[1장 연습문제]

1. 소프트웨어 공학의 기본 원칙이라고 볼 수 없는 것은? ④
① 품질 높은 소프트웨어 상품 개발
② 지속적인 검증 시행
③ 결과에 대한 명확한 기록 유지
④ 최대한 많은 인력 투입

2. 다음 설명에 해당하는 소프트웨어 개발 프로세스 방법은? ④
- 설계가 변경되어도 이를 잘 수용할 수 있도록 짧게 반복하면서 소프트웨어를 개발하는 방법
- 반복적이고 진화적인 프로세스와 유사하지만 경험의 축적으로 나온 모델
- 소프트웨어 개발이 인력 집약적이라는 데 관심을 두고 사람과 팀 구성에 초점을 맞춤

① RAD 모델(Rapid Application Development Model)
② 반복/점증 모델(Iterative/Incremental Model)
③ 나선형 모델(Spiral Model)
④ 애자일 프로세스 모델(Agile Process Model)

3. 소프트웨어 생명주기 모델 중에서 고전적 생명주기 모델로 선형 순차적 모델이라고도 하며 타당성 검토, 계획, 요구사항 분석, 구현, 테스트, 유지보수의 단계를 통해 소프트웨어를 개발하는 모형은?
①
① 폭포수 모형
② 애자일 모형
③ 컴포넌트 기반 방법론
④ 6GT 모형

4. 폭포수 모델의 특징으로 거리가 먼 것은? ①
① 개발 중 발생한 요구사항을 쉽게 반영할 수 있다.
② 순차적인 접근 방법을 이용한다.
③ 단계적 정의와 산출물이 명확하다.
④ 모델의 적용 경험과 성공사례가 많다.

5. 다음 설명에 해당하는 생명주기 모델로 가장 옳은 것은? ③

가장 오래된 모델로 적용 사례가 많이 있다. 요구사항의 변경이 어려우며 각 단계의 결과가 확인되어야만 다음 단계로 넘어간다. 선형 순차적 모델로 고전적 생명주기 모델이라고도 한다.

① 패키지 모델

② 코코모 모델
③ 폭포수 모델
④ 관계형 모델

6. 소프트웨어 개발 프로세스 모델 중 폭포수 waterfall 모델에 대한 설명으로 옳지 않은 것은? ④
① 요구사항 분석을 완료한 후 설계 작업을 시작할 수 있다 .
② 개발 후반부가 되어야 실행 가능한 소프트웨어가 만들어진다.
③ 단계별 산출물을 체계적으로 문서화할 수 있다.
④ 소프트웨어 요구사항의 변경이 많은 경우에 적합한 모델이다

7. 프로토타입을 지속적으로 발전시켜 최종 소프트웨어 개발까지 이르는 방법으로 위험 관리가 중심인 소프트웨어 생명주기 모델은? ①
① 나선형 모델
② 델파이 모델
③ 폭포수 모델
④ 기능 점수 모델

8. 소프트웨어 개발 모델 중 나선형 모델의 4가지 주요 활동이 순서대로 나열된 것은? ②
Ⓐ 계획 수립 Ⓑ 사용자 평가 Ⓒ 개발 및 검증 Ⓓ 위험 분석

① Ⓐ-Ⓑ-Ⓓ-Ⓒ순으로 반복
② Ⓐ-Ⓓ-Ⓒ-Ⓑ순으로 반복
③ Ⓐ-Ⓑ-Ⓒ-Ⓓ순으로 반복
④ Ⓐ-Ⓒ-Ⓑ-Ⓓ순으로 반복

9. XP(eXtreme Programming)의 5가지 가치로 거리가 먼 것은? ③
① 용기
② 의사소통
③ 정형 분석
④ 피드백

10. XP의 기본 원리로 볼 수 없는 것은? ①
① Linear Sequential Method
② Pair Programming
③ Collective Ownership
④ Continuous Integration

11. 애자일 기법에 대한 설명으로 맞지 않는 것은? ②
① 절차와 도구보다 개인과 소통을 중요하게 생각한다.
② 계획에 중점을 두어 변경 대응이 난해하다.
③ 소프트웨어가 잘 실행되는 데 가치를 둔다.
④ 고객과의 피드백을 중요하게 생각한다.

12. 스크럼(scrum)의 제품 기능 목록(product backlog)에 대한 설명으로 옳지 않은 것은? ③
① 제품 기능 목록에 있는 업무 목록은 프로젝트를 수행하는 동안 수정되고 정제된다.
② 제품 기능 목록의 업무 중 우선순위가 높은 항목부터 개발한다.
③ 제품 기능 목록에 있는 업무의 우선순위를 결정한 후에는 변경하지 않는다.
④ 제품 책임자(product owner)가 제품 기능 목록을 관리한다.

13. 다음 설명에 해당하는 스크럼 관련 활동은? ①
스프린트가 끝나는 시점이나 일정 주기로 수행한다. 이 활동을 통해 프로젝트를 진행하는 과정에서 드러난 좋았던 점, 여러 가지 문제나 미진한 점 등을 도출한다. 이 활동을 통해 이미 설정된 프로세스로만 프로젝트를 진행하지 않고 지속적으로 개선해 변화하는 비즈니스 환경에 보다 능동적으로 적응할 수 있도록 한다.

① 스프린트 회고
② 스프린트 검토 회의
③ 일일 스크럼 회의
④ 배포 계획

14. 소프트웨어 개발 단계를 시간의 흐름에 따라 4개의 범주(도입, 상세, 구축, 이행)로 나누고 각 범주에는 요구사항 도출부터 설계, 구현, 평가까지 개발 생명주기가 포함되어 있는 방법론은? ②
① XP(eXtreme Programming)
② UP(Unified Process)
③ CMM(Capability Maturity Model)
④ SPICE(Software Process Improvement and Capability dEtermination)

15. 테일러링(Tailoring) 개발 방법론의 내부 기준에 해당하지 않는 것은? ④
① 납기/비용
② 기술환경
③ 구성원 능력
④ 국제 표준 품질기준

[참고] **테일러링 개발 방법론**
소프트웨어 개발 방법론을 사용할 때 프로젝트의 특성에 맞게 절차나 사용기법 등을 수정, 보완하는 것을 말한다.

16. 공학의 특성과 소프트웨어 공학의 목표를 설명하시오.

공학의 특성:
- '공학'이 붙으면 문제를 해결할 때 무한정의 시간과 비용이 드는 것이 아니라 정해진 기간과 주어진 비용이라는 제한이 생긴다.
- 즉 공학은 과학적 지식을 활용해 문제를 해결하는 데 한정된 기간과 비용의 제약을 받는다.
- 공학이 발전하면 문제 해결을 위한 기술이 축적되고 공학적 원리가 개발된다.
- 그리고 이를 실무에 적용해 문제 해결의 절차를 만들고 반복적인 절차를 개선해 표준을 만들어낸다. 그러면 초보자도 이 표준 지침을 따라 문제를 해결할 수 있다.
- 요리할 줄 모르는 사람도 레시피(표준)를 따르다 보면 맛있는 요리가 완성되는 것과 같다.

소프트웨어 공학의 목표:
- 소프트웨어 공학은 '소프트웨어를 개발하는 과정에 공학적인 원리를 적용해 보자'는 것.
- 소프트웨어 공학의 목표는 소프트웨어 개발의 어려움을 해결하고, 효율적인 개발을 통해 생산성을 향상해 사용자가 만족하는 고품질의 소프트웨어 제품을 만들기 위해서이다.

17. 단계적 개발 모델에서 점증적 개발 방법과 반복적 개발 방법의 예를 만들어보시오.

- 점증적 개발 방법: 식사 유형에 비유해보자. 일반적으로 양식 코스 요리를 먹을 때는 야채 샐러드나 수프와 같은 전채 요리를 먼저 먹은 후에 스테이크와 같은 메인 요리를 먹는다. 그런 다음 후식을 먹는다. 이와 같이 점증적 방법도 '하나가 끝나면 그다음, 또 하나가 끝나면 그다음...'처럼 하나씩 늘려 간다.
- 소프트웨어 개발에서 점증적 방법: 중요하다고 생각되는 부분부터 차례로 개발한 후 그 일부를 사용하면서 개발 범위를 점차 늘려 가는 방식이다.

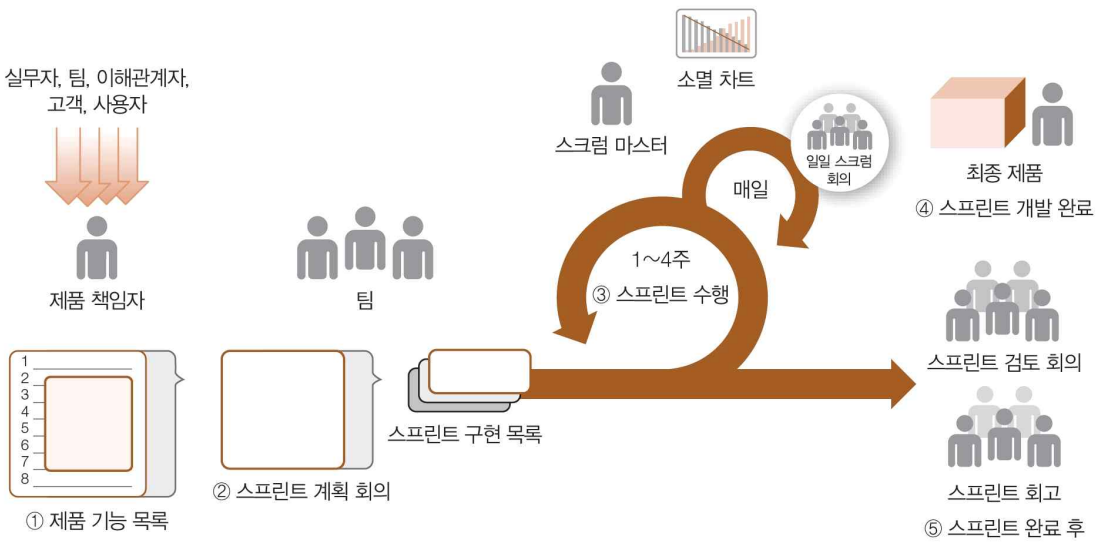
- 반복적 개발 방법: 한정식 상차림에 비유할 수 있다. 한정식을 먹을 때는 모든 음식을 한 상에 가득 차려놓고 조금씩 모두 맛본다. 그중 맛있는 음식은 한 번 더 먹고, 아주 맛있는 음식이라면 여러 번 먹고 식사를 마친다.
- 소프트웨어 개발에서 반복적 개발 방법: 초기에 시스템 전체를 일차적으로 개발해 인도한 후, 각 서브시스템의 기능과 성능을 변경 및 보강해 완성도를 높인다. 이렇게 업그레이드된 릴리스 버전을 다시 내놓는 것이다. 이 방식은 초기의 요구사항이 불분명한 경우에 적합하다.

18. 폭포수 모델과 애자일 방법의 차이점을 비교해서 설명하시오.

표 1-1 애자일 프로세스 모델과 폭포수 모델의 비교

구분	애자일 프로세스 모델	폭포수 모델
추가 요구사항의 수용	처음 수집한 요구사항을 전체 중 일부로 인정하고 시작하므로 언제든지 추가 요구사항이 있을 것으로 간주한다. 따라서 추가 요구사항을 수용할 수 있는 방법으로 설계되어 있다.	요구사항 분석이 완전히 완료된 후에 설계 단계로 넘어가므로 새로운 요구사항을 추가하기 쉽지 않다. 추가 요구사항을 반영하기 어려운 구조다.
릴리스 시점	가능하면 자주, 빨리 제품에 대한 프로토타입을 만들어 사용자에게 보여준다. 이러한 방식을 반복적으로 수행해 최종 제품을 만들기 때문에 자주 릴리스된다.	요구사항에 대한 분석, 설계, 구현 과정이 끝나고 최종 완성된 제품을 릴리스한다.
시작 상태	계속적인 추가 요구사항을 전제로 하는 방식이라 시작 단계에서는 부족한 점이 많지만 점차 완성도가 높아진다.	한 번 결정된 단계는 그 이후에 변동이 적어야 한다. 따라서 완성도를 최대한 높여 다음 단계로 넘어가기 위해 시작 단계의 완성도가 매우 높다.
고객과의 의사소통	사용자와 함께 일한다는 개념을 담고 있다. 처음부터 사용자의 참여를 유도하고 많은 대화를 하면서 개발을 진행한다.	사용자 요구사항을 정의한 후 사용자에게 더는 추가 요구가 없다는 확답을 받고 개발에 들어간다. 산출물을 근거로 하기 때문에 사용자와의 대화는 적다.
진행 상황 점검	개발자와 사용자는 개발 초기부터 지속적으로 진행 상황을 공유하며 함께 관심을 갖고 진행해 나간다.	단계별 산출물을 중요시하기 때문에 단계별 산출물에 대한 결과로 개발의 진척 상황을 점검한다.
분석, 설계, 구현 진행 과정	분석, 설계, 구현이 하나의 단계와 그 단계 안의 반복마다 한꺼번에 진행된다. 다만 어떤 단계에서는 분석이 많고 구현이 적고 어떤 단계에서는 분석이 적고 구현이 많다는 차이만 있을 뿐이다.	분석, 설계, 구현 과정이 명확하다. 각 과정에서 생산되는 산출물 중심의 개발 방식이기 때문에 단계가 명확히 구별되어 있다. 따라서 분석이 끝난 후 설계를, 설계가 끝난 후 구현 작업을 진행한다.
모듈(컴포넌트) 통합	개발 초기부터 빈번한 통합을 통해 문제점을 빨리 발견하고 수정하는 방식을 택한다. 문제점을 빨리 발견하므로 비용을 절감할 수 있다는 장점이 있다.	V 모델에서 설명한 것처럼 구현이 완료된 후에 모듈 간의 통합 작업을 수행한다.

19. 아래 그림을 보고 전체 내용을 글로 정리해보시오.



[단계 1] 제품 기능 목록(product backlog)을 작성한다.

- 제품 기능 목록은 일반적인 개발 방법의 요구사항에서 기능 목록과 같다고 보면 이해하기 쉽다. 즉 사용자가 요구하는 제품의 기능 목록을 말하며, 제품에 관한 모든 요구사항에 대해 우선순위가 정해져 있다.

이 우선순위는 고객측 대표인 제품 책임자(product owner)가 결정한다. 즉 제품 책임자가 요구사항 목록에 우선순위를 매겨 제품 기능 목록을 작성한다.
- 제품 기능 목록은 우선순위가 매겨진, 사용자의 요구사항 목록이라고 할 수 있으며 사용자와 계속 미팅하면서 목록이 완성된다. 한 번 결정된 제품 기능 목록은 확정된 것이 아니고 개발 중이라도 수정이 가능하지만 일반적으로 한 주기가 끝날 때까지는 제품 기능 목록을 수정하지 않는다.

[단계 2] 작성된 제품 기능 목록을 가지고 스프린트 계획 회의를 한다.
- 스프린트 계획 회의에서는 스프린트 구현 목록과 스프린트 개발 시간을 추정한다.

[단계 3] 스프린트를 수행한다.
- 이 단계는 스프린트를 개발한다. 개발 과정에서 일일 스크럼 회의를 통하여 매일의 할 일을 확인한다. 이때 변경 사항은 스프린트 현황판을 이용한다. 또한 개발의 진척도는 소멸 차트를 이용한다.

[단계 4] 스프린트 개발이 완료된다.
- 스프린트 개발이 완료되면 실행 가능한 최종 제품이 생산된다.

[단계 5] 스프린트 개발 완료로 최종 제품이 생산되면 다음 과정을 통해 스프린트를 되돌아봐야 한다.
- 스프린트 검토 회의(sprint review)
- 스프린트 회고(sprint retrospective)
- 두 번째 스프린트 계획 회의

20. 소프트웨어 공학 책을 만들기 위한 원고를 작성한다고 할 때 스프린트 구현 목록을 작성하시오.

※ 다음의 예처럼 작성하면 됩니다.

스프린트	사용자 스토리	SP	작업	MD	개발자
singleton 패턴	singleton 패턴을 사용하지 않은 일반 프로그램	1	개념 내용 기술	2h	김치수
			EspressoMachine 클래스 코드 개발		이장훈
			User 클래스 코드 개발		이장훈
	singleton 패턴을 사용하지 않은 일반 프로그램(Thread 사용)	1	개념 내용 기술	2h	김치수
			EspressoMachine 클래스 코드 개발		이장훈
			User 클래스 코드 개발		이장훈
	singleton 패턴을 사용(Thread, 동기화 사용)	1	개념 내용 기술	3h	김치수
			EspressoMachine 클래스 코드 개발		이장훈
			User 클래스 코드 개발		이장훈
	singleton 패턴을 사용(처음부터 객체 생성)	1	개념 내용 기술	3h	김치수
			EspressoMachine 클래스 코드 개발		이장훈
			User 클래스 코드 개발		이장훈
	singleton 패턴(DCL 사용)	1	개념 내용 기술	3h	김치수
			EspressoMachine 클래스 코드 개발		이장훈
			User 클래스 코드 개발		이장훈

[2장 연습문제]

1. UML 모델에서 사용하는 구조 다이어그램에 속하지 않는 것은? ④

① Class Diagram
② Object Diagram
③ Component Diagram
④ Activity Diagram

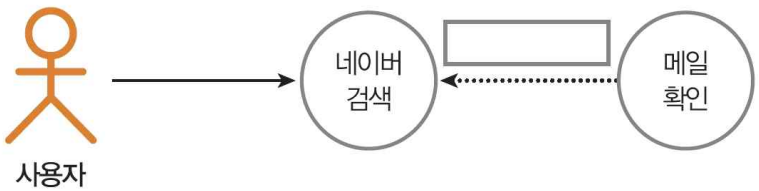
2. UML의 기본 구성 요소가 아닌 것은? ②

① Things
② Terminal
③ Relationship
④ Diagram

3. UML에서 활용되는 다이어그램 중 시스템의 동작을 표현하는 행위(behavioral) 다이어그램에 해당하지 않는 것은? ④

① 유스케이스 다이어그램(Usecase Diagram)
② 시퀀스 다이어그램(Sequence Diagram)
③ 활동 다이어그램(Activity Diagram)
④ 배치 다이어그램(Deployment Diagram)

4. 아래 그림에서 네모 칸에 들어갈 가장 적합한 것은? ①



① <<extend>>
② <<include>>
③ <<interface>>
④ <<log>>

5. UML에서 시퀀스 다이어그램의 구성 항목에 해당하지 않는 것은? ③

① 생명선
② 실행
③ 확장

④ 메시지

6. UML 확장 모델에서 스테레오 타입 객체를 표현할 때 사용하는 기호로 맞는 것은? ①

① << >>
② (( ))
③ { { } }
④ [[ ]]

7. 활동(activity) 다이어그램의 구성 요소로 옳지 않은 것은? ④

① 분기와 합류
② 동기화 막대
③ 신호
④ 전이

8. 다음은 UML 2.0 다이어그램에 대한 내용이다. [보기 1]과 [보기 2]를 바르게 연결한 것은? ②

[보기 1]

㉠ 시스템의 컴파일 시 계층적 구조를 기술하는 다이어그램이다.
㉡ 시퀀스 다이어그램과 활동 다이어그램을 혼합한 다이어그램이다.
㉢ 하나의 클래스 실행 시 내부 구조를 상세하게 표현하고자 할 때 사용하는 복합 구조를 표현하는 다이어그램이다.

[보기 2]

ⓐ 인터랙션 오버뷰 다이어그램(interaction overview diagram)
ⓑ 컴포지트 스트럭처 다이어그램(composite structure diagram)
ⓒ 패키지 다이어그램(package diagram)

㉠ ㉡ ㉢

① ⓐ ⓑ ⓒ
② ⓒ ⓐ ⓑ
③ ⓐ ⓒ ⓑ
④ ⓒ ⓑ ⓐ

9. 클래스(class) 다이어그램에 대한 설명으로 옳지 않은 것은? ③

① 개별 사례를 보고 공통점을 찾아 인식하는 확장형(extensional)이 있다.
② 일반적인 개념의 속성과 동작을 안에서 파악해 적용하는 내재형(intentional)이 있다.
③ 오퍼레이션이나 처리 과정이 수행되는 동안 일어나는 일을 단계적으로 표현하고자 할 때 사용하는 다이어그램이다.
④ 클래스 다이어그램은 도메인 개념과 속성 및 관계를 나타내는 중요한 모델이다.

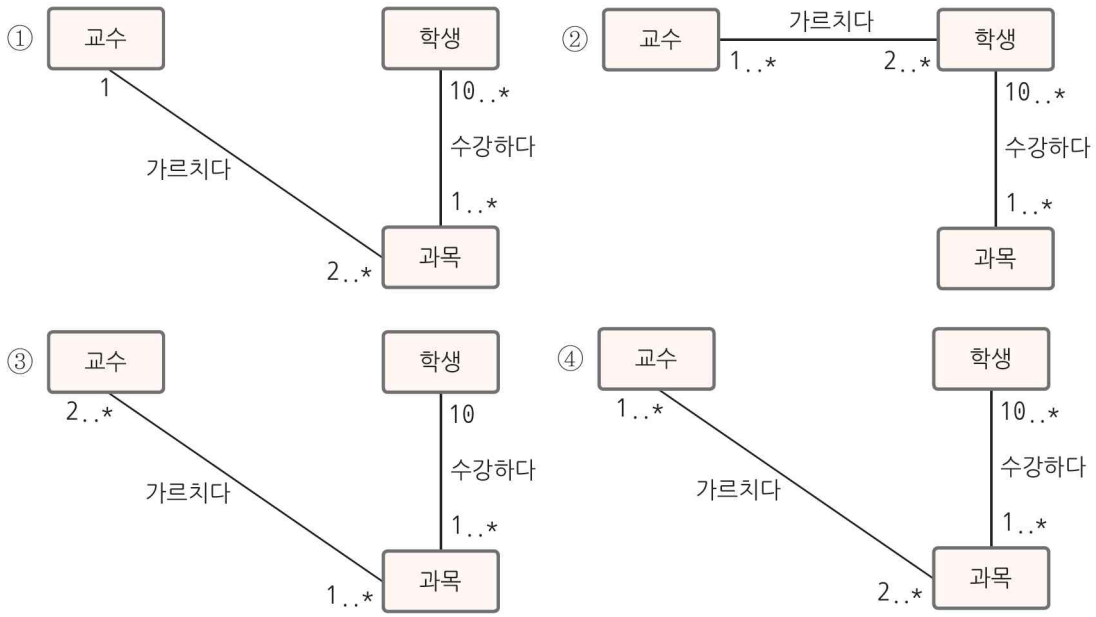
10. 유스케이스에 대한 설명으로 옳은 것만을 모두 고르면? ②

㉠ 개발자 관점에서 요구사항을 정의한다.
㉡ 액터는 시스템의 외부 대상이므로 개발 범위에 포함되지 않는다.
㉢ 유스케이스 명세서는 비기능 요구사항을 기술해서는 안 된다.
㉣ 유스케이스와 이를 이용하는 액터와의 관계는 연관 관계를 이용해 표현한다.

① ㉠, ㉢
② ㉡, ㉣
③ ㉠, ㉢, ㉣
④ ㉡, ㉢, ㉣

11. UML 클래스 다이어그램으로 가장 잘 나타낸 것은? ④

- '교수'는 적어도 두 '과목' 이상을 가르쳐야 한다.
- '과목'은 1명 이상의 '교수'가 가르쳐야 한다.
- '과목'은 10명 이상의 '학생'들이 수강해야 한다.
- '학생'은 한 '과목' 이상을 수강해야 한다.



12. UML 다이어그램의 설명이 옳지 않은 것은? ②

① 유스케이스(usecase) 다이어그램: 시스템의 기능을 모델링함
② 상태(state) 다이어그램: 클래스 사이의 메시지 교환을 시간의 흐름에 따라 표현함
③ 클래스(class) 다이어그램: 시스템의 정적인 구조를 나타냄
④ 활동(activity) 다이어그램: 시스템의 동작 특징을 나타냄

13. 도출된 유스케이스가 적절한지 나타내는 것으로 옳지 않은 것은? ④

① 수작업으로 이루어지는 것은 유스케이스가 될 수 없다.

② 액터가 원하는 최종 결과만 유스케이스로 도출했는지 확인한다.
③ 유스케이스가 액터에 의해 수행되는지 확인한다.
④ 유스케이스 내의 이벤트 흐름 일부를 사용해도 유스케이스가 될 수 있다.

14. 유스케이스를 찾는 방법으로 옳지 않은 것은? ④
① 시스템이 제공할 기능이 유스케이스가 될 수 있다.
② 사용자의 수작업 업무가 개발할 시스템의 유스케이스가 될 수 있다.
③ 데이터베이스의 등록/수정/삭제/조회 기능은 유스케이스가 될 수 있다.
④ 비기능 요소도 유스케이스가 될 수 있다.

15. 유스케이스 다이어그램에서 액터를 찾는 방법으로 옳지 않은 것은? ①
① 시스템을 관리하는 사람을 찾는다.
② 시스템에 자료를 등록/수정/삭제/조회하는 사람을 찾는다.
③ 시스템을 유지관리하는 사람을 찾는다.
④ 유스케이스의 기능 중 권한은 없지만 역할을 대신 해주는 사람을 찾는다.

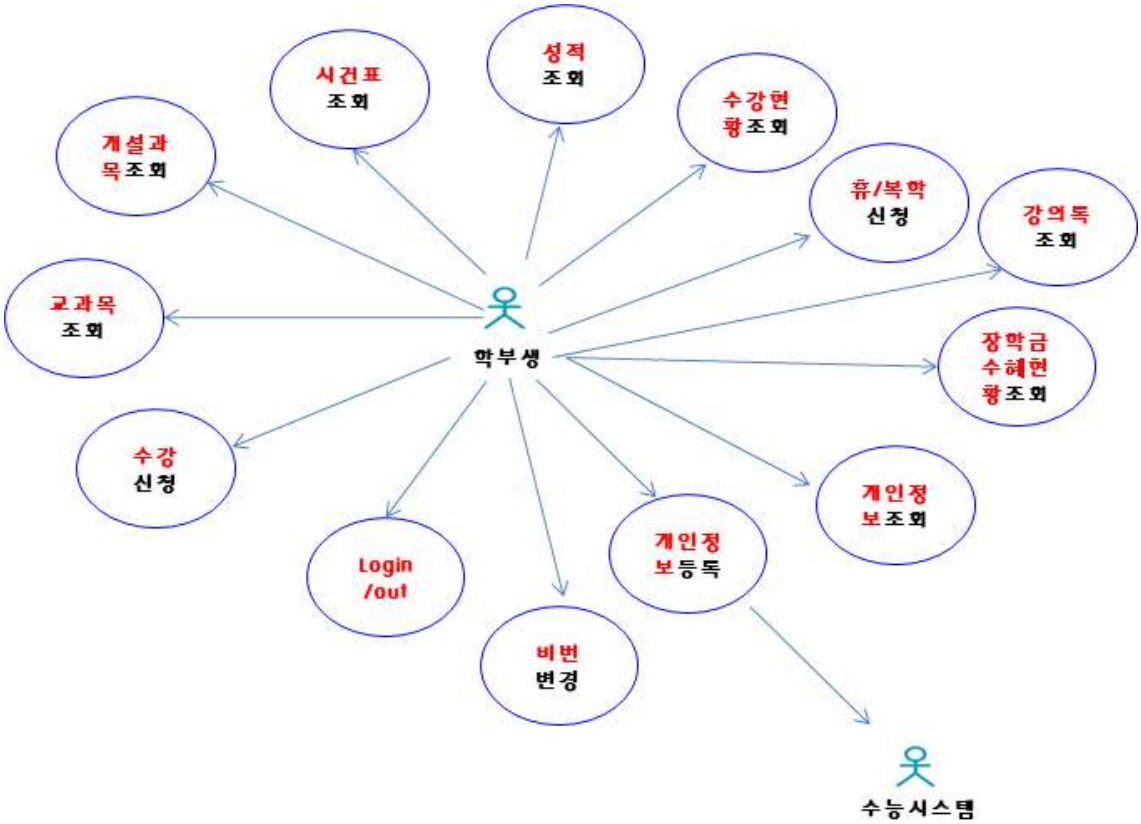
16. 학사관리시스템을 개발할 때 학생들이 사용하는 메뉴를 찾고 이를 유스케이스 다이어그램으로 작성하시오.

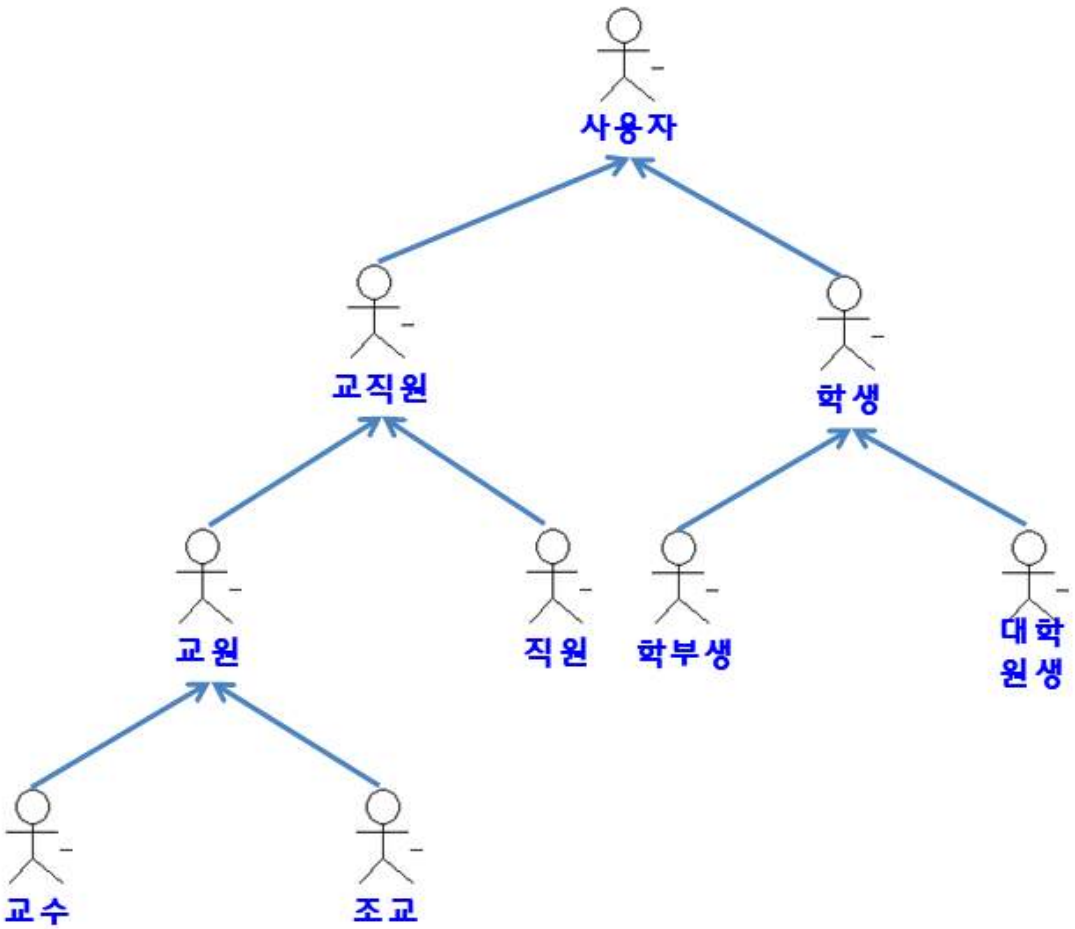
교과목조회, 개설과목조회, 수강신청, 수강현황조회, 휴/복학신청, 성적조회, 시간표조회, 강의록조회, 장학금수혜현황조회, 개인정보등록, 개인정보조회, 로그인/로그아웃, 비번 변경

17. 학사관리시스템을 개발할 때 '성적조회' 유스케이스에 대한 이벤트 흐름을 작성하시오.

학생 액터	시스템
학생은 학사관리 시스템이 접속한다.	
	시스템은 학사관리시스템 메뉴를 보여준다. ▶ 학생이 사용 가능한 메뉴만 보여준다.
학생은 '성적조회'메뉴를 선택한다.	
	시스템은 학생의 성적을 화면에 보여준다. 보여주는 항목은 다음과 같다. 학번, 이름, 과목명, 학점, 등급, 총학점

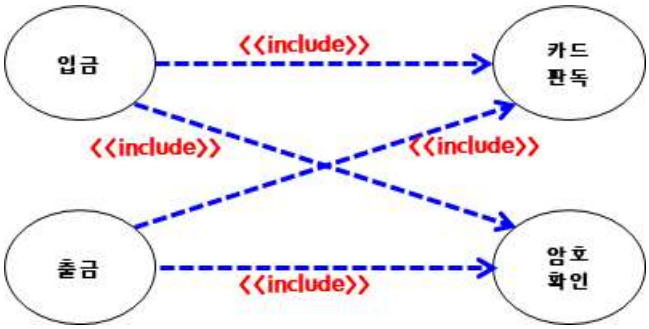
18. 액터의 일반화 관계에 대한 예를 만들어보시오.





19. 포함 관계와 확장 관계의 예를 각각 하나씩 만들어보시오.

[포함 관계]



[확장 관계]

20 순차 다이어그램의 예를 만들어보시오.

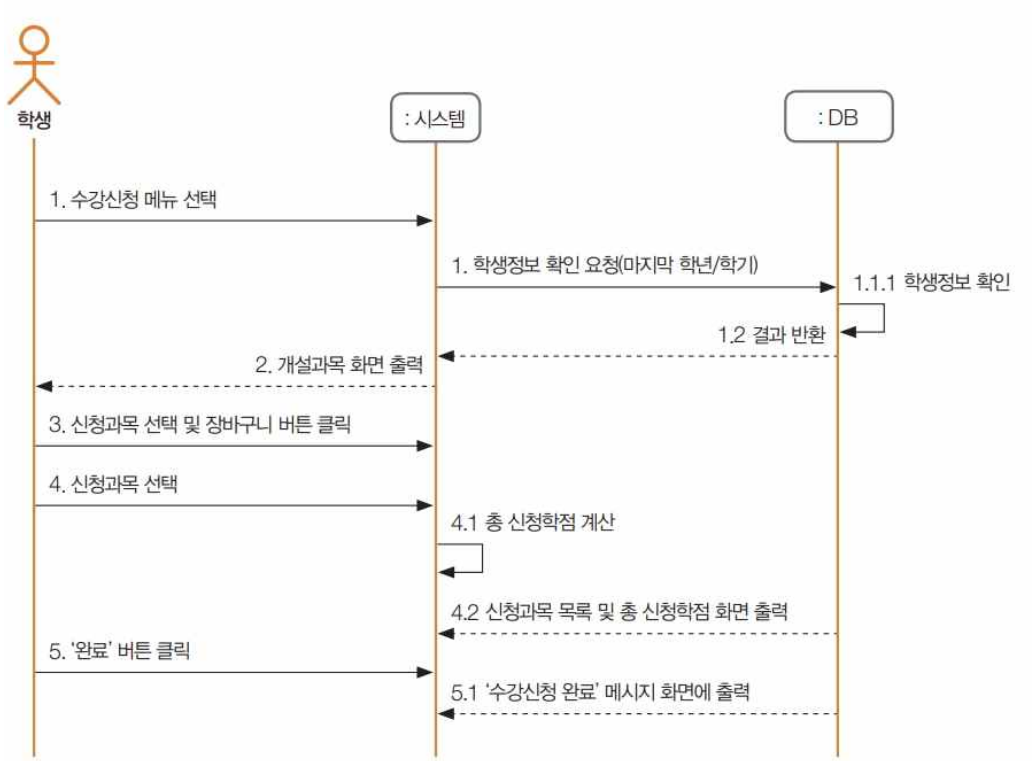


그림 2-28 '수강신청' 순차 다이어그램

[3장 연습문제]

1. LOC 기법에 의해 예측된 총 라인 수가 5만 라인, 프로그래머의 월평균 생산성이 2백 라인, 개발에 참여할 프로그래머가 10명일 때 개발 소요 기간은? ①
① 25개월
② 50개월
③ 200개월
④ 2,000개월

2. COCOMO 모델 중 기관 내부에서 개발된 중소 규모의 소프트웨어로 일괄 자료 처리나 과학기술 계산용, 비즈니스 자료 처리용으로 5만 라인 이하의 소프트웨어를 개발하는 유형은? ②
① embedded
② organic
③ semi-detached
④ semi-embedded

3. COCOMO 모델의 프로젝트 유형으로 거리가 먼 것은? ④
① Organic
② Semi-detached
③ Embedded
④ Sequential

4. 비용 예측을 위한 기능 점수 방법에 대한 설명 중 가장 옳지 않은 것은? ②
① 입력, 출력, 질의, 파일, 인터페이스의 개수로 소프트웨어의 규모를 표현한다.
② 기능 점수는 원시 코드의 구현에 이용되는 프로그래밍 언어에 종속적이다.
③ 경험을 바탕으로 단순, 보통, 복잡한 정도에 따라 가중치를 부여한다.
④ 프로젝트의 영향도와 가중치의 합을 이용해 실질 기능 점수를 계산한다.

5. 소프트웨어 프로젝트의 비용 결정 요소와 가장 관련이 적은 것은? ③
① 개발자의 능력
② 요구되는 신뢰도
③ 하드웨어의 성능
④ 개발 제품의 복잡도

6. 소프트웨어 비용 추정 모형(estimation models)이 아닌 것은? ④
① COCOMO

② Putnam
③ Function-Point
④ PERT

[참고]
- **Putnam 모형**: Rayleight-Norden 곡선의 노력 분포도를 이용한 비용 산정 방법
- **SLIM**: Putnam 모형을 기초로 해서 만든 자동화 도구

7. Rayleigh-Norden 곡선의 노력 분포도를 이용한 프로젝트 비용 산정 기법은? ①
① Putnam 방법
② 델파이 기법
③ COCOMO 방법
④ 기능 점수 방법

8. Putnam 방법을 기초로 해서 만든 자동화 추정 도구는? ②
① SQLR/30
② SLIM
③ MESH
④ NFV

[참고] **NFV**: 네트워크 기능 가상화(Network Function Virtualization)

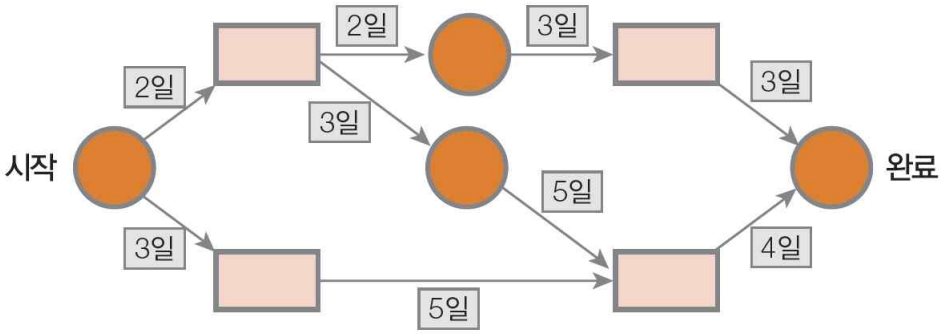
9. 기능 점수(Functional Point, FP) 방법에서 비용 산정에 이용되는 요소가 아닌 것은? ①
① 클래스 인터페이스
② 명령어(사용자 질의 수)
③ 데이터파일
④ 출력보고서

10. 10 기능 점수 FP를 계산하기 위해 고려할 대상으로 옳지 않은 것은? ④
① 외부 조회(EQ)
② 내부 논리 파일(ILF)
③ 외부 연계 파일(EIF)
④ 내부 출력(IO)

11. CPM(Critical Path Method)에 대한 설명으로 올바르지 않은 것은? ③
① CPM 네트워크는 노드와 간선으로 구성된 네트워크이다.
② CPM 네트워크는 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는 데 사용된다.

③ CPM 네트워크에서 작업의 선후 관계는 파악되지 않아도 무관하다.
④ CPM 네트워크를 효과적으로 사용하기 위해서는 필요한 시간을 정확히 예측해야 한다.

12. CPM 네트워크가 다음과 같을 때 임계 경로의 소요 기일은? ③



① 10일
② 12일
③ 14일
④ 16일

13. 다음 표는 프로젝트를 수행하는 데 필요한 작업, 소요 기간, 선행 작업을 나타낸 것이다. 작업 T5를 담당한 개발자가 이직해 대체 인력을 확보했으나 대체 인력의 교육에 15일이 소요되어 작업 T5의 소요 기간이 35일로 변경되었다. 프로젝트를 완료하기까지 필요한 최소 소요 기간은 개발자 이직 전보다 얼마나 증가하는가? ①

작업	소요 기간(일)	선행 작업
T1	10	-
T2	15	T1
T3	15	-
T4	10	T2, T3
T5	20	T3
T6	20	T5
T7	15	T4
T8	15	T5, T7

① 5일
② 10일
③ 15일
④ 35일

14. 소프트웨어 개발 비용에 영향을 주는 설명으로 가장 거리가 먼 것은? ④
① 프로그래머의 자질은 소프트웨어 개발 비용에 지대한 영향을 미친다.

② 소프트웨어의 복잡도는 개발 비용에 크게 영향을 준다고 할 수 있다.
③ 소프트웨어 규모도 비용 증가에 영향을 많이 준다.
④ 개발 기간을 단축하려면 인력과 자원을 늘리면 된다.

15. 다음 중 비용 산정 방법이 아닌 것은? ④
① 전문가 판단 기법
② 델파이 기법
③ 원시 코드 라인 LOC 기법
④ CPM 기법

16. 소프트웨어 개발에 대한 법적 타당성의 예를 찾아 설명하시오.

오픈소스(open source):
소프트웨어 개발에서 오픈소스를 사용하는 것은 비용 절감 측면에서 매우 효율적이다. 그런데 오픈소스를 사용할 때 보안 및 라이선스 위험을 대수롭지 않게 여겨 어려움을 겪는 경우를 흔히 볼 수 있다. 2018년 한 기업이 오픈소스 분쟁 소송으로 205만 달러(한화로 약 23억 원)를 지불하고 합의한 사례가 있다. 또 라이선스 위반율은 2017년 기준 30%가 넘을 정도로 심각한 수준이다.
오픈소스의 취지가 개방된 원시 코드를 자유롭게 수정해 사용할 수 있는 것인데 왜 이런 문제들이 발생할까? 사실 오픈소스는 원시 코드가 개방되어 있다는 것이지 아무렇게나 가져다 사용할 수 있는 것은 아니다. 오픈소스도 상용 소프트웨어처럼 저작권, 특허권과 같은 지식 재산권으로 보호받는 소프트웨어다. 따라서 법적인 문제가 발생하지 않으려면 오픈소스를 사용할 때 어디까지 무료로 사용할 수 있는지 확인해야 한다. 또 라이선스를 이용하기 위해 준수할 사항이 무엇인지 확인하고 그 범위 안에서 사용해야 한다. 결국 오픈소스를 문제없이 사용하려면 법적 타당성(legal feasibility)을 충분히 검토해야 한다.

17. 학사관리시스템을 개발할 때 간트 차트를 이용한 일정 계획표를 작성하시오.

ID	활동명	시작일	완료일	기간	2021년 1	2	3	4	5	6	7	8	9	10	11	12
1	학적부 관리	21.01.01	21.02.28	2												
2	학적 변동 관리	21.01.01	21.01.31	1												
3	교과 관리	21.03.01	21.05.31	3												
4	수업 관리	21.06.01	21.07.31	2												
5	수강관리	21.07.01	21.09.30	3												
6	성적 관리	21.10.01	21.12.31	3												

계획된 작업량 현재까지의 작업량

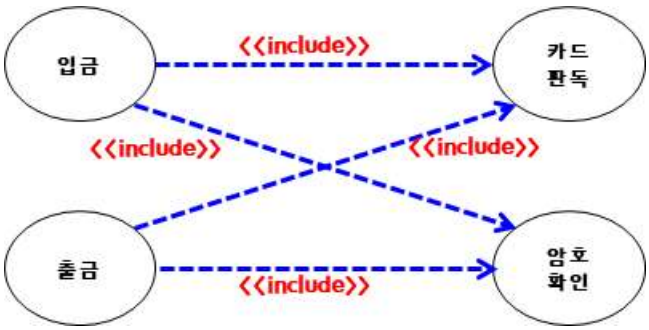
그림 3-15 간트 차트를 이용한 프로젝트 일정 계획표

18. 표에서 A5 개발자를 교체했다. 그런데 새 인력이 투입되기 위해 15일간의 교육이 필요하다. 따라서 A5의 소요 기간이 35일로 변경되었다. 모든 작업을 완료하기까지 필요한 최소 소요 기간은 개발자를 교체하기 전보다 얼마나 증가하는가? 5일

작업	소요 기간(일)	선행 작업
A1	10	-
A2	15	A1
A3	15	-
A4	10	A, A3
A5	20	A3
A6	20	A5
A7	15	A4
A8	15	A5, A7

19. 포함 관계와 확장 관계의 예를 각각 하나씩 만들어보시오.

[포함 관계]



[확장 관계]

// 2장 19번 문제(112쪽)와 중복되어 2쇄 도서부터는 문제 삭제 예정입니다.

20. 기능 항목에 대한 복잡도는 아래의 표와 같다. [표 1]과 같이 기능 항목에 대한 복잡도가 주어지고 입력은 10개, 출력은 5개, 질의는 8개, 파일은 28개, 응용 인터페이스는 4개이며, 복잡도는 단순이다. 개발팀의 생산성은 주당 60 기능 점수로 구현된다. [표 2]를 참조해 프로젝트를 수행하기 위한 노력을 추정한 값은?

3.33 persons-week

[표 1]

	항목	복잡도		
		단순	보통	복잡
1	입력(트랜잭션)	3	4	6
2	출력(화면/양식)	4	5	7
3	질의	2	4	6
4	파일	5	10	15
5	응용 인터페이스	4	7	10

[표 2]

특성	처리복잡도	비고
데이터 통신	5	0: 영향 없음
분산데이터 처리	2	1: 약간의 영향
처리속도	0	2: 어느 정도 영향
트랜잭션 처리	2	3: 보통 정도의 영향
하드웨어 이용	3	4: 중요한 영향
온라인 데이터 입력	0	5: 심각한 영향
온라인 갱신 정도	0	
사용자 편의성	2	
계산의 복잡성	2	
재사용성	2	
설치의 용이함	2	
운영의 용이함	0	
이식성	3	
유지 보수성	2	

[4장 연습문제]

1. 요구사항 분석에 필요한 기술로 가장 거리가 먼 것은? ③
① 청취와 인터뷰 질문 기술
② 분석과 중재 기술
③ 설계 및 코딩 기술
④ 관찰 및 모델 작성 기술

2. 소프트웨어 개발 방법 중 요구사항 분석(requirements analysis)과 거리가 먼 것은? ④
① 비용과 일정에 대한 제약 설정
② 타당성 조사
③ 요구사항 정의 문서화
④ 설계 명세서 작성

3. 요구 명세 기법에 대한 설명으로 틀린 것은? ②
① 비정형 명세 기법은 사용자의 요구를 표현할 때 자연어를 기반으로 서술한다.
② 비정형 명세 기법은 사용자의 요구를 표현할 때 Z 비정형 명세 기법을 사용한다.
③ 정형 명세 기법은 사용자의 요구를 표현할 때 수학적인 원리와 표기법을 사용한다.
④ 정형 명세 기법은 비정형 명세 기법에 비해 표현이 간결하다.

4. 공통 모듈에 대한 명세 기법 중 해당 기능에 대해 일관되게 이해하고 한 가지로 해석될 수 있도록 작성하는 원칙은? ②
① 상호작용성
② 명확성
③ 독립성
④ 내용성

5. 대학교 학사관리시스템의 기능 요구사항으로 옳지 않은 것은? ④
① 담당 교수는 자신이 맡은 과목의 성적을 입력 또는 수정한다.
② 학적관리자는 학생의 정보를 등록, 삭제할 수 있다.
③ 학생은 자신이 수강한 모든 과목의 성적을 조회할 수 있다 .
④ 시스템 장애로 인한 정지 시간이 1년에 10시간을 넘지 않아야 한다.

6. SoftTech사에서 개발된 것으로 구조적 요구분석을 위해 블록 다이어그램을 채택한 자동화 도구는?

④

① SREM
② PSL/PSA
③ HIPO
④ SADT

[참고]
• **SREM**(Software Requirements Engineering Methodology): RSL/REVS 이용, TRW사가 우주 국방 시스템 그룹에 의해 실시간 처리 요구사항을 명확히 기술할 목적으로 개발
• **PSL**(Problem Statement Language): 요구사항 기술 언어
• **PSA**(Problem Statement Analyzer): 미시간대학교에서 개발한 것으로 PSL로 기술한 요구사항을 분석해 보고서를 출력하는 자동화 도구
• **HIPO**(Hirearchy Input Process Output): 입력, 처리, 출력으로 구성된 하향식 소프트웨어 개발을 위한 문서화 도구로 시스템의 기능을 여러 고유 모듈로 분할해 이들 간의 인터페이스를 계층 구조로 표현
• **SADT**(Structured Analysis and Design Technique): SoftTech사에서 개발된 것으로 구조적 요구분석을 하기 위해 블록 다이어그램을 채택한 자동화 도구

요구사항 분석을 위한 자동화 도구(CASE 도구): 요구사항을 자동으로 분석하고, 요구분석명세서를 기술하도록 개발된 도구
• **RSL**(Requirement Statement Language): 요소(개체와 개념)/속성(수식)/관계/구조를 기술하는 요구사항 기술 언어
• **REVS**(Requirement Engineering and Validation System): RSL로 작성된 요구사항을 자동으로 분석해 분석 명세서 출력

TAGS(Technology for Automated Generation of System)
• 시스템 공학 방법 응용에 대한 자동 접근 방법
• 개발 주기 전 과정에 이용할 수 있는 통합 자동화 도구
• IORL 언어(요구사항 명세 언어)를 사용해 개발 주기 전 과정에서 이용할 수 있는 통합 자동화 도구

7. 자료 사전에서 자료의 생략을 의미하는 기호는? ④
① { }
② * *
③ =
④ ( )

8. 자료 흐름도(Data Flow Diagram, DFD)에 대한 설명으로 틀린 것은? ③
① 자료 흐름 그래프 또는 버블(bubble) 차트라고도 한다.
② 구조적 분석 기법에 이용된다.
③ 시간 흐름을 명확하게 표현할 수 있다.
④ DFD의 요소는 화살표, 원, 사각형, 직선(단선/이중선)으로 표시한다.

9. 자료 흐름도(DFD)의 구성요소에 포함되지 않는 것은? ④

① process
② data flow
③ data store
④ data dictionary

10. 럼바우의 객체지향 분석 절차를 가장 바르게 나열한 것은? ①

① 객체 모형 → 동적 모형 → 기능 모형
② 객체 모형 → 기능 모형 → 동적 모형
③ 기능 모형 → 동적 모형 → 객체 모형
④ 기능 모형 → 객체 모형 → 동적 모형

[참고] **제임스 럼바우**(James Rumbaugh)
객체 모델링 기법 OMT의 분석 모델(객체 모델 → 동적 모델 → 기능 모델)

11. 럼바우의 객체지향 분석 기법에서 동적 모델링에 활용되는 다이어그램은? ③

① 객체 다이어그램(Object Diagram)
② 패키지 다이어그램(Package Diagram)
③ 상태 다이어그램(State Diagram)
④ 자료 흐름도(Data Flow Diagram)

[참고] **제임스 럼바우**
정보 모델링 → Object Diagram
동적 모델링 → State Diagram
기능 모델링 → DFD

12. NS 차트(Nassi-Schneiderman chart)에 대한 설명으로 거리가 먼 것은? ③

① 논리의 기술에 중점을 둔 도형식 표현 방법이다.
② 연속, 선택 및 다중 선택, 반복 등의 제어 논리 구조로 표현한다.
③ 주로 화살표를 사용해 논리적인 제어 구조로 흐름을 표현한다.
④ 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는 데 적합하다.

[참고] **NS 차트**
- 프로그램의 처리 흐름을 박스 형태의 그림으로 표현
- 화살표가 없고 전체적인 논리 구조를 한 페이지에 표현할 수 있어 쉽게 이해 가능
- 입구와 출구가 하나이며 if-then-else/do-while/do-until 구조를 시각적으로 보기 쉽게 표현

13. HIPO(Hierarchy Input Process Output)에 대한 설명으로 거리가 먼 것은? ①

① 상향식 소프트웨어 개발을 위한 문서화 도구이다.
② HIPO 차트 종류에는 가시적 도표, 총체적 도표, 세부적 도표가 있다.
③ 기능과 자료의 의존 관계를 동시에 표현할 수 있다.
④ 보기 쉽고 이해하기 쉽다.

[참고] **HIPO 차트**
- 입력, 처리, 출력으로 구성된 하향식 소프트웨어 개발을 위한 문서화 도구
- 시스템의 기능을 여러 고유 모듈로 분할해 이들 간의 인터페이스를 계층 구조로 표현

HIPO 차트의 종류
- 가시적 도표(도식 목차(visual table of contents)): 전체의 기능 및 흐름을 보여주는 계층 구조도
- 총체적 도표(총괄 도표(overview diagram)): 입력, 처리, 출력에 대한 전반적인 정보를 제공하는 도표
- 세부적 도표(상세 도표(detail diagram)): 구성하는 기본 요소를 상세히 기술하는 도표

14. 객체지향 분석 방법론 중 E-R 다이어그램을 사용해 객체의 행위를 모델링하며 객체 식별, 구조 식별, 주체 정의, 속성 및 관계 정의, 서비스 정의 등의 과정으로 구성되는 것은? ①
① Coad와 Yourdon 방법
② Booch 방법
③ Jacobson 방법
④ Wirfs-Brocks 방법

[참고] **Coad와 Yourdon 방법**
- E-R 다이어그램을 이용해 개체의 활동을 데이터 모델링하는 데 초점을 둔 기법
- 객체 식별, 구조 식별, 주제 정의, 속성과 인스턴스 연결 정의, 연산과 메시지 연결 정의 등의 과정으로 주로 관계를 분석함

Booch 방법
- 미시적, 거시적 개발 프로세스를 모두 사용하는 분석 방법
- 클래스와 객체를 분석하고 식별해 클래스의 속성과 연산을 정의한 OOD(Object Oriented Design) 방법

Jacobson 방법
- 유스케이스 기반의 OOSE(Object Oriented Software Engineering)

Wirfs-Brocks 방법
- 분석, 설계의 구분이 명확하지 않고 요구사항을 평가해 설계 작업까지 수행하는 기법

※ objectory process(Jacobson) + OMT(Rumbaugh) + OOD(Booch) = UP → RUP

15. 어떤 소프트웨어의 고장 간 평균 시간(Mean Time Between Failures, MTBF)이 2,500시간이고 평균 수리 시간(Mean Time To Repair, MTTR)이 100시간일 때 평균 가동 시간(Mean Time To Failure, MTTF)과 가용성(availability)은?
2,400시간, 96%

16. 학사관리시스템을 개발할 때 비기능 요구사항의 예를 작성하시오.

- 수강신청할 수 있는 날짜는 학년별로 달라야 한다.
- 수강신청에서 동시 접속자 2,000명까지 1기가바이트 속도를 유지해야 한다.
- 프로그래밍언어는 파워빌더와 자바를 사용해야 한다.

17. 요구분석명세서 작성 시 주의사항 5가지를 쓰시오.

- 사용자가 쉽게 읽고 이해할 수 있도록 작성
- 개발자가 설계와 코딩에 효과적으로 사용할 수 있도록 작성
- 비기능 요구를 명확히 작성
- 테스트 기준으로 사용할 수 있도록 정량적으로 작성
- 품질에 대한 우선순위 명시

18. '사용자의 요구를 표현할 때 수학적 원리와 표기법을 이용하며 대표적으로 사용되는 것이 Z 정형 명세 언어'인 요구 명세 기법은?
정형 명세 기법

19. '요구분석명세서, 설계서, 구현코드에 대해 같은 내용을 쉽게 찾을 수 있게 되어 있는가?'에 해당하는 요구사항 검증 항목은?
추적 가능성

[5장 연습문제]

1. 바람직한 소프트웨어 설계 지침이 아닌 것은? ③
① 적당한 모듈의 크기를 유지한다.
② 모듈 간의 접속 관계를 분석해 복잡도와 중복을 줄인다.
③ 모듈 간의 결합도는 강할수록 바람직하다.
④ 모듈 간의 효과적인 제어를 위해 설계에서 계층적 자료 조직이 제시되어야 한다.

2. 효과적인 모듈 설계를 위한 유의사항으로 거리가 먼 것은? ③
① 모듈 간의 결합도를 약하게 하면 모듈 독립성이 향상된다.
② 복잡도와 중복성을 줄이고 일관성을 유지한다.
③ 모듈의 기능은 예측할 수 있어야 하며 지나치게 제한적이어야 한다.
④ 유지보수가 용이해야 한다.

3. 소프트웨어 설계 시 구축된 플랫폼의 성능 특성 분석에 사용되는 측정 항목이 아닌 것은? ④
① 응답시간(response Time)
② 가용성(availability)
③ 사용률(utilization)
④ 서버 튜닝(server tuning)

4. 소프트웨어의 상위 설계에 속하지 않는 것은? ②
① 아키텍처 설계
② 모듈 설계
③ 인터페이스 정의
④ 사용자 인터페이스 설계

5. 05 객체지향 설계가 갖는 특징으로 옳지 않은 것은? ②
① 객체지향 설계에서 중요한 것은 시스템을 구성하는 객체와 속성, 연산을 정의하는 것이다.
② 객체지향 설계는 하나의 커다란 작업을 여러 개의 작은 작업으로 분할하고 분할된 각각의 소작업을 함수(모듈)로 구현하는 것이다.
③ 객체지향 설계에서는 주어진 객체의 특성을 분석해 공통된 특징을 갖는 슈퍼 클래스를 생성하는 추상화 기법을 통해 객체의 설계 비용과 시간을 줄일 수 있다.
④ 객체지향 설계에서는 캡슐화를 통해 객체의 세부 내용 변경에 의해 발생할 수 있는 오류의 파급을 줄일 수 있다.

6. 객체지향 기법의 캡슐화(encapsulation)에 대한 설명으로 틀린 것은? ④

① 인터페이스가 단순화된다.
② 소프트웨어 재사용성이 높아진다.
③ 변경 발생 시 오류의 파급 효과가 작다.
④ 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것을 의미한다.

7. 객체지향 소프트웨어 공학에서 하나 이상의 유사한 객체를 묶어서 하나의 공통된 특성을 표현한 것은? ②

① 트랜지션
② 클래스
③ 시퀀스
④ 서브루틴

8. 소프트웨어 개발 프레임워크의 적용 효과로 볼 수 없는 것은? ②

① 공통 컴포넌트 재사용으로 중복 예산 절감
② 기술 종속으로 인한 선행 사업자 의존도 증대
③ 표준화된 연계 모듈 활용으로 상호 운용성 향상
④ 개발 표준에 의한 모듈화로 유지보수 용이

9. 응집도의 종류 중에서 서로 간에 어떠한 의미 있는 연관 관계도 지니지 않는 기능 요소로 구성되는 경우이며 서로 다른 상위 모듈에 의해 호출되어 처리상의 연관성이 없는 서로 다른 기능을 수행하는 경우의 응집도는? ④

① Functional Cohesion
② Sequential Cohesion
③ Logical Cohesion
④ Coincidental Cohesion

10. 시스템에서 모듈 사이의 결합도(coupling)에 대한 설명으로 옳은 것은? ③

① 한 모듈 내에 있는 처리요소들 사이의 기능적인 연관 정도를 나타낸다.
② 결합도가 높으면 시스템 구현 및 유지보수 작업이 쉽다.
③ 모듈 간의 결합도를 약하게 하면 모듈 독립성이 향상된다.
④ 자료 결합도는 내용 결합도보다 결합도가 높다.

11. CBD(Component Based Development)에 대한 설명으로 틀린 것은? ④

① 개발 기간 단축으로 인한 생산성 향상
② 새로운 기능 추가가 쉬운 확장성
③ 소프트웨어 재사용이 가능
④ 1960년대까지 가장 많이 적용되었던 소프트웨어 개발 방법

12. UI 설계 원칙에서 누구나 쉽게 이해하고 사용할 수 있어야 한다는 것은? ②
① 유효성
② 직관성
③ 무결성
④ 유연성

13. 객체지향의 개념에서 다형성(polymorphism)의 주된 설명으로 옳지 않은 것은? ①
① 속성과 관련된 오퍼레이션을 클래스 안에 묶어서 하나로 취급한다.
② 하나의 인터페이스에서 메소드를 데이터 타입 data type 및 파라미터 수를 변경해 재정의가 가능하다.
③ 하나의 인터페이스를 일관성 있게 사용 중심에서 제공할 수 있다.
④ 오버로딩(overloading), 오버라이딩(overriding)을 이용한 재사용성을 높일 수 있다.

14. 인터페이스 구현에 사용하는 기술 중에서 다음 내용이 설명하는 것은? ④
javaScript를 사용한 비동기 통신 기술로 클라이언트와 서버 간에 XML 데이터를 주고받는 기술

① Procedure
② Trigger
③ Greedy
④ AJAX

[참고]
• **Greedy**(탐욕법, 알고리즘 기법): 단계마다 지금 당장 가장 좋은 방법만을 선택하는 가장 쉽고 직관적인 알고리즘 설계 패러다임 중 하나
• **AJAX(**Asynchronous JavaScript and XML): 웹 서버와 비동기적으로 데이터를 교환하고 조작하기 위한 XML, XSLT, XMLHttpRequest로 Ajax 애플리케이션은 XML/XSLT 대신 미리 정의된 HTML이나 일반 텍스트, JSON, JSON-RPC를 이용할 수 있다.

15. 소프트웨어의 사용자 인터페이스 개발 시스템(User Interface Development System)이 가져야 할 기능이 아닌 것은? ④
① 사용자 입력의 검증
② 에러 처리와 에러 메시지 처리
③ 도움과 프롬프트(prompt) 제공
④ 소스 코드 분석 및 오류 복구

16. 아래의 문장은 객체지향 설계에 대한 잘못된 설명이다. 문장을 바르게 고치시오.
객체지향 설계는 하나의 커다란 작업을 여러 개의 작은 작업으로 분할하고, 분할된 각각의 소작업을 함수(모듈)로 구현하는 것이다.

객체지향 설계는 하나의 커다란 작업을 여러 개의 작은 작업으로 분할하고, 분할된 각각의 소작업을 클래스로 구현하는 것이다.

17. 아래의 문장에서 잘못된 부분을 바르게 고치시오.

전체 객체에 독립된 객체로 존재할 수 없는 부분 객체도 있는데 이와 같은 관계를 특별히 집합(aggregation) 관계라고 한다.

전체 객체에 독립된 객체로 존재할 수 없는 부분 객체도 있는데 이와 같은 관계를 특별히 합성 관계(composition relationship)라고 한다.

18. 다음 설명에 해당하는 응집력은? 순차적 응집

모듈 안에 있는 하나의 소작업에 대한 결과가 다른 작업의 입력이 되는 경우로 예를 들어 '다음 트랜잭션을 읽고 마스터 파일을 변경함'과 같은 모듈은 이것과 관계가 있다.

19. 학사관리시스템의 학생 사용 화면을 사용자 인터페이스 설계 지침을 고려해 만들어보시오.

정답 생략

20. 아래 프로그램은 두 정수를 입력받아 큰 값을 출력한다. 이 프로그램에서 caller 함수와 max 함수 간의 결합도와 max 함수의 응집도는 무엇인가?

데이터 결합, 기능적 응집

```
void caller(int a, int b) {
    int maxRooml = max(a, b);
    printf("%d", maxRoom);
}
int max(int c, int d) {
    if (c > d) return c;
    else return d;
}
```

[6장 연습문제]

1. 소프트웨어 아키텍처의 공통된 특징을 설명한 것 중 옳지 않은 것은? ④
① 개발할 소프트웨어에 대한 전체 구조를 다룬다.
② 구성 요소들이 인터페이스를 통해서 어떻게 상호작용하는지를 정의한다.
③ 설계 시 적용되는 원칙과 지침이 있어야 한다.
④ 세부내용도 자세히 다루는 것이 좋다.

2. 소프트웨어 아키텍처 설계 시 고려사항으로 적절하지 않은 것은? ②
① 의사소통 도구로 활용할 수 있어야 한다.
② 구현에 대한 제약 사항까지 정의할 필요는 없다.
③ 품질 속성을 결정해야 한다.
④ 재사용할 수 있게 설계해야 한다.

3. SEI에서 제시하는 SAiP(Software Architecture in Practice)의 품질 속성 분류로 관계가 가장 적은 것은? ④
① 시스템 품질 속성
② 비즈니스 품질 속성
③ 아키텍처 품질 속성
④ 설계 품질 속성

4. 시스템이 운용될 수 있는 확률로 사용 중인 시스템이 장애 발생 없이 서비스를 제공할 수 있는 능력을 나타내는 품질 속성은? ①
① 가용성
② 변경 용이성
③ 성능
④ 사용성

5. 학사관리시스템에 적합한 소프트웨어 아키텍처 스타일은? ②
① 클라이언트-서버 스타일
② 데이터 중심형 스타일
③ 계층 모델
④ 데이터 흐름 모델

6. 파이프 필터 형태의 소프트웨어 아키텍처에 대한 설명으로 옳은 것은? ②
① 노드와 간선으로 구성된다.

② 서브시스템이 입력 데이터를 받아 처리하고 결과를 다음 서브시스템으로 넘겨주는 과정을 반복한다.
③ 계층 모델이라고도 한다.
④ 3개의 서브시스템(모델, 뷰, 제어)으로 구성되어 있다.

7. 파이프 필터 형태의 소프트웨어 아키텍처에 대한 설명으로 옳은 것은? ③
① 컴포넌트 사이에 복잡한 상호작용이 필요한 시스템에 가장 적합하다.
② 사용자가 개입해 데이터 흐름을 전환할 경우에 사용된다.
③ 서브시스템이 입력 데이터를 받아 처리하고 결과를 다른 시스템에 보내는 작업이 반복된다.
④ 모든 필터가 동시에 작동하는 병렬처리 형식이다.

8. 다음 내용은 아키텍처 '4+1' 관점 중 어떤 것에 해당하는가? ③

물리적 시스템에서 사용하는 소프트웨어 서브시스템의 모듈들(원시 코드, 데이터파일, 컴포넌트, 실행 파일 등으로 구성)이 서로 어떤 연관 관계를 맺고 또 설계와 어떻게 연결 관계를 나타내는지에 관심이 있다.

① 시나리오 관점
② 물리적 관점
③ 개발 관점
④ 프로세스 관점

9. 소프트웨어 아키텍처의 4+1 관점에 대한 설명으로 옳지 않은 것은? ②
① 시나리오 관점에서는 외부 행위자에 의해 인식되는 시스템의 기능 요구사항을 보여주는 데 초점을 둔다.
② 논리적 관점에서는 계층 구조, 제약 사항, 코드 재사용 등과 같은 시스템 구현을 위한 요건을 보여주는 데 초점을 둔다.
③ 프로세스 관점에서는 독자적인 제어 스레드를 가질 수 있는 액티브 클래스에 초점을 둔다.
④ 물리적 관점에서는 물리적인 시스템을 구성하고 있는 부분의 분산 형태와 설치에 초점을 둔다.

10. 다음 설명에 해당하는 것은? ③

비슷한 유형의 응용 프로그램들을 위해 재사용이 가능한 아키텍처와 협력하는 소프트웨어 산출물의 통합된 집합으로, 특정 클래스의 재사용뿐만 아니라 응용 프로그램을 위한 핵심 아키텍처를 제공해 설계의 재사용을 지원한다.

① 컴포넌트(component)
② 웹서비스(web service)
③ 프레임워크(framework)
④ 클래스 라이브러리(class library)

11. 아키텍처 스타일과 이를 기반으로 하는 시스템의 관계를 짝지은 것으로 가장 관계가 적은 것은?
③
① 파이프 필터 구조 - 스프레드시트 시스템
② 계층 구조 - OSI 참조 모델
③ 클라이언트 서버 구조 - 인터넷 쇼핑몰
④ 저장소 구조 - 급여 시스템

12. MVC(Model-View-Controller) 아키텍처에서 Model의 역할로 옳은 것은? ③
① 이벤트 형태로 사용자 입력을 처리한다.
② 처리 결과 및 콘텐츠를 사용자에게 보여주는 기능을 수행한다.
③ 애플리케이션과 관련된 데이터 및 데이터 처리에 대한 로직을 가지고 있다 .
④ 최신 데이터를 가져와 표시된 정보를 갱신한다.

13. 소프트웨어 아키텍처를 명시적으로 설계하고 문서화를 통해서 얻을 수 있는 장점과 가장 관련이 적은 것은? ④
① 소프트웨어 개발 참여자 간의 의사소통 도구
② 시스템 분석
③ 대규모 재사용
④ 구현의 상세화

14. RUP의 4+1 관점을 나타내는 용어가 아닌 것은? ①
① logical view
② process view
③ implementation view
④ deployment view

15. 개발자와 시스템 통합자를 위한 것으로 실제 구동 환경을 살펴봄으로써 논리적 관점과 같이 시스템 내부의 구조(클래스 간의 관계, 클래스의 동작, 클래스 간의 상호작용)에 초점을 맞추는 관점은?
③
① 시나리오 관점
② 논리적 관점
③ 프로세스 관점
④ 개발 관점

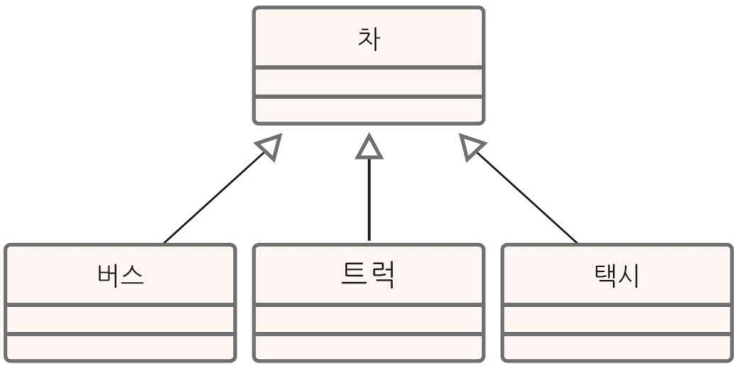
16. 객체지향 프로그램에서 데이터를 추상화하는 단위는? ②
① 메서드
② 클래스
③ 상속성

④ 메시지

17. 객체지향 기법에서 클래스들 사이의 '부분-전체(part-whole)' 관계 또는 '부분(is-a-part-of)'의 관계로 설명되는 연관성을 나타내는 용어는? ④

① 일반화
② 추상화
③ 캡슐화
④ 집단화

18. 아래의 UML 모델에서 '차' 클래스와 각 클래스의 관계로 옳은 것은? ③



① 추상화 관계
② 의존 관계
③ 일반화 관계
④ 그룹 관계

19. 객체지향 소프트웨어 공학에서 하나 이상의 유사한 객체를 묶어 하나의 공통된 특성으로 표현한 것은? ②

① 트랜지션
② 클래스
③ 시퀀스
④ 서브루틴

20. 다중성(multiplicity)의 기호에 대한 설명으로 옳지 않은 것은? ①

① 3..6: 3부터 6까지의 모든 숫자를 나타낸다.
② 2. 4. 6: 2 또는 4 또는 6을 나타낸다.
③ 1. 4..6: 1 또는 4 또는 5 또는 6을 나타낸다.
④ 0..1: 0 또는 1을 나타낸다.

21. 클래스 간의 관계에 대한 설명으로 옳지 않은 것은? ④

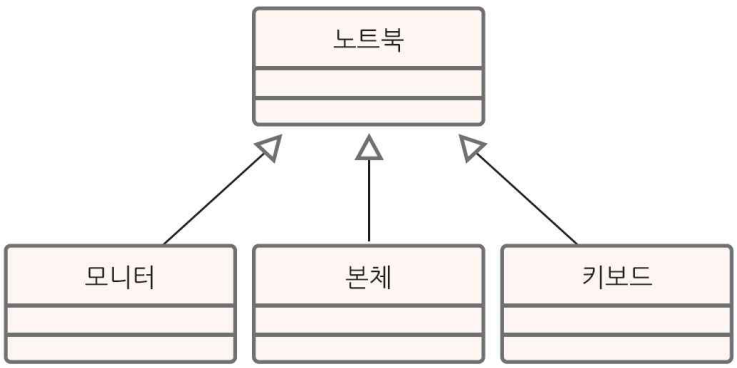
① 연관 관계의 다중성(multiplicity)은 두 클래스의 연관 관계에서 실제로 연관을 가지는 객체의 수를 나타낸다.
② 집합(aggregation) 관계는 전체와 그 객체의 구성요소 사이의 관계를 나타내며 whole-part 관계를 나타낸다.
③ 일반화(generalization) 관계는 일반적인 클래스와 구체적인 클래스 간의 관계, 즉 상속 개념을 나타내며 'is a kind of '의 관계를 나타낸다.
④ 의존 관계는 하나의 클래스에 있는 멤버 함수의 인자가 변해도 다른 클래스에 영향을 미치지 않는 관계를 의미한다.

22. 다음에 해당하는 용어로 적합한 것은? ③

모양은 상속 구조처럼 생겼지만 상속 관계를 맺을 수 없고 인터페이스 클래스는 스테레오타입으로 <<interface>>를 사용하며 하위 클래스와의 연관은 일반화 관계와 다르게 점선으로 나타낸다. 또 화살표의 머리 부분은 하위 클래스에서 상위 클래스로 향하고 있고 속이 빈 삼각형 모양이다.

① 추상화 관계
② 의존 관계
③ 실체화 관계
④ 그룹 관계

23. 다음의 클래스 다이어그램과 같은 관계에 해당하는 것은? ③



① 합성(composition) 관계
② 집합 관계
③ 실체화 관계
④ 일반화 관계

24. 상위 클래스의 객체가 들어갈 자리에 하위 클래스의 객체를 넣어도 문제없이 잘 작동함을 나타내는 클래스 설계 원칙은? ④

① 인터페이스 분리 원칙
② 의존 관계 역전 원칙
③ 개방 폐쇄 원칙
④ 리스코프 교체 원칙

25. 다음 내용이 설명하는 객체지향 설계 원칙은? ①

- 클라이언트는 자신이 사용하지 않은 메서드와 의존 관계를 맺으면 안 된다.
- 클라이언트는 사용하지 않은 인터페이스 때문에 영향을 받아서는 안 된다.

① 인터페이스 분리 원칙
② 단일 책임 원칙
③ 개방 폐쇄 원칙
④ 리스코프 교체 원칙

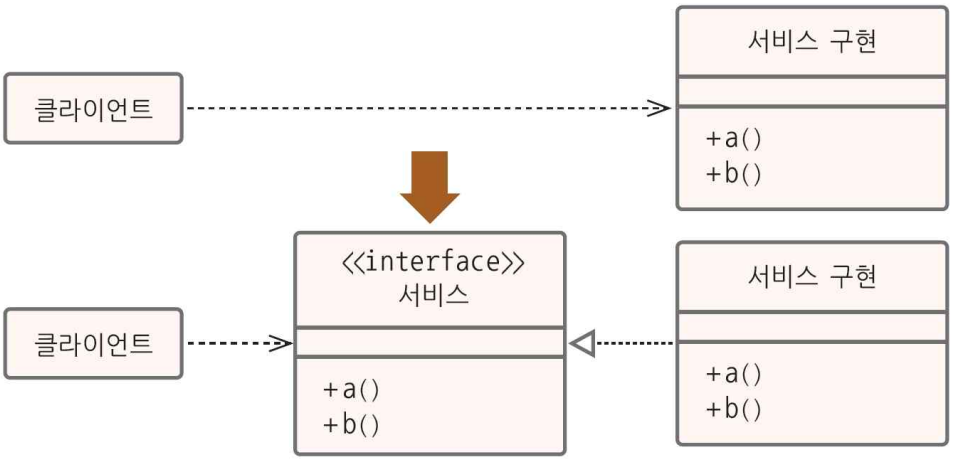
26. 객체지향 설계 원칙 중에서 서브 타입(상속받은 하위 클래스)은 어디에서나 자신의 기반 타입(상위 클래스)으로 교체할 수 있어야 함을 의미하는 원칙은? ③

① ISP(Interface Segregation Principle)
② DIP(Dependency Inversion Principle)
③ LSP(Liskov Substitution Principle)
④ SRP(Single Responsibility Principle)

27. 클래스 설계 원칙에 대한 설명으로 옳은 것은? ②

① 인터페이스 분리 원칙(Interface Segregation Principle): 추상 클래스나 인터페이스에 의존하지 않고 자주 변경되는 클래스에 의존해야 한다.
② 개방 폐쇄 원칙(Dependency Inversion Principle): 클래스는 확장(extension)에 대해 열려 있어야 하고 변경(change)에 대해서는 닫혀 있어야 한다.
③ 리스코프 교체 원칙(Liskov Substitution Principle): 여러 개의 책임을 가진 클래스는 하나의 책임을 가진 클래스로 대체되어야 한다.
④ 의존 관계 역전 원칙(Dependency Inversion Principle): 클라이언트는 자신이 사용하는 메서드와 의존 관계를 갖지 않도록 해야 한다.

28. 그림과 같이 서비스 구현 클래스의 a( )와 b( ) 연산을 사용하는 클라이언트 클래스가 서비스 구현 클래스에 직접 의존하는 관계에서 클라이언트 클래스가 서비스 인터페이스에 의존하고 서비스 구현 클래스는 서비스 인터페이스를 구현하는 것으로 설계를 변경했다. 다음 중 이와 가장 관련이 깊은 SOLID 설계 원칙은? ③



① 단일 책임 원칙(Single Responsibility Principle)
② 리스코프 교체 원칙(Liskov Substitution Principle)
③ 의존 관계 역전 원칙(Dependency Inversion Principle)
④ 인터페이스 분리 원칙(Interface Segregation Principle)

29. 다음 중 클린(clean) 코드 작성 원칙으로 거리가 먼 것은? ②
① 누구든지 쉽게 이해하는 코드 작성
② 중복이 최대화된 코드 작성
③ 다른 모듈에 미치는 영향 최소화
④ 단순 명료한 코드 작성

30. 다음에서 설명하는 클린 코드 작성 원칙은? ②

- 한 번에 한 가지 처리만 수행한다.
- 클래스/메서드/함수를 최소 단위로 분리한다.

① 다형성
② 단순성
③ 추상화
④ 의존성

[참고] **클린 코드(clean code)**
- 읽기 쉬운 코드, 즉 모든 팀원이 이해(understandability)하기 쉽도록 작성된 코드
- 클래스 설계 관점의 클린 코드: SRP, OCP, LSP, ISP, DIP

31. 아키텍처의 필요성에 대해 간략히 작성하시오.

복잡하고 규모가 큰 소프트웨어를 개발하려면 전체 구조가 유기적으로 잘 구성되어야 한다. 또한 사용자가 만족할 만한 품질 좋은 소프트웨어를 개발하려면 요구분석과 설계 단계부터 품질 특성을 고려해서 개발해야 한다. 따라서 잘 정의된 구조의 품질 좋은 소프트웨어를 만들려면 소프트웨어 아키텍처가 필요하다. 아키텍처 설계로 소프트웨어가 어떤 구조이고 어떻게 동작할 것인지를 예측할 수 있으며 변경에 유연

하게 대처할 수 있다.

32. 소프트웨어 아키텍처의 공통된 특징을 간략히 작성하시오.

소프트웨어 아키텍처는 소프트웨어의 골격이 되는 기본 구조로 시스템 전체에 대한 큰 밑그림이다. 소프트웨어 아키텍처의 특징은 다음과 같다.
- 소프트웨어의 골격을 나타내는 추상화된 전체 구조를 제공한다.
- 소프트웨어를 이루고 있는 여러 구성 요소(서브시스템, 컴포넌트)를 다룬다.
- 인터페이스를 통해 소프트웨어의 구성 요소가 어떻게 상호작용하는지를 정의한다.
- 세부 내용보다는 중요 내용(설계자(architect)가 주관적으로 판단하고 결정한 내용)만 다룬다.
- 설계에 적용되는 원칙과 지침이 있다

33. 시스템 품질 속성 중에서 '사용성'을 높일 수 있는 예를 작성하시오.

시스템을 사용할 때 발생할 수 있는 여러 가지 상황을 극복할 수 있도록 이를 반영해 아키텍처 설계를 해야 한다. 예를 들어 시스템의 도움말 기능은 사용자에게 실제 도움을 줄 수 있어야 하고 사용자가 원치 않는 상황에 놓인 경우에도 친숙한 사용자 인터페이스를 제공해 당황하지 않도록 설계해야 한다. 또한, 사용자의 실수에도 오류의 영향을 최소화하도록 되돌리기(undo)나 취소(cancel) 기능을 제공해야 한다. 이처럼 아키텍처 설계 시에는 사용자가 시스템을 사용할 때 불편을 느끼지 않도록 사용성이 반영되어야 한다.

34. MVC 모델의 장점을 간략히 작성하시오.

MVC 스타일은 사용자 인터페이스에 해당하는 뷰와 데이터 및 데이터 처리 로직에 해당하는 모델을 독립적으로 분리해 변경에 대한 영향이 덜 미치도록 한 것이 장점이다. 이렇게 하면 사용자 인터페이스가 자주 변경되더라도 모델에는 영향을 주지 않는다. 이처럼 약한 결합으로 설계하면 서로 영향을 덜 미치기 때문에 구조 변경을 요청할 때 수정하기가 쉽다.

35. 아키텍처 스타일을 이해할 수 있는 일상에서의 예를 만들어보시오.

저녁 식사에 손님을 초대하면

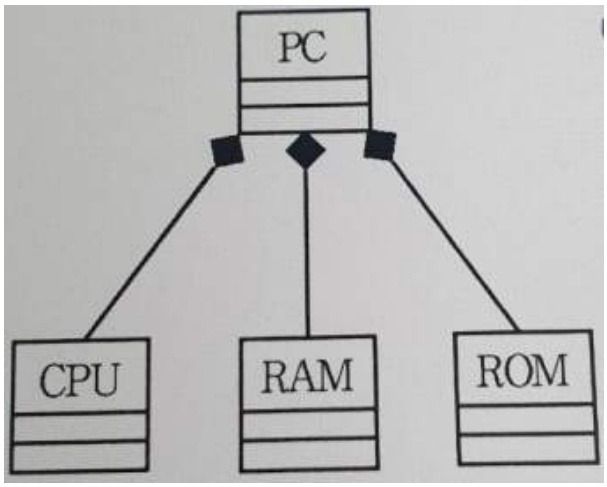
어떤 종류(Style)의 음식을 대접할 것인지 고민한다.
- 종류를 결정(한식, 일식, 중식 등)

종류(style)가 결정되면
- 그에 맞는 재료가 결정된다.
- 조리 방법이 결정된다.
- 음식 담을 그릇이 결정 된다.

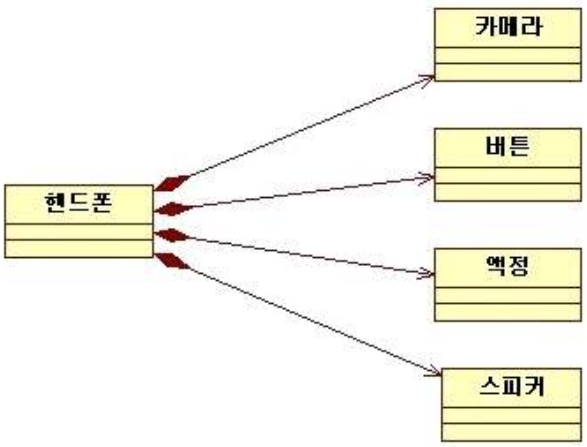
∴ architecture style이 결정되면
- SW 특성과 전체 구조를 알 수 있다.
- 개발 방법도 알 수 있다.

36. 아래 프로그램에 대한 클래스 다이어그램을 작성하시오.

```
class PC {
      private CPU cpu;
      private RAM[] rams;
      private ROM rom;
      public PC(){
            this.cpu = new CPU();
            this.rams = new RAM[2];
            this.rom = new ROM();
      }
……
}
```



37. 포함(composition) 관계의 예를 찾아 다이어그램 형태로 작성하시오(단, 노트북 예는 제외).

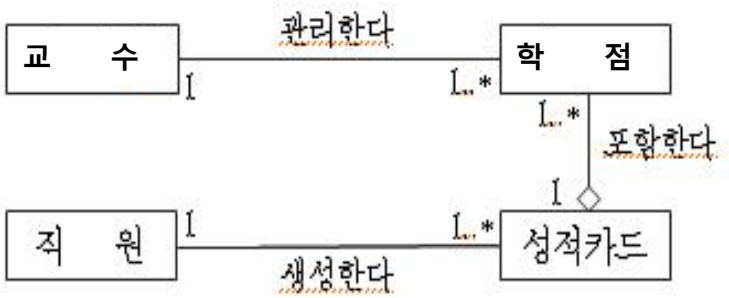


38. 다음 설명에 적합한 용어는? method overriding

- 원(circle)의 면적을 구하는 getArea( ) 함수를 가진 객체
- 사각형(rectangle)의 면적을 구하는 getArea( ) 함수를 가진 객체
- 삼각형(triangle)의 면적을 구하는 getArea( ) 함수를 가진 객체
- 원, 사각형, 삼각형은 getArea( ) 함수를 가지고 있으므로 면적을 구하려면 getArea( ) 메시지를 받으면 수행된다. 그러나 각각의 함수에서 면적을 구하는 방법은 모두 다를 것이다.

39. 다음 사항을 가장 잘 표현한 클래스 다이어그램은?

- '교수'는 적어도 한 과목의 '학점'을 관리한다.
- '학점'은 1명의 '교수'로부터 관리된다.
- '학점'은 하나의 '학점 카드'에 포함된다.
- '학점 카드'는 적어도 하나 이상의 '학점'을 포함한다.
- '직원'은 적어도 하나 이상의 '학점 카드'를 생성한다.
- '학점 카드'는 1명의 '직원'에 의해서만 생성된다.



40. 아래의 프로그램에서 ① ~ ④ 문장을 수행할 때 오류가 발생하지 않는 것을 모두 고르시오.

②, ③

```
abstract class A {
}
class B extends A {
}
...
A inst1 =new A();       // ①
A inst2 =new B();       // ②
B inst3 =new B();       // ③
B inst4 =new A();       // ④
```

[7장 연습문제]

1. 디자인 패턴 사용의 장점과 단점에 대한 설명으로 거리가 먼 것은? ④
① 소프트웨어 구조 파악이 용이하다.
② 객체지향 설계 및 구현의 생산성을 높이는 데 적합하다.
③ 재사용을 위한 개발 시간이 단축된다.
④ 절차형 언어와 함께 이용될 때 효율이 극대화된다.

2. GoF(Gang of Four) 디자인 패턴 분류에 해당하지 않는 것은? ④
① 생성 패턴
② 구조 패턴
③ 행위 패턴
④ 추상 패턴

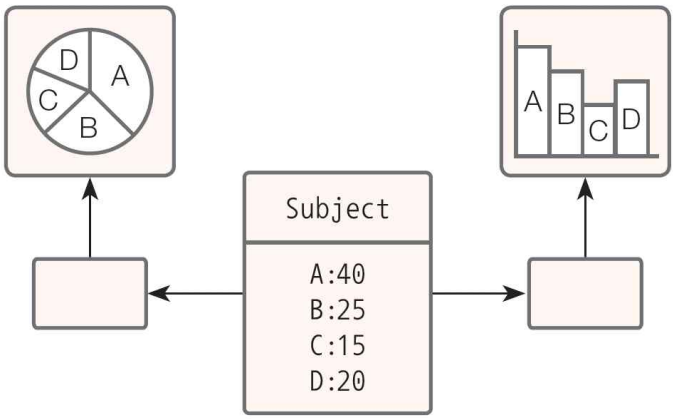
3. 다음 내용이 설명하는 디자인 패턴은? ③
• 객체를 생성하기 위한 인터페이스를 정의하며 어떤 클래스가 인스턴스화될 것인지는 서브 클래스가 결정하도록 하는 것
• Virtual-Constructor 패턴이라고도 함

① visitor 패턴
② observer 패턴
③ factory method 패턴
④ bridge 패턴

4. 객체지향 소프트웨어 설계 시 디자인 패턴을 구성하는 요소로 가장 거리가 먼 것은? ①
① 개발자 이름
② 문제 및 배경
③ 사례
④ 샘플 코드

5. 디자인 패턴 중에서 행위적 패턴에 속하지 않는 것은? ③
① command 패턴
② observer 패턴
③ prototype 패턴
④ state 패턴

6. 그림과 같이 관찰 대상(subject)의 데이터(A~D)에 변화가 발생하면 이 변화를 탐지해 여러 가지 방식으로 사용자에게 디스플레이하는 프로그램을 작성하고자 한다. 이 프로그램에 적용할 수 있는 디자인 패턴은? ④

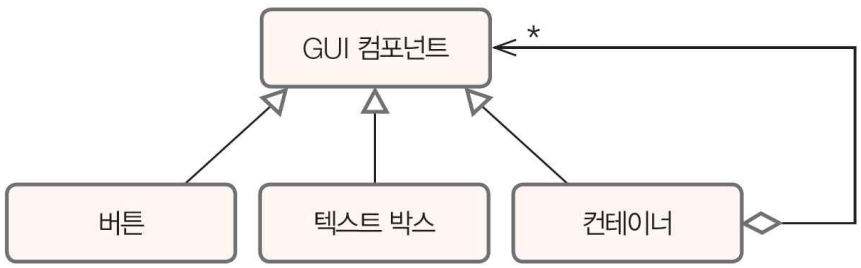


① decorator 패턴
② flyweight 패턴
③ mediator 패턴
④ observer 패턴

7. GoF의 디자인 패턴에서 행위 패턴에 속하는 것은? ②
① builder
② vsitor
③ prototype
④ bridge

8. 차량 내비게이션 소프트웨어에서 GPS 신호를 수신하는 경우와 수신하지 못하는 경우에 따라 차량의 위치를 구하는 다른 알고리즘을 선택할 때 가장 적합한 설계 패턴은? ④
① decorator 패턴
② adapter 패턴
③ composite 패턴
④ strategy 패턴

9. 다음 클래스 다이어그램으로 표현한 설계에서 사용한 디자인 패턴은? ②



① adapter 패턴

② composite 패턴
③ observer 패턴
④ factory Method 패턴

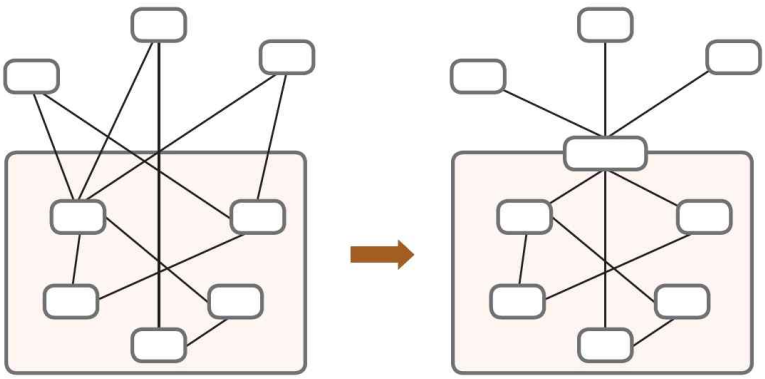
10. [보기 1]의 디자인 패턴 분류와 [보기 2]의 디자인 패턴을 바르게 연결한 것은? ②

[보기 1]
㉠ 생성 패턴
㉡ 구조 패턴
㉢ 행위 패턴

[보기 2]
ⓐ bridge 패턴
ⓑ singleton 패턴
ⓒ interpreter 패턴

	㉠	㉡	㉢
①	ⓐ	ⓑ	ⓒ
②	ⓑ	ⓐ	ⓒ
③	ⓑ	ⓒ	ⓐ
④	ⓒ	ⓐ	ⓑ

11. 그림과 같이 서브시스템 사이의 의사소통 및 종속성을 최소화하기 위해 단순화된 하나의 인터페이스를 제공하는 디자인 패턴은? ④



① adapter 패턴
② bridge 패턴
③ decorator 패턴
④ facade 패턴

12. 다음에서 설명하는 디자인 패턴에 해당하는 것은? ②

이미 만들어져 있는 클래스를 사용하고 싶지만 인터페이스가 원하는 방식과 일치하지 않을 때 또는 관련성이 없거나 예측하지 못한 클래스와 협동하는 재사용 가능한 클래스를 생성하기 원할 때 사용한다.

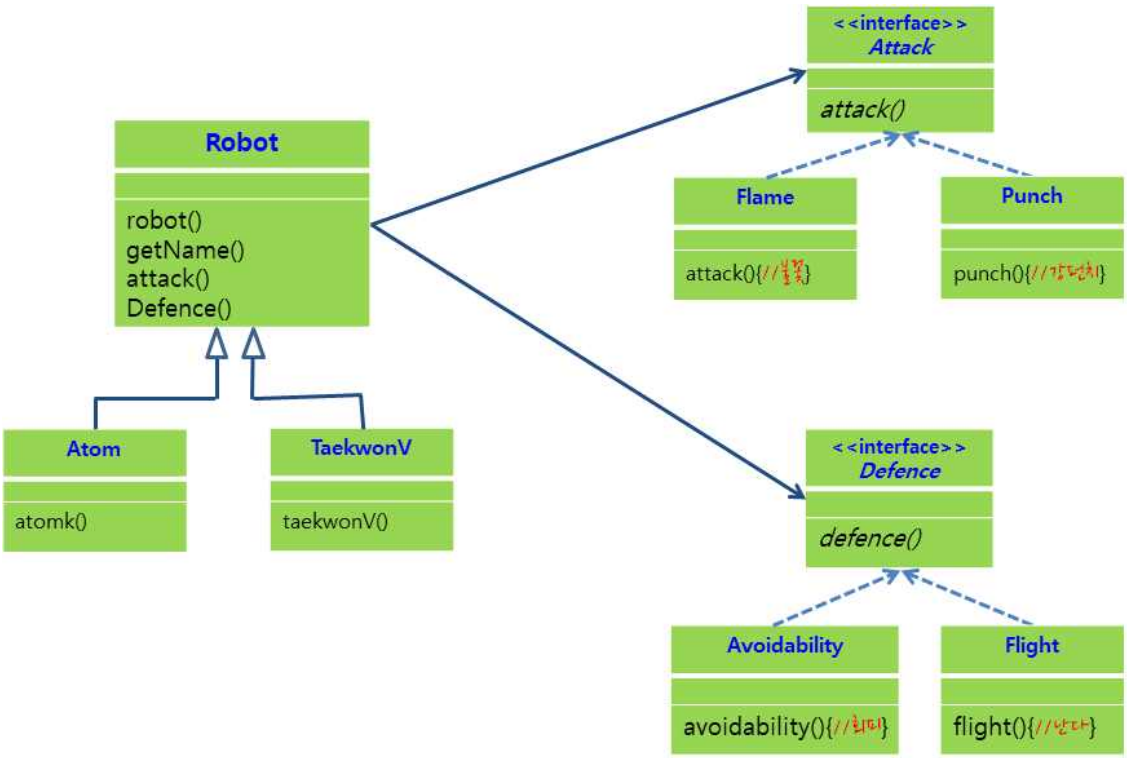
① bridge 패턴
② adapter 패턴
③ composite 패턴
④ facade 패턴

13. 디자인 패턴에 대한 설명으로 옳지 않은 것은? ③
① observere 패턴: 어떤 객체의 상태가 변할 때 그 객체에 의존성을 가진 다른 객체들이 그 변화를 통지받고 자동으로 갱신될 수 있게 만든다.
② mediator 패턴: 객체의 상호작용을 캡슐화하는 객체를 정의한다.
③ composite 패턴: 연산을 적용할 원소의 클래스를 변경하지 않고도 새로운 연산을 정의할 수 있게 한다.
④ bridge 패턴: 구현에서 추상을 분리해 이들이 독립적으로 다양성을 가질 수 있도록 한다.

14. 다음 설명에 해당하는 소프트웨어 설계 패턴은? ①
일 대 다의 객체 의존 관계를 정의하며 한 객체의 상태가 변화되었을 때 의존 관계에 있는 다른 객체에게 자동으로 변화를 통지한다.
① observer 패턴
② factory method 패턴
③ decorator 패턴
④ strategy 패턴

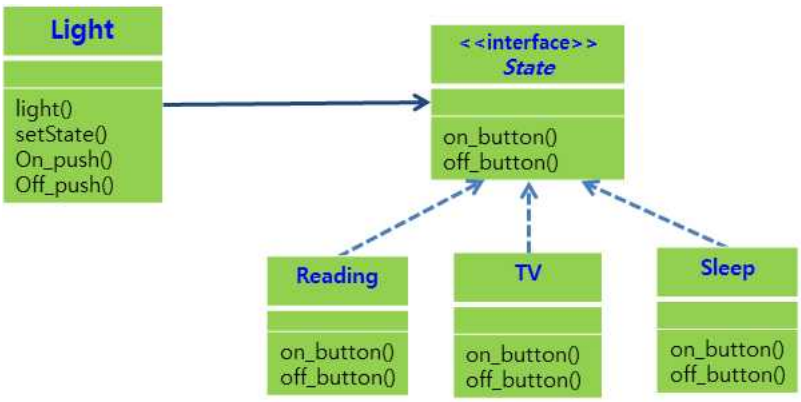
15. 다음 중 성격이 다른 설계 패턴은? ①
① bridge 패턴
② factory method 패턴
③ prototype 패턴
④ singleton 패턴

16. 내용에 대해 전략(strategy) 패턴을 사용한 클래스 다이어그램을 작성하시오.
현재 로봇(Robot)은 아톰(Atom)과 태권브이(TaekwonV) 2가지 종류가 있다. 그리고 로봇의 기능도 공격(Attack)과 방어(Defence) 기술을 가지고 있다. 아톰의 공격 기술은 불꽃(Flame), 방어 기술은 날기(Flight)이다. 태권브이의 공격 기술은 강펀치(Punch), 방어 기술은 회피(Avoidability)이다. 로봇의 종류는 계속 추가 및 삭제될 수 있고 공격과 방어 기술도 추가 및 삭제될 수 있다.



17. 다음 내용에 대해 상태(state) 패턴을 사용한 클래스 다이어그램을 작성하시오.

사람 손의 터치로 상태가 바뀌는 조명이 있다. 꺼짐(Off) 상태에서 한 번 터치하면 독서 상태로 바뀌고 독서 상태에서 터치하면 TV 시청 상태로 바뀐다. TV 시청 상태에서 터치하면 수면 상태로 바뀌고 수면 상태에서 터치하면 꺼짐 상태로 바뀐다.



18. decorator 패턴의 예를 만들고 이 패턴을 적용한 클래스 다이어그램을 작성하시오.

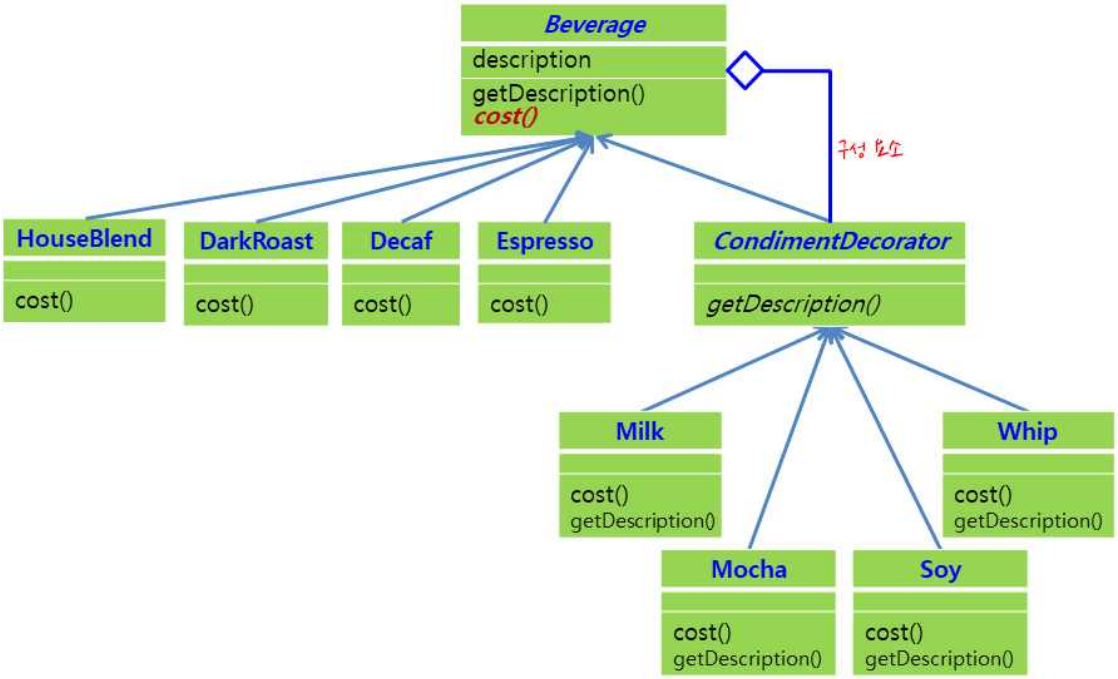
['커피숍주문시스템'의 요구사항]

- 커피숍에는 여러 종류의 음료(beverage)가 있다.
- 커피는 HouseBlend, DarkRoast, Decaf, Espresso...등 수많은 종류가 있고 계속 늘어난다.
- 커피숍 주문 시스템의 핵심 기능은 여러 종류의 음료에 대해 가격(cost)을 계산하는 것이다.

[고객이 원하는 커피는... 수십 종의 커피가 존재...]

{커피+우유}, {커피+두유}, {커피+모카(초콜릿) },

{커피+우유+휘핑 크림}, {커피+두유+휘핑 크림}, {커피+모카(초콜릿)+ 휘핑 크림}...



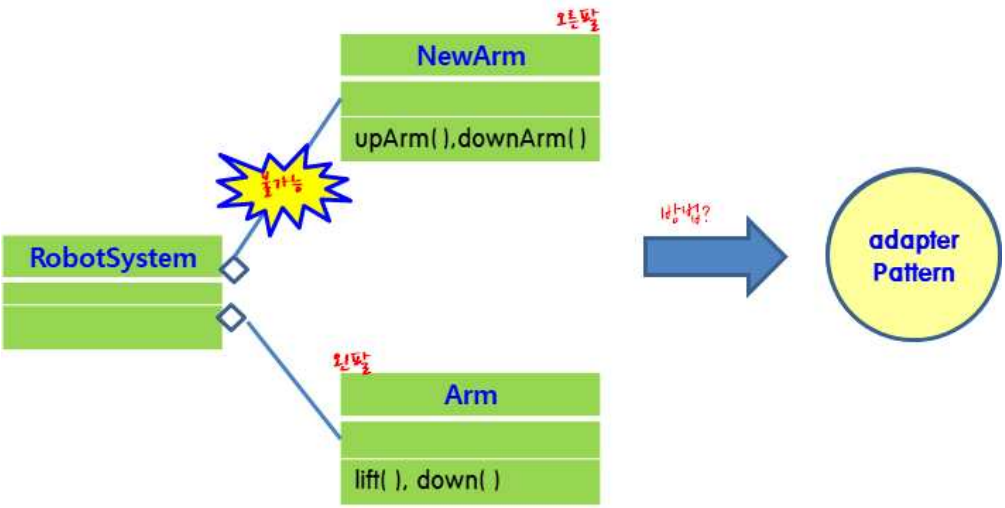
19. adapter 예를 만들어 이 패턴을 적용한 클래스 다이어그램을 작성하시오.

['로봇시스템 구축'의 요구사항]

- 로봇시스템을 개발한다. 왼쪽 팔은 기존에 개발된 로봇 팔을 그대로 사용한다. 오른쪽 팔은 시간 관계상 최신 로봇 팔 컴포넌트를 구입하여 사용한다.

[문제점]

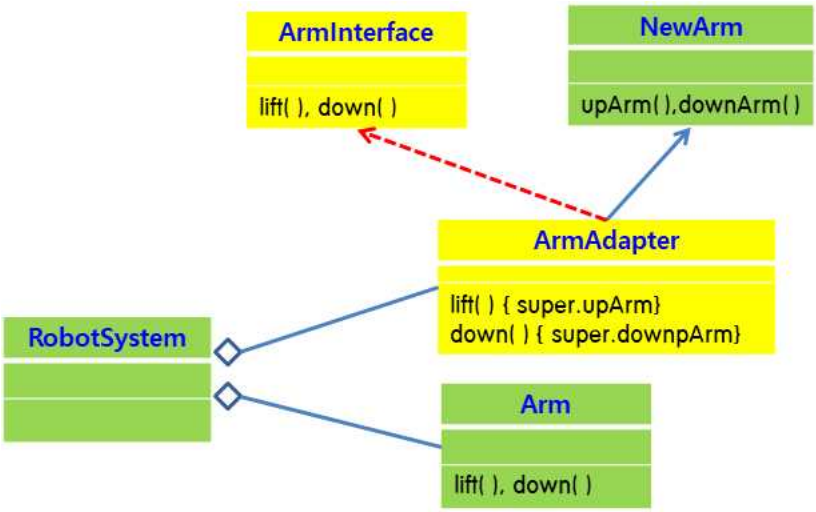
구입한 로봇 팔이 기존 인터페이스를 제공하지 않는다. 그리고 기존 로봇 팔의 올림/내림 인터페이스는 lift(), down()을 사용하는데 구입한 로봇 팔의 올림/내림 인터페이스는 upArm(), downArm() 메서드를 사용한다.



[해결안]

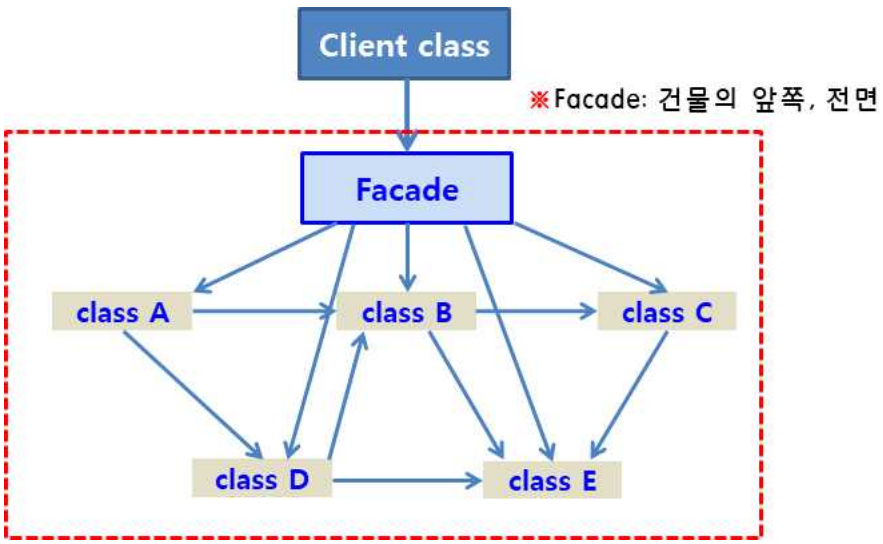
새 로봇 팔을 상속받고 interface를 실현하여 새 로봇 팔도 표준 API를 가짐.

lift(): {super.upArm} 로봇 팔 올림, Down(): {super.downArm} 로봇 팔 내림



20. facade 패턴의 필요성과 장점 및 단점을 설명하시오.

■ **facade 패턴의 필요성**



facade 패턴은 그림처럼 클라이언트와 클래스 사이에 facade라는 객체를 세워놓음으로써 복잡한 관계를 정리(구조화)한 것이다. 즉 모든 관계가 전면에 세워진 facade 객체를 통해서만 이루어질 수 있게 단순한 인터페이스를 제공(단순한 창구기능)하는 것이다. facade 패턴을 이용하면 서브시스템 내부에서 작동하는 많은 클래스의 관계나 사용법을 의식하지 않고 facade에서 제공하는 단순화된 하나의 인터페이스만 사용하므로 클래스 간의 의존 관계가 줄어들고 복잡성도 낮아지는 효과를 볼 수 있다.

여기서 facade 객체는 클라이언트의 요청이 발생했을 때 서브시스템 내의 특정한 객체에 요청을 전달하는 역할을 한다. 이 역할을 수행하려면 facade 객체는 서브시스템의 클래스에 어떤 기능이 있는지(어떤 요청을 처리하는지) 알고 있어야 한다. 서브시스템의 클래스는 각각의 기능을 구현한다. 즉 facade 객체로부터 할당된 작업을 실제로 처리한다. 그러나 facade 객체에 대해 어떤 정보도 가지고 있지 않다.

■ **facade 패턴의 장점 및 단점**

- 클라이언트는 facade로 제공되는 인터페이스만 알고 있으면 사용이 가능하다.
- 클라이언트는 객체의 내부 구조를 상세히 알 필요가 없다.

- 클라이언트가 서브 기능을 쉽게 사용할 수 있도록 단순화시킨다.
- 강한 결합 구조가 느슨한 구조로 변경되어 객체의 의존 관계를 줄여준다.
- facade 클래스가 새로 만들어져 관리할 클래스가 하나 더 늘어난 것은 단점이다.

[8장 연습문제]

01. 좋은 프로그래밍 습관에 관한 설명 중 옳지 않은 것은? ④
① 소프트웨어 유지보수를 용이하게 하기 위해 일관성 있고 의미 있는 변수 이름을 선택한다.
② 함수와 변수 이름은 참조 범위가 최소가 되게 정의한다.
③ 읽기 용이한 코드를 만들기 위해 한 줄에 하나의 문장만을 기술하고 공백 라인을 적절하게 사용한다.
④ 함수는 되도록 다양한 기능을 수행하도록 작성해 함수의 개수를 줄인다.

02. 코딩과 테스트 단계의 관계를 바르게 설명한 것은? ③
① 코딩과 테스트 작업은 가능한 동일한 사람이 해야 한다.
② 통합 테스트는 코딩 단계에서 이루어진다.
③ 코딩은 일부 단위 테스트를 포함한다.
④ 코딩 작업은 디버깅 작업을 포함한다.

03. 표준 코딩 스타일의 장점이 아닌 것은? ④
① 가독성
② 간결하고 명확한 코딩
③ 개발 시간 단축
④ 코딩 길이가 줄어듦

04. 프로그램 표준 코딩 규칙 중 명칭에 관한 규칙에 해당하지 않는 것은? ②
① 명칭의 길이는 31자 이내로 한다.
② 변수명과 함수명은 같게 사용해도 된다.
③ 변수의 이름은 소문자로 시작한다.
④ 포인터의 이름은 참조하는 변수 이름의 첫 글자를 대문자로 한다.

05. 소스 형식에 관한 규칙 중 틀린 것은? ④
① 하나의 원시 파일은 200줄 이내로 한다.
② 한 줄의 길이는 80자 이내로 한다.
③ 함수의 내용은 70줄 이내로 한다.
④ 중괄호 { 의 시작과 끝은 새로운 열의 시작에 위치한다.

06. 주석에 대한 바른 코딩 스타일이 아닌 것은? ③
① 주석과 코드는 일치해야 한다.
② 잘못된 코드에는 주석을 달지 말고 다시 작성한다.
③ 주석은 초보 프로그래머를 위해서 많이 달수록 좋다.

④ 주석은 코드만 읽어 알기 어려운 사항을 적어 넣는 것이 좋다.

07. C 언어에서 연산자 우선순위가 높은 것에서 낮은 것으로 바르게 나열한 것은? ①

㉠ ( )　　㉡ ==　　㉢<　　㉣<<　　㉤ ||　　㉥ /

① ㉠, ㉥, ㉣, ㉢, ㉡, ㉤
② ㉠, ㉣, ㉥, ㉢, ㉡, ㉤
③ ㉠, ㉣, ㉥, ㉢, ㉤, ㉡
④ ㉠, ㉥, ㉣, ㉤, ㉡, ㉢

08. 구현 단계에서의 작업 절차를 순서에 맞게 나열한 것은? ②

㉠ 코딩 작업을 한다.　　㉡ 코딩 작업을 계획한다.
㉢ 코드를 테스트한다.　　㉣ 컴파일한다.

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

09. C 언어에서 정수 자료형으로 옳은 것은? ①
① int
② float
③ char
④ double

10. C 언어에서 비트논리 연산자에 해당하지 않는 것은? ②
① ^
② ?
③ &
④ ~

11. C 언어에서 사용할 수 없는 변수명은? ②
① student2019
② text-color
③ _korea
④ amount

12. 다음의 경우에 C 언어에서 배열 b[5]의 값은? ①

```
static int b[9] = {1, 2, 3};
```

① 0
② 1
③ 2
④ 3

13. C 언어에서 변수로 사용할 수 없는 것은? ④
① data02
② int01
③ _sub
④ short

14. C 언어에서 문자열을 정수형으로 변환하는 라이브러리 함수는? ①
① atoi( )
② atof( )
③ itoa( )
④ ceil( )

15. C 언어에서 산술 연산자가 아닌 것은? ④
① %
② *
③ /
④ =

16. 다음 C 프로그램의 결과 값은? 25

```
#include <stdio.h>
int main(void) {
    int i;
    int sum = 0;
    for(i=1; i <=10; i=i+2)
        sum= sum+i;
    printf("%d", sum);
    return 0;
}
```

17. 다음 C 프로그램을 실행할 때의 결과는? 121

```
#include <stdio.h>
int main(int argc, char *argv[]) {
	int a[2][2] = {{11, 22}, {44, 55}};
	int i, sum = 0;
	int *p;
	p=a[0];
	for(i=1; i<4; i++)
		sum+= *(p+i);
	printf("%d", sum);
	return 0;
}
```

18. 다음 C 프로그램을 실행할 때의 결과는? 66

```
#include <stdio.h>
int main(int argc, char *argv[]) {
	char a;
	a = 'A'+1;

	printf("%d", a);
	return 0;
}
```

19. 다음 코드의 문제점을 설명하고 바르게 수정하시오.

```
int i = 0;
while(1) {
	if(i == 0) {
		break;
	}
	else if(i == 20) {
		break;
	}
}
```

[정답]

```
// 반복을 중단하기 위해 break 문을 한번 사용한다.
int i = 0;
while(1) {
      ……
      if(i == 0) || (i == 20) {
            break;
      }
      ……
}
```

[9장 연습문제]

1. 하향식 통합에서 모듈 간의 통합 시험을 위해 일시적으로 필요한 조건만을 가지고 임시로 제공되는 시험용 모듈을 무엇이라고 하는가? ①

① stub
② driver
③ procedure
④ function

2. 인터페이스의 요구사항 검토 방법에 대한 설명이 옳은 것은? ②

① 리팩토링: 작성자 이외의 전문 검토 그룹이 요구사항명세서를 상세히 조사해 결함, 표준 위배, 문제점 등을 파악한다.
② 동료 검토: 요구사항명세서 작성자가 요구사항명세서를 설명하고 이해관계자들이 설명을 들으면서 결함을 발견한다.
③ 인스펙션: 자동화된 요구사항 관리 도구를 이용해 요구사항의 추적성과 일관성을 검토한다.
④ CASE 도구: 검토 자료를 회의 전에 배포해 사전 검토한 후 짧은 시간 동안 검토 회의를 진행하면서 결함을 발견한다.

[참고] **리팩토링**: 외부 동작을 바꾸지 않으면서 내부 구조를 개선하는 방법

3. 소스 코드의 품질 분석 도구 중 정적 분석 도구가 아닌 것은? ③

① pmd
② cppcheck
③ valMeter
④ checkstyle

[참고]
- **정적 분석 도구**: pmd, cppcheck, SonarQube, checkstyle, ccm, corvertura 등
- **동적 분석 도구**: Avalanche, Valgrind 등
- **pmd(**programming mistake detector): Java 소스 코드 분석기로 코드에서 발견된 문제를 보고하는 정적 분석 도구이다.
- **cppcheck**: C, C++ 언어에서 사용하는 정적 분석 도구이다.
- **checkstyle**: 개발된 코드가 코딩 룰을 얼마나 잘 따르는지 분석해주는 정적 분석 도구이다.
- **FindBugs**: Java 프로그램에서 100여 개의 잠재적인 에러 타입을 찾아주는 정적 분석 도구이다.

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

① Brooks의 법칙
② Boehm의 법칙

③ Pareto의 법칙
④ Jackson의 법칙

[참고]
• **Brooks의 법칙**: 진행 중인 프로젝트에 새로운 개발 인력이 투입될 경우 기존 업무 분석 시간이 필요해져 빠른 시간 내에 프로젝트를 완료할 수 없다.
• **Boehm의 법칙**: 비용 산정을 위한 COCOMO 모델 제시
• **Pareto의 법칙**: 결과의 80%가 전체 원인의 20%에서 일어나는 현상
• **Jackson의 법칙**: JSON 데이터 구조를 처리해주는 라이브러리

5. 다음에서 설명하는 테스트 용어는? ③

• 테스트의 결과가 참인지 거짓인지를 판단하기 위해 사전에 정의된 참값을 입력해 비교하는 기법 및 활동을 말한다.
• 종류에는 참, 샘플링, 휴리스틱, 일관성 검사가 존재한다.

① 테스트 케이스
② 테스트 시나리오
③ 테스트 오라클
④ 테스트 데이터

6. <명세>는 어떤 과목의 통과 여부를 결정하는 프로그램에 대한 명세이다. <코드>의 프로그램은 <명세>에 따라 작성했지만 오류가 있다. <코드>의 오류를 검출할 수 있는 테스팅 기법과 테스트 입력을 바르게 짝지은 것은? ①

[명세]
입력 점수가 70보다 크거나 같으면 통과하고 그렇지 않으면 통과하지 못한다. 점수는 0 이상 100 이하 범위를 갖는 정수형이다. 프로그램의 반환 값이 0이면 통과, 1이면 통과하지 못함, -1이면 입력이 범위를 벗어났음을 나타낸다.

[코드]

```
int passOrNot(int score) {
    if ((score > 100) || (score < 0)) return -1;
    if (score > 70) return 0;
    else return 1;
}
```

① 경계 값 분석 기법, 70
② 경계 값 분석 기법, 100
③ 동등 분할 기법, 50
④ 동등 분할 기법, 80

7. White Box Testing에 대한 설명으로 옳지 않은 것은? ①

① Base Path Testing, Boundary Value Analysis가 대표적인 기법이다.
② 소스 코드의 모든 문장을 한 번 이상 수행함으로써 진행된다.
③ 모듈 안의 작동을 직접 관찰할 수 있다.
④ 산출물의 기능별로 적절한 프로그램의 제어 구조에 따라 선택, 반복 등의 부분을 수행함으로써 논리적 경로를 점검한다.

8. 다음이 설명하는 애플리케이션 통합 테스트 유형은? ①

- 깊이 우선 방식 또는 넓이 우선 방식이 있다.
- 상위 컴포넌트를 테스트하고 점증적으로 하위 컴포넌트를 테스트한다.
- 하위 컴포넌트 개발이 완료되지 않으면 스텁 stub을 사용하기도 한다.

① 하향식 통합 테스트
② 상향식 통합 테스트
③ 회귀 테스트
④ 빅뱅 테스트

9. 개발자 A는 <명세>에 따라 <코드>를 작성한 후 테스트를 수행했다. A는 100% 문장 커버리지를 달성하면서 동시에 프로그램의 오류를 발견할 수 있었다. A가 사용한 테스트 입력은? 단, 단축 연산(short-circuit evaluation)은 수행하지 않는다. ①

[명세]
두 정수를 입력받아 두 정수 중 적어도 하나가 음수이면 두 정수의 곱을 반환하고 그렇지 않다면 두 정수의 합을 반환한다.

[코드]

```
int foo(int v1, int v2) {
    int v3 = v1*v2;
    if (v1 >= 0 || v2 >= 0)
        v3 = v1+v2;
    return v3;
}
```

① v1 = - 2, v2 = 2
② v1 = 2, v2 = 0
③ v1 = - 2, v2 = - 2
④ v1 = 0, v2 = 0

10. T. McCabe의 순환 복잡도(Cyclomatic Complexity)에 대한 설명으로 옳지 않은 것은? ①
① 사이클로매틱 수는 각 모듈에 대한 제어도 fan-out를 이용해 측정한다.
② 사이클로매틱 수는 코드 전체에서 독립적인 경로의 수를 선형적으로 측정한다.
③ 사이클로매틱 수는 그래프 이론을 기반으로 해 코드를 동등한 제어 흐름 그래프로 변환한 다음에 메트릭을 결정하기 위한 그래프의 속성을 이용해 계산한다.

④ 원시 코드의 구조적인 복잡성을 알아내는 측도이다.

11. 인터페이스 구현 검증 도구가 아닌 것은? ①

① ESB
② xUnit
③ STAF
④ NTAF

[참고]
• **ESB(**Enterprise Service Bus): 비즈니스 내에서 서비스, 애플리케이션, 자원을 연결하고 통합하는 미들웨어
• **xUnit**: 테스트 프레임워크로서 소프트웨어의 함수나 클래스와 같은 서로 다른 구성 원소(단위)를 테스트함
• **STAF(**Software Testing Automation Framework): 테스트 프레임워크로 서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원. 테스트 대상 프로그램을 통해 테스트를 수행하고 통합해 자동화하는 검증 도구
• **NTAF(**NHN Test Automation Framework): FitNesse와 STAF의 장점을 결합해 개발된 테스트 자동화 프레임워크

12. 인터페이스 구현 검증 도구 중 아래에서 설명하는 것은? ②

• 서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
• 각 테스트 대상 분산 환경에 데몬을 사용해 테스트를 수행하고 통합해 자동화하는 검증 도구

① xUnit
② STAF
③ FitNesse
④ RubyNode

[참고]
• **FitNesse**: 웹 기반 테스트 케이스의 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크. 사용자가 작성한 테스트 케이스를 가지고 자동으로 원하는 값에 대한 테스트를 수행
• **Ruby**: 간결함과 생산성을 강조한 동적 객체지향 스크립트 프로그래밍 언어
• **Watir**: Ruby 기반의 웹 애플리케이션 테스트 프레임워크. 모든 언어 기반의 웹 애플리케이션 테스트와 브라우저 호환성 테스트 기능
• **Selenium**: 다양한 브라우저 지원 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크. 테스트 스크립트 언어를 학습할 필요 없이 기능 테스트를 만들기 위한 플레이백 도구 제공

13. 다음은 어떤 시스템의 유지보수를 위한 요구사항의 일부이다. 아래 요구사항에 따라 수행한 유지보수 활동이 기존 기능에 영향을 끼쳤는지 알아보기 위해 수행하는 테스팅은? ①

Req - 01	윈도 환경에서 동작하는 시스템을 리눅스 환경에서도 동작하도록 한다.
Req - 02	SMS 문자 발송 기능의 오류를 수정한다.
Req - 03	논리 흐름을 보다 이해하기 쉽도록 코드 구조를 개선한다.

① 회귀 테스팅(regression testing)
② 사용성 테스팅(usability testing)
③ 성능 테스팅(performance testing)
④ 보안성 테스팅(security testing)

14. 소프트웨어 테스팅 문서에 관한 국제 표준은? ③

① ISO/IEC 9899
② IEEE 828
③ IEEE 829
④ IEEE 1042

[참고]
- **ISO/IEC 9899**: C 언어 관련 표준
- **IEEE 828**: 소프트웨어 형상 관리를 위한 표준
- **IEEE 829**: 소프트웨어 테스트를 위한 표준
- **IEEE 1042**: 소프트웨어 형상 관리를 위한 IEEE 가이드

15. 15 다음은 무엇에 사용되는 도구인가? ④

- JUnit
- Mockito
- JMeter

① 버전 관리
② 프로젝트 관리
③ 소프트웨어 설계
④ 소프트웨어 테스팅

[참고]
- **JUnit**: 자바용 단위 테스트 도구
- **Mockito**: 단위 테스트를 위한 Java mocking framework
- **JMeter**: 기능의 부하 테스트 및 측정을 지원하는 Java Application 도구

16. 점수가 80점 이상이면 '합격', 80점 이하이면 '불합격'이고 점수 범위는 0 ~ 100까지 정수이다. 이것을 프로그램으로 작성하고 이 프로그램을 테스트할 때 적합한 테스트 기법은 무엇이며 적절한 테스트 데이터는 얼마인가?

[프로그램 코드]

```
int passOrNot(int score) {
    if ((score > 100) || (score < 0)) return -1;
    if (score >= 80) return 0;
    else return 1;
}
```

경계값분석, 80

17. 아래 프로그램을 문장 검증 기준(statement coverage)을 사용해 테스트하려 한다. 이때 필요한 최소 테스트 케이스의 개수는 얼마인가? 1개

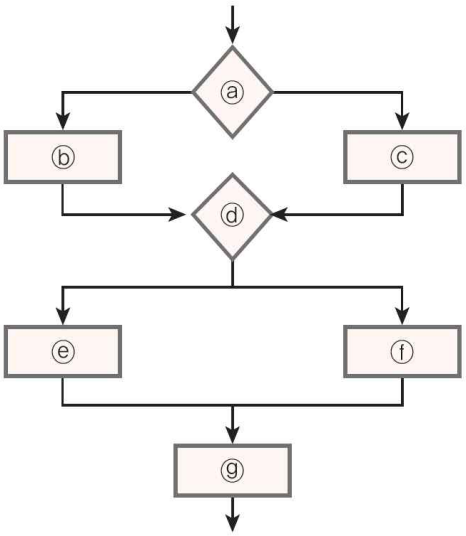
```
int kcs(int a[8], int x) {
    int i = 0;
    int c = 0;
        for(i = 0; I < 8; i++) {
            if(a[i] == x) c++;
        }
    return c;
}
```

18. 아래 프로그램을 분기/조건 검증 기준(branch/condition coverage)을 사용해 테스트하려 한다. 이를 만족하는 테스트 데이터 집합은?

(a:0, b:10, c:60), (a: 5, b: 20, c:70)

```
void test(int x, int y, int z) {
    int p =10;
    if((x > 0 && y <= 15)|| z > 65)
        k=k * 2;
    else
        k = k * 1;
    printf("%d", k);
}
```

19. 아래 그림과 같은 프로그램을 테스트한다. 문장 검증 기준, 분기 검증 기준, 기본 경로 테스트를 만족하도록 하기 위한 최소 테스트 케이스의 개수는 각각 몇 개인가? 단, 제어 흐름도의 모든 경로는 실행할 수 있다.



문장 검증 기준-0개, 분기 검증 기준-2개, 기본 경로 테스팅 -3개

20. 다음 코드에서 순환 복잡도는 얼마인가? 3개

```
int test(int a, int b, int c) {
    if(a >= b) {
        if(a >= c)
            result = a;
        else
            result = c;
    }
    else
        result = b;
}
```

[10장 연습문제]

1. 다음 설명은 누가 생각하는 품질 속성인가? ①

처음에 계획한 개발 비용과 개발 기간 내에 개발을 해 추가 비용 부담이 발생하지 않는 소프트웨어를 좋은 소프트웨어라고 생각할 수 있다.

① 프로젝트 관리자 관점
② 개발자 관점
③ 사용자 관점
④ 구매 담당자 관점

2. W. E. 페리의 품질에 관한 설명 중 옳지 않은 것은? ①

① 정확성(correctness_: 사용자가 요구한 기능을 정확하고 일관되게 원하는 정밀도로 수행할 수 있는 정도
② 무결성 integrity: 허가받지 않은 사용자가 데이터 접근을 통해 변경을 시도할 때 얼마나 보호할 수 있는지 정도
③ 상호운용성 interoperability: 한 소프트웨어를 다른 소프트웨어와 얼마나 쉽게 연계 또는 결합해 정보를 교환할 수 있는지 정도
④ 유지보수 용이성 maintainability: 프로그램 내에 존재하는 오류를 찾아 수정하고 패치할 때 얼마나 쉽게 변경할 수 있는지 정도

3. McCall의 소프트웨어 품질 요소에 대한 설명으로 옳지 않은 것은? ③

① 사용성(usability): 사용자가 쉽게 이해하고 사용이 용이하며 흥미를 느끼는 정도
② 정확성(correctness): 기능적인 요구사항 만족과 고객 목적 이행 정도
③ 이식성(portability): 다른 시스템과 정보 교류의 연동 및 통합 정도
④ 신뢰성(reliability): 필요한 정확성으로 기능을 수행하는 정도

4. 04 CMM(Capability Maturity Model)의 레벨로 옳지 않은 것은? ④

① 최적 단계
② 관리 단계
③ 정의 단계
④ 계획 단계

5. CMMI 프로세스 성숙도 수준(maturity level) 중 관리(managed) 단계의 특징으로만 묶은 것은? ①

㉠ 기본적인 프로젝트 관리 프로세스가 정의되어 비용, 일정, 기능 등을 추적할 수 있다.
㉡ 새로운 프로젝트에 대한 계획과 관리가 이전의 성공한 프로젝트에 근거해 이루어진다.
㉢ 조직의 소프트웨어 프로세스를 전담하는 소프트웨어 공학 프로세스 그룹이 있다.

㉣ 프로세스 개선을 지속적으로 추진해 프로세스 능력 수준을 높인다.

① ㉠, ㉡
② ㉡, ㉢
③ ㉠, ㉡, ㉣
④ ㉡, ㉢, ㉣

6. CMMI 모델의 정의 단계(defined level)에 해당하는 프로세스 영역으로 옳지 않은 것은? ④

① 제품 통합(product integration)
② 의사결정 분석 및 해결(decision analysis and resolution)
③ 조직 교육 관리(organizational training)
④ 조직 프로세스 성과 관리(organizational process performance)

7. 소프트웨어 개발 표준 중 소프트웨어 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준은? ③

① SCRUM
② ISO/IEC 12509
③ SPICE
④ CASE

8. 소프트웨어 품질 측정을 위해 개발자 관점에서 고려해야 할 항목으로 거리가 먼 것은? ④

① 정확성
② 무결성
③ 사용성
④ 간결성

9. ISO/IEC 9126의 소프트웨어 품질 특성 중 기능성(functionality)의 하위 특성으로 옳지 않은 것은? ①

① 학습성
② 적합성
③ 정확성
④ 보안성

10. 어떤 웹 서비스 시스템은 다음과 같은 특징을 가지고 있다. 이 시스템과 관련해 ISO/IEC 9126 품질 특성 중에서 개선할 필요가 있는 것은? ②

- 온라인/오프라인 도움말을 제공하지 않는다.
- 시스템이 제공하는 기능을 메뉴명으로 이해하기 어렵다.
- 모든 웹페이지에서 홈페이지로 바로 가는 '홈 버튼'이 제공되지 않아 이전 페이지로 이동하는 '뒤로 가기 버튼'을 사용해 여러 단계를 거쳐 홈페이지로 갈 수밖에 없다.

① 효율성(efficiency)

② 사용성(usability)
③ 이식성(portability)
④ 유지보수성(maintainability)

11. 소프트웨어 품질 목표 중 주어진 시간 동안 주어진 기능을 오류 없이 수행하는 정도를 나타내는 것은? ③
① 직관성
② 사용 용이성
③ 신뢰성
④ 이식성

12. 패키지 소프트웨어의 일반적인 제품 품질 요구사항 및 테스트를 위한 국제 표준은? ③
① ISO/IEC 2196
② IEEE 19554
③ ISO/IEC 12119
④ ISO/IEC 14959

13. 소프트웨어 프로세스에 대한 개선 및 능력 측정 기준에 대한 국제 표준은? ④
① ISO 14001
② IEEE 802.5
③ IEEE 488
④ SPICE

14. 다음은 ISO의 소프트웨어 프로세스 평가를 위한 국제 표준인 SPICE에 대한 설명이다. 이에 해당하는 프로세스 범주는? ②
시스템과 소프트웨어 제품을 개발하는 모든 프로세스, 즉 요구사항 분석(명세화), 설계, 구현, 테스트 등이 이 범주에 속한다.
① 조직 프로세스(organization process)
② 공학 프로세스(engineering process)
③ 고객 - 공급 프로세스(customer-supplier process)
④ 지원 프로세스(support process)

15. SPICE 모델의 프로세스 수행 능력 단계로 옳지 않은 것은? ①
① 정량적 관리 단계
② 수행 단계
③ 관리 단계
④ 예측 단계

16. 아래의 문장은 ISO/IEC 9126 품질 특성 중에서 어떤 특성에 해당하는가?

사용성

- 도움말(온라인/오프라인)이 없다.
- 메뉴명만 보고는 어떤 기능인지 쉽게 알 수 없다.
- '홈 버튼'이 없어 이전 페이지로 이동하려면 '뒤로 가기 버튼'을 사용해 여러 단계를 거쳐 홈페이지로 가야 한다.

17. 소프트웨어 개발 작업에 일관적이고 체계적인 구조(framework)를 제공하기 위해 1995년에 ISO/IEC에서 제정한 소프트웨어 생명주기 공정 국제 표준은?

ISO/IEC 12207

18. 다음 설명에 해당하는 것은?

- 소프트웨어 품질 평가를 위한 국제 표준이다.
- 소프트웨어 제품 품질 관련 모델을 통합하기 위한 모델로 제시되었다.
- 품질 관리, 품질 모델, 품질 측정, 품질 요구사항, 품질 평가 등으로 구성된다.

ISO 25000

19. 표준 프로세스의 필요성을 설명하시오.

표준 프로세스를 설명하기 전에 음식 만드는 비유를 통해 표준 프로세스의 필요성을 설명한다. 음식을 처음 만드는 입장에서 무엇인가 보고 따라할 수 있는 문서나 사진이나 동영상 등이 있다면 큰 두려움 없이 하나씩 따라할 것이다. 이것을 우리는 레시피라고 하는데 이 레시피대로 따라하면 음식 솜씨가 없는 사람도 어느 정도의 맛은 낼 수 있을 것이다. 또 음식점에서 요리사가 바뀌어도 음식점 자체의 레시피를 가지고 있으면 손님 입장에서는 언제나 일정한 맛을 느낄 수 있을 것이다.

소프트웨어 개발에서도 마찬가지이다. 처음 실무에서 소프트웨어 개발에 참여하는 초보 개발자에게는 레시피와 같은 문서가 있으면 그대로 따라하면서 주어진 일을 수행할 수 있을 것이다. 이것이 표준 프로세스이다. 회사 내에 이러한 표준 프로세스가 존재하면 초보 개발자도 어느 정도 일의 생산성을 낼 수 있을 것이다. 만일 이러한 표준 프로세스가 없다면 생산성도 떨어질 뿐만 아니라, 통일된 개발이 되질 않아 나중에 유지보수의 어려움을 야기시킬 수가 있다.

20. 학습(용이)성이 좋은 경우와 그렇지 않은 경우의 예를 들어보시오.

사용자가 어떤 동작 또는 선택을 할 때 지금까지 보편적으로 생각했던 대로 할 수 있도록 해줌으로써 다시 생각해봐야하는 고민을 덜어주는 것이다. 예를 들어 신호등 색깔이 주는 의미에 일반인들은 익숙해 있다. 초록색은 '간다'라는 진행의 의미, 빨간색은 '정지'라는 멈춤의 의미로 받아들여진다. 따라서 진행과 멈춤의 의미를 나타내는 버튼을 만든다면 신호등 색깔과 일치하는 버튼의 색깔을 만들어야 사용자가 그 색깔만 보고도 고민 없이 쉽게 판단하고 사용할 수 있다. 즉 많은 사람이 공통적으로 생각하는 방식에 따라 사용할 수 있게 제공하는가 여부를 나타낸다. 또 생활 속에서 익숙한 표지판 등을 활용하여 쉽게 그 의미

를 알 수 있게 해준다.

[11장 연습문제]

1. PMBOK의 프로젝트 위험 관리 프로세스에 대한 설명으로 옳지 않은 것은? ③
① 위험 대응 기획은 위험에 대한 대응을 어떻게 할 것인지 대응 전략을 세우고 대응 전략 이후에도 남아 있을 위험과 이차적인 위험, 위험 대응을 위해 필요한 시간과 비용, 위험에 대한 비상 계획과 예비 계획 등을 세우는 프로세스이다.
② 위험 관리 기획은 위험을 언제, 어떤 방법으로 어떻게 관리할 것인가를 계획하는 프로세스이다.
③ 위험 식별은 도출된 위험이 미치는 영향이 얼마나 큰지, 얼마나 자주 발생하는지 등을 분석하는 프로세스이다.
④ 위험 모니터링 및 통제는 식별된 위험에 대해 추적하고 잔존하는 위험을 감시하며 새롭게 발견되는 위험을 식별하고 위험 감소 효과를 평가하는 프로세스이다.

2. 소프트웨어 형상 관리의 의미로 적절한 것은? ②
① 비용에 관한 사항을 효율적으로 관리하는 것
② 개발 과정의 변경 사항을 관리하는 것
③ 테스트 과정에서 소프트웨어를 통합하는 것
④ 개발 인력을 관리하는 것

3. 제품 소프트웨어의 형상 관리 역할로 틀린 것은? ③
① 형상 관리를 통해 이전 리버전이나 버전에 대한 정보에 접근 가능해 배포본 관리에 유용
② 불필요한 사용자의 소스 수정 제한
③ 프로젝트 개발 비용을 효율적으로 관리
④ 동일한 프로젝트에 대해 여러 개발자 동시 개발 가능

4. 형상 관리 도구의 주요 기능으로 거리가 먼 것은? ①
① 정규화(normalization)
② 체크인(check-in)
③ 체크아웃(check-out)
④ 커밋(commit)

[참고]
- **체크아웃**: 변경사항 관련 없이 서버 형상을 로컬 형상으로 가져오는 것
- **커밋**: 로컬 형상을 서버 형상으로 보내는 것으로 체크인이라고도 함

5. 소프트웨어 형상 관리에서 관리 항목에 포함되지 않는 것은? ④
① 프로젝트 요구분석서
② 소스 코드

③ 운영 및 설치 지침서
④ 프로젝트 개발 비용

6. 형상 관리의 형상 제어(configuration control) 활동에서 수행하는 작업으로만 묶은 것은? ②

㉠ 형상 항목과 형상 식별자 선정
㉡ 변경 요청사항에 대한 심사 및 변경 실시
㉢ 변경 내용을 확인하고 베이스라인 수립
㉣ 형상 관리 계획서대로 형상 관리가 진행되고 있는지 검증

① ㉠, ㉡
② ㉡, ㉢
③ ㉡, ㉣
④ ㉢, ㉣

7. 소프트웨어 설치 매뉴얼에 대한 설명으로 틀린 것은? ③
① 설치 과정에서 표시될 수 있는 예외 상황에 관련 내용을 별도로 구분해 설명한다.
② 설치 시작부터 완료할 때까지 전 과정을 빠짐없이 순서대로 설명한다.
③ 설치 매뉴얼은 개발자 기준으로 작성한다.
④ 설치 매뉴얼에는 목차, 개요, 기본사항 등이 기본적으로 포함되어야 한다.

8. 빌드 자동화 도구에 대한 설명으로 틀린 것은? ④
① Gradle은 실행할 처리 명령을 모아 태스크로 만든 후 태스크 단위로 실행한다.
② 빌드 자동화 도구는 지속적인 통합개발환경에서 유용하게 활용된다.
③ 빌드 자동화 도구에는 Ant, Gradle, Jenkins 등이 있다.
④ Jenkins는 Groovy 기반으로 한 오픈 소스로 안드로이드 앱 개발 환경에서 사용된다.

[참고]
- 빌드 자동화 도구: Ant, Make, Maven, Gradle, Jenkins
- **Gradle**: 프로젝트를 위한 범용 빌드 도구로서 Groovy를 기반으로 한다.
- **Jenkins**: 소프트웨어 개발 시 지속적 통합(continuous integration) 서비스를 제공하는 툴

9. CASE Computer Aided Software Engineering의 주요 기능으로 옳지 않은 것은? ④
① S/W 생명주기 전 단계의 연결
② 그래픽 지원
③ 다양한 소프트웨어 개발 모형 지원
④ 언어 번역

10. 소프트웨어 재공학이 소프트웨어 재개발에 비해 갖는 장점으로 거리가 먼 것은? ④
① 위험 부담 감소

② 비용 절감
③ 시스템 명세의 오류 억제
④ 개발 시간의 증가

[참고]
소프트웨어 재공학: 기존 소프트웨어를 버리지 않고 기능을 개선하거나 기능을 새로운 소프트웨어로 재활용하는 소프트웨어 재사용 공법으로 종류로는 재구성(restructuring), 역공학(reverse engineering), 이관(migration), 재사용(reuse)이 있다.

11. 리팩토링(refactoring)에 대한 설명으로 옳지 않은 것은? ②
① 리팩토링의 대상은 읽기 어려운 코드, 중복된 로직의 코드, 복잡한 조건문이 있는 코드 등이 대표적이다.
② 리팩토링은 실행 중인 프로그램의 기능 변경이 수반되어야 한다.
③ 리팩토링을 통해 프로그램의 이해가 쉬워진다.
④ 리팩토링은 결함을 찾는 데 도움을 준다.

[참고] **리팩토링**: 외부 동작을 바꾸지 않으면서 내부 구조를 개선하는 방법

12. 외계인 코드(alien code)에 대한 설명으로 옳은 것은? ②
① 로직이 복잡해 이해하기 어려운 프로그램을 의미한다.
② 아주 오래되거나 참고 문서 또는 개발자가 없어 유지보수 작업이 어려운 프로그램을 의미한다.
③ 오류가 없어 디버깅 과정이 필요 없는 프로그램을 의미한다.
④ 사용자가 직접 작성한 프로그램을 의미한다.

[참고] **외계인 코드**: 프로그램에 대한 문서화도 되어 있지 않고 코드를 개발한 개발자도 없어 유지보수가 어려운 코드

13. 리팩토링의 대상이 되는 코드 스멜(code smell)에 해당하지 않는 것은? ③
① 읽기 어려운 프로그램
② 중복된 로직을 가진 프로그램
③ 외부에서 보이는 기능을 변경해야 하는 프로그램
④ 복잡한 조건문이 포함된 프로그램

[참고] 코드 스멜 : 문제를 일으킬 가능성이 있는 소스 코드

14. 미들웨어 솔루션의 유형에 포함되지 않는 것은? ②
① WAS
② Web Server
③ RPC

④ ORB

[참고] **미들웨어 솔루션**
- **WAS**(Web Application Server): 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리
- **RPC**(Remote Procedure Call): 응용 프로그램의 프로시저를 사용해 원격 프로시저를 마치 로컬 프로시저처럼 호출하는 방식
- **ORB**(Object Request Broker): 코바 CORBA 표준 스펙을 구현

15. 제품 소프트웨어 패키징 도구 활용 시 고려 사항이 아닌 것은? ④
① 제품 소프트웨어의 종류에 적합한 암호화 알고리즘을 고려한다.
② 추가로 다양한 이기종 연동을 고려한다.
③ 사용자 편의성을 위한 복잡성 및 비효율성 문제를 고려한다.
④ 내부 콘텐츠에 대한 보안은 고려하지 않는다.

[참고] **제품 소프트웨어 패키징 도구**
- 배포를 위한 패키징 시에 디지털 콘텐츠의 지적 재산권을 보호하고 관리하는 기능을 제공한다.

패키지 도구 활용 시 고려 사항
- 반드시 암호화/보안을 고려한다.
- 다양한 이기종 연동을 고려한다.
- 사용자 편의성을 위한 복잡성 및 비효율성 문제를 고려한다.
- 제품 소프트웨어의 종류에 적합한 암호화 알고리즘을 적용한다.

16. 변경 관리의 필요성을 설명하시오.

"서로 다른 버전의 원시 파일에 어떤 차이점이 있는가"와 같은 질문에 바로 대답할 수 있도록 각 버전의 정보를 데이터베이스화하여 언제라도 과거의 릴리스된 파일로 작업할 수 있도록 관리하는 데 있다. 즉 파일의 이력이나 차이점을 관리해 애플리케이션의 버전과 각 원시 파일이나 문서를 유용하게 활용하기 위함이다.

17. 버전 번호를 매기는 일반적인 과정을 예를 들어 설명하시오.

문서를 작성한 후 처음 저장할 때 파일명을 '150613_형상 관리.hwp'와 같이 붙인다. 이 파일명을 보면 내용과 최종 저장한 날짜를 짐작할 수 있다. 또 같은 날 몇 개의 파일이 만들어질 경우 '150613_형상 관리_V01.hwp', '150613_형상 관리_V02.hwp'처럼 버전의 약자인 V를 사용하면 같은 날 몇 번째 저장한 파일인지 알 수 있다. 이렇게 하다보면 파일 수가 많아져 복잡할 수 있으니까 최종 버전을 남겨두고 모두 OLD라는 폴더에 보관해둔다. 이렇게 이전 파일을 버리지 않고 보관하는 이유는 무엇일까? 문서 내용을 수정하다보면 전에는 필요 없을 것 같아 삭제한 그림이나 내용을 복원해 사용할 수도 있기 때문이다.

18. 최종 릴리스가 되기까지 과정을 버전 항목 트리를 이용해 예를 들어 설명하시오.

[그림 11-2]는 소프트웨어 개발 중에 생성된 버전 트리를 나타내고 있다. 요구분석 단계에서 생성된 요구분석명세서는 3개의 버전(V1.1, V1.2, V1.3)으로 진화해왔다. 또 설계 단계의 최종 산출물인 설계 사양서는 4개의 버전(V1.1, V1.2, V1.3, V1.4)으로 진화해왔다. 마찬가지로 코딩의 결과물인 원시 코드도 5개 의 버전(V1.1, V1.2, V1.3, V1.4, V1.5)으로 진화해왔다.

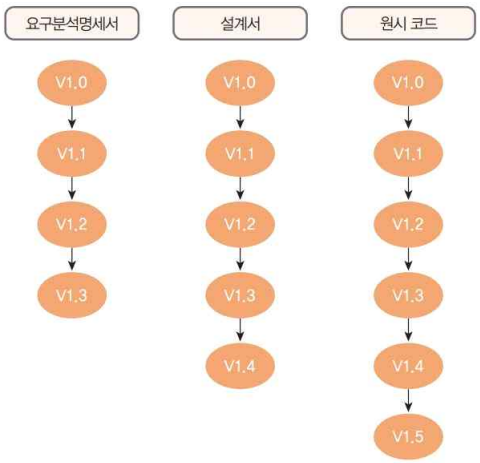


그림 11-2 버전 항목 트리

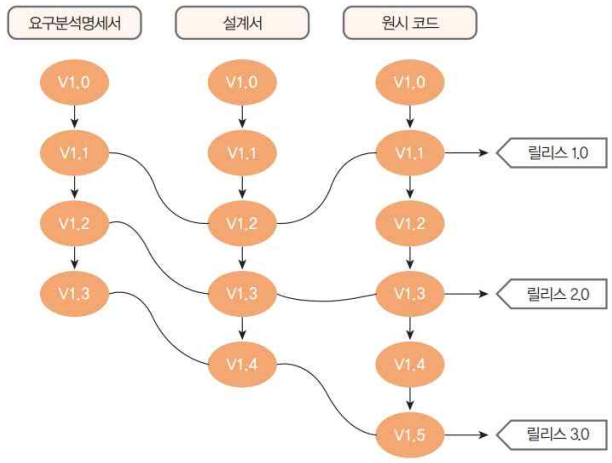


그림 11-3 릴리스된 버전 항목을 나타내는 트리

[그림 11-3]을 보면 최종 릴리스를 구성하는 각 단계의 버전은 다음과 같음을 알 수 있다.

- 릴리스 1.0: 요구 분석 명세서(V1.1) - 설계 사양서(V1.2) - 원시 코드(V1.1)
- 릴리스 2.0: 요구 분석 명세서(V1.2) - 설계 사양서(V1.3) - 원시 코드(V1.3)
- 릴리스 3.0: 요구 분석 명세서(V1.3) - 설계 사양서(V1.4) - 원시 코드(V1.5)

처음 제품으로 출시된 것이 릴리스 1.0인데 그 구성은 (V1.1, V1.2, V1.1)로 이루어졌다. 그리고 각 단계는 계속해서 버전 업을 시키고 있고 특정 시점에서 또 새로운 릴리스 2.0을 출시하게 된다. 그 구성은 (V1.2, V1.3, V1.3)으로 되어 있다.

19. 병합을 이용한 파일 작업 과정을 예를 들어 설명하시오.

병합은 서로 다른 개발 흐름에서 변경된 내용을 하나의 개발 흐름으로 통합할 때 사용한다. [그림 11-8]에서 패치 파일 V1.3.1.3이 V1.6에 통합된 것을 볼 수 있다. 그러므로 V1.6부터의 파일은 Ver.2015의 오류가 해결된 파일이다.

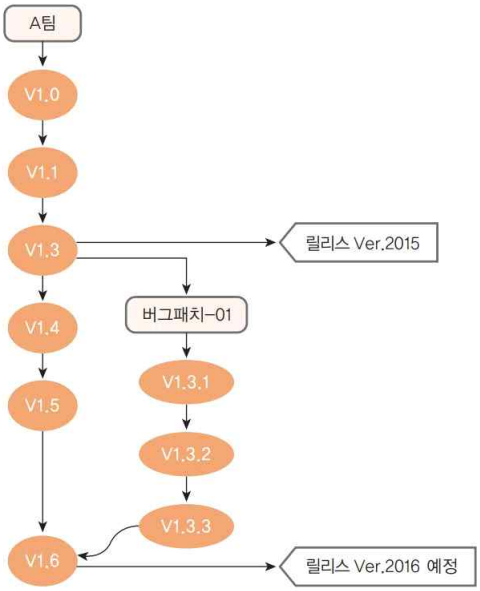


그림 11-8 병합을 이용한 파일 작업

20. 형상 관리의 효과를 설명하시오.

형상 관리를 통해 언제라도 특정 시간대에 가장 안정적인 버전의 소프트웨어를 유지할 수 있도록 소프트웨어 제품이 변경되어가는 상태에 대한 가시성을 확보해준다. 또한 누가 변경했는지 변경된 것은 무엇인지 언제 변경되었는지 왜 변경했는지와 같은 질문에 대답해준다.

형상 관리는 궁극적으로 프로젝트를 개발하는 동안 생산성과 안전성을 높여 좋은 품질의 소프트웨어를 생산하고 유지보수도 용이하게 해주는 데 목적이 있다. 만약 잦은 요구사항 변경과 수정된 산출물의 이력이 정확하게 관리되지 않고 관련자 모두에게 변경 내용이 제대로 전달되지 않는다면 이중 작업과 중복 작업이 발생할 수 있고 그로 인해 작업에 혼란이 생겨 결국 원하는 결과를 얻을 수 없게 된다. 그러므로 형상 관리에서는 적절한 변경 관리를 통하여 무절제한 변경을 사전에 예방하고 변경에 따른 부작용을 최소화한다. 또한 형상 관리를 통해 프로젝트를 적절히 통제하여 체계적이고 효율적으로 관리할 수 있으며 가시성과 추적성을 보장함으로써 소프트웨어의 생산성과 품질을 높일 수 있다.