[정답 및 해설] [서버프로그램 구현>개발환경 구축]

개발환경 구축 1. ① 클라이언트 ② 서버

하드웨어 환경은 사용자와의 인터페이스 역할을 하는 클라이언트 서버 개발환경과 클라이언트와 통신하여 서비스를 제공하는 서버 개발환경으로 구성된다.

개발환경 구축 2. (ㄱ), (ㄷ), (ㄹ)

(ㄴ) 웹 서버는 정적 파일들을 저장하고 관리한다.

개발환경 구축 3. ① 웹 서버 ② 웹 애플리케이션 서버 ③ 데이터베이스 서버 ④ 파일 서버

* 서버 개발환경 구성
- 웹(Web) 서버
- 웹 애플리케이션(Application) 서버
- 데이터베이스(Database) 서버
- 파일(File) 서버

개발환경 구축 4. ① (ㄹ) ② (ㅁ) ③ (ㄷ) ④ (ㅂ) ⑤ (ㄱ) ⑥ (ㄴ)

* 개발 소프트웨어
- 요구사항 관리 도구 - 설계/모델링 도구
- 구현 도구 - 빌드 도구
- 형상관리 도구 - 테스트 도구

개발환경 구축 5. 형상관리

형상관리(SCM: Software Configuration Management)란 소프트웨어의 개발 과정에서 발생하는 산출물의 변경사항을 관리하기 위한 활동이다.

개발환경 구축 6. ① (ㄹ) ② (ㄴ) ③ (ㄱ) ④ (ㄷ)

* 형상관리 절차
- 1단계: 형상 식별
- 2단계: 변경 제어
- 3단계: 형상 상태 보고
- 4단계: 형상 감사

개발환경 구축 7. 형상 식별→변경 제어→형상 상태 보고→형상 감사

* 형상관리 절차
- 1단계: 형상 식별
- 2단계: 변경 제어
- 3단계: 형상 상태 보고
- 4단계: 형상 감사

[정답 및 해설] [서버프로그램 구현>개발환경 구축]

개발환경 구축 8. (ㅅ)→(ㄱ)→(ㄷ)→(ㄴ)→(ㄹ)→(ㅂ)→(ㅁ)

* 개발환경 준비 수행 순서
- 1단계: 목표 시스템 환경 분석
- 2단계: 요구사항 분석
- 3단계: 개발 언어 선정
- 4단계: 개발 도구 선정
- 5단계: 빌드 도구 선정
- 6단계: 형상관리 도구 선정
- 7단계: 테스트 도구 선정

개발환경 구축 9. 적정성, 효율성, 이식성, 친밀성, 범용성, 알고리즘과 계산상의 난이도, 소프트웨어의 수행 환경, 자료 구조의 난이도, 개발 담당자와의 경험 지식 등

* 개발 언어 선정 기준 고려 항목
- 적정성
- 효율성
- 이식성
- 친밀성
- 범용성
- 알고리즘과 계산상의 난이도
- 소프트웨어의 수행 환경
- 자료 구조의 난이도
- 개발 담당자와의 경험 지식

개발환경 구축 10. ① (ㄹ) ② (ㄷ) ③ (ㄴ) ④ (ㅁ) ⑤ (ㄱ)

* 개발 언어 선정 기준 주요 고려 항목
- 적정성
- 효율성
- 이식성
- 친밀성
- 범용성

[정답 및 해설] [서버프로그램 구현>공통 모듈 구현]

공통 모듈 구현 1. 모듈화

모듈화(Modularity)에 대한 설명이다. 모듈화의 목적은 소프트웨어 복잡도가 감소하고, 변경이 쉬우며 프로그램 구현을 용이하게 하기 위함이다.

공통 모듈 구현 2. ① 결합도 ② 응집도

* 결합도(Coupling): 모듈 간에 상호 의존도
* 응집도(Cohesion): 모듈 안의 요소들이 서로 관련되어 있는 정도

공통 모듈 구현 3. ① 시간적 응집도 ② 절차적 응집도 ③ 순차적 응집도 ④ 기능적 응집도

* 응집도 종류
: 우연적 < 논리적 < 시간적 < 절차적 < 교환적(통신적) < 순차적 < 기능적

공통 모듈 구현 4. ① 자료(데이터) 결합도 ② 제어 결합도 ③ 공통 결합도 ④ 내용 결합도

* 결합도 종류
: 자료(데이터) < 스탬프 < 제어 < 외부 < 공통 < 내용

공통 모듈 구현 5. ① 우연적 응집도 ② 기능적 응집도 ③ 순차적 응집도 ④ 시간적 응집도 ⑤ 논리적 응집도 ⑥ 교환적 응집도

* 응집도 종류
: 우연적 < 논리적 < 시간적 < 절차적 < 교환적(통신적) < 순차적 < 기능적

공통 모듈 구현 6. ① 스탬프 결합도 ② 외부 결합도 ③ 내용 결합도 ④ 공통 결합도 ⑤ 자료 결합도

* 결합도 종류
: 자료(데이터) < 스탬프 < 제어 < 외부 < 공통 < 내용

공통 모듈 구현 7. ① interface ② implements

* interface: 인터페이스 생성
* implements: 상속 받을 인터페이스 사용

공통 모듈 구현 8. ① Fan-In ② Fan-Out

- 공유도(Fan-In): 어떤 모듈을 제어(호출)하는 상위 모듈의 개수
- 제어도(Fan-Out): 어떤 모듈에 의해 제어(호출)되는 하위 모듈의 개수

[정답 및 해설] [서버프로그램 구현>공통 모듈 구현]

공통 모듈 구현 9. ① 1 ② 2

- 모듈 B에서의 Fan-In: A
- 모듈 B에서의 Fan-Out: E, F

공통 모듈 구현 10. 재사용(Reuse)

- 재사용(Reuse)
: 이미 개발된 기능들을 파악하고 재구성하여 새로운 시스템 또는 기능 개발에 사용하기 적합하도록 최적화 시키는 작업이다.

공통 모듈 구현 11. 공통 모듈

- 공통 모듈: 공통적으로 사용할 수 있는 모듈

공통 모듈 구현 12. 정확성, 명확성, 완전성, 일관성, 추적성

* 공통 모듈 명세 기법
- 정확성(Correctness)
- 명확성(Clarity)
- 완전성(Completeness)
- 일관성(Consistency)
- 추적성(Traceability)

공통 모듈 구현 13. ① 추적성 ② 완전성 ③ 정확성 ④ 명확성 ⑤ 일관성

* 공통 모듈 명세 기법
- 정확성
- 명확성
- 완전성
- 일관성
- 추적성

공통 모듈 구현 14. ① (ㄷ) ② (ㄹ) ③ (ㄴ) ④ (ㄱ)

* 테스트 프로세스의 구성
1) 계획 및 제어 2) 분석 및 설계
3) 구현 및 실행 4) 평가
5) 완료

공통 모듈 구현 15. (ㅁ)→(ㄱ)→(ㅂ)→(ㄹ)→(ㄷ)→(ㄴ)

* 공통 모듈 테스트 수행 순서
- 1단계: 단위 테스트 케이스 작성을 위한 참조 문서를 수집한다.
- 2단계: 단위 테스트 케이스를 작성한다.
- 3단계: 작성된 단위 테스트 케이스를 내부 검토한다.
- 4단계: 작성된 단위 테스트 케이스를 고객에게 승인을 획득한다.
- 5단계: 테스트를 명세하기 위한 테스트 도구를 설정한다.
- 6단계: 테스트 결과를 명세화 한다.

[정답 및 해설] [서버프로그램 구현>공통 모듈 구현]

공통 모듈 구현 16. (ㄴ)→(ㄷ)→(ㄱ)

* 단위 테스트 케이스 작성 순서
- 1단계. 단위 테스트 방식을 결정한다.
- 2단계. 단위 테스트의 범위를 결정한다.
- 3단계. 단위 테스트의 방식과 범위에 따라 테스트 케이스를 작성한다.

공통 모듈 구현 17. ① (ㄱ) ② (ㄴ) ③ (ㄷ)

* 테스트 프로세스의 구성
1) 계획 및 제어
: 테스트의 목표와 목적을 달성하기 위해 필요한 활동을 계획하고, 계획에 준수하여 진행되고 있는지 지속적인 제어
2) 분석 및 설계
: 일반적이고 추상적인 테스트의 목적을 구체화하여 테스트 시나리오와 테스트 케이스로 변환
3) 구현 및 실행
: 테스트를 효과적이고 효율적으로 수행하기 위해 테스트 케이스들을 조합하고 테스트 수행 시 필요한 정보들을 포함하는 테스트 프로시저를 명세
4) 평가
: 계획 단계에서 정의하였던 테스트 목표에 맞게 테스트가 수행되었는지를 평가하고 진행 상황을 기록
5) 완료
: 테스트 수행 시 명세했던 조건들을 수집하고 테스트 수행 시 발생했던 사항 및 경험들을 축적

[정답 및 해설] [서버 프로그램 구현>서버 프로그램 구현]

서버 프로그램 구현 1. 프로세스

- 프로세스: 개인이나 조직이 한 개 이상의 정보 자원의 입력을 통해 가치 있는 산출물을 제공하는 모든 관련 활동들의 집합이다.

서버 프로그램 구현 2. ① (ㄷ) ② (ㅁ) ③ (ㄴ) ④ (ㄱ) ⑤ (ㄹ)

* 프로세스의 구성 요소
- 프로세스 책임자(Owner): 프로세스의 성과와 운영을 책임지는 구성원으로, 프로세스를 설계하고 지속적으로 유지하는 사람이다.
- 프로세스 맵(Map): 상위 프로세스와 하위 프로세스의 체계를 도식화하여 전체 업무의 청사진을 표현한다.
- 프로세스 Task 정의서: 기대하는 결과물을 산출하기 위해 Task(최소 업무 단위)들이 어떻게 운영되어야 하는지에 대한 문서이다.
- 프로세스 성과 지표: 프로세스의 과정과 결과를 고객 입장에서 정량적으로 표현한 성과 측정 지표이다.
- 프로세스 조직: 프로세스를 성공적으로 수행하기 위해 개인들의 업무를 유기적으로 수행하는 구성원이다.
- 경영자의 리더십(Leadership): 경영자는 프로세스의 중요성을 인식하고 기업의 경영 방침을 확고하게 해야 한다.

서버 프로그램 구현 3. 프로세스 맵

- 프로세스 맵(Map): 상위 프로세스와 하위 프로세스의 체계를 도식화하여 전체 업무의 청사진을 표현한다.

서버 프로그램 구현 4. (ㄴ)→(ㄱ)→(ㄷ)

* 세부 업무 프로세스를 확인 절차
- 1단계: 프로그램 설계서를 확인한다.
- 2단계: 화면 설계서를 확인한다.
- 3단계: 화면 레이아웃을 확인한다

서버 프로그램 구현 5. 프레임워크(Framework)

* 프레임워크(Framework)
- 효율적인 정보 시스템 개발을 위한 코드 라이브러리, 애플리케이션 인터페이스, 설정 정보 등의 집합으로서 재사용이 가능하도록 소프트웨어 구성에 필요한 기본 뼈대를 제공한다.
- 넓은 의미로 정보 시스템의 개발 및 운영을 지원하는 도구 및 가이드 등을 포함한다.

[정답 및 해설] [서버 프로그램 구현>서버 프로그램 구현]

서버 프로그램 구현 6. ① (ㄱ) ② (ㄷ) ③ (ㄴ)

* 상세 설계를 기반으로 한 담당 업무 확인 절차
: 업무 프로세스를 확인하기 위해서는 애플리케이션 설계 단계에서 작성한 프로그램 관리 대장을 통해 작성해야 할 업무 프로그램들을 확인하고 프로그램 설계서와 업무 프로세스 맵을 확인한다.
1. 프로그램 관리 대장을 확인한다.
: 프로그램 관리 대장은 설계 단계에서 작성된 구현해야 할 전체 시스템의 프로그램 목록 산출물이다.
2. 업무 프로세스 맵을 확인한다.
: 프로그램 관리 대장에서 확인한 프로세스에 대한 업무 프로세스 체계를 확인한다.

서버 프로그램 구현 7. 스프링(Spring)

* 스프링(Spring)
- 자바 플랫폼을 위한 오픈 소스 경량형 애플리케이션 프레임워크
- 동적인 웹 사이트를 개발하기 위한 여러 가지 서비스를 제공하고 있으며, 대한민국 공공기관의 웹 서비스 개발 시 사용을 권장하고 있는 전자정부 표준프레임워크의 기반 기술로서 쓰이고 있다.

서버 프로그램 구현 8. 전자정부 프레임워크

* 전자정부 프레임워크
- 우리나라의 공공부문 정보화 사업 시 효율적인 정보 시스템의 구축을 지원하기 위해 필요한 기능 및 아키텍처를 제공하는 프레임워크이다.
- 오픈 소스 기반의 범용화가 되고 공개된 기술을 활용함으로써 특정 업체의 종속성을 배제하고 사업별 공통 컴포 넌트의 중복 개발을 방지한다.

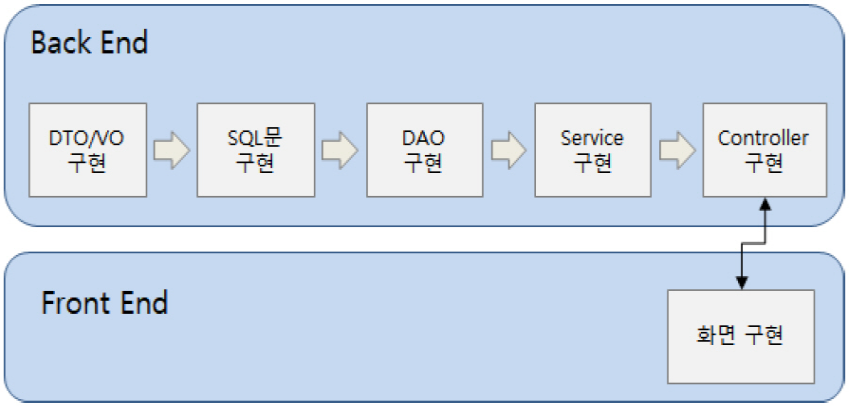
서버 프로그램 구현 9. ① (ㄴ) ② (ㄱ) ③ (ㄷ) ④ (ㄹ)

* 프레임워크의 특징
- 모듈화(modularity): 프레임워크는 인터페이스에 의한 캡슐화를 통해서 모듈화를 강화하고 설계와 구현의 변경에 따르는 영향을 극소 화하여 소프트웨어의 품질을 향상시킨다.
- 재사용성(reusability): 프레임워크 컴포넌트를 재사용하는 것은 소프트웨어의 품질을 향상시키고, 개발자의 생산성도 높여 준다.
- 확장성(extensibility): 프레임워크는 다형성(polymorphism)을 통해 애플리케이션이 프레임워크의 인터페이스를 확장할 수 있게 한다.
- 제어의 역흐름(inversion of control): 개발자가 관리하고 통제해야 하는 객체들의 제어 권한을 프레임워크에 넘겨 생산성을 향상시킨다.

[정답 및 해설] [서버 프로그램 구현>서버 프로그램 구현]

서버 프로그램 구현 10. ① 서버 영역(Back End) ② 화면 영역(Front End)

* 세부 업무 프로세스를 기반으로 업무 프로그램 구현
- 프로그램을 구현하기 위해 서버 영역(Back End)과 화면 영역(Front End)을 구현한다.
- 순서에 상관없이 구현을 해도 무방하다.



서버 프로그램 구현 11. DAO

- DAO(Data Access Object): 서버 프로그램 구현을 위해 생성하는 객체 중 하나로, 데이터베이스에 접근하고 SQL을 활용하여 데이터를 실제로 조작하는 데이터 접근 객체

서버 프로그램 구현 12. ① Model ② View ③ Controller

* MVC 구조
MVC는 구현하려는 전체 어플리케이션을 Model, View, Controller로 구분하여 유저 인터페이스와 비즈니스 로직을 서로 분리하여 애플리케이션의 시각적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 애플리케이션을 만들 수 있다.
- Model: 사용자 요청을 처리해 사용자에게 출력할 데이터를 만 드는 요소
- View: 모델이 처리한 결과를 화면에 보여주는 요소
- Controller: 사용자 요청을 받아 그 요청을 처리할 Model을 호출하고 Model이 처리 후 결과를 View 에게 전달

서버 프로그램 구현 13. Controller

- Controller: 서버 프로그램 개발을 위해 구현되는 모듈 중 하나로, 사용자의 요청에 적절한 서비스를 호출하여 그 결과를 사용자에게 반환하는 객체

[정답 및 해설] [서버 프로그램 구현>서버 프로그램 구현]

서버 프로그램 구현 14. (ㄴ), (ㄷ), (ㅁ), (ㅂ)

* 소프트웨어 테스트의 원칙
- 개발자가 자신이 개발한 프로그램 및 소스코드를 테스팅 하지 않는다.
- 효율적인 결함 제거 법칙 사용(낚시의 법칙, 파레토의 법칙): 효율적으로 결함을 발견, 가시화, 제거, 예방의 순서로 하여 정량적으로 관리할 수 있어야 한다.
- 완벽한 소프트웨어 테스팅은 불가능하다.
- 테스트는 계획 단계부터 해야 한다.
- 살충제 패러독스: 동일한 테스트 케이스로 반복 실행하면 더 이상 새로운 결함을 발견할 수 없으므로 주기적으로 테스트 케이스를 점검하고 개선해야 한다
- 오류-부재의 궤변: 사용자의 요구 사항을 만족하지 못한다면 오류를 발견하고 제거해도 품질이 높다고 말할 수 없다.

서버 프로그램 구현 15. 테스트 커버리지

* 소프트웨어 테스트의 명세
1. 테스트 결과 정리: 테스트가 완료되면 테스트 계획과 테스트 케이스 설계부터 단계별 테스트 시나리오, 테스트 결과까지 모두 포함 된 문서를 일관성 있게 작성한다.
2. 테스트 요약 문서 : 테스트 계획, 소요 비용, 테스트 결과에 의해 판단 가능한 대상 소프트웨어의 품질 상태를 포함한 요약 문서를 작 성한다.
3. 품질 상태: 품질 상태는 품질 지표인 테스트 성공률, 발생한 결함의 수와 결함의 중요도, 테스트 커버리지 등이 포함된다.
- 테스트 커버리지: 프로그램의 테스트가 충분히 되었는가의 수행 정도를 퍼센트로 나타내는 지표
4. 테스트 결과서 : 결함에 관련된 내용을 중점적으로 기록하며, 결함의 내용, 결함의 재현 순서를 상세하게 기록한다.
5. 테스트 실행 절차 및 평가: 단계별 테스트 종료 시 테스트 실행 절차를 리뷰하고 결과에 대한 평가를 수행하며, 그 결과에 따라 실행 절차를 최적화하여 다음 테스트에 적용한다.

[정답 및 해설] [서버 프로그램 구현>배치 프로그램 구현]

배치 프로그램 구현 1. 배치 프로그램

* 배치 프로그램
- 배치 프로그램은 사용자와의 상호 작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 것을 말한다.
- 배치 프로그램의 필수 요소에는 대용량 데이터, 자동화, 견고함, 안정성, 성능이 있다.
- 배치 프로그램은 자동 수행 주기에 따라 정기 배치, 이벤트성 배치, On-Demand 배치로 나누어진다.

배치 프로그램 구현 2. 배치 스케줄러

- 배치 스케줄러: 배치 프로그램이 일괄 처리를 위해 주기적으로 발생하거나 반복적으로 발생하는 작업을 원활히 수행하도록 지원하는 도구로, 주로 사용되는 도구로는 스프링 배치, 쿼츠 스케줄러, 크론 등이 있다.

배치 프로그램 구현 3. ① (ㄱ) ② (ㄹ) ③ (ㄴ) ④ (ㄷ)

* 배치 프로그램의 필수 요소
- 대용량 데이터: 대용량의 데이터를 처리할 수 있어야 한다.
- 자동화: 심각한 오류 상황 외에는 사용자의 개입 없이 동작해야 한다.
- 견고함: 유효하지 않은 데이터의 경우도 처리해서 비정상적인 동작 중단이 발생하지 않아야 한다.
- 안정성: 어떤 문제가 생겼는지, 언제 발생했는지 등을 추적할 수 있어야 한다.
- 성능: 주어진 시간 내에 처리를 완료할 수 있어야 하고, 동시에 동작하고 있는 다른 애플리케이션을 방해하지 말아야 한다.

배치 프로그램 구현 4. ① (ㄴ) ② (ㄱ) ③ (ㄷ)

* 배치 프로그램 자동 수행 주기에 따른 분류
- 정기 배치: 일/주/월과 같이 정해진 기간에 정기적으로 수행
- 이벤트성 배치: 특정 조건을 설정해 두고 조건이 충족될 때만 수행
- On-Demand 배치: 사용자 요청에 의해 수행

[정답 및 해설] [서버 프로그램 구현>배치 프로그램 구현]

배치 프로그램 구현 5. 스프링 배치(Spring Batch)

* 스프링 배치(Spring Batch)
- 스프링 배치는 Spring Source사와 Accenture사의 공동 작업으로 2007년에 탄생한 배치 기반 오픈 소스 프레임워크이다.
- 스프링의 DI, AOP 및 다양한 엔터프라이즈 지원 기능을 사용하며 스프링 프레임워크의 특성을 그대로 가져와 스프링이 가지고 있는 다양한 기능들을 모두 사용할 수 있다.
- 배치 처리 시 공통적으로 필요한 컴포넌트를 제공한다.
- 트랜잭션 관리, 작업 재시작, 로그 관리, 추적, 작업 처리 통계, 모니터링 등의 다양한 기능을 제공한다.

배치 프로그램 구현 6. (ㄱ), (ㄷ), (ㅂ)

* 스프링 프레임워크 주요 특징
- IoC(Inversion Of Control, 제어의 역행, 제어의 역흐름)
- DI(Dependency Injection, 의존성 주입)
- AOP(Aspects Oriented Programming, 관점 지향 프로그래밍)

배치 프로그램 구현 7. ① 페이징(paging) ② Chunk

* 스프링 배치의 핵심 컴포넌트
- Job: 배치 처리를 의미하는 애플리케이션 컴포넌트이다.
- Step: Job의 각 단계를 의미한다. Job은 일련의 연속된 Step으로 구성된다.
- JPA(Java Persistence API): [1] 페이징 기능을 제공한다.
- Job Repository: Job Execution(작업 실행) 관련 메타데이터를 저장하는 기반 컴포넌트이다.
- Item: Data Source로부터 읽거나 Data Source로 저장하는 각 레코드를 의미한다.
- [2]Chunk: 특정 크기를 갖는 아이템 목록을 의미한다.

[1] 페이징: 한정된 기억 용량으로 될 수 있는대로 다수의 프로그램을 넣고, 동시에 처리할 수 있도록 하기 위해 프로그램 을 한 번에 처리할 수 있는 적당한 크기로 분할하여 페이지 단위로 처리하는 것.
[2]Chunk: 배치에서 한번에 트랜잭션으로 처리할 단위.

배치 프로그램 구현 8. Quartzz(쿼츠) 스케줄러

* Quartzz(쿼츠) 스케줄러
- 쿼츠 스케줄러는 스프링 프레임워크로 개발되는 응용 프로그램들의 일괄 처리를 위한 다양한 기능을 제공하는 오픈 소스 라이브러리이다.
- 수행할 작업과 실행 Trigger를 분리하여 일괄 처리 작업에 유연성을 제공한다.

[정답 및 해설] [서버 프로그램 구현>배치 프로그램 구현]

배치 프로그램 구현 9. ① -e ② -l ③ -r

* Cron
- 편집기에서 요일/월/일/시/분을 기준으로 수행할 명령어를 지정한다.
- Cron은 리눅스의 스케줄러 도구로 crontab 명령어를 통해 작업을 예약할 수 있다.

배치 프로그램 구현 10. 디버그(Debug) 또는 디버깅(Debugging)

- 디버그: 컴퓨터 프로그램 개발 단계 중에 발생하는 시스템의 논리적인 오류나 비정상적 연산을 찾아내고 그 원인을 밝히고 수정하는 작업 과정을 뜻한다. 일반적으로 디버깅 하는 방법으로 테스트 상의 체크, 기계를 사용하는 테스트, 실제 데이터를 사용해 테스트 하는 법이 있다.
- 디버거: 디버그를 돕는 도구이다. 디버거는 디버깅을 하려는 코드에 중단점을 지정하여 프로그램 실행을 중단하고, 코드를 단계적으로 실행하여 저장된 값을 확인 할 수 있도록 지원한다.