[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 테스트 케이스 설계]

애플리케이션 테스트 케이스 설계 1. 상용

* 응용 소프트웨어
- 상용 소프트웨어 - 서비스 제공 소프트웨어

애플리케이션 테스트 케이스 설계 2. ① (ㄷ) ② (ㄴ) ③ (ㄱ)

* 상용 소프트웨어
- 산업 범용 소프트웨어: 시스템 소프트웨어, 미들웨어, 응용 소프트웨어
- 산업 특화 소프트웨어

애플리케이션 테스트 케이스 설계 3. ① (ㄴ) ② (ㄱ) ③ (ㄷ) ④ (ㄹ)

* 서비스 제공 소프트웨어
- 신규 개발 소프트웨어 - 기능 개선 소프트웨어
- 추가 개발 소프트웨어 - 시스템 통합 소프트웨어

애플리케이션 테스트 케이스 설계 4. ① 확인(Validation) ② 검증(Verification)

* 테스트에 대한 시각
1. 확인(Validation) 테스트 – 사용자 입장
: 고객의 요구사항에 맞게 구현되었는지 확인
2. 검증(Verification) 테스트 – 개발자(시험자) 입장
: 설계 명세서에 맞게 만들어졌는지 점검

애플리케이션 테스트 케이스 설계 5. 오류 발견 관점, 오류 예방 관점, 품질 향상 관점

* 소프트웨어 테스트의 필요성
- 오류 발견 관점 - 오류 예방 관점
- 품질 향상 관점

애플리케이션 테스트 케이스 설계 6. ① 오류 예방 ② 품질 향상 ③ 오류 발견

* 소프트웨어 테스트의 필요성
- 오류 발견 관점 - 오류 예방 관점
- 품질 향상 관점

애플리케이션 테스트 케이스 설계 7. (ㄱ), (ㄹ), (ㅁ)

* 소프트웨어 테스트의 기본 원리
- 테스팅은 결함이 존재함을 밝히는 활동이다.
- 완벽한 테스팅은 불가능하다.
- 테스팅은 개발 초기에 시작해야 한다.
- 결함 집중(Defect Clustering): 애플리케이션 결함의 대부분은 소수의 특정한 모듈에 집중되어 존재한다.
- 살충제 패러독스(Presticide Paradox)
- 테스팅은 정황(Context)에 의존한다.
- 오류-부재의 궤변(Absence of Errors Fallacy)
- 기타: 테스트는 별도 팀에서 수행한다. 작은 부분에서 시작해서 점차 확대하며 진행한다.

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 테스트 케이스 설계]

애플리케이션 테스트 케이스 설계 8. 살충제 패러독스 (Presticide Paradox)

- 살충제 패러독스(Presticide Paradox): 동일한 테스트 케이스로 반복 실행하면 결함을 발견할 수 없으므로 주기적으로 테스트 케이스를 리뷰하고 개선

애플리케이션 테스트 케이스 설계 9. 오류-부재의 궤변 (Absence of Errors Fallacy)

- 오류-부재의 궤변(Absence of Errors Fallacy)
: 소프트웨어 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 할 수 없다.

애플리케이션 테스트 케이스 설계 10. 파레토의 법칙

- 파레토의 법칙: 전체 결함의 80%는 소프트웨어 제품의 전체 기능 중 20%에 집중되어 있다.

애플리케이션 테스트 케이스 설계 11. 테스트 계획서, 테스트 케이스, 테스트 시나리오, 테스트 결과서

* 소프트웨어 테스트 산출물
- 테스트 계획서 - 테스트 케이스
- 테스트 시나리오 - 테스트 결과서

애플리케이션 테스트 케이스 설계 12. ① (ㄴ) ② (ㄷ) ③ (ㄱ) ④ (ㄷ)

* 소프트웨어 테스트 산출물
- 테스트 계획서 - 테스트 케이스
- 테스트 시나리오 - 테스트 결과서

애플리케이션 테스트 케이스 설계 13. ① 정적 ② 동적 ③ 인스펙션, 코드검사, 워크스루 ④ 화이트박스 테스트, 블랙박스 테스트

* 프로그램 실행 여부에 따른 소프트웨어 테스트의 유형
- 정적 테스트 - 동적 테스트

애플리케이션 테스트 케이스 설계 14. 워크스루 (Walkthrough)

- 워크스루 (Walkthrough): 비공식적 검토과정으로 개발에 참여한 팀으로 구성 가능.

애플리케이션 테스트 케이스 설계 15. (ㄷ), (ㄹ), (ㅁ)

* 화이트박스 테스트: 프로그램의 내부 로직을 보면서 테스트를 수행하는 구조적 테스트
(ㄱ), (ㄴ) 블랙박스 테스트에 대한 설명이다.

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 테스트 케이스 설계]

애플리케이션 테스트 케이스 설계 16. 기초 경로 검사

* 화이트박스 테스트 종류
- 기초 경로 검사(Basic Path Testing)
- 제어 구조 검사

애플리케이션 테스트 케이스 설계 17. ① 루프 검사 ② 조건 검사 ③ 데이터 흐름 검사

* 제어 구조 검사
- 조건 검사(Condition Testing)
- 루프 검사(Loop Testing)
- 데이터 흐름 검사(Data Flow Testing)

애플리케이션 테스트 케이스 설계 18. 동치 분할 검사, 경계값 분석, 원인-효과 그래프 검사, 오류 예측 검사, 비교 검사

* 블랙박스 테스트 종류
- 동치(동등) 분할 검사(Equivalence Partitioning)
- 경계값 분석(Boundary Value Analysis)
- 원인-효과 그래프 검사(Cause-Effect Graphing Testing)
- 오류 예측 검사(Fault Based Testing)
- 비교 검사(Comparison Testing)

애플리케이션 테스트 케이스 설계 19. (ㄱ), (ㄷ), (ㄹ)

* 블랙박스 테스트: 사용자 요구사항 명세서를 보면서 테스트를 수행하는 기능 테스트
(ㄴ) 화이트박스 테스트에 대한 설명이다.

애플리케이션 테스트 케이스 설계 20. 구문 커버리지, 결정 커버리지, 조건 커버리지, 조건/결정 커버리지, 변형 조건/결정 커버리지

* 코드 커버리지: 화이트박스 테스트의 검증 기준
- 구문(Statement) 커버리지
- 결정(Decision) 커버리지
- 조건(Condition) 커버리지
- 조건/결정 커버리지
- 변경 조건/결정 커버리지

애플리케이션 테스트 케이스 설계 21. ① 경계값 분석 ② 동치 분할 검사 ③ 비교 검사 ④ 원인-효과 그래프 검사 ⑤ 오류 예측 검사

* 블랙박스 테스트 종류
- 동치(동등) 분할 검사 - 경계값 분석
- 원인-효과 그래프 검사 - 오류 예측 검사
- 비교 검사

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 테스트 케이스 설계]

애플리케이션 테스트 케이스 설계 22. ① 기능 기반 커버리지 ② 라인 커버리지 ③ 코드 커버리지

* 테스트 커버리지: 주어진 테스트 케이스에 의해 수행되는 소프트웨어의 테스트 범위를 측정하는 테스트 품질 측정 기준
- 기능 기반 커버리지 - 라인 커버리지
- 코드(소스코드) 커버리지

애플리케이션 테스트 케이스 설계 23. ① (ㅅ) ② (ㄱ) ③ (ㄴ) ④ (ㅂ) ⑤ (ㅁ) ⑥ (ㄹ) ⑦ (ㄷ)

* 테스트 목적에 따른 테스트 기법
- 회복(Recovery) 테스트
- 안전(Security) 테스트
- 강도(Stress) 테스트
- 성능(Performance) 테스트
- 구조(Structure) 테스트
- 회귀(Regression) 테스트
- 병행(Parallel) 테스트

애플리케이션 테스트 케이스 설계 24. ① 구조 기반 테스트 ② 경험 기반 테스트 ③ 명세 기반 테스트

* 테스트 종류에 따른 테스트 기법
- 명세 기반 테스트 - 구조 기반 테스트
- 경험 기반 테스트

애플리케이션 테스트 케이스 설계 25. ① (ㄴ) ② (ㅁ) ③ (ㄱ) ④ (ㅂ) ⑤ (ㄹ) ⑥ (ㄷ)
⑦ (ㄹ)→(ㅁ)→(ㄷ)→(ㄴ)→(ㅂ)→(ㄱ)

* 테스트 케이스 작성 순서
- 1단계: 테스트 계획 검토 및 자료 확보
- 2단계: 위험 평가 및 우선순위 결정
- 3단계: 테스트 요구사항 정의
- 4단계: 테스트 구조 설계 및 테스트 방법 결정
- 5단계: 테스트 케이스 정의
- 6단계: 테스트 케이스 타당성 확인 및 유지 보수

애플리케이션 테스트 케이스 설계 26. 테스트 오라클

- 테스트 오라클(Test Oracle): 테스트의 결과가 참인지 거짓인지를 판단하는 도구

애플리케이션 테스트 케이스 설계 27. ① 참 오라클 ② 샘플링 오라클 ③ 휴리스틱 오라클 ④ 일관성 검사 오라클

* 테스트 오라클 유형
- 참(True) 오라클 - 샘플링(Sampling) 오라클
- 휴리스틱(Heuristic) 오라클
- 일관성 검사(Consistent) 오라클

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 테스트 케이스 설계]

애플리케이션 테스트 케이스 설계 29. 알파 테스트

* 인수 테스트
- 알파 테스트 - 베타 테스트
- 사용자 인수 테스트 - 운영상의 인수 테스트
- 계약 인수 테스트 - 규정 인수 테스트

애플리케이션 테스트 케이스 설계 29. ① 단위 테스트 ② 통합 테스트 ③ 시스템 테스트 ④ 인수 테스트

* 테스트 단계
- 단위 테스트 - 통합 테스트
- 시스템 테스트 - 인수 테스트

애플리케이션 테스트 케이스 설계 30. ① 인수 테스트 ② 단위 테스트 ③ 시스템 테스트 ④ 통합 테스트

* 테스트 단계
- 단위 테스트 - 통합 테스트
- 시스템 테스트 - 인수 테스트

애플리케이션 테스트 케이스 설계 31. 테스트 시나리오

- 테스트 시나리오: 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트케이스들을 묶은 집합

애플리케이션 테스트 케이스 설계 32. ① 하드웨어 ② 소프트웨어

* 테스트 환경 구축의 유형
- 하드웨어 기반의 테스트 환경 구축
- 소프트웨어 기반의 테스트 환경 구축
- 가상 시스템 기반의 테스트 환경 구축

애플리케이션 테스트 케이스 설계 33. 테스트 데이터

- 테스트 데이터: 컴퓨터의 동작이나 시스템의 적합성을 시험하기 위해 특별히 개발된 데이터 집합

애플리케이션 테스트 케이스 설계 34. ① 시작조건 ② 종료조건

* 테스트 조건
- 테스트 시작 조건 - 테스트 종료 조건
- 테스트 성공과 실패의 판단 기준

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 통합 테스트 수행]

애플리케이션 통합 테스트 수행 1. 빅뱅 방식

* 통합 테스트
- 점증적인 통합 방식 - 비점증적인 통합 방식

애플리케이션 통합 테스트 수행 2. ① 스텁(Stub) ② 드라이버(Driver)

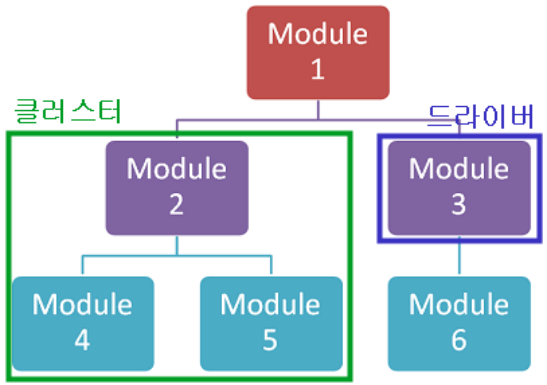
- 스텁(Stub): 모듈 간에 통합 시험을 하기 위해 일시적으로 제공되는 시험용 모듈
- 드라이버(Driver): 하위 모듈은 있으나 상위 모듈이 없는 경우 하위 모듈 구동하기 위한 제어 프로그램

애플리케이션 통합 테스트 수행 3. 혼합식 통합 테스트

- 혼합식 통합 테스트: 상위 레벨은 하향식 통합, 하위 레벨은 상향식 통합을 사용한다.

애플리케이션 통합 테스트 수행 4. 클러스터

- 클러스터:
하나의 주요 제어 모듈과 관련된 종속 모듈의 그룹



애플리케이션 통합 테스트 수행 5. 회귀 테스팅 (Regression Testing)

* 회귀 테스팅: 통합 테스트가 완료된 후에 변경된 모듈이나 컴포넌트가 있다면 새로운 오류 여부를 확인하기 위한 테스팅

애플리케이션 통합 테스트 수행 6. 테스트 자동화 도구

테스트 자동화 도구에 대한 설명이다.

애플리케이션 통합 테스트 수행 7. ① 빅뱅 ② 상향식 통합 ③ 하향식 통합

* 통합 테스트
- 비점증적 통합방식: 빅뱅
- 점증적 통합 방식: 상향식 통합, 하향식 통합, 혼합식 통합

애플리케이션 통합 테스트 수행 8. 휴먼 에러(Human Error)

휴먼 에러(Human)에 대한 설명이다.

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 통합 테스트 수행]

애플리케이션 통합 테스트 수행 9. (ㄱ), (ㄷ), (ㅁ)

(ㄴ) 상용 도구의 경우 고가, 유지 관리 비용이 높아 추가 투자가 필요하다.
(ㄹ) 사용자 인터페이스가 없는 서비스의 경우에도 정밀한 테스트가 가능하다.

애플리케이션 통합 테스트 수행 10. 테스트 하네스(Test Harness)

테스트 하네스(Test Harness)에 대한 설명이다.

애플리케이션 통합 테스트 수행 11. ① (ㄷ) ② (ㄹ) ③ (ㄱ) ④ (ㄴ) ⑤ (ㅁ)

* 테스트 자동화 도구 유형
- 정적 분석 도구(Static Analysis Tools)
- 테스트 실행 도구(Test Execution Tools)
- 성능 테스트 도구(Performance Test Tools)
- 테스트 통제 도구(Test Control Tools)
- 테스트 장치(Test Harness, 테스트 하네스 도구)

애플리케이션 통합 테스트 수행 12. ① 테스트 슈트 ② 테스트 스크립트 ③ 테스트 드라이버 ④ 테스트 스텁 ⑤ 테스트 케이스 ⑥ 목 오브젝트

* 테스트 하네스 구성요소
- 테스트 드라이버(Test Driver)
- 테스트 스텁(Test Stub)
- 테스트 케이스(Test Case)
- 테스트 슈트(Test Suites)
- 테스트 스크립트(Test Script)
- 목 오브젝트(Mock Object)

애플리케이션 통합 테스트 수행 13. ① (ㄱ) ② (ㄹ) ③ (ㄴ) ④ (ㄷ)

* 테스트 단계별 테스트 자동화 도구
- 테스트 계획: 요구사항 관리
- 테스트 분석/설계: 테스트케이스 생성
- 테스트 수행: 테스트 자동화, 정적 분석, 동적 분석, 성능 테스트, 모니터링
- 테스트 관리: 커버리지 측정, 형상관리, 결함 추적/관리

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 통합 테스트 수행]

애플리케이션 통합 테스트 수행 14. (ㄱ)→(ㅁ)→(ㄴ)→(ㄹ)→(ㄷ)

* 애플리케이션 통합 테스트 수행 순서
- 1단계: 통합 테스트 계획서 검토
- 2단계: 테스트 환경 준비
- 3단계: 통합 테스트 케이스 및 시나리오 검토
- 4단계: 통합 모듈 및 인터페이스가 요구사항을 충족하는지 테스트 수행
- 5단계: 통합 테스트 결과 기록

애플리케이션 통합 테스트 수행 15. (ㄹ)→(ㅁ)→(ㄷ)→(ㄴ)→(ㅂ)→(ㄱ)

* 애플리케이션 테스트 프로세스
- 1단계: 테스트 계획
- 2단계: 테스트 분석 및 디자인
- 3단계: 테스트 케이스 및 시나리오 작성
- 4단계: 테스트 수행
- 5단계: 테스트 결과 평가 및 리포팅
- 6단계: 결함 추적 및 관리

애플리케이션 통합 테스트 수행 16. ① (ㄴ) ② (ㄷ) ③ (ㄱ)

* 소프트웨어 결함
- 에러/오류
- 결함/결점/버그
- 실패/문제

애플리케이션 통합 테스트 수행 17. ① (ㅁ) ② (ㄴ) ③ (ㄷ) ④ (ㄱ) ⑤ (ㄹ)

* 테스트 리포팅
- 테스트 결과 정리
- 테스트 요약 문서
- 품질 상태
- 테스트 결과서
- 테스트 실행 절차 및 평가

애플리케이션 통합 테스트 수행 18. (ㄷ)→(ㄹ)→(ㅁ)→(ㄴ)→(ㅅ)→(ㄱ)→(ㅂ)

* 결함 관리 프로세스
- 1단계: 에러 발견
- 2단계: 에러 등록
- 3단계: 에러 분석
- 4단계: 결함 확정
- 5단계: 결함 할당
- 6단계: 결함 조치
- 7단계: 결함 조치 검토 및 승인

애플리케이션 통합 테스트 수행 19. ① 결함 추세 ② 결함 분포 ③ 결함 에이징

* 결함 추이 분석 유형
- 결함 분포 분석
- 결함 추세 분석
- 결함 에이징 분석

[정답 및 해설] [애플리케이션 테스트 케이스 관리>애플리케이션 통합 테스트 수행]

애플리케이션 통합 테스트 수행 20. ① 결함 할당 ② 에러 등록 ③ 에러 분석 ④ 에러 발견 ⑤ 결함 확정 ⑥ 결함 조치 ⑦ 결함 조치 검토 및 승인

* 결함 관리 프로세스
: 에러 발견→에러 등록→에러 분석→결함 확정→결함 할당→결함 조치→결함 조치 검토 및 승인

애플리케이션 통합 테스트 수행 21. ① 결함 관리 ② 결함 추이 분석

- 결함 관리: 결함을 추적하고 관리하는 활동
- 결함 추이 분석: 결함 분석 후 향후 어떤 결함이 발생할지를 추정하는 작업

애플리케이션 통합 테스트 수행 22. ① (ㄴ) ② (ㄹ) ③ (ㄱ) ④ (ㄷ)

* 애플리케이션 테스트 결과 분석
- 1단계: 통합 테스트 결과서 검토
- 2단계: 테스트 결과서의 결함 내용을 확인하고, 결함 관리 활동 수행
- 3단계: 결함 관리 대장의 내용을 확인하고, 결함을 분석 및 평가
- 4단계: 결함에 대한 추이 분석을 하고, 잔존 결함에 대해 추정

애플리케이션 통합 테스트 수행 23. ① 심각도 ② 우선순위

- 결함 심각도: 발생한 결함이 어떤 영향을 끼치며, 그 결함이 얼마나 치명적인지를 나타내는 척도
- 결함 우선순위: 발생한 결함이 얼마나 빠르게 처리되어야 하는지를 결정하는 척도

애플리케이션 통합 테스트 수행 24. (ㄴ)→(ㅁ)→(ㄹ)→(ㄷ)→(ㄱ)

* 애플리케이션 개선 조치사항 작성 순서
- 1단계: 통합 테스트 결과 상세 분석
- 2단계: 테스트 커버리지 평가
- 3단계: 테스트 케이스 통계 및 분석을 통해 테스트 충분성 여부를 파악
- 4단계: 발견된 결함에 대한 개선 조치사항 작성
- 5단계: 결함 조치 이력 관리

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

애플리케이션 성능 개선 1. ① (ㄷ) ② (ㄹ) ③ (ㄴ) ④ (ㄱ)

* 애플리케이션의 성능을 측정하기 위한 지표
- 처리량(Throughput)
- 응답 시간(Response Time)
- 경과 시간(Turnaround Time, 소요 시간, 반환 시간)
- 자원 사용률(Resource Usage)

애플리케이션 성능 개선 2. ① 성능 점검 도구 ② 모니터링 도구

* 유형별 성능 분석 도구
- 성능 점검 도구 - 모니터링 도구

애플리케이션 성능 개선 3. ① (ㄱ), (ㄴ), (ㄹ) ② (ㄷ), (ㅁ), (ㅂ)

- 데이터베이스 연결 및 쿼리 실행 시 발생되는 성능 저하 원인: 데이터베이스 락, 불필요한 데이터베이스 페치, 연결 누수, 부적절한 커넥션 풀 크기 등
- 내부 로직으로 인한 성능 저하 원인: 웹 애플리케이션의 인터넷 접속 불량, 특정 파일의 업로드, 다운로드로 인한 성능 저하, 정상적으로 처리되지 않은 오류 처리로 인한 성능 저하 등

애플리케이션 성능 개선 4. ① (ㄹ) ② (ㄷ) ③ (ㄴ) ④ (ㄱ)

* 애플리케이션 성능 저하 원인
- 데이터베이스 연결 및 쿼리 실행 시 발생되는 성능 저하 원인
- 내부 로직으로 인한 성능 저하 원인
- 외부 호출(HTTP, 소켓 통신)로 인한 성능 저하 원인
- 잘못된 환경 설정이나 네트워크 문제로 인한 성능 저하 원인

애플리케이션 성능 개선 5. (ㄱ)→(ㄷ)→(ㄴ)→(ㄹ)

* 애플리케이션 성능 분석 수행 절차
- 1단계: 애플리케이션 성능 점검을 위한 도구의 유형 정리
- 2단계: 애플리케이션 성능 점검 계획서 작성
- 3단계: 애플리케이션 성능 테스트 수행
- 4단계: 성능 테스트 결과 분석을 통해 성능 저하 요인 발견

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

애플리케이션 성능 개선 6. (ㅁ)→(ㄴ)→(ㄹ)→(ㄱ)→(ㄴ)→(ㄷ)

* 애플리케이션 성능 개선 수행 절차
- 1단계: 애플리케이션 성능 개선 방안 검토
- 2단계: 코드 최적화 기법을 통한 성능 개선 방안 작성
- 3단계: 아키텍처 조정을 통한 성능 개선 방안 작성
- 4단계: 프로그램 호출 순서 조정을 통한 성능 개선 방안 작성
- 5단계: 소스 코드 품질 분석 도구를 활용하여 애플리케이션 성능 개선
- 6단계: 애플리케이션 성능 현황 관리

애플리케이션 성능 개선 7. ① 클린 코드(Clean Code) ② 나쁜 코드(Bad Code)

- 클린 코드: 잘 작성되어 가독성이 높고, 단순하며, 의존성을 줄이고, 중복을 최소화하여 깔끔하게 잘 정리된 코드를 말한다.
- 나쁜 코드: 다른 개발자가 로직을 이해하기 어렵게 작성된 코드이다. 대표적인 사례로 처리 로직의 제어가 정제되지 않고 서로 얽혀 있는 스파게티 코드, 변수나 메소드에 대한 이름 정의를 알 수 없는 코드, 동일한 처리 로직이 중복 되게 작성된 코드 등이 있다.

애플리케이션 성능 개선 8. ① 스레드 ② 정적 분석 도구 ③ 동적 분석 도구

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

애플리케이션 성능 개선 9. ① (ㄷ) ② (ㄱ) ③ (ㄴ) ④ (ㅁ) ⑤ (ㄹ)

* 클린 코드 작성 원칙
- 가독성
- 단순성
- 의존성
- 중복성
- 추상화

애플리케이션 성능 개선 10. ① (ㄹ) ② (ㄱ) ③ (ㄷ) ④ (ㄴ) ⑤ (ㅁ)

* 소스 코드 최적화 기법의 유형
- 클래스 분할 배치 기법
- 느슨한 결합 기법
- 코딩 형식 기법
- 좋은 이름 사용
- 적절한 주석문 사용 방법

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

애플리케이션 성능 개선 11. ① (ㅁ) ② (ㄹ) ③ (ㄱ) ④ (ㄴ) ⑤ (ㄷ)

* Spring MVC 컴포넌트
- Dispatcher Servlet - HandlerMapping
- Controller - View Resolver
- View

애플리케이션 성능 개선 12. 팩토리 메소드(Factory Method)

* 팩토리 메소드(Factory Method): 객체 생성을 위한 인터페이스를 정의한 후 상속한 서브 클래스를 이용하여 객체를 생성한다.

애플리케이션 성능 개선 13. (ㄴ), (ㄷ), (ㅁ)

* 소스 코드 품질 분석 도구를 활용한 애플리케이션 성능 개선 단계
- String 클래스를 StringBuffer 또는 StringBuilder 클래스로 수정하여 코딩한다.
- 루프 내 불필요한 메소드의 호출이 반복되지 않도록 코딩한다.
- 입출력 발생 최소화를 통하여 성능을 개선한다.
- System.out.println()을 사용하지 않음으로써 성능 개선한다: 파일, 콘솔에 로그를 남기면 애플리케이션 대기 시간이 발생된다. 이에 대응하여 Log4j 로거를 사용함으로써 성능을 개선한다.

애플리케이션 성능 개선 14. (ㄷ)

* 애플리케이션 성능 개선 수행 절차
- 1단계: 애플리케이션 성능 개선 방안 검토
- 2단계: 코드 최적화 기법을 통한 성능 개선 방안 작성
- 3단계: 아키텍처 조정을 통한 성능 개선 방안 작성
- 4단계: 프로그램 호출 순서 조정을 통한 성능 개선 방안 작성
- 5단계: 소스 코드 품질 분석 도구를 활용하여 애플리케이션 성능 개선
- 6단계: 애플리케이션 성능 현황 관리