

대분류 / 20
정보통신

중분류 / 01
정보기술

소분류 / 02
정보기술개발

세분류 / 02
응용SW엔지니어링

학습모듈 / 03

03

애플리케이션 구현

LM2001020203_14v2

응용SW엔지니어링 학습모듈

01. 요구사항 확인



02. 애플리케이션 설계



03. 애플리케이션 구현



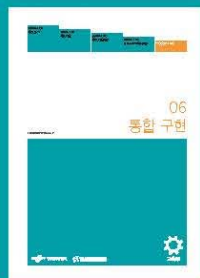
04. 화면 구현



05. 데이터 입출력 구현



06. 통합 구현



07. 개발자 테스트



08. 정보시스템 이행



09. 제품소프트웨어 패키징



10. 소프트웨어공학 활용

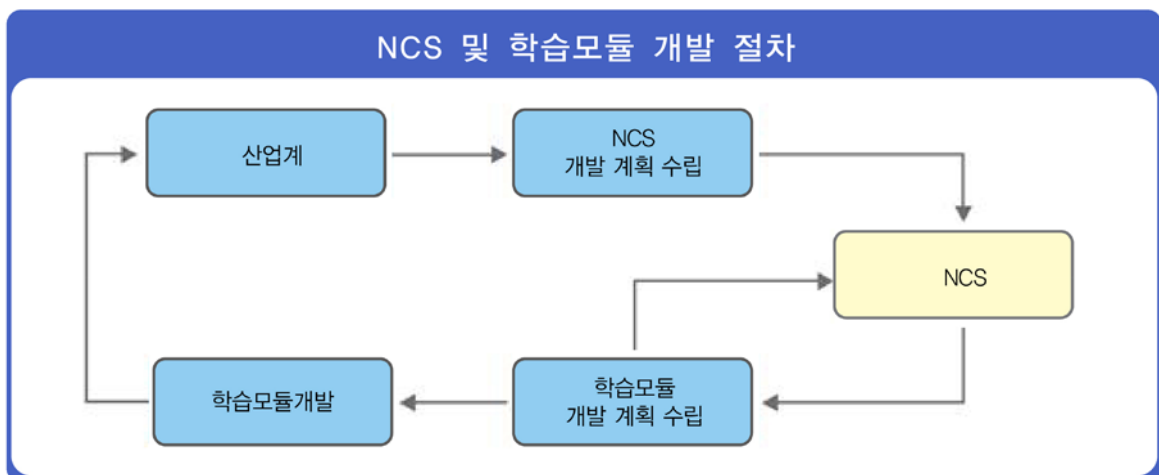


NCS 학습모듈의 이해

※ 본 학습모듈은 「NCS 국가직무능력표준」 사이트(<http://www.ncs.go.kr>) 에서 확인 및 다운로드 할 수 있습니다.

(1) NCS 학습모듈이란?

- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, NCS 학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다. NCS 학습모듈은 구체적 직무를 학습할 수 있도록 이론 및 실습과 관련된 내용을 상세하게 제시하고 있습니다.

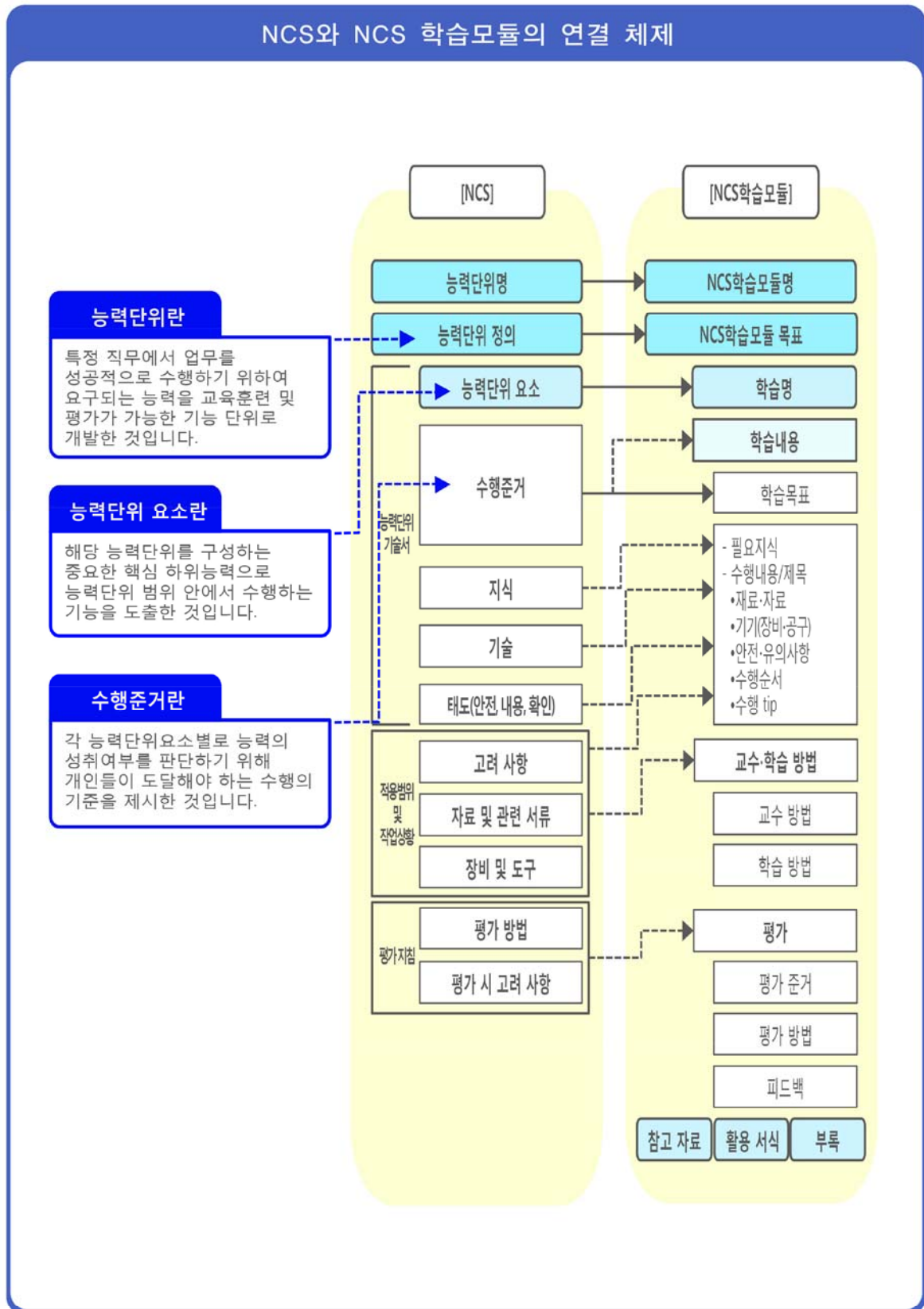


- NCS 학습모듈은 다음과 같은 특징을 가지고 있습니다.

첫째, NCS 학습모듈은 산업계에서 요구하는 직무능력을 교육훈련 현장에 활용할 수 있도록 성취목표와 학습의 방향을 명확히 제시하는 가이드라인의 역할을 합니다.

둘째, NCS 학습모듈은 특성화고, 마이스터고, 전문대학, 4년제 대학교의 교육기관 및 훈련기관, 직장교육기관 등에서 표준교재로 활용할 수 있으며 교육과정 개편 시에도 유용하게 참고할 수 있습니다.

- NCS와 NCS 학습모듈 간의 연결 체제를 살펴보면 아래 그림과 같습니다.



(2) NCS 학습모듈의 체계

- NCS 학습모듈은 1.학습모듈의 위치, 2.학습모듈의 개요, 3.학습모듈의 내용 체계, 4.참고 자료, 5.활용 서식/부록 으로 구성되어 있습니다.

1. NCS 학습모듈의 위치

- NCS 학습모듈의 위치는 NCS 분류 체계에서 해당 학습모듈이 어디에 위치하는지를 한 눈에 볼 수 있도록 그림으로 제시한 것입니다.

예시 : 이·미용 서비스 분야 중 네일미용 세분류

NCS-학습모듈의 위치

대분류	이용·숙박·여행·오락·스포츠
중분류	이·미용
소분류	아·미용 서비스

세분류	능력단위	학습모듈명
헤어미용	네일 샵 위생 서비스	네일샵 위생서비스
피부미용	네일 화장을 제거	네일 화장을 제거
메이크업	네일 기본 관리	네일 기본관리
네일미용	네일 랩	네일 랩
이용	네일 팁	네일 팁
	젤 네일	젤 네일
	아크릴릭 네일	아크릴 네일
	평면 네일아트	평면 네일아트
	융합 네일아트	융합 네일아트
	네일 샵 운영관리	네일샵 운영관리

학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용 단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어서 1개의 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습모듈로 나누어 개발할 수도 있습니다.

2. NCS 학습모듈의 개요

구 성

- NCS 학습모듈 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로서 **학습모듈의 목표**, **선수 학습**, **학습모듈의 내용 체계**, **핵심 용어**로 구성되어 있습니다.

학습모듈의 목표	해당 NCS 능력단위의 정의를 토대로 학습목표를 작성한 것입니다.
선수 학습	해당 학습모듈에 대한 효과적인 교수·학습을 위하여 사전에 이수해야 하는 학습모듈, 학습 내용, 관련 교과목 등을 기술한 것입니다.
학습모듈의 내용 체계	해당 NCS 능력단위요소가 학습모듈에서 구조화된 방식을 제시한 것입니다.
핵심 용어	해당 학습모듈의 학습 내용, 수행 내용, 설비·기자재 등 가운데 핵심적인 용어를 제시한 것입니다.

활 용 안 내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈

네일 기본관리 학습모듈의 개요

학습모듈의 목표

고객의 네일 보호와 미적 요구 충족을 위하여 효과적인 네일 관리로 프리에지 형태 만들기, 큐티클 정리하기, 컬러링하기, 보습제 도포하기, 마무리를 할 수 있다.

선수학습

네일숍 위생서비스(JM1201010401_14v2)

학습모듈의 내용체계

학습	학습내용	NCS 능력단위요소		
		코드번호	요소명칭	수준
1. 프리에지 형태 만들기	1-1. 네일 파일에 대한 이해와 활용 1-2. 프리에지 형태 파일링	1201010403_12v2.1	프리에지 모양 만들기	3
2. 큐티클 정리하기	2-1. 네일 기본관리 매뉴얼 이해 2-2. 큐티클 관리	1201010403_14v2.2	큐티클 정리하기	3
3. 컬러링하기	3-1. 컬러링 매뉴얼 이해 3-2. 컬러링 방법 선정과 작업 3-3. 쉘 컬러링 작업	1201010403_14v2.3	컬러링	3
4. 보습제 도포하기	4-1. 보습제 선정과 도포 4-2. 각질제거	1201010403_14v2.4	보습제 바른기	2
5. 네일 기본관리 마무리하기	5-1. 유휴기 제거 5-2. 네일 기본관리 마무리와 정리	1201010403_14v2.5	마무리하기	3

핵심 용어

프리에지, 니퍼, 푸서, 플리시, 네일 파일, 스웨이형, 스웨이 오프형, 라운드형, 오발형, 포인트형

학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악할 수 있도록 지도하는 것이 필요합니다.

선수학습은

교수자나 학습자가 해당 모듈을 교수 또는 학습하기 이전에 이수해야 할 학습내용, 교과목, 핵심 단어 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것이 필요합니다.

핵심 용어는

학습모듈을 통해 학습되고 평가되어야 할 주요 용어입니다. 또한 당해 모듈 또는 타 모듈에서도 핵심 용어를 사용하여 학습내용을 구성할 수 있으며, 「NCS 국가 직무능력표준」 사이트(www.ncs.go.kr)에서 색인(찾아보기) 중 하나로 이용할 수 있습니다.

3. NCS 학습모듈의 내용 체계

구 성

- NCS 학습모듈의 내용은 크게 **학습**, **학습 내용**, **교수·학습 방법**, **평가** 로 구성되어 있습니다.

학습	해당 NCS 능력단위요소 명칭을 사용하여 제시한 것입니다. 학습은 크게 학습 내용, 교수·학습 방법, 평가로 구성되며 해당 NCS 능력단위의 능력단위 요소별 지식, 기술, 태도 등을 토대로 학습 내용을 제시한 것입니다.
학습 내용	학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성하였으며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 업무의 표준화된 프로세스에 기반을 두고 실제 산업현장에서 이루어지는 업무활동을 다양한 방식으로 반영한 것입니다.
교수·학습 방법	학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간의 상호 작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다.
평가	평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거, 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다.

활 용 안 내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈의 내용

학습 1	프리에지 형태 만들기(LM1201010403_14v2.1)
학습 2	큐티를 정리하기(LM1201010403_14v2.2)
학습 3	컬러링하기(LM1201010403_14v2.3)
학습 4	보습제 도포하기(LM1201010403_14v2.4)
학습 5	네일 기본관리 마무리하기(LM1201010403_14v2.5)

학습은

해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 학습은 일반교과의 '대단원'에 해당되며, 모듈을 구성하는 가장 큰 단위가 됩니다. 또한 완성된 직무를 수행하기 위한 가장 기본적인 단위로 사용할 수 있습니다.

학습내용은

요소 별 수행준거를 기준으로 제시하였습니다. 일반교과의 '중단원'에 해당합니다.

학습목표는

모듈 내의 학습내용을 이수했을 때 학습자가 보여줄 수 있는 행동수준을 의미합니다. 따라서 일반 수업시간의 과목목표로 활용할 수 있습니다.

필요지식은

해당 NCS의 지식을 토대로 해당 학습에 대한 이해와 성과를 높이기 위해 알아야 할 주요 지식을 제시하였습니다. 필요지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행순서 내용과 연계하여 교수·학습으로 진행할 수 있습니다.

3-1. 컬러링 매뉴얼 이해

학습목표

- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 얇게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다.

필요 지식 /

□ 컬러링 매뉴얼

컬러링 작업 전, 이세톤 또는 네일 폴리시 리무버를 사용하여 손톱표면과 큐티를 주변, 손톱 밑 부분까지 깨끗하게 유분기를 제거해야 한다. 컬러링의 순서는 Base coating 1회 → Polishing 2회 → 컬러수정 → Top coating 1회 → 최종수정의 순서로 한다. 베이스코트는 착색을 방지하고 발림성 향상을 위해 가장 먼저 도포하며 컬러링의 마지막에 컬러의 유지와 광택을 위해 톱코트를 도포한다. 네일 보강제(Nail Strengthner)를 바를 시에는 베이스코트를 도포하기 전에 사용한다.

수행 내용 / 컬러링 매뉴얼 실습하기

재료·자료

- 컬러링 관련 네일 미용 자료들
- 컬러바구니, 베이스코트, 네일 폴리시, 톱코트, 오렌지우드스티, 탈지면, 폴리시리무버, 디스펜서 등

기기(장비·공구)

- 컴퓨터, 빔 프로젝터, 스크린 등

안전·유의사항

- 컬러링 재료들의 냄새를 직접적으로 맡지 않도록 유의한다.
- 컬러링 제품들이 대부분 유리병에 들어 있기 때문에 깨지지 않도록 각별히 조심한다.
- 컬러링 제품들은 상온에 마르기 때문에 개봉 후 뚜껑을 잘 닫도록 한다.

수행 순서

[1] 네일 폴리시를 바르게 잡는다.

1. 손바닥에 네일 폴리시를 놓고 약지 소지를 이용하여 네일 폴리시를 잡는다.
2. 폴리시를 왼 손의 엄지와 검지와 고객의 작업손가락을 잡는다.
3. 폴리시를 왼 손의 중지 손가락을 굳게 펴서 받침대가 되도록 한다.
4. 반대편 손으로 네일 폴리시의 뚜껑을 열고 소지 손가락을 펴서 네일 폴리시를 왼 중지 손가락 위에 받쳐놓는다.
5. 다양한 형태의 폴리시를 잡아본다.

수행 tip

- 흰색이 많이 섞인 네일 폴리시의 경우는 붓의 각도를 높이 세워서 빠르게 브러시 작업을 해야 붓 자국이 나지 않는다.
- 컬러링은 기본 2회 정도이나 컬러에 따른 도포량과 컬러감에 따라 1~3회 사이로 증감할 수 있다.

수행 내용은

모듈에 제시한 것 중 기술(Skill)을 습득하기 위한 실습 과제로 활용할 수 있습니다.

재료·자료는

수행 내용을 수행하는데 필요한 재료 및 준비물로 실습 시 필요 준비물로 활용할 수 있습니다.

기기(장비·공구)는

수행 내용을 수행하는데 필요한 기본적인 장비 및 도구를 제시하였습니다. 제시된 기기 외에도 수행에 필요한 다양한 도구나 장비를 활용할 수 있습니다.

안전·유의사항은

수행 내용을 수행하는데 안전상 주의해야 할 점 및 유의사항을 제시하였습니다. 수행 시 유념해야 하며, NCS의 고려사항도 추가적으로 활용할 수 있습니다.

수행 순서는

실습과제의 진행 순서로 활용할 수 있습니다.

수행 tip은

수행 내용에서 수행의 수월성을 높일 수 있는 아이디어를 제시하였습니다. 따라서 수행tip은 지도상의 안전 및 유의사항 외에 전반적으로 적용되는 주안점 및 수행과제 목적에 대한 보충설명, 추가사항 등으로 활용할 수 있습니다.

학습3 교수·학습 방법

교수·학습 방법은

학습목표를 성취하는데 필요한 교수 방법과 학습 방법을 제시하였습니다.

교수 방법

- 컬러링 제품의 성분과 컬러별 정도의 차이, 베이스코트와 톱코트의 역할, 폴리시 잡는 방법, 큐어링 시간 등의 내용을 화면 자료와 함께 설명한다.
- 서식지를 활용하여 네일 컬러링 방법을 그림으로 그려 보게 한 뒤, 다양한 컬러링의 매뉴얼을 그려서 숙지하도록 한다.
- 겔 컬러링 시 유의사항을 계속 숙지시키도록 하며, 큐어링 시간에 대해 작성하도록 한다.

교수 방법은

해당 학습활동에 필요한 학습내용, 학습내용과 관련된 학습 자료명, 자료 형태, 수행내용의 진행 방식 등에 대하여 제시하였습니다. 또한 학습자의 수업참여도를 제고하기 위한 방법 및 수업진행상 유의사항 등도 제시하였습니다. 선수학습이 필요한 학습을 학습자가 숙지하였는지 교수자가 확인하는 과정으로 활용할 수도 있습니다.

학습 방법

- 컬러링을 위한 재료의 필요성과 사용방법을 숙지하고 컬러링 매뉴얼 과정에 맞추어 작업 내용을 이해한다.
- 컬러링의 다양성에 대한 용어를 숙지하고 진행과정에 맞추어 내용을 작업한다.
- 겔 컬러링 시 적합한 큐어링 시간을 선택해서 큐어링 해본다.

학습 방법은

해당 학습활동에 필요한 학습자의 자기주도적 학습 방법을 제시하였습니다. 또한 학습자가 숙달해야 할 실기능력과 학습과정에서 주의해야 할 사항 등으로 제시하였습니다. 학습자가 학습을 이수하기 전에 반드시 숙지해야 할 기본 지식을 학습하였는지 스스로 확인하는 과정으로 활용할 수 있습니다.

학습3 평가

평가 준거

- 평가자는 학습자가 학습 목표 및 평가 항목에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습내용	평가항목	성취수준		
		상	중	하
컬러링 매뉴얼 이해	<ul style="list-style-type: none"> 고객의 요구에 따라 네일 폴리시 색상의 질감을 만들기 위한 베이스코트를 아주 얇게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시를 일찍 얇이 균일하게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다. 			

평가 방법

- 작업장 평가

학습내용	평가항목	성취수준		
		상	중	하
컬러링 매뉴얼 이해	<ul style="list-style-type: none"> 고객의 요구에 따라 네일 폴리시 색상의 질감을 만들기 위한 베이스코트를 아주 얇게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시를 일찍 얇이 균일하게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다. 			

피드백

- 작업장 평가
 - 작업 결과물을 확인하여 수정사항을 제시하고 수정 부분을 인지하도록 한다.

평가는

해당 NCS 능력단위 평가방법과 평가 시 고려 사항을 준용하여 작성하였습니다. 교수자 및 학습자가 평가항목 별 성취수준을 확인하는데 활용할 수 있습니다.

평가 준거는

학습자가 해당 학습을 어느 정도 성취하였는지를 평가하기 위한 기준을 제시하고 있습니다. 학습목표와 연계하여 단위수업 시간에 평가항목 별 성취수준을 평가하는데 활용할 수 있습니다.

평가 방법은

NCS 능력단위의 평가방법을 준용하였으며, 평가 준거에 따른 평가방법을 2개 이상 제시하였습니다. 평가방법으로는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS의 능력단위 요소 별 수행 수준을 평가하는데 가장 적절한 방법을 선정하여 활용할 수 있습니다.

피드백은

평가 후에 학습자들에게 평가 결과를 피드백하여 부족한 부분을 알려주고, 학습 결과가 미진한 경우, 해당 부분을 다시 학습하여 학습목표를 달성하는 데 활용할 수 있습니다.

4. 참고 자료

참고자료

- 김미원(2011). 『Nail Study』. 서울: 사)한국네일저서서비스협회.
- 민광경(2015). 『미용사(네일)평가』. 서울: 예문사.
- 박은국(2014). 『네일미용』. 서울: 정담미디어.

참고자료는


해당 학습모듈의 필요지식에 대한 출처와 인용한 참고자료 및 사이트를 제시하였습니다.

5. 활용 서식/부록

활용서식

프리에지 형태 실습지

1. 프리에지 형태의 이해

모양	이름	특징
	{ Square nail }	<ul style="list-style-type: none"> 강한 느낌의 사각형태 네일의 양끝 모서리 부분이 90° 사각의 형태이다. 발톱의 형태 활용 새인성 발톱의 보정시에 적용

활용서식은

평가 서식, 실습시트 등 교수학습 시 활용 가능한 다양한 서식들로 구성하였습니다. 과제 진행에서 평가에 이르기까지 필요한 서식을 해당 학습모듈의 특성에 맞춰 개발하거나 기존의 양식을 활용하여 제시하였습니다.

부록

네일 기본관리 도구와 재료 목록

목록	비고	준비
위생가운	흰색	작업자 착용
위생 마스크	흰색	작업자 착용
보호안경	투명한 렌즈 (안경으로 대체 가능)	작업자 착용
세면접리함	재질, 색상 무관	작업대

부록은

활용서식 이외에 교수학습과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

[NCS-학습모듈의 위치]

대분류	정보통신	
중분류	정보기술	
소분류	정보기술개발	

세분류	능력단위	학습모듈명
SW아키텍처		
응용SW 엔지니어링	요구사항확인	요구사항확인
시스템 엔지니어링	애플리케이션 설계	애플리케이션 설계
DB엔지니어링	애플리케이션 구현	애플리케이션 구현
NW엔지니어링	화면 구현	화면 구현
보안 엔지니어링	데이터 입출력 구현	데이터 입출력 구현
UI/UX 엔지니어링	통합 구현	통합 구현
시스템SW 엔지니어링	개발자 테스트	개발자 테스트
	정보시스템 이행	정보시스템 이행
	제품소프트웨어 패키징	제품소프트웨어 패키징
	소프트웨어공학 활용	소프트웨어공학 활용

차 례

학습모듈의 개요	1
----------	---

학습 1. 개발 환경 구축하기

1-1 개발환경 사양 이해	3
1-2 개발환경 구축	9
• 교수·학습 방법	20
• 평가	21

학습 2. 공통 모듈 구현하기

2-1 공통 모듈 작성	23
2-2 공통 모듈 테스트	31
• 교수·학습 방법	45
• 평가	46

학습 3. 서버 프로그램 구현하기

3-1 서버 프로그램 작성	48
3-2 서버 프로그램 테스트	56
• 교수·학습 방법	59
• 평가	60

학습 4. 배치 프로그램 구현하기

4-1 배치 프로그램 작성	62
4-2 배치 프로그램 테스트	71
• 교수·학습 방법	76
• 평가	77

학습 5. 개발자 단위 테스트하기

5-1 단위테스트 계획수립	79
5-2 단위 테스트 수행 및 검증	83
5-3 단위 테스트 결함 관리	86
• 교수·학습 방법	91
• 평가	92

학습 6. 애플리케이션 성능 개선하기

6-1 애플리케이션 성능측정과 개선	94
• 교수·학습 방법	101
• 평가	102

참고 자료	104
-------------	-----

애플리케이션 구현 학습모듈의 개요

학습모듈의 목표

응용 소프트웨어 개발에 필요한 환경을 구축하고, 애플리케이션 설계를 바탕으로 공통 모듈, 서버 프로그램과 배치 프로그램을 구현하고, 단위테스트를 수행할 수 있다.

선수학습

SW아키텍처 이행(2001020106_14v2), 개발언어 지식, 데이터베이스 지식, 알고리즘

학습모듈의 내용체계

학습	학습내용	NCS 능력단위요소		
		코드번호	요소명칭	수준
1. 개발환경 구축하기	1-1 개발환경 사양 이해 1-2 개발환경 구축	2001020203_14v2.1	개발환경 구축하기	2
2. 공통 모듈 구현하기	2-1 공통 모듈 작성 2-2 공통 모듈 테스트	2001020203_14v2.2	공통 모듈 구현하기	4
3. 서버 프로그램 구현하기	3-1 서버 프로그램 작성 3-2 서버 프로그램 테스트	2001020203_14v2.3	서버 프로그램 구현하기	3
4. 배치 프로그램 구현하기	4-1 배치 프로그램 작성 4-2 배치 프로그램 테스트	2001020203_14v2.4	배치 프로그램 구현하기	3
5. 개발자 단위 테스트하기	5-1 단위테스트 계획수립 5-2 단위테스트 수행 및 검증 5-3 단위테스트 결함 관리	2001020203_14v2.5	개발자 단위 테스트하기	2
6. 애플리케이션 성능 개선하기	6-1 애플리케이션 성능측정과 개선	2001020203_14v2.6	애플리케이션 성능 개선하기	4

핵심 용어

요구사항, 시스템 아키텍처, 공통 모듈, 서버, 형상관리, 단위 테스트

학습 1	개발 환경 구축하기 (LM2001020203_14v2.1)
학습 2	공통 모듈 구현하기(LM2001020203_14v2.2)
학습 3	서버 프로그램 구현하기(LM2001020203_14v2.3)
학습 4	배치 프로그램 구현하기(LM2001020203_14v2.4)
학습 5	개발자 단위 테스트하기(LM2001020203_14v2.5)
학습 6	애플리케이션 성능 개선하기(LM2001020203_14v2.6)

1-1. 개발환경 사양 이해

학습 목표

- 응용 소프트웨어 개발에 필요한 하드웨어 및 소프트웨어의 필요 사항을 검토하고 이에 따라, 개발환경에 필요한 준비를 수행할 수 있다.

필요 지식 /

① 개발환경 구축의 이해

개발환경을 구축하기 위해서는 우선 해당 프로젝트의 목적 및 구축 설계에 대한 명확한 이해가 필요하며 이에 따라 하드웨어, 소프트웨어의 선정이 이루어져야 한다. 또한 개발에 사용되는 제품들의 성능과 사용 편의성, 그리고 라이선스 등에 대한 내용도 파악해야 한다.

② 개발환경 구축에 필요한 소프트웨어와 하드웨어에 대한 이해

적합한 개발환경 구성을 위해서는 어떤 종류의 개발 소프트웨어들이 존재하며 각 종류마다 어떤 특성을 갖는지에 대해 판단할 수 있어야 한다. 그리고 이러한 소프트웨어가 설치되는 하드웨어에 대한 이해도 필요하다.

1. 개발을 위해 사용되는 소프트웨어의 종류와 특성

(1) 구현도구

프로그램을 개발할 때 가장 많이 사용되는 도구로서 코드의 작성 및 편집, 디버깅 등과 같은 다양한 작업이 가능하며 Eclipse, NetBeans, IntelliJ 등 다양한 소프트웨어들이 사용되고 있다. 구현에 사용되는 소프트웨어는 어떤 프로그래밍 언어로 개발 되는지에 따라 선택하여 사용한다.

(2) 테스트 도구

개발 과정 중에 필요한 테스트에 사용되는 소프트웨어 도구들로 코드의 테스트, 테스트에 대한 리포팅 및 분석 등의 작업이 가능하다. 사용되는 도구들에는 JUnit, Spring Test 등이 있다.

(3) 형상관리 도구

대다수의 프로젝트들은 다수의 개발자들로 구성된 팀 단위로 프로젝트가 진행되며 개발자들의 전체 소스 및 사용되는 리소스들에 대한 관리와 히스토리 관리를 위해 형상관리 도구가 사용된다. 대표적인 형상관리 도구로는 Git, CVS, Subversion 등이 있다.

(4) 빌드 도구

개발자가 작성한 코드에 대한 빌드 및 배포, 그리고 프로젝트에 사용되는 다양한 구성요소들과 라이브러리에 대한 의존성 관리에 사용하는 도구로 Ant, Maven 등이 있다.

2. 개발환경 하드웨어의 이해

프로그램 개발에 사용되는 소프트웨어들은 사용되는 컴퓨터 그리고 OS에 따라 적합한 것을 선정하여야 한다. 우선적으로, 개발에 사용되는 컴퓨터 하드웨어의 사양을 파악하여 해당 컴퓨터 성능에 적합한 소프트웨어를 선정하여야 하며, 이후에는 OS 종류에 따라 제공되는 개발 소프트웨어를 선택하여 사용한다.

수행 내용 / 개발환경 사양 이해하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

다음 예시를 통해 개발환경 구축에 필요한 도구에 대해 파악해 보자.

1. 프로젝트 주제: 웹 애플리케이션 개발
2. 개발 언어: Java, HTML5, CSS3, Servlet/JSP, JavaScript, jQuery, Ajax, SQL
3. 개발 인원: 15명
4. 개발 기간: 3개월
5. 개발 H/W 사양
 - 프로세서: Intel(R) Core(TM) i7-4700HQ CPU 2.40GHz
 - 메모리: 16GB
 - OS: Windows8 64bit

① 사용 언어 및 하드웨어 사양을 고려한 구현 도구를 선정한다.

1. 개발언어의 선정

(1) 개발언어의 선정기준

<표 1-1> 개발언어 선정기준

고려 항목	설명
적정성	대상 업무의 성격, 즉 개발하고자 하는 시스템이나 응용프로그램의 목적에 적합해야 한다.
효율성	프로그래밍의 효율성이 고려되어야 한다.
이식성	일반적인 PC 및 OS에 개발환경이 설치 가능해야 한다.
친밀성	프로그래머가 그 언어를 이해하고 사용할 수 있어야 한다.
범용성	다양한 과거 개발 실적이나 사례가 존재하고 광범위한 분야에 사용되고 있어야 한다.

※ 위 고려 항목은 참고사항으로 활용되며, 위 항목들 이외에도 알고리즘과 계산상의 난이도, 소프트웨어의 수행환경, 자료 구조의 난이도, 개발 담당자의 경험과 지식 등이 고려되어야 한다.

(2) 개발언어의 선정

웹 애플리케이션 개발 프로젝트에서 범용성, 이식성이 뛰어나고 대상 업무의 성격에 적합한 Java 및 JavaScript/HTML5를 Server와 Client 개발 언어로 선정한다.

(가) Server Side: 웹 애플리케이션 개발에 적합한 Java로 선정

(나) Client Side: 차세대 웹표준인 HTML5와 JavaScript를 선정

(3) 개발언어의 사용현황

위 항에서 본 바와 마찬가지로 2015년 11월까지, 전 세계 프로그램 언어시장은 Java와 JavaScript가 독보적인 우위를 차지하고 있다고 볼 수 있다.

2. 개발도구의 선정

(1) 통합 개발환경의 종류 및 특징

다양한 프로그래밍 언어를 지원하는 통합 개발 환경의 예로, 비주얼 스튜디오, 이클립스를 들 수 있다. 이클립스는 자바를 기본적으로 지원하지만, 파이썬, 펄, 루비, 포트란, C, C++, PHP, 코볼, JSP 등과 같은 언어들도 추가적으로 설치할 수 있다. 각 언어의 추가 설치본은 각자 고유의 디버거를 비롯한 다양한 도구들을 가지고 있다.

<표 1-2> 통합 개발환경 종류 및 주요 특징

통합 개발환경	개발사	지원OS	지원언어	라이선스
Eclipse	IBM 이클립스재단	MS-Windows Linux OSX Solaris AIX	Java C C++ PHP JSP 외 다수	Eclipse Public License
Lazarus	Lazarus Team	MS-Windows Linux OSX FreeBSD	Pascal	GPL, GNU LGPL, 기타
Anjuta	GNOME 프로젝트	Linux	C C++	GPL
Wide Studio	와이드 스튜디오 프로젝트	Linux	C C++	MIT
Code::Blocks	Code::Blocks Team	MS-Windows Linux OSX	C C++	GPL v3.0
Visual Studio	MicroSoft	MS-Windows	VisualBasic .Net VisualC++ VisualC# 등	상용
Delphi	엠바카데로 테크놀로지	MS-Windows	Pascal	상용
C++ Builder	엠바카데로 테크놀로지	MS-Windows	C C++	상용

출처: <https://ko.wikipedia.org>, 위키피디아(통합 개발환경의 종류)

(2) 통합 개발환경의 선정

Eclipse IDE를 개발도구로 선정: 풍부한 기능과 Plug-In을 보유하고 있으며, 다양한 적용사례가 존재한다.

② 프로그램의 배포 및 라이브러리 관리를 위한 빌드 도구를 선정한다.

1. 빌드도구 Maven과 Ant의 특징

<표 1-3> 빌드도구 Maven과 Ant 비교

Maven	Ant
기 구현된 Goal 수행	프로젝트 특화된 Target 수행
프로젝트 전체 정보를 정의	빌드 프로세스만 정의
빌드 생명주기, 표준화된 디렉토리 레이아웃	매우 복잡한 빌드 스크립트
재사용 가능한 플러그인, 저장소	스크립트가 재사용 가능하지 않음
매우 빠른 속도로 발전하고 있음	발전속도가 느려짐

2. 빌드도구의 선정

(가) 프로젝트 팀원의 친밀도와 숙련도에 따라 결정한다.

(나) 본 예제에서는 Ant를 선정한다.

③ 프로젝트 수행에 적합한 테스트 도구를 선정한다.

1. 테스트 도구의 종류

<표1-4> 테스트 도구 종류

테스트 활동	테스트 도구	내용
테스트 계획	요구사항 관리	고객 요구사항 정의 및 변경사항 관리
테스트 분석/설계	테스트 케이스 생성	테스트 기법에 따른 테스트 데이터 및 케이스 작성
	커버리지 분석	대상 시스템에 대한 테스트 완료 범위 척도
	테스트 자동화	기능 테스트 등 테스트 도구를 활용하여 자동화를 통한 테스트의 효율성을 높일 수 있음
테스트 수행	정적분석	코딩표준, 런타임 오류 등을 검증
	동적분석	대상 시스템 시뮬레이션을 통한 오류 검출
	성능 테스트	가상사용자를 인위적으로 생성하여 시스템 처리능력 측정
	모니터링	시스템 자원(CPU, Memory 등) 상태 확인 및 분석 지원 도구
테스트 통제	형상관리	테스트 수행에 필요한 다양한 도구 및 데이터 관리
	테스트 관리	전반적인 테스트 계획 및 활동에 대한 관리

출처: 공개SW포털 <http://www.oss.kr>(테스트 활동에 따른 도구 분류)

2. 테스트 도구의 선정

Java에서 가장 널리 사용되는 Unit Test Tool인 JUnit을 사용한다.

④ 개발 인원을 고려한 형상관리 도구를 선정한다.

1. 형상관리 도구의 종류

<표 1-5> 형상관리 도구 종류

Git	SVN	CVS
2005년	2000년	1990년
분산버전관리시스템 (distributed revision control) 리눅스 토발즈가 리눅스 커널 개발에 이 용하려고 개발	CVS를 대체하기 위해 개발, 현재는 아파치재단에서 개발	
로컬저장(분산저장)으 로 Offline 개발 가능, 속도빠름	CVS와 비교해서 장점 - 폴더 이동이 자유롭다. - Commit단위로 revision 관리 - Atomic Commit: 행동단위로 복구가능	
OpenCV MPC-HC	BitCoin	Boost Library FileZiller 등이 CVS 기반으로 관리되고 있음

2. 형상관리 도구의 선정

(1) 개발 시스템 환경등을 고려하여 선정한다.

(2) 본 예제에서는 Git를 선정한다.

수행 tip

- 각 종류별 개발 도구들에 대해 이해하고 장점과 단점들에 대해 파악하여 프로젝트 요구사항에 적합한 도구들을 선정하는 것이 중요하다.

1-2. 개발환경 구축

학습 목표

- 응용 소프트웨어 개발에 필요한 하드웨어 및 소프트웨어를 설치하고 설정하여 개발 환경을 구축할 수 있다.
- 사전에 수립된 형상관리 방침에 따라, 운영정책에 부합하는 형상관리 환경을 구축할 수 있다.

필요 지식 /

① 개발환경 구성

일반적으로 시스템 환경은 프로그램 개발을 위한 개발환경, 테스트를 위한 테스트환경, 실제 시스템이 운영되는 운영환경과 백업환경 등으로 분류할 수 있다. 이 중 개발환경 구축에 필요한 하드웨어와 소프트웨어 명세를 살펴보기로 한다.

1. 개발 하드웨어

개발 하드웨어 환경은 운영환경과 유사한 구조로 구성하는 것이 원칙이며, 개발용 하드웨어 환경을 구축하기 위해서는 다음과 같은 하드웨어 구성을 고려하여야 한다.

(1) 클라이언트(Client)

시스템에서 제공하는 서버 서비스를 활용하기 위해 거래를 발생시키는 하드웨어로서, 일반적으로 PC(웹화면) 또는 핸드폰(모바일 앱)이 클라이언트로 활용된다.

(2) 서버(Server)

용도에 따라 애플리케이션 서버, 데이터베이스서버, 파일서버 등으로 나눌 수 있다.

(가) 애플리케이션 서버

미들웨어인 WAS(Web Application Server)와 개발된 애플리케이션이 설치되는 하드웨어이다.

(나) 데이터베이스 서버

데이터베이스가 설치되는 하드웨어이다.

(다) 파일서버

개발 시 작성되는 산출물 및 소스코드 등 제반 파일을 저장하고, 공유하기 위한 하드웨어이다.

2. 개발 소프트웨어

개발 소프트웨어 환경은 운영환경과 동일한 구조로 구성하는 것이 원칙이며, 개발용 소프트웨어 환경을 구축하기 위해서는 다음과 같은 유형의 소프트웨어 구성을 고려하여야 한다.

(1) 시스템 소프트웨어

(가) 운영체제(OS: Operating System)

하드웨어 기동을 위한 운영체제로서, Windows/Linux/UNIX 등의 환경으로 구성되는데, 일반적으로 상용 소프트웨어 명세는 하드웨어를 제공하는 벤더에서 제공한다.

(예) Windows, Linux, UNIX(HPUX, Solaris, AIX 등

(나) JVM(Java Virtual Machine)

Java 관련 응용 프로그램을 기동하기 위한 인터프리터 환경으로, 적용버전을 개발 표준에서 명시하여 모든 개발자가 동일한 버전을 적용하는 것이 좋다.

(다) WAS(Web Application Server)

웹 애플리케이션을 수행하는 미들웨어로서, 웹서버와 JSP/Servlet 애플리케이션 수행을 위한 엔진으로 구성된다.

(예) Tomcat, JEUS, Weblogic, Websphere 등

(라) DBMS

데이터 저장과 관리를 위한 데이터베이스 소프트웨어이다.

(예) Oracle, DB2, Sybase, SQL Server, MySQL 등

(2) 개발용 소프트웨어

(가) 분석설계 도구

(나) 데이터 모델링 도구

(다) 에디터: Eclipse 등

(라) 빌드도구: Ant 등

(마) 테스트 도구: JUnit 등

(바) 형상관리 도구: Git 등

(사) 프레임워크

(아) UI 도구

(자) 레포트생성 도구

(차) 인터페이스 도구: EAI/ESB 등

(카) 배치작업 스케줄러

(3) 모니터링 도구

(가) APM 도구

(나) 정적분석도구

(다) 시스템 성능측정 도구

(라) 데이터베이스 성능측정 도구

(마) 시스템 관리 도구

(바) 네트워크 관리 도구

② 형상관리 구성

1. 정의

형상관리(SCM: Software Configuration Management)란, 소프트웨어의 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동으로, 다음과 같은 특성을 가지고 있다.

- (1) 소프트웨어 변경의 원인을 알아내고 제어하며 적절히 변경되고 있는지 확인하여 해당 담당자에게 통보하는 작업이다.
- (2) 형상관리는 소프트웨어 개발의 전 단계는 물론, 유지보수 단계에서도 수행되는 활동이다.
- (3) 형상관리는 소프트웨어 개발의 전체 비용을 줄이고, 개발과정의 여러 가지 문제점 발생요인이 최소화 되도록 보증하는 것을 목적으로 한다.

2. 형상관리 절차

형상관리는 품질 보증을 위한 중요한 요소로서 다음과 같은 절차를 통해 수행된다.

(1) 형상 식별

(가) 형상관리 대상을 식별하여 이름과 관리 번호를 부여하고, 계층(Tree) 구조로 구분하여 수정 및 추적이 용이하도록 하는 작업으로 베이스라인의 기준을 정하는 활동이다.

(나) 형상관리 대상이 되는 형상인 형상항목은 다음과 같다.

- 1) 소프트웨어 공학 기반 표준과 절차: 방법론, WBS, 개발표준
- 2) 소프트웨어 프로젝트 계획서
- 3) 소프트웨어 요구사항 명세서
- 4) 소프트웨어 아키텍처, 실행 가능한 프로토타입
- 5) 소프트웨어 화면, 프로그램 설계서

- 6) 데이터베이스 기술서: 스키마, 파일 구조, 초기 내용
- 7) 소스 코드 목록 및 소스 코드
- 8) 실행 프로그램
- 9) 테스트 계획, 절차, 결과
- 10) 시스템 사용 및 운영과 설치에 필요한 매뉴얼
- 11) 유지보수 문서: 변경 요청서, 변경 처리 보고서 등

(2) 변경 제어

식별된 형상항목의 변경 요구를 검토, 승인하여 적절히 통제함으로써 현재의 베이스라인에 잘 반영될 수 있도록 조정하는 작업으로, 적절한 형상통제가 이루어지기 위해서는 형상통제위원회 승인을 통한 통제가 이루어질 수 있어야 한다.

(3) 형상 상태 보고

베이스라인의 현재 상태 및 변경 항목들이 제대로 반영되는지 여부를 보고하는 절차로써 형상의 식별, 통제, 감사 작업의 결과를 기록 및 관리하고 보고서를 작성하는 작업이다.

(4) 형상 감사

베이스라인의 무결성을 평가하기 위해 확인, 검증 과정을 통해 공식적으로 승인하는 작업을 말한다.

수행 내용 / 개발환경 구축하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

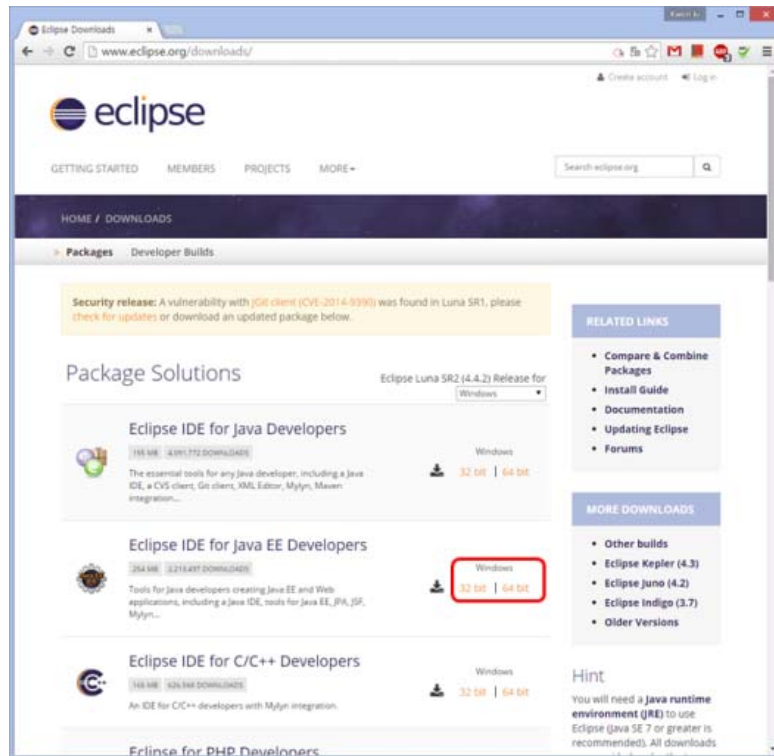
안전 · 유의사항

- 실습 후에는 실습 기기 및 컴퓨터 전원을 끈다.

수행 순서

① 통합 개발 환경 도구 설치를 수행한다.

1. 요구사항에 적합한 Eclipse를 다운받아 해당 파일을 확인한다.

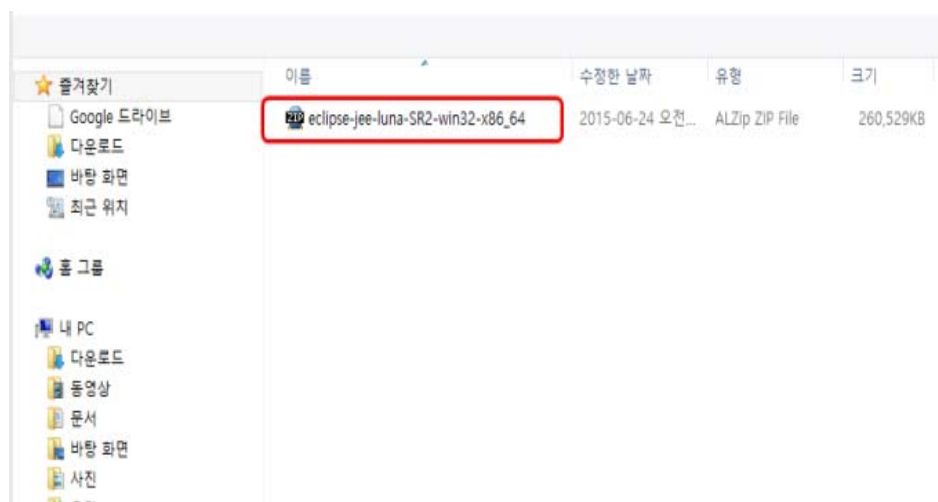


출처: <http://www.eclipse.org/downloads/>

[그림 1-1] Eclipse 다운받기

2. Eclipse의 설치를 진행한다.

(1) 압축 풀기



[그림 1-2] 압축 풀기

(2) Eclipse 실행

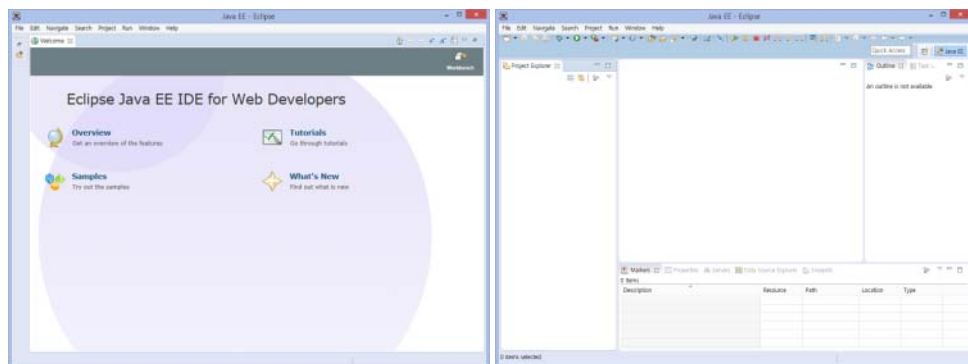
eclipse.exe 실행 파일을 선택하여 Eclipse를 실행한다.

이름	수정된 날짜	유형	크기
configuration	2015-06-24 오전...	파일 폴더	
dropins	2015-02-19 오전...	파일 폴더	
features	2015-06-24 오전...	파일 폴더	
p2	2015-06-24 오전...	파일 폴더	
plugins	2015-06-24 오전...	파일 폴더	
readme	2015-06-24 오전...	파일 폴더	
.eclipseproduct	2015-01-28 오전...	ECLIPSEPRODUCT...	1KB
artifacts	2015-02-19 오전...	XML 파일	249KB
eclipse	2015-02-19 오전...	응용 프로그램	314KB
eclipse	2015-02-19 오전...	구성 설정	1KB
eclipseec	2015-02-19 오전...	응용 프로그램	26KB
epl-v10	2015-01-28 오전...	Chrome HTML D...	13KB
notice	2015-01-28 오전...	Chrome HTML D...	9KB

[그림 1-3] Eclipse 실행

(3) Eclipse 실행 확인

Eclipse 실행 이후에는 workspace를 생성하고 eclipse의 실행 화면을 확인한다.



[그림 1-4] Eclipse 실행 확인

3. Eclipse를 실행하고 workspace를 설정한다.

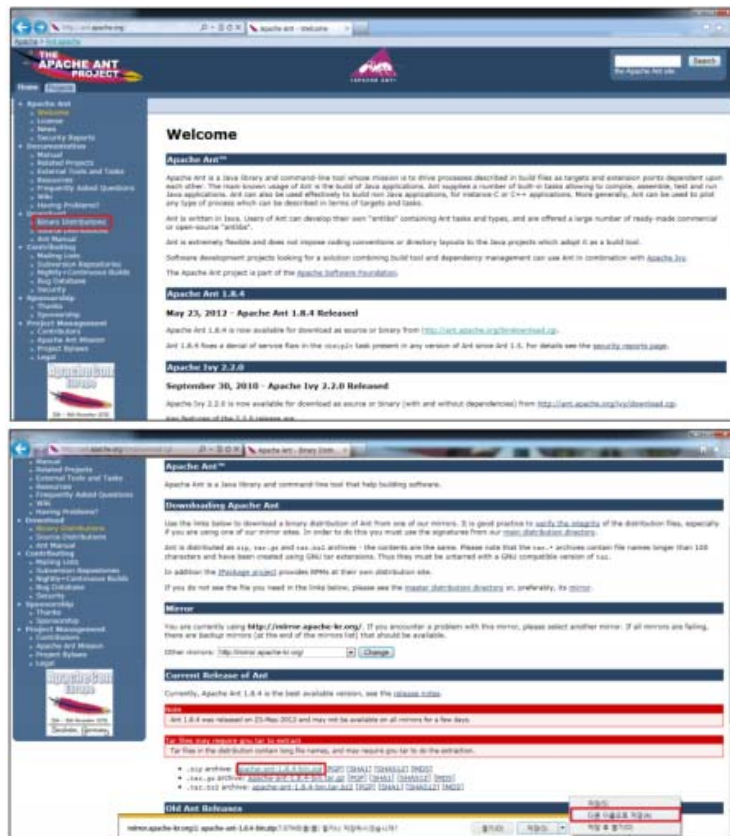
4. 새로운 프로젝트를 생성하고 확인한다.

② 빌드 도구 설치를 수행한다.

1. Ant를 다운받아 해당 파일을 확인한다.

(1) 대부분의 Java IDE는 Ant를 기본적으로 포함하고 있지만 Eclipse 외부나 업데이트 상황에 설치를 진행한다.

(2) 다운 경로: <http://ant.apache.org/>에 접속하여 Binary Distributions 선택

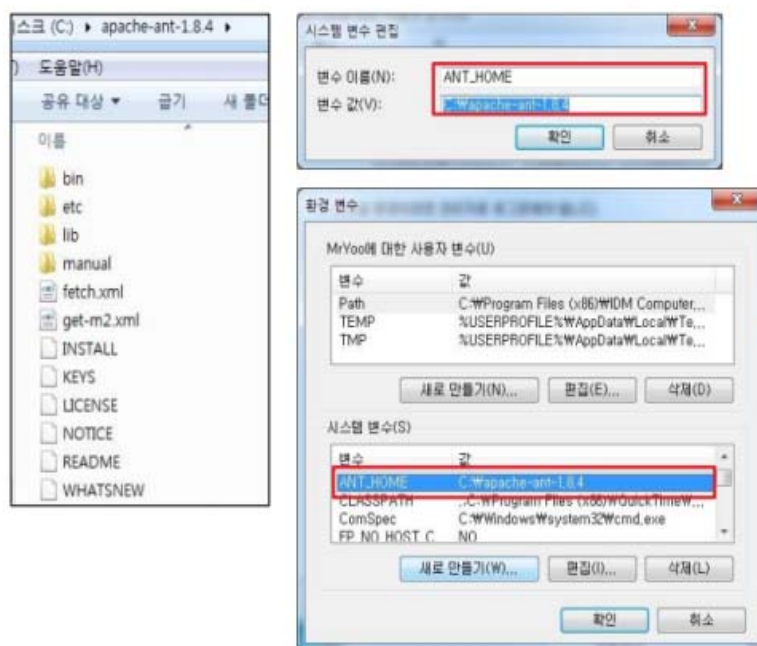


출처: <https://www.swbank.kr/helper/tool/toolMain.do>

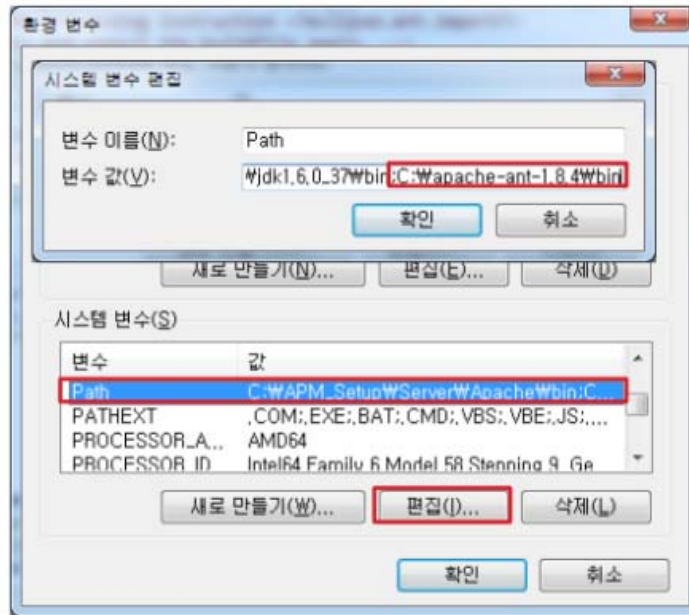
[그림 1-5] Ant 다운받기

2. Ant의 설치를 진행한다.

다운받기가 완료되면 압축을 해제하고 환경변수를 설정한다.



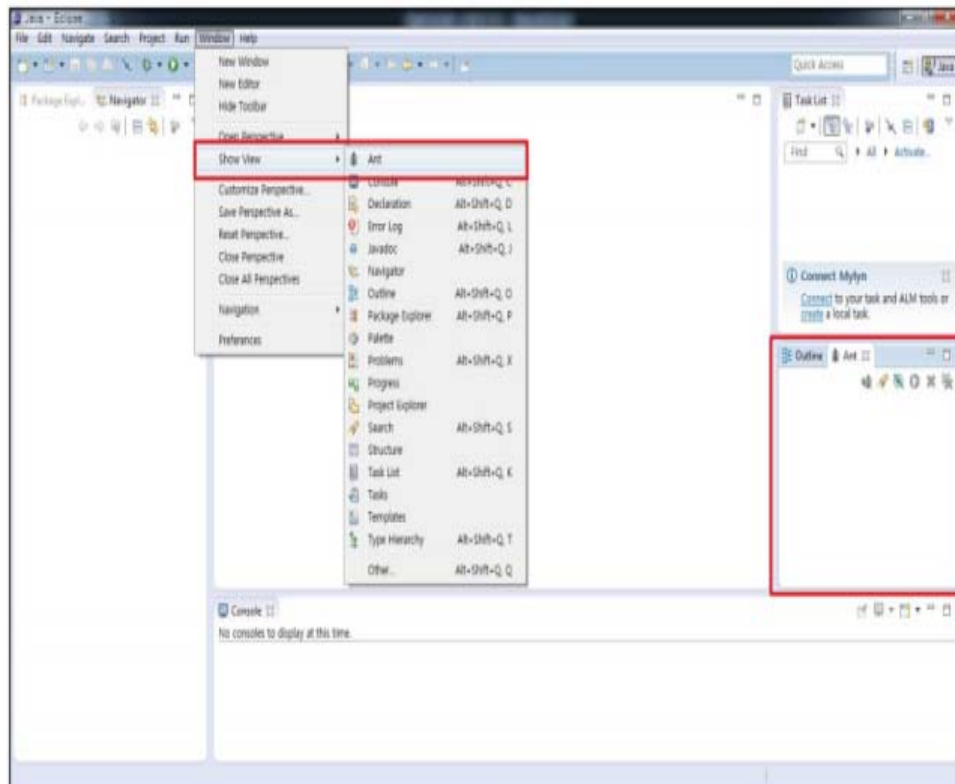
[그림 1-6] Ant 환경 설정-1



[그림 1-7] Ant 환경 설정-2

3. Ant의 설치 여부를 확인한다.

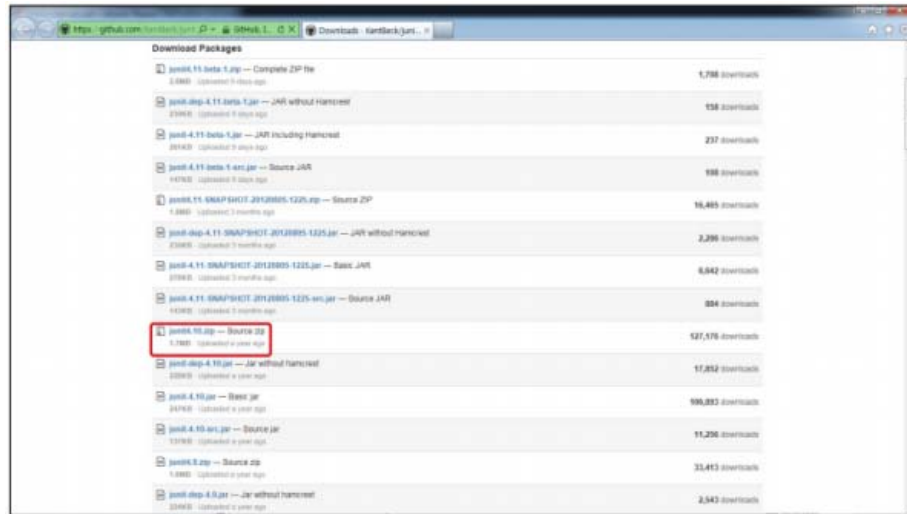
Eclipse를 실행하여 Windows -> Show View -> Ant를 선택하여 Ant 윈도우즈가 생성 되는 것을 확인한다.



[그림 1-8] Ant 설치 확인

③ 테스트 도구의 설치를 수행한다.

1. JUnit을 다운받아 해당 파일을 확인한다.

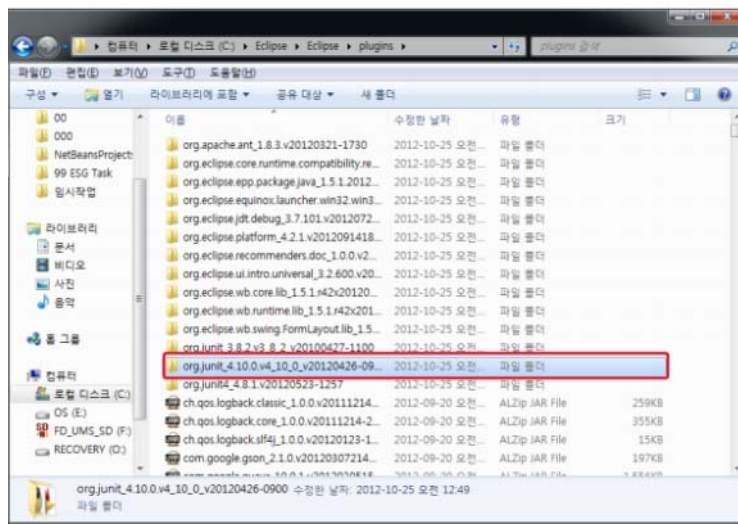


출처: <https://www.swbank.kr/helper/tool/toolMain.do>

[그림 1-9] JUnit 다운받기

2. JUnit의 설치를 진행한다.

다운받은 JUnit의 압축을 해제하고 Eclipse가 설치된 폴더 내의 plugins에 복사하여 설치를 완료한다.

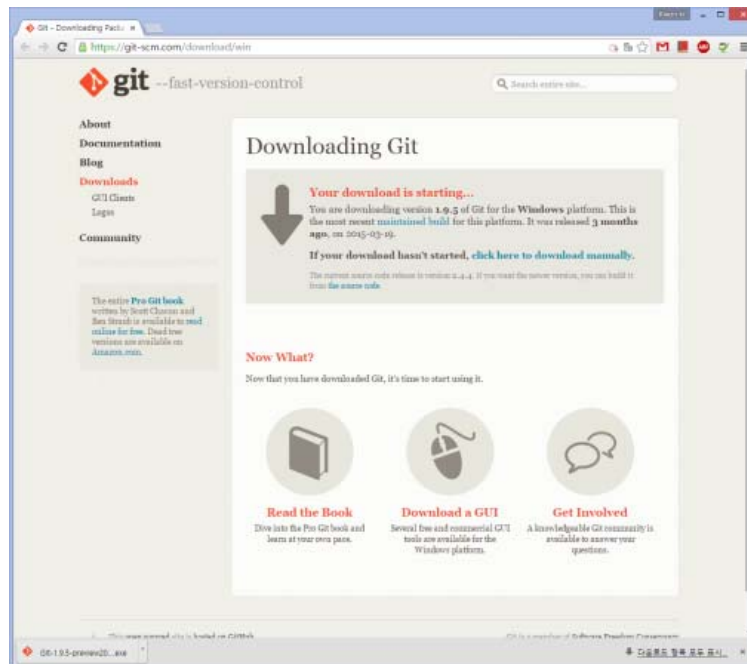


[그림 1-10] JUnit 설치

3. JUnit의 설치 여부를 확인한다.

4 형상관리 도구의 설치를 수행한다.

1. Git를 다운받아 해당 파일을 확인한다.

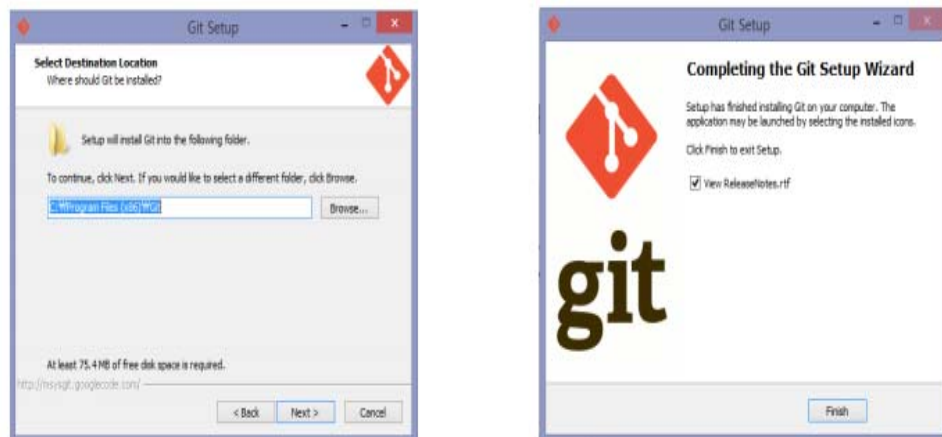


출처: <http://git-scm.com/download/win>

[그림 1-11] Git 다운받기

2. Git의 설치를 진행한다.

다운받은 파일을 실행하고 안내에 따라 설치한다.



[그림 1-12] Git 설치

3. Git의 설치 후 관련 환경을 설정한다.

- (1) Git 설치를 완료한 이후에는 사용자 정보를 설정한다.
- (2) 사용자 정보는 사용자의 이름과 e-mail을 최우선으로 설정한다.
- (3) 사용자 이름 설정: `git config --global user.name "사용자 이름"`

(4) 사용자 e-mail 설정: `git config --global user.email` 이메일 주소

```
root@test-server:/# git config --global user.name "John"
root@test-server:/# git config --global user.email John@gmail.com
```

[그림 1-13] 사용자 이름 및 이메일 설정

(5) 설정 확인: `git config --list`

```
root@test-server:/# git config --list
user.name=John
user.email=John@gmail.com
```

[그림 1-14] 사용자 이름 및 이메일 설정확인

수행 tip

- 개발에 필요한 하드웨어와 소프트웨어를 파악하여 해당 명세대로 개발환경을 구축하고, 형상관리 지침에 따라 형상관리 환경을 구축할 수 있도록 한다.

학습 1 교수 · 학습 방법

교수 방법

- 응용 소프트웨어 개발에 필요한 하드웨어 및 소프트웨어 제품유형에 대해 설명하고, 각 제품의 용도에 대해 제시한 후 수업을 진행한다.
- 여러 개발 도구들에 대한 이해 및 이전 사용 경험 여부에 대해 파악한 후 수업을 진행한다.
- 여러 개발 도구의 매뉴얼 및 홈페이지 등을 방문하여 다양한 자료를 제시한 후 설명한다.
- 다양한 개발 도구의 설치 과정에 대해 단계적 실습이 이루어질 수 있도록 지도한다.
- 형상관리 방침에 포함되는 내용에 대해 설명하고, 형상관리 환경 구축방안을 정리하여 설명한다.

학습 방법

- 다양한 개발 도구들의 용도와 목적에 대해 이해한다.
- 개발 도구들에 대한 자료들을 찾아 활용방안에 참고할 수 있도록 한다.
- 다양한 개발 도구들의 설치 및 실행까지의 과정에 대해 실습한다.
- 직접 개발 도구들을 개발환경에 구축하여 제공 기능에 대해 실습하여 학습한다.
- 형상관리 환경구축을 위한 실습을 해 봄으로써, 구축 필요성에 대해 이해한다.

학습 1 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
개발환경 사양 이해	- 응용 소프트웨어 개발에 필요한 하드웨어 및 소프트웨어의 필요 사항을 검토하고 이에 따라, 개발환경에 필요한 준비를 수행할 수 있다.			
개발환경 구축	- 응용 소프트웨어 개발에 필요한 하드웨어 및 소프트웨어를 설치하고 설정하여 개발환경을 구축할 수 있다.			
	- 사전에 수립된 형상관리 방침에 따라, 운영정책에 부합하는 형상관리 환경을 구축할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
개발환경 사양 이해	- 개발 환경 요구사항에 대한 이해			
	- 개발 도구들의 역할에 대한 이해			
개발환경 구축	- 개발 도구들의 설치 및 활용에 대한 이해			
	- 형상관리 지침의 이해와 환경구축 능력			

- 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
개발환경 사양 이해	- 개발 환경 요구사항에 대한 이해			
	- 개발 도구들의 역할에 대한 이해			
개발환경 구축	- 개발 도구들의 설치 및 활용에 대한 이해			
	- 형상관리 지침의 이해와 환경구축 능력			

피드백

1. 평가자 체크리스트

- 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.
- 개발환경에 필요한 하드웨어와 소프트웨어의 누락여부를 점검하여 정리하여 돌려준다.
- 개발환경에 필요한 개발도구를 선정하여 구축하였는지를 점검하여 정리하여 돌려준다.
- 형상관리 환경구축 후 정상적인 형상관리가 되는지를 확인하여 비정상 동작시 그 원인을 파악하여 정리한 후 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 개발환경 구축 절차에 대한 정확성을 파악하여 미비사항에 대한 내용을 전달한다.
- 형상관리 환경구축 절차에 대한 정확성을 파악하여 미비사항에 대한 내용을 전달한다.

학습 1	개발환경 구축하기(LM2001020203_14v2.1)
학습 2	공통 모듈 구현하기 (LM2001020203_14v2.2)
학습 3	서버 프로그램 구현하기(LM2001020203_14v2.3)
학습 4	배치 프로그램 구현하기(LM2001020203_14v2.4)
학습 5	개발자 단위 테스트하기(LM2001020203_14v2.5)
학습 6	애플리케이션 성능 개선하기(LM2001020203_14v2.6)

2-1. 공통 모듈 작성

학습 목표

- 공통 모듈의 상세 설계를 기반으로 프로그래밍 언어와 도구를 활용하여 업무 프로세스 및 서비스의 구현에 필요한 공통 모듈을 작성할 수 있다.
- 소프트웨어 측정지표 중 모듈 간의 결합도는 줄이고 개별 모듈들의 내부 응집도를 높인 공통 모듈을 구현할 수 있다.

필요 지식 /

① 개발환경에 적용된 아키텍처 패턴에 대한 이해

1. 중규모(Mid-Range) 급 이상의 시스템에서 애플리케이션은 효율적인 개발 및 유지보수를 위해 계층화(Layered Architecture) 형태로 구성한다.

2. 계층화 아키텍처 패턴의 장단점

<표 2-1> 계층화 아키텍처 패턴의 장단점

장점	단점
1. 계층별 컴포넌트의 종속성이 최소화되어 부분적인 변경 시에 영향도가 적다.	1. 동작이 변경될 경우 단계별로 재작업이 필요하다.
2. 표준을 지원한다.	2. 효율이 낮다.
3. 컴포넌트의 재사용성이 뛰어나다.	3. 불필요한 작업이 수행될 수 있다.
4. 교환가능성이 확보된다.	4. 계층의 적절한 개수나 규모를 결정하는 것이 어렵다.

② 공통 모듈에 대한 이해

정보시스템 구축 시 자주 사용하는 기능들로서 재사용이 가능하게 패키지로 제공하는 독립된 모듈을 의미한다.

<표 2-2> 전자정부 표준프레임워크 구성

구분	분류	기능
공통기술 (129종)	사용자디렉토리/ 통합인증	·사용자통합인증(2종), 로그인(1종)*
	보안	·실명확인(2종), 역할/권한관리(5종), 암호화/복호화(1종)
	통계/리포팅	·통계(5종)
	협업	·게시판(10종)*, 동호회(4종)*, 일정관리(9종)*, 전자우편(1종), 주소록/명함록(2종)*, 문자메시지(1종), 전자결재(1종)
	사용자지원	·사용자관리(3종), 약관관리(3종)*, 개인화(1종)*, 온라인참여(9종)*, 온라인헬프(6종)*, 정보제공/알림(29종)*
	시스템관리	·공통코드관리(7종), 메뉴관리(3종), 프로그램관리(1종), 로그관리(6종), 배치관리(3종), 시스템관리(3종), 장애관리(2종)
	시스템/서비스 연계	·시스템연계(4종)*
	디지털자산관리	·지식관리(5종)
모바일 공통기술 (10종)	협업	·모바일실시간 공지서비스(1종), 오프라인웹서비스(2종), 모바일위치정보연계서비스(1종), 모바일기기식별(1종)
	시스템/서비스 연계	·시스템연계(2종)
	디지털자산관리	·모바일 차트/그래프(1종), 모바일 사진앨범(1종), 모바일 멀티미디어제어(1종)
요소기술 (91종)		·달력(3종), 시스템(51종), 웹 에디터(1종), 인쇄/출력(3종), 쿠키/세션(2종), 포맷/계산/변환(24종), 메시지처리(4종), 인터페이스/화면(2종) ·파일업로드/다운로드(1종)

출처: 한국정보화진흥원 표준프레임워크센터(2015.06), 정보시스템 구축 발주자를 위한 표준프레임워크 적용가이드 Ver. 3.5

③ 소프트웨어 모듈 결합도

- 어떤 모듈이 다른 모듈에 의존하는 정도를 나타내는 것이다.
- 결합도는 보통 응집도(Cohesion)과 대비된다. 낮은 결합도는 종종 높은 응집도와 관련이 있으며, 그 역도 마찬가지이다.
- 결합도와 응집도라는 소프트웨어 측정 지표(Software Metric)는, 구조적 설계의 개발자인 래리 콘스탄틴(Larry Constantine)이 만들었다.
- 낮은 결합도는 종종 구조화가 잘된 컴퓨터 시스템의 지표이며, 좋은 설계이며, 높은 응

집도를 검비하면, 높은 가독성과 유지보수성이라는 일반적인 목표를 이루게 된다.

5. 결합도 순서

자료결합도 -> 스탬프 결합도 -> 제어 결합도 -> 외부 결합도 -> 공통 결합도 -> 내용 결합도

<표 2-3> 결합도 유형

결합 유형	설명
내용 결합도 (Content Coupling 또는 Pathological Coupling)	하나의 모듈이 다른 모듈의 내부 동작을 수정하거나 내부 동작에 의존하는 상태이다.(예: 다른 모듈의 로컬 데이터에 접근하는 경우) 따라서 한 모듈이 데이터를 생성하는 방법(위치, 타입, 타이밍)을 변경하면, 다른 모듈의 변경이 필요하다.
공통 결합도 (Common Coupling 또는 Global Coupling)	두 개의 모듈이 같은 글로벌 데이터를 공유하는 상태이다. (예) 전역 변수 공유 자원(변수)을 변경하면, 그 자원(변수)을 사용하는 모든 모듈의 변경이 필요하다.
외부 결합도 (External Coupling)	두 개의 모듈이 외부에서 도입된 데이터 포맷, 통신 프로토콜, 또는 디바이스 인터페이스를 공유할 때 발생한다. 이는 기본적으로 외부 툴이나 디바이스와의 통신과 관련이 있다.
제어 결합도 (Control Coupling)	하나의 모듈이 다른 모듈로 무엇을 해야 하는지에 대한 정보를 넘겨줌으로써 다른 모듈의 흐름을 제어하는 경우이다.
스탬프 결합도 (Stamp Coupling)	모듈들이 데이터 구조를 공유하고, 서로 다른 일부만을 사용하는 경우이다. 접근할 필요가 없는 필드만 수정되는 경우에도(데이터의 배치가 변경되므로), 레코드(필드)를 읽는 방법을 변경해야 한다.
자료 결합도 (Data Coupling)	모듈들이 파라미터 등을 통해 데이터를 공유하는 경우이다. 각 데이터가 기본적인 것(Elementary Peice)이고, 그 데이터들이 공유되는 유일한 데이터여야 한다. (예, 제곱근을 계산하는 함수로 하나의 정수를 전달하는 경우)
메시지 결합도 (Message Coupling)	가장 낮은 결합도이다. 이는 State Decentralization을 통해 이를 수 있고, 컴포넌트 간의 통신은 파라미터나 메시지 패싱을 통해 이루어져 한다.
결합도 없음 (No Coupling)	모듈이 어떠한 다른 모듈과도 통신하지 않는 경우이다.

④ 소프트웨어 모듈 응집도

1. 한 모듈 안의 기능들의 연관성 정도를 나타내는 것이다.

2. 응집도 순서

기능적 응집도 -> 순차적 응집도 -> 통신적 응집도 -> 절차적 응집도 -> 시간적 응집도
-> 논리적 응집도 -> 우연적 응집도

<표 2-4> 응집도 유형

응집 유형	설명
우연적 응집도 (coincidental cohesion)	관련 없는 요소들로 구성된 모듈로 단순히 일정한 크기로 분할된 경우이다. 모듈화 장점이 없다. 유지보수 작업이 어렵다.
논리적 응집도 (logical cohesion)	유사한 기능들이 하나의 모듈 안에 구성된 경우이다. 사용하는 매개변수에 따라 처리하는 내용이나 경로가 달라진다. 예) 오류처리: 자판기의 잔액부족, 음료수부족 출력처리: 직원 인사정보 출력, 회계정보 출력
시간적 응집도 (temporal cohesion)	특정한 시점에서 작업을 수행하는 경우, 시간의 흐름에 따라 작업 순서가 정렬되는 응집관계이다. 예) 초기치 설정, 종료처리 등
절차적 응집도 (procedural cohesion)	입출력을 공유하지 않으나 순서에 따라 수행될 필요가 있는 경우이다. 예) Restart루틴: 총계 출력하고 화면을 지우고 메뉴를 표시
통신적(교환적) 응집도 (communicational cohesion)	여러가지 기능을 수행하며 모듈 내 구성 요소들이 같은 입력자료를 이용하거나 동일 출력 데이터를 만들어 내는 경우이다. 예) 같은 입력자료를 사용하여 A를 계산한 후 B를 계산
순차적 응집도 (sequential cohesion)	하나의 기능에서 생성된 출력자료가 다음 기능의 입력자료로 사용되는 경우이다. 예) 행렬 입력 후 그 행렬의 역행렬을 구해서 이를 출력
기능적 응집도 (functional cohesion)	하나의 기능을 수행하는 데 필요한 요소들만 포함한 경우이다. 구조도 최하위 모듈에서 많이 발견된다.

수행 내용 / 공통 모듈 작성하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

- 1 공통모듈로 식별된 오브젝트를 선언(declaration)한다.

<회원가입 Servlet 예제>

```
package join;

import java.io.IOException;
import java.io.PrintWriter;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Joincontroller extends HttpServlet {

    @Override

    protected void doPost(HttpServletRequest req, HttpServletResponse
    resp) throws ServletException, IOException {

        ...

    }

}
```

- ② 오브젝트의 속성(property)을 정의한다.

```
JoinDTO dto = new JoinDTO();  
  
JoinDAO dao = new JoinDAO();  
  
PrintWriter out = resp.getWriter();  
  
int result;
```

- ③ Method 프로토타입을 정의한다.

```
public JoinDAO {  
  
    ...  
  
}  
  
public int insertjoin(JoinDTO dto) {  
  
    ...  
  
}
```

- ④ I/O 오브젝트(DTO/VO)를 정의한다.

```
public class JoinDTO {  
  
    String id;  
  
    String pw;  
  
    String name;  
  
    String nick;  
  
    String addr;  
  
    String email;  
  
    Date birth;  
  
    public JoinDTO() {  
  
        // TODO Auto-generated constructor stub  
  
    }  
  
    ...  
  
}
```

⑤ Input Validation Check 로직을 구현한다.

```
if (!this.isType(year, "int")) {  
    error = true;  
    System.err.println("Error: must be a whole number.");  
} else {  
    error = false;  
}
```

⑥ Main Logic을 구현한다.

```
public class Joincontroller extends HttpServlet{  
    @Override  
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)  
        throws ServletException, IOException {  
        req.setCharacterEncoding("UTF-8");  
        //req.setCharacterEncoding("EUC-KR");  
        //한글처리  
        JoinDTO dto = new JoinDTO();  
        JoinDAO dao = new JoinDAO();  
        PrintWriter out = resp.getWriter();  
        int result;  
  
        dto.setId(req.getParameter("id"));  
        dto.setPw(req.getParameter("pw"));  
        dto.setName(req.getParameter("name"));  
        dto.setNick(req.getParameter("nick"));  
        dto.setAddr(req.getParameter("addr"));  
        dto.setEmail(req.getParameter("email"));  
  
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy/mm/dd");  
        Date date = null;
```

```

        try {
            date = sdf.parse(req.getParameter("year")+"/
            "+req.getParameter("month")+"/"+req.getParameter("day"));
            //입력받은 년,월,일을 date 포맷으로 변경
        } catch (ParseException e) {
            e.printStackTrace();
        }
        dto.setBirth(date);
        ...
    }
}

```

7 Output Validation Check 로직을 구현한다.

```

public class Joincontroller extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse
    resp)
        throws ServletException, IOException {
        ...
        result = dao.insertjoin(dto);
        if(result==0){
            System.out.println("Failed");
        }else{
            System.out.println("Sucsses");
        }
    }
}

```

수행 tip

- 공통 모듈의 특성과 명세를 파악하여 개발 언어와 도구를 활용하여 해당 모듈이 정상적으로 작동할 수 있도록 구현할 수 있어야 한다.

2-2. 공통 모듈 테스트

학습 목표

- 개발된 공통 모듈의 내부 기능과 제공하는 인터페이스에 대해 테스트할 수 있는 테스트 케이스를 작성하고, 단위 테스트를 수행하기 위한 테스트 조건을 명세화할 수 있다.

필요 지식 /

① 단위 테스트 도구 적용방법

소프트웨어 개발에서 단위 테스트인 Unit Testing은 구현코드의 개별 단위의 적합성 혹은 정확성을 확인하기 위한 방법이다. 이 단위의 정의는 테스트 시나리오에 따라 다를 수 있다.

예를 들어서 C와 같은 절차적 프로그래밍 언어에서는 하나의 단위가 일반적으로 하나의 프로시저 또는 함수이다. 하지만 객체지향 언어에서는 하나의 메서드가 될 수 있다. 단위 테스트에서 하나의 테스트 단위는 테스트 가능한 가장 작은 부분으로 생각하면 무난하다.

1. 단위 테스트는 버그를 찾기 위한 것이 아니다.

(1) 단위 테스트의 의도를 정확히 이해하는 것은 중요하다. 단위 테스트는 단순히 버그를 찾기 위한 효과적인 방법이 아니다. 정의에 따르면 단위 테스트는 시스템의 각각의 단위들을 개별적으로 조사하는 것이다. 시스템이 구현되어 실제 환경에서 동작할 때 모든 단위들은 완벽하게 하나의 유기체로 동작해야한다. 하지만 시스템은 각 독립적으로 테스트되는 단위의 단순한 결합 그 이상으로 복잡하고 또한 에러가 발생하기 쉽다. 컴포넌트 X, Y가 독립적으로 잘 작동한다는 것이 이 컴포넌트들이 서로 호환된 다든가 혹은 정확하게 조합되어졌다는 것은 아니다.

(2) 따라서 단순히 버그를 찾기 위한 것이라면 일반적으로 검증자가 일일이 테스트하듯이 전체 시스템을 실제 통합환경에서 실행하는 것이 훨씬 효과적인 테스트가 될 수 있다. 그리고 이러한 테스트를 자동화한 것을 통합 테스트라고 하는데, 이것은 일반적으로 단위 테스트와는 다른 기술들을 사용한다.

(3) “단위 테스트는 TDD(Test Driven Development)에서 그러한 것처럼 반드시 시스템 디자인 단계의 일부분으로 보아야 한다.” 이렇게 함으로써 시스템 디자이너는 시스템의 가장 작은 단위 모듈을 인식할 수 있고 또한 개별적으로 테스트할 수 있다.

2. 하나의 테스트 케이스는 단위 기능 중 하나의 시나리오만 테스트하라.

(1) 단위 테스트 작성 시 가장 중요하게 인식할 점은 테스트 단위가 복수의 테스트 시나

리오들을 가질 수 있다는 것이다. 그리고 모든 테스트 시나리오들은 독립적인 테스트 코드로 작성되어야 한다. 예를 들어 두 매개변수를 가지고 처리한 후 값을 돌려주는 함수의 테스트 케이스를 작성한다고 하면, 다음과 같은 테스트 시나리오가 가능할 것이다.

(가) 첫 번째 파라미터가 널 값일 경우 예외 객체를 반환해야 한다.

(나) 두 번째 파라미터가 널 값일 경우 예외 객체를 반환해야 한다.

(다) 두 개의 파라미터 모두가 널 값일 경우 예외 객체를 반환해야 한다.

(라) 파라미터가 정상 범위 안일 경우 작업 실행 후 결과 값을 반환해야 한다.

(2) 이러한 세분화된 테스트 케이스들은 코드를 수정하거나 리팩토링 시 효과적이다. 왜냐하면 단위 테스트만 수행하면 코드의 수정이 코드의 의도된 기능을 망가뜨렸는지 확인할 수 있기 때문이다. 또한 기능을 수정한다면 최소한의 테스트 코드만 수정하면 되기 때문이다.

3. 불필요한 검증 구문을 작성하지 마라.

(1) 단위 테스트는 시스템의 특정 단위가 어떻게 동작하는지에 대한 디자인 스펙이지 단순히 단위 내의 코드가 행하는 모든 것을 관찰하는 것이 아니다.

(2) 단위 내의 모든 것에 대해 검증구문을 작성하지 마라. 대신 테스트하려는 하나의 시나리오에 집중한다. 테스트 코드를 이렇게 작성하지 않으면 하나의 이유로 여러 테스트 케이스가 실패 할 수 있다. 결국 이것은 프로젝트에 아무런 도움이 되지 않는다 (왜냐하면 테스트 코드를 보고 문제점을 찾을 수 없기 때문이다.).

4. 각 테스트는 독립적이어야 한다.

(1) 다른 테스트에 의존적인 꼬리에 꼬리를 무는 단위 테스트케이스를 작성하지 않는다. 이러한 테스트들은 테스트의 근본적인 실패 원인을 테스트 결과를 통해 알 수 없다. 결국 별도의 디버깅 작업을 수행해야 한다. 또한 이러한 상호 의존적인 테스트 코드는 유지보수도 번거롭다. 왜냐하면 하나의 테스트 코드를 수정할 경우 의존성을 가지고 있는 다른 코드로 수정해야 할 경우가 생기기 때문이다.

(2) 테스트의 선결 조건을 설정하기 위해서는 @Before/@After와 같은 테스트 프레임워크가 제공하는 어노테이션을 사용하라. 만약 서로 다른 테스트 구문을 위해 @Before/@After 어노테이션 구문 안에서 여러 가지 다른 세팅을 해야 한다면 별도의 새로운 테스트 클래스를 생성하는 것을 고려한다.

5. 모든 외부 서비스와 상태들에 테스트 더블을 사용하라.

(1) 그렇게 하지 않으면 공통된 외부 조건을 사용하는 테스트 구문들의 결과가 서로에게 영향을 미친다. 결국 테스트 구문 실행 순서에 따라 테스트 결과가 달라지거나 네트

워크 망이나 데이터베이스의 조건에 따라 결과가 달라진다.

- (2) 게다가 외부 서비스의 버그들로 인해서 테스트 결과가 실패로 끝날 수 있다(테스트 구문이 정적 변수들을 변화시키게 하지 않는다. 어쩔 수 없이 변화시켜야 한다면 적어도 테스트 시작 바로 전에 이 변수들을 초기화 시켜야 한다.).

6. 시스템 설정파일에 관한 단위 테스트를 작성하지 마라.

- (1) 정의에 따르면 시스템 설정은 단위 테스트의 범위가 아니다(그래서 그러한 별도의 설정파일로 분리된다). 시스템 설정값에 대한 단위 테스트를 작성하고 싶다면 설정값을 읽는 모듈에 대한 하나 혹은 두 개의 경우만 테스트한다.
- (2) 시스템 설정에 대한 모든 경우의 수를 테스트를 하는 것은 결국 하나밖에 증명하지 못한다. “난 복사 후 붙여넣기 할 줄 알아요”

7. 단위 테스트 케이스의 이름은 명확하고 일관되게 테스트의 의미를 반영해야 한다.

- (1) 이것은 언제나 명심하고 실천해야 하는 중요한 점이다. 테스트 케이스의 이름은 항상 테스트의 의도가 무엇인지 반영해야 한다. 단순히 테스트하려는 하는 단위의 클래스와 메서드의 조합을 테스트 케이스의 이름으로 사용하는 것은 좋은 생각이 아니다. 클래스나 메서드의 이름을 변경해야 하는 경우가 생길 때 테스트 케이스들의 이름도 매번 수정해야 한다.
- (2) 그러나 단위 테스트 케이스의 이름이 단위의 기능을 반영하는 논리적인 이름이라면 단위 기능이 바뀌지 않는 경우에는 테스트 케이스 이름은 언제나 동일하다.
- (3) 아래는 테스트 케이스 이름의 좋은 예들이다.

(가) TestCreateEmployee_NullId_ShouldThrowException

(나) TestCreateEmployee_NegativeId_ShouldThrowException

(다) TestCreateEmployee_DuplicateId_ShouldThrowException

(라) TestCreateEmployee_ValidId_ShouldPass

8. 외부 시스템이나 서비스에 대한 의존성이 가장 낮은 메서드들에 대해 테스트를 먼저 작성하라. 그리고 확장해 가라.

- (1) 예를 들어, Employee 모듈을 테스트한다고 하자. 가장 먼저 Employee 모듈을 생성하는 코드부터 테스트를 작성한다. 왜냐하면 이 시나리오가 가장 낮은 외부 의존성을 가지기 때문이다. 이 시나리오가 성공한다면, 데이터베이스에 접근하는 테스트 코드를 추가한다.
- (2) 데이터베이스에 Employee 정보를 가질려면 먼저 Employee 모듈을 생성하는 테스트 시나리오를 통과해야 한다. 만약 모듈을 생성하는 코드에 버그가 있다면 훨씬 빨리

발견할 수 있다.

9. 예상된 예외 사항을 테스트하는 단위 테스트 코드를 작성하라.

- (1) 예상된 예외 사항을 검증하기 위한 테스트 케이스를 작성해야 하는 경우도 있다. 이 때에는 아래와 같이 try/catch 구문으로 결과를 검증하려고 하지 않는다.

```
try{  
  
    methodOccursDomainSpecificException()  
  
    assertFail( "Exception should be occurred" );  
  
}  
  
catch(DomainSpecificException e){  
  
}
```

- (2) 대신 아래와 같이 JUNIT이 제공하는 아래의 방법을 사용하라.

```
1.  
  
@Test(expected=SomeDomainSpecificException.SubException.class)
```

10. 가장 적합한 검증 구문을 사용하라.

각 테스트 케이스에 사용할 수 있는 많은 검증 구문이 있을 수 있다. 하지만 각 경우마다 이유와 의도에 맞게 가장 적합한 것을 사용한다.

11. 검증 구문 파라미터들은 적합한 순서대로 배치하라.

검증 구문은 대개 두 개의 파라미터를 취한다. 첫 번째 것은 테스트 결과가 패스할 때 기대하는 정상 값이고, 두 번째 것은 테스트 결과 값이다. 이 두 값들을 순서대로 작성하라. 이렇게 하면 테스트 실패 시 에러 메시지를 통해 무엇이 잘못되었는지 쉽게 확인할 수 있다.

12. 테스트를 위한 코드는 제품 코드에서 분리되어야 한다.

- (1) 빌드 스크립트에서 테스트를 위한 코드는 실제 제품 코드와 같이 전달되지 않게 한다.
- (2) 테스트 코드 내에서 아무것도 출력하지 않는다(Do not print anything out in unit tests).
- (3) 만약 테스트 케이스가 가이드라인들에 따라 제대로 작성되어졌다면 별도의 출력문이 필요하지 않다. 만약 출력문에 대한 필요성을 느낀다면 테스트 코드를 재검증 하라. 무엇인가 테스트 코드 중 잘못된 점이 있다.

13. 정적 변수를 테스트 클래스에 사용하지 마라. 만약 사용했다면 각 테스트 케이스 실행

시마다 재초기화해라.

이미 각 테스트들 간의 독립성에 대해 중요성을 언급했다. 따라서 절대 Static 변수들을 사용하지 않는다. 만약 어쩔수 없이 사용해야 하는 경우가 발생한다면 각 테스트 케이스의 시작 전에 반드시 재초기화시킨다.

14. 예외 발생 시 단순히 테스트를 실패하기 위한 Catch 구문을 작성하지 마라.

테스트 코드 내에 어떤 메서드가 예외를 발생시킬 수 있다면 단순히 테스트를 실패하기 위해 Catch 구문을 사용하지 않는다. 대신 Throws 구문을 테스트 코드 선언 시에 사용한다. 되도록이면 별도의 특정 예외 객체보다는 일반적인 Exception 예외 객체를 사용한다. 이 가이드라인은 테스트 커버리지 증가에도 도움이 된다.

15. 간접적인 테스트들에 의존하지 마라.

하나의 테스트가 본래 의도된 시나리오 외 또 다른 시나리오도 테스트 한다고 가정하지 않는다. 이것은 혼란만 가져온다. 대신 또 다른 시나리오에 대한 추가적인 테스트 케이스를 작성한다.

16. 단위 테스트를 자동으로 실행하게 빌드 스크립트를 작성해라.

테스트 케이스들이 빌드 스크립트를 통해 자동적으로 실행되게 한다. 이것은 테스트 실행 환경과 애플리케이션의 신뢰성을 높여준다.

17. 단위 테스트들의 실행을 생략하지 마라.

만약 단위 테스트의 코드가 적절하지 않다면 코드를 삭제하고, @Ignore 어노테이션을 사용하지 않는다. 소스코드의 부적절한 테스트 케이스를 포함하는 것은 아무런 도움이 되지 않는다.

18. 테스트 결과를 XML 형태로 출력하라.

즐거운 것은 좋은 것이다. 이 가이드라인은 테스트 케이스 작성을 즐겁게 한다. 예를 들어 JUNIT을 빌드 스크립트와 통합시켜서 XML- 로 테스트 결과를 출력 후 이것을 - 추가적인 포매팅 작업을 통해 테스트 결과를 보기 좋게 바꿀 수 있다.

② 디버깅 도구의 사용법

1. 디버깅(Debugging) 또는 디버그(Debug)의 개념

(1) 디버그(Debug), 디버깅 (Debugging) 혹은 수정은 컴퓨터 프로그램의 정확성이나 논리적인 오류(버그)를 찾아내는 테스트 과정을 뜻한다. 디버깅(Debugging), 수정이라고도 한다.

(2) 일반적으로 디버깅을 하는 방법으로 테스트 상의 체크, 기계를 사용하는 테스트, 실

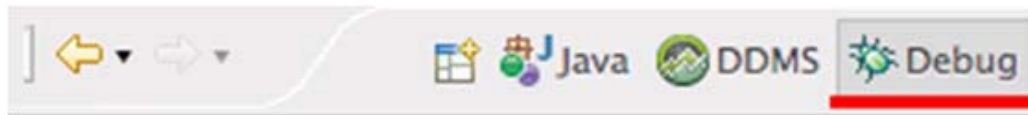
제 데이터터를 사용해 테스트하는 법이 있다.

- (3) 디버거(Debugger)는 디버그를 돕는 도구이다. 디버거는 주로 원하는 코드에 중단점을 지정하여 프로그램 실행을 정지하고, 메모리에 저장된 값을 살펴보며, 실행을 재개하거나, 코드를 단계적으로 실행하는 등의 동작을 한다. 고급 디버거들은 메모리 충돌 감지, 메모리 누수 감지, 다중 스레드 관리 등의 기능도 지원한다.

2. Eclipse 디버거 사용법

(1) 이클립스 디버거 뷰

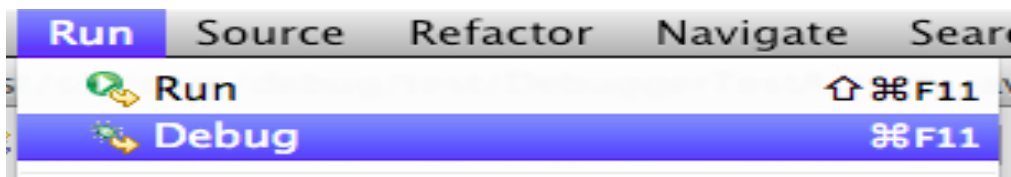
이클립스는 디버거 뷰를 제공하여 디버거를 사용할 수 있도록 한다. 디버거 뷰는 우측 상단에 Debug 뷰에 들어가면 확인할 수 있다.



[그림 2-1] 이클립스 디버거 뷰

(2) 디버깅의 시작

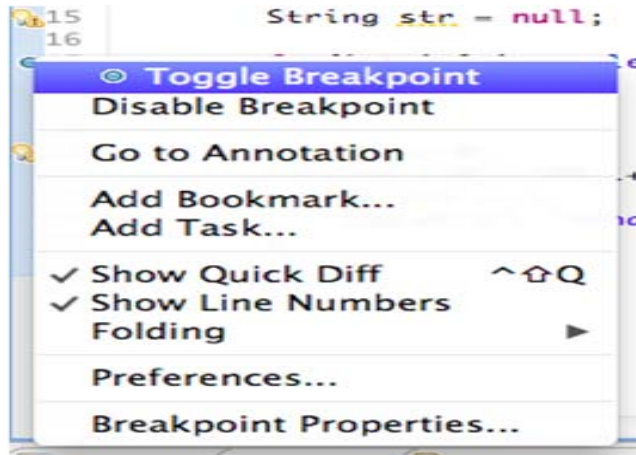
이클립스 상단메뉴>Run에서 Debug를 실행하면 디버거가 작동하게 된다.



[그림 2-2] 디버깅 시작

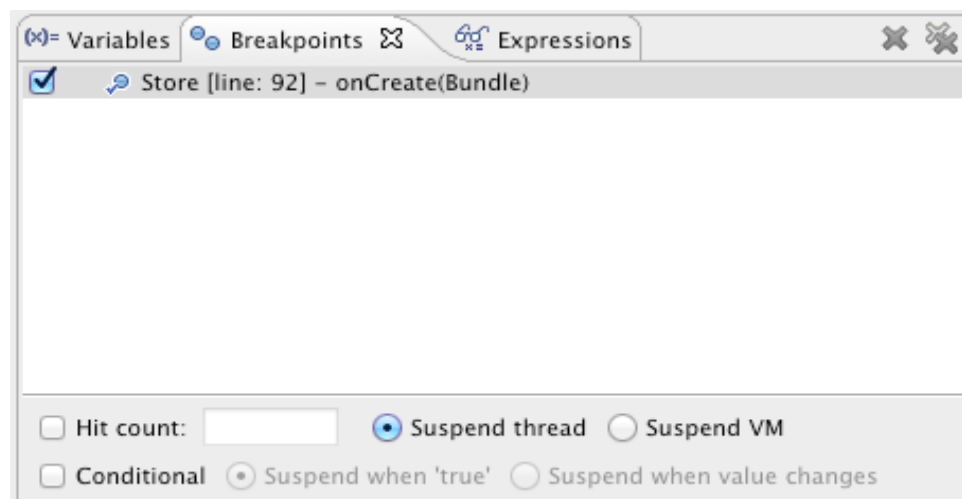
(3) 브레이크 포인트 설정과 뷰

- (가) 에러가 일어나는 라인이나 혹은 의심이 가는 변수를 추적할 수 있는 라인 위치에 브레이크 포인트를 지정하고 프로그램을 디버깅하면, 해당 라인을 실행할 때 디버거가 작동하게 되고 그곳에서 프로그램을 라인 별로 진행해 가며 관찰을 진행할 수 있다.
- (나) 브레이크 포인트 설정 방법은, 편집기 왼쪽에 파란 부분(마커 바)을 더블 클릭한다. 혹은 마우스를 우클릭하여 'Toggle Breakpoint' 를 선택한다. 설정 후 다시 더블 클릭하게 되면 브레이크 포인트가 사라진다(디버깅 진행 중에도 브레이크 포인트를 추가로 설정 할 수 있다.).



[그림 2-3] 브레이크 포인트 설정과 뷰

- (다) 디버그의 브레이크 포인트 뷰에서 지금까지 걸어 놓은 모든 브레이크 포인트들의 위치를 확인할 수 있고 활성화/비활성화, 삭제도 할 수 있다. 여러 브레이크 포인트가 걸려 있을 때에는 이 탭에서 확인하고 관리하는 것이 더 유용하다.



[그림 2-4] 브레이크 포인트 위치확인

(4) 스텝 단위 진행

- (가) 디버그 모드로 실행 중, 지정되어 있는 브레이크 포인트에 다다르면 디버거가 작동하고, 해당 라인부터 라인단위의 실행이 가능해진다.(Step-by-Step)
- (나) 디버그 뷰의 버튼 사용법은 아래와 같다.



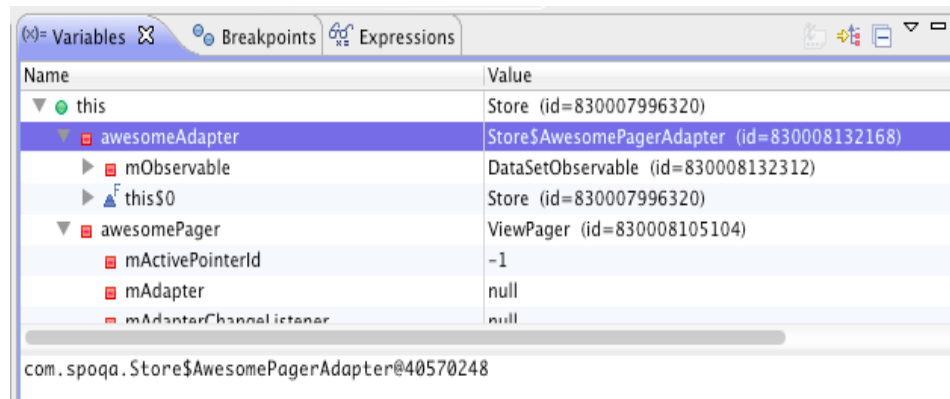
[그림 2-5] 디버그 뷰 버튼 화면

<표 2-5> 디버그 뷰 버튼 설명

용어	설명
Resume	다음 브레이크 포인트를 만날 때까지 진행
Suspend	현재 작동하고 있는 Thread를 멈춤
Terminate	프로그램(디버그모드 실행)을 종료
Step Into	Method를 포함한 라인을 만나면 Method 안으로 진입
Step Over	다음 라인으로 이동(Method가 있어도 Method 안으로 진입하지 않고 다음 라인으로 이동)
Step Return	Current Method에서 즉시 Return
Drop to Frame	Method를 처음부터 다시 실행

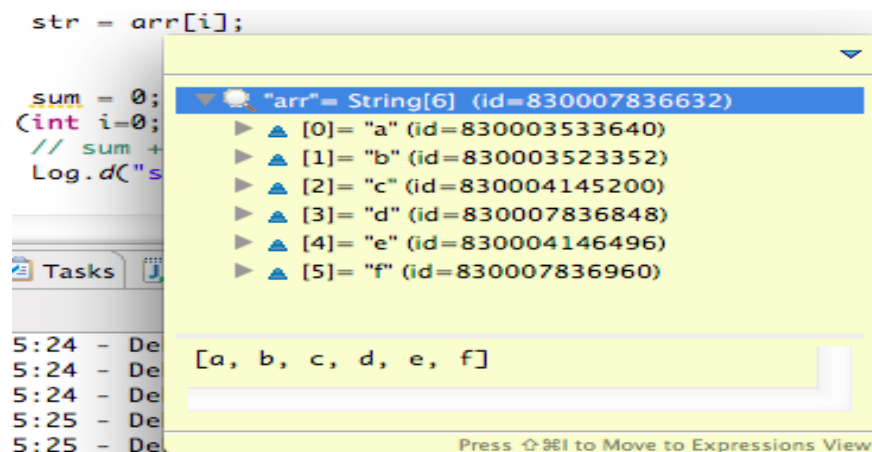
(5) 변수 뷰(Variables View)

(가) 변수의 값이나 오브젝트의 상태를 표시하는 뷰



[그림 2-6] 변수 뷰 탭 화면

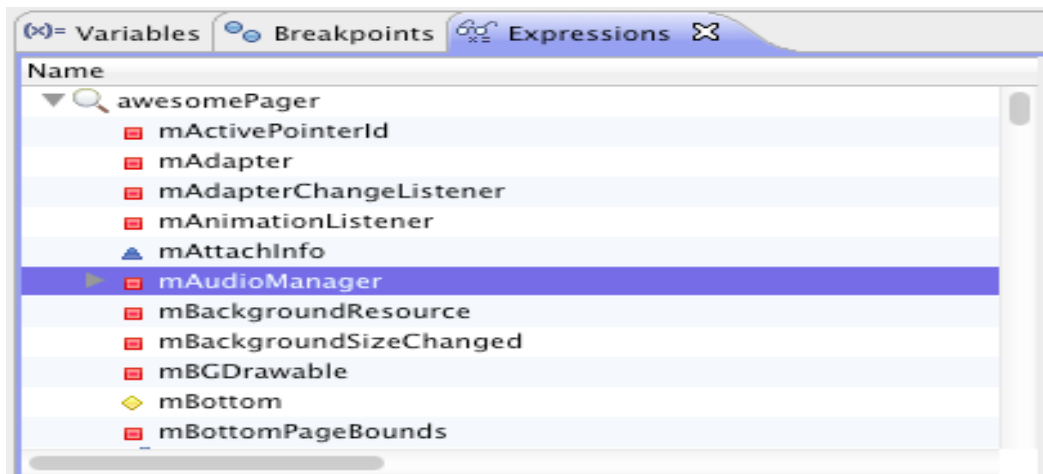
(나) Eclipse Editor의 변수 위치에서 마우스 우클릭>Inspect를 선택하면 Inspector 창이 뜨게 되고, 이 창에서 해당 변수를 Expression 뷰에서 지속적으로 변수의 상태를 관찰할 수 있다.



[그림 2-7] 변수 뷰 활용

(6) Expression View

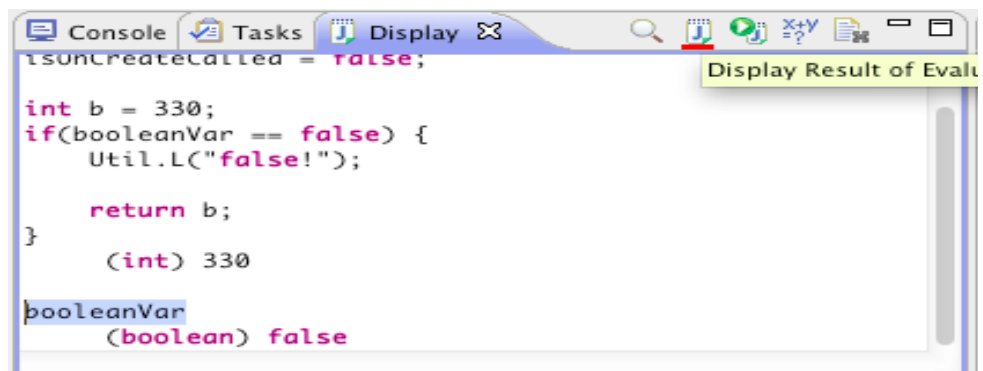
변수 이름을 입력하거나 수행해 보고 싶은 명령어를 직접 입력하여 결과를 확인할 수 있다.



[그림 2-8] Expression 뷰 활용

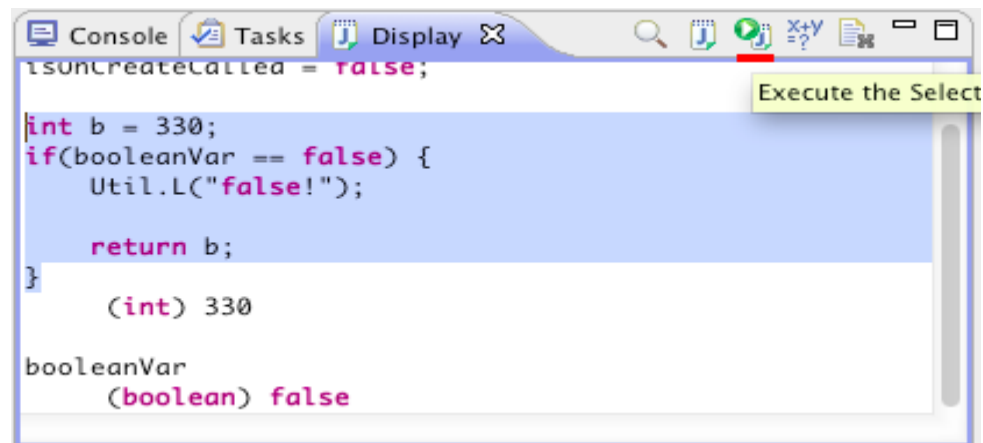
(7) Display View

(가) 현재의 Context에서 사용가능한 명령어를 실행하거나 변수의 값을 조작할 수 있다.



[그림 2-9] Display 뷰 활용-1

(나) 우상단의 두 번째 버튼을 선택하여 값을 반환받을 수 있다.



[그림 2-10] Display 뷰 활용-2

(다) 세 번째 버튼을 선택하여 Display View에서 선택한 영역을 실행할 수 있다.

(8) Conditional Break Point

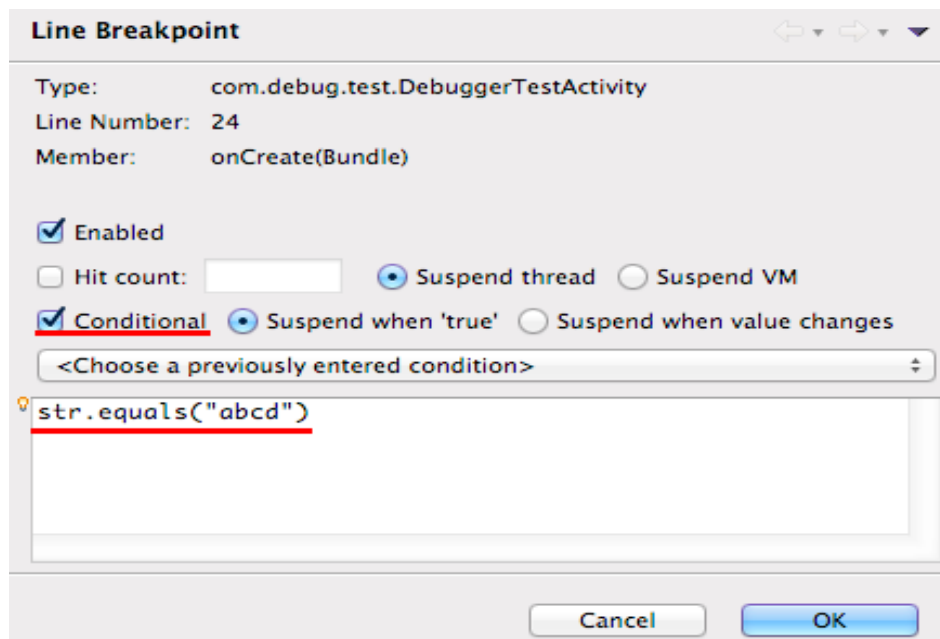
(가) Loop 구문과 같은 반복문 내부의 코드에 브레이크 포인트를 지정할 때, 특정 Index 값에서 브레이크 포인트가 동작하도록 조건을 설정할 수 있다.

(나) 브레이크 포인트를 지정하고 그 위치에서 마우스 우클릭>Breakpoint Properties를 선택한다.



[그림 2-11] Conditional Break Point 지정-1

(다) Conditional Check Box를 Check하고 원하는 조건식을 입력한다.



[그림 2-12] Conditional Break Point 지정-2

(라) Hit count에 값을 지정하면 해당 라인의 브레이크 포인트가 Hit count만큼 실행된 이후 디버깅화면으로 전환하게 된다.



[그림 2-13] Conditional Break Point 지정-3

수행 내용 / 공통 모듈 테스트 하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

① 테스트 케이스를 작성한다.(Simple Case 예제)

테스트케이스 ID	2.2	테스트케이스명	Change PIN
수행시스템	ATM	하위시스템	PIN
작성자	홍길동	작성일자	20XX-XX-XX
수행자	홍길동	수행일자	20XX-XX-XX

선행조건	사용자는 유효한 ATM카드를 사용해야 한다. 현재 PIN은 1234이다. 시스템은 Main Menu에서 시작한다.
------	-----------------------------------------------------------------------

No	Action	예상응답	결과	비고
1	'Change PIN'버튼 클릭	새로운 PIN을 생성할 것인지 묻는다.		
2	'5555' 입력	PIN 입력창이 Pop-up된다. 입력항목이 입력Focus를 가진다.		
3	'5555' 재입력	입력확인 항목이 입력 Focus를 가진다.		
4	'확인' 버튼 클릭	시스템의 Main Menu로 돌아간다.		
5	후행 확인			

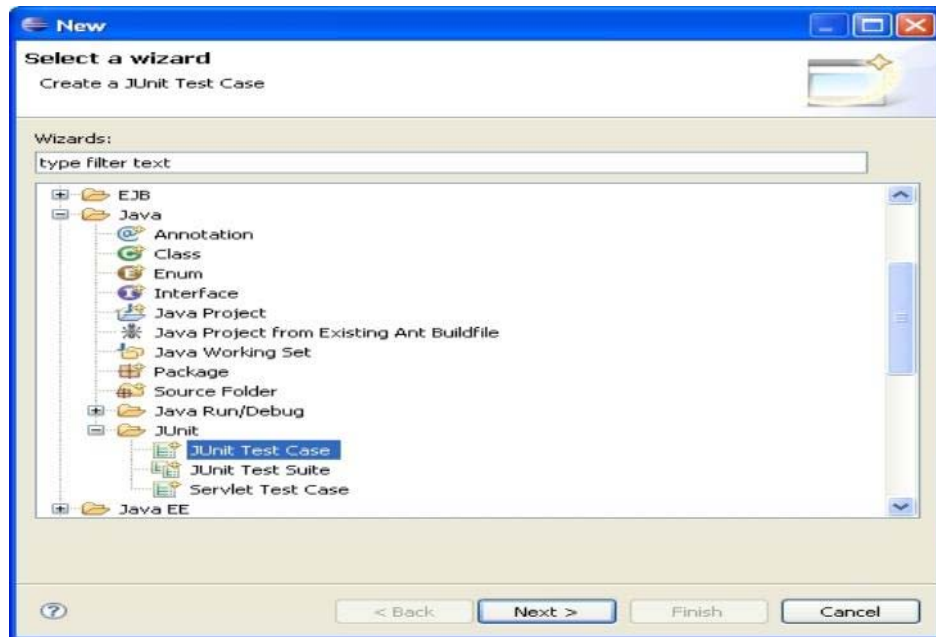
후행확인	DB에 새로운 PIN값 '5555'의 저장여부를 확인한다.
------	----------------------------------

② 테스트 데이터를 작성한다.

테스트에 필요한 DB Data 또는 File Data를 규칙에 맞게 생성한다.

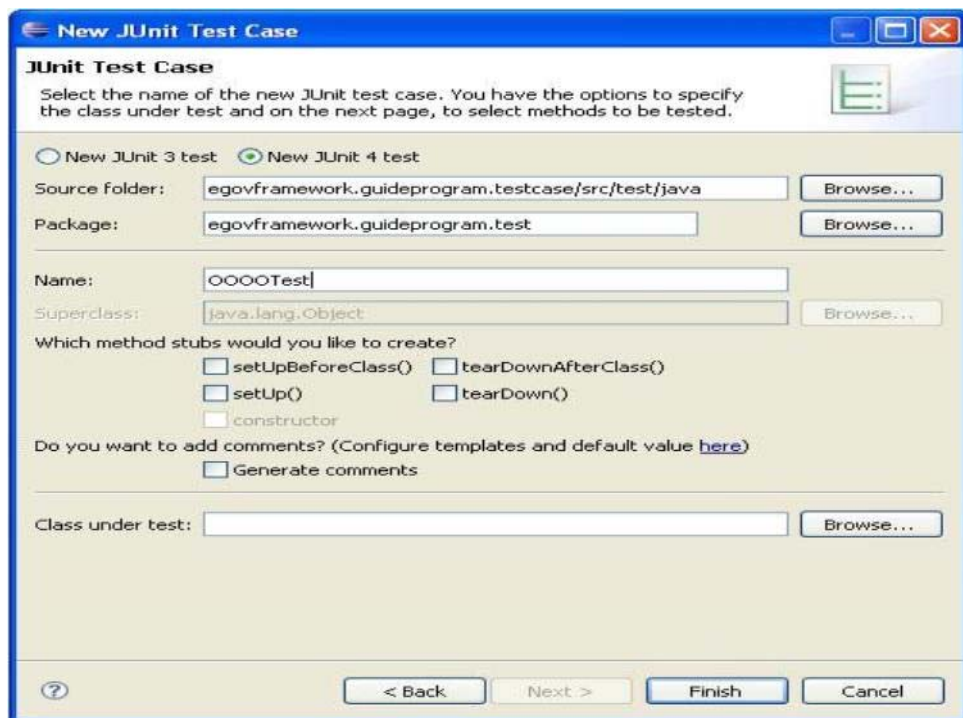
③ 테스트 도구를 설정한다.

1. JUnit을 사용하기 위한 Wizard를 선택하고 Next를 클릭한다.



[그림 2-14] JUnit 테스트케이스 생성-1

2. Test Case 의 이름을 입력하고 Finish를 클릭한다.



[그림 2-15] JUnit 테스트케이스 생성-2

3. Stub method 생성을 클릭한 경우 아래와 같이 생성된다.

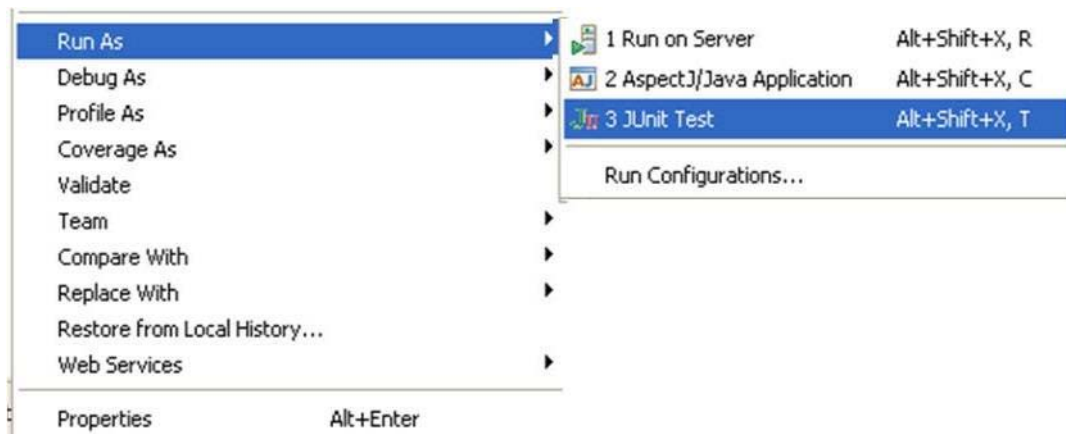
```
public class OOOOTest {  
    @BeforeClass  
    public static void setUpBeforeClass() throws Exception {  
    }  
    @AfterClass  
    public static void tearDownAfterClass() throws Exception {  
    }  
    @Before  
    public void setUp() throws Exception {  
    }  
    @After  
    public void tearDown() throws Exception {  
    }  
}
```

출처: 전자정부프레임워크 TestCase

http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev:tst:test_case

4 테스트 도구를 실행한다.

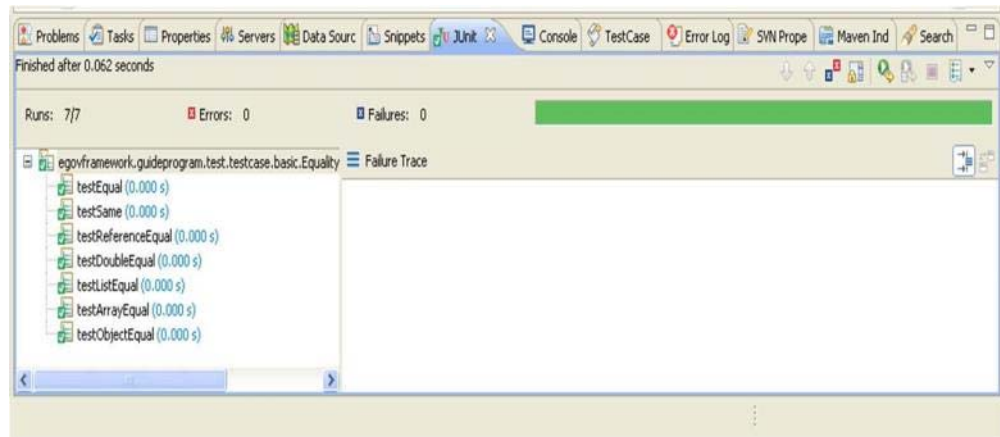
테스트 프로그램 오른쪽 버튼 클릭 후 Run As... 를 이용하여 수행한다.



[그림 2-16] JUnit 실행

⑤ 테스트 결과를 명세화한다.

1. 테스트 결과를 확인한다.



[그림 2-17] 테스트 결과 확인

2. 테스트케이스 계획서에 결과 및 필요사항을 기록한다.

⑥ 결함을 보완한다.

오류 디버깅을 통해 결함을 보완한다.

수행 tip

- 개발된 공통 모듈의 기능과 인터페이스 테스트를 위한 테스트케이스를 작성하고, 그에 따른 단위테스트를 수행할 수 있도록 한다.

학습 2 교수 · 학습 방법

교수 방법

- 공통 모듈의 상세 설계 내용을 이해하여 이를 개발언어를 활용하여 작성할 수 있는 절차를 구체적으로 제시한 후 내용에 대해 설명한다.
- 공통 모듈은 여러 애플리케이션에서 활용됨에 따라 최적화된 코드로 작성되어야 함을 인지시키며, 효율적 코드 작성의 필요성을 제시하고 이해시킬 수 있도록 한다.
- 공통 모듈 식별을 위한 결합도와 응집도의 개념에 대해 구체적으로 설명하고, 이런 원리를 어떻게 공통 모듈에 적용하는지에 대한 내용을 중심으로 설명한다.
- 개발된 공통 모듈을 테스트 할 수 있는 테스트케이스 작성 기법에 대해 특성 및 차이점을 비교분석하여 제시하고 설명한다.
- 공통 모듈 테스트를 위해 작성된 테스트케이스를 효율적으로 테스트 할 수 있는 도구선정시 고려사항을 제시하고 설명한다.

학습 방법

- 공통 모듈 상세 설계서를 파악하여 개발언어로 작성할 수 있는 절차에 대해 이해한다.
- 공통 모듈의 중요성을 인지하여 효율적 코드로 작성될 수 있도록 고급기법 습득의 필요성에 대해 인식하고 해당 내용을 학습한다.
- 공통 모듈 식별 원리인 결합도와 응집도에 대해 이해함으로써 모듈 작성 시 각 원칙에 충실하게 코드를 작성할 수 있어야 함을 이해한다.
- 개발된 공통 모듈 테스트를 위한 테스트케이스 작성기법에 대해 이해하고, 모듈 특성을 고려한 작성기법 선정 및 코드 작성법에 대해 학습한다.
- 공통 모듈 테스트를 위해 작성된 테스트케이스를 효율적으로 테스트할 수 있는 도구를 선정하고 활용하는 방법에 대해 학습한다.

학습 2 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
공통 모듈 작성	- 공통 모듈의 상세 설계를 기반으로 프로그래밍 언어와 도구를 활용하여 업무 프로세스 및 서비스의 구현에 필요한 공통 모듈을 작성할 수 있다.			
	- 소프트웨어 측정지표 중 모듈간의 결합도는 줄이고 개별 모듈들의 내부 응집도를 높인 공통 모듈을 구현할 수 있다.			
공통 모듈 테스트	- 개발된 공통 모듈의 내부 기능과 제공하는 인터페이스에 대해 테스트할 수 있는 테스트 케이스를 작성하고, 단위 테스트를 수행하기 위한 테스트 조건을 명세화 할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
공통 모듈 작성	- 공통 모듈 명세에 따른 기능을 개발언어와 도구를 활용하여 개발된 모듈의 완전성			
	- 결합도를 줄이고 응집도를 높일 수 있는 공통 모듈 개발의 기본원칙에 대한 이해			
공통 모듈 테스트	- 공통 모듈의 기능과 인터페이스 테스트를 위한 테스트케이스의 적정성			

• 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
공통 모듈 작성	- 공통 모듈 명세에 따른 기능을 개발언어와 도구를 활용하여 개발된 모듈의 완전성			
	- 결합도를 줄이고 응집도를 높일 수 있는 공통 모듈 개발의 기본원칙에 대한 이해			
공통 모듈 테스트	- 공통 모듈의 기능과 인터페이스 테스트를 위한 테스트케이스의 적정성			

피드백

1. 평가자 체크리스트

- 실습 과정을 체크리스트에 따라 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.
- 공통 모듈 명세를 파악하여 작성된 개발 모듈이 설계서대로 개발되었는지 확인하여 비정상 동작 시 그 문제점을 정리하여 돌려준다.
- 결합도와 응집도의 개념의 이해도를 파악하여 부족한 이해도에 대해 정리하여 돌려준다.
- 공통 모듈 테스트를 위한 테스트케이스 작성 기법과 테스트 결과를 보고 미비사항을 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 개발언어와 도구의 활용에 따른 공통 모듈 명세서에 정의된 기능이 정상적으로 동작하는지를 보고 비정상 동작의 원인과 조치 방안에 대해 정리하여 돌려준다.
- 개발된 공통 모듈의 결합도와 응집도를 파악하여 미비사항을 정리하여 돌려준다.
- 공통 모듈 명세를 파악하여 작성된 개발 모듈이 정상적으로 동작하는지 확인하여 비정상 동작 시 그 문제점을 정리하여 돌려준다.

학습 1	개발환경 구축하기(LM2001020203_14v2.1)
학습 2	공통 모듈 구현하기(LM2001020203_14v2.2)
학습 3	서버 프로그램 구현하기 (LM2001020203_14v2.3)
학습 4	배치 프로그램 구현하기(LM2001020203_14v2.4)
학습 5	개발자 단위 테스트하기(LM2001020203_14v2.5)
학습 6	애플리케이션 성능 개선하기(LM2001020203_14v2.6)

3-1. 서버 프로그램 작성

학습 목표

- 애플리케이션 설계를 기반으로, 프로그래밍 언어와 도구를 활용하여 서버 프로그램 구현 기술에 부합하는 서버 프로그램을 개발할 수 있다.
- 클라이언트 프로그램에 대한 종속도를 낮출 수 있고 쉽게 연동할 수 있는 서버 프로그램을 개발할 수 있다.

필요 지식 /

① 프레임워크에 대한 이해

1. 소프트웨어 프레임워크의 정의

(1) 효율적인 정보시스템 개발을 위한 코드 라이브러리, 애플리케이션 인터페이스 규약 (Application Programming Interface), 설정정보 등의 집합으로서 소프트웨어 구성에 필요한 기본 뼈대를 제공한다.

(2) 광의적으로 정보시스템의 개발 및 운영을 지원하는 도구 및 가이드 등을 포함한다.

2. 프레임워크의 특징

<표 3-1> 프레임워크 특징

특징	설명
모듈화 (modularity)	프레임워크는 구현을 인터페이스 뒤에 감추는 캡슐화를 통해서 모듈화를 강화한다. 프레임워크의 모듈화는 설계와 구현의 변경에 따르는 영향을 극소화함으로써 손쉽게 소프트웨어의 품질을 향상시킬 수 있게 한다.
재사용성 (reusability)	프레임워크가 제공하는 인터페이스는 여러 애플리케이션에서 반복적으로 사용할 수 있는 일반적인 컴포넌트를 정의할 수 있게

특징	설명
	<p>함으로써 재사용성을 높여준다. 프레임워크 재사용성은 도메인 지식과 경험있는 개발자들의 이전의 노력을 활용하여, 애플리케이션의 요구사항과 소프트웨어 설계에 대한 공통의 솔루션을 반복적으로 재개발하고, 그에 대해 유효성을 다시 확인하는 작업을 피할 수 있게 한다.</p> <p>프레임워크 컴포넌트를 재사용하는 것은 소프트웨어의 품질, 성능, 신뢰성, 상호 운용성을 향상시킬 뿐만 아니라, 프로그래머의 생산성을 상당히 높여준다.</p>
확장성 (extensibility)	<p>프레임워크는 다형성(polymorphism)을 통해 애플리케이션이 프레임워크의 인터페이스를 확장할 수 있게 한다. 프레임워크 확장성은 새로운 애플리케이션 서비스와 특성을 커스터마이징하는 것을 보장하는 데 필수적인 사항이며, 또한 프레임워크를 애플리케이션의 가변성으로부터 분리함으로써 재사용성의 이점을 얻게 한다.</p>
제어의 역흐름 (inversion of control)	<p>일반적으로 어떤 모듈을 호출함으로써 해당 모듈을 재사용하게 된다. 그러나 프레임워크에서는 이와는 반대되는 제어 흐름으로의 재사용성을 지원한다. 여기에서 이른바 “헐리우드 원리(Hollywood Principle)” 즉, “나를 부르지 마라. 내가 너를 부를 것이다” 라는 원리가 적용되는 것이다. 즉, 프레임워크 코드가 전체 애플리케이션의 처리흐름을 제어하며, 특정한 이벤트가 발생할 때 다형성(Polymorphism)을 통해 애플리케이션이 확장한 메서드를 호출함으로써 제어가 프레임워크로부터 애플리케이션으로 거꾸로 흐르게 한다. 이러한 제어의 역흐름을 통해서 프레임워크가 외부의 이벤트에 대해서 애플리케이션이 어떠한 메서드들을 수행해야 하는지를 결정할 수 있게 한다.</p>

3. 프레임워크의 구성 요소

〈표 3-2〉 프레임워크 구성 요소

특징	설명
실행환경	자바 기반의 응용시스템 개발 시 필수적인 기능을 패턴화하여 미리 구현해 둔 라이브러리 코드 묶음
개발환경	설계, 구현, 테스트 등 개발 생명주기(Life Cycle)상에 필요한 지원 도구 모음
운영환경	표준프레임워크를 기반으로 개발한 시스템에 대하여 운영 시에 필요한 의사소통 및 모니터링 도구 모음
관리환경	표준프레임워크에 대한 기술지원, 업그레이드 등을 관리하기 위한 표준프레임워크센터 내부 업무시스템

※ 예시) 전자정부 표준프레임워크의 구성요소

② 데이터 저장계층 또는 영속계층(Persistence Layer)에 대한 이해

DAO/DTO/VO란 영속계층(Persistence Layer)에서 사용되는 특정 패턴을 통해 구현되는 Java Bean이다.

<표 3-3> DAO/DTO/VO

용어	설명
DAO (Data Access Object)	특정 타입의 데이터베이스나 다른 지속적인 메커니즘(Persistence Mechanism)에 추상 인터페이스를 제공하는 객체이다. 애플리케이션 호출을 데이터 저장 부분(Persistence Layer)에 매핑함으로써 DAO는 데이터베이스의 세부내용을 노출하지 않고 특정 데이터 조작 기능을 제공한다. 이 고립성은 단일 책임 원칙(Single Responsibility Principle)에 기반한다.
DTO (Data Transfer Object)	<p>DTO는 프로세스 사이에서 데이터를 전송하는 객체를 의미한다. 이것을 이용하는 이유는 프로세스 간의 커뮤니케이션이 주로 개별 호출이 부담스러운 작업일 경우가 많은 원격 인터페이스(예, 웹 서비스)에 의해 이루어지기 때문이다. 대부분의 개별 호출이 클라이언트와 서버 간의 왕복 시간을 소모하기 때문에, 호출 횟수를 줄이는 방법 중 하나는 몇 번의 호출에 의해 전송될 데이터를 모으는 DTO를 이용해서 한번만 호출하게 하는 것이다.</p> <p>DTO와 DAO의 차이는 DTO는 스스로의 데이터를 저장 및 회수하는 기능을 제외하고 아무 기능도 갖고 있지 않다는 것이다. DTO는 테스트가 필요한 비즈니스 로직을 갖고 있지 않아야 하는 간단한 객체이다.</p>
VO (Value Object)	<p>VO는 간단한 독립체(Entity)를 의미하는 작은 객체를 의미한다. VO의 값은 그 정체성에 의해 결정되지 않는데, 이 뜻은 두 VO들은 그 두 가지가 같은 값을 가지고 있을 때 동일한 것이지 같은 객체라고 해서 동일하지는 않다는 것이다.</p> <p>작기 때문에, 같은 독립체를 대변하는 복수의 같은 VO들이 존재할 수 있다. 가끔 하나의 인스턴스에 의존하고 그에 기반한 레퍼런스를 사용하기보다는 새 객체를 생성하는 것이 더 간편하다.</p> <p>VO는 불변해야 하며, 이는 두 VO가 같도록 생성되었으면 같은 값으로 남아 있어야 한다는 것을 암시한다.</p> <p>자바에서 VO는 작은 자바빈이나 DTO 혹은 POJO(Plain Old Java Object, 간단한 자바 클래스의 인스턴스)를 의미한다.</p>

수행 내용 / 서버 프로그램 작성하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

- ① 처리흐름 정의 및 오브젝트를 선언(Declaration)한다.

1. 처리 흐름

<표 3-4> 처리흐름 정의 절차

순서	처리
1	Jsp에서 요청
2	JavaScript에서 유효성 검사 및 Submit
3	web.xml에서 servlet-mapping으로 해당 컨트롤러로 이동
4	해당 컨트롤러에서 받은 데이터를 회원 DTO 클래스에 Set
5	Insert할 데이터를 DB작업을 수행하기 위해 DAO로 전달
6	DAO는 mybatis를 이용하여 DB에 데이터를 Insert하고 결과를 리턴
7	Controller는 받은 결과에 따라 작업을 성공/실패로 출력

2. 오브젝트 선언

```
package join;

import java.io.IOException;

import java.io.PrintWriter;

import java.text.ParseException;

import java.text.SimpleDateFormat;
```

```

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class Joincontroller extends HttpServlet {

    @Override

    protected void doPost(HttpServletRequest req, HttpServletResponse
    resp) throws ServletException, IOException {

        ...

    }

}

```

② DB Table 정의 및 오브젝트 속성(Property)을 정의한다.

<표 3-5> Table 정의 및 오브젝트 속성 정의

순서	컬럼	자료형	길이
1	id	VARCHAR2	20
2	pw	VARCHAR2	20
3	name	VARCHAR2	15
4	nick	VARCHAR2	20
5	addr	VARCHAR2	40
6	email	VARCHAR2	30
7	birth	DATE	

```

JoinDTO dto = new JoinDTO();

JoinDAO dao = new JoinDAO();

PrintWriter out = resp.getWriter();

int result;

```

③ Method 프로토타입을 정의한다.

```
public JoinDAO {  
    ...  
}  
  
public int insertjoin(JoinDTO dto) {  
    ...  
}
```

④ I/O 오브젝트인 DTO/VO를 정의한다.

```
public class JoinDTO {  
    String id;  
    String pw;  
    String name;  
    String nick;  
    String addr;  
    String email;  
    Date birth;  
    public JoinDTO() {  
        // TODO Auto-generated constructor stub  
    }  
    ...  
}
```

⑤ Input Validation Check를 구현한다.

```
if (!this.isType(year, "int")) {  
    error = true;  
    System.err.println("Error: must be a whole number.");  
} else {  
    error = false;  
}
```

6 Main Logic을 구현한다.

```
public class Joincontroller extends HttpServlet{

    @Override

    protected void doPost(HttpServletRequest req, HttpServletResponse
    resp) throws ServletException, IOException {

        req.setCharacterEncoding("UTF-8");

        //req.setCharacterEncoding("EUC-KR");

        //한글처리

        JoinDTO dto = new JoinDTO();

        JoinDAO dao = new JoinDAO();

        PrintWriter out = resp.getWriter();

        int result;

        dto.setId(req.getParameter("id"));

        dto.setPw(req.getParameter("pw"));

        dto.setName(req.getParameter("name"));

        dto.setNick(req.getParameter("nick"));

        dto.setAddr(req.getParameter("addr"));

        dto.setEmail(req.getParameter("email"));


        SimpleDateFormat sdf = new SimpleDateFormat("yyyy/mm/dd");

        Date date = null;

        try {

            date =

            sdf.parse(req.getParameter("year")+"/"+req.getParameter("month")+"/"+r
            eq.getParameter("day"));

            //입력받은 년,월,일을 date 포맷으로 변경

        } catch (ParseException e) {

            e.printStackTrace();

        }

    }

}
```

```

        }
        dto.setBirth(date);
        ...
    }
}

```

7 Output Validation Check 구현테스트 케이스를 작성한다.

```

public class Joincontroller extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        ...
        result = dao.insertjoin(dto);
        if(result==0){
            System.out.println("Failed");
        }else{
            System.out.println("Sucsses");
        }
    }
}

```

수행 tip

- 클라이언트 프로그램에 독립적으로 활용가능하며, 개발환경으로 정의된 개발언어와 도구의 특성을 충분히 반영하여 생산성과 품질이 확보된 서버 프로그램을 개발하는 것이 중요하다.

3-2. 서버 프로그램 테스트

학습 목표

- 개발된 서버 프로그램 내부 기능과 제공하는 인터페이스에 대해 테스트할 수 있는 테스트 케이스를 작성하고, 단위 테스트를 수행하기 위한 테스트 조건을 명세화할 수 있다.

필요 지식 /

서버 프로그램 테스트를 위한 테스트케이스 생성 및 단위 테스트를 위한 테스트 도구의 활용
이 공통 모듈 테스트와 동일하므로 해당 내용을 준용하기로 한다.

① 단위 테스트 도구의 사용

1. 테스트 작성

- (1) 프로젝트 “JUnitTest” 를 작성한다.
- (2) TestCase의 서브클래스에서 테스트를 구현한다(표준 클래스 마법사 또는 특수 테스트 케이스 마법사를 사용하여 이를 수행할 수 있다).

2. 테스트 실행

도구 모음에서 실행 단추를 누른다.

② 디버깅 도구의 사용법

☞ 학습내용2-2 공통 모듈 테스트 참조

수행 내용 / 서버 프로그램 테스트하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 도구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

서버 프로그램 테스트를 위한 테스트케이스 생성 및 단위 테스트를 위한 테스트 도구의 활용이 공통 모듈 테스트와 동일하므로 해당 내용을 준용하기로 한다.

① 테스트 케이스를 작성한다.

② 테스트 데이터를 작성한다.

테스트에 필요한 DB Data 또는 File Data를 규칙에 맞게 생성한다.

③ 테스트 도구를 설정한다.

1. JUnit을 사용하기 위한 Wizard를 선택하고 Next를 클릭한다.
2. Test Case의 이름을 입력하고 Finish를 클릭한다.

④ 테스트 도구를 실행한다.

테스트 프로그램 오른쪽 버튼 클릭 후 Run As...를 이용하여 수행한다.

⑤ 테스트 결과를 명세화한다.

1. 테스트 결과를 확인한다.
2. 테스트케이스 계획서에 결과 및 필요사항을 기록한다.

⑥ 결함을 보완한다.

오류 디버깅을 통해 결함을 보완한다.

수행 tip

- 개발된 서버 프로그램이 제공해야 할 기능과 인터페이스를 테스트하기 위해 적절한 테스트케이스를 작성함으로써 원활한 단위테스트가 수행될 수 있도록 준비하는 것이 중요하다.

학습 3 교수 · 학습 방법

교수 방법

- 애플리케이션 설계 내용을 개발 언어와 도구를 활용하여 개발하는 절차에 대해 구체적으로 설명한다.
- 애플리케이션 개발 시 개발 생산성을 높이기 위한 도구 유형과 각 도구의 특성에 대해 제시하여 효율적인 개발을 할 수 있도록 도구 활용방안에 대해 설명한다.
- 서버 프로그램이 클라이언트 프로그램에 종속되지 않도록 유연성을 확보하는 방안에 대해 정리하여 설명한다.
- 개발된 서버 프로그램에서 제공하는 기능에 대한 테스트케이스 작성기법에 대해 정리하여 설명한다.
- 개발 모듈의 단위 테스트 수행을 위한 테스트 조건 명세화 절차와 내용에 대해 정리한 후 상세히 설명한다.

학습 방법

- 애플리케이션 설계 내용을 개발 언어와 도구를 활용하여 개발하는 절차에 대해 학습하고 숙지할 수 있도록 학습한다.
- 애플리케이션 개발 시 개발 생산성을 높이기 위한 도구 유형과 각 도구의 특성에 대해 인지하여 효율적인 개발을 할 수 있도록 실습한다.
- 서버 프로그램이 클라이언트 프로그램에 종속되지 않도록 유연성을 확보하는 방안에 대해 학습한다.
- 개발된 서버 프로그램에서 제공하는 기능에 대한 테스트케이스 작성기법에 대해 학습하여, 이를 실무에 활용할 수 있을 정도로 반복적으로 실습한다.
- 개발 모듈의 단위 테스트 수행을 위한 테스트 조건 명세화 절차와 내용에 대해 이해한다.

학습 3 평가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
서버프로그램 작성	- 애플리케이션 설계를 기반으로, 프로그래밍 언어와 도구를 활용하여 서버 프로그램 구현 기술에 부합하는 서버 프로그램을 개발할 수 있다.			
	- 클라이언트 프로그램에 대한 종속도를 낮출 수 있고 쉽게 연동할 수 있는 서버 프로그램을 개발할 수 있다.			
서버프로그램 테스트	- 개발된 서버 프로그램 내부 기능과 제공하는 인터페이스에 대해 테스트할 수 있는 테스트 케이스를 작성하고 단위 테스트를 수행하기 위한 테스트 조건을 명세화할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
서버프로그램 작성	- 애플리케이션 설계서와 서버 프로그램 간 정합성			
	- 클라이언트 프로그램과의 독립성과 연동 용이성			
서버프로그램 테스트	- 서버 프로그램 테스트케이스와 테스트 조건 적정성			

• 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
서버프로그램 작성	- 애플리케이션 설계서와 서버 프로그램 간 정합성			
	- 클라이언트 프로그램과의 독립성과 연동 용이성			
서버프로그램 테스트	- 서버 프로그램 테스트케이스와 테스트 조건 적정성			

피드백

1. 체크리스트를 통한 관찰

- 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.
- 애플리케이션 설계서에 따라 서버 프로그램이 작성되었는지를 확인하여, 미비사항을 보완할 수 있도록 미비내역을 정리하여 돌려준다.
- 클라이언트 프로그램이 변경됨에 따른 서버 프로그램의 영향도 분석결과를 파악하여, 독립성에 문제가 있는 경우를 발췌하고, 이를 극복할 수 있는 방안에 대해 정리하여 돌려준다.
- 서버 프로그램 테스트케이스 작성 기법을 확인하고, 미비사항을 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 서버 프로그램 테스트를 위한 테스트케이스 적용기법에 대해 파악한 후, 프로그램 특성에 맞는 기법선정 방안의 미비사항을 정리하여 돌려준다.
- 서버 프로그램 테스트케이스 생성 방법을 확인하고, 미비사항을 정리하여 돌려준다.

학습 1	개발환경 구축하기(LM2001020203_14v2.1)
학습 2	공통 모듈 구현하기(LM2001020203_14v2.2)
학습 3	서버 프로그램 구현하기(LM2001020203_14v2.3)
학습 4	배치 프로그램 구현하기 (LM2001020203_14v2.4)
학습 5	개발자 단위 테스트하기(LM2001020203_14v2.5)
학습 6	애플리케이션 성능 개선하기(LM2001020203_14v2.6)

4-1. 배치 프로그램 작성

학습 목표

- 애플리케이션 설계를 기반으로 프로그래밍 언어와 도구를 활용하여 배치 프로그램 구현 기술에 부합하는 배치 프로그램을 개발할 수 있다.
- 목표 시스템을 구성하는 하위 시스템 간의 연동 시, 안정적이고 안전하게 동작할 수 있는 배치 프로그램을 개발할 수 있다.
- 개발하고자 하는 목표 시스템의 잠재적 보안 취약성이 제거될 수 있도록 배치 프로그램을 개발할 수 있다.

필요 지식 /

① 배치 프레임워크-Spring Batch 예시 및 설명

1. 배치 애플리케이션의 필수요소

배치 애플리케이션의 필수요소는 다음 <표 4-1>과 같다.

<표 4-1> 배치 애플리케이션 필수요소

요소	설명
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 동작하고있는 다른 애플리케이션을 방해하지 말아야 한다.

2. Spring Batch의 핵심기능

Spring Batch의 핵심 기능은 다음 <표 4-2>와 같다.

<표 4-2> Spring Batch 핵심기능

요소	설명
스프링 프레임워크 기반	스프링의 DI, AOP 및 다양한 엔터프라이즈 지원 기능을 사용한다.
배치기반 처리	데이터를 읽고 쓸 때 가장 효과적인 방법을 이용하도록 강제한다.
자체제공 컴포넌트	데이터베이스나 파일로부터 데이터를 읽거나 쓰는 등의 배치 처리 시에 공통적으로 필요한 컴포넌트를 제공한다.
견고함과 안정성	선언적 생략과 처리 실패 후 재시도 설정을 제공한다.
성능	주어진 시간 내에 처리를 완료할 수 있어야 하고 동시에 동작하고 있는 다른 애플리케이션을 방해하지 말아야 한다.

3. Spring Batch가 제공하는 저장소 관련 기술

Spring Batch가 제공하는 저장소 관련 기술은 다음 <표 4-3>과 같다.

<표 4-3> Spring Batch 제공 기술

기능	설명
JDBC	페이징, 커서, 배치 업데이트 기능을 제공한다.
Hibernate	페이징, 커서 기능을 제공한다.
JPA(Java Persistence API)	페이징 기능을 제공한다.
iBatis	페이징 기능을 제공한다.
Flat File	델리미터 또는 고정 길이로 구분된 파일을 지원한다.
XML	스프링 OXM 기반의 StAX(Streaming API for XML)을 사용하여 파싱한다. OXM은 JAXB(Java Architecture for XML Binding), XStream과 Castor를 지원한다.

4. Spring Batch의 핵심 컴포넌트

Spring Batch의 핵심 컴포넌트 구성 및 제공 기능은 다음 <표 4-4>와 같다.

<표 4-4> Spring Batch 핵심컴포넌트와 제공기능

컴포넌트(구성요소)	설명
Job Repository	Job Execution 관련 메타데이터를 저장하는 기반 컴포넌트이다.
Job Launcher	Job Execution 을 실행하는 기반 컴포넌트이다.
JPA(Java Persistence API)	페이징 기능을 제공한다.
Job	배치 처리를 의미하는 애플리케이션 컴포넌트이다.
Step	Job의 각 단계를 의미한다. Job은 일련의 연속된 Step으로 구성된다.
Tasklet	Step 내에서 반복 또는 트랜잭션 처리 용도로 사용된다.
Item	Datasource로부터 읽어 들인 또는 Datasource로 저장하는 각 레코드를 의미한다.
Chunk	특정 크기를 갖는 아이템 목록을 의미한다.
Item Reader	데이터 소스로부터 아이템을 읽어 들이는 컴포넌트이다.
Item Processor	Item Reader로 읽어 들인 아이템을 Item Writer를 사용해 저장하기 전에 처리하는 컴포넌트이다.
Item Writer	Item Chunk 를 데이터 소스에 저장하는 컴포넌트이다.

5. Spring Batch와 외부 시스템과의 관계

Spring Batch의 Job은 Crontab 등과 같은 시스템 스케줄러, 스크립트(셸, Perl 등) 등에 의해 발생된 다양한 이벤트에 의해 실행된다.

6. Spring Batch의 아키텍처

Spring Batch의 아키텍처는 Run, Job, Application, Data Tier로 구성되고, 각 Tier 별 특성을 설명하면 다음과 같다.

(1) Run Tier

Application의 Scheduling 및 실행을 담당하는 Tier로써, 스프링배치에서는 따로 Scheduling의 기능은 제공하지 않고 Quartz 같은 외부 모듈이나 Cron을 이용하도록 권고하고 있다.

(2) Job Tier

전체적인 Job의 수행을 책임지는 Tier로써, Job 내의 각 Step들을 지정한 상태와 정책에 따라 순차적으로 수행한다.

(3) Application Tier

Job을 수행하는 데 필요한 Component로 구성된다.

(4) Data Tier

Database, File 등 물리적 데이터소스와 결합이 이루어지는 영역이다.

② 배치스케줄러

1. 스케줄링의 개념

Scheduling 서비스는 애플리케이션 서버 내에서 주기적으로 발생하거나 반복적으로 발생하는 작업을 지원하는 기능(Job Scheduler라고도 한다.)

2. Quartz 스케줄러 (Open Source Batch Scheduler)

Quartz 스케줄러는 다음 <표 4-5>와 같다.

<표 4-5> Quartz 스케줄러 구성 및 제공기능

주요 요소	설명
Scheduler	Quartz 실행 환경을 관리하는 핵심 개체이다.
Job	사용자가 수행할 작업을 정의하는 인터페이스로서 Trigger 개체를 이용하여 스케줄할 수 있다.
JobDetail	작업명과 작업그룹과 같은 수행할 Job에 대한 상세 정보를 정의하는 개체이다.
Trigger	정의한 Job 개체의 실행 스케줄을 정의하는 개체로서 Scheduler 개체에게 Job 수행시점을 알려주는 개체이다.

출처: 전자정부프레임워크 - Scheduling 서비스

Quartz 스케줄러는 수행 작업을 정의하는 Job과 실행 스케줄을 정의하는 Trigger를 분리함으로써 유연성을 제공한다. Job과 실행 스케줄을 정의한 경우, Job은 그대로 두고 실행 스케줄만을 변경할 수 있다. 또한 하나의 Job에 여러 개의 실행 스케줄을 정의할 수 있다.

(1) Quartz 스케줄러 사용 예제

Quartz 스케줄러의 이해를 돕기 위해 간단한 예제를 살펴본다. 다음 예는 Quartz 메뉴얼에서 참조한 것으로 Quartz를 사용하는 방법과 사용자 Job을 설정하는 방법을 보여 준다.

(가) 사용자 정의 Job

사용자는 Job 개체를 생성하기 위해 org.quartz.Job 인터페이스를 구현하고 심각한 오류가 발생한 경우 JobExecutionException 예외를 던질 수 있다. Job 인터페이스는 단일 메서드로 execute()을 정의한다.

```
public class DumbJob implements Job {  
    public void execute(JobExecutionContext context)  
        throws JobExecutionException {  
        System.out.println("DumbJob is executing.");  
    }  
}
```

- 1) DumbJob은 Job 인터페이스의 execute() 메서드를 구현한다.
- 2) execute() 메서드는 단순히 Job이 수행됨을 표시하는 메시지를 출력한다.

(나) Quartz 사용 코드

```
JobDetail jobDetail =  
    new JobDetail("myJob", // Job 명  
        sched.DEFAULT_GROUP, // Job 그룹명('null' 값인 경우  
        DEFAULT_GROUP 으로 정의됨)  
        DumbJob.class); // 실행할 Job 클래스  
  
Trigger trigger = TriggerUtils.makeDailyTrigger(8, 30); // 매일 08시 30분 실행  
trigger.setStartTime(new Date()); // 즉시 시작  
trigger.setName("myTrigger");  
  
sched.scheduleJob(jobDetail, trigger);
```

- 1) 우선 Job 설정을 위해 JobDetail 클래스를 정의한다.
- 2) TriggerUtils를 이용하여 매일 8시30분 실행하는 Trigger를 생성한다.
- 3) 마지막으로, Scheduler에 JobDetail과 Trigger를 등록한다.

③ 파일처리 기법

1. File Sorting의 유형

File Sorting의 유형은 다음 <표 4-6>과 같다.

<표 4-6> File Sorting 유형

File Sorting 유형	설명
Internal Sorting	데이터가 적어서 주메모리 내에 Load하여 Sort하는 유형 수행시간이 빠르고 시스템 부하가 External Sorting에 비해 비교적 적게 발생한다.
External Sorting	데이터가 많아서 주메모리 용량을 초과하여 보조저장장치 (Disk 등)에 저장되어 있는 File을 Sorting하는 유형 주메모리에는 제한된 수의 데이터만을 유지한다. 시스템 부하와 성능을 고려하여야 한다. Run: 하나의 File을 여러 개의 Sub-File로 나누어 Internal Sorting 기법을 사용하여 Sort한 File 여러개로 분리된 Run을 Merge하여 최종적으로 Sort된 하나 의 File을 생성한다.

2. File Merge의 유형

<표 4-7> File Merge 유형

File Merge 유형	설명
m-원 합병 (M-way Merge)	m개(합병의 원 수)의 입력 파일을 동시에 처리하는 합병 입력 파일 m개, 하나의 출력 파일: m+1개의 파일을 사용 많은 입출력: 한 패스에 합병이 끝나지 않으면 런들을 다시 분배하기 위해 복사, 이동해야 함 이상적 정렬/합병: m개의 런에 m개의 입력 파일 사용하여 한 번의 m-원 합병을 적용 단점: 화일의 재분배 => 많은 I/O 필요
균형 합병 (Balanced Merge)	출력할 때, 미리 다음 단계에 사용할 입력 화일로 재분배 즉, 출력 화일을 다음 단계의 입력 화일로 직접 사용 m-원 자연합병: m + 1 개의 화일 필요 m-원 균형합병: 2m 개의 화일 필요 (m 입력화일, m 출력화일)
다단계 합병 (Polyphase Merge)	m 개의 입력 화일과 1 개의 출력 화일 입/출력 화일수가 같지 않음: “불균형” 합병 초기 입력 화일에 대한 런의 분배가 복잡 입력 화일의 어느 하나가 공백이 될 때까지 런들을 합병 공백이 된 입력 화일이 다음 합병 단계의 출력 화일이 됨
계단식 합병 (Cascade Merge)	정렬/합병 과정에서 런들의 복사작업을 줄이려는 불균형 합 병의 또 다른 형태

3. File 처리 시 성능 고려사항

- (1) 정렬 합병되는 레코드의 수
- (2) 레코드의 크기
- (3) 이용될 저장 장치의 수
- (4) 이용 가능한 입출력 채널에서의 저장 장치의 분포
- (5) 입력 파일에서 키 값의 분포
- (6) 가능한 런(서브화일)의 길이를 길게 만들어 런의 수를 최소화
- (7) 동시에 합병할 수 있는 런의 수를 늘리면 합병의 패스 수는 감소
- (8) 보조 저장 장치에 분산 저장하면 부수적인 입출력 연산 동반

수행 내용 / 배치 프로그램 작성하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

① Job과 Step을 정의한다.(XML Configuration)

```
<job id="ncs" incrementer="incrementer" xmlns="http://www.springframework.org/schema/batch">
  <step id="loginCount">  -- 복수의 Step을 정의할 수 있다.
    <tasklet>
      <chunk reader="ncsReader" writer="ncsWriter" commit-interval="1" />
    </tasklet>
  </step>
</job>

<bean id="ncsReader" class="org.springframework.batch.item.database.IbatisPagingItemReader">
  <property name="queryId" value="ncs.selectLoginCount" /> -- Query의 NameSpace ID
  <property name="sqlMapClient" ref="dataSqlMapClient" /> -- Database 선택
</bean>

<bean id="ncsWriter" class="org.springframework.batch.item.database.IbatisPagingItemWriter">
  <property name="queryId" value="ncs.insertLoginCount">
  <property name="sqlMapClient" ref="ncsSqlMapClient"/>
</bean>
```

② DB 처리를 위한 sqlQuery.xml을 작성한다.(iBatis 예)

```
<resultMap id="loginNcs" class="loginNCS">
    <result property="ncsDate" column="JOIN_DATE" />
    <result property="loginCount" column="LOGIN_COUNT" />
</resultMap>

<select id="selectLoginCount" resultMap="loginNcs">
    -- SELECT Query
</select>

<insert id="insertLoginCount">
    -- INSERT Query
</insert>
```

③ Item Read/Processor/Write를 구현한다.

```
public class NcsReader implements ItemReader<LoginNcs> {
    public LoginNcs read() {
        LoginNcs loginNcs = new LoginNcs();
        LoginCountService result = textCountService.selectLoginCount();
        return result;
    }
}
```

수행 tip

- 시스템간 연동시 개발된 배치 프로그램이 안정적이고 보안성이 확보되어 동작할 수 있으며, 개발환경으로 정의된 개발언어와 도구의 특성을 충분히 반영하여 생산성과 품질이 확보된 배치 프로그램을 개발하는 것이 중요하다.

4-2. 배치 프로그램 테스트

학습 목표

- 개발된 배치 프로그램을 테스트할 수 있는 테스트 케이스를 작성하고 단위 테스트를 수행하기 위한 테스트 조건을 명세화 할 수 있다.

필요 지식 /

① 스크립트 작성 시 유의사항

1. 환경변수값 명시

Cron이나 스케줄러에서 실행 시, 기존에 .profile이나 .cshrc 등에 적용되어 있는 쉘환경 변수를 읽어들이지 못할 수도 있기 때문에 실행스크립트에서 .profile, .cshrc등을 실행 시키거나 필요한 환경변수를 직접적으로 명시해야만 한다.

2. Background 실행

스크립트 내에서 복수의 실행을 동시에 동작시키고자 할 때, Ampersand(&)를 명령라인 마지막에 추가한다.

3. Background 실행 프로세스의 대기(wait)

복수의 Background 명령을 실행시키고 난 뒤, 스크립트를 즉시 종료하지 않고 모든 작업이 완료되었을 경우에 종료하도록 하려면 wait 명령어를 추가한다.

② 시스템 모니터링 방법(CPU, MEM, DISK 등)

1. 기본 명령어: sar, iostat, vmstat 등이 있다.

2. Text Based 시스템 모니터링 도구

(1) top(Popular)

```
last pid: 86494; load averages: 0.83, 0.65, 0.69 up 67+22:48:43 14:44:15
227 processes: 1 running, 224 sleeping, 2 zombie
CPU: 20.2% user, 0.0% nice, 6.5% system, 0.2% interrupt, 73.1% idle
Mem: 1657M Active, 1868M Inact, 273M Wired, 190M Cache, 112M Buf, 11M Free
Swap: 4500M Total, 249M Used, 4251M Free, 5% Inuse

  PID USERNAME  THR PRI NICE   SIZE    RES STATE  C  TIME  WCPU COMMAND
86460 www         1   4   0    150M    30204K accept 1  0:02 11.18% php-cgi
86458 www         1   4   0    150M    29912K accept 0  0:02  8.98% php-cgi
86463 pgsq        1   4   0    949M     99M sbwait 1  0:01  7.96% postgres
85885 www         1   4   0    150M    35204K accept 2  0:07  7.57% php-cgi
85274 www         1   4   0    149M    40868K sbwait 3  0:27  5.18% php-cgi
85267 www         1   4   0    151M    40044K sbwait 2  0:33  4.59% php-cgi
85884 www         1   4   0    150M    41584K accept 2  0:14  4.59% php-cgi
85887 pgsq        1   4   0    951M    128M sbwait 1  0:04  4.20% postgres
85886 pgsq        1   4   0    949M    161M sbwait 0  0:08  3.37% postgres
86459 pgsq        1   4   0    949M    75960K sbwait 2  0:01  3.37% postgres
85279 pgsq        1   4   0    950M    192M sbwait 2  0:14  2.39% postgres
85269 pgsq        1   4   0    950M    199M sbwait 1  0:19  2.20% postgres
85268 www         1   4   0    152M    44356K sbwait 2  0:32  1.17% php-cgi
85273 pgsq        1   4   0    950M    215M sbwait 0  0:19  1.17% postgres
97882 pgsq        1  44   0   26020K    6832K select 0 46:55  0.00% postgres
  892 root         1   4   0    3160K      8K -      2 13:33  0.00% nfsd
1796 root         1  44   0   19780K   13660K select 3 12:43  0.00% Xvfb
```

[그림 4-1] top 실행화면

(2) Topas(IBM AIX)

```
Topas Monitor for host: dczgdb1
Thu Jun 19 17:45:37 2014 Interval: 2

CPU User% Kern% Wait% Idle%
ALL 49.0 7.3 12.7 31.0

Network KBPS I-Pack O-Pack KB-In KB-Out
Total 57.1K 6347.0 5680.5 28.6K 28.4K

Disk Busy% KBPS TPS KB-Read KB-Writ
Total 100.0 219.0K 2677.0 211.3K 7908.8

FileSystem KBPS TPS KB-Read KB-Writ
Total 2.7K 867.5 2.3K 393.7

Name PID CPU% PgSp Owner
oracle 8585414 6.3 13.9 grid
oracle 11665410 6.2 14.5 grid
oracle 8782652 6.2 21.8 grid
oracle 9110390 6.2 15.0 grid
oracle 11862422 6.2 12.8 grid
oracle 5636898 0.0 14.5 grid
oracle 9044574 0.0 12.8 grid
oracle 7405702 0.0 12.8 grid
oracle 7995628 0.0 12.7 grid
oracle 9437708 0.0 11.5 grid

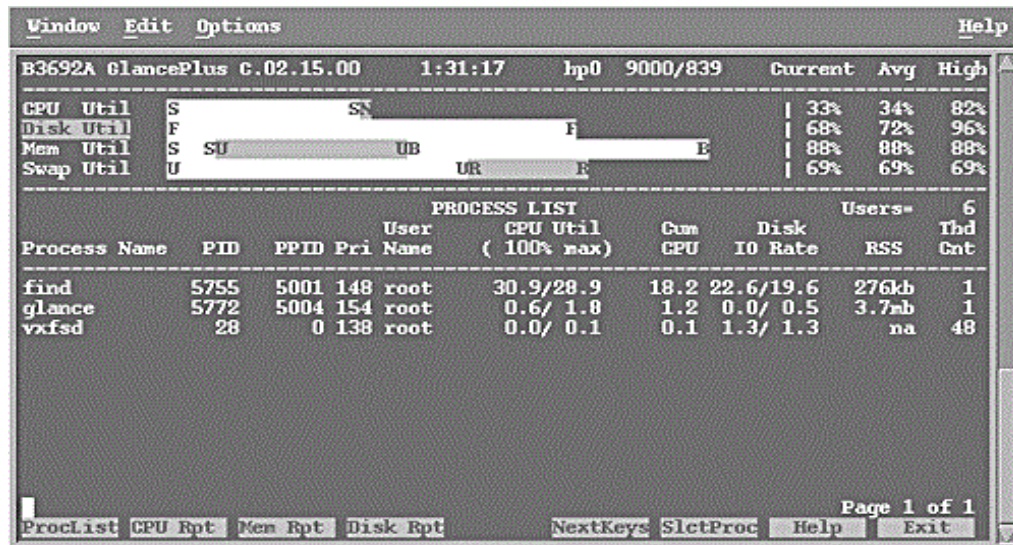
EVENTS/QUEUES FILE/TTY
Cswitch 18480 Readch 240.8M
Syscall 167.0K Writch 28.4M
Reads 7488 Rawin 0
Writes 5333 Ttyout 369
Forks 13 Igets 0
Execs 19 Namei 5166
Runqueue 5.5 Dirblk 0
Waitqueue 0.0

MEMORY
PAGING Real,MB 62720
Faults 7590 % Comp 84
Steals 0 % Noncomp 13
PgspIn 0 % Client 13
PgspOut 0
PageIn 0 PAGING SPACE
PageOut 77 Size,MB 65024
Sios 77 % Used 13
% Free 87

NFS (calls/sec)
SerV2 0 WPAR Activ 0
CliV2 0 WPAR Total 0
SerV3 0 Press: "h"-help
CliV3 0 "q"-quit
```

[그림 4-2] Topas 실행화면

(3) GlancePlus(HP-UX)



[그림 4-3] GlancePlus 실행화면

3. 기타 상용 모니터링 도구 등을 사용할 수 있다.

③ 배치 프로그램 테스트 시 유의사항

1. 소스코드 내에 SQL실행 전후 로깅하여야 한다.
2. 반복테스트를 위한 원복스크립트를 준비하여야 한다.

④ 정합성 검증 방법

1. 물리검증

- (1) Count 검증(건수 검증)
- (2) Sum 검증(합계 검증)
- (3) Physical R-I(Referencial Integrity) 검증

2. 논리검증

- (1) Cross 검증(교차 검증)
- (2) Logical R-I(Referencial Integrity) 검증

수행 내용 / 배치 프로그램 테스트 하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 도구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

- 테스트 케이스를 작성한다.

테스트케이스 ID	4.1	테스트케이스명	Login Count 집계
수행시스템	NCS	하위시스템	LOGIN
작성자	홍길동	작성일자	20XX-XX-XX
수행자	홍길동	수행일자	20XX-XX-XX

선행조건	사용자는 해당일자에 1회이상 로그인 해야 한다.
------	----------------------------

No	Action	예상응답	결과	비고
1	runLoginNcs.sh 실행	LoginNcs 배치애플리케이션이 실행된다.		
5	후행 확인			

후행확인	Login Count 집계테이블에 해당 사용자의 Login Count Update 여부를 확인
------	------------------------------------------------------

- 테스트 데이터를 작성한다.(예시)

테스트를 수행할 사용자 계정으로 Login을 반복 수행하거나, Login History 테이블에 정보를 Insert한다.

- ③ 테스트 스크립트를 실행한다.

```
runLoginNcs.sh [사용자계정] [해당일자]
```

- ④ 테스트 결과를 명세화한다.

목적테이블의 대상정보를 DB에서 확인한다.

사용자계정	접속일자	접속횟수
NCSUSER	20XX-XX-XX	27

- ⑤ 오류를 디버깅한다.

테스트 결과가 정상이 아닐 경우, 오류 유형을 분석하여 디버깅한다.

수행 tip

- 개발된 배치 프로그램이 제공해야 할 기능과 인터페이스를 테스트하기 위해 적절한 테스트케이스를 작성함으로써, 원활한 단위테스트가 수행될 수 있도록 준비하는 것이 중요하다.

학습 4 교수 · 학습 방법

교수 방법

- 애플리케이션 설계 명세를 기반으로 선정된 개발언어와 도구를 활용하여 개발하는 절차에 대해 정리하여 설명한다.
- 배치 프로그램 작성 시 연동되는 하위 시스템과 연동하는 방안에 대해 정리하여 설명한다.
- 배치 프로그램 작성 시 고려해야 하는 보안요건을 정의하여, 이를 반영하기 위한 절차와 기법에 대해 정리하여 설명한다.
- 배치 프로그램 테스트를 위한 테스트케이스 작성 절차와 기법에 대해 자료로 정리하여 제시하고, 이에 대해 상세히 설명한다.
- 배치 프로그램 단위테스트를 수행하기 위한 테스트 조건을 명세화하는 방안에 대해 제시하고, 상세히 설명한다.

학습 방법

- 애플리케이션 설계 명세를 기반으로 선정된 개발언어와 도구를 활용하여 개발하는 절차에 대해 학습한다.
- 배치 프로그램 작성 시 연동되는 하위 시스템과 연동하는 방안을 학습하고 실습한다.
- 배치 프로그램 작성 시 고려해야 하는 보안요건을 정의하여, 이를 반영하기 위한 절차와 기법에 대해 학습한다.
- 배치 프로그램 테스트를 위한 테스트케이스 작성 절차와 기법에 대해 학습하고 해당 절차와 기법에 따라 실습함으로써 실무에서 활용할 수 있도록 한다.
- 배치 프로그램 단위테스트를 수행하기 위한 테스트 조건을 명세화하는 방안에 대해 학습한다.

학습 4 평가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
배치 프로그램 작성	- 애플리케이션 설계를 기반으로 프로그래밍 언어와 도구를 활용하여 배치 프로그램 구현 기술에 부합하는 배치 프로그램을 개발할 수 있다.			
	- 목표 시스템을 구성하는 하위 시스템간의 연동 시, 안정적이고 안전하게 동작할 수 있는 배치 프로그램을 개발할 수 있다.			
	- 개발하고자 하는 목표 시스템의 잠재적 보안 취약성이 제거될 수 있도록 배치 프로그램을 개발할 수 있다.			
배치 프로그램 테스트	- 개발된 배치 프로그램을 테스트할 수 있는 테스트 케이스를 작성하고, 단위 테스트를 수행하기 위한 테스트 조건을 명세화 할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
배치 프로그램 작성	- 애플리케이션 설계서와 배치 프로그램 간 정합성			
	- 배치 프로그램 내의 보안요소 반영의 적정성			
배치 프로그램 테스트	- 배치 프로그램 테스트케이스와 테스트 조건 적정성			

- 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
배치 프로그램 작성	- 애플리케이션 설계서와 배치 프로그램 간 정합성			
	- 배치 프로그램 내의 보안요소 반영의 적정성			
배치 프로그램 테스트	- 배치 프로그램 테스트케이스와 테스트 조건 적정성			

피드백

1. 평가자 체크리스트

- 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.
- 애플리케이션 설계서에 따라 배치 프로그램이 작성되었는지를 확인하여, 미비사항을 보완할 수 있도록 미비내역을 정리하여 돌려준다.
- 배치 프로그램 내에 적용된 보안요소를 파악하고 사전에 정의된 보안요건을 만족하는지 검토하여 미비사항을 정리하여 돌려준다.
- 배치 프로그램 테스트케이스와 조건의 적정성을 파악하고 미비사항을 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 배치 프로그램 테스트를 위한 테스트케이스 적용기법에 대해 파악한 후, 프로그램 특성에 맞는 기법선정 방안과 테스트케이스 생성 방법에 대해 설명해 준다.
- 배치 프로그램 수행결과 사전에 정의된 보안요건을 만족하여 수행되는지를 점검하고 미비사항을 정리하여 돌려준다.
- 배치 프로그램 기능 검증을 위해 작성된 테스트케이스의 완전성을 검토하고 미비사항을 정리하여 돌려준다.

학습 1	개발환경 구축하기(LM2001020203_14v2.1)
학습 2	공통 모듈 구현하기(LM2001020203_14v2.2)
학습 3	서버 프로그램 구현하기(LM2001020203_14v2.3)
학습 4	배치 프로그램 구현하기(LM2001020203_14v2.4)
학습 5	개발자 단위 테스트하기 (LM2001020203_14v2.5)
학습 6	애플리케이션 성능 개선하기(LM2001020203_14v2.6)

5-1. 단위테스트 계획수립

학습 목표

- 구현한 응용 소프트웨어 단위가 설계 내용을 반영하는지 여부를 판단하기 위한 단위테스트의 표준, 절차, 기법 등을 정의할 수 있다.

필요 지식 /

① 소프트웨어 테스트

- 소프트웨어의 결함이 존재함을 보이는 과정
- IEEE 정의: 시스템이 정해진 요구를 만족하는지, 예상과 실제 결과가 어떤 차이를 보이는지 수동 또는 자동 방법을 동원하여 검사하고 평가하는 일련의 과정
 - Validation-유효성(우리가 필요했던 프로그램인가? 요구사항에 맞는 프로그램인가?
Are we building the right product?)
-> 소프트웨어가 명시된 요구사항을 만족하는지 확인한다.
 - Verification-검증(올바르게 제작되었는가? 개발방법론이 제대로 평가되고 진행되었는가? Are we building the product right?)
-> 개발 프로젝트 중 각 단계별로 고객의 요구사항이 반영되었는지 검사, 품질검토(위크스루, 인스펙션) 등의 기법이 있다.
- 소프트웨어 테스트의 종류
 - 화이트 박스 테스트 (코드기반 시험)
 - Statement Coverage
모든 코드가 한번은 실행되게 입력한다. (코드의 에러여부 파악)

(나) Decision Coverage(branch test)

코드의 흐름에서 모든 진행을 테스트. 예를 들어 if 구문이 2개라면 4개의 흐름(4 종류의 입력)을 만들어야 한다.

(다) Condition Coverage

- 1) 조건문이 있을 때 , 예를 들어 if 구문의 조건을 보고 조건에 맞는 input과 안 맞는 input을 넣어서 유입되는지 테스트한다.
- 2) if(조건상황들) 이라면 조건상황들이 true인 조건과, false인 조건을 찾아서 입력한다.

(라) Multiple Condition Coverage

- 1) 조건문의 true false에 모든 상황을 고려한다.
- 2) 조건이 if(A&B) 일때 4가지 입력이 발생(T,T T,F F,T F,F)한다.

(2) 블랙 박스 테스트 (Input, Output 기반의 시험)

(가) Syntax Testing

입력에 올바른 값과 올바르지 않는 값을 넣어서 테스트한다.

(나) Equivalent Partitioning

입력을 동등하게 쪼개고(예를 들어서 학점 프로그램이라면 0~70, 71~80, 81~90, 91~100으로 쪼갬다.), 각 영역의 대표값을 입력하는 테스트(45, 73, 87, 95 를 입력 해서 각각 F, C, B, A 가 나와야 한다)한다.

(다) Boundary Testing

입력값의 경계를 테스트한다. 예를들어 학점 프로그램이라면 A와 B의 경계인 89, 90, 91을 입력해본다.

(라) Decision Table

입력값의 종류를 결정한다.

수행 내용 / 단위테스트 계획수립 하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

① 테스트 배경 및 범위를 설정한다.

1. Use Case Modeling 산출물과 개발계획서를 참조하여 테스트의 배경과 범위를 간략하게 기술한다.
2. 테스트활동의 범위를 정의한다.
단위테스트는 개발된 각 클래스의 메서드가 정상적으로 구현되었는지를 확인하여 오류를 최소화하기 위한 작업이다.

② 테스트 수행에 필요한 표준 항목을 정의한다.

1. 모든 테스트에서 기록되어야 할 항목들을 정의하여 테스트 수행계획서 양식을 작성한다.
2. 테스트 수행계획서 항목 (예시)
 - (1) 프로젝트명
 - (2) 테스트ID
 - (3) 테스트명
 - (4) 테스트종류
 - (5) 시스템명
 - (6) Sub시스템명
 - (7) 테스트 수행계획서 작성자

- (8) 테스트 수행계획서 작성일자
- (9) 테스트 수행 담당자
- (10) 테스트 수행 일자

3. 단위테스트 수행계획서 (예시)

NCS 사용자 관리 프로젝트				
테스트케이스 ID	4.1	테스트케이스명	Login Count 집계	
수행시스템	NCS	하위시스템	LOGIN	
작성자	홍길동	작성일자	20XX-XX-XX	
수행자	홍길동	수행일자	20XX-XX-XX	
선행 조건 사용자는 해당일자에 1회이상 로그인 해야 한다.				
No	Action	예상응답	결과	비고
1	runLoginNcs.sh 실행	LoginNcs 배치애플리케이션이 실행된다.		
5	후행 확인			
후행확인 Login Count 집계테이블에 해당 사용자의 Login Count Update 여부를 확인				

③ 테스트 합격 기준을 정의한다.

1. 수행 동작에 대한 기능 합격 기준의 정의

- (1) 클래스에 구현된 각각의 메서드는 100% 의도한 대로 동작해야 한다.
- (2) 입력값에 대한 유효성 체크가 정상적으로 동작해야 한다.
- (3) 예외적인 데이터에 대한 처리가 동작해야 한다.

2. 실행성능에 대한 합격 기준의 정의

- (1) 메서드의 호출시점부터 반환(return)시점까지의 정상처리 시간이 목표시간 안에 완료되어야 한다.
- (2) 예외처리가 발생하는 경우도 목표시간 안에 완료되어야 한다.

3. 자원사용량에 대한 합격 기준의 정의

- (1) 실행시점(Running-Time)에서 시스템 자원의 규모에 맞게 적절한 자원만 사용되어야 한다.
- (2) 유헤시점(Idle-Time)에서는 자원사용량(특히 CPU)이 최소화되어야만 한다.

수행 tip

- 개발된 프로그램이 설계서 내용을 모두 반영할 수 있도록 단위테스트 표준, 절차, 기법을 정의하는 것이 중요하다.

5-2. 단위테스트 수행 및 검증

학습 목표

- 기능요구사항을 분석하여 단위테스트 계획을 수립하고, 단위 테스트 계획대로 단위 모듈/컴포넌트 별로 테스트를 수행할 수 있다.
- 단위 모듈/컴포넌트가 설계 내용을 만족하는지 여부를 계획한 단위 테스트 케이스에 따라 검증할 수 있다.

필요 지식 /

① 단위테스트의 개념

1. 단위 테스트는 시스템의 소스코드 로직들을 점검하는 단계로, 소스코드의 Class나 Method단위의 검증을 수행한다.
2. 단위테스트는 개발 단계에서 개발자 또는 개발팀 차원에서 직접 수행을 하게 된다. 자신이 코딩한 Method나 Class가 정상적으로 동작하는지를 테스트하는데, 주로 기능위주의 작동여부를 점검한다.

② 단위테스트의 유형

<표 5-1> 단위테스트 유형

유형	설명
In container Test	자바 애플리케이션에서는 주로 JUnit과 같은 xUnit 시리즈의 단위 테스트 프레임워크를 사용하는데, Tomcat이나 WebLogic과 같은 미들웨어 위에서 작동하는 애플리케이션의 경우 테스트가 반드시 해당 미들웨어 위에서 동작해야 한다.
Mock up	<p>단위 테스트는 소프트웨어 구성 요소의 각 컴포넌트를 독립된 환경에서 테스트하는 것이다.</p> <p>일반적으로 소프트웨어 컴포넌트는 혼자서 동작할 수 없고 다른 컴포넌트에 대해서 종속성(Dependency)을 가지고 있기 때문에 종속관계에 있는 컴포넌트가 완성되지 않거나 컴포넌트에 오류가 있으면 정상적으로 테스트를 진행할 수 없다.</p> <p>이런 경우에 가상의 테스트용 클래스와 메서드를 구현하는데, 이를 Mock up Class라고 한다.</p> <p>이 Mock up의 경우 Class와 Method의 정의는 있지만, 비즈니스 로직은 구현되어 있지 않고, input에 대해서 정해진 output 값만을 내는 형식이 된다.</p> <p>이러한 Mock-up Class는 직접 구현할 수도 있지만, EasyMock (http://www.easymock.org)과 같은 프레임워크를 사용하면 조금 더 쉽게 구현할 수 있다.</p>

유형	설명
Continuous Integration - CI	<p>구현이 끝난 코드는 단위 테스트를 통해서 검증을 하고, 소스 관리 시스템에 저장(Commit) 하게 된다.</p> <p>저장된 소스코드는 다른 사람이 작성한 소스 코드와 함께 다시 컴파일이 되서 모든 단위테스트를 다시 거치게 된다.</p> <p>이번 저장에 작성한 단위 테스트를 포함해서 이를 포함한 예전 단위 테스트까지 모두 같이 테스트하는 것을 Regression Test(회귀 테스트)라고 한다.</p> <p>회귀 테스트를 하는 이유는 예전 코드의 변경이 없더라도 새로운 코드가 기존 로직에 영향을 줄 수 있기 때문에, 새 코드가 기존 코드에 대해서 결함을 발생시키지 않았음을 검증하기 위해서 수행한다.</p> <p>이러한 일련의 과정은 보통 자동화된 환경에서 이루어지는 데 이를 Continuous Integration-CI라고 한다.</p>

수행 내용 / 단위테스트 수행 및 검증하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

① 테스트데이터를 생성한다.

1. 단위테스트에 필요한 데이터를 직접 정의하여 작성한다.

- (1) 장점: 필요한 데이터를 목적에 맞게 자유롭게 생성 가능하다.

(2) 단점: 수작업으로 작업함에 따른 시간과 공수가 많이 소요된다.

2. 테스트데이터 생성도구를 활용한다.

Databene Benerator와 같은 자동 테스트 데이터 생성 오픈소스 도구를 활용한다.

(1) Data 합성(Data Synthesization)

(2) 익명 Data 생성(Production Data Anonymization)

(3) XML 설정파일

<표 5-2> XML 설정파일 항목 및 설명

항목	설명
execute	attribute의 sql문을 직접 실행시키며, create,drop DDL문을 실행시켜 세팅작업을 하는데 사용한다.
setup	root태그, xsi, xsd설정, 그대로 유지
comment	주석태그
import	domains, platforms를 import
setting	properties 설정
include	property files loading
iterate	file에서 data들을 읽어서 consumer에 저장 data 생성하는 부분
generate	id: column name generator: data generation func지정 reference: FK임을 지정 targetType(Fk 관계가 있는대상 테이블) attribute: type별 특정 값 random 추출 consumer: 해당 데이터를 fixed length등의 포맷지정으로 외부로 export

② 단위 모듈 테스트를 실행한다.

테스트 프로그램 오른쪽 버튼 클릭 후 Run As... 를 이용하여 수행한다.

③ 테스트 결과를 기록한다.

1. 테스트 결과를 확인한다.

2. 단위테스트 수행계획서에 결과 및 필요사항을 기록한다.

수행 tip

- 수립된 단위 테스트 계획에 따라 단위 테스트를 수행하기 위한 테스트케이스를 검증하고, 이를 통해 단위 테스트를 수행하는 과정을 이해하는 것이 중요하다.

5-3. 단위테스트 결함 관리

학습 목표

- 단위 테스트 결과, 발견된 결함과 이슈를 식별하고, 단위 테스트 결과 분석을 통하여 테스트의 충분성 여부를 검증할 수 있다.
- 단위 테스트 결과, 발견된 결함에 대한 개선의 시스템 반영 여부를 검증하고, 필요할 경우 시정조치를 실시할 수 있다.

필요 지식 /

① SW결함 분석기법

1. 결함처리 프로세스

<표 5-3> 결함처리 프로세스

프로세스	설명
Report Defect	테스트팀에서 발견된 결함은 결함 관리 시스템 (Defect Management System)에 기록된다.
Yank Defect	개발팀에서는 자신의 모듈에서 발생한 Defect를 꺼내 가지고 온다.
Assign Defect	개발팀 리더는 Defect를 개발팀의 일정과 리소스, 그리고 담당에 따라서 개발자에게 배치한다.
Fix Defect	Defect를 할당받은 개발자는 담당 테스트 엔지니어와 함께 재현과 추적 등을 통해서 Defect를 수정한다.
Confirm Fix & Add Test Case	Defect가 수정되었으면, Defect를 검증할 수 있는 단위 Test Case를 보강하여 Fix된 코드와 함께 소스 관리 시스템에 저장한다.
Change defect status to "Resolved"	Defect Management System에 해당 Defect를 해결됨(Resolved) 상태로 바꾸고, 테스트 엔지니어에게 다시 배당한다.
Confirm Test	테스트 엔지니어는 해결된 Defect를 다시 테스트하여 문제가 없는지 검증한다.
Close the Defect	문제가 없을 경우에는 해당 Defect를 Close한다. 만약 문제가 해결되지 않았으면 다시 Defect Management System을 통해서 해당 개발자에게 다시 Assign하고 4번 단계(Fix Defect)부터 다시 반복하여 해결한다

2. 결함 리포팅

테스트 중 발견된 결함은 다음과 같은 항목에 의해 저장하여 관리한다.

<표 5-4> 결함 관리

항목	설명
Number (번호)	결함의 고유 번호이다.
Title (제목)	결함의 내용을 간단하게 한줄로 기록한다.
Description (설명)	결함에 대한 자세한 내용을 서술한다. 구체적으로 어떤 결함인지, 그리고 결함을 재현하는 절차와 분석 내용들을 서술한다.
Module (모듈)	전체 시스템 중에서 결함이 발견된 컴포넌트나 모듈명을 명시한다.
Version & Fixed Version (발견 버전과 수정 버전)	Version은 결함이 발견된 버전, Fixed Version은 해당 결함이 해결된 버전이다. 결함은 버전에 따라서 다르게 발생할 수 있기 때문에 반드시 이 항목이 정의되어야 한다.
Servirity , Priority (시급도와 우선순위)	시간적으로 빨리 처리해야 하는 경우 시급도가 높은 것을 의미하고, 결함의 우선순위는 위중도(위험도)를 의미한다. 결함의 우선순위가 높을수록 중결함이 되고, 낮을수록 경결함이 된다. 시급도가 높은 결함중 우선순위가 높은 결함을 우선적으로 처리한다. 시급도와 우선순위는 각각 5단계 정도로 나눠서 정의하는 것이 바람직하며, 일반적인 결함은 중간 단계인 3단계 정도로 하며, 시급도와 우선순위가 가장 높은 결함의 경우에는 최우선적으로 처리해야 하기 때문에 긴급하게 개발팀과 테스트 자원을 투입해야 한다.
Status (결함 처리 상태)	현재 결함이 처리되고 있는 상태를 기록한다.
Fixed code (코드 수정 내용)	결함을 수정하였을 경우에는 수정 내용을 기록해야 한다. 결함 내용 수정은 소스 코드 수정인 경우가 많은데, 이때는 소스 코드 수정 내용을 기록한다. 별도로 소스 코드 내용을 기록하는 것보다는 소스코드 관리 시스템(SCM)에 수정코드를 반영한 버전(Revision Number)을 기록해 놓으면 된다.
Attachment (첨부 파일)	결함을 기록할때는 첨부 파일을 사용하는데, 이 첨부 파일은 결함 발생 당시의 로그나 데이터 (CPU 사용률, 네트워크 상황), 결함을 추적하는 데 사용된 코드나 샘플, 결함이 해결되었을 경우 패치 같은 것을 첨부한다.

3. 결함 상태의 정의

결함은 아래에서 정의한 상태로 진행된다.

<표 5-5> 결함 상태

항목	설명
New	테스트 과정에서 결함이 발견되어 결함 관리 시스템에 등록된 상태이다.
Opened	등록된 결함이 개발팀에게 넘어갔을 때의 상태를 Opened라고 한다. 결함이 발생한 모듈을 개발한 개발팀에게 결함이 전달된 상태로, 아직 담당자는 정해지지 않은 상태이다.
Assigned	결함을 수정할 개발자에게 결함이 할당된 상태이다.
NMI (Need More Information)	“추가 정보가 필요함” 상태로, 개발팀에서 결함의 내용을 검토했을 때, 결함의 원인과 정확한 증상들을 확인할 수 없을 때, 추가적인 로그나 재현절차를 테스트 팀에 다시 요구 하는 상태이다. 개발팀은 자료가 부족한 결함에 대해서 NMI 상태로 바꾼 후에, 결함을 보고한 테스트 엔지니어에게 다시 배당한다.
In Progress	개발자가 결함을 확인하고, 수정 작업 중인 단계이다
Postponed	결함 처리 과정에서 결함의 중요도가 낮거나 다른 결함 수정 일정 또는 개발일정에 따라서 우선순위 조정이 필요할 때, 결함의 처리를 미뤄놓는 (지연시키는) 상태이다.
Resolved	결함이 해결된 상태이다. 해결은 되었으나, 개발팀 차원에서 결함이 해결된 것이고, 아직 테스트팀으로 부터 결함에 대한 재 테스트를 통한 확인을 받지 못한 상태이다. 결함이 Resolved 상태로 바뀌면 개발팀은 확인테스트를 요청하기 위해서 테스트 팀에 다시 해당 결함을 Assign한다.
Closed	테스트팀에 의해 결함의 수정 내용이 확인되고 제품의 소스코드에 반영된 상태로, 결함 처리의 최종 단계이다.
Attachment (첨부 파일)	결함을 기록할 때는 첨부 파일을 사용하는데, 이 첨부 파일은 결함 발생 당시의 로그나 데이터(CPU 사용률, 네트워크 상황), 결함을 추적하는데 사용된 코드나 샘플, 결함이 해결되었을 경우 패치 같은 것을 첨부한다.

② 디버깅 기법

1. 파괴식 검사 기법

- (1) 소스코드를 직접 수정해서 필요한 정보를 얻는 방법
- (2) 디버깅 로그 추가, 특정 동작 수행 코드 삽입 등

2. 비파괴식 검사 기법

- (1) 디버깅 도구(디버거)를 이용한 방법이 대표적이다.(2-2 공통 모듈 테스트의 ② 디버깅 도구의 사용법 참조)
- (2) System Call 추적 Utility의 사용(Strace, Truss 등)
- (3) Library Call 추적 Utility의 사용 (Ltrace 등)
- (4) File의 사용현황 및 사용자 정보 확인 (Lsof, Fuser 등)

수행 내용 / 단위테스트 결함 관리하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 도구)

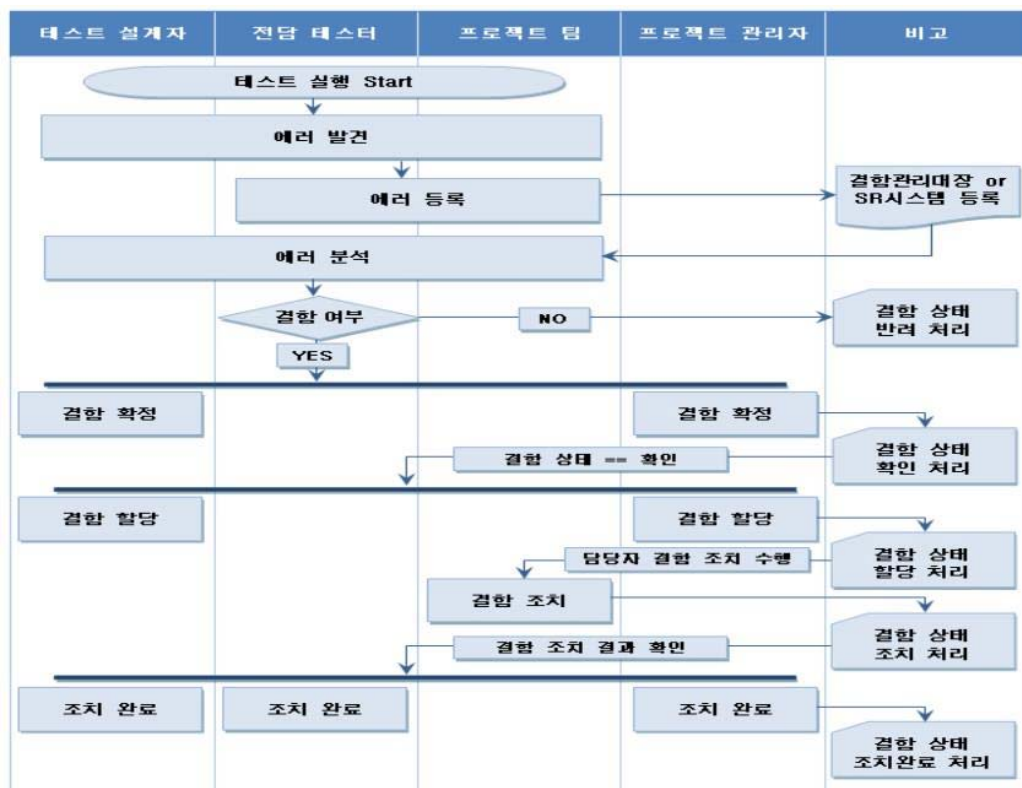
- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

〈결함 관리활동 작업흐름도〉



출처: http://www.oss.kr/oss_repository6/67274(공개SW포털)

[그림 5-1] 결함 관리 활동 작업 흐름도

① 에러를 발견한다.

요구사항 분석, 설계, 테스트 수행 중에 에러가 발견될 경우, 전문 테스터와 프로젝트 팀과 의논한다.

② 발견된 에러를 등록한다.

결함 관리대장에 발견된 에러를 등록한다.

③ 등록된 에러를 분석한다.

등록된 에러를 분석하여 단순 에러인지 실제 결함인지를 분석한다.

④ 결함을 확정한다.

등록된 내용이 결함으로 확정될 경우, 결함을 결함확정 상태로 설정한다.

⑤ 담당자별 결함을 할당한다.

결함을 해결할 담당자를 지정하고, 해당 결함을 할당한다. 이때 결함은 할당 상태가 된다.

⑥ 결함을 조치한다.

결함에 대한 수정 활동을 수행하고, 수정활동이 완료된 경우 결함을 결함조치 상태로 설정한다.

⑦ 결함 조치내용을 검토하여 승인한다.

수정이 완료된 결함에 대한 확인 테스트를 수행하여, 정상적으로 결함이 수정되었는지를 확인한다. 정상적으로 조치 완료된 경우 결함을 조치완료 상태로 설정한다.

수행 tip

- 단위테스트 결과 분석을 통해 테스트의 충분성을 검증하고, 테스트 중 발견된 결함과 이슈를 식별하여 조치한 후, 해당 결함이 시스템에 반영되었는지 여부를 검증하는 것이 중요하다.

학습 5 교수 · 학습 방법

교수 방법

- 단위테스트 목적과 이를 위한 표준에 대해 정리하여 설명한다.
- 단위테스트 목적을 달성하기 위한 절차와 기법에 대해 정리하여 설명한다.
- 단위테스트 계획대로 단위 모듈별 테스트 수행을 위한 테스트케이스 작성기법을 제시한 후 설명한다.
- 단위테스트 결과 발견된 결함과 이슈를 식별하는 절차에 대해 설명한다.
- 단위테스트 결과 발견된 결함에 대한 사후 조치(반영여부, 시정조치)방안에 대해 정리하여 설명한다.

학습 방법

- 단위테스트 목적과 이를 위한 표준에 대해 이해하여 적용할 수 있도록 학습한다.
- 단위테스트 목적을 달성하기 위한 절차와 기법에 대해 이해하고 적용할 수 있도록 실습한다.
- 단위테스트 계획대로 단위 모듈별 테스트 수행을 위한 테스트케이스 작성기법을 학습하여 적용할 수 있도록 실습한다.
- 단위테스트 결과 발견된 결함과 이슈를 식별하는 절차에 대해 이해한다.
- 단위테스트 결과 발견된 결함에 대한 사후 조치(반영여부, 시정조치)방안에 대해 이해한다.

학습 5 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
단위테스트 계획 수립	- 구현한 응용 소프트웨어 단위가 설계 내용을 반영하는지 여부를 판단하기 위한 단위테스트의 표준, 절차, 기법 등을 정의할 수 있다.			
단위테스트 수행 및 검증	- 기능요구사항을 분석하여 단위테스트 계획을 수립하고, 단위 테스트 계획대로 단위 모듈/컴포넌트 별로 테스트를 수행할 수 있다. - 단위 모듈/컴포넌트가 설계 내용을 만족하는지 여부를 계획한 단위 테스트 케이스에 따라 검증할 수 있다.			
단위테스트 결함 관리	- 단위 테스트 결과 발견된 결함과 이슈를 식별하고, 단위 테스트 결과 분석을 통하여 테스트의 충분성 여부를 검증할 수 있다. - 단위 테스트 결과 발견된 결함에 대한 개선의 시스템 반영 여부를 검증하고, 필요할 경우 시정조치를 실시할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
단위테스트 계획 수립	- 단위테스트 표준, 절차, 기법의 적정성			
단위테스트 수행 및 검증	- 단위테스트 계획의 적정성 - 단위 모듈/컴포넌트 테스트 수행가능 여부 - 단위 테스트케이스에 따른 검증 여부			
단위테스트 결함 관리	- 단위 테스트 결과 분석을 통한 결함 식별 및 시정조치의 적정성			

• 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
단위테스트 계획 수립	- 단위테스트 표준, 절차, 기법의 적정성			
단위테스트 수행 및 검증	- 단위테스트 계획의 적정성 - 단위 모듈/컴포넌트 테스트 수행가능 여부 - 단위 테스트케이스에 따른 검증 여부			
단위테스트 결함 관리	- 단위 테스트 결과 분석을 통한 결함 식별 및 시정조치의 적정성			

피드백

1. 평가자 체크리스트

- 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.
- 단위 테스트 관련 표준과 절차, 기법을 파악해 보고 미비사항을 정리하여 돌려준다.
- 단위 테스트 계획내의 항목 누락여부를 파악한 후 정리하여 돌려준다.
- 단위테스트 케이스가 결함 식별을 위해 충분한지 검토하여 미비사항을 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 단위테스트 표준과 절차에 대해 검토하여 미비사항을 표시하여 돌려준다.
- 발생 결함에 대한 식별과정과 시정조치 절차와 기록관리 문서 내용에 대해 파악한 후 미비사항을 정리하여 돌려준다.

학습 1	개발환경 구축하기(LM2001020203_14v2.1)
학습 2	공통 모듈 구축하기(LM2001020203_14v2.2)
학습 3	서버 프로그램 구현하기(LM2001020203_14v2.3)
학습 4	배치 프로그램 구현하기(LM2001020203_14v2.4)
학습 5	개발자 단위 테스트하기(LM2001020203_14v2.5)
학습 6	애플리케이션 성능 개선하기 (LM2001020203_14v2.6)

6-1. 애플리케이션 성능측정과 개선

학습 목표

- 실 데이터를 기반으로 테스트를 수행하여 애플리케이션의 성능을 확인하고 목표 성능이 충족되도록 개선할 수 있다.
- 애플리케이션 성능을 개선하기 위해, 기 정의된 프로그래밍 언어 표준 및 가이드라인에 따른 코드 품질 메트릭을 이해하고 적용할 수 있다.
- 애플리케이션 성능을 개선하기 위해, 프로그래밍 언어와 이의 표준에 대한 이해를 바탕으로 소스코드에 내재된 품질 수준을 분석하기 위한 도구를 활용할 수 있다.

필요 지식 /

① Performance Engineering

1. 시스템의 목표 성능(응답 시간과 동시 접속자수)을 정의하고, 이를 달성하기 위해서, 시스템의 구조를 반복적으로 개선하는 작업을 의미한다.
2. 성능 목표의 정의부터, 최적의 성능을 내기 위한 디자인 및 구현과 같은 개발 초기의 설계 부분과 개발 후의 운영단계의 모니터링까지 모든 과정을 포함한다.
 - (1) 분석 단계
 - (가) 초기 요구 사항 분석 및 시스템 기획 단계에서는 성능에 대한 목표를 정해야 한다.
 - (나) 목표 응답시간은 어떻게 되는지, 시스템을 사용할 총 사용자수와 동시에 시스템을 사용하는 동시접속자 수가 어떻게 되는지와 같은 성능 목표를 정의한다.
 - (다) 또한 고려해야 하는 사항 중의 하나는 성능 모델이다. 시스템에 부하가 어떤 패턴으로 들어오는지를 정의할 필요가 있다.
 - (2) 설계 단계
 - (가) 목표 성능과 용량을 달성할 수 있는 규모의 시스템으로 설계를 진행한다.

- (나) 성능 관점에서 시스템 디자인은 항상 Peak Time(최대 성능)에 맞춰서 디자인이 된다. 최대 성능을 기반으로 전체 시스템이 받아낼 수 있는 용량과 응답 시간을 고려해야 한다.
 - (다) 특히 성능과 용량은 애플리케이션 디자인뿐만 아니라 Technology Selection에도 많은 영향을 받는다. 어떤 하드웨어를 사용할 것인지, 어떤 미들웨어나 프레임워크를 사용할 것인지에 따라 용량과 성능의 차이가 많이 발생하기 때문에, 디자인 단계부터 성능과 용량을 감안해서 시스템을 설계해야 한다.
 - (라) 하드웨어 관점에서는 예전에는 성능 모델을 산정한 후에 Peak Time 기준(최대 성능 요구)으로 시스템을 설계하고, 하드웨어를 구매하였으나, 근래에는 클라우드를 이용하여 필요시에만 하드웨어를 탄력적으로 사용하는 Auto Scale Out 모델을 많이 사용한다.
 - (마) 기업 내부의 업무처럼(예를 들어 이메일), 부하가 일정하고 예측이 가능한 경우에는 Fixed된 사이즈의 하드웨어를 사용하도록 설계하고, 출시 이벤트 행사 사이트와 같이 부하가 갑자기 몰리는 시스템의 경우 클라우드를 고려해보는 것도 권장할 만하다.
 - (바) 또한 빠른 응답 시간이 필요할 경우 SSD 디스크를 사용하거나, RAID 구성도 5보다는 1+0 등을 고려하는 등, 성능 모델에 따라서 적절한 하드웨어 선정과 구성 설계가 필요하다.
 - (사) 미들웨어나 프레임워크 관점에서도 정의된 성능 모델에 따라 적절한 제품군과 설계 구조를 채택해야 한다. 100,000 사용자 정도의 시스템 규모에서는 RDBMS를 사용해도 성능이나 용량상에 문제가 없다. 그러나 50,000,000 사용자 정도를 지원해야 하는 시스템의 경우 그냥 RDBMS를 사용할 수 없다. Sharding이나, NoSQL과 같은 다른 차원의 접근이 필요하다.
- (3) 개발단계
- (가) 개발 단계는 개발프로세스 챕터에서 설명하였듯이, Risk가 높은 부분과 아키텍처에 관련되는 부분, 난이도가 높은 부분, 핵심 기능등을 개발 초기의 스프린트에서 개발한다.
 - (나) 초기 스프린트가 끝나고 릴리즈가 되어서 성능 테스트가 가능한 QA나 스테이징 환경으로 시스템이 이전되면 Performance Engineering 역량을 이 단계에 집중하여, 시스템의 아키텍처와 모듈들이 성능 목표를 달성할 수 있는지 지속적으로 테스트하고 튜닝을 수행한다.
 - (다) 초기 단계에 성능 목표의 달성 가능 여부가 판단되어야 아키텍처 변경이 가능하고, 주요 성능 이슈들을 초반에 발견해야 발견된 성능 문제들에 대해서는 같은

문제가 발생하지 않도록 디자인 가이드나 코딩 가이드를 개발자들에게 배포하여 성능에 대한 위험도를 줄일 수 있다.

(4) 최종 테스트 단계

앞의 단계에서 성능과 용량을 고려해서 설계가 되었고, 개발 초기 단계에서 성능과 용량 부분의 검증을 제대로 하였다면, 최종 테스트 단계에서는 개발된 최종 시스템에 대한 성능과 용량 부분의 측정과 미세 튜닝(애플리케이션의 병목을 찾아서 부분적으로 수정하거나, 하드웨어나 미들웨어의 Configuration하는 수준)을 하는 정도로 마무리가 되어야 한다.

(5) 운영 단계

마지막으로 시스템이 운영 단계로 넘어가게 되면 테스트 시에 발견되지 않은 성능적인 문제가 있을 수 있기 때문에, 모니터링 도구를 사용하여 지속적으로 성능을 모니터링 하고, 성능에 문제가 있는 부분을 지속적으로 수정해야 한다.

② 프로그램 성능향상 기법의 이해

1. 시스템호출(System Call)의 사용빈도 감소

(1) read/write 등의 System Call에는 많은 System Resource가 사용된다. System Resource는 read/write의 대상 총량이 아닌 호출빈도에 영향을 받으므로, 즉시성 데이터가 아닌 경우 메모리버퍼 등을 구성하여 임계값 이상이 되었을 때에만 호출되도록 구성한다.

(2) 동적자원할당(Memory Allocation, Object Creation 등) 또한 System에 많은 오버헤드를 유발하는 요소로서, 가급적 프로그램 기동 시에 Pre-Allocated된 Object를 재사용하도록 구성하여 Run-Time 시에 동적으로 생성되는 경우가 적도록 구현한다.

2. I/O집중에 따른 경쟁(Competition) 감소

공통 모듈의 일반적인 구성요소 중 “채번(Get Number)” 기능의 구현 시, 채번정보영역에 대한 동시접근으로 인하여 경쟁(Competition)이 발생할 수 있다. 이러한 경쟁은 프로그램 실행상의 지연을 초래하므로, 경쟁 자체를 회피하거나(Sequence Object 등의 사용을 통해) 분산(경쟁대상을 세분화)시키는 방법을 사용하여 구현한다.

3. Garbage Collection의 회피

Object는 생성시점에도 많은 오버헤드가 발생되지만, 생성된 Object를 회수할때는 더 많은 오버헤드가 발생된다. 이를 위해 Static Method사용, 임시 Object 생성금지, Primitive Datatype 등을 사용하여 구현한다.

4. Object 재사용

Object의 재사용을 위해 Pool Management, Method를 사용한 Object 초기화, Static

Instance Variable의 사용에 의한 단일 클래스 인스턴스를 구현한다.

5. Method 호출 감소(Inline Method 사용)

Method Inline에 의한 Method 호출을 감소시킨다.

예)

```
public class MethodTest {
    int counter = 0;

    public void method1() {
        for (int i = 0; i < 1000; i++) {
            addCount();
            System.out.println("counter=" + counter);
        }
    }

    public int addCount() {
        counter = counter + 1;
        return counter;
    }

    public static void main(String[] args) {
        InlineMe im = new InlineMe();
        im.method1();
    }
}
```

아래와 같이 addCount()를 수정하면, 컴파일 단계에서 Inline Method로 변환되어서 실제로 Method를 호출하지 않으면서 같은 결과를 반환하게 된다.

```
public int addCount() {
    counter = counter + 1;
}
```

수행 내용 / 애플리케이션 성능측정과 개선하기

재료 · 자료

- 프로젝트 아키텍처 가이드라인

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 개발전용 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

① 성능테스트를 위한 시스템을 분석한다.

1. 업무프로세스 및 애플리케이션을 분석한다.
 - (1) 주요 업무프로세스 분석
 - (2) 해당 업무 특성 파악 => 인터뷰, Survey, Observation 등의 활동
 - (3) 테스트 대상 업무 선정 => 응답속도 및 사용빈도 등을 분석
2. 실행로그를 분석한다.
 - (1) 사용자 현황 분석 => End-User, 운영자, 개발자 등
 - (2) 이용패턴 분석
 - (3) 시스템 자원 통계 분석
 - (4) 에러로그 분석
3. 부하량을 분석한다.
 - (1) 주요 업무 부하량 분석 => Peak-Time 최대부하 발생 시
 - (2) 부하모델 작성

② 성능테스트 방안을 설계한다.

1. 현행 시스템 목표 성능치를 선정한다.
 - (1) 현행 시스템 분석을 통한 성능 기준 수립

- (2) 동시 사용자수 증가에 따른 응답시간 기준 수립
- (3) 웹 애플리케이션 시스템, C/S 애플리케이션 시스템에 따른 별도 기준 수립
- (4) Peak-Time 시의 성능 기준 반영
- 2. 테스트케이스를 작성한다.
 - (1) 사용자 이용패턴과 유사한 케이스 선정
 - (2) 시스템 아키텍처 검증 고려
- 3. 테스트 시나리오를 작성한다.
 - (1) 대량 사용자
 - (2) 시스템 자원을 많이 사용하는 애플리케이션
 - (3) 수행 횟수 및 가상 사용자 수 선정
- 4. 테스트 스크립트를 작성한다.
 - (1) 반복 및 재사용을 고려한 스크립트 작성
 - (2) 재사용을 위한 초기화 스크립트 작성
 - (3) 테스트 스크립트 정확성 검증
- 5. 테스트 데이터를 생성한다.
 - (1) 애플리케이션 수행에 사용되는 실 데이터 생성
 - (2) 테스트 데이터 적합성 검증
 - (3) 통상적으로 '전일자 수행내역' 을 추출하여 데이터를 생성함

③ 성능테스트를 실행한다.

- 1. 테스트 환경을 설정한다.
 - (1) 테스트 장비 설치 및 환경 구성
 - (2) 실행권한 부여 및 시스템 모니터링에 필요한 관리자 권한 부여
- 2. 테스트를 수행한다.
 - (1) 테스트 시나리오별 테스트 수행
 - (2) 테스트 결과 분석
 - (3) 테스트 시나리오 및 시스템 설정 변경
 - (4) 변경된 테스트 시나리오별 테스트 수행
- 3. 모니터링 데이터를 수집한다.
 - (1) Server Side: CPU, MEMORY, DISK I/O, Concurrent Connection, N/W, Process, Thread

(2) Client Side: Response Time(MIN/MAX/AVG 등)

④ 성능테스트 결과를 평가한다.

1. 테스트 결과 분석

- (1) Server Side 모니터링 데이터 분석(Process별, Thread 별, Application 유형별)
- (2) Client Side 모니터링 데이터 분석(화면별, 이벤트 별)

2. 성능개선 항목 도출

- (1) 성능개선 요소 도출
- (2) 추가적인 테스트 필요성 분석

3. 성능테스트 결과서 작성

- (1) 성능테스트 결과 기록
- (2) 성능관점에서의 문제점 기록
- (3) 시스템 자원사용 관점에서의 문제점 기록
- (4) 시스템 사용자 및 사용량 증가에 따른 예상 기록
- (5) 성능개선 방안 제시

수행 tip

- 실 데이터를 활용하여 테스트를 수행하는 과정에서 애플리케이션 성능을 분석하여 당초 계획한 품질목표에 부합할 수 있도록 품질메트릭과 관련 도구를 활용할 수 있는 것이 중요하다.

학습 6 교수 · 학습 방법

교수 방법

- 개발된 애플리케이션이 달성해야 할 품질목표에 대한 자료를 정리하여 설명한다.
- 애플리케이션이 달성해야 할 품질목표를 위해 성능개선 영역 범위가 어디까지 인지 제시하여 설명한다.
- 성능개선을 위한 개발표준과 가이드라인에 대한 필요성과 기대효과를 이해할 수 있도록 설명한다.
- 성능개선을 위한 개발표준 제정대상 항목에 어떤 것들이 있으며, 각 항목에서 제정하는 표준내용이 무엇인지 정리하여 설명한다.
- 작성된 소스코드의 품질수준을 측정하기 위한 정적분석 도구의 유형과 도구 적용을 위해 고려해야 할 사항을 정리하여 설명한다.

학습 방법

- 개발된 애플리케이션이 달성해야 할 품질목표를 이해한다.
- 애플리케이션이 달성해야 할 품질목표를 위한 성능개선 영역 범위를 이해하고, 성능개선을 위한 절차와 기법을 습득한다.
- 성능개선을 위한 개발표준과 가이드라인에 대한 필요성과 기대효과를 이해한다.
- 성능개선을 위한 개발표준 제정대상 항목에 어떤 것들이 있으며, 각 항목이 어떻게 소스코드에 반영되는지를 이해하고, 학습한다.
- 작성된 소스코드의 품질수준을 측정하기 위한 정적분석 도구의 유형과 도구 적용방안에 대해 이해하여, 도구를 통해 품질수준을 측정하는 과정을 학습한다.

학습 6 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
애플리케이션 성능측정과 개선	- 실 데이터를 기반으로 테스트를 수행하여 애플리케이션의 성능을 확인하고, 목표 성능이 충족되도록 개선할 수 있다.			
	- 애플리케이션 성능을 개선하기 위해, 기 정의된 프로그래밍 언어 표준 및 가이드라인에 따른 코드 품질 메트릭을 이해하고 적용할 수 있다.			
	- 애플리케이션 성능을 개선하기 위해, 프로그래밍 언어와 이의 표준에 대한 이해를 바탕으로 소스코드에 내재된 품질 수준을 분석하기 위한 도구를 활용할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
애플리케이션 성능측정과 개선	- 애플리케이션 성능확인방안 이해			
	- 애플리케이션 성능개선을 위한 코드 품질메트릭 적용 적정성			
	- 애플리케이션 소스코드 품질 수준 분석도구 활용 적정성			

• 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
애플리케이션 성능측정과 개선	- 애플리케이션 성능확인방안 이해			
	- 애플리케이션 성능개선을 위한 코드 품질메트릭 적용 적정성			
	- 애플리케이션 소스코드 품질 수준 분석도구 활용 적정성			

피드백

1. 평가자 체크리스트

- 실습 과정에서 평가한 후에 개선해야 할 사항 등을 정리하여 돌려준다.
- 애플리케이션 성능 확인방안에 대한 내용을 확인하여 미비사항을 정리하여 돌려준다.
- 애플리케이션 소스코드 성능개선을 위한 코드 품질메트릭을 확인하고, 보완사항을 정리하여 돌려준다.
- 소스코드 품질확보를 위한 정적분석도구 선정의 타당성과 활용범위를 파악하여 미비사항이나 보완사항을 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 애플리케이션 성능 확보를 위한 접근 방안의 타당성에 대해 검토하여 돌려준다
- 애플리케이션 소스코드 코드 품질메트릭 적용결과를 보고 추가적으로 포함해야 할 내용에 대해 설명한다.
- 애플리케이션 소스코드 품질수준 분석을 위한 분석도구 활용결과를 보고, 결과 활용방안 미비사항을 정리하여 돌려준다.



- 전병선(2015). 『JAVA EE 아키텍트 핸드북』, 와우북스.
- 결함 관리활동 작업 흐름도, http://www.oss.kr/oss_repository6/67274(공개SW포털)에서 2015.10.10. 인출.
- 통합 개발환경의 종류, 위키디피아 <https://ko.wikipedia.org/>에서 2015.10.10. 인출.
- 테스트 활동에 따른 도구 분류, 공개SW포털 <http://www.oss.kr>에서 2015.10.10. 인출.
- Eclipse 다운받기, <http://www.eclipse.org/downloads>에서 2015.10.10. 인출.
- Ant 다운받기, <https://www.swbank.kr/helper/tool/toolMain.do>에서 2015.10.10. 인출.
- JUnit 다운받기, <https://www.swbank.kr/helper/tool/toolMain.do>에서 2015.10.10. 인출.
- Git 다운받기, <http://git-scm.com/download/win>에서 2015.10.10. 인출.
- Spring Batch, <http://springsource.tistory.com/79>에서 2015.10.10. 인출.

NCS 학습모듈 개발진

(대표집필자)

강석진(이비스툼)

(집필진)

김보운(이화여자대학교)

김홍진(LG CNS)

유은희

장현섭((주)커리텍)

주선태(T3Q)

진권기(이비스툼)

최재준

(검토진)

김승현(경희대학교)

엄기영(우리에프아이에스)

장온순(한국IT컨설팅)

조상욱(세종대학교)

조성호(삼성카드)

(개발기관)

최기원(한국소프트웨어기술진흥협회)

이두현(한국소프트웨어기술진흥협회)

(연구기관)

옥준필(한국직업능력개발원)

김상진(한국직업능력개발원)

김성남(한국직업능력개발원)

김지영(한국직업능력개발원)

문한나(한국직업능력개발원)

홍서희(한국직업능력개발원)

*표시는 NCS 개발진임

※ 본 학습모듈은 자격기본법 시행령 제8조 국가직무능력표준의 활용에 의거하여 개발하였으며
저작권법 25조에 따라 관리됩니다.

※ 본 학습모듈은 <http://www.ncs.go.kr>에서 확인 및 다운로드할 수 있습니다.



www.ncs.go.kr