명세서에서 명시된 인터페이스의 세부 기능을 확인한다.
- ④ 정의된 인터페이스 기능 구현은 표준화되고 사람들이 보기 쉽게 정형화한다.

인터페이스의 기능 구현은 인터페이스의 기능, 데이터 표준, 모듈 설계 명세서를 통해 정의합니다.

출제예상

2. 다음 중 모듈 세부 설계서에 대한 설명으로 가장 옳지 않은 것은?

- ① 모듈 세부 설계서는 하나의 독립적인 기능을 수행하는 모듈의 구성 요소와 세부적인 동작을 정의한 것이다.
- ② 대표적인 모듈 세부 설계서에는 컴포넌트 명세서와 인터페이스 명세서가 있다.
- ③ 컴포넌트 명세서는 컴포넌트의 개요 및 내부 클래스의 동작 등을 정의한 것이다.
- ④ 인터페이스 명세서는 연계 업무와 연계에 참여하는 송·수신 시스템의 정보 등을 정의한 것이다.

인터페이스 명세서는 컴포넌트 명세서 중 인터페이스 클래스의 세부 조건 및 기능 등을 정의한 것입니다.

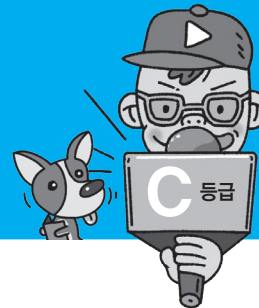
출제예상

3. 다음 중 컴포넌트 명세서의 항목이 아닌 것은?

- ① 컴포넌트 ID
- ② 사전/사후 조건
- ③ 내부 클래스
- ④ 인터페이스 클래스

사전/사후 조건은 인터페이스 명세서의 항목입니다.

▶ 정답 : 1. ① 2. ④ 3. ②



전문가의 조언

인터페이스 구현은 송신 측에서 전송한 데이터를 수신 측에서 받아 처리할 수 있도록 지원하는 작업입니다. 인터페이스를 구현하는 방법에는 여러 가지가 있지만 가장 많이 사용되는 데이터 통신을 이용한 인터페이스 구현 방법과 인터페이스 엔티티를 이용한 인터페이스 구현 방법에 대해 공부하도록 하겠습니다. 두 인터페이스 구현 방법을 구분할 수 있도록 정리하세요.

인터페이스 기능 구현에 대한 자세한 내용은 Section 063을 참조하세요.

파싱(Parsing)

파싱은 주어진 문장이 정의된 문법 구조에 따라 완전한 문장으로 사용될 수 있는가를 확인하는 작업을 말합니다.

인터페이스 객체를 생성할 데이터

인터페이스 객체를 생성할 데이터는 일반적으로 데이터베이스에 있는 정보를 SQL을 통해 선택한 후 JSON으로 생성합니다.

AJAX(Asynchronous JavaScript and XML)

AJAX는 웹 페이지에서 자바 스크립트(JavaScript) 등을 이용하여 XML로 데이터를 교환 및 제어함으로써 사용자가 웹 페이지와 자유롭게 상호 작용할 수 있도록 하는 기술을 의미합니다.

1 인터페이스 구현

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

- 정의된 인터페이스 기능 구현*을 기반으로 구현 방법 및 범위 등을 고려하여 인터페이스 구현 방법을 분석한다.
- 분석된 인터페이스 구현 정의를 기반으로 인터페이스를 구현한다.
- 인터페이스를 구현하는 대표적인 방법에는 데이터 통신을 이용한 방법과 인터페이스 엔티티를 이용한 방법이 있다.

2 데이터 통신을 이용한 인터페이스 구현

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

- 주로 JSON이나 XML 형식의 데이터 포맷을 사용하여 인터페이스를 구현한다.

예 JSON을 이용한 인터페이스 구현 순서

- 1 인터페이스의 객체 생성 및 전송을 위해 인터페이스 객체를 생성할 데이터*를 각 시스템 및 환경에 맞게 선택한다.
- 2 선택한 데이터를 JSON을 이용하여 인터페이스 객체를 생성한다.
- 3 송신 측에서 JSON으로 생성한 인터페이스 객체를 AJAX* 기술 등을 이용하여 수신 측으로 보낸다.
- 4 수신 측에서 인터페이스 객체를 수신해 파싱한 후 처리한다.
- 5 수신 측에서 송신 측에 처리 결과를 보낸다.



JSON / XML

JSON(JavaScript Object Notation)

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

- 비동기 처리에 사용되는 AJAX에서 XML을 대체하여 사용되고 있습니다.

XML(eXtensible Markup Language)

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

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

※ SGML(Stand Generalized Markup Language): 텍스트, 이미지, 오디오 및 비디오 등을 포함하는 멀티미디어 전자문서들을 다른 기종의 시스템들과 정보의 손실 없이 효율적으로 전송, 저장 및 자동 처리하기 위한 언어

3 인터페이스 엔티티를 이용한 인터페이스 구현

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

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

예 인터페이스 테이블을 이용한 인터페이스 구현 순서

- ① 인터페이스 이벤트가 발생하면 인터페이스 테이블에 인터페이스 데이터를 기록한다(Write)*.
- ② 송신 측 인터페이스 테이블에서 정해진 주기*에 따라 인터페이스 데이터를 전송*한다.
- ③ 수신 측 인터페이스 테이블에 인터페이스 데이터가 입력되면 정해진 주기에 따라 인터페이스 데이터를 읽는다(Read).
- ④ 수신 측 인터페이스 테이블에서 인터페이스 데이터를 읽은 후 사전에 정의된 데이터 트랜잭션을 수행한다.

인터페이스 데이터 기록

인터페이스 데이터 기록(Write)은 데이터 무결성 유지 및 인터페이스 이력 관리 등을 위해 삽입(Insert)만 가능하고 수정(Update) 및 삭제(Delete)는 불가능합니다.

정해진 주기

일반적으로 정해진 주기는 즉시, 주기적, 특정기간 이후 등이 있습니다.

인터페이스 데이터 전송

인터페이스 데이터 전송을 위해서는 일반적으로 디비 커넥션(DB Connection)이 수신 측 인터페이스 테이블과 연결되어 있어야 하고 프로시저(Procedure)나 트리거(Trigger) 등을 통해 수신 인터페이스 테이블로 전송합니다.

※ **디비 커넥션(DB Connection)** : 수신 시스템의 웹 애플리케이션 서버(WAS)에서 송신 시스템 데이터베이스(DB)로 연결해 주는 시스템 연계 기술

※ **프로시저(Procedure)** : 프로그램에서 특정한 작업을 수행하는 코드 모음으로, 이름이 있음

※ **트리거(Trigger)** : 데이터베이스 시스템에서 데이터의 입력, 갱신, 삭제 등의 이벤트가 발생할 때마다 자동적으로 수행되는 사용자 정의 프로시저



기출문제 따라잡기

Section 064

출제예상

1. 다음 중 인터페이스 구현에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스 기능 구현을 기반으로 인터페이스를 어떻게 구현할지 분석한 후 이를 기반으로 인터페이스를 구현한다.
- ② 인터페이스를 구현하는 대표적인 방법으로는 데이터 통신을 이용한 방법과 인터페이스 엔티티를 이용한 방법으로 나눌 수 있다.
- ③ 데이터 통신을 이용한 인터페이스 구현 방법은 주로 HTML 형식의 데이터 포맷을 사용하여 인터페이스를 구현한다.
- ④ 인터페이스 엔티티를 이용한 인터페이스 구현 방법은 일반적으로 인터페이스 테이블을 엔티티로 활용한다.

데이터 통신을 이용한 인터페이스 구현 방법은 주로 JSON나 XML 형식의 데이터 포맷을 사용하여 인터페이스를 구현합니다.

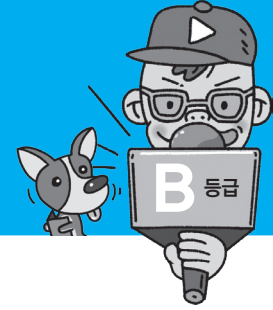
출제예상

2. 다음 중 인터페이스 엔티티를 이용한 인터페이스 구현 방법에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스가 필요한 시스템 사이에 별도의 인터페이스 엔티티를 두어서 상호 연계하는 방식이다.
- ② 일반적으로 인터페이스 테이블을 엔티티로 활용한다.
- ③ 1개의 인터페이스 테이블을 사용할 수도 있고 송신 및 수신 인터페이스 테이블을 각각 활용할 수도 있다.
- ④ 송신 테이블과 수신 테이블의 구조는 반드시 동일하게 설계해야 한다.

송·수신 인터페이스 테이블의 구조는 상황에 따라 서로 다르게 설계할 수 있습니다.

▶ 정답 : 1. ③ 2. ④



전문가의 조언

인터페이스를 구현하고 동작하다 보면 시스템 고장, 불안정한 네트워크, 용량 초과 데이터 등의 이유로 인해 인터페이스 기능 구현이 실패되는 경우가 있습니다. 이를 대비하여 인터페이스 예외 처리를 정의해 두면 인터페이스 기능 구현이 실패할 경우 예외 처리를 통해 문제를 해결할 수 있습니다. 인터페이스 예외 처리는 데이터 통신을 이용한 처리와 인터페이스 엔티티를 이용한 처리로 나뉩니다. 각 예외 처리를 구분할 수 있도록 정리하세요.

데이터 통신을 이용한 인터페이스 구현 방법과 JSON과 XML에 대한 자세한 내용은 Section 064를 참조하세요.

정합성

정합성은 모순 없는 정확한 특성을 의미하는 용어로, 데이터베이스에 저장된 데이터는 항상 정확해야 한다는 특성을 표현할 때 사용됩니다.

1 인터페이스 예외 처리의 개요

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

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

2 데이터 통신을 이용한 인터페이스 예외 처리

데이터 통신을 이용한 인터페이스 예외 처리 방법은 JSON*, XML* 등 인터페이스 객체를 이용해 구현한 인터페이스 동작이 실패할 경우를 대비한 것으로, 인터페이스 객체의 송·수신 시 발생할 수 있는 예외 케이스를 정의하고 각 예외 케이스마다 예외 처리 방법을 기술한다.

- 시스템 환경, 송·수신 데이터, 프로그램 자체 원인 등 다양한 원인으로 인해 예외 상황이 발생한다.

예 1 인터페이스 객체 송신 실패 시 예외 처리 방안

구분	예외 상황	예외 처리 방안
시스템 환경	네트워크 불안정	<ul style="list-style-type: none"> • POST 이후 오류 메시지를 확인한다. • 서버를 찾지 못하는 404 오류일 경우 네트워크 또는 서버 상태를 확인한다.
송신 데이터	송신 데이터 크기, 데이터 정합성* 체크 오류 발생	<ul style="list-style-type: none"> • 송신 데이터의 원활한 전송을 위해 사전에 데이터를 정제한다. • 데이터 송신 시 데이터 크기 및 정합성을 체크하는 기능을 추가하여 미리 예방한다.
프로그램 자체 원인	송신 데이터 생성 시 프로세스의 논리적 결함	<ul style="list-style-type: none"> • 논리적 결함을 수정한다. • 충분한 테스트를 통해 사전 예방한다. • 프로세스에 따라 예상되는 예외를 사용자에게 알람을 통해 알려 준다.

예 2 인터페이스 객체 수신 실패 시 예외 처리 방안

구분	예외 상황	예외 처리 방안
시스템 환경	네트워크 및 서버 불안정	입력 대기 큐에 요청을 적재한 후 순차적으로 처리하여 서버가 정상적으로 가동될 때 동작할 수 있도록 한다.
수신 데이터	특수문자 등으로 파싱 시 오류 발생	특수문자 입력 케이스를 미리 파악한 다음 파싱 시 오류가 발생하지 않는 문자로 우선 대체하고 이후에 다시 처리한다.

프로그램 자체 원인	수신 데이터 처리 시 프로그램의 논리적 결함	<ul style="list-style-type: none"> • 논리적 결함을 수정한다. • 충분한 테스트를 통해 사전 예방한다. • 프로세스에 따라 예상되는 예외를 사용자에게 알람을 통해 알려 준다. • 예외사항이 수신 되지 않도록 송신 측 프로그램을 수정한다.
---------------	--------------------------	--

3 인터페이스 엔티티를 이용한 인터페이스 예외 처리

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

예1 송신 인터페이스 테이블을 이용한 인터페이스 기능 실패 시 예외 처리 방안

프로세스	예외 상황	예외 처리 방안
인터페이스 데이터 생성	<ul style="list-style-type: none"> • 선택 SQL, 프로그램 오류 • 데이터 객체 생성 오류 	<ul style="list-style-type: none"> • 오류 발생 시 사용자에게 알람을 통해 알려준다. • 예외 케이스의 재발 방지를 위해 프로그램을 개선한다.
인터페이스 테이블에 입력	<ul style="list-style-type: none"> • 입력 SQL 오류 • 데이터 정합성 오류 	<ul style="list-style-type: none"> • 입력 실패 결과와 원인을 인터페이스 테이블에 기록한다. • 입력 실패 결과를 사용자에게 알람을 통해 알려준다. • 예외 케이스의 재발 방지를 위해 프로그램을 개선한다.
인터페이스 데이터 전송	DB Connection* 오류	<ul style="list-style-type: none"> • 통신 결과를 통해 인터페이스 실패 결과와 원인을 인터페이스 테이블에 기록한다. • 인터페이스 실패 결과와 원인을 사용자와 관리자에게 이메일 등으로 전송한다.
	데이터 전송 주체의 논리적 오류	<ul style="list-style-type: none"> • 인터페이스 실패 결과와 원인을 인터페이스 테이블에 기록한다. • 인터페이스 실패 결과를 사용자와 관리자에게 이메일 등으로 전송한다. • 예외 케이스의 재발 방지를 위해 프로그램을 개선한다.

예2 수신 인터페이스 테이블을 이용한 인터페이스 기능 실패 시 예외 처리 방안

프로세스	예외 상황	예외 처리 방안
인터페이스 데이터 읽기	데이터 선택 시 오류	<ul style="list-style-type: none"> • 수신 측 사용자에게 알람을 통해 예외사항을 알려준다. • 인터페이스 테이블에 예외사항을 기록한다. • 재발되지 않도록 프로그램을 개선한다.
데이터 트랜잭션	데이터 트랜잭션 시 프로그램의 논리상 오류	<ul style="list-style-type: none"> • 사용자에게 알람을 통해 예외사항을 알려준다. • 인터페이스 테이블에 예외사항을 기록한다. • 재발되지 않도록 프로그램을 개선한다.
처리 결과 응답	DB Connection 오류	<ul style="list-style-type: none"> • 인터페이스 테이블에 예외사항을 기록한다. • 송 · 수신자에게 이메일 등으로 예외사항을 알려준다.



전문가의 조언

인터페이스 엔티티를 이용한 인터페이스 구현 방법에 대한 자세한 내용은 Section 064를 참조하세요.



전문가의 조언

일반적으로 인터페이스 테이블을 엔티티로 활용하므로, 송 · 수신 테이블을 예로 사용하였으며, 인터페이스 테이블을 통한 인터페이스 트랜잭션 실패 시에는 인터페이스 프로세스 구간별로 예외 처리 방안을 정의합니다.

DB Connection

DB Connection은 수신 시스템의 웹 애플리케이션 서버(WAS)에서 송신 시스템 데이터베이스(DB)로 연결해주는 시스템 연계 기술을 의미합니다.



기출문제 따라잡기

Section 065

출제예상

1. 다음 중 인터페이스 객체 송·수신 시 예외가 발생하는 경우에 해당하지 않는 것은?

- ① 시스템 환경
- ② 송신 데이터
- ③ 프로그램 자체 원인
- ④ 인터페이스 테이블

인터페이스 객체 송·수신 시 예외하면 '시스템 환경, 송·수신 데이터, 프로그램' 자체 원인이란 것 잊지마세요.

출제예상

2. 다음 중 인터페이스 객체 송신 시 발생하는 예외 상황 및 처리 방법에 대한 설명으로 가장 옳지 않은 것은?

- ① 시스템의 네트워크가 불안정할 경우 입력 대기 큐를 통해서 요청을 쌓아 놓고 순차적으로 처리하여 서버 정상 가동 시 동작할 수 있도록 한다.
- ② 송신 데이터 생성 시 프로세스의 논리적 결함이 발생할 경우 논리적 결함을 수정하고 충분한 테스트로 사전 예방한다.
- ③ 송신 시스템에서 404 오류가 발생할 경우 네트워크 또는 서버의 상태를 확인한다.
- ④ 송신 데이터 크기 및 데이터 정합성에 오류가 발생할 경우 적절한 송신 데이터 및 데이터 형태로 전송되도록 사전에 데이터를 정제한다.

①번의 내용은 인터페이스 객체 수신 시 발생하는 예외 상황 및 처리 방법에 해당합니다.

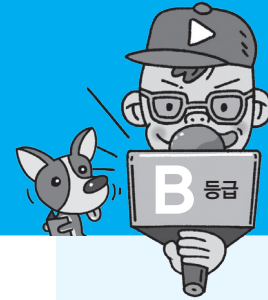
출제예상

3. 다음 중 송신 인터페이스 엔티티를 이용한 인터페이스 기능 실패 시 예외 처리 방법에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스 데이터 생성 시 선택 SQL 또는 프로그램 오류가 발생할 경우 알람을 통해 사용자에게 오류 발생을 알리고 예외 케이스가 재발하지 않도록 프로그램을 개선한다.
- ② 인터페이스 테이블에서 입력 SQL의 오류가 발생할 경우 입력 실패 결과와 원인을 인터페이스 테이블에 기록한다.
- ③ 인터페이스 데이터 전송 시 데이터 전송 주체의 논리적 오류가 발생할 경우 사용자와 관리자에게 직접 전화하여 알려준다.
- ④ 인터페이스 데이터 전송 시 DB Connection 오류가 발생할 경우 통신 결과를 읽어서 실패 결과와 원인을 인터페이스 테이블에 기록한다.

인터페이스 오류 결과나 원인은 오류 메시지나 메일 등을 통해 사용자와 관리자에게 알려줍니다.

▶ 정답 : 1. ④ 2. ① 3. ③



1 인터페이스 보안의 개요

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

2 인터페이스 보안 취약점 분석

- 인터페이스 기능이 수행되는 각 구간들의 구현 현황을 확인하고 각 구간에 어떤 보안 취약점이 있는지를 분석한다.
- 인터페이스 기능이 수행되는 각 구간의 구현 현황은 송·수신 영역의 구현 기술 및 특징 등을 구체적으로 확인한다.
- 확인된 인터페이스 기능을 기반으로 송신 데이터 선택, 송신 객체 생성, 인터페이스 송·수신, 데이터 처리 결과 전송 등 영역별로 발생할 수 있는 보안 취약점을 시나리오 형태로 작성한다.

3 인터페이스 보안 기능 적용

- 분석한 인터페이스 기능과 보안 취약점을 기반으로 인터페이스 보안 기능을 적용한다.
- 인터페이스 보안 기능은 일반적으로 네트워크, 애플리케이션, 데이터베이스 영역에 적용한다.

네트워크 영역	<ul style="list-style-type: none"> • 인터페이스 송·수신 간 스니핑(Sniffing) 등을 이용한 데이터 탈취 및 변조 위험을 방지하기 위해 네트워크 트래픽에 대한 암호화를 설정한다. • 암호화는 인터페이스 아키텍처에 따라 IPsec*, SSL*, S-HTTP* 등의 다양한 방식으로 적용한다.
애플리케이션 영역	<p>소프트웨어 개발 보안 가이드를 참조하여 애플리케이션 코드 상의 보안 취약점을 보완하는 방향으로 애플리케이션 보안 기능을 적용한다.</p>
데이터베이스 영역	<ul style="list-style-type: none"> • 데이터베이스, 스키마, 엔티티의 접근 권한과 프로시저(Procedure), 트리거(Trigger)* 등 데이터베이스 동작 객체의 보안 취약점에 보안 기능을 적용한다. • 개인 정보나 업무상 민감한 데이터의 경우 암호화나 익명화* 등 데이터 자체의 보안 방안도 고려한다.

전문가의 조언

인터페이스는 서로 다른 시스템 사이에서 데이터를 원활히 주고받을 수 있는 통로로 사용되므로 데이터 변조 및 탈취 등의 위험에 노출되어 있습니다. 인터페이스의 보안성 향상을 위해서는 인터페이스가 구현되는 각 구간에서 보안에 취약한 부분을 확인한 후 알맞은 보안 기능을 적용해야 합니다. 보안성 향상을 위한 인터페이스 보안 취약점 분석 및 보안 기능 적용 방법에 대해 정리하세요.

- IPsec(IP Security) : 네트워크 계층에서 IP 패킷 단위의 데이터 변조 방지 및 은닉 기능을 제공하는 프로토콜입니다.
- SSL(Secure Sockets Layer) : TCP/IP 계층과 애플리케이션 계층 사이에서 인증, 암호화, 무결성을 보장하는 프로토콜입니다.
- S-HTTP(Secure Hypertext Transfer Protocol) : 클라이언트와 서버 간에 전송되는 모든 메시지를 암호화 하는 프로토콜입니다.

트리거(Trigger)

데이터베이스 시스템에서 데이터의 입력, 갱신, 삭제 등의 이벤트가 발생할 때마다 자동적으로 수행되는 절차형 SQL을 의미합니다.

데이터 익명화

데이터에 포함된 개인식별정보를 영구적으로 삭제하거나 알아볼 수 없는 형태로 변환하는 것을 의미합니다.



스니핑(Sniffing) / 소프트웨어 개발 보안

스니핑(Sniffing)

스니핑은 네트워크의 중간에서 남의 패킷 정보를 도청하는 해킹 유형의 하나로 수동적 공격에 해당합니다.

- 네트워크 내의 패킷들은 대부분 암호화되어 있지 않아 스니핑 같은 해킹 기법에 이용당하기 쉽습니다.

소프트웨어 개발 보안

애플리케이션 소스 코드에 존재할 수 있는 보안 취약점의 발견, 제거, 보안을 고려한 기능 설계 및 구현 등 소프트웨어 개발 과정에서 지켜야 할 일련의 보안 활동으로, 시큐어 코딩(Secure Coding)이라고도 불립니다.



기출문제 따라잡기

Section 066

출제예상

1. 다음 중 인터페이스 보안에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스는 시스템 모듈 간 통신 및 정보 교환을 지원하므로 데이터 변조·탈취 및 인터페이스 모듈 자체에 보안 취약점이 있다.
- ② 인터페이스 보안 취약점을 분석하기 위해 송·수신 영역의 구현 기술 및 특징을 구체적으로 분석한다.
- ③ 인터페이스 보안 기능은 일반적으로 Network, Transport, DataBase 영역에 적용한다.
- ④ 보안 취약점 분석 시 각 단계 영역별로 일어날 수 있는 시나리오를 가정하여 자세하게 분석한다.

인터페이스 보안 기능은 일반적으로 네트워크(Network), 애플리케이션(Application), 데이터베이스(DataBase) 영역에 적용합니다.

출제예상

2. 안전한 소프트웨어 개발을 위해 소스 코드에 존재할 수 있는 보안 취약점의 발견 및 제거, 보안을 고려한 기능 설계 및 구현 등 소프트웨어 개발 과정에서 지켜야 할 일련의 보안 활동을 의미하는 용어는?

- ① 시큐어 코딩
- ② APM
- ③ 데이터베이스 암호화
- ④ SSL

소프트웨어 개발 과정(Coding) 중에 지켜야 할 보안(Secure) 활동은 무엇일까요?

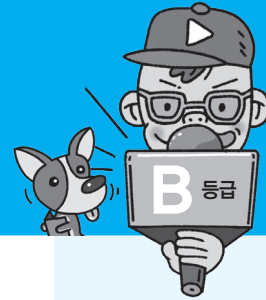
출제예상

3. 다음 중 인터페이스 보안 기능 적용에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스 보안 기능은 먼저 인터페이스 기능 및 보안 취약점을 확인한 후 이를 기반으로 적용한다.
- ② 네트워크 구간에 보안 기능 적용 시 중간자에 의한 데이터 탈취 및 위·변조를 막기 위해 네트워크 트래픽에 암호화를 수행한다.
- ③ 애플리케이션 구간에 보안 기능 적용 시 민감 데이터의 경우에는 데이터 자체에 대한 암호화, 익명화 등의 보안 방안을 고려한다.
- ④ 데이터베이스에 보안 기능 적용 시 데이터베이스의 접근 권한 및 데이터베이스 동작 객체의 보안 취약점을 보완하는 기능을 적용한다.

애플리케이션 영역의 보안은 소프트웨어 개발 보안 가이드를 참조하여 애플리케이션 코드 상 보안 취약점을 보완하는 방향으로 기능을 적용합니다.

▶ 정답 : 1. ③ 2. ① 3. ③



1 연계 테스트의 개요

연계 테스트는 구축된 연계 시스템과 연계 시스템의 구성 요소*가 정상적으로 동작하는지 확인하는 활동이다.

- 연계 테스트는 연계 테스트 케이스(Testcase) 작성, 연계 테스트 환경 구축, 연계 테스트 수행, 연계 테스트 수행 결과 검증 순으로 진행된다.

2 연계 테스트 케이스 작성

연계 테스트 케이스 작성은 연계 시스템 간의 데이터 및 프로세스의 흐름을 분석하여 필요한 테스트 항목을 도출하는 과정이다.

- 송·수신용 연계 응용 프로그램의 단위 테스트* 케이스와 연계 테스트 케이스를 각각 작성한다.
 - 송·수신용 연계 응용 프로그램의 단위 테스트 케이스
 - ▶ 송·수신 시스템에서 확인해야 할 항목을 도출한다.
 - ▶ 송·수신 시스템에서 단순 개별 데이터의 유효값을 확인*하는 경우의 수와 데이터 간의 연관 관계를 확인*하는 경우의 수로 구분하여 작성한다.
 - 연계 테스트 케이스
 - ▶ 송·수신용 연계 응용 프로그램의 기능상 결함을 확인하는 단위 테스트 케이스 형태로 작성한다.
 - ▶ 단위 테스트 케이스는 연계 테이블 간 송·수신 절차의 앞뒤로 연결하여 흐름을 확인할 수 있는 내용으로 작성한다.

3 연계 테스트 환경 구축

연계 테스트 환경 구축은 테스트의 일정, 방법, 절차, 소요 시간 등을 송·수신 기관과의 협의를 통해 결정하는 것이다.

- 연계 서버 또는 송·수신용 어댑터(Adapter) 설치, 연계를 위한 IP(Internet Protocol) 및 포트(Port) 허용 신청, 연계를 위한 DB 계정 및 테이블과 데이터 생성 등의 테스트 환경을 구축한다.

4 연계 테스트 수행

연계 테스트 수행은 연계 응용 프로그램을 실행하여 연계 테스트 케이스의 시험 항목 및 처리 절차 등을 실제로 진행하는 것이다.

전문가의 조언

연계 테스트는 쉽게 말해 송·수신 시스템이 이상없이 데이터를 주고받는지 확인하는 것을 말합니다. 연계 테스트의 테스트 케이스(Testcase) 작성, 테스트 환경 구축, 테스트 수행, 수행 결과 검증에 대해 정리하세요.

연계 시스템의 구성 요소

연계 시스템의 구성 요소에는 송·수신 모듈, 연계 서버, 모니터링 현황 등이 있습니다.

단위 테스트

단위 테스트는 모듈 안에 있는 개별적인 코드 단위가 정상적으로 작동하는지 확인하는 것을 의미합니다.

단순 개별 데이터의 유효값 확인

단순 개별 데이터 유효값 확인은 데이터 타입, 데이터 길이, 필수 입력 여부 등이 기존에 정의된 데이터 입력 조건과 맞는지 확인하는 것입니다.

데이터 간의 연관 관계 확인

데이터 간 연관 관계 확인은 사용자의 요청이 수신 시스템의 운영 DB에 등록되지 않은 데이터를 변경하거나, 존재하지 않는 테이블에 데이터를 등록하는 경우 수행하는 점검 작업을 의미합니다.

- 송·수신용 연계 응용 프로그램의 단위 테스트를 먼저 수행한다.
- 단위 테스트의 수행을 완료한 후 연계 테스트 케이스에 따라 데이터 추출, 데이터 송·수신, 데이터 반영 과정 등의 연계 테스트를 수행한다.

5 연계 테스트 수행 결과 검증

연계 테스트 수행 결과 검증은 연계 테스트 케이스의 시험 항목 및 처리 절차를 수행한 결과가 예상 결과와 동일한지를 확인하는 것이다.

- 연계 테스트 수행 결과는 다음과 같은 테스트 케이스 항목별 검증 방법을 이용하여 검증한다.
 - 운영 DB 테이블의 건수를 확인하는 방법
 - 테이블 또는 파일을 열어 데이터를 확인하는 방법
 - 파일 생성 위치에서 파일 생성 여부 및 파일 크기를 확인하는 방법
 - 연계 서버에서 제공하는 모니터링 현황을 확인하는 방법
 - 시스템에서 기록하는 로그(Log)를 확인하는 방법



기출문제 따라잡기

Section 067

출제예상

1. 다음 보기가 설명하고 있는 것은?

구축된 연계 시스템과 연계 시스템의 구성 요소가 정상적으로 동작하는지 확인하고 검증하는 활동이다.

- ① 연계 모듈 구현
- ② 연계 데이터 보안
- ③ 연계 테스트
- ④ 연계 메커니즘 구현

무엇을 확인 및 검증하는 것을 흔히 테스트(Test)라고 하지 않나요?

출제예상

2. 다음 중 연계 테스트의 수행 순서를 올바르게 나열한 것은?

- ㉠ 연계 테스트 수행
- ㉡ 연계 테스트 케이스 작성
- ㉢ 연계 테스트 환경 구축
- ㉣ 연계 테스트 수행 결과 검증

- ① ㉠ → ㉡ → ㉢ → ㉣
- ② ㉡ → ㉢ → ㉠ → ㉣
- ③ ㉢ → ㉡ → ㉠ → ㉣
- ④ ㉠ → ㉢ → ㉡ → ㉣

연계 테스트의 수행 순서는 무조건 외우기보단 내용을 이해하는 것이 좋습니다. 연계 테스트는 테스트 케이스를 작성하고 환경을 구축한 다음 테스트를 수행하여 결과를 확인하고 검증하는 것입니다.

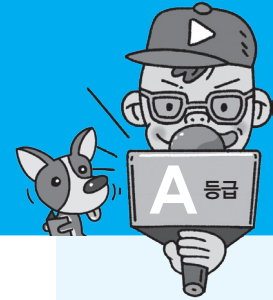
출제예상

3. 다음 중 연계 테스트 결과의 테스트 케이스 항목별 검증 방법으로 옳지 않은 것은?

- ① 운영 DB 테이블의 건수를 확인하는 방법
- ② 시스템에서 기록하는 로그(Log)를 확인하는 방법
- ③ 테이블 또는 파일을 열어 데이터를 확인하는 방법
- ④ 수신 시스템에서 제공하는 모니터링 현황을 확인하는 방법

모니터링 현황은 연계 서버에서 제공합니다.

▶ 정답 : 1. ③ 2. ② 3. ④



1 인터페이스 구현 검증의 개요

인터페이스 구현 검증은 인터페이스가 정상적으로 문제없이 작동하는지 확인하는 것이다.

- 인터페이스 구현 검증 도구와 감시 도구를 이용하여 인터페이스의 동작 상태를 확인한다.

2 인터페이스 구현 검증 도구

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

도구	기능
xUnit	Java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크이다.
STAF	서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크이다.
FitNesse	웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크이다.
NTAF	FitNesse의 장점인 협업 기능과 STAF의 장점인 재사용 및 확장성을 통합한 NHN(Naver)의 테스트 자동화 프레임워크이다.
Selenium	다양한 브라우저 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크이다.
watir	Ruby*를 사용하는 애플리케이션 테스트 프레임워크이다.

3 인터페이스 구현 감시 도구

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

전문가의 조언

인터페이스 구현 검증은 인터페이스와 관련된 프로그램이 정상적으로 구현되어 작동하는지 확인하는 것을 말합니다. 인터페이스 구현 검증 도구는 종류별 기능을 구분해서 숙지하고, 인터페이스 구현을 검증하는 방법에 대해 정리하세요.

인터페이스 구현 검증과 감시의 차이점

인터페이스 구현 검증은 인터페이스의 입·출력값이 예상과 일치하는지 확인하는 것이고, 인터페이스 구현 감시는 구현된 인터페이스가 외부 시스템과 연결 모듈 사이에서 정상적으로 동작하는지 확인하는 것입니다.

Ruby

Ruby는 마츠모토 유키히로가 개발한 인터프리터 방식의 객체 지향 스크립트 언어입니다.

스카우터(Scouter)

스카우터는 애플리케이션 및 OS 자원에 대한 모니터링 기능을 제공하는 오픈소스 APM 소프트웨어입니다.

제니퍼(Jennifer)

제니퍼는 애플리케이션의 개발부터 테스트, 운영, 안정화까지, 전 단계에 걸쳐 성능을 모니터링하고 분석해주는 APM 소프트웨어입니다.

인터페이스 기능 구현 정의에 대한 자세한 내용은 Section 063을 참조하세요.



APM(Application Performance Management/Monitoring)

APM은 애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구를 의미합니다.

- APM은 리소스 방식과 엔드투엔드(End-to-End)의 두 가지 유형이 있습니다.
 - 리소스 방식 : Nagios, Zabbix, Cacti 등
 - 엔드투엔드 방식 : VisualVM, 제니퍼, 스카우터 등

4 인터페이스 구현 검증 도구 및 감시 도구 선택

- 인터페이스 기능 구현 정의*를 통해 구현된 인터페이스 명세서의 세부 기능을 참조하여 인터페이스의 정상적인 동작 여부를 확인하기 위한 검증 도구와 감시 도구의 요건을 분석한다.
- 분석이 끝나면 시장 및 솔루션 조사를 통해서 적절한 인터페이스 구현을 검증하고 감시하는데 필요한 인터페이스 구현 검증 도구와 감시 도구를 선택한다.

5 인터페이스 구현 검증 확인

- 인터페이스 구현 검증 도구를 이용하여 외부 시스템과 연계 모듈의 동작 상태를 확인한다.
- 최초 입력값과 입력값에 의해 선택되는 데이터, 생성되는 객체의 데이터 등 전반적인 인터페이스 동작 프로세스상에서 예상되는 결과값과 실제 검증값이 동일한지를 비교한다.
- 추가적으로 각 단계별 오류 처리도 적절하게 구현되어 있는지 확인한다.

6 인터페이스 구현 감시 확인

- 인터페이스 구현 감시 도구를 이용하여 외부 시스템과 연결 모듈이 서비스를 제공하는 동안 정상적으로 동작하는지 확인한다.
- 인터페이스 동작 여부, 에러 발생 여부 등 감시 도구에서 제공해 주는 리포트를 활용한다.



Section 068

1. 다음 중 인터페이스 구현 검증 및 감시에 대한 설명으로 가장 옳지 않은 것은?

2. 다음 중 애플리케이션의 흐름 모니터링과 성능 예측을 통해 최적의 애플리케이션 상태를 보장 및 관리하는 것을 의미하는 용어는?

3. 다음 중 테스트 다중화 도구에 대한 설명으로 가장 옳지 않은 것은?

4. 다음 중 xUnit, STAF, FitNesse, NTAf 등과 관련 깊은 것은?

- 5장 인터페이스 구현 287



전문가의 조언

인터페이스 오류 확인 방법을 즉시 확인하는 방법과 주기적으로 확인하는 방법으로 구분하여 정리하고, 오류 보고서 작성의 특징을 기억하세요.

1 인터페이스 오류 확인 및 처리 보고서의 개요

- 인터페이스는 독립적으로 떨어져 있는 시스템 간 연계를 위한 기능이므로 인터페이스에서 발생하는 오류는 대부분 중요한 오류이다.
- 인터페이스 오류 발생 시 사용자 또는 관리자는 오류사항을 확인하고 오류 처리 보고서를 작성하여 보고 체계에 따라 관리 조직에 보고해야 한다.
- 인터페이스 오류 확인 방법에는 오류 발생 즉시 확인하는 방법과 주기적인 확인 방법이 있다.

2 인터페이스 오류 발생 즉시 확인

- 인터페이스 오류가 발생하면 화면에 오류 메시지를 표시하고 자동으로 SMS (Simple Message System), 이메일을 발송하므로 즉시 오류 발생을 확인할 수 있다.
- 인터페이스 오류 발생을 즉시 처리하는 것은 가장 직관적인 방법이기 때문에 가장 많이 사용되고 있다.

예 인터페이스 오류 발생 즉시 확인

오류 확인 방법	시나리오
오류 메시지 알람 표시	<ul style="list-style-type: none"> • 인터페이스 오류 발생 시 사용자 화면에 오류 메시지 창을 알람 형태로 표시한다. • 사용자는 오류 발생 즉시 알 수 있지만 관리자는 사용자를 통해야만 알 수 있다.
오류 SMS 발송	<ul style="list-style-type: none"> • 오류 발생 시 자동으로 사용자와 관리자에게 SMS를 발송한다. • 사용자, 관리자 모두 오류 발생 즉시 오류를 알 수 있다.
오류 내역 이메일 발송	<ul style="list-style-type: none"> • 오류 발생 시 자동으로 사용자와 관리자에게 이메일을 발송한다. • 이메일을 확인해야 오류 발생 사실을 알 수 있다.

3 주기적인 인터페이스 오류 발생 확인

- 시스템 관리자가 시스템의 현재 상태를 보여주는 시스템 로그나 인터페이스 오류 관련 테이블 등을 통해 주기적으로 오류 발생 여부를 확인한다.
- 발생하는 오류에 대한 정보가 주기적으로 축적되면 오류의 원인 파악이 용이하기 때문에 오류의 재발을 방지할 수 있는 계획을 세울 수 있다.

예 주기적인 인터페이스 오류 발생 확인

오류 확인 방법	시나리오
인터페이스 오류 로그 확인	<ul style="list-style-type: none"> • 인터페이스 오류 발생 시 관련 오류를 별도의 로그 파일*로 생성하여 보관한다. • 자세한 오류 원인 및 내역을 확인할 수 있다. • 오류 로그 파일은 시스템 관리자나 운영자만 확인이 가능하다.
인터페이스 오류 테이블 확인	<ul style="list-style-type: none"> • 인터페이스 관련 테이블에 오류사항을 기록한다. • 오류사항의 확인이 쉬워 운영자가 관리하기 용이하다. • 오류사항이 구체적이지 않아 별도의 분석이 필요한 경우가 있다.
인터페이스 감시 (APM*) 도구 사용	<ul style="list-style-type: none"> • 스카우터(Scouter)나 제니퍼(Jennifer) 등의 인터페이스 감시 도구를 사용하여 주기적으로 오류 발생 여부를 확인한다. • 인터페이스의 전반적인 상황을 확인할 수 있다.

로그 파일(Log File)

컴퓨터 시스템의 모든 실행 내역을 수집하여 기록하는 파일을 의미합니다.

APM(Application Performance Management)

애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구를 의미합니다.

4 인터페이스 오류 처리 보고서 작성

인터페이스 오류 처리 보고서는 인터페이스 작동 시 발생하는 오류의 발생 및 종료 시점, 원인 및 증상, 처리사항 등을 정리한 문서이다.

- 인터페이스 오류 처리 보고서는 오류 발생 즉시 신속하게 작성하여 조직의 보고 체계에 따라 보고한다.
- 인터페이스 오류 처리 보고서는 일반적인 정형화된 형식이 없기 때문에 조직 또는 오류 발생 시 상황에 맞춰 작성한다.
- 인터페이스 오류 처리 보고서는 오류 관련 사항을 시간 경과에 따라 기록한다.
- 다음은 보고 시기에 따른 인터페이스 오류 처리 보고서의 특징이다.

보고 시기	특징
최초 발생 시	<ul style="list-style-type: none"> • 인터페이스 오류 발생 상황을 인지하면 신속하게 조직에 보고하고 대응 조직을 만든다. • 오류 발생 구간 및 시점, 영향도 등을 간이 보고서, 이메일, SMS 등으로 보고한다.
오류 처리 경과 시	오류 처리 진행 상황과 오류에 관한 공지사항 등록 등을 보고한다.
완료 시	<ul style="list-style-type: none"> • 최종 조치 후 내부 조직과 고객사에 완료됐음을 보고한다. • 오류 발생 시점, 오류 처리 경과, 오류 재발 방지 대책 등 종합적인 내용을 보고한다.



출제예상

1. 다음 중 인터페이스 오류에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스에서 발생하는 오류는 중요한 오류인 경우가 많으므로 사용자나 관리자는 인터페이스 오류 사항을 잘 확인하고 보고하여야 한다.
- ② 인터페이스 오류 처리 방법은 크게 오류 발생 즉시 처리하는 방법과 주기적으로 처리하는 방법이었다.
- ③ 인터페이스 오류 발생 즉시 처리하려면 인터페이스 오류 시스템 로그를 별도로 작성하여 파일로 보관한다.
- ④ 주기적으로 인터페이스 오류를 처리하려면 인터페이스 관련 테이블에 오류사항을 기록한다.

인터페이스 오류 발생 즉시 처리는 '알람, 이메일, SMS'를, 주기적 처리는 '시스템 로그 파일, 테이블'이란 것을 기억해 두세요.

출제예상

2. 다음 중 인터페이스 오류 발생 즉시 처리하는 방법에 대한 설명으로 가장 옳지 않은 것은?

- ① 사용자 화면에서 에러 메시지를 알람 형태로 보여 준다.
- ② 오류 발생 시 사용자와 관리자에게 이메일을 통해 알려 준다.
- ③ 오류 발생 시 사용자와 관리자에게 SMS를 통해 알려 준다.
- ④ 인터페이스 관련 테이블에 오류 내역을 작성하여 알려 준다.

위의 문제를 풀었기 때문에 어렵지 않게 답을 찾을 수 있겠죠.

출제예상

3. 다음 중 주기적인 인터페이스 오류 발생 처리 방법에 대한 설명으로 가장 옳지 않은 것은?

- ① 시스템 로그나 인터페이스 오류 관련 테이블 등을 시스템 관리자가 주기적으로 확인하여 오류 발생 여부를 확인한다.
- ② 인터페이스 오류 테이블을 활용하면 오류 내역 및 원인을 구체적으로 확인할 수 있다.
- ③ 인터페이스 감시 도구를 활용하면 인터페이스의 전반적인 상황을 모두 확인할 수 있다.
- ④ 오류 이력이 쌓이면 주기적으로 발생하는 오류 원인의 분석 및 파악이 용이하므로 오류의 재발을 막을 수 있다.

인터페이스 오류 테이블을 이용할 경우 오류사항이 구체적이지 않아 별도의 분석이 필요한 경우가 있습니다.

출제예상

4. 다음 중 인터페이스 오류 처리 보고서에 대한 설명으로 가장 옳지 않은 것은?

- ① 인터페이스 오류 처리 보고서는 인터페이스 구현 시 발생하는 오류의 발생 및 종료 시점, 원인 및 증상, 조치 사항 등을 정리한 문서이다.
- ② 인터페이스 오류 처리 보고서는 정형화된 형식에 따라 작성한다.
- ③ 인터페이스 오류 처리 보고서는 신속하게 작성하여 보고 체계에 맞게 보고 하는 것이 중요하다.
- ④ 인터페이스 오류 처리 보고서는 인터페이스 오류 발생 시 상황 인지 및 조치 사항을 시간 경과에 따라 작성한다.

인터페이스는 기업의 특성이나 송·수신 시스템에 따라 다를 수 있는데 정형화된 형식이 있을까요?

▶ 정답 : 1. ③ 2. ④ 3. ② 4. ②



1. 서로 다른 시스템 또는 컴포넌트 간에 데이터 교환 및 처리를 위한 목적으로 각 시스템의 교환 데이터 및 업무, 송·수신 주체 등이 정의되어 있는 문서는?

- ① 인터페이스 요구사항 정의서
- ② 인터페이스 설계서
- ③ 인터페이스 업무정의서
- ④ 인터페이스 데이터 표준

2. 다음 중 시스템 인터페이스 설계서에 대한 설명으로 옳은 것은?

- ① 한 시스템의 인터페이스 현황을 확인하기 위하여 시스템이 갖는 인터페이스 목록과 인터페이스 명세를 보여주는 문서이다.
- ② 인터페이스를 통한 각 세부 기능의 개요, 세부 기능이 동작하기 전에 필요한 사전·사후 조건 등을 정의한 문서이다.
- ③ 정적, 동적 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 통해 정의한 문서이다.
- ④ 제공하는 인터페이스 서비스에 대한 상세 명세를 표현하는 문서이다.

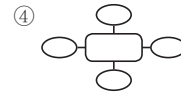
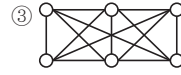
3. EAI의 구축 유형 중 Hybrid 방식에 대한 설명으로 가장 옳지 않은 것은?

- ① Hub & Spoke와 메시지 버스 방식의 혼합 방식이다.
- ② 그룹 내에는 Hub & Spoke 방식을, 그룹 간에는 메시지 버스 방식을 사용한다.
- ③ 두 가지 방식 중 필요 시 한 가지 방식으로 EAI 구현이 가능하다.
- ④ 데이터 병목 현상을 최소화 할 수 있다.

4. 애플리케이션 간의 통합 측면에서 EAI와 유사하다고 볼 수 있으나 애플리케이션 보다는 서비스 중심으로 통합을 지향하는 아키텍처 또는 기술을 의미하는 것은?

- ① MOM
- ② Point-to-Point
- ③ ESB
- ④ Message Bus

5. EAI의 구축 유형 중 가장 기본적인 애플리케이션 통합 방식인 Point-to-Point의 개념도는?



6. 다음 중 시스템 인터페이스를 위한 내·외부 모듈 연계 기술이 아닌 것은?

- ① ESM
- ② Hybrid
- ③ Point-to-Point
- ④ Hub & Spoke

7. 다음 중 컴포넌트 명세서에 있는 인터페이스 클래스의 세부 조건 및 기능 등을 정의한 것은?

- ① 인터페이스 목록
- ② 인터페이스 명세서
- ③ 인터페이스 데이터 표준
- ④ 인터페이스 업무 정의서

8. 하나의 독립적인 기능을 수행하는 모듈의 구성 요소와 세부적인 동작을 정의한 것으로, 컴포넌트의 구성 요소와 동작을 정의한 컴포넌트 명세서와 컴포넌트와 컴포넌트 간 상호 작용을 정의한 인터페이스 명세서로 구성된 것은?

- ① 인터페이스 설계서
- ② 테이블 정의서
- ③ 인터페이스 요구사항 정의서
- ④ 모듈 세부 설계서

9. 속성-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷으로, AJAX에서 많이 사용되며, XML을 대체하는 주요 데이터 포맷은?

- ① JSON
- ② SGML
- ③ UML
- ④ API

10. 다음 중 XML에 대한 설명으로 옳은 것은?

- ① 인터넷의 표준 문서인 하이퍼텍스트 문서를 만들기 위해 사용하는 언어이다.
- ② 다른 특수한 목적을 갖는 마크업 언어를 만드는 데 사용하도록 권장하는 다목적 마크업 언어이다.
- ③ 시스템 개발 과정에서 시스템 개발자와 고객 간의 원활한 의사소통을 위해 표준화한 객체지향 모델링 언어이다.
- ④ 속성-값 쌍(Attribute-Value Pairs)으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷이다.



11. 다음 중 송 · 수신 인터페이스 엔티티를 사용하여 인터페이스 동작을 할 때 발생할 수 있는 상황이 아닌 것은?

- ① 데이터 정합성 오류
- ② 데이터 전송 주체의 논리적 오류
- ③ DB Connection 오류
- ④ 송신 데이터 생성 시 프로세스의 논리적 결함

12. 다음 중 인터페이스 보안 기능을 적용하는 일반적인 영역에 해당하지 않는 것은?

- ① Application ② DataBase
- ③ Transport ④ Network

13. 다음 중 연계 테스트에 대한 설명으로 가장 옳지 않은 것은?

- ① 연계 테스트는 '테스트 케이스 작성 → 환경 구축 → 테스트 수행 → 결과 검증' 순으로 진행된다.
- ② 연계 테스트 환경 구축은 연계 시스템 간의 데이터 및 프로세스의 흐름을 분석하여 필요한 테스트 항목을 도출하는 과정이다.
- ③ 연계 테스트 수행은 연계 테스트 케이스의 시험 항목 및 처리 절차 등을 수행하는 것이다.
- ④ 연계 테스트 결과 검증은 테스트 수행 결과가 예상 결과와 동일한지를 확인하는 것이다.

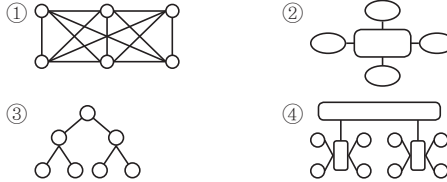
14. 다음 중 인터페이스 구현 검증 도구에 대한 설명으로 옳은 것은?

- ① xUnit : Java, C++, .Net 등 다양한 언어를 지원하는 단위 테스트 프레임워크
- ② Selenium : 웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크
- ③ STAF : FitNesse와 STAF의 장점을 통합한 NHN의 테스트 자동화 프레임워크
- ④ FitNesse : Ruby를 사용하는 애플리케이션 테스트 프레임워크

15. 다음 중 인터페이스 오류 발생 즉시 확인하는 방법이 아닌 것은?

- ① 알람
- ② SMS
- ③ 이메일
- ④ 로그 파일

16. EAI 구축 유형 중 Hub & Spoke 방식의 개념도는?



17. 다음 중 인터페이스 명세서의 항목이 아닌 것은?

- ① 인터페이스명
- ② 오퍼레이션명
- ③ 반환값
- ④ 컴포넌트명

18. 다음 중 인터페이스 객체 송 · 수신 시 발생할 수 있는 예외 원인이 아닌 것은?

- ① 네트워크 및 서버 불안정
- ② 데이터 트랜잭션 시 프로그램의 논리상 오류
- ③ 특수문자 등으로 파싱 시 오류
- ④ 수신 인터페이스 데이터 처리 시 프로그램의 논리적 결함

19. 인터페이스 보안 기능을 적용할 때 DB, Schema, Entity 등의 접근 권한과 Procedure, Trigger 등 데이터베이스 동작 객체의 보안 취약점을 보완해야 하는 영역은?

- ① Application ② DataBase
- ③ Network ④ Transport

**2. Section 060**

②번은 상세 기능별 인터페이스 명세서, ③번은 정적·동적 모형을 통한 인터페이스 설계서, ④번은 데이터 정의를 통한 인터페이스 설계서에 대한 설명이다.

3. Section 061

Hybrid 방식은 데이터 병목 현상을 최소화 할 수 있다.

4. Section 061

- MOM(Message Oriented Middleware, 메시지 지향 미들웨어) : 메시지 기반의 비동기형 메시지를 전달하는 방식의 미들웨어
- Point-to-Point : 가장 기본적인 애플리케이션 통합 방식으로, 애플리케이션을 1:1로 연결함
- Message Bus : 애플리케이션 사이에 미들웨어를 두어 처리하는 방식

5. Section 061

①번은 Message Bus, ②번은 Hybrid, ④번은 Hub & Spoke의 개념도이다.

6. Section 061

ESM(Enterprise Security Management)은 다양한 장비에서 발생하는 로그 및 보안 이벤트를 통합하여 관리하는 보안 솔루션이다.

10. Section 064

①번은 HTML(Hyper Text Markup Language), ③번은 UML(Unified Modeling Language), ④번은 JSON(JavaScript Object Notation)에 대한 설명이다.

11. Section 065

④번은 인터페이스 객체 송·수신 시 발생할 수 있는 예외 상황이다.

12. Section 066

인터페이스 보안 기능은 일반적으로 네트워크(Network), 애플리케이션(Application), 데이터베이스(DataBase) 영역에 적용한다.

13. Section 067

연계 테스트 환경 구축은 테스트의 일정, 방법, 절차, 소요 시간 등을 송·수신 기관과의 협의를 통해 결정하는 것이다. ②번의 내용은 연계 테스트 케이스 작성에 대한 설명이다.

14. Section 068

다른 보기가 틀린 이유는 다음과 같다.

- ② Selenium은 다양한 브라우저 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크이다.
- ③ STAF는 서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크이다. ③번의 내용은 NTAF에 대한 설명이다.
- ④ FitNesse는 웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크이다. ④번의 내용은 watir에 대한 설명이다.

15. Section 069

로그 파일 작성은 주기적인 인터페이스 오류 확인 방법에 해당한다.

16. Section 061

Hub & Spoke는 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중적 방식이다.

17. Section 063

인터페이스 명세서 항목

인터페이스ID, 인터페이스명, 오퍼레이션명, 오퍼레이션 개요, 사전 조건, 사후 조건, 파라미터, 반환값 등

18. Section 065

②번은 송·수신 인터페이스 엔티티를 사용하여 인터페이스를 수행할 때 발생할 수 있는 예외 상황이다.

19. Section 066

인터페이스 보안 기능

- 네트워크 영역 : 인터페이스 송·수신 간 스니핑(Sniffing) 등을 이용한 데이터 탈취 및 변조 위협을 방지하기 위해 네트워크 트래픽에 대한 암호화 설정
- 애플리케이션 영역 : 소프트웨어 개발 보안 가이드를 참조하여 애플리케이션 코드 상의 보안 취약점을 보완하는 방향으로 애플리케이션 보안 기능 적용
- 데이터베이스 영역 : 데이터베이스, 스키마, 엔티티의 접근 권한과 프로시저(Procedure), 트리거(Trieger) 등 데이터베이스 동작 객체의 보안 취약점에 보안 기능 적용