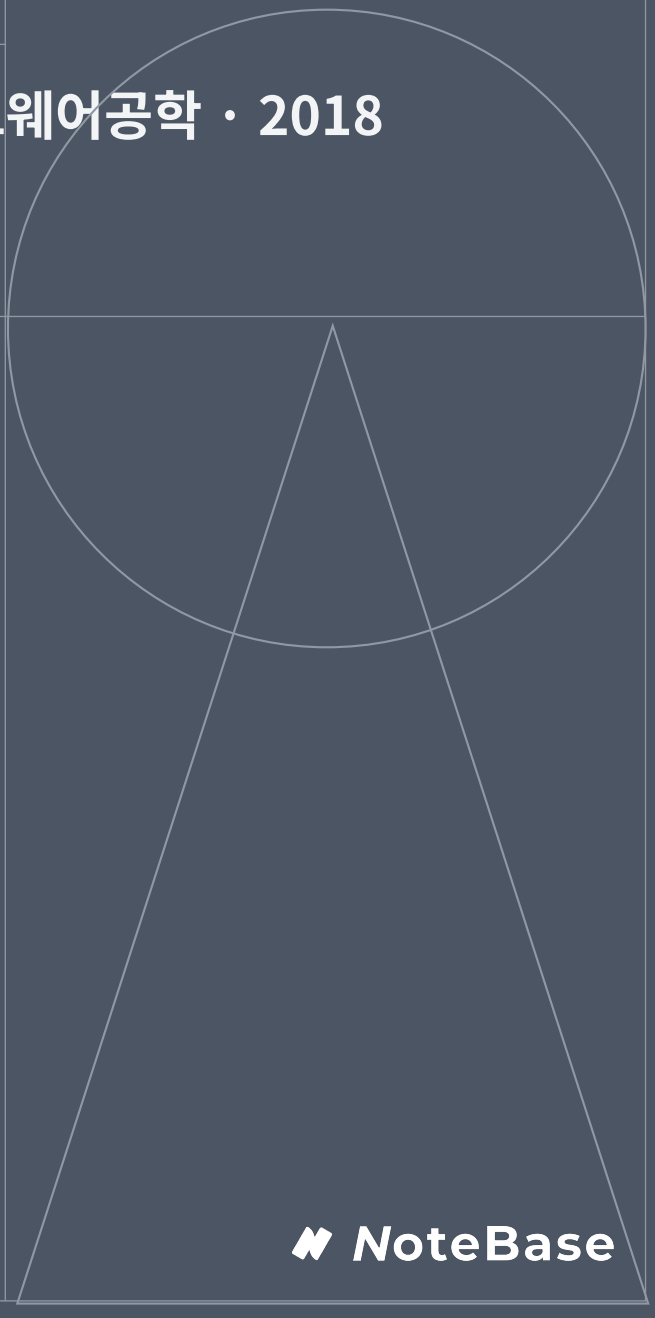
기출OX 해설집

공무원/국가직/7급 · 소프트웨어공학 · 2018



NoteBase

기출OX 해설집

공무원/국가직/7급 · 소프트웨어공학 · 2018

공무원/국가직/7급 · 소프트웨어공학 · 2018

문 1. 소프트웨어 개발 프로세스 모델 중 폭포수(waterfall) 모델에 대한 설명으로 옳지 않은 것은?

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.

정답 4번

출제 포인트

이 문제는 폭포수 모델의 순차적 개발 구조와 요구사항 변화 대응 한계를 구별할 수 있는지를 묻는다.

핵심 개념

폭포수 모델은 요구분석, 설계, 구현, 시험을 순차적으로 진행하는 전통적 개발 모형이다. 요구사항이 비교적 안정적일 때 적합하고, 각 단계 산출물을 문서화하는 데 강점이 있다. 반대로 요구사항 변경이 잦은 환경에는 유연하게 대응하기 어렵다.

선지별 해설

1번 O: 폭포수 모델은 앞 단계의 결과가 확정된 뒤 다음 단계로 넘어가는 순차형 모델이므로, 요구사항 분석이 완료된 후 설계 작업을 시작하는 방식이 맞다.
2번 O: 실행 가능한 소프트웨어는 구현과 시험 단계를 거친 뒤에야 가시적으로 만들어지므로, 개발 후반부에 이르러서야 동작하는 결과물이 나타나는 편이다.
3번 O: 폭포수 모델은 단계별 산출물을 명확히 남기고 검토하는 구조이므로, 요구사항 명세서·설계서·시험 문서 등을 체계적으로 문서화할 수 있다.
4번 X: 틀린 표현은 '요구사항의 변경이 많은 경우에 적합'이다. 폭포수 모델은 이전 단계로 되돌아가 수정하기가 비효율적이어서 요구사항 변화가 많은 프로젝트에는 부적합하다. 올바르게는 '요구사항이 비교적 명확하고 변경이 적은 경우에 적합한 모델'이다.

NoteBase

오답 포인트

문서화가 잘 된다는 장점을 개발 유연성까지 높다고 오해하면 4번에 끌리기 쉽다. 폭포수 모델의 강점은 통제와 문서화이지, 변화 대응성이 아니다.

다시 나오면 이렇게 판단

개발 프로세스 문제가 나오면 먼저 순차형인지 반복형인지부터 구분한다. 단계 확정 후 다음 단계로 진행하고 변경 비용이 크면 폭포수 모델로 판단할 수 있다.

암기 포인트

폭포수 모델은 순차 진행·문서화 강점·변경 대응 약점이다.

NoteBase

문 2. 소프트웨어 통합 프로세스(UP) 모델의 구체화 단계(elaboration phase)에서 하는 주요 활동으로 옳지 않은 것은?

① 사용자 환경에서 인수 테스트 수행

② 아키텍처 설계

③ 요구사항 분석

④ 중대한 위험 요소 식별 및 해결

정답 1번

출제 포인트

이 문제는 UP의 구체화 단계와 전이 단계의 대표 활동을 구분할 수 있는지를 묻는다.

핵심 개념

구체화 단계는 요구사항을 정교화하고 시스템 아키텍처의 기반을 확립하며 주요 위험 요소를 제거하는 단계이다. 키워드는 아키텍처, 요구사항 분석, 위험 해소이다.
전이 단계는 완성된 시스템을 실제 사용자 환경으로 넘기면서 배포, 인수 테스트, 사용자 교육을 수행하는 단계이다. 키워드는 사용자 환경, 인수 테스트, 운영 전환이다.

선지별 해설

1번: X. 사용자 환경에서의 인수 테스트는 제품을 실제 운영 환경으로 옮기는 전이 단계 활동이다. 구체화 단계의 핵심은 아키텍처 확립과 요구사항 정교화, 주요 위험 제거이다. 틀린 표현은 "사용자 환경에서 인수 테스트 수행"이며, 구체화 단계에 맞게 고치면 "아키텍처 설계와 요구사항 분석, 주요 위험 요소 식별 및 해결 수행"이 된다.
2번: O. 구체화 단계에서는 시스템의 실행 가능한 골격을 세우기 위해 아키텍처를 설계한다. UP에서 이 단계의 가장 중요한 산출물 중 하나가 아키텍처 기반이다.
3번: O. 구체화 단계에서는 핵심 요구사항과 유스케이스를 더 구체적으로 분석하여 이후 구현의 기준을 확정한다. 따라서 요구사항 분석은 구체화 단계의 주요 활동에 해당한다.
4번: O. 구체화 단계의 목적 중 하나가 프로젝트의 중대한 위험 요소를 조기에 식별하고 해결하는 것이다. 위험을 초기에 줄여야 이후 구축 단계의 실패 가능성을 낮출 수 있다.

오답 포인트

선지에 "테스트"라는 단어가 들어가면 개발 중간 단계 활동으로 오해하기 쉽지만, "사용자 환경"과 "인수 테스트"가 함께 나오면 전이 단계를 떠올려야 한다. 구체화 단계는 만드는 준비 단계이고, 전이 단계는 넘겨서 쓰게 하는 단계라는 차이가 핵심이다.

NoteBase

다시 나오면 이렇게 판단

선지가 아키텍처 확립, 요구사항 정교화, 위험 제거를 말하면 구체화 단계로 판단한다. 선지가 실제 사용자 환경 배포, 인수 테스트, 교육, 운영 전환을 말하면 전이 단계로 판단한다.

암기 포인트

구체화는 아키텍처·요구사항·위험 해소 단계이고, 전이는 배포·인수 테스트·사용자 전환 단계이다.

NoteBase

문 3. 기능점수(FP)를 계산하기 위해 고려할 대상으로 옳지 않은 것은?

① 외부조회(EQ)

② 내부논리파일(ILF)

③ 외부연계파일(EIF)

④ 내부출력(IO)

정답 4번

출제 포인트

이 문제는 기능점수(FP)의 측정 대상이 되는 다섯 가지 기능 유형을 정확히 구분하는지를 묻는 문제이다.

핵심 개념

FP는 사용자 관점의 기능 규모를 세는 방식이며, 측정 대상은 트랜잭션 기능인 EI·EO·EQ와 데이터 기능인 ILF·EIF의 다섯 가지이다.
EQ는 내부 파일을 단순 조회해 입력과 출력이 오가지만 복잡한 가공이 크지 않은 기능이다.
ILF는 시스템 내부에서 유지·관리하는 논리적 데이터 집합이다.
EIF는 외부 시스템이 유지하지만 현재 시스템이 참조하는 논리적 데이터 집합이다.
IO는 FP의 표준 기능 유형 명칭이 아니며, 출력 기능의 정식 명칭은 EO이다.

선지별 해설

① O: 외부조회(EQ)는 기능점수 계산 대상이다. 사용자가 질의를 보내고 시스템이 내부 데이터를 조회해 응답하는 트랜잭션 기능으로 분류된다.
② O: 내부논리파일(ILF)은 기능점수 계산 대상이다. 애플리케이션 경계 내부에서 유지되는 사용자 식별 가능 데이터 집합이므로 데이터 기능에 해당한다.
③ O: 외부연계파일(EIF)은 기능점수 계산 대상이다. 현재 시스템 바깥에서 유지되지만 현재 시스템이 참조하는 데이터 집합을 세는 항목이다.
④ X: 틀린 부분은 "내부출력(IO)"라는 명칭이다. FP의 표준 항목에는 IO가 없고, 올바른 표현은 "외부출력(EO)"이다.

오답 포인트

FP 약어를 글자만 비슷하게 외우면 IO와 EO를 혼동하기 쉽다. 특히 입력·출력·조회의 트랜잭션 기능 세 가지와 파일 기능 두 가지를 묶어 기억하지 않으면 존재하지 않는 약어를 정답처럼 고르기 쉽다.

NoteBase

다시 나오면 이렇게 판단

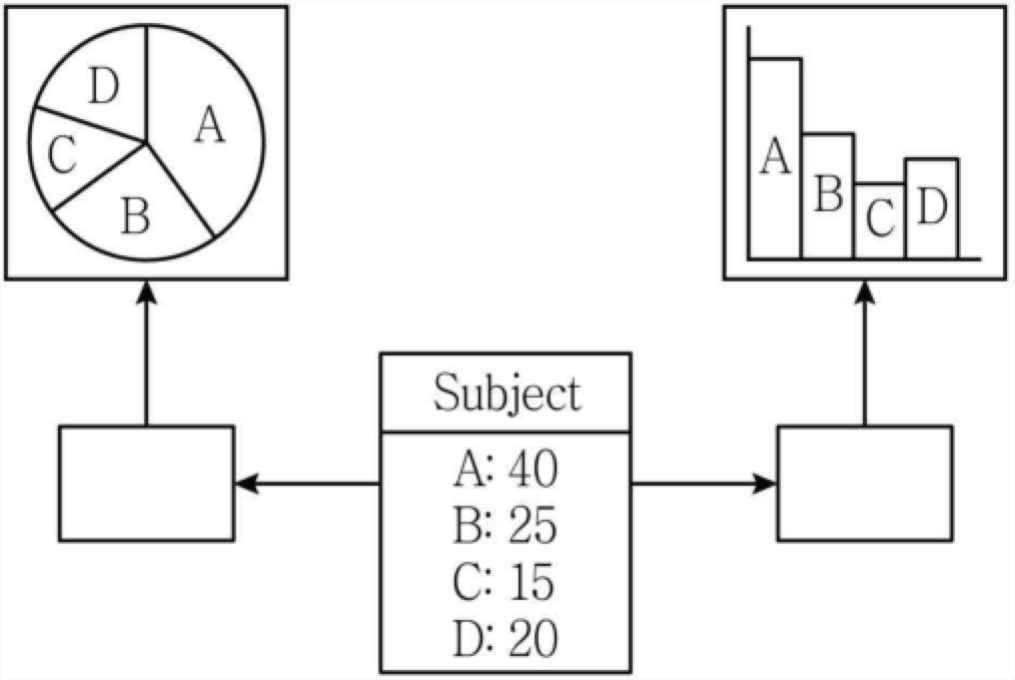
FP 문제는 먼저 다섯 유형이 EI, EO, EQ, ILF, EIF인지 확인하면 된다. 약어가 이 다섯 개 밖의 형태로 나오면 오답 후보로 보고, 트랜잭션 기능 3개와 데이터 기능 2개로 다시 분류해 점검하면 된다.

암기 포인트

FP 5유형은 EI·EO·EQ·ILF·EIF이며, IO는 없다.

NoteBase

문 4. 그림과 같이 관찰대상(Subject)의 데이터(A ~ D)에 변화가 발생하면 이 변화를 탐지하여 여러 가지 방식으로 사용자에게 디스플레이하는 프로그램을 작성하고자 한다. 이 프로그램에 적용할 수 있는 디자인 패턴은?



① Decorator 패턴

② Flyweight 패턴

③ Mediator 패턴

④ Observer 패턴

정답 4번

출제 포인트

이 문제는 Subject의 상태 변화가 발생했을 때 여러 화면 객체가 자동으로 통지받아 각각 다른 형태로 갱신되는 구조를 보고 Observer 패턴을 식별하는 문제이다.

NoteBase

선지별 해설

1번 X: Decorator는 기존 객체에 기능을 동적으로 덧붙이는 패턴이다. 그림의 핵심은 기능 추가가 아니라 Subject의 데이터 변경이 여러 표시 객체로 전파되는 구조이다. 틀린 표현은 변화 통지 구조를 Decorator로 보는 점이며, 올바르게는 부가 기능을 계층적으로 덧붙일 때 Decorator이다.
2번 X: Flyweight는 많은 객체가 공통 내부 상태를 공유하여 메모리를 절약하는 패턴이다. 그림의 핵심은 상태 공유가 아니라 하나의 Subject와 여러 화면 객체 사이의 일대다 통지 관계이다. 틀린 표현은 여러 화면이 같은 데이터를 본다는 이유로 Flyweight로 보는 점이며, 올바르게는 대량 객체의 공통 상태를 공유할 때 Flyweight이다.
3번 X: Mediator는 여러 객체의 상호작용을 중앙 객체가 중재하도록 만드는 패턴이다. 그림에는 객체 간 복잡한 상호작용 조정자가 아니라 Subject의 변경을 받아 갱신되는 관찰자들이 나타난다. 틀린 표현은 변경 통지 구조를 Mediator로 보는 점이며, 올바르게는 객체 간 협력을 중앙에서 조정할 때 Mediator이다.
4번 O: Observer는 한 객체의 상태가 변하면 그 객체에 의존하는 여러 객체에게 자동으로 통지되고 갱신되도록 하는 패턴이다. 그림에서 Subject의 데이터 A~D가 바뀌면 원형 차트와 막대 그래프가 함께 갱신되므로 Observer 패턴에 해당한다.

오답 포인트

도식에서 화살표가 Subject에서 여러 표시 객체로 뻗고, 같은 데이터를 서로 다른 화면으로 보여 준다는 점이 핵심 함정이다. 화면이 여러 개라는 이유만으로 기능 확장인 Decorator나 상태 공유인 Flyweight로 오해하면 틀린다.

다시 나오면 이렇게 판단

한 객체의 상태 변화가 여러 출력 객체에 자동 반영되면 Observer를 먼저 떠올린다. 기능을 덧붙이는지, 상태를 공유하는지, 상호작용을 중앙에서 조정하는지를 함께 비교하면 다른 패턴과 구별된다.

암기 포인트

하나의 Subject가 바뀌자 여러 View가 함께 갱신되면 Observer이다.

NoteBase

문 5. CMMI 모델의 정의 단계(defined level)에 해당하는 프로세스 영역으로 옳지 않은 것은?

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)

정답 4번

출제 포인트

이 문제는 CMMI 성숙도 3단계인 정의 단계와 4단계인 정량적 관리 단계를 구분할 수 있는지를 묻는다.

핵심 개념

정의 단계(Level 3)는 조직 표준 프로세스를 문서화하고 프로젝트 전반에 일관되게 적용하는 단계이다. 제품 통합, 의사결정 분석 및 해결, 조직 교육 관리는 모두 이 단계의 대표 프로세스 영역이다. 정량적 관리 단계(Level 4)는 프로세스 성과를 수치로 측정하고 통계적으로 관리하는 단계이며, 조직 프로세스 성과 관리가 여기에 속한다.

선지별 해설

1번 O: 제품 통합은 개별 구성요소를 통합해 완전한 제품을 만드는 프로세스 영역으로 정의 단계(Level 3)에 포함된다.
2번 O: 의사결정 분석 및 해결은 대안들을 기준에 따라 분석해 합리적으로 선택하는 프로세스 영역으로 정의 단계(Level 3)에 해당한다.
3번 O: 조직 교육 관리는 조직 구성원에게 필요한 지식과 기술을 체계적으로 제공하는 프로세스 영역으로 정의 단계(Level 3)에 속한다.
4번 X: 틀린 부분은 "정의 단계"이다. 조직 프로세스 성과 관리는 프로세스 수행 결과를 정량적으로 관리하는 영역이므로 정량적 관리 단계(Level 4)에 해당한다.

오답 포인트

정의 단계는 프로세스를 표준화하고 정착시키는 단계이고, 정량적 관리 단계는 그 성과를 수치로 관리하는 단계라는 차이를 놓치면 4번을 맞는 말처럼 보기 쉽다. "프로세스 성과"라는 표현이 나오면 측정과 통계 관리 여부를 먼저 떠올려야 한다.

다시 나오면 이렇게 판단

CMMI 문제에서는 먼저 해당 영역이 표준 프로세스의 정립과 활용에 관한 것인지, 아니면 성과를 수치로 측정하고 관리하는 것인지 구분하면 된다. 전자이면 Level 3, 후자이면 Level 4로 판단할 수 있다.

NoteBase

암기 포인트

Level 3는 표준화, Level 4는 정량관리이다.

NoteBase

문 6. PMBOK의 프로젝트 위험 관리 프로세스에 대한 설명으로 옳지 않은 것은?

① 위험 대응 기획은 위험에 대한 대응을 어떻게 할 것인지 대응 전략을 세우고, 그 대응 전략 이후에도 남아 있을 위험과 이차적인 위험, 위험 대응을 위해 필요한 시간과 비용, 위험에 대한 비상 계획, 예비 계획 등을 세우는 프로세스이다.

② 위험 관리 기획은 위험들을 언제, 어떤 방법으로, 어떻게 관리할 것인가를 계획하는 프로세스이다.

③ 위험 식별은 도출된 위험들이 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.

정답 3번

출제 포인트

이 문제는 PMBOK의 위험 관리 프로세스별 정의를 구분하는 문제이다.

핵심 개념

위험 관리 계획은 위험 관리를 언제, 어떤 방법으로, 어떤 절차로 수행할지 정하는 상위 계획이다.
위험 식별은 프로젝트에 영향을 줄 수 있는 위험을 찾아 목록화하고 그 특성을 문서화하는 프로세스이다.
위험 대응 계획은 위협과 기회에 대해 어떤 대응 전략을 쓸지 정하고, 잔여 위험, 이차적 위험, 비상 계획과 예비 계획까지 마련하는 프로세스이다.
위험 모니터링 및 통제는 식별된 위험과 잔여 위험을 추적하고 새 위험을 찾아 대응 효과를 점검하는 프로세스이다.

NoteBase

선지별 해설

① O
위험 대응 계획은 위험에 대한 대응 전략을 정하고, 대응 후에도 남는 잔여 위험과 대응 과정에서 새로 생길 수 있는 이차적 위험, 비상 계획과 예비 계획까지 포함한다.

② O
위험 관리 계획은 위험 관리 활동을 언제, 어떤 방법으로, 어떻게 운영할지를 정하는 프로세스이므로 설명이 맞다.

③ X
틀린 부분은 "위험 식별은 도출된 위험들이 미치는 영향이 얼마나 큰지, 얼마나 자주 발생하는지 등을 분석하는 프로세스"라는 표현이다.
위험 식별은 위험을 찾아내고 문서화하는 단계이며, 영향과 발생 가능성을 분석하는 것은 정성적 위험 분석 또는 정량적 위험 분석에 해당한다.
올바르게는 "위험 식별은 프로젝트에 영향을 줄 수 있는 위험을 식별하고 그 특성을 문서화하는 프로세스"이다.

④ O
위험 모니터링 및 통제는 식별된 위험을 추적하고 잔여 위험을 감시하며 새 위험을 식별하고 위험 대응의 효과를 평가하는 프로세스이므로 설명이 맞다.

오답 포인트

③은 위험 식별과 위험 분석을 일부러 섞어 놓은 선지이다. "영향이 얼마나 큰지", "얼마나 자주 발생하는지"라는 표현이 나오면 이미 식별 단계가 아니라 분석 단계라는 점이 핵심 함정이다.

다시 나오면 이렇게 판단

선지에 위험을 찾아 목록화한다는 뜻이 있으면 위험 식별이고, 발생 가능성이나 영향도를 따지면 위험 분석이며, 대응 전략과 비상 계획을 세우면 위험 대응 계획, 추적과 효과 평가를 말하면 위험 모니터링 및 통제로 판단한다.

암기 포인트

식별은 찾는 단계, 분석은 따지는 단계, 대응은 준비하는 단계, 모니터링은 추적하는 단계이다.

NoteBase

문 7. 대학교 학사관리시스템의 기능적 요구사항으로 옳지 않은 것은?

① 담당 교수는 자신이 맡은 과목의 성적을 입력 또는 수정한다.

② 학적관리자는 학생 정보를 등록, 삭제할 수 있다.

③ 학생은 자신이 수강한 모든 과목의 성적을 조회할 수 있다.

④ 시스템 장애로 인한 정지시간이 1년에 10시간을 넘지 않아야 한다.

정답 4번

출제 포인트

이 문제는 기능 요구사항과 비기능 요구사항의 구분을 묻는 문제이다.

핵심 개념

기능 요구사항은 시스템이 무엇을 해야 하는지를 정의한 요구사항이다. 입력, 수정, 삭제, 조회처럼 사용자가 이용하는 서비스와 처리 기능이 여기에 해당한다.
비기능 요구사항은 시스템이 어느 수준으로 동작해야 하는지를 정의한 요구사항이다. 성능, 가용성, 신뢰성, 응답시간, 허용 정지시간 같은 품질 제약이 여기에 해당한다.

선지별 해설

① X
담당 교수가 자신이 맡은 과목의 성적을 입력 또는 수정하는 기능은 시스템이 제공해야 하는 업무 처리 기능이다. 따라서 기능 요구사항이므로 정답 선택지는 아니다.

② X
학적관리자가 학생 정보를 등록, 삭제하는 것은 사용자 역할에 따른 데이터 관리 기능이다. 따라서 기능 요구사항이므로 정답 선택지는 아니다.

③ X
학생이 자신이 수강한 과목의 성적을 조회하는 것은 조회 기능에 해당한다. 따라서 기능 요구사항이므로 정답 선택지는 아니다.

④ O
시스템 장애로 인한 정지시간이 1년에 10시간을 넘지 않아야 한다는 내용은 시스템이 제공하는 기능이 아니라 가용성과 신뢰성 수준에 대한 제약이다. 틀린 표현은 기능적 요구사항이라는 해석이다. 올바르게는 비기능 요구사항이다.

NoteBase

오답 포인트

입력, 등록, 조회 같은 동사가 보이면 기능 요구사항일 가능성이 크고, 시간 한도나 성능 기준이 보이면 비기능 요구사항일 가능성이 크다. ④는 문장에 기능 동사가 없는데도 시스템 관련 내용이라는 이유만으로 기능 요구사항으로 오인하기 쉽다.

다시 나오면 이렇게 판단

선지가 시스템의 행위나 서비스 내용을 말하면 기능 요구사항으로 판단하고, 선지가 품질 수준이나 운영 제약을 말하면 비기능 요구사항으로 판단한다.

암기 포인트

무엇을 한다는 기능 요구사항이고, 어느 정도로 잘해야 한다는 비기능 요구사항이다.

NoteBase

문 8. 애자일 프로세스 모델에 해당하지 않는 것은?

① 동적 시스템 개발 방법(DSDM)

② 마르미(MaRMI)-RE

③ 스크럼(scrum)

④ 적응형 소프트웨어 개발(ASD)

정답 2번

출제 포인트

이 문제는 대표적인 애자일 프로세스 모델과 요구공학·재공학 계열 방법론을 구분하는 문제이다.

핵심 개념

DSDM은 짧은 개발 주기와 사용자 참여를 중시하는 애자일 개발 방법이다.
스크럼은 스프린트, 역할, 이벤트를 중심으로 반복 개발하는 대표적 애자일 프레임워크이다.
ASD는 변화 수용과 학습을 강조하는 애자일 개발 방법이다.
MaRMI-RE는 기존 시스템의 현대화·재공학과 관련된 방법론으로, 애자일 프로세스 모델 분류에 넣지 않는다.

선지별 해설

① O: 동적 시스템 개발 방법(DSDM)은 반복적 개발, 사용자 참여, timeboxing을 특징으로 하는 대표적 애자일 방법론이므로 애자일 프로세스 모델에 해당한다.
② X: 틀린 부분은 "애자일 프로세스 모델"이라는 분류이다. MaRMI-RE는 요구공학·레거시 시스템 현대화 및 재공학 쪽의 방법론이므로, 올바르게는 "애자일 프로세스 모델에 해당하지 않는다"이다.
③ O: 스크럼(scrum)은 스프린트 단위의 반복 개발과 팀 협업을 핵심으로 하는 대표적 애자일 프레임워크이므로 애자일 프로세스 모델에 해당한다.
④ O: 적응형 소프트웨어 개발(ASD)은 변화 대응, 협업, 학습을 중시하는 애자일 방법론이므로 애자일 프로세스 모델에 해당한다.

오답 포인트

DSDM, 스크럼, ASD는 모두 시험에서 자주 묶여 나오는 애자일 방법론이다. 반면 ②는 약어에 RE가 붙어 있어 애자일 요구사항 관리처럼 보이기 쉬우나, 실제 분류는 재공학·현대화 방법론이어서 이름만 보고 애자일로 오인하면 틀리기 쉽다.

NoteBase

다시 나오면 이렇게 판단

애자일 여부를 묻는 문제에서는 반복 개발, 짧은 주기, 고객 참여, 변화 대응, 협업 같은 키워드가 있으면 애자일 후보로 본다. 반대로 재공학, 현대화, 요구분석 전담, 레거시 전환 성격이 강하면 애자일 프로세스 모델과 구분한다.

암기 포인트

DSDM·스크럼·ASD는 애자일, MaRMI-RE는 재공학 계열이다.

NoteBase

문 9. 익스트림 프로그래밍의 테스트에 대한 설명으로 옳지 않은 것은?

① 코드를 작성하기 전에 테스트 케이스를 먼저 작성한다.

② 각 사용자 스토리에 대해 테스트 케이스를 작성한다.

③ 프로그램을 큰 단위로 나누어 릴리스 직전 테스트를 수행한다.

④ 자동화된 테스팅 도구 사용을 권장한다.

정답 3번

출제 포인트

이 문제는 익스트림 프로그래밍의 테스트 원칙과 전통적 후행 테스트 방식의 차이를 묻는 문제이다.

핵심 개념

익스트림 프로그래밍의 테스트는 코드를 다 만든 뒤 한 번에 검증하는 방식이 아니라, 짧은 반복 안에서 테스트를 먼저 만들고 계속 실행하는 방식이다.
테스트 우선 개발은 구현 전에 테스트 케이스를 작성하는 방식이고, 사용자 스토리별 인수 테스트는 고객 요구가 충족되는지를 확인하는 방식이다.
작은 릴리스와 지속적 테스트가 핵심이며, 릴리스 직전의 대규모 일괄 테스트는 XP의 성격과 다르다.

선지별 해설

① O
코드를 작성하기 전에 테스트 케이스를 먼저 작성하는 것은 XP의 테스트 우선 개발 원칙에 해당한다.

② O
XP에서는 각 사용자 스토리를 기준으로 인수 테스트를 마련하여 요구사항 충족 여부를 확인한다.

③ X
틀린 부분은 "큰 단위로 나누어"와 "릴리스 직전 테스트"이다.
XP는 프로그램을 작은 단위의 반복과 작은 릴리스로 개발하며, 테스트도 릴리스 직전에 몰아서 하지 않고 개발 전반에 걸쳐 지속적으로 수행한다.
올바르게는 "프로그램을 작은 단위로 반복 개발하고, 단위 테스트와 인수 테스트를 지속적으로 수행한다"이다.

④ O
XP는 반복적으로 실행 가능한 자동화된 테스트를 중시하므로 자동화된 테스팅 도구 사용을 권장한다.

NoteBase

오답 포인트

③은 전통적 개발의 대규모 후행 테스트 이미지를 XP에 섞어 놓은 선지이다. "큰 단위"와 "릴리스 직전"이라는 표현이 나오면 XP의 작은 릴리스, 지속적 테스트 원칙과 충돌한다.

다시 나오면 이렇게 판단

애자일과 XP 관련 선지는 테스트 시점과 릴리스 단위를 먼저 본다. 구현 전에 테스트를 만들고 짧은 반복마다 계속 검증하면 XP 쪽이고, 개발 말기에 한꺼번에 테스트하면 XP가 아니다.

암기 포인트

XP는 선테스트, 작은 릴리스, 지속적 자동 테스트가 핵심이다.

NoteBase

문 10. 유스케이스에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 개발자의 관점에서 요구사항을 정의한다.
ㄴ. 액터는 시스템의 외부대상이므로 개발 범위에 포함되지 않는다.
ㄷ. 유스케이스 명세서에는 비기능적 요구사항을 기술해서는 안 된다.
ㄹ. 유스케이스와 이를 이용하는 액터와의 관계는 연관관계를 이용하여 표현한다.

① ㄱ, ㄷ
② ㄴ, ㄹ
③ ㄱ, ㄷ, ㄹ
④ ㄴ, ㄷ, ㄹ

정답 2번

출제 포인트

이 문제는 유스케이스의 관점, 액터의 위치, 유스케이스 명세서의 기록 범위, 액터와 유스케이스의 관계 표현을 정확히 구분하는지를 묻는 문제이다.

핵심 개념

유스케이스는 액터의 관점에서 시스템이 제공해야 할 기능적 서비스를 기술한 것이다.
액터는 시스템 경계 밖에서 유스케이스와 상호작용하는 외부 주체이다.
유스케이스 명세서에는 기본 흐름뿐 아니라 제약사항이나 특별 요구사항 형태의 비기능 요구사항도 함께 적을 수 있다.
액터와 유스케이스의 연결은 유스케이스 다이어그램에서 연관관계로 표현한다.

NoteBase

선지별 해설

① X: ㄱ, ㄷ의 조합이므로 틀리다. ㄱ의 "개발자의 관점"이 틀린 표현이다. 유스케이스는 개발자 관점이 아니라 사용자나 외부 액터 관점에서 요구 기능을 정의한다. ㄷ의 "기술해서는 안 된다"도 틀린 표현이다. 비기능 요구사항은 유스케이스 명세서의 특별 요구사항이나 제약사항으로 기술할 수 있다.
② O: ㄴ, ㄹ의 조합이므로 옳다. ㄴ은 액터가 시스템 외부의 역할이므로 시스템 개발 범위 자체에는 포함되지 않는다는 설명으로 맞다. ㄹ도 액터와 유스케이스의 사용 관계를 유스케이스 다이어그램에서 연관관계로 연결한다는 설명이므로 맞다.
③ X: ㄱ, ㄷ, ㄹ의 조합이므로 틀리다. ㄹ은 맞지만 ㄱ과 ㄷ이 틀려서 전체 조합이 성립하지 않는다. ㄱ은 "개발자의 관점"을 "액터의 관점"으로 고쳐야 하고, ㄷ은 "기술해서는 안 된다"를 "기술할 수 있다"로 고쳐야 한다.
④ X: ㄴ, ㄷ, ㄹ의 조합이므로 틀리다. ㄴ과 ㄹ은 맞지만 ㄷ이 틀려서 전체 조합이 성립하지 않는다. ㄷ의 틀린 표현은 "기술해서는 안 된다"이며, 올바르게는 비기능 요구사항도 유스케이스 명세서에 포함될 수 있다.

오답 포인트

유스케이스를 기능 목록 정도로만 외우면 ㄱ의 "개발자의 관점"을 지나치기 쉽고, 유스케이스 다이어그램만 떠올리면 ㄷ에서 명세서에 비기능 요구사항을 적지 않는다고 오해하기 쉽다. 이 문항은 관점과 문서화 범위를 바꿔 넣는 함정을 사용하였다.

다시 나오면 이렇게 판단

유스케이스 문항은 먼저 "누구 관점에서 기능을 본 것인가"를 확인하고, 다음으로 액터가 시스템 안인지 밖인지, 마지막으로 명세서가 기능 흐름만 적는지 제약과 특별 요구사항까지 담는지를 순서대로 점검하면 된다. 조합형 보기에서는 개별 진술의 진위를 확정한 뒤 조합을 고르면 된다.

암기 포인트

유스케이스는 액터 관점, 액터는 시스템 밖, 명세서는 비기능 요구사항도 포함 가능, 액터-유스케이스는 연관관계이다.

NoteBase

문 11. 다음 설명에 해당하는 테스팅은?

◯ 소프트웨어의 설계 한계를 넘어서는 부하를 준다.
◯ 시스템의 장애 행동을 테스트한다.
◯ 시스템이 최대 부하가 걸렸을 때만 보일 수 있는 결함을 드러나게 한다.

① 보안(security) 테스팅
② 회귀(regression) 테스팅
③ 스트레스(stress) 테스팅
④ 조합(combinatorial) 테스팅

정답 3번

출제 포인트

이 문제는 스트레스 테스트와 보안 테스트, 회귀 테스트, 조합 테스트의 목적 차이를 구분하는 문제이다.

핵심 개념

스트레스 테스트는 설계 한계를 넘는 과부하나 극한 조건에서 시스템의 안정성과 장애 대응을 확인하는 테스트이다.
보안 테스트는 비인가 접근, 취약점, 공격 저항성처럼 보안 위협 대응을 확인하는 테스트이다.
회귀 테스트는 변경 이후 기존 기능이 깨지지 않았는지를 확인하는 테스트이다.
조합 테스트는 입력값이나 조건의 다양한 조합에서 결함이 생기는지를 확인하는 테스트이다.

선지별 해설

① X: 틀린 부분은 보안 테스트로 보는 판단이다. 제시문은 보안 취약점이나 공격 대응이 아니라 설계 한계를 넘는 부하와 장애 행동을 말하므로 보안 테스트가 아니다. 올바르게는 보안 테스트는 접근 통제, 인증, 취약점 같은 보안 요소를 검증한다.
② X: 틀린 부분은 회귀 테스트로 보는 판단이다. 회귀 테스트는 수정이나 배포 뒤 기존 기능의 정상 동작 여부를 다시 확인하는 것이므로 최대 부하 상황과는 기준이 다르다. 올바르게는 회귀 테스트는 변경 후 부작용 여부를 점검한다.
③ O: 소프트웨어의 설계 한계를 넘는 부하를 주고, 시스템의 장애 행동을 테스트하며, 최대 부하에서만 드러나는 결함을 찾는다는 설명은 모두 스트레스 테스트의 특징에 해당한다.
④ X: 틀린 부분은 조합 테스트로 보는 판단이다. 조합 테스트는 여러 입력 조건의 조합을 체계적으로 줄여 결함을 찾는 방식이지, 과부하를 걸어 한계 상황을 보는 테스트가 아니다. 올바르게는 조합 테스트는 입력 조합에 따른 상호작용 결함을 검증한다.

NoteBase

오답 포인트

장애가 발생한다는 표현만 보고 보안 테스트나 일반 성능 테스트로 넓게 묶으면 오답이 된다. 이 문항의 핵심 단어는 설계 한계를 넘는 부하, 장애 행동, 최대 부하에서만 드러나는 결함이다.

다시 나오면 이렇게 판단

문장에 한계 초과 부하, 극한 상황, 장애 유발, 최대 부하 시 결함이 보이면 스트레스 테스트로 판단한다. 변경 후 재검사는 회귀 테스트, 취약점 검사는 보안 테스트, 입력 조합 검사는 조합 테스트로 구분한다.

암기 포인트

스트레스는 한계 초과 부하, 회귀는 변경 후 재검사, 보안은 취약점, 조합은 입력 조합이다.

NoteBase

문 12. <보기 1>의 디자인 패턴 분류와 <보기 2>의 디자인 패턴을 바르게 연결한 것은?

<보기 1>
ㄱ. 생성 패턴
ㄴ. 구조 패턴
ㄷ. 행위 패턴
<보기 2>
A. Bridge 패턴
B. Singleton 패턴
C. Interpreter 패턴

① ㄱ: A, ㄴ: B, ㄷ: C
② ㄱ: B, ㄴ: A, ㄷ: C
③ ㄱ: B, ㄴ: C, ㄷ: A
④ ㄱ: C, ㄴ: A, ㄷ: B

정답 2번

출제 포인트

이 문제는 GoF 디자인 패턴을 생성 패턴, 구조 패턴, 행위 패턴으로 정확히 분류할 수 있는지를 묻는 문제이다.

핵심 개념

생성 패턴은 객체 생성 방식을 다루어 생성 과정을 유연하게 만드는 분류이며 키워드는 객체 생성이다.
구조 패턴은 클래스나 객체를 조합해 더 큰 구조를 만드는 분류이며 키워드는 조합과 연결이다.
행위 패턴은 객체 사이의 책임 분배와 상호작용 방식을 다루는 분류이며 키워드는 메시지와 행위이다.
Singleton은 생성 패턴으로 인스턴스를 하나만 만들고 전역 접근을 제공한다.
Bridge는 구조 패턴으로 추상화와 구현을 분리해 독립적으로 확장할 수 있게 한다.
Interpreter는 행위 패턴으로 문법 규칙을 클래스로 표현해 해석 동작을 수행한다.

NoteBase

선지별 해설

① X
ㄱ을 A로 연결했는데 Bridge는 구조 패턴이므로 생성 패턴과 맞지 않는다. 틀린 표현은 생성 패턴-Bridge 연결이며 올바르게는 생성 패턴-Singleton이다.
ㄴ을 B로 연결했는데 Singleton은 생성 패턴이므로 구조 패턴과 맞지 않는다. 올바르게는 구조 패턴-Bridge이다.
ㄷ을 C로 연결한 부분만 맞다.

② O
ㄱ-B는 생성 패턴-Singleton 연결로 맞다.
ㄴ-A는 구조 패턴-Bridge 연결로 맞다.
ㄷ-C는 행위 패턴-Interpreter 연결로 맞다.
세 분류가 모두 정확히 대응한다.

③ X
ㄱ-B는 맞지만 ㄴ을 C로 연결한 부분이 틀리다. 틀린 표현은 구조 패턴-Interpreter 연결이며 올바르게는 구조 패턴-Bridge이다.
ㄷ을 A로 연결한 부분도 틀리다. Bridge는 행위 패턴이 아니라 구조 패턴이므로 ㄷ은 C가 되어야 한다.

④ X
ㄱ을 C로 연결했는데 Interpreter는 행위 패턴이므로 생성 패턴과 맞지 않는다. 올바르게는 ㄱ-B이다.
ㄴ-A는 맞지만 ㄷ을 B로 연결한 부분이 틀리다. Singleton은 생성 패턴이므로 행위 패턴 자리에 올 수 없고 ㄷ은 C이다.

오답 포인트

Bridge와 Interpreter처럼 이름만 보고 기능을 연상하면 구조 패턴과 행위 패턴을 바꾸기 쉽다. 또 Singleton은 구현 형태가 단순해서 구조 패턴처럼 착각하기 쉽지만 핵심은 객체 생성 통제이므로 생성 패턴이다.

다시 나오면 이렇게 판단

패턴 이름을 보면 먼저 무엇을 다루는지 기준 축을 잡는다. 객체를 어떻게 만들면 생성 패턴, 객체를 어떻게 조합하면 구조 패턴, 객체가 어떻게 상호작용하면 행위 패턴으로 판단된다. 그다음 대표 패턴을 분류표에 대입하면 된다.

암기 포인트

Singleton은 생성, Bridge는 구조, Interpreter는 행위로 묶인다.

NoteBase

문 13. 응집도를 강한 것부터 순서대로 나열할 때, ㉠ ~ ㉣에 들어갈 용어를 바르게 연결한 것은?

기능적 응집 - 순차적 응집 - (㉠) - (㉡) - (㉢) - (㉣) - 우연적 응집

① ㉠: 절차적 응집, ㉡: 교환적 응집, ㉢: 시간적 응집, ㉣: 논리적 응집
② ㉠: 절차적 응집, ㉡: 교환적 응집, ㉢: 논리적 응집, ㉣: 시간적 응집
③ ㉠: 교환적 응집, ㉡: 절차적 응집, ㉢: 시간적 응집, ㉣: 논리적 응집
④ ㉠: 논리적 응집, ㉡: 절차적 응집, ㉢: 교환적 응집, ㉣: 시간적 응집

정답 3번

출제 포인트

이 문제는 모듈 응집도의 강도를 강한 것부터 약한 것까지 순서대로 배열할 수 있는지를 묻는 문제이다.

올바른 순서

기능적 응집 → 순차적 응집 → 교환적 응집 → 절차적 응집 → 시간적 응집 → 논리적 응집 → 우연적 응집이다.
기능적 응집은 하나의 명확한 기능을 수행하므로 가장 강하다.
순차적 응집은 앞 단계의 출력이 다음 단계의 입력이 되므로 기능적 다음으로 강하다.
교환적 응집은 동일한 자료를 중심으로 관련 작업이 묶인 경우이다.
절차적 응집은 같은 처리 순서를 따른다는 공통점만 있다.
시간적 응집은 같은 시점에 수행된다는 이유로 묶인다.
논리적 응집은 논리적으로 유사한 기능을 한 모듈에 넣은 형태라 실제 결합 근거가 더 약하다.
우연적 응집은 서로 관련 없는 요소가 우연히 묶인 경우이므로 가장 약하다.

선지별 해설

1번 X. ㄱ을 절차적 응집으로 둔 것이 틀렸다. 순차적 응집 다음은 동일한 자료를 매개로 묶이는 교환적 응집이다. 올바르게는 교환적 응집 → 절차적 응집 → 시간적 응집 → 논리적 응집이다.
2번 X. ㄱ과 ㄴ이 서로 바뀌었고, ㄷ과 ㄹ도 바뀌었다. 절차적 응집은 교환적 응집보다 약하고, 시간적 응집은 논리적 응집보다 강하다. 올바르게는 교환적 응집 → 절차적 응집 → 시간적 응집 → 논리적 응집이다.
3번 O. 순차적 응집 다음에 교환적 응집, 그다음 절차적 응집, 시간적 응집, 논리적 응집이 오는 배열이 맞다. 자료 공유의 관련성이 절차 공유보다 강하고, 시간 기준의 묶음이 논리적 분기 묶음보다 강하다는 점을 반영한 순서이다.
4번 X. 논리적 응집을 너무 앞에 둔 것이 틀렸다. 논리적 응집은 유사 기능을 한곳에 모은 수준이라 시간적 응집보다도 약하다. 올바르게는 교환적 응집 → 절차적 응집 → 시간적 응집 → 논리적 응집이다.

NoteBase

오답 포인트

교환적 응집과 절차적 응집의 위치, 시간적 응집과 논리적 응집의 위치를 뒤바꾸기 쉽다는 점이 함정이다. 같은 자료를 공유하는 관계는 같은 순서만 따르는 관계보다 강하고, 같은 시점에 수행된다는 공통성은 단지 논리적으로 비슷한 기능을 묶는 것보다 강하다.

다시 나오면 이렇게 판단

응집도 순서 문제는 하나의 기능 수행인지, 앞뒤 단계의 입출력 의존인지, 같은 자료를 쓰는지, 같은 절차만 따르는지, 같은 시점인지, 단지 유사 기능인지의 기준으로 내려가며 판단하면 된다.

암기 포인트

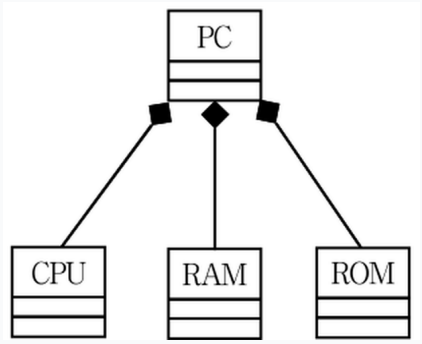
응집도는 기능 → 순차 → 교환 → 절차 → 시간 → 논리 → 우연 순으로 약해진다.

NoteBase

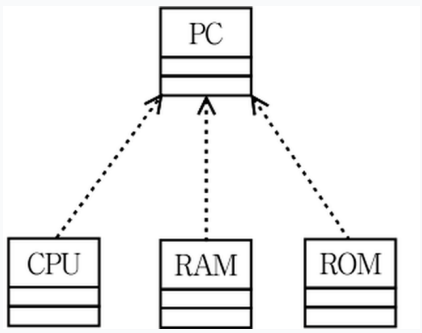
문 14. 다음 코드와 부합하는 클래스 간의 관계로 옳은 것은?

```
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();
 }
....
}
```

①

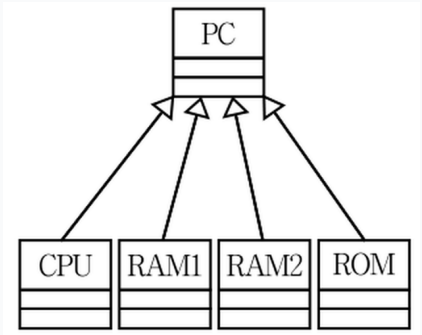


②

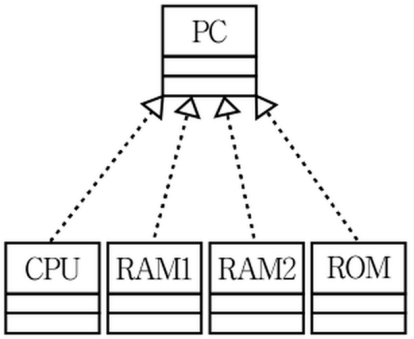


NoteBase

③



④



정답 1번

출제 포인트

이 문제는 코드에서 `new`로 직접 생성한 객체와 배열 선언을 보고 PC가 CPU, RAM, ROM을 자신의 구성 요소로 포함하는 합성 관계를 도식으로 판별하는 문제이다.

NoteBase

선지별 해설

① O
PC 생성자에서 `CPU`, `RAM[2]`, `ROM`을 직접 생성하므로 PC가 부품 객체를 강하게 소유하는 합성 관계로 보는 것이 맞다. 클래스 다이어그램은 PC와 CPU, RAM, ROM 사이의 포함 관계를 표현한 1번이 코드와 가장 잘 맞는다.

② X
틀린 표현은 점선 화살표로 의존 관계를 나타낸 부분이다. 이 코드는 객체를 잠깐 사용하는 수준이 아니라 필드로 보유하고 생성자에서 직접 생성하므로, 올바르게는 PC에서 CPU, RAM, ROM으로의 합성 관계로 나타내야 한다.

③ X
틀린 표현은 빈 삼각형으로 일반화 관계를 그린 부분과 `RAM1`, `RAM2`를 별도 클래스로 분리한 부분이다. `CPU`, `RAM`, `ROM`은 PC의 하위 클래스가 아니며, `RAM[2]`는 같은 RAM 클래스 요소가 두 개 있음을 뜻할 뿐 `RAM1`, `RAM2`라는 상속 클래스가 아님을 나타낸다. 올바르게는 PC가 CPU, RAM, ROM을 포함하는 합성 관계로 표시한다.

④ X
틀린 표현은 점선과 빈 삼각형으로 구현 또는 일반화에 가까운 관계를 그린 부분과 `RAM1`, `RAM2`를 독립 클래스처럼 나눈 부분이다. 이 코드는 인터페이스 구현 관계가 아니라 멤버 객체 포함 관계이므로, 올바르게는 PC와 CPU, RAM, ROM 사이의 합성 관계로 나타내야 한다.

오답 포인트

`new`가 보이면 관계를 먼저 판단해야 한다. 필드로 선언한 뒤 생성자에서 직접 생성하면 의존 관계보다 합성 관계로 보는 것이 맞는데, 점선 화살표나 빈 삼각형의 UML 기호만 외워 두고 고르면 ②, ③, ④에 끌리기 쉽다. 또 `RAM[2]`를 보고 `RAM1`, `RAM2`라는 별도 클래스로 착각하는 것도 대표적인 함정이다.

다시 나오면 이렇게 판단

필드 선언과 생성자 코드를 함께 본다. 객체를 멤버로 보유하고 생성자에서 직접 `new`로 만들면 합성 관계이고, 메서드 내부에서 일시적으로 참조만 하면 의존 관계이다. 배열 표기는 클래스의 상속 분리가 아니라 같은 클래스의 다중성 또는 개수 표현으로 판단한다.

암기 포인트

멤버 필드 + 생성자 `new`는 합성, 점선 화살표는 의존, 빈 삼각형은 상속 또는 구현이다.

NoteBase

문 15. UML 시퀀스(sequence) 다이어그램의 구성요소로 옳지 않은 것은?

① 포크(fork)

② 생명선(lifeline)

③ 메시지(message)

④ 복합적 부분(combined fragment)

정답 1번

출제 포인트

이 문제는 UML 시퀀스 다이어그램의 구성요소와 활동 다이어그램의 제어 노드를 구분하는지 묻는 문제이다.

핵심 개념

시퀀스 다이어그램은 객체 사이의 상호작용을 시간 순서대로 표현하는 다이어그램이며, 대표 요소로 생명선, 메시지, 활성 구간, 복합적 부분이 있다.
포크는 활동 다이어그램에서 하나의 흐름을 여러 병행 흐름으로 나누는 제어 노드이다.

선지별 해설

① X. 틀린 부분은 '포크'이다. 포크는 시퀀스 다이어그램 요소가 아니라 활동 다이어그램의 병행 분기 제어 노드이다. 올바르게는 시퀀스 다이어그램의 구성요소로 생명선, 메시지, 활성 구간, 복합적 부분 등이 제시되어야 한다.
② O. 생명선은 객체나 참여자의 존재와 시간 경과를 세로선으로 나타내는 시퀀스 다이어그램의 기본 요소이다.
③ O. 메시지는 객체 사이에 주고받는 호출, 응답, 신호를 나타내므로 시퀀스 다이어그램의 핵심 구성요소이다.
④ O. 복합적 부분은 alt, opt, loop처럼 상호작용의 조건·반복·선택 구조를 묶어 표현하는 요소이므로 시퀀스 다이어그램의 구성요소가 맞다.

오답 포인트

UML 전체에서 쓰이는 용어라는 이유로 다이어그램 종류를 구분하지 않으면 헷갈리기 쉽다. ①은 UML 용어 자체는 맞지만 시퀀스 다이어그램이 아니라 활동 다이어그램 요소라는 점이 함정이다.

다시 나오면 이렇게 판단

보기의 용어를 먼저 어떤 UML 다이어그램에 속하는지 분류하면 된다. 객체 간 시간 순서의 상호작용을 나타내면 시퀀스 다이어그램, 병행 흐름의 분기와 합류를 나타내면 활동 다이어그램으로 판단된다.

암기 포인트

시퀀스는 생명선·메시지·복합적 부분, 포크는 활동 다이어그램의 분기 노드이다.

NoteBase

문 16. 역공학(reverse engineering) 프로세스에 대한 설명으로 옳지 않은 것은?

① 코드 난독화를 수행하면 역공학을 쉽게 할 수 있다.

② 제품의 추상화가 하위 수준에서 상위 수준으로 수행된다.

③ 소스 코드를 분석하여 이해하기 쉬운 다이어그램으로 변형한다.

④ 기존 시스템을 이해하기 위해 시스템을 구성하는 주요 요소들과 이들 사이의 관계를 파악한다.

정답 1번

출제 포인트

이 문제는 역공학의 목적과 진행 방향을 정확히 이해하는지를 묻는 문제이다.

핵심 개념

역공학은 이미 구현된 시스템을 분석하여 구성 요소와 그 관계를 파악하고, 소스 코드 같은 저수준 표현에서 설계 모델 같은 고수준 추상으로 올라가는 과정이다.
코드 난독화는 프로그램의 내부 구조를 숨겨 분석을 어렵게 만드는 기법이므로 역공학을 돕는 것이 아니라 방해한다.

선지별 해설

1번 X: 틀린 부분은 '역공학을 쉽게 할 수 있다'이다. 코드 난독화는 변수명, 제어 흐름, 구조를 알아보기 어렵게 만들어 프로그램 이해를 방해한다. 올바르게는 '코드 난독화는 역공학을 어렵게 만든다'이다.
2번 O: 역공학은 구현 결과물에서 설계나 구조를 추출하는 과정이므로 추상화가 하위 수준에서 상위 수준으로 수행된다.
3번 O: 역공학은 소스 코드를 분석하여 구조와 동작을 이해하기 쉬운 다이어그램이나 모델로 바꾸는 작업을 포함한다.
4번 O: 기존 시스템을 이해하기 위해 시스템의 주요 구성 요소와 요소 사이의 관계를 파악하는 것은 역공학의 대표적 목적이다.

오답 포인트

난독화와 역공학을 둘 다 '코드를 다루는 작업' 정도로만 보면 1번을 맞는 설명처럼 착각하기 쉽다. 그러나 역공학은 이해와 추출의 방향이고, 난독화는 그 이해를 막는 방향이라는 점이 정반대이다.

다시 나오면 이렇게 판단

역공학 관련 선지는 '기존 시스템을 이해하는가', '저수준에서 고수준으로 추상화하는가', '구조와 관계를 드러내는가'를 기준으로 본다. 반대로 구조를 숨기거나 분석을 어렵게 만드는 설명은 역공학의 성격과 어긋난다.

NoteBase

암기 포인트

역공학은 숨기는 작업이 아니라 분석하여 위로 끌어올리는 작업이다.

NoteBase

문 17. 다음 설명에 해당하는 것은?

비슷한 유형의 응용프로그램들을 위해 재사용이 가능한 아키텍처와 협력하는 소프트웨어 산출물의 통합된 집합으로, 특정 클래스의 재사용뿐만 아니라 응용프로그램을 위한 핵심 아키텍처를 제공하여 설계의 재사용을 지원한다.

① 컴포넌트(component)

② 웹서비스(web service)

③ 프레임워크(framework)

④ 클래스 라이브러리(class library)

정답 3번

출제 포인트

이 문제는 컴포넌트, 웹서비스, 프레임워크, 클래스 라이브러리의 재사용 단위와 제공 범위를 구별하는 문제이다.

핵심 개념

프레임워크는 유사한 응용프로그램군을 위해 핵심 아키텍처와 제어 흐름의 골격을 미리 제공하는 재사용 구조이다.
컴포넌트는 독립적으로 조합 가능한 기능 단위로서 특정 기능을 캡슐화해 재사용하는 자산이다.
웹서비스는 네트워크를 통해 표준 인터페이스로 호출하는 서비스 단위이다.
클래스 라이브러리는 필요한 클래스나 API를 호출해 쓰는 코드 재사용 집합이다.
구별 기준은 기능 조각을 주는지, 응용프로그램의 뼈대와 설계 구조까지 주는지이다.

선지별 해설

① X: 컴포넌트는 재사용 가능한 소프트웨어 부품이지만, 설명처럼 응용프로그램군 전체를 위한 핵심 아키텍처를 제공하는 통합 골격은 아니다. 틀린 표현은 "핵심 아키텍처를 제공하여 설계의 재사용을 지원"하는 부분이다. 올바르게는 "독립 배포와 조합이 가능한 기능 단위를 제공하여 구현의 재사용을 지원한다"이다.
② X: 웹서비스는 네트워크를 통해 기능을 제공하는 서비스 형태이지, 비슷한 응용프로그램들의 공통 구조와 설계 골격을 제공하는 개념은 아니다. 틀린 표현은 "재사용이 가능한 아키텍처와 협력하는 소프트웨어 산출물의 통합된 집합"이다. 올바르게는 "표준 프로토콜로 접근하는 서비스 인터페이스를 제공한다"이다.
③ O: 프레임워크는 특정 클래스 하나의 재사용을 넘어서, 유사한 응용프로그램을 만들 때 공통으로 쓰는 핵심 아키텍처와 협력 구조를 제공한다. 제어 흐름과 설계 골격까지 제공한다는 점이 제시문과 정확히 일치한다.
④ X: 클래스 라이브러리는 클래스들의 모음으로 필요한 클래스를 호출해 사용하는 재사용 자산이다. 그러나 응용프로그램 전체의 아키텍처나 협력 구조를 선제적으로 제공하지는 않는다. 틀린 표현은 "응용프로그램을 위한 핵심 아키텍처를 제공"하는 부분이다. 올바르게는 "클래스 수준의 코드 재사용을 제공한다"이다.

NoteBase

오답 포인트

프레임워크와 클래스 라이브러리는 모두 재사용 자산이라는 점 때문에 헷갈리기 쉽다. 그러나 라이브러리는 필요한 코드를 가져다 쓰는 수준이고, 프레임워크는 응용프로그램의 뼈대와 협력 방식까지 정해 준다는 점이 핵심 차이이다.

다시 나오면 이렇게 판단

문장에 "유사한 응용프로그램군", "핵심 아키텍처", "설계의 재사용"이 함께 나오면 프레임워크를 먼저 판단한다. 기능 단위이면 컴포넌트, 네트워크 서비스이면 웹서비스, 클래스 집합이면 클래스 라이브러리로 구분한다.

암기 포인트

프레임워크는 응용프로그램의 뼈대와 흐름을 주고, 라이브러리와 컴포넌트는 기능 요소를 준다.

NoteBase

문 18. CRC 카드에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 클래스의 연산에 대한 상세 알고리즘 설계를 위해 이용되는 도구이다.
ㄴ. 클래스의 연산과 속성을 파악하는 데 이용되는 도구이다.
ㄷ. 협력 클래스를 파악하는 데 이용되는 도구이다.
ㄹ. 카드의 상단에 클래스 이름, 왼쪽 열에 협력자, 오른쪽 열에 클래스 책임을 나열한다.

① ㄱ, ㄹ
② ㄴ, ㄷ
③ ㄱ, ㄴ, ㄷ
④ ㄴ, ㄷ, ㄹ

정답 2번

출제 포인트

이 문제는 CRC 카드의 목적과 카드 구성 요소를 정확히 구분하는지를 묻는 문제이다.

핵심 개념

CRC 카드는 Class-Responsibility-Collaborator 카드로, 객체지향 분석 초기에 클래스가 무엇을 알고 무엇을 하는지와 어떤 다른 클래스와 협력하는지를 정리하는 도구이다.
클래스 이름은 상단에 적고, 책임은 왼쪽, 협력자는 오른쪽에 적는다.
책임은 클래스가 수행하는 행동과 유지하는 정보의 관점이므로 연산과 속성을 파악하는 단서가 되지만, 세부 알고리즘을 설계하는 단계의 도구는 아니다.

NoteBase

선지별 해설

① X: ㄱ과 ㄹ이 포함되어 있어 틀리다. ㄱ의 틀린 부분은 "연산에 대한 상세 알고리즘 설계"이다. CRC 카드는 책임과 협력 관계를 찾는 개념적 도구이며, 실제로는 클래스의 역할과 상호작용을 정리하는 데 쓴다. ㄹ의 틀린 부분은 "왼쪽 열에 협력자, 오른쪽 열에 클래스 책임"이다. 실제로는 왼쪽에 클래스 책임, 오른쪽에 협력자를 적는다.
② O: ㄴ과 ㄷ만 묶었으므로 옳다. ㄴ은 CRC 카드의 책임이 클래스의 행동과 보유 정보로 이어지므로 연산과 속성을 파악하는 데 활용된다는 점에서 맞다. ㄷ은 CRC 카드가 어떤 클래스와 협력하는지를 함께 적는 도구이므로 맞다.
③ X: ㄱ이 포함되어 있어 틀리다. ㄱ의 틀린 부분은 "상세 알고리즘 설계"이다. CRC 카드는 알고리즘 상세화보다 책임 분담과 협력 클래스 식별이 목적이다. 올바르게는 "클래스의 책임과 협력 관계를 도출하는 데 이용되는 도구"이다.
④ X: ㄹ이 포함되어 있어 틀리다. ㄹ의 틀린 부분은 책임과 협력자의 위치를 뒤바꾼 점이다. 올바르게는 "카드 상단에 클래스 이름, 왼쪽에 클래스 책임, 오른쪽에 협력자를 나열한다"이다.

오답 포인트

CRC의 C, R, C를 외우지 않고 막연히 객체지향 도구로만 기억하면 ㄱ처럼 상세 설계 도구로 착각하거나 ㄹ처럼 책임과 협력자의 위치를 바꿔 고르기 쉽다. 또 책임이 행동과 보유 정보까지 포괄한다는 점을 놓치면 ㄴ을 과도하게 좁게 보아 틀리기 쉽다.

다시 나오면 이렇게 판단

CRC가 나오면 먼저 Class-Responsibility-Collaborator의 세 요소를 확인한다. 그다음 분석 초기의 책임 배분 도구인지, 구현 단계의 상세 알고리즘 설계 도구인지를 구분하면 된다. 카드 배치는 상단 클래스명, 왼쪽 책임, 오른쪽 협력자로 판단한다.

암기 포인트

CRC는 클래스명 위, 책임 왼쪽, 협력자 오른쪽이며, 목적은 상세 설계가 아니라 책임과 협력 정리이다.

NoteBase

문 19. 테스팅에서 프로그램의 실제 실행결과가 올바른 결과인지를 판단하는 메커니즘은?

① 테스트 하니스(test harness)

② 테스트 적합성 기준(test adequacy criteria)

③ 테스트 오라클(test oracle)

④ 테스트 종료 기준(test completion criteria)

정답 3번

출제 포인트

이 문제는 테스트 하니스, 테스트 적합성 기준, 테스트 오라클, 테스트 종료 기준의 역할을 구별할 수 있는지를 묻는다.

핵심 개념

테스트 하니스는 테스트 실행을 자동화하고 필요한 드라이버·스텁·로그 수집을 제공하는 실행 환경이다.
테스트 적합성 기준은 현재 테스트 집합이 충분한지를 판단하는 기준이다.
테스트 오라클은 실제 실행 결과가 기대 결과와 일치하는지를 판정하는 기준 또는 메커니즘이다.
테스트 종료 기준은 테스트를 언제 끝낼지를 정하는 조건이다.

선지별 해설

1번 X: 테스트 하니스는 테스트를 실행하고 주변 환경을 제공하는 도구 집합이다. 틀린 부분은 '올바른 결과인지 판단하는 메커니즘'이라는 역할 부여이다. 실제로는 실행 자동화와 결과 수집을 담당하고, 정오 판정은 테스트 오라클이 담당한다.
2번 X: 테스트 적합성 기준은 테스트 케이스가 얼마나 충분한지, 어떤 범위까지 커버했는지를 판단하는 기준이다. 틀린 부분은 '실제 실행결과가 올바른 결과인지'라는 표현이다. 실제로는 테스트의 충분성을 판단하는 기준이다.
3번 O: 테스트 오라클은 프로그램의 실제 실행 결과를 기대 결과와 비교하여 정답 여부를 판단하는 메커니즘이다. 발문의 '실제 실행결과가 올바른 결과인지'를 직접 판정하므로 정답이다.
4번 X: 테스트 종료 기준은 결함 발견률, 일정, 커버리지 목표 등으로 테스트를 종료할 시점을 정하는 기준이다. 틀린 부분은 '실행결과가 올바른지 판단'이라는 기능 설명이다. 실제로는 테스트를 언제 멈출지를 결정하는 기준이다.

오답 포인트

하니스, 적합성 기준, 종료 기준은 모두 테스트 관리에 쓰이는 용어이므로 역할이 섞이기 쉽다. 이 문항은 '정오 판정' 기능을 오라클에만 연결해야 한다는 점을 놓치면 1번이나 2번으로 흔들리기 쉽다.

NoteBase

다시 나오면 이렇게 판단

선지의 핵심 동사가 무엇인지 먼저 본다. 실행을 돕는지, 충분성을 재는지, 정오를 판정하는지, 종료 시점을 정하는지로 나누면 용어가 바로 구분된다.

암기 포인트

오라클은 정오 판정, 하니스는 실행 환경, 적합성 기준은 충분성, 종료 기준은 중단 시점이다.

NoteBase

문 20. 다음 표는 프로젝트를 수행하는 데 필요한 작업, 소요 기간, 선행 작업을 나타낸 것이다. 작업 T5를 담당한 개발자가 이직하여 대체 인력을 확보하였으나 대체 인력의 교육에 15일이 소요되어, 작업 T5는 소요 기간이 35일로 변경되었다. 프로젝트를 완료하기까지 필요한 최소 소요 기간은 개발자 이직 전보다 얼마나 증가하는가?

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

정답 1번

출제 포인트

이 문제는 작업 선후관계로 임계경로를 찾고, 특정 작업 기간 증가가 전체 프로젝트 최소 완료 기간에 얼마나 반영되는지를 계산하는 문제이다.

조건 정리

주어진 값: 기존 T5는 20일, 변경 후 T5는 35일이므로 T5는 15일 증가한다.
구해야 할 값: 개발자 이직 전후의 프로젝트 전체 최소 소요 기간 차이이다.
주의할 조건: 어떤 작업의 증가분이 그대로 프로젝트 증가분이 되는 것은 아니며, 변경된 작업이 최종 임계경로를 얼마나 늦추는지를 따져야 한다.

NoteBase

사용할 공식

프로젝트 최소 완료 기간은 마지막 작업들의 완료 시점 중 최댓값이다.
이 원리를 쓰는 이유: 이 문제는 각 경로의 완료 시점을 비교해 가장 늦게 끝나는 경로를 찾아야 전체 증가 일수를 판단할 수 있기 때문이다.

계산 과정

1. 변경 전 완료 시점을 계산하면 T1→T2→T4→T7→T8 경로가 65일로 가장 길다.
2. 변경 전 T3→T5→T6 경로는 55일이고, T3→T5→T8 경로는 50일이다.
3. T5가 35일로 늘어나면 T3→T5→T6 경로는 70일, T3→T5→T8 경로는 65일이 된다.
4. 변경 후 전체 최소 완료 기간은 70일이므로 기존 65일에서 5일 증가한다.

선지별 해설

① O
- 변경 전 전체 최소 소요 기간은 65일이고 변경 후는 70일이므로 5일 증가한다.
- T5는 15일 늘었지만 원래 전체 임계경로가 아니었으므로 증가분 15일이 그대로 프로젝트 증가분이 되지 않는다.

② X
- 10일 증가는 계산 결과와 맞지 않는다. 변경 후 최종 완료일은 70일이므로 증가분은 5일이다.
- 틀린 표현은 '프로젝트 전체 최소 소요 기간이 10일 증가한다'이고, 올바르게는 '5일 증가한다'이다.
- 10일은 T5가 늘어난 뒤 T8 완료 시점만 어림잡아 비교하거나 T6 경로가 70일까지 늘어나는 점을 놓칠 때 나오는 오답이다.

③ X
- 15일은 T5 자체의 증가분일 뿐 프로젝트 전체 증가분이 아니다.
- 틀린 표현은 'T5가 15일 늘었으니 프로젝트도 15일 증가한다'이고, 올바르게는 '프로젝트는 65일에서 70일로만 늘어난다'이다.
- 임계경로 분석 없이 변경된 작업의 증가분을 그대로 답으로 쓰면 이 오답에 걸린다.

④ X
- 35일은 변경된 T5의 소요 기간 자체이지, 프로젝트 증가 기간이 아니다.
- 틀린 표현은 '프로젝트가 35일 증가한다'이고, 올바르게는 'T5 기간이 35일로 바뀌었고 프로젝트 증가분은 5일이다'이다.
- 변경 후 작업 기간과 전체 일정 증가분을 구분하지 못하면 나오는 오답이다.

오답 포인트

이 문제의 함정은 T5의 증가분 15일이나 변경 후 기간 35일을 그대로 프로젝트 증가분으로 착각하는 데 있다. 전체 일정 문제는 반드시 최장 경로가 어떻게 바뀌는지를 다시 계산해야 한다.

NoteBase

다시 나오면 이렇게 판단

프로젝트 일정 문제는 선행 작업으로 각 작업의 완료 시점을 먼저 구하고, 변경 전 임계경로와 변경 후 임계경로를 각각 비교해 전체 완료일의 차이만 답으로 판단한다.

암기 포인트

작업 증가분과 프로젝트 증가분은 같지 않으며, 답은 항상 변경 후 임계경로의 완료일 차이이다.

NoteBase