[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

[기출 예상 문제]
1. 소프트웨어 테스트 프로세스에 대한 설명으로 가장 적절한 것은?

① 계획 수립 단계: 테스트 시나리오 및 케이스 관리 프로세스
② 결과 단계: 형상 및 배포 관리 프로세스
③ 수행 단계: 테스트 결과 검토 프로세스
④ 계획 수립 단계: 테스트 환경 관리 프로세스

[기출 예상 문제]
2. 애플리케이션 테스트 프로세스 유형 중 테스트 수행 관리 후에 수행하는 관리 프로세스로 가장 거리가 먼 것은?

① 애플리케이션 테스트 환경 관리 프로세스
② 코드 커버리지 관리 프로세스
③ SQL 성능 최적화 관리 프로세스
④ Data 모델 형상 관리 프로세스

[기출 예상 문제]
3. 소프트웨어 테스트 기본 원칙으로 가장 거리가 먼 것은?

① 테스팅은 결함이 존재함을 밝히는 활동이다.
② 완벽한 테스팅을 해야한다.
③ 테스팅은 정황(Context)에 의존한다.
④ 테스팅은 개발 초기에 시작해야 한다.

[기출 예상 문제]
4. 동일한 테스트 케이스로 테스트를 반복적으로 수행하면 나중에는 더 이상 새로운 결함을 찾아내지 못한다는 의미를 가진 용어로 가장 적절한 것은?

① 파레토 법칙(Pareto Principle)
② 오류-부재의 궤변(Absence of Errors Fallacy)
③ 결함 집중(Defect Clustering)
④ 살충제 패러독스(Peticide Paradox)

[기출 예상 문제]
5. 다음은 무엇에 대한 설명인가?

> 이 세상에 어떠한 소프트웨어도 오류가 완벽하게 해소된 제품을만들어 낼 수 없음을 의미한다. 테스트는오류를 100% 없애는것이 목적이 아니라 일정 수준이하로 줄이는 것이 목적이다.

① 살충제 패러독스(Pesticide Paradox)
② 결함 집중(Defect Clustering)
③ 오류-부재의 궤변(Absence of Errors Fallacy)
④ 정황에 의존(Testing is context dependent)

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

[이전 기출 문제]
6. 다음 중 테스트의 특징이 아닌 것은?

① 테스트는 프로그램 개발팀에서 자체적으로 실시한다.
② 완벽한 테스트는 불가능하다.
③ 테스트는 오류의 유입을 방지할 수 있다.
④ 테스트는 오류를 발견하려고 프로그램을 수행시키는 것이다.

[기출 예상 문제]
7. 다음은 무엇에 대한 설명인가?

특정한 프로그램의 일부분 또는 경로에 따라 수행하거나, 특정한 요구사항을 준수하는지 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과로 구성된 테스트 항목의 명세서를 말한다.

① 경로 기반 테스팅　② 테스트 케이스
③ 요구사항 명세서　④ 설계 테스트 명세서

[이전 기출 문제]
8. 모든 테스트 케이스를 위한 예상된 결과를 무엇이라 하는가?

① 테스트 하니스(Test harness)
② 테스트 오라클(Test Oracle)
③ 테스트 스터브(Test Stub)
④ 테스트 드라이브(Test Drive)

[기출 예상 문제]
9. 다음은 무엇에 대한 설명인가?

- 테스트 케이스의 일반적 형식을 결정하고, 테스트 케이스 분류 방법을 결정한다. - 테스트 절차, 장비, 도구, 테스트 문서화 방법을 결정한다.

① 테스트 계획 검토 및 자료 확보
② 테스트 구조 설계 및 테스트 방법 결정
③ 테스트 케이스 정의
④ 테스트 요구사항 정의

[기출 예상 문제]
10. 테스트 케이스 작성 절차로 가장 적절한 것은?

ㄱ. 테스트 요구사항 정의 ㄴ. 테스트 케이스 타당성 확인 및 유지보수 ㄷ. 테스트 계획 검토 및 자료 확보 ㄹ. 위험 평가 및 우선순위 결정 ㅁ. 테스트 케이스 정의 ㅂ. 테스트 구조 설계 및 테스트 방법 결정

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

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

[기출 예상 문제]
11. 특정한 몇 개의 입력 값에 대해서만 기대하는 결과를 제공해 주는 테스트 오라클은?

① 참(True) 오라클
② 샘플링(Sampling) 오라클
③ 휴리스틱(Heuristic) 오라클
④ 일관성 검사(Consistent) 오라클

[기출 예상 문제]
12. 테스트오라클에 대한 설명으로 가장 거리가 먼 것은?

① 테스트의 결과가 참인지 거짓인지를 판단하기 위한 활동이다.
② 참(True) 오라클은 모든 입력 값에 대하여 기대하는 결과를 생성함으로써 발생된 오류를 모두 검출할 수 있는 오라클이다.
③ 휴리스틱(Heuristic) 오라클은 샘플링 오라클을 개선한 오라클로, 특정 입력 값에 대해 올바른 결과를 제공하고, 나머지 값들에 대해서는 추정으로 처리하는 오라클이다.
④ 추정 오라클은 항공기, 임베디드, 발전소 소프트웨어 등 미션 크리티컬한 업무에 적용한다.

[기출 예상 문제]
13. 다음 중 참 오라클을 사용하는 업무로 가장 거리가 먼 것은?

① 게임 ② 임베디드
③ 항공기 ④ 발전소

[기출 예상 문제]
14. 모듈 간의 인터페이스 연계를 검증하고 오류를 확인, 모듈 간의 상호 작용 및 연계 동작이 제대로 기능하는지 점검하는 테스트로 가장 적절한 것은?

① 단위 테스트 ② 통합 테스트
③ 시스템 테스트 ④ 인수 테스트

[기출 예상 문제]
15. V-모델과 테스트 레벨에 대한 설명으로 가장 거리가 먼 것은?

① 소프트웨어 설계 단계에서 통합 테스트를 진행한다.
② 소프트웨어 개발 시 세부 구현 클래스나 메서드에 대한 검증을 한다.
③ 시스템 테스트는 요구사항 분석 단계에서 진행한다.
④ 마지막 테스트는 고객이 최종적으로 요구사항 충족에 대한 검증 수행하는 인수테스트이다.

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

[이전 기출 문제]
16. 개발과정에서 실시되는 시험의 순서를 바르게 나열한 것은?

ㄱ. 모듈 내부적인 오류를 발견하기 위한 시험 ㄴ. 사용자의 요구사항을 만족하는지 판단하는 시험 ㄷ. 모듈간의 인터페이스 오류를 발견하기 위한 시험

① ㄱ-ㄴ-ㄷ ② ㄴ-ㄱ-ㄷ
③ ㄱ-ㄷ-ㄴ ④ ㄴ-ㄷ-ㄱ

[기출 예상 문제]
17. 테스트 시나리오 작성 시 유의점에 대한 설명으로 가장 거리가 먼 것은?

① 테스트 항목을 하나의 시나리오에 모두 작성하여야 한다.
② 고객의 요구사항과 설계 문서를 토대로 테스트 시나리오를 작성한다.
③ 각 테스트 항목은 테스트 데이터, 테스트 케이스, 예상 결과, 확인 등 항목을 포함하여 작성한다.
④ 시스템별, 모듈별, 항목별로 테스트 시나리오를 분리하여 작성한다.

[기출 예상 문제]
18. 테스트 시나리오 관리 프로세스로 가장 적절한 것은?

ㄱ. 테스트 케이스 도출 ㄴ. 테스트 케이스 피어리뷰 ㄷ. 테스트 케이스 검토 및 보안 ㄹ. 테스트 시나리오 도출 ㅁ. 테스트 케이스 업로드

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

[기출 예상 문제]
19. 테스트 시나리오 관리 프로세스에서 테스트 케이스가 만들어지면 품질 관리자와 테스트 리더가 검토하는 단계는?

① 테스트 시나리오 도출
② 테스트 케이스 피어리뷰
③ 테스트 케이스 업로드
④ 테스트 케이스 검토 및 보안

[기출 예상 문제]
20. 테스트 지식 체계(ISO 29119)와 관련 없는 것은?

① 키워드 기반 테스팅 ② 테스트 프로세스
③ 테스트 패턴 ④ 테스트 문서화

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트 케이스 설계]

[기출 예상 문제]
21. 테스트 지식 체계(ISO 29119)의 개념과 정의에 대한 설명으로 가장 거리가 먼 것은?

① 소프트웨어 테스팅 개념
② 위험 기반 테스팅(Risk-based Testing)
③ 동적 테스트 프로세스 문서
④ 조직과 프로젝트 관점에서의 소프트웨어 테스팅

[기출 예상 문제]
22. 사전 정의된 키워드 모음으로 테스트 케이스를 만드는 것으로, 테스트 단계에서 자연어 대신 키워드를 사용함으로써 테스트 케이스는 이해하기 쉽고 유지보수가 용이한 테스팅으로 가장 적절한 것은?

① 동적 테스팅 ② 키워드 주도 테스팅
③ 관리 자동화 테스팅 ④ 경험기반 테스팅

[기출 예상 문제]
23. 다음 중 테스트 설계 기법과 가장 관련이 없는 것은?

① 구조기반 설계 ② 경험기반 설계
③ 명세기반 설계 ④ 정적기반 설계

[기출 예상 문제]
24. ISO12119의 파트2 테스트 프로세스와 가장 거리가 먼 것은?

① 다계층 프로세스 모델
② 조직의 테스트 프로세스
③ 테스트 관리 프로세스
④ 테스트 서브 프로세스

[기출 예상 문제]
25. 테스트 지식 체계(ISO 29119)에 대한 설명으로 가장 적절한 것은?

① 파트1 – 테스트 문서화
② 파트2 – 개념과 정의
③ 파트3 – 테스트 프로세스
④ 파트4 – 테스트 기술

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

[기출 예상 문제]
1. 다음은 무엇에 대한 설명인가?

각 단계별 테스트 수행 후 발생한 결함의 재발 방지를 위해, 유사 결함 발견시 처리 시간 단축을 위해 결함을 추적하고 관리하는 활동이다.

① 테스트 지원 관리 ② 테스트 결함 관리
③ 테스트 자동화 관리 ④ 테스트 수행 관리

[기출 예상 문제]
2. 다음 중 결함 관리 도구가 아닌 것은?

① JIRA ② Mantis
③ Bugzilla ④ ATLASSIAN

[기출 예상 문제]
3. 결함 관리 도구에 대한 설명으로 가장 거리가 먼 것은?

① 결함 간에 연관관계 정보를 제공해야 한다.
② 결함 등록 및 상태 변경 시 담당자에게 이벤트 통보 기능이 되어야 한다.
③ 프로세스 및 워크플로우 변경이 불가능해야 한다.
④ 다중 프로젝트 지원이 가능해야 한다.

[기출 예상 문제]
4. 다음은 무엇에 대한 설명인가?

- 버그 관리, 개발 Task용 이슈 관리, 소스 코드 형상 관리 및 위키 기반의 문서 관리 도구이다. - Python으로 작성되어 있고, DB는 SQLite를 사용한다. - 웹에서 게시판 형태로 사용이 용이하며 티켓 발행으로 팀원 간 원활한 의사소통을 제공한다.

① Bugzilla ② Quality Center
③ Clear Quest ④ Trac

[기출 예상 문제]
5. V모델 관점에서 테스트 자동화 도구를 구분한 것으로 가장 거리가 먼 것은?

① 설계 단계 – 명세 기반 테스트 설계 도구
② 구현/테스트 단계 – 테스트 관리 도구
③ 설계 단계 – 코드 기반 테스트 설계 도구
④ 구현/테스트 단계 – 기능 테스트 수행 도구

[기출 예상 문제]
6. 프로그램을 수행하지 않고 분석하는 도구는?

① 정적 분석 도구 ② 동적 분석 도구
③ 커버리지 측정 도구 ④ 시뮬레이션 도구

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

[기출 예상 문제]
7. 테스트 작업 유형별 자동화 도구 기능 및 역할에 대한 설명으로 가장 적절한 것은?

① 코드 분석 도구: 원시 코드 문법적 정합성, 위반 사례 검출
② 요구사항 관리 도구: 테스트 관리 협업 환경 지원
③ 순서 검사 도구: 논리 그래프, 결함 체크
④ 성능 테스팅 도구: 반복 테스트 자동 수행, 캡처 및 리플레이 기능

[기출 예상 문제]
8. 다음 중 동적 테스팅 도구가 아닌 것은?

① 모니터링 도구 ② 단위 테스트 도구
③ 순서 검사 도구 ④ 테스트 실행 도구

[기출 예상 문제]
9. 소프트웨어가 실행 도중에만 발생하는 시간 의존성과 메모리 누수(Memory Leaks)와 같은 결함 발견에 활용하는 도구로 가장 적절한 것은?

① 소프트웨어 성능 및 모니터링 도구
② 소프트웨어 테스트 실행 및 로깅 지원 도구
③ 소프트웨어 정적 분석 지원 도구
④ 소프트웨어 테스트 관리 지원 도구

[기출 예상 문제]
10. 소프트웨어 테스트 관리 지원 도구에 대한 설명으로 가장 거리가 먼 것은?

① 실행된 테스트와 테스트 활동 관리를 지원한다.
② 테스트 실행 도구나 결함 추적 도구, 요구사항 관리 도구와의 인터페이스 역할을 한다.
③ 테스트 진행 상황에 대한 리포트 생성, 발견된 결함의 정량적인 분석을 지원한다.
④ 동적 테스트를 하기 전에 결함을 발견할 수 있도록 지원한다.

[기출 예상 문제]
11. 소프트웨어 테스트 실행 및 로깅 지원 도구에 대한 설명으로 가장 거리가 먼 것은?

① 스크립트 언어의 도움으로 저장된 입력값과 예상 결과를 이용하여 테스트를 실행하고 실제 결과와 비교한다.
② 오픈 소스에는 Selenium, STAF/STAX 등의 실행 자동화 도구가 있다.
③ 측정하고자 하는 특정 유형의 코드 구조(구문, 분기 등)가 몇 퍼센트 수행되었는가를 측정한다.
④ 특정 시스템 리소스의 사용량을 지속적으로 분석하고 확인한다.

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

[기출 예상 문제]

12. 소프트웨어 테스트 관리 지원 도구와 소프트웨어 정적 분석 지원 도구로 분류할 때 성격이 다른 것은?

① SVN
② Confluence
③ CppCheck
④ Testlink

[기출 예상 문제]

13. 다음은 빈 칸에 들어갈 용어로 가장 적절한 것은?

> 애플리케이션 통합 테스트 수행 방식은 (ㄱ) 방식과 (ㄴ) 방식으로 나누어 진다. (ㄱ) 방식은 개발된 컴포넌트 중 일부를 테스트하고 점차적으로 컴포넌트를 늘려가면서 테스트하는 방식이다. (ㄴ) 방식은 모든 컴포넌트를 사전에 통합하여 한꺼번에 테스트하는 방식이다.

	(ㄱ)	(ㄴ)
①	상향식 통합	하향식 통합
②	하향식 통합	상향식 통합
③	점증적	빅뱅
④	빅뱅	점증적

[기출 예상 문제]

14. 다음은 무엇에 대한 설명인가?

> - 리뷰 프로세스에 관한 정보를 저장, 리뷰 코멘트를 저장한다.
> - 동적 테스트를 하기 전에 결함을 발견할 수 있도록 지원한다. 특히, 코딩 표준을 지킬 것을 강제하고, 구조와 의존관계를 분석한다.
> - 소스 코드의 복잡도를 측정한다.

① 소프트웨어 테스트 관리 지원 도구
② 소프트웨어 정적 분석 지원 도구
③ 소프트웨어 테스트 실행 및 로깅 지원 도구
④ 소프트웨어 성능 및 모니터링 도구

[이전 기출 문제]

15. 통합 테스트에 대한 설명으로 옳지 않은 것은?

① 회귀 테스트는 복잡하고 시간이 중요한 프로젝트에 적용하면 효율적이다.
② 점증적인 통합은 빅뱅 방식에 비해 인터페이스를 완전하게 테스트할 기능성이 더 높다.
③ 객체지향 시스템의 드라이버는 사용자 인터페이스를 대체하여 사용함으로써 인터페이스의 구현 이전에 기능성 테스트를 수행 할 수 있다.
④ 객체지향 시스템의 스텁은 하나 이상의 협동 클래스들이 아직 완전히 구현되지 않은 상황에서 사용할 수 있다.

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

[이전 기출 문제]
16. 통합 테스트 단계에 적용할 기법을 바르게 선택한 것은?

① 명령어 처리 모듈을 먼저 구현하고 모듈 내부 기능을 시험하기 위하여 상향식(bottom-up)을 채택하였다.
② 시스템 구조도의 최하위층에 있는 모듈을 먼저 구현하고 테스트하기 위하여 하향식(top- down)을 채택하였다.
③ 특수하고 중요한 기능을 수행하는 최소 모듈 집합을 먼저 구현하고 테스트하기 위하여 연쇄식(threads)을 채택하였다.
④ 일정 계획의 융통성을 획득하기 위하여 동시식(big-bang)을 채택하였다.

[이전 기출 문제]
17. 다음과 같은 보기의 장점을 살릴 수 있는 통합 테스트 방식은?

- 전체 프로그램의 통합 개발이 시작되는 시점부터 개발 도중에 계속 이루어질 수 있다.
- 상위 모듈들을 먼저 테스트할 수 있어서 시스템의 전체 골격이나 기능을 조기에 테스트 해 볼 수 있다.
- 개발 도중에 기능이 완전하지는 않지만 프로그램을 사용해 볼 수는 있다.

① 상향식(bottom-up) ② 연쇄식(threads)
③ 하향식(top-down) ④ 빅뱅(big-bang)

[이전 기출 문제]
18. 통합테스트 방식 중 하향식 통합 방식에 대한 설명으로 옳지 않은 것은?

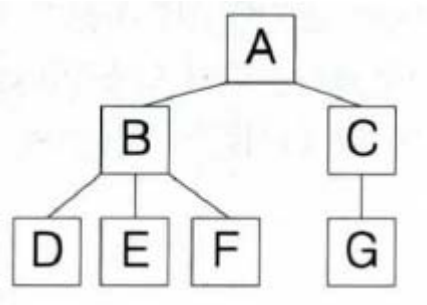
① 시스템 구조도의 가장 위 모듈로부터 아래 모듈로 내려가면서 통합되는 방법이다.
② 점증적 통합 형태 이므로 하드웨어 사용이 분산되고 오류의 원인을 찾기 쉽다.
③ 테스트 초기에 시험의 뼈대가 갖추어지지 않아 시스템에 사용해 볼 기회가 적다.
④ 시스템 계층구조와 상위층의 중요한 인터페이스를 조기에 시험할 수 있다.

[이전 기출 문제]
19. 하향식(top-down) 설계, 구현, 테스트에서 아직 완성되지 않은 하부 모듈을 대신하기 위해 사용되는 의미 없는 모듈을 무엇이라 하나?

① 테스트 하네스(test harness)
② 스텁(stub)
③ 테스트 케이스(test case)
④ 테스트 드라이버(test driver)

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

[이전 기출 문제]
20. 다음 컴포넌트 계층 구조의 소프트웨어에 대해 하향식 통합 시험을 수행하는 경우, 시험이 이루어지는 통합 단위 순서로 옳은것은?



① A – BC – DEFG – ABCDEFG
② A – BC – DEF – G
③ A – ABC – ABCDEFG
④ D – E – F – G – BDEF – CG –ABCDEFG

[이전 기출 문제]
21. 상향식(Bottom-Up) 통합테스트의 과정을 순서대로 나열한 것은?

ㄱ. 드라이버(Driver)라는 제어 프로그램의 작성 ㄴ. 낮은 수준의 모듈들을 클러스터(Cluster)로 결합 ㄷ. 클러스터의 검사 ㄹ. 드라이버를 제거하고 클러스터를 상위로 결합

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

[이전 기출 문제]
22. 다음 중 상향식 통합 테스트 방법을 틀리게 설명한 것은?

① 최하위 모듈을 먼저 통합하여 시험하는 방법이다.
② 점증적으로 통합시켜 나가기 때문에 오류 발견이 쉽고 하드웨어 사용을 분산시킨다.
③ 테스트 드라이버(Driver)가 필요하다.
④ 테스트 초기에 시스템의 뼈대가 갖추어지지 않는다.

[이전 기출 문제]
23. 통합 테스팅 (Integration Testing)에 대한 설명으로 옳은 것은?

① 통합 테스팅 동안 발생하는 주요 어려움은 오류들을 지역화(localization)하는 것이다.
② 상향식 통합은 시스템의 계층구조와 상위층의 중요한 인터페이스를 조기에 테스팅할 수 있다.
③ 하향식 통합은 최하위 모듈을 먼저 통합하여 테스팅하는 방식이다.
④ 단위 모듈 테스팅을 철저하게 하면 통합 테스팅을 수행할 필요가 없다.

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

[이전 기출 문제]

24. 통합시험에서 모듈을 결합하여 시험하는 방식에 대한 설명으로 틀린 것은?

① 동시식(Big-Bang) 방법은 모든 모듈이 구현되고 시험된 후에 통합시험을 수행한다.
② 상향식 방법은 하위 모듈부터 통합하여 시험하기 때문에 스터브(Stub)의 사용이 필요하다.
③ 연쇄식(Threads) 방법은 특수하고 중요한 기능을 수행하는 모듈 집합을 먼저 구현하고 시험한다.
④ 하향식 방법은 상위층 모듈을 먼저 시험하므로 시스템의 계층 구조와 상위층의 중요한 인터페이스를 조기에 시험할 수 있다.

[기출 예상 문제]

25. 애플리케이션 통합 테스트 수행 절차로 옳은 것은?

ㄱ. 통합 테스트 케이스 설계
ㄴ. 통합 테스트 데이터 준비
ㄷ. 통합 테스트 수행 및 결과 확인
ㄹ. 통합 테스트 보고 및 종료
ㅁ. 결함 등록

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

[기출 예상 문제]

26. 다음 중 통합 테스트 유형에 대한 설명으로 가장 거리가 먼 것은?

① 사용 기반(Use Based) 테스트는 독립 클래스를 테스트 한 후 독립 클래스의 종속 클래스를 테스트한다.
② 스레드 기반(Thread Base) 테스트는 세부 기능의 모듈부터 주요 기능으로 범위를 넓혀 나가는 통합 테스트 기법이다.
③ 빅뱅(Bigbang) 테스트는 모든 모듈을 한꺼번에 테스트하는 기법이다.
④ 점증적(Incremental) 테스트는 주요 기능을 먼저 테스트 한 후 점차 범위를 넓혀가는 테스트이다.

[기출 예상 문제]

27. 애플리케이션 통합 테스트 수행 절차에 대한 설명으로 가장 거리가 먼 것은?

① 테스트 케이스에 따라 데이터를 입력하고 예상 결과와 비교한다.
② 테스트 결과가 예상 결과와 다를 경우 결함 등록을 하고, 결함 화면을 캡처하여 붙인다.
③ 테스트 데이터에서 개인 정보는 변경하지 않고 그대로 사용한다.
④ 통합 테스트 데이터는 운영 데이터베이스에서 일부를 추출하여 테스트 데이터를 만든다.

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

[기출 예상 문제]
1. 소스 코드 품질 분석 도구의 분석 내용으로 가장 거리가 먼 것은?

① 코딩 스타일 ② 코드의 복잡도
③ 스레드 결함 ④ 디자인 패턴 종류

[기출 예상 문제]
2. 다음 중 동적 분석 도구로 가장 적절한 것은?

① Valgind ② Checkstyle
③ PMD ④ McCabelQ

[기출 예상 문제]
3. 작성된 소스코드를 실행시키지 않고, 코드 자체만으로 코딩 표준준수여부, 코딩 스타일 적정 여부, 잔존 결함 발견 여부를 확인하는 코드 분석 도구는?

① 설계 분석 도구 ② 구조 분석 도구
③ 정적 분석 도구 ④ 동적 분석 도구

[기출 예상 문제]
4. 소스 코드 품질 분석 도구의 종류로 가장 적절한 것은?

① 동적 분석 도구 ② 명세 기반 도구
③ 품질 분석 도구 ④ 코드 분석 도구

[이전 기출 문제]
5. 다음을 위해 적합한 작업은 무엇인가?

> 소프트웨어의 설계를 개선하고. 소프트웨어에 대한 이해를 증진하며, 개발속도를 높이기 위해 필드를 한 클래스에서 다른 클래스로 옮기거나 메서드의 특정 코드를 추출하여 다른 메서드로 만드는 등 코드를 더 구조화 시킨다. 단, 외부 동작은 바뀌지 않으면서 내부구조만 개선되어야 한다.

① Refixing ② Refactoring
③ Remaking ④ Reengineering

[이전 기출 문제]
6. 리팩토링(Refactoring)에 대한 설명으로 옳은 것으로만 묶은 것은?

> ㄱ. 소프트웨어 디자인을 개선시킨다.
> ㄴ. 기능을 지속적으로 추가하는 작업이다.
> ㄷ. 소프트웨어를 이해하기 쉽게 만들고, 프로그램을 빨리 작성하게 도와준다.
> ㄹ. 겉으로 보이는 동작의 변화없이 내부구조를 변경하는 것이다.
> ㅁ. 버그를 수정하는 작업이다.

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

[소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

[기출 예상 문제]
7. 클린 코드(Clean Code)와 나쁜 코드(Bad Code)에 대한 설명으로 가장 거리가 먼 것은?

① 소스 코드 최적화는 읽기 쉽고 변경 및 추가가 쉬운 클린 코드(Clean Code)를 작성하는 것이다.
② 나쁜 코드(Bad Code)가 많아질 경우 잦은 오류가 발생할 가능성이 높다.
③ 클린 코드(Clean Code)의 특징에는 변수나 메소드에 대한 정의를 알 수 없게 이름을 짓는다.
④ 클린 코드(Clean Code)가 많을수록 애플리케이션 설계가 개선 된다.

[기출 예상 문제]
8. 클린 코드 작성 원칙이 아닌 것은?

① 가독성 ② 단순성 ③ 의존성 ④ 명세성

[기출 예상 문제]
9. 클린 코드 작성 원칙에 대한 설명으로 가장 거리가 먼 것은?

① 가독성: 코드 작성 시 들여쓰기 기능을 사용한다.
② 단순성: 클래스/메소드/함수를 최소 단위로 분리한다.
③ 의존성: 코드의 변경이 다른 부분에 영향이 없게 작성한다.
④ 추상화: 한 번에 한 가지 처리만 수행한다.

[기출 예상 문제]
10. 다음 중 소스코드 최적화 기법이 아닌 것은?

① 클래스 간 강한 결합(Tightly Coupling) 기법
② 클래스 분할 배치 기법
③ 적절한 주석문 사용
④ 네이밍 룰 정의

[기출 예상 문제]
11. 객체의 생성과 사용을 분리함으로써 소프트웨어 의존성을 최소화하기 위해 이용할 수 있는 디자인 패턴으로 가장 적절한 것은?

① 옵저버 패턴 ② 팩토리 메소드 패턴
③ 싱글톤 패턴 ④ 어댑터 패턴

[기출 예상 문제]
12. SpringMVC 구조에 대한 설명으로 가장 적절한 것은?

① 뷰(View)는 결과를 표시할 어떤 뷰를 선택할지 결정한다.
② 모델 앤 뷰(Model And View)는 결과 데이터인 모델 객체를 표현한다.
③ 핸들러 매핑(Handler Mapping)은 웹 요청시 해당 URL에 매핑되는 컨트롤러를 검색, 결정하는 컴포넌트이다.
④ 컨트롤러(Controller)는 수행 결과와 반영하는 모델 데이터 객체, 페이지 정보 및 뷰로 이루어져 있다.

[정답] [소프트웨어 개발>애플리케이션 테스트 관리]

1. 애플리케이션 테스트케이스 설계

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	2	4	3	1	2	2	2	4	2	4	1	2	3	3	1	2	4	3
21	**22**	**23**	**24**	**25**															
3	2	4	4	4															

2. 애플리케이션 통합 테스트

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	4	3	4	2	1	1	3	1	4	4	3	3	2	1	3	3	3	2	3
21	**22**	**23**	**24**	**25**	**26**	**27**													
2	2	1	2	1	2	3													

3. 애플리케이션 성능 개선

1	2	3	4	5	6	7	8	9	10	11	12								
4	1	3	1	2	2	3	4	4	1	2	3								

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트케이스 설계]

애플리케이션 테스트케이스 설계 1. ①

소프트웨어 테스트 프로세스
- 계획 수립 단계: 테스트 계획 수립 프로세스, 테스트 시나리오 및 케이스 관리 프로세스
- 테스트 수행 단계: 테스트 환경 관리 프로세스, 테스트 수행 관리 프로세스, 코드 커버리지 관리 프로세스, 형상 및 배포 관리 프로세스
- 테스트 결과 단계: 테스트 결과 검토 프로세스

애플리케이션 테스트케이스 설계 2. ①

① 애플리케이션 테스트 환경 관리 프로세스는 테스트 수행 관리 이전에 수행한다.

애플리케이션 테스트케이스 설계 3. ②

② 완벽한 테스팅은 불가능하다. 무한 경로, 무한 입력 값, 무한 시간이 소요되어 완벽하게 테스트 할 수 없으므로 리스크 분석과 우선 순위를 토대로 테스트에 집중할 것을 의미한다.

애플리케이션 테스트케이스 설계 4. ④

살충제 패러독스(Pesticide Paradox): 동일한 테스트 케이스로 반복 실행하면 결함을 발견할 수 없으므로 주기적으로 테스트 케이스를 리뷰하고 개선해야 한다.

애플리케이션 테스트케이스 설계 5. ③

오류-부재의 궤변(Absence of Errors Fallacy)에 대한 설명이다.

애플리케이션 테스트케이스 설계 6. ①

① 테스트는 구현과는 관계없는 독립된 팀에 의하여 수행되어야 한다.

애플리케이션 테스트케이스 설계 7. ②

테스트 케이스에 대한 설명이다.

애플리케이션 테스트케이스 설계 8. ②

테스트 오라클은 테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된 참 값을 입력하여 비교하는 기법 및 활동을 말한다.

애플리케이션 테스트케이스 설계 9. ②

테스트 케이스의 작성 절차 중 테스트 구조 설계 및 테스트 방법 결정 단계에 대한 설명이다.

애플리케이션 테스트케이스 설계 10. ④

테스트 케이스 작성 절차
테스트 계획 검토 및 자료 확보→위험 평가 및 우선순위 결정→테스트 요구사항 정의→테스트 구조 설계 및 테스트 방법 결정→테스트 케이스 정의→테스트 케이스 타당성 확인 및 유지보수

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트케이스 설계]

애플리케이션 테스트케이스 설계 11. ②

테스트 오라클 유형
- 참(True) 오라클: 모든 입력 값에 대하여 기대하는 결과를 생성함으로써 발생된 오류를 모두 검출할 수 있는 오라클이다.
- 샘플링(Sampling) 오라클: 특정한 몇 개의 입력 값에 대해서만 기대하는 결과를 제공해 주는 오라클이다.
- 휴리스틱(Heuristic) 오라클: 샘플링 오라클을 개선한 오라클로, 특정 입력 값에 대해 올바른 결과를 제공하고, 나머지 값들에 대해서는 휴리스틱(추정)으로 처리하는 오라클이다.
- 일관성 검사(Consistent) 오라클: 애플리케이션 변경이있을 때, 수행 전과 후의 결과 값이 동일한지 확인하는 오라클이다.

애플리케이션 테스트케이스 설계 12. ④

항공기, 임베디드, 발전소 소프트웨어 등 미션 크리티컬한 업무에는 참(True) 오라클을 적용해야 한다.

애플리케이션 테스트케이스 설계 13. ①

- 참 오라클: 항공기, 임베디드, 발전소 소프트웨어 등
- 샘플링/추정 오라클: 일반, 업무용, 게임, 오락 등

애플리케이션 테스트케이스 설계 14. ②

- 단위 테스트: 구현된 단위 모듈(함수, 서브루틴, 컴포넌트 등)의 기능 수행 여부를 판정하고 내부에 존재하는 논리적 오류를 검출한다.
- 통합 테스트: 모듈 간의 인터페이스 연계를 검증하고 오류를 확인, 모듈 간의 상호 작용 및 연계 동작이 제대로 기능하는지 점검한다.
- 시스템 테스트: 단위, 통합 테스트 후 전체 시스템이 정상적으로 작동하는지 판정하는 기능을 점검한다.
- 인수 테스트: 사용자 요구분석 명세서에 명시된 사항을 모두 충족하는지 판정하고 시스템이 예상대로 동작하고 있는지 점검한다.

애플리케이션 테스트케이스 설계 15. ③

③ 시스템 테스트는 기능명세 분석 단계에서 진행한다. 요구사항 분석 단계에서는 인수 테스트를 진행한다.

애플리케이션 테스트케이스 설계 16. ③

개발 과정에서 실시되는 시험의 순서는 단위 테스트(모듈 내부적인 오류를 발견하기 위한 시험)→통합 테스트(모듈간의 인터페이스 오류를 발견하기 위한 시험)→시스템 테스트(사용자의 요구사항을 만족하는지 판단하는 시험) 순으로 수행된다.

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트케이스 설계]

애플리케이션 테스트케이스 설계 17. ①

① 테스트 항목을 하나의 시나리오에 모두 작성하지 않고, 시스템별, 모듈별, 항목별 테스트 시나리오를 분리하여 작성한다.

애플리케이션 테스트케이스 설계 18. ②

테스트 시나리오 관리 프로세스
테스트 시나리오 도출→테스트 케이스 도출→테스트 케이스 검토 및 보안→테스트 케이스 피어리뷰→테스트 케이스 업로드

애플리케이션 테스트케이스 설계 19. ④

테스트 케이스 검토 및 보완 단계에서 테스트 케이스가 만들어지면 품질 관리자와 테스트 리더가 검토를 한다.

애플리케이션 테스트케이스 설계 20. ③

테스트 지식 체계(ISO 29119)
- 파트1: 개념과 정의 - 파트2: 테스트 프로세스
- 파트3: 테스트 문서화 - 파트4: 테스트 기술
- 파트5: 키워드 기반 테스팅

애플리케이션 테스트케이스 설계 21. ③

③ 파트3. 테스트 문서화와 관련된 내용이다.
테스트 지식 체계 파트1. 개념과 정의
- 전체 시리즈에 대한 가이드를 제공
- 소프트웨어 테스팅 개념
- 조직과 프로젝트 관점에서의 소프트웨어 테스팅
- 소프트웨어 생명 주기 모델에서의 일반적인 테스팅 프로세스
- 위험 기반 테스팅(Risk-based Testing)
- 테스트 서브 프로세스(Test Sub-process)

애플리케이션 테스트케이스 설계 22. ②

파트5. 키워드 주도 테스팅에 대한 설명이다.

애플리케이션 테스트케이스 설계 23. ④

테스트 기법
- 분석 기법: 동적 분석, 정적 분석
- 실행 기법: 화이트박스 실행, 블랙 박스 실행
- 설계 기법: 구조기반 설계, 명세기반 설계, 경험기반 설계
- 자동화 기법: 설계 자동화, 실행/모니터링 자동화, 관리 자동화

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 테스트케이스 설계]

애플리케이션 테스트케이스 설계 24. ④

④ 파트1. 개념과 정의에 관련된 내용이다.
[테스트 지식 체계(ISO 29119)] 파트2. 테스트 프로세스
- 조직, 테스트 관리, 동적 테스트의 3가지 수준의 다계층 프로세스 모델을 설명
- 다계층 프로세스 모델(Multi - Layer Process Model)
- 조직 의 테스트 프로세스(Organizational Test Process)
- 테스트 관리 프로세스(Test Management Process)
- 동적 테스트 프로세스(Dynamic Test Process)

애플리케이션 테스트케이스 설계 25. ④

테스트 지식 체계(ISO 29119)
- 파트1: 개념과 정의 - 파트2: 테스트 프로세스
- 파트3: 테스트 문서화 - 파트4: 테스트 기술
- 파트5: 키워드 기반 테스팅

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

애플리케이션 통합 테스트 1. ②

테스트 결함 관리에 대한 설명이다.

애플리케이션 통합 테스트 2. ④

④ Atlassian(아틀라시안)은 소프트웨어를 만드는 소프트웨어 기업이다.

애플리케이션 통합 테스트 3. ③

③ 프로세스 및 워크플로우 변경이 가능해야 한다.

애플리케이션 통합 테스트 4. ④

결함 관리 도구 중 오픈 소스 도구인 Trac에 대한 설명이다.

애플리케이션 통합 테스트 5. ②

- 설계 단계: 명세 기반 테스트 설계 도구, 코드 기반 테스트 설계 도구, 테스트 관리 도구
- 구현/테스트 단계: 정적 분석 도구, 리뷰 및 인스펙션 도구, 커버리지 측정 도구, 동적 분석 도구, 성능/부하/시뮬레이션 도구, 기능 테스트 수행 도구

애플리케이션 통합 테스트 6. ①

정적 분석 도구는 프로그램을 수행하지 않고 분석하는 도구이다. (복잡도 측정 등)

애플리케이션 통합 테스트 7. ①

② 요구사항 관리 도구: 요구사항 작성 및 추적성 모니터링 지원
③ 순서 검사 도구: 이벤트 순서, 오류 순서 지적
④ 성능 테스팅 도구: 가상의 부하(load)발생을 통한 성능/부하/스트레스 테스트 실행

애플리케이션 통합 테스트 8. ③

③ 순서 검사 도구는 정적 테스팅 도구이다.
- 동적 테스팅 도구: 단위 테스트 도구, 테스트 실행 도구, 성능 테스팅 도구, 커버리지 측정 도구, 동적 분석 도구, 모니터링 도구

애플리케이션 통합 테스트 9. ①

소프트웨어 성능 및 모니터링 도구
- 소프트웨어가 실행 도중에만 발생하는 시간 의존성과 메모리 누수(Memory Leaks)와 같은 결함 발견에 활용한다.
- 소프트웨어의 성능/부하/스트레스를 테스트한다.
- 특정 시스템 리소스의 사용량을 지속적으로 분석하고 확인한다.

애플리케이션 통합 테스트 10. ④

④ 소프트웨어 정적 분석 지원 도구에 대한 설명이다.

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

애플리케이션 통합 테스트 11. ④

④ 소프트웨어 성능 및 모니터링 도구에 대한 설명이다.

애플리케이션 통합 테스트 12. ③

①, ②, ④ 테스트 관리 지원 도구
③ 소프트웨어 정적 분석 지원 도구

애플리케이션 통합 테스트 13. ③

애플리케이션 통합 테스트 수행 방법
- 점증적인 방식과 빅뱅 방식으로 나누어 진다.
- 점증적인 방식은 개발된 컴포넌트 중 일부를 테스트하고 점차적으로 컴포넌트를 늘려가면서 테스트하는 방식이다. 점증적인 방법은 다시 상향식 통합과 하향식 통합으로 나누어 진다.
- 빅뱅 방식은 모든 컴포넌트를 사전에 통합하여 한꺼번에 테스트하는 방식이다.

애플리케이션 통합 테스트 14. ②

소프트웨어 정적 분석 지원 도구에 대한 설명이다.

애플리케이션 통합 테스트 15. ①

① 회귀 테스트(Regression Testing)는 이미 통합 테스트를 완료한 컴포넌트가 어떠한 변화로 인해 의도하지 않은 오류가 생기지 않았음을 보증하기 위해 반복 테스트하는 것을 말한다.

애플리케이션 통합 테스트 16. ③

① 하향식 통합에 대한 설명이다.
② 상향식 통합에 대한 설명이다.
④ 동시식 통합은 모든 모듈이 구현되고 테스트되기 전에는 통합 테스트를 실행할 수 없으므로 일정 계획의 융통성을 획득하기 어렵다.

애플리케이션 통합 테스트 17. ③

하향식 테스트(top-down test)는 상위 모듈부터 아래 방향으로 하향식으로 통합하면서 테스트를 진행한다. 주요 기능을 초기에 테스트 가능하고, 아직 작성되지 않은 컴포넌트인 스텁(Stub)을 개발하여 테스트를 수행한다.

애플리케이션 통합 테스트 18. ③

③ 상향식 통합 방식에 대한 설명이다.

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

애플리케이션 통합 테스트 19. ②

① 테스트 하네스(test harness): 시스템의 기능 일부만 시험하기 위하여 소프트웨어를 변경하는 것
② 스텁(stub): 시험 대상 모듈이 호출하는 또 다른 기능을 대신하여 쓰인 간이 소프트웨어
④ 테스트 드라이버(test driver): 시험 대상 모듈을 호출하는 간이 소프트웨어

애플리케이션 통합 테스트 20. ③

하향식 통합 시험은 깊이 우선 통합, 너비 우선 통합에 따라 하위 스텁들을 하나씩 실제 모듈로 대체한다.
- 깊이 우선 통합: A - AB - ABDEF - AC - ACG 또는 A - AC - ACG - AB - ABDEF
- 너비 우선 통합: A - ABC - ABCDEFG

애플리케이션 통합 테스트 21. ②

상향식 통합테스트 과정
낮은 수준의 모듈을 클러스터로 결합→드라이버(driver) 작성→클러스터 검사→드라이버 제거 및 클러스터 상향 조합

애플리케이션 통합 테스트 22. ②

② 하향식 통합 테스트에 대한 설명이다.
* 상향식 통합 테스트
- 하위 레벨 모듈부터 점진적 모듈 통합
- test driver 필요
- 개발 초기 단계에서 병행작업 가능
- 마지막 통합 단계에 이를 때 까지 주 프로그램이 아님
- 컴퓨터 검증은 시스템 개발 최종 단계에서만 가능
* 하향식 통합 테스트
- 상위 레벨 모듈부터 점진적 모듈 통합
- test stub 필요
- 명령어 처리 모듈을 먼저 구현하고 시험
- 주요 기능을 조기에 시험 가능
- 하드웨어 사용이 분산되고 오류 발견이 용이

애플리케이션 통합 테스트 23. ①

② 하향식 통합 테스트에 대한 설명이다.
③ 상향식 통합 테스트에 대한 설명이다.
④ 단위 모듈 테스팅은 시스템 개개의 모듈들을 시험하고, 통합 테스트는 몇 개의 모듈을 결합하여 시험하므로 서로 독립적으로 테스팅이 수행되어야 한다.

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 통합 테스트]

애플리케이션 통합 테스트 24. ②

② 상향식 방법은 하위 모듈부터 통합하여 시험하기 때문에 테스트 드라이버(test driver)의 사용이 필요하다.

애플리케이션 통합 테스트 25. ①

애플리케이션 통합 테스트 수행 절차
통합 테스트 케이스 설계→통합 테스트 데이터 준비→통합 테스트 수행 및 결과 확인→결함 등록→테스트 결과 보고 및 종료

애플리케이션 통합 테스트 26. ②

② 스레드 기반(Tread Base) 테스트는 특수하고 중요한 기능을 수행하는 모듈 집합을 먼저 구현하고 시험한다.

애플리케이션 통합 테스트 27. ③

③ 보안을 위하여 테스트 데이터에서 개인정보는 스크램블링(Scrambling) 한다.

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

애플리케이션 성능 개선 1. ④

소스 코드에 대한 코딩 스타일, 설정된 코딩 표준, 코드의 복잡도, 코드 내에 존재하는 메모리 누수 현황, 스레드의 결함 등을 발견하기 위하여 소스 코드 품질 분석 도구를 사용한다.

애플리케이션 성능 개선 2. ①

②, ③ 정적 분석 도구
④ 코드 복잡도 도구

애플리케이션 성능 개선 3. ③

- 정적 분석 도구: 작성된 소스 코드를 실행시키지 않고, 코드 자체로만으로 코딩 표준 준수 여부, 코딩 스타일 적정 여부, 잔존 결함 발견 여부를 확인하는 코드 분석 도구
- 동적 분석 도구: 애플리케이션을 실행하여 코드에 존재하는 메모리 누수 현황을 발견하고, 발생한 스레드의 결함 등을 분석하기 위한 도구

애플리케이션 성능 개선 4. ①

소스 코드 품질 분석 도구는 정적 분석 도구와 동적 분석 도구가 있다.

애플리케이션 성능 개선 5. ②

리팩토링(Refactoring)이란 코드의 외부 행위는 바꾸지 않고 내부 구조를 개선시켜 소프트웨어 시스템을 변경하는 프로세스이다. 이는 버그 발생의 기회를 최소화 시켜주는 코드 정화(Clean up)방법이다.

애플리케이션 성능 개선 6. ②

ㄴ. 기능(동작)의 변경없이 내부구조를 변경하는 것이다.
ㅁ. 버그를 수정하는 작업은 디버깅(Debugging)이다.
리팩토링(Refactoring)은 버그를 찾는데 도움을 준다.

애플리케이션 성능 개선 7. ③

③ 나쁜 코드(Bad Code)의 대표적인 사례로 변수나 메소드에 대한 이름 정의를 알 수 없는 코드가 있다.

애플리케이션 성능 개선 8. ④

클린 코드 작성 원칙: 가독성, 단순성, 의존성, 중복성, 추상화

[정답 및 해설] [소프트웨어 개발>애플리케이션 테스트 관리>애플리케이션 성능 개선]

애플리케이션 성능 개선 9. ④

④ 단순성에 대한 설명이다.
- 추상화: 클래스/메소드/함수에 대해 동일한 수준의 추상화를 한다. 상세 내용은 하위 클래스/메소드/함수에서 구현한다.

애플리케이션 성능 개선 10. ①

소스코드 최적화 기법
- 클래스 분할 배치 기법
- 클래스 간 느슨한 결합(Loosely Coupled) 기법
- 코딩 형식 기법
- 적절한 주석문 사용

애플리케이션 성능 개선 11. ②

Factory Method(팩토리 메서드) 패턴: 객체 생성을 위한 인터페이스를 정의한 후 상속한 서브 클래스를 이용하여 객체를 생성한다. 객체의 생성과 사용을 분리함으로써 소프트웨어 의존성을 최소화 시킬수 있다.

애플리케이션 성능 개선 12. ③

Spirng MVC 구조
- 디스패처 서블릿(Dispatcher Servlet)은 Spring MVC 프레임워크의 Front Controller, 웹 요청과 응답의 수명주기를 주관하는 컴포넌트이다.
- 핸들러 매핑(Handler Mapping)은 웹 요청시 해당 URL에 매핑되는 컨트롤러를 검색, 결정하는 컴포넌트이다.
- 컨트롤러(Controller)는 비즈니스 로직을 수행하고 결과를 Model And View에 반영하는 컴포넌트이다.
- 모델 앤 뷰(Model And View)는 수행 결과와 반영하는 모델 데이터 객체, 이동할 페이지 정보 및 뷰로 이루어져 있다.
- 뷰 리졸버(View Resolver)는 결과를 표시할 어떤 뷰를 선택할지 결정한다.
- 뷰(View)는 결과 데이터인 모델 객체를 표현한다.