Report

도메인 분석 및 SW 설계

- Elaboration phase (iteration 3, 4) -

교수님 : 이정태 교수님

제출 일 : 2020.07.12

조 이름 : 소프트

이름	*학번*	*분반*
안창희(팀장)	**201723272**	**B**
장유림	**201821328**	**B**
정우택	**201520943**	**B**
박재영	**201421122**	**D**
이연주	**201521485**	**B**

2. 목차

목차

3. 그림 목차

[Revision History]

버전	*일자*	*설명*	*저자*
Elaboration iteration 1 초안	2020.05.28	최초 버전	소프트
Elaboration iteration 1 ver.1	2020.06.01	1) Domain Model 수정 2) Logical Architecture 수정	소프트
Elaboration iteration 1 ver.2	2020.06.02	SSD, OC 수정	소프트
Elaboration iteration 2 초안	2020.06.10	Vision 수정	소프트
Elaboration iteration 3 초안	2020.07.02	SSD attribute 수정, OC 수정	소프트
Elaboration iteration 3 ver.1	2020.07.03	Domain 모델 수정	소프트
Elaboration iteration 3 ver.2	2020.07.04	Use Case realization 설계	소프트
Elaboration iteration 3 ver.3	2020.07.05	Code 설계	소프트

4.Vision

4.1 Control information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Vision
Document owner	소프트조
Document identification no	4
Document status	Final

4.2 Revision History

버전	*일자*	*설명*	*저자*
Vision 초안	2020.04.22	초기미팅, 각 파트에 대한 방향점 협의 후 파트배정	전원
Vision v1.0	2020.4.27	각 파트 초기작성	전원
Vision v1.1	2020.05.02	1.4 stakeholder descriptions 에 플랫폼 관리 시스템 추가	장유림
Vision v1.2	2020.05.04	introduction 수정, business opportunity 자료 추가 및 수정, alternatives and competition 수정	이연주
Vision v1.3	2020.05.20	stakeholder 수정	정우택
Vision v1.4	2020.06.01	Product 수정	안창희
Vision v1.5	2020.06.02	Product 수정	이연주
Vision v.1.6	2020.06.10	Introduction 수정, stakeholder 추가	전원

4.3 Introduction

Vision 문서는 온라인 신발 쇼핑몰에서 판매자들이 효율적인 상품관리를 할 수 있도록 돕는 시스템인 'ShoeRack'의 개요를 명세한 문서이다. 따라서 온라인 쇼핑몰 시스템의 사용자 및 개발자를 비롯한 관계자들을 독자로 한다. 'ShoeRack' 시스템을 도입하고자 하는 관계자들은 'ShoeRack'의 개발 배경 및 목적, 이해관계자의 종류와 참여 목적, 기존 타 시스템들과의 차이점, 제약사항 등을 파악할 수 있다.

4.4 Positioning

4.4.1 Business Opportunity

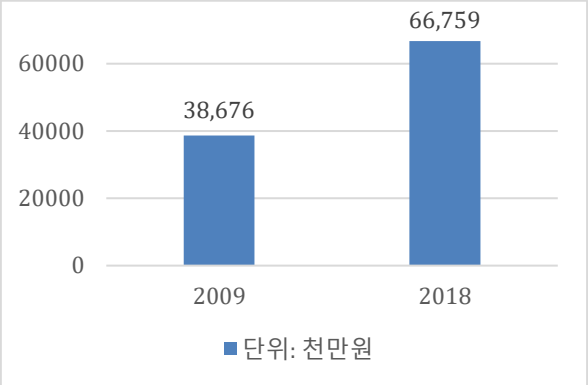


그림 1 국내 신발 소매시장

국내 신발소매시장은 전체 패션시장에서 2009 년 3 조 8676 억원 규모였다. 2017 년부터 패션업체들이 의류시장 불황을 극복하기 위해 시즌에 구애 없는 신발라인을 강화하면서, **2009 년 대비 2018 년에는 18% 이상 성장하여 6 조원대**까지 커졌다.

		2018년		2019년		2019년			
						전년비		구성비	
2018/2019년		온라인	모바일	온라인	모바일	온라인	모바일	온라인	모바일
전체 온라인 합계		1,137,297	690,950	1,345,830	867,005	18.3	25.5	100	100
패션	패션 전체	376,485	229,769	426,879	272,055	13.4	18.4	31.7	31.4
	의복	138,991	81,193	148,705	91,390	7.0	12.6	11.0	10.5
	신발	18,932	12,733	21,018	14,473	11.0	13.7	1.6	1.7
	가방	22,397	16,588	26,158	20,239	16.8	22.0	1.9	2.3
	패션용품 및 액세서리	23,669	15,641	25,780	17,748	8.9	13.5	1.9	2.0
	스포츠 레저용품	37,985	21,761	42,310	24,906	11.4	14.5	3.1	2.9
	화장품	98,404	55,137	122,986	73,114	25.0	32.6	9.1	8.4
	아동, 유아용품	36,107	26,717	39,921	30,185	10.6	13.0	3.0	3.5

그림 2 국내 온라인패션시장 거래액 및 구성비

한편 국내 신발시장 성장하는 추세에 따라, 온라인 신발 시장도 성장하고 있다. 2019 년 국내 온라인 거래 규모는 전년대비 18.3% 증가한 134 조 5 천 830 억원이다. 국내 온라인 패션시장 규모는 42 조 6 천 879 억원으로 전년 대비 13.4% 성장했으며, 이 중 **신발 온라인 거래액은 전년대비 11.0% 증가한 1 조 4 천 473 억원**을 기록했다. 앞으로 온라인 신발 시장은 오프라인에서 소비자들이 온라인에서 구매하는 비중이 높아지는 트렌드가 성장을 견인할 것으로 보인다. 매년 늘어가는 신규 온라인 전환 신발 구매자들을 사로잡는 서비스가 시장의 주도권을 차지할 것임에는 의심의 여지가 없어 보인다.

4.4.2 Problem statement

problem statement 요약

The problem of	신발 소매 판매자의 오프라인 시장 진출에 적지 않은 초기 자본이 필요하고, 신발 소매 판매자들은 오프라인 시장 진출에 부담을 느낀다.
Affects	신발 소매 판매자의 온라인 시장 진출 요구가 증가하였다.
the impact of which is	온라인 시장에서 신발 소매 업체의 파편화로 인하여, 신발 소매 업체에 대해 잘 알지 못하는 소비자들의 신발 소매 업체로의 접근이 더욱 어려워졌다. 많은 신발 소매 업체들은 온라인 시장에서 큰 수익을 내기 힘들어진다.
a successful solution would be	여러 신발 소매 업체들이 자신의 상품을 등록하여 판매할 수 있는 통합 온라인 쇼핑몰이 이러한 문제의 개선책이 될 수 있다.

의류 온라인 쇼핑몰을 시작하는 사업자는 90% 이상이 옷이 좋아서 시작하고, 그 시작 또한 1 인 또는 2 인 동업이 대부분을 이룬다.[1] 이들은 상품생산량이 많지 않고 적은 매출에도 브랜드 존속에 큰 영향을 받는다. 많은 상품판매자가 자사의 신발을 판매하고 싶어하지만 오프라인 신발매장에 진출하는 데에는 적지 않은 초기 자본을 지출해야 한다. 그래서 온라인 시장에 진출하여 신발을 판매하고 싶어하는 소규모 브랜드가 증가하였다. 이는 결국 온라인 시장에서 신발

[1] 인터넷기사 '지그재그 신규 의류 쇼핑몰 시장 진입의 어려움', 2020

소매업체들의 파편화로 이어졌다. 소규모 신발 브랜드에 대해 잘 알지 못하는 소비자들이 소규모 브랜드에 대해 더욱 접근하기 어려워진 것이다. 신발 소매 업자가 온라인 시장에 진출하여 수익을 내는 일도 힘들어졌다. 이러한 문제 상황에 대해 여러 신발 소매 업체들이 자신의 상품을 등록하여 판매할 수 있는 통합 온라인 쇼핑몰이 개선책으로 제시될 수 있다.

4.4.3 Product Position Statement

Product Position Statement, Alternatives and Competition 요약

For	소규모 신발 브랜드의 판매자
Who	온라인 시장에서 소규모 신발 브랜드의 파편화로 인하여, 온라인 시장에서 수익을 내지 못하고 있는 판매자를 대상으로 한다.
The '상품 관리 시스템'	상품 관리 시스템은, 소규모 신발 브랜드들을 통합한 온라인 쇼핑몰을 구성하는 하위 시스템이다. 상품 관리 시스템은 소규모 신발 브랜드의 판매자가 통합 온라인 쇼핑몰에 자신의 상품을 쉽고 간편하게 등록하고 관리할 수 있게 한다.
That	소규모 신발 브랜드의 판매자는 통합된 온라인 쇼핑몰을 이용함으로써, 소규모 신발 브랜드에 대해 잘 알지 못하는 소비자들에게 효과적으로 자신의 브랜드를 알릴 수 있다. 이는 온라인 시장에서 소규모 신발 브랜드의 파편화 문제에 대한 개선책이 될 수 있다. 소규모 신발 브랜드의 판매자는 개인 온라인 쇼핑몰을 운영했을 때보다, 통합 온라인 쇼핑몰을 이용했을 때, 큰 수익을 얻게 되고 개인 브랜드의 크기를 키울 수 있다.
Unlike	신발 시장에서, 통합 온라인 쇼핑몰을 운영하고 있는 회사는 없다. 하지만 그 범위를 의류 시장으로 확장했을 때, 지그재그와 하이버가 경쟁사가 될 수 있다. 이 둘은 소비자들에게 저평가되고 적게 노출되어있는 브랜드, 즉 소규모의 의류 온라인 쇼핑몰을 통합한 플랫폼이다.

Our product	소규모 의류 판매자가 지그재그와 하이버를 이용하여 온라인에서 의류를 판매하고 싶다면, 자체적인 온라인 쇼핑몰을 이미 보유하고 있어야 한다. 개인 온라인 쇼핑몰을 이미 운영하고 있는 업체만 해당 플랫폼에 자신의 상품을 등록하여 판매할 수 있는 것이다. 하지만 소프트조에서는 스토어팜 형식으로 소규모 신발 판매자들이 개인적으로 운영하는 온라인 쇼핑몰을 가지고 있지 않더라도, 자신의 상품을 등록하고 판매할 수 있도록 할 것이다.

소프트 조는 온라인 시장에서 소규모 신발 브랜드의 파편화로 인하여, 온라인 시장에서 수익을 내지 못하고 있는 판매자를 대상으로 하는 통합 온라인 쇼핑몰의 하위 시스템인 '상품 관리 시스템'을 설계할 것이다. 상품 관리 시스템은 소규모 신발 브랜드의 판매자가 통합 온라인 쇼핑몰에 자신의 상품을 쉽고 간편하게 등록하고 관리할 수 있게 한다. 소규모 신발 브랜드의 판매자는 통합된 온라인 쇼핑몰을 이용함으로써, 소규모 신발 브랜드에 대해 잘 알지 못하는 소비자들에게 효과적으로 자신의 브랜드를 알릴 수 있다. 이는 온라인 시장에서 소규모 신발 브랜드의 파편화 문제에 대한 개선책이 될 수 있다. 소규모 신발 브랜드의 판매자는 개인 온라인 쇼핑몰을 운영했을 때보다, 통합 온라인 쇼핑몰을 이용했을 때, 큰 수익을 얻게 되고 개인 브랜드의 크기를 키울 수 있다.

4.4.4 Alternatives and Competition

신발시장만을 타겟으로 소규모 판매를 통합한 온라인 신발 쇼핑몰 플랫폼은 아직 없었다. 신발시장에서 의류시장으로 범주를 넓혀보았다. 소프트 조가 지향하는 온라인 신발 쇼핑몰 플랫폼과 유사한 가장 대표적인 서비스로는 지그재그와 하이버가 있었다. 지그재그와 하이버는 소비자에게 저평가되고 적게 노출 되어있는 브랜드, 즉 소규모 의류 판매자 온라인 쇼핑몰의 상품들을 통합하여 구매자들에게 보여주는 플랫폼이다. 의류 온라인 쇼핑몰 플랫폼인 이 두 서비스는 판매자들이 기본적으로 자체 홈페이지를 보유하고 있어야하는 한계점이 있었다. 소프트 조에서는 신발 판매자들이 개인 소유 온라인 쇼핑몰이 없어도 온라인 쇼핑몰에서 직접 상품을 등록하고 관리할 수 있는 스토어 팜 형식으로 이들과 차별화된 온라인 신발 쇼핑몰 플랫폼을 제공하고자 한다.

4.4.5 Stakeholder Descriptions

아래 그림은 상품 관리 시스템의 stakeholder 를 요약한 그림이다. 내부 stakeholder 는 시스템 안에 본인이 할 업무가 존재하는 인원들이고, 외부 stakeholder 는 시스템을 지원하거나, 사회적 책임을 지게

하는 인원이다.

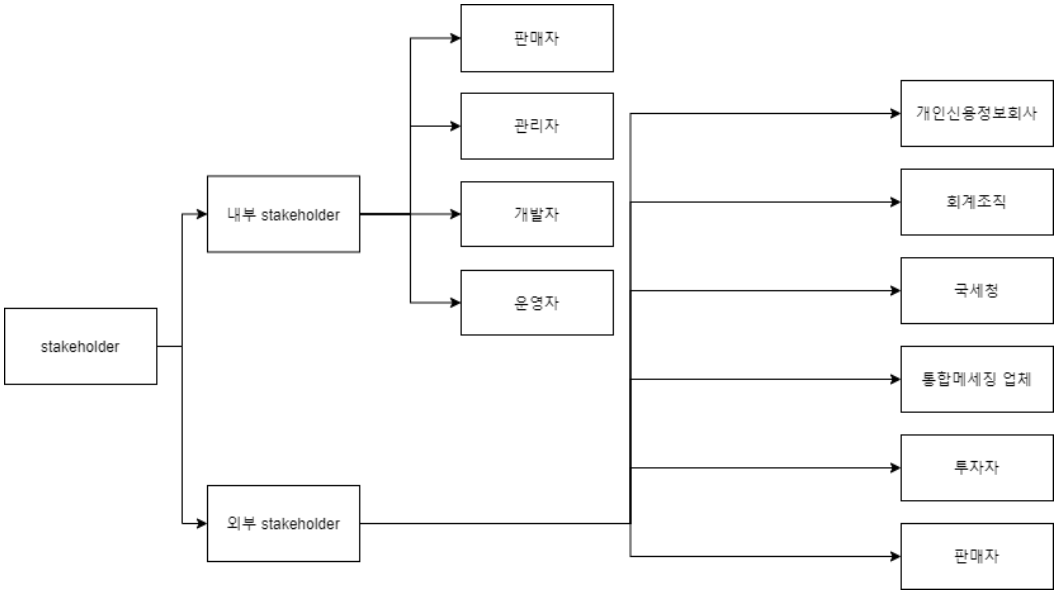


그림 3 Stakeholder 요약

Stakeholder 상세 설명

Name	***Description***	***Responsibilities***
판매자	신발을 팔기 위해 쇼핑몰을 이용하는 업체	시스템에 상품을 등록하여 판매하고 수익을 창출한다.
구매자	판매자가 등록한 상품을 구매하는 사람.	등록된 상품을 열람하고 구매한다.
관리자	시스템을 기술적 문제를 관리 감독하는 사람	시스템 로그를 보며 전반적인 시스템을 관제하고 유지 보수 및 관리한다.
운영자	온라인쇼핑몰의 행정적인 운영을 관리 감독하는 사람.	사용자의 불만을 해소하고 판매자들의 상품 등록, 카테고리 등록을 관리/감독한다.
개발자	시스템 개발을 담당하는	시스템에 들어가는 각종 기능들을 설계 및

	사람.	구현한다.
투자자	시스템 개발에 필요한 자금을 투자하는 사람.	시스템 개발에 필요한 각종 자금을 투자한다.
개인신용정보회사	판매자 및 구매자의 실명 정보를 확인하는 업체.	판매자의 가입과정에서 실명 확인 시 실제 본인 여부의 확인해준다.
회계 조직	시스템 내에서 일어나는 모든 거래를 기록하는 사람.	시스템 내에 통용되는 자금을 관리하고 이를 바탕으로 판매자들의 매출 정보를 각 판매자들에게 알려준다.
국세청	판매자가 승인된 사업자인지 확인하는 조직.	사업자 등록번호를 통해 사업자등록상태를 알려준다.
통합 메시징 업체	시스템 내의 주요 이벤트를 주요 이해관계자들에게 알려주는 조직.	다채널 메시지 통합관리 서비스를 제공하여 판매자들에게 상품 정보와 관련된 주요 변동사항을, 구매자들에게는 구매 관련 정보를 메일을 대신 발송해준다.

4.4.6 Detailed Descriptions of inner stakeholder

다음은 시스템에서 주요한 역할을 하는, 자세한 설명이 필요한 stakeholder 에 대한 설명이다.

- **판매자**

Type	Casual users
Success Criteria	본인이 원한 상품관리 시스템이 만들어지는 것이다. 이 시스템을 사용하면서 상품판매자는 더 많은 노출효과와 상품관리를 더 편하게 할 수 있다.

Involvement	판매자는 시스템의 user 로 직접 본인의 requirement 를 개발자에게 말한다.
Deliverables	프로젝트 문서와 시제품, 결론적으로 시스템을 요구한다.
Comments/ Issues	판매자는 운영자와 끊임없는 소통을 해야한다.

- **관리자**

Type	Suppliers
Success Criteria	상품관리 시스템이 에러가 날 경우 신속하게 시스템을 복구하여 판매자들이 시스템 사용에 애로사항이 없어야한다.
Involvement	완성된 시스템의 오류체크와 시스템 관리를 직접한다.
Deliverables	소프트웨어 개발자가 만든 시스템을 요구한다.
Comments/ Issues	문제를 해결하기 위해 항시 상주해 있어야 하며, 많은 오류를 처리하는 데에 있어 기술적인 부족함이 나타날 경우 큰 문제가 된다.

- **개발자**

Type	Developers
Success Criteria	판매자의 requirement 에 맞춰 시스템을 만드는데, 정해진 기간안에 완벽한 제품을 제공해야한다.
Involvement	직접 프로그램 코드를 작성한다.
Deliverables	-

Comments/ Issues	비용적, 시간적 문제 프로젝트 진행 시 나타날 수 있다.

- **운영자**

Type	Operator
Success Criteria	판매자와 구매자 사이에서 일어나는 갈등 등을 중재하며, 판매자가 내부규약에 벗어나는 행동(예를 들어, 판매자가 타인의 저작권을 침해하는 상품을 등록하는 경우)을 하는 일이 없도록 감독해야 한다.
Involvement	시스템의 행정적인 운영에 참여한다.
Deliverables	판매자의 상품 등록 활동, 구매자의 구매 활동 과정에서 나타나는 애로사항 및 변동사항을 기반으로 운영한다.
Comments/ Issues	애로사항에 대한 대처가 향후 시스템에 대한 사용자 평가에 큰 영향을 미치게 된다. 매뉴얼에 따라 일관성 있는 대처를 보여야 할 책임이 요구된다.

4.5 Product Description

신발가게를 운영하는 판매자가 신발 상품을 판매하고 재고를 채우기 위해서 상품에 대한 통합적인 관리가 필요하다. 판매자는 상품관리시스템에서 어떤 카테고리로 분류하여 보여줄 것인가, 상품에 대한 어떤 품목을 어떻게 관리할 것인가, 재고수량은 언제 채워야 하는가 등 오프라인에서 상품 관리 전반 업무를 수행할 수 있다. 판매자가 언제 어디서든지 편리하게 자신의 상품을 관리할 수 있어야 한다. 따라서 PC 또는 스마트 기기에 따라 달라지는 웹 페이지 화면에 대해서 고려하여 제작할 예정이다.

4.5.1 Context Diagram

온라인쇼핑몰을 판매자 입장에서 바라볼 때, 상품관리시스템과 관련된 actor 와 상품관리시스템과 연계되는 외부시스템을 Context Diagram 에 나타냈다.

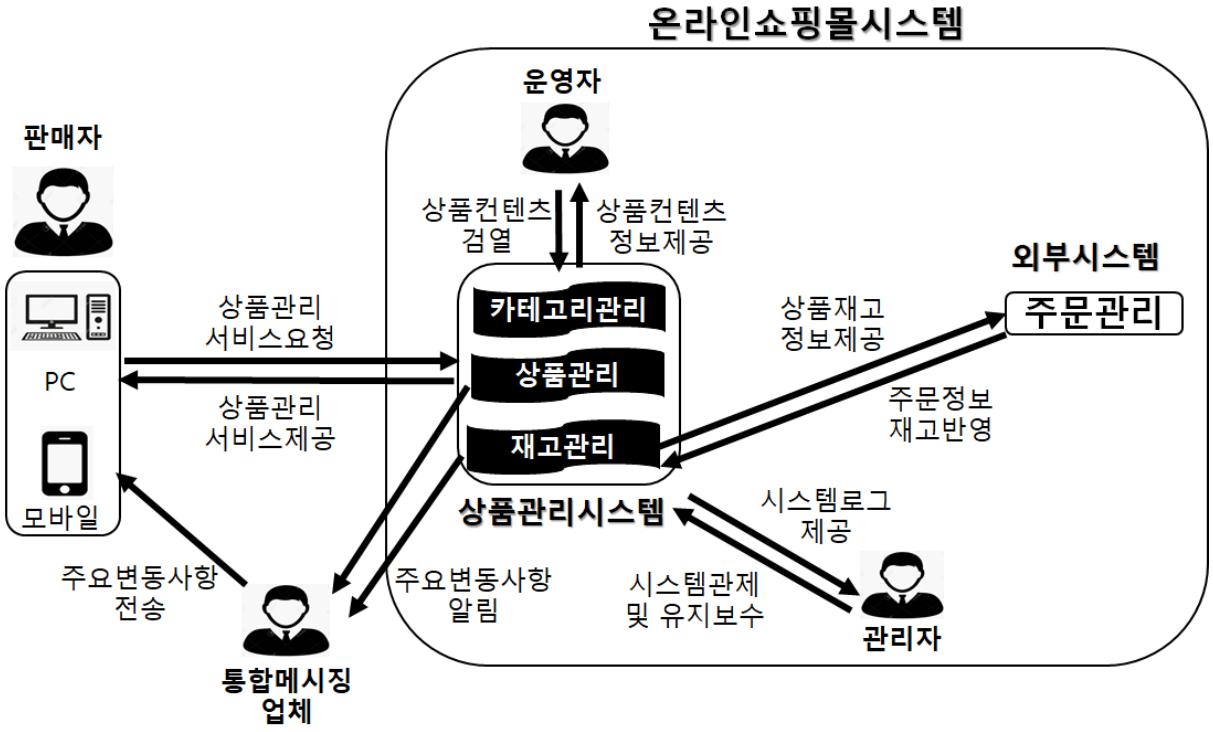


그림 4 Context Diagram

4.5.2 System Features

[Feature 1. 분류관리]

그림 5 분류 예

온라인 쇼핑몰 화면 상단에 노출할 카테고리를 관리할 수 있다. 대분류 하위에 중분류, 중분류 하위에는 소분류로 세분화된다. **대분류**는 **여성, 남성**으로 고정되어 있다. 또한 각 대분류에 대한 중분류는 **운동화, 구두, 샌들/슬리퍼, 시즌 슈즈**로 주어진다. 중분류에 대한 소분류는 판매자가 새로 추가할 수 있다. 적절하지 않은 소분류를 등록했을 시 플랫폼관리자가 반려한다.

Feature 1-1 소분류 관리: 대분류와 중분류를 선택하여 소분류를 추가, 삭제할 수 있다.

Feature 1-2 소분류 이력: 소분류에 대한 관리내역을 볼 수 있다.

[Feature 2. 재고관리]

상품별로 색상, 사이즈, 포장여부 등 옵션(이하 품목)이 다르다. 그 품목별로 재고 수량이 다르기 때문에 판매자는 품목별로 재고관리를 할 수 있다.

그림 6 재고관리 예

Feature 2-1 재고내역 열람: 판매상품의 품목별로 재고현황을 확인할 수 있다.

Feature 2-2 재고 입고: 품목별로 입고되는 수량을 추가할 수 있다.

Feature 2-3 진열 여부 선택: 품목별로 진열상태를 결정할 수 있다.

[Feature 3. 상품관리]

Feature 3-1 상품등록: 대/중/소분류를 지정하고 상품정보를 입력하여 팔고 싶은 신발을 추가할 수 있다.

Feature 3-2 상품 상세보기: 상품목록에서 볼 수 없는 상품에 대한 상세한 정보를 볼 수 있다. 예를 들면 상품에 대한 이미지가 있을 수 있고, 주문관리시스템에 있는 배송정보가 있다.

Feature 3-3 상품조회: 등록된 모든 상품에 대한 정보를 볼 수 있다.

그림 7 상품관리 예

5. Business Rules

5.1 Control Information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Business Rules
Document owner	소프트조
Document identification no	5
Document status	Final

5.2 Revision History

버전	*일자*	*설명*	*저자*
Business Rules 초안	2020.04.26	최초 버전	안창희
Business Rules v1.0	2020.05.02	Rule6 추가	안창희

5.3 Rule List

ID	*Rule*	*Changeability*	*Source*
RULE1	구매자에게 상품의 검색 및 선택기능을 제공해야됨	온라인 쇼핑몰은 구매를 원하는 이용자가 구매신청을 함에 있어서 재화 등의 검색 및 선택을 알기 쉽게 제공하여야 한다.	온라인 쇼핑몰 표준약관
RULE2	필요한 최소한의 정보를 수집해야함	온라인 쇼핑몰은 이용자의 개인정보 수집시 서비스제공을 위하여 필요한 범위에서 최소한의 개인정보를 수집한다.	온라인 쇼핑몰 표준약관

RULE3	최신의 정보를 제공해야함	온라인 쇼핑몰은 최신의 '상품정보'를 유지해야 하며, 등록된 상품정보가 기재오류, 누락 등으로 사실과 다를 경우 바로 상품 정보를 수정 또는 변경하여야한다	종합쇼핑몰 운영규정
RULE4	허위 내용이 등록되면 안됨	이용자는 신청 또는 변경시 허위 내용의 등록 행위를 하여서는 안된다.	온라인 쇼핑몰 표준약관
RULE5	지적 재산권을 침해 하면 안됨	온라인 쇼핑몰을 제외한 기타 제 3 자의 저작권 등 지적재산권에 대한 침해하는 행위를 하면 안된다.	온라인 쇼핑몰 표준약관
RULE6	등록된 사업자만 판매자로 받아들여아함	새로운 브랜드의 입점절차에서는 사업자 등록증과 통신판매 신고증, 사업자통장, 대표자 인감증명의 서류를 통해 검증절차를 진행해야한다.	국내 온라인 쇼핑몰 입점절차

6. Use-Case Model

6.1 Control information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Use-Case Model
Document owner	소프트조
Document identification no	6
Document status	Final

6.2 Revision History

버전	***일자***	***설명***	***저자***
Use-Case Model 초안	2020.04.22	초기미팅, 각 파트에 대한 방향 점 협의 후 파트배정	안창희
Use-Case Model v1.0	2020.4.25	회의내용 바탕으로 초기작성	전원
Use-Case Model v1.1	2020.4.27	1 차수정	전원
Use-Case Model v1.2	2020.5.3	다이어그램 변경 후 새로 작성	안창희, 정우택
Use-Case Model v1.3	2020.5.25	Use case 메인 시나리오 수정	안창희, 정우택
Use-Case Model v1.4	2020.6.1	Use case 대안 시나리오 수정	정우택
Use-Case Model v1.5	2020.6.10	분석 방법 및 중간 결과 및 Use case 선정	전원
Use-Case Model v1.6	2020.7.02	manage inventory 수정	정우택
Use-Case Model v1.7	2020.7.03	각 Use case 시나리오	전원

6.3 Use Case Diagram

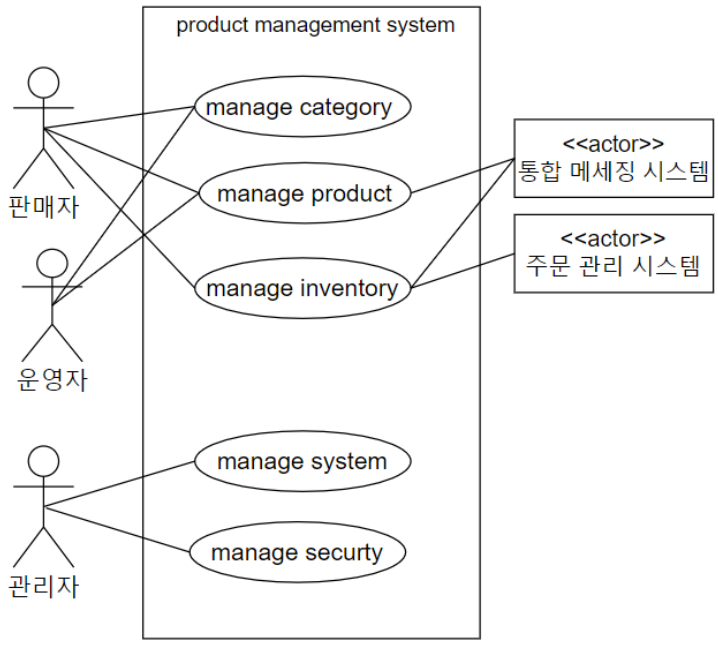


그림 8 상품관리시스템의 Use Case Diagram

6.4 분석 방법 및 중간 결과

유스케이스 모델링의 관점은 액터와 액터의 목적을 발견하고 가치 있는 결과를 내는 해결책을 만드는데에 있다. 주요 액터는 시스템 경계에 따라서 달라지므로, 가장 먼저 시스템 경계를 확실히 설정해야 할 것이다. 'ShoeRack' 상품 관리 시스템의 유스케이스는 다음과 같은 과정으로 선정하였다.

상품 관리 시스템의 유스케이스 선정 단계 요약

[1 단계]	시스템의 경계를 설정한다. 외부 주요 액터 또는 지원 액터 등 시스템 외부에 있는 것들을 정의함으로써 시스템 경계를 명확히 설정하였다.
[2 단계]	시스템 서비스를 사용하여 목적을 수행하는 주요 액터를 식별한 후, Actor-Goal List 를 작성하여 각각의 주요 액터에 대한 목적을 도출한다.
[3 단계]	액터의 목적을 충족시키는 유스케이스를 목적에 따라 명명하고 정의한다.

- **[1 단계] 시스템 경계 설정**

본 프로젝트에서 목표 시스템은 상품 관리 시스템이다. Vision 의 Product Description 에서 정의한 상품 관리 시스템의 피처로 분류 관리, 재고 관리, 상품 관리가 있다. 각 피처에 대한 관련성을 기준으로 상품 관리 시스템의 경계를 명확히 하고자 한다.

즉, 통합 메시징 시스템, 주문관리 시스템은 시스템 외부에 존재하며 상품 관리 시스템의 분류 관리, 재고 관리, 상품 관리 피처의 동작을 도울 뿐 특정한 목적을 가지고 상품 관리 시스템을 이용하지 않는다. 목표 시스템은 위 시스템(통합 메시징 시스템, 주문관리 시스템)에 대한 완전한 기능을 포함하지 않는다. 반면, 판매자, 운영자, 관리자는 특정한 목적을 가지고 상품 관리 시스템에 직접적으로 참여하는 주요 액터이다. 유스케이스는 주요 액터에 대한 시스템 이용 목적을 추출한 후 정의할 것이다.

- **[2 단계] 주요 액터와 목적 식별**

상품 관리 시스템의 주요 목적인 '상품 관리'에 참여하는 액터들은 상품을 등록/삭제하는 판매자, 판매자가 상품 등록/삭제하는 일련의 단계를 검수하는 운영자, 판매자와 운영자의 행동을 원활하게 할 수 있도록 시스템의 운영을 총괄하는 관리자가 있다. 1 단계에서 제시한 상품 관리 시스템 피처를 사용하여 특정 목적을 수행하는 주요 액터와 제시된 액터의 목적을 요약한 Actor-Goal List 는 다음과 같다.

Actor-Goal List

액터	목적
판매자	-정의되어 있는 분류의 현황을 조회하고 등록/삭제하기를 원한다. -등록되어있는 상품현황을 조회하고 등록/삭제 하기를 원한다. -등록된 상품별 재고현황을 조회하고 수정하기를 원한다.
운영자	-정의되어 있는 분류의 현황을 조회하고 변경사항을 검수하기를 원한다. -등록되어있는 상품현황을 조회하고 변경사항을 검수하기를 원한다.
관리자	-시스템의 오류를 파악하고 해결하기를 원한다. -데이터를 외부로부터 보호하기를 원한다.

- **[3 단계] 주요 액터의 목적을 충족시키는 유스케이스 정의**

추출한 주요 액터와 목적을 바탕으로, 다음과 같은 기준으로 유스케이스를 선정하였다.

선정한 유스케이스

액터와 목적	유스케이스	적합성 근거
-[판매자] 정의되어 있는 분류의 현황을 조회하기를 원한다. -[판매자] 분류를 등록/삭제하기를 원한다. -[운영자] 변경사항을 검수하기를 원한다.	분류관리	EBP 테스트 - 판매자가 상품 분류를 어떻게 했느냐에 판매자의 개성이 드러나며, 이러한 개성이 매출에 영향을 준다.
-[판매자] 등록되어있는 상품현황을 조회하기를 원한다. -[판매자] 상품을 등록/삭제 하기를 원한다. -[운영자] 변경사항을 검수하기를 원한다.	상품관리	EBP 테스트 - 판매자의 상품 등록에 따라 구매자가 어떠한 상품을 보고 구매할지를 결정할 수 있다.
-[판매자] 등록된 상품별 재고현황을 조회하고 수정하기를 원한다.	재고관리	EBP 테스트 - 판매자가 상품을 원활하게 공급하기 위해 상품의 재고관리는 필수적이다.
-[관리자] 시스템의 오류를 파악하고 해결하기를 원한다.	시스템관리	보스 테스트 - 상품관리가 원활하게 진행되어야 주요 액터들이 원활하게 시스템을 이용할 수 있으며, 이는 시스템 운영에 필수적이다.
-[관리자] 데이터를 외부로부터 보호하기를 원한다.	보안관리	보스 테스트 - 시스템 사용자들의

		개인정보에 대한 보안은 시스템의 신뢰성과 직결된다.

6.5 Use Case Text

6.5.1 Brief Description

앞서 추출한 유스케이스들에 대한 Brief Description 은 다음과 같다.

Use Case Brief Description

use case name	***description***
분류 관리	판매자는 카테고리 종류 관리를 위해 상품 관리 시스템에 접속한다. 판매자는 상품 관리 메인 화면에서 '카테고리 관리'를 선택한다. 시스템은 최상위 카테고리를 보여준다. 시스템이 최상위 카테고리 목록을 판매자에게 보여준다. 판매자는 조회하고자 하는 카테고리를 선택한다. 시스템이 선택된 카테고리 목록을 보여준다. 판매자는 선택한 카테고리 하위에서 카테고리를 추가,, 삭제할 수 있다. 그러나 판매자의 요청은 시스템에 곧바로 반영되지 않는다. 시스템 운영자는 판매자가 변경하고자 하는 내용이 적절한지 검수한다. (예를 들어, 신발 관련 카테고리가 아닌 것을 등록하려고 하는지) 운영자의 검수가 완료되면 비로소 카테고리 변경 내역이 반영된다.
상품 관리	판매자는 상품 관리를 위해 상품 관리 시스템에 접속한다. 판매자는 상품 관리 메인 화면에서 '상품 관리'를 선택한다. 시스템은 상품 정보 검색바와 전체 상품 목록(상품명, 상품 번호, 재고)을 보여준다. 판매자는 검색어를 입력하고 '검색' 버튼을 선택할 수 있다. 시스템은 검색어(상품명 또는 상품번호)와 일치하는 상품 목록을 보여준다. 판매자는 특정 상품을 선택하여 등록되어 있는 상품의 정보를 조회할 수 있다. 상품 상세 정보 조회 화면에서 판매자는 상품의 정보를 삭제할 수 있다.

재고 관리	판매자는 재고 관리를 위해 상품관리 시스템에 접속한다. 판매자는 상품 관리 메인 화면에서 '재고 관리'를 선택한다. 시스템은 상품 재고 검색바와 전체 상품 재고 목록(상품명, 상품 번호, 재고)을 보여준다. 판매자는 검색어를 입력하고 '검색' 버튼을 선택할 수 있다. 시스템은 검색어(상품명 또는 상품번호)와 일치하는 상품 재고 목록을 보여준다. 판매자는 특정 상품을 선택하여 상품의 재고를 조회할 수 있다. 판매자는 상품 재고 조회 화면에서 상품의 재고를 변경할 수 있다.
시스템 관리	관리자는 시스템이 원활하게 운영되도록 시스템에서 발생할 수 있든 다양한 에러를 수정하며, 유지보수 한다.
보안 관리	시스템 사용자들로부터 비롯된 개인정보 데이터들이 유실되거나 유출되지 않도록 관리한다.

6.5.2 Fully Dressed Format Description

앞서 추출한 유스케이스 중 복잡하다고 생각되거나 상품 관리 시스템에서 중요한 비중을 차지하는 것에 대한 Fully Dressed Format Description 은 다음과 같다.

- **Manage category use case**

유스케이스 번호	1
유스케이스 이름	manage category(카테고리 관리)
수준	사용자-목적
주요 액터	판매자
관련자 및 관심사항	- 판매자는 기본적으로 기존에 등록되어 있는 카테고리의 정보를 "조회"하기를 원하다. 또한 이를 "삭제"하려고 한다. 또한 온라인 쇼핑몰에 새로운 카테고리 정보를 "등록"하기도 원한다.

	- 상품 구매자는 판매자가 등록한 카테고리 정보를 기준으로 상품정보가 분류되어 표기되기를 원한다. - 시스템 관리자는 기본적으로 제공되는 분류와 더불어 판매자가 등록한 분류가 적절한지 검수하기를 원한다.
선행 조건	판매자는 상품 관리 권한을 가진 ID 로 로그인 하고, 카테고리 관리 기능을 실행한다.
후행 조건	판매자가 등록하거나 삭제한 카테고리 정보가 저장된다.
기본 시나리오	1. 시스템이 대분류 카테고리를 관리자에게 보여준다. 2. 판매자가 조회하고자 하는 중분류 카테고리를 선택한다. 3. 시스템이 카테고리의 중분류 카테고리 목록을 보여준다. 4. 판매자는 자신이 등록하고자 하는 소분류 카테고리 위치를 확보한다. 5. 판매자는 등록하고자 소분류 카테고리 이름과 카테고리 코드를 입력한 후 저장 기능을 실행한다. 6. 판매자는 자신이 삭제하고자 하는 소분류 카테고리를 선택한다. 7. 지정된 카테고리에 대해서 삭제 기능을 실행한다.
대안 시나리오	**[카테고리 정보의 중복]** 5a. 등록하고자 하는 카테고리 이름 또는 코드가 이미 존재하는 경우. 1. 카테고리 코드와 이름은 카테고리 고유 정보로 중복되어 저장될 수 없다. 시스템은 이미 동일한 정보를 가진 카테고리가 존재한다는 메시지를 보여준다.

	[시스템이 실패하는 경우] *a. 카테고리 관리를 위해 연결된 모든 상태와 이벤트는 기본 시나리오의 어떤 단계에서도 복구될 수 있다. 1. 판매자는 시스템을 재시작하고 이전 상태로의 복구를 요청한다. 2. 시스템은 이전 상태로 복구한다. 2a. 시스템이 복구 되는 데 방해가 되는 예외적인 것들이 발생하는 경우 1. 시스템은 판매자에게 오류를 표시하고, 이를 기록하며 초기의 깨끗한 상태로 들어간다. 2. 판매자는 기본 시나리오 1 로 이동하여, 카테고리 관리를 새롭게 시작한다.
발생 빈도	빈번하게 일어날 것이다.
특수한 요구사항	- 시스템이 사용자에게 정보를 출력하는 것은 30 초 이내에 이루어져야 한다. - 판매자가 불필요한 카테고리를 등록한다고 여겨질 경우, 시스템 관리자가 판매자에게 분류 조정을 요청할 수 있다.

- **Manage product use case**

유스케이스 번호	2
유스케이스 이름	manage product(상품 관리)
수준	사용자-목적
주요 액터	판매자

관련자 및 관심사항	- 판매자는 온라인 쇼핑몰에 새로운 상품 정보를 "등록"하고, 기존에 등록되어 있는 상품의 정보를 "삭제"하려고 한다. 또한 등록되어 있는 특정 상품 정보를 "검색"하고 상세정보를 조회하고자 한다. - 상품 구매자는 판매자가 등록한 상품 정보를 바탕으로 원하는 상품을 구매하기를 원한다. - 시스템 관리자는 판매자가 허위 정보를 등록하지 않는지 확인하고자 한다.
선행 조건	판매자는 상품 관리 권한을 가진 ID 로 로그인 하고, 상품 관리 기능을 실행한다.
후행 조건	상품 판매가 등록하거나 삭제한 상품 정보가 저장된다.
기본 시나리오	1. 판매자는 **필수 상품 정보**[2]를 입력하여 등록한다. 2. 시스템은 등록된 상품 정보를 저장한다. 3. 시스템은 전체 상품 목록을 보여준다. 4. 판매자는 특정 상품을 선택한다. 5. 시스템은 해당 상품에 대한 상세 정보를 보여준다. 6. 판매자는 상품을 삭제한다. 7. 시스템은 상품목록에서 상품을 삭제한다.
대안 시나리오	*a. **[시스템이 실패하는 경우]** 상품 관리를 위해 연결된 모든 상태와 이벤트는 기본 시나리오의 어떤 단계에서도 복구될 수 있다. 1. 판매자는 시스템을 재시작하고 이전 상태로의 복구를 요청한다.

[2] 필수 상품 정보에 대한 내용은 Glossary 문서를 참고 하시오.

	2. 시스템은 이전 상태로 복구한다. 2a. 시스템이 복구되는 데 방해가 되는 예외적인 것들이 발생하는 경우 1. 시스템은 판매자에게 오류를 표시하고, 이를 기록하며 초기상태로 돌아간다. 1a. **[필수 상품 정보를 입력하지 않는 경우]** 1. 시스템은 "필수 정보 입력 오류" 문구를 보여준다. 2. 시스템은 이전 '상품 등록' 상태로 되돌린다. 3a. [**여러 개의 상품을 한꺼번에 삭제하는 경우]** 1. 판매자는 한 개 이상의 상품들을 선택하고 삭제 기능을 실행한다. 2. 시스템은 "정말 삭제하겠습니까?" 문구를 보여준다. 3. 시스템은 등록된 상품목록에서 선택된 상품들을 삭제한다.
발생 빈도	매우 빈번하게 일어날 것이다.
특수한 요구사항	- 시스템이 사용자에게 정보를 출력하는 것의 90%는 3 초 이내에 이루어져야 한다. - 판매자가 허위 상품 정보를 등록하는 경우, 시스템 관리자가 판매자에게 제재를 가할 수 있다.

- **Manage inventory use case**

유스케이스 번호	3
유스케이스 이름	manage inventory(재고 관리)

수준	사용자-목적
주요 액터	판매자
관련자 및 관심사항	1. 판매자는 상품의 실시간 재고 정보를 확인하고 싶다. 2. 판매자는 입고시마다 기록된 재고 정보 이력을 관리하고싶다.
선행 조건	판매자는 상품 관리 권한을 가진 ID 로 로그인 하고, 재고 관리 기능을 실행한다.
후행 조건	판매자가 수정한 재고 정보가 반영되어야 한다.
기본 시나리오	1. 시스템은 판매자의 모든 상품에 대한 재고 현황을 옵션별로 구분하여 보여준다. 2. 판매자는 특정 상품에 대한 옵션을 선택한다. 3. 판매자는 수정기능을 선택한다. 4. 판매자는 수정할 상품재고를 입력하고 저장 기능을 선택한다. 5. 시스템은 수정된 재고정보를 저장한다.
대안 시나리오	2a. **[변경 수량 오류]** 변경을 하려는 수량이 음수일 경우 1. 시스템은 판매자에게 재입력 요청한다 2. 판매자는 자연수를 재입력한다. 3. 기본 시나리오 8 로 이동한다. ***a. [시스템이 실패하는 경우]** 상품 관리를 위해 연결된 모든 상태와 이벤트는 기본 시나리오의 어떤 단계에서도 복구될 수 있다.

	판매자는 시스템을 재시작하고 이전 상태로의 복구를 요청한다. 시스템은 이전 상태로 복구한다. 2a. 시스템이 복구되는 데 방해가 되는 예외적인 것들이 발생하는 경우 시스템은 판매자에게 오류를 표시하고, 이를 기록하며 초기상태로 돌아간다.
발생 빈도	종종 일어난다.
특수한 요구사항	- 시스템이 사용자에게 정보를 출력하는 것의 90%는 3 초 이내에 이루어져야 한다. - 판매자가 허위 상품 정보를 등록하는 경우, 시스템 관리자가 판매자에게 제재를 가할 수 있다

7. Supplementary Specification

7.1 Control Information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Supplementary Specification
Document owner	소프트조
Document identification no	7
Document status	Final

7.2 Revision History

버전	*일자*	*설명*	*저자*
Supplementary Specification 초안	2020.04.30	2020.04.28 일 조교님의 화상 Q&A 시간 때 들은 답변을 기반으로 작성된 최초 버전, FURPS 와 Implementation, Interface, Legal 부분 작성	박재영
Supplementary Specification v1.0	2020.05.01	2020.04.30 일 팀원 미팅 이후 Functionality 와 Interface 부분 수정	박재영
Supplementary Specification v1.1	2020.05.04	Interface 와 legal 수정	이연주
Supplementary Specification v1.2	2020.05.30	FURPS 수정 및 security, 인터페이스(사용자, 소프트웨어, 하드웨어) 수정	이연주, 박재영
Supplementary Specification v1.3	2020.06.13	FURPS 수정	안창희
Supplementary Specification v1.4	2020.06.18	Functionality 수정	빅재영, 정우택, 이연주

7.3 Introduction

이 문항에서는 유스케이스 부분에 언급된 요구사항들 외에 ShoeRack 에 필요한 요구사항을 다루고자한다. 또한 실질적인 기능 외에 플랫폼을 구축하는데 부가적인 요구사항을 명시한다.

7.4 Functionality

Feature 1. 권한 분리

- 판매자인지 구매자인지 따로 식별할 수 있어야 한다.
- 판매자가 접근할 수 있는 기능을 구매자가 접근하지 못하게 권한을 분리해야 한다.

Feature 2. 예외처리

- 에러 상황에 대하여 항상 시스템을 복구하거나 중지시킬 처리수단을 갖추어야한다.

Feature 3. 기록 관리

- 시스템 에러상황은 항상 로그 데이터로 기록되어야 한다.
- 상품, 고객 데이터의 변경 이력을 기록으로 항상 유지해야한다.

Feature 4. 변동사항 알림 기능

- 시스템 내 데이터의 주요 변동사항에 대하여 판매자들에게 메일로 알려준다.
- 알림 수신 여부를 판매자가 직접 정할 수 있다.

7.5 Usability

Feature 1. 호환성

- Windows, 맥 os, 리눅스 3 가지 운영체제에서 모두 정상적으로 동작해야한다.
- UI 는 출력 디스플레이변화에 유연하게 대응 가능해야한다.

Feature 2. 인적 요소

- 상품 정보는 간단하고 직관적으로 읽을 수 있어야 한다.
- 상품정보와 재고는 실시간으로 갱신되어야 한다.
- 판매자에게 최대한 구매자와 동일한 사용경험을 제공하는 동시에 상품관리를

수행할 수 있도록 해야 한다.

7.6 Reliability

Feature 1. 실시간 운영

- 시스템은 정상상태에서 24 시간 동안 무중단 서비스를 제공해야 한다.

Feature 2. 백업

- 시스템은 영구저장소를 통해 데이터베이스를 운용하며 항상 모든 데이터를 최신화하여 저장해야 한다.

Feature 3. 장애 복구

- 어떤 작업을 실행시키기 전에 그 이전의 상태를 저장하고 다음 작업을 실행시키는 형식으로 구성해서 시스템에 에러가 발생한 경우, 에러 발생 이전의 상태로 되돌아갈 수 있다

7.7 Performance

Feature 1. 평균 응답시간

- 사용자의 요청에 대한 결과를 화면에 출력할 때 4 초 이내에 보여주어야 한다.

Feature 2. 평균처리시간

- 사용자의 등록요청에 대한 처리를 3 초 이내에 해야 한다.

Feature 3. 동시 처리

- 전체 사용자 수는 최대 1000 명으로 한다.
- 최대 100 명의 동시 로그인 사용자를 수용해야 한다.

Feature 4. 최대 처리

- 로그인, 상품등록 등 기능에서 초당 최소한 100 건의 사용자 입력정보를 처리할 수 있어야 한다.

7.8 Supportability

Feature 1. 진화성

- 판매자의 매출을 효율적으로 올리기 위하여 필요한 경우 기능추가를 해야 한다.
- 이용자수가 늘어나거나, 이용자를 확대시켜서 신발외의 다른 의류의 관리도 가능하게 할 수 있다.

7.9 Implementation

Feature 1. 시스템 이식성

- 시스템은 호환 가능한 LINUX 운영체제에서 수행될 수 있어야 한다.

7.10 Security

Feature 1. 사용자 인증

- 시스템은 사용자가 접근을 시도할 때 인증을 해야 한다.
- 시스템은 인증할 때 개인정보보호지침을 준수해야 한다.

Feature 2. 사용자 접근 제어

- 시스템은 사용자의 아이디/비밀번호를 통해 사용자 유형을 파악한다.
- 접근제어 목록에 따라 권한에 따라 데이터의 접근 수준을 구분해야 한다.
- 시스템은 사용자가 일정 회수 이상 로그인 정보가 틀린 경우 접근을 제한해야 한다.

Feature 3. 데이터 무결성

- 시스템은 인가된 사용자가 외부에서 사용자 정보를 변경하려고 할 때 감사추적을 해야 한다.
- 시스템은 이용자 ID, 이용 시간, 조회 내용, 사용 현황 등의 조회 이력을 관리해야 한다.
- 시스템은 데이터 변경 이력과 데이터 접근 현황에 대한 로그 관리를 해야 한다.

Feature 4. 데이터 복구

- 시스템은 대량 데이터 Unload 및 데이터 복구를 관리해야 한다.
- 시스템은 보안 사고가 발생하여 데이터가 위변조, 손실되면 1 시간 이내에 데이터를 복구해야

한다.

Feature 5. 보안 정책 및 지침 준수

- 시스템은 소프트웨어 개발보안 가이드를 준수하여 개발해야 한다.

7.11 Interface

Interface 1. 사용자 인터페이스

- **홈페이지 사이트 구성**

  특정 브라우저에 종속되지 않고 상품판매자, 상품구매자들에게 웹 기반의 독립적인 서비스 지원이 가능해야 한다.

  PC, 모바일, Tablet 환경을 고려해서 동적인 화면을 구성해야 한다.

- **디자인 구성**

  모든 창은 현재 화면 외의 새로운 팝업창에서 열고 닫을 수 있도록 설계한다.

  상품판매자, 상품구매자들이 시각에 대한 피로도를 최소화 시키고 가독성을 극대화 시키기 위한 색상을 사용한다.

- **화면 사이즈 구성**

  상품판매자, 상품관리자가 사용하는 단말기에 독립적인 최적화 화면 및 정보를 제공한다.

  PC, 모바일에 최적화된 화면을 구성하되, 상대적으로 화면이 작은 모바일에서는 스크롤을 통해 세로로 정보를 제공할 수 있도록 설계한다.

Interface 2. 소프트웨어 인터페이스

- 시스템은 입력된 상품판매자의 실명 확인을 위해서 외부 실명확인시스템과 연동한다.
- 시스템은 플랫폼 관리자가 각 상품판매자들의 매출 정보들을 각 상품판매자들에게 전송하기 위해서 회계시스템과 연동한다.
- 시스템은 상품판매자의 사업자등록번호를 확인하기 위해서 외부 사업자등록번호 조회 시스템과 연동한다.
- 시스템은 플랫폼관리자가 상품판매자의 상품이 정상적으로 등록되었고 판매를 할 수 있음을 상품판매자에게 알려주기 위하여 메일시스템과 연동한다.

- 시스템은 이전 버전의 시스템에서 사용하던 '상품관리자 정보' 를 새로운 시스템에서도 사용해야 한다. 시스템 업데이트에 대한 정보를 상품판매자들에게 제공하기 위해서 메일시스템과 연동한다.

Interface 3. 하드웨어 인터페이스

- **모바일 기기와 PC**

  모바일 기기와 PC, 모든 하드웨어에서 ShoeRack 에서의 기능들을 동일하게 수행할 수 있다.

  모바일 스크린 운영체제에서는 일반적인 모니터로 인식되어 스크린 터치는 하나의 이벤트로 인식된다

8. Glossary

8.1 Control Information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Glossary
Document owner	소프트조
Document identification no	8
Document status	Final

8.2 Revision History

버전	*일자*	*설명*	*저자*
Glossary 초안	2020.04.26	프로젝트에 사용되는 용어를 정리한 Glossary 의 초안 작성	장유림
Glossary v.1.0	2020.04.27	신발 사이즈와 상품 판매가에 대한 검증 규칙 수정, 상품 코드의 구성 수정	장유림
Glossary v.1.1	2020.05.02	자바, 깃허브, 프론트엔드, 백엔드, 워드 프레스 용어 추가	장유림
Glossary v.1.15	2020.05.03	이클립스, 클라우드 서비스 용어 추가	이연주
Glossary v1.2	2020.05.04	상품분류 수정, 상품재고, 홈화면 추가	장유림
Glossary v1.3	2020.06.02	자바, 이클립스 등 기술적인 용어 삭제	장유림
Glossary v1.4	2020.06.02	SSD 관련 Definition 추가	이연주, 박재영
Glossary v1.5	2020.06.17	상품 분류 코드와 상품 코드 생성 규칙 추가	장유림

8.3 Definition

용어	*정의 및 정보*	*형식*	*검증 규칙*	*별칭*
상품	판매를 위한 제품, 본 프로젝트에서는 '신발'을 의미한다.	-	-	제품
상품 판매자	상품 관리 시스템을 이용하여 온라인 쇼핑몰에 자신의 브랜드 상품을 등록한 후 상품을 판매하는 사람이다. 소규모 브랜드의 판매 담당 직원 또는 대표가 판매자에 해당한다.	-	-	판매자
상품 분류	상품을 비슷한 범주를 가진 것끼리 묶는 것을 의미한다. 캔버스화, 스니커즈 등이 해당된다. 기본적인 카테고리는 플랫폼이 제공한다. 판매자는 플랫폼이 제공하는 기본 카테고리를 삭제할 수 없다. 단, 자신의 브랜드에 특화된 특별한 카테고리에 대해서 등록, 삭제가 가능하다.	-	-	분류, 카테고리
상품 옵션	상품의 사이즈와 상품의 색상에 대한 구체적인 정보를 의미한다.	-	-	옵션
신발 사이즈	신발의 세로 길이를 의미한다. 1cm 의 반이 되는 5mm 를 반사이즈로 정해서 230, 235, 240, 245 과 같이 0.5cm 씩 크기가 커지도록 신발을 판매한다.	양의정수, 밀리미터 단위	3 자리 정수이어야 한다. 5 밀리미터 단위로 증가한다.	사이즈, 크기
상품 판매가	상품 구매자가 상품을 구매하기 위해 판매자에게 지불해야 하는 돈의 액수이다.	양의정수, 원화 단위	0 에서 999,999 사이의 값이어야 한다.	판매가, 가격
상품 재고	상품에 대한 주문이 들어오면, 당장 판매할 수 있는 상품의 수량을 의미한다.	양의정수	0 에서 999,999 사이의 값이어야 한다.	재고
상품 제조사	상품을 제조하여 해당 쇼핑몰에 납품하는 업체에 대한 정보이다. 즉,	-	-	제조사

	판매자가 누구인지 구별하는 정보이다.			
(상품관리시스템) 홈화면	상품 관리 시스템의 manage category, manage products, manage inventory 메뉴를 보여주는 화면이다. 상품 관리 시스템에 접속했을 때, 가장 먼저 보여진다.	-	-	메인화면
필수상품정보	상품 등록 시 필수적으로 입력해야 하는 정보를 나타낸다. 상품명, 상품 가격, 상품 요약, 상품 상세 설명이 이에 속한다.	-	-	상품정보
재고정보	상품 코드(product code)와 상품 수량이 재고정보에 속한다.	-	-	-

- 상품 분류 코드와 상품 코드에 대한 생성 규칙

상품을 분류하는 기준이 되는, 상품 분류 코드는 대, 중, 소분류 조합(1 + 2 + 4 자리, 문자열)에 따라 다르게 부여된다. 자세한 대, 중, 소 분류는 아래 그림을 참조하라. 또한 상품을 구별하는 기준이 되는, 상품 코드(최대 4 자리, 문자열)는 해당 분류에 해당하는 상품에 대하여 등록순으로 차례대로 값이 부여된다.

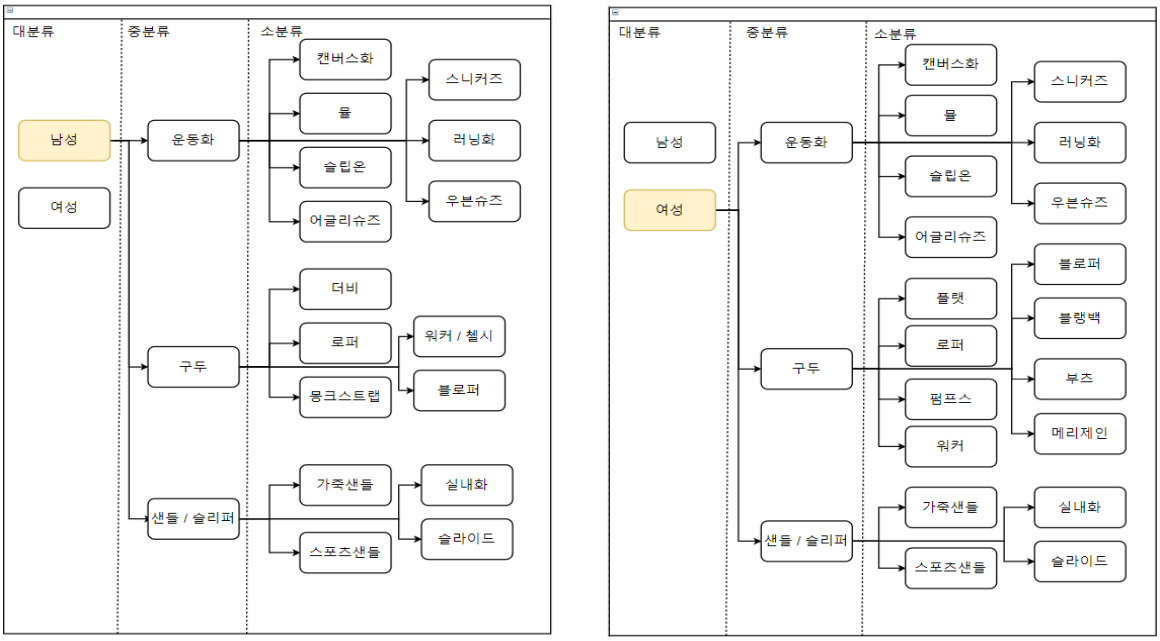


그림 9 상품 분류 지도

L0 M17 S0072 P0003

대분류 중분류 소분류 상품번호(등록순)

그림 10 상품 코드 예

9. Domain Model

9.1 Control Information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Domain Model
Document owner	소프트조
Document identification no	9
Document status	Final

9.2 Revision History

버전	***일자***	***설명***	***저자***
Domain Model 초안	2020.05.25	Domain Model 분류리스트 추가	이연주, 정우택
Domain Model v.1.0	2020.05.27	Domain Model Class Diagram 추가	전원
Domain Model v.1.1	2020.06.1	속성, 새로운 데이터 타입 추가	이연주, 정우택 박재영
Domain Model v1.2	2020.07.02	Domain Model Class Diagram Option 추가	정우택, 박재영 장유림
Domain Model v1.3	2020.07.03	Domain Model Class 수정	정우택, 박재영 장유림
Domain Model v1.4	2020.07.11	Domain Model 클래스 재구성 및 연관관계 수정	정우택, 박재영, 장유림

9.3 Introduction

도메인은 관련 도메인의 개념 클래스 또는 실세계 객체들의 시각적 표현이다. 이 문서는 프로젝트 OOA(객체 지향 분석 기법)을 통해 real world 의 상품 관리에 관련된 model 을 추출하였다. 이를 통해 프로젝트 개발자는 비즈니스 도메인에 대한 주요개념과 용어를 이해할 수 있다.

9.4 Class Diagram

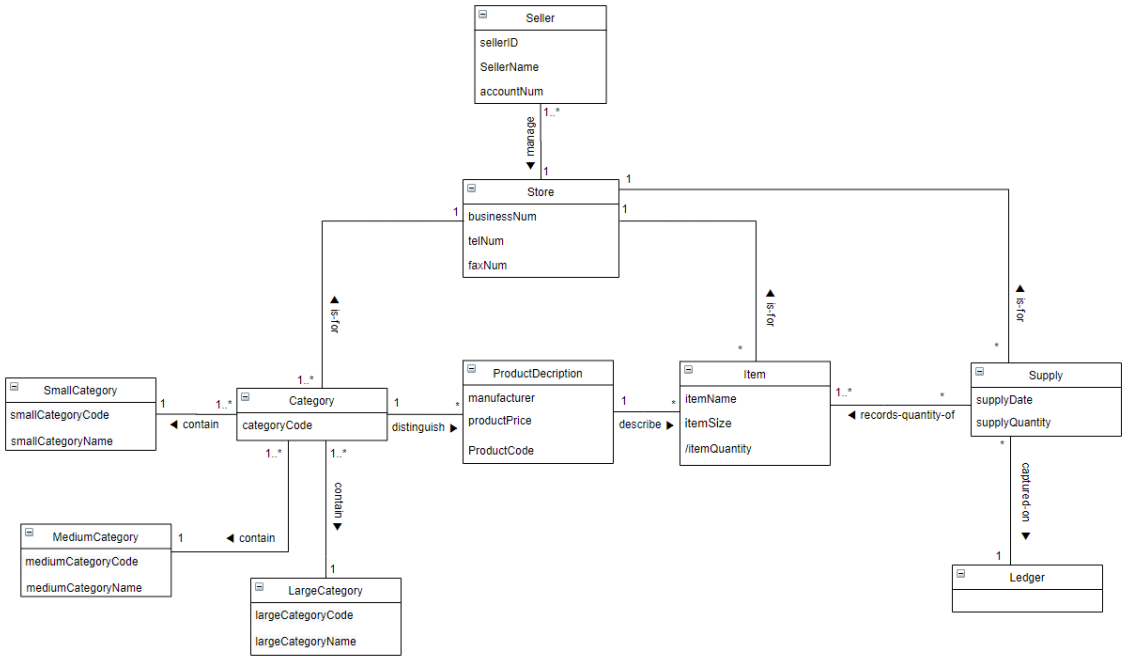


그림 11 Domain Model Class Diagram

9.5 분석 방법 및 초기 결과

도메인 모델이란 도메인 안에 존재하는 개념적 클래스나 실제 상황의 객체를 시각적으로 표현한 것이다. 도메인 모델을 생성하는 방법은 다음과 같다.

1. 개념적 클래스를 찾는다.
2. 이들을 UML 클래스 다이어그램 안에 클래스로 그린다..
3. 연관 관계와 속성을 추가한다.

도메인 모델을 생성하기 위해, 가장 먼저 개념적 클래스를 찾아야 한다. 개념적 클래스를 찾는 방법에도 다음과 같은 전략이 있다.

1. 기존의 모델을 재사용하거나 수정한다.

2. 분류 리스트(category list)를 사용한다.
3. 유스케이스에서 명사구를 식별한다.

개념적 클래스를 찾기 위해 방법 2 와 방법 3 을 혼용하였다. 아래 그림은 상품 관리 시스템이 필요로 하는 것과 시스템에서 고려할 가치가 있는 일반적인 항목을 포함하는, 개념적 클래스 후보들의 분류 리스트이다.

개념적 클래스 후보 - 분류 리스트 사용

개념적 클래스의 분류	후보
비즈니스 트랜잭션	Manage, Supply, Display
트랜잭션 라인 아이템	LineItem
트랜잭션이나 트랜잭션 라인 아이템과 관련된 상품이나 서비스	Item, Category
트랜잭션이 어디에 기록되는가?	Ledger
트랜잭션에 대한 사람이나 조직의 역할	Seller, PlatformManager, SystemAdministrator
물리적 객체	Shoes, Computer, Board
사물에 대한 설명	ProductDescription, Categoryinfo
사물에 대한 기록	Ledger, RegisterLog
같이 협력하는 시스템	SMSsystem(통합메세징시스템), OrderingManagementSystem(주문관리 시스템)
작업을 수행하기 위해 자주 참고하는 일정, 메뉴얼, 문서	OrderList

아래 그림은 유스케이스에서 명사와 명사구를 식별한 것이다. 도메인 모델은 중요한 도메인 개념과 어휘를 가시화한 것으로, 대부분 유스케이스에서 발견된다. 명사구 일부는 개념적 클래스의 후보이고, 또 다른 일부는 클래스의 속성이 될 수 있다. 이 방법의 단점은 자연어의 부정확성이다. 여러 다른 명사구들이 동일한 개념적 클래스나 속성을 표현할 수도 있다. 때문에 개념적 클래스 분류 리스트와 혼합해서 사용한다.

개념적 클래스 후보 - 유스케이스에서 추출된 명사구

유스케이스	추출된 명사구
카테고리 관리	소, 중 ,대 분류 카테고리, 관리자, 판매자, 조회, 카테고리 명, 카테고리 코드
상품 관리	판매자, 등록, 상품 목록, 최상위 카테고리, 최하위 카테고리, 상품 정보, 상품 요약, 검색, 검색 키워드, 상품 이름, 상품 상세 정보
재고 관리	상품, 판매자, 상품 재고, 검색

분류 리스트와 명사구 분석을 이용하여, 도메인에 대한 개념적 클래스 후보 리스트를 생성하였다. 상품 관리 시스템은 특히 정보 관리 시스템이기 때문에, 시스템에 정보가 기록되어야 하는 객체의 선별을 강조하여 최종 개념적 클래스를 추출하였다. 속성과 연관관계가 표시되지 않은 초기 도메인 모델은 아래와 같다. 상품에 대해 설명 클래스(description class)가 추가되어 있는데, 이는 상품과 카테고리 정보가 과다하게 중복되는 것을 막기 위함이다. 또한, 상품 하나가 판매되어 시스템에서 해당 상품의 인스턴스가 삭제되더라도 그 상품에 대한 정보는 시스템에 기록되어야 하기 때문이다.

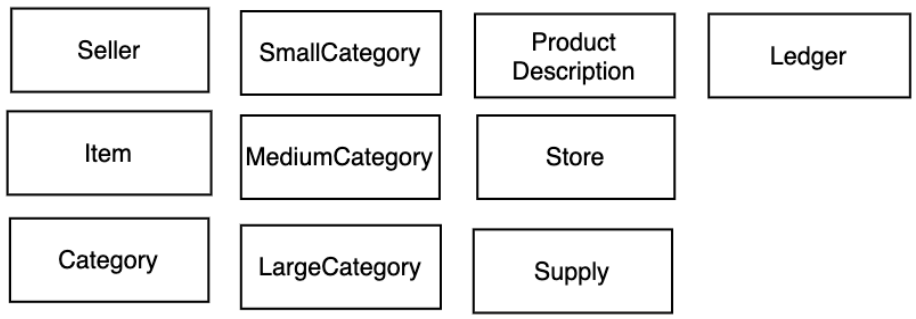


그림 12 개념적 클래스

개념적 클래스 설명

Class Name	***description***
Seller	상품 판매자를 나타낸다.
item	하나의 상품을 나타낸다. (실제 상품 하나)

Product Description	Product 에 대한 설명 클래스이다. (공통 상품 정보)
Category	한 개 상품에 대한 대분류, 중분류, 소분류 조합이다.
SmallCategory	상품 소분류.
MediumCategory	상품 중분류.
LargeCategory	상품 대분류.
Supply	상품에 대한 재고 공급이다.
Ledger	상품에 대한 재고변경이력이다.

9.6 Association

앞서 추출한 개념적 클래스 사이의 연관 관계는 다음과 같다.

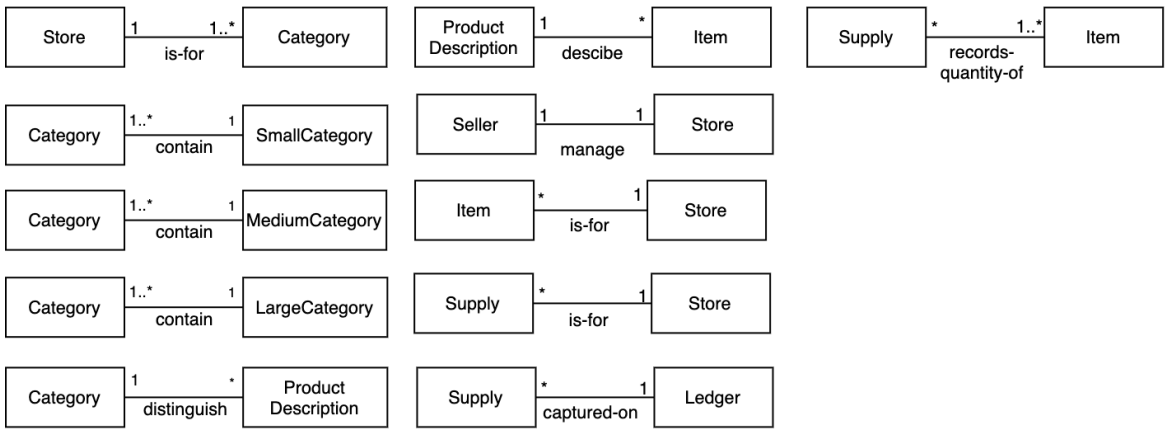


그림 13 개념적 클래스 사이의 연관 관계

개념적 클래스 연관성 설명

연관 관계	*설명*
Store — Category	분류(Category)들은 상점(Store)을 위해(상점의 상품들을 분류) 존재한다.
Category — Small Category	카테고리(Category)는 소분류 카테고리(SmallCategory)를

	포함한다.
Category — Medium Category	카테고리(Category)는 중분류 카테고리(MediumCategory)를 포함한다.
Category — Large Category	카테고리(Category)는 대분류 카테고리(LargeCategory)를 포함한다.
Category — Product Description	하나의 분류(Category)는 상품 정보(Product Description)를 포함한다(가진다).
Product Description — Item	하나의 상품 정보(Product Description)는 여러 개의 상품(Item)을 설명할 수 있다.
Seller – Store	판매자(Seller)는 상점(Store)를 관리한다.
Item — Store	상품(Item)은 상점(Store)을 위해(판매되어 매출 생성) 존재한다.
Supply — Store	공급(Supply)은 상점(Store)을 위해(상품의 재고를 추가) 존재한다.
Supply — Ledger	모든 공급(Supply)은 장부(Ledger)에 기록된다.
Supply — Item	공급(Supply)은 상품(Item)의 수량을 기록한다.

9.7 Attribute

각 개념적 클래스가 갖는 속성에 대한 설명은 다음과 같다.

개념적 클래스가 나타내는 속성

Class Name	***Attributes***
Seller	- sellerID: 판매자 식별 번호. - sellerName: 판매자 실명. - accountNum: 판매자 계좌번호.
Store	- businessNum: 사업장의 사업자번호. - telNum: 사업장 연락처. - faxNum: 사업장 팩스번호.

Item	- itemName: 아이템 이름 - itemSize: 아이템의 사이즈 - itemQuantity: 아이템의 수량을 의미한다.
Product Description	- manufacturer: 제조사의 이름을 의미한다. - productCode: 카테고리코드를 포함하는 상품 고유 번호이다.. - productPrice: 상품 판매가.
Category	- categorySetCode: 하나의 상품이 속한 대/중/소 분류 코드의 조합.
(Small,Medium, Large)Category	-(Small,Medium,Large)categoryName: (대,중,소) 상품 분류 기준명. -(Small,Medium,Large)categoryCode: (대,중,소) 상품 분류 기준에 대한 고유 번호. 같은 분류범주(대/중/소)에 속하는 코드값은 중복되지 않게 부여된다. 카테고리의 L/M/S 은 각각 대/중/소 코드를 나타낸다.
Ledger	-
Supply	- supplyDateTime: 재고 업데이트 시간.. - supplyQuantity: 재고량.

다음은 개념적 클래스의 속성을 표현하기 위해 필요한 데이터 타입 클래스이다. 날짜는 년도, 월, 일, 시간, 즉 여러 부분으로 구성되어 있다. 숫자나 문자와 같은 하나의 정보가 아닌 별도의 데이터 타입 클래스를 정의하여 각 부분을 나누어서 관리한다. 상품 분류 코드 또한 같은 맥락이다. 특히 상품 분류 코드는, 여러 부분으로 구성되어 있다. 상품 분류 코드는 대분류, 중분류, 소분류에 대한 고유 번호가 차례대로 나열된 것이다. 별도의 데이터 타입 클래스를 정의하여 관리하면, 분류 코드에 대한 파싱이나 검증이 간단해질 것이다. (분류 코드에 부여 규칙에 대한 자세한 내용은 7 번 문서 Glossary 를 참조하라.

9.8 새로운 데이터타입

Entity Name	***Data***	***Logic***
Date	날짜 및 시간(supplyDateTime)	검증
categoryCode	분류 통합 코드(categorySetCode)	파싱, 검증
ProductCode	상품 고유 코드(productCode)	파싱, 검증

10. SSD, OC

10.1 Control information

Document title	온라인 쇼핑몰 상품 관리 시스템의 System Sequence Diagram, Operation Contract
Document owner	소프트조
Document identification no	10
Document status	elaboration

10.2 Revision History

버전	*일자*	*설명*	*저자*
SSD, OC v1.0	2020.05.26	유스케이스를 바탕으로 SSD 작성	박재영, 장유림
SSD, OC v1.1	2020.05.30	System Event 추출 근거, System Operation 선정 근거 작성	장유림
SSD, OC v1.2	2020.05.31	Operation Contract 작성	박재영
SSD, OC v1.3	2020.06.02	SSD 및 Operation Contract 수정	박재영
SSD, OC v1.4	2020.06.02	Manage Product SSD, OC 수정	이연주
SSD, OC v1.5	2020.07.02	SSD attribute 수정	정우택, 장유림 박재영
SSD, OC v1.6	2020.07.03	OC 수정	정우택, 장유림 박재영
SSD, OC v1.7	2020.07.11	SSD, OC attribute 수정	정우택, 장유림 박재영

10.3 System Sequence Diagram

- manage category 시나리오의 SSD

manage category 시나리오는 판매자가 온라인 쇼핑몰의 상품들을 분류하기 위한 카테고리 정보를 "등록"하고, "삭제"하는 것이다. manage category 유스케이스를 바탕으로 작성한 SSD 는 다음과 같다.

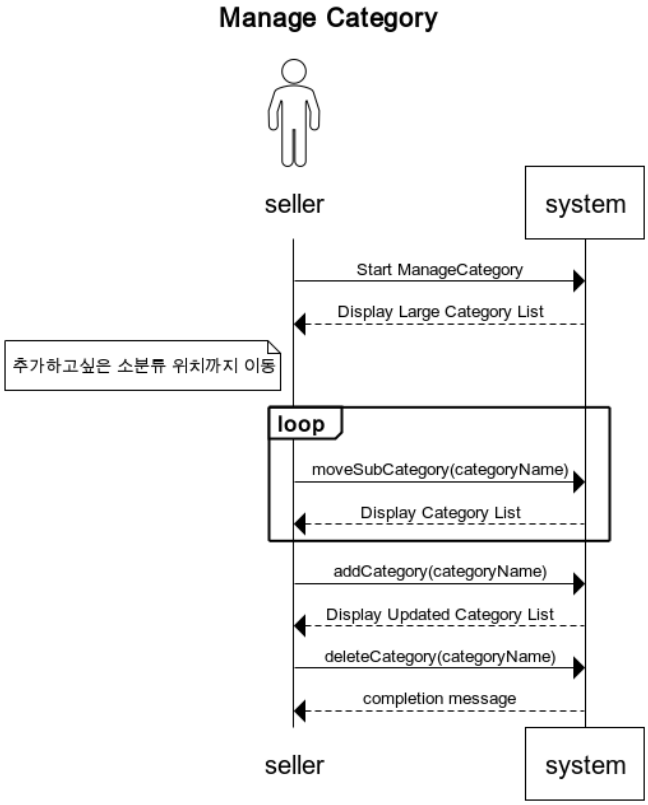


그림 14 manage category 시나리오 SSD

- manage product 시나리오의 SSD

manage product 시나리오는 판매자가 온라인 쇼핑몰의 상품들을 추가, 삭제하는 것이다. manage product 유스케이스를 바탕으로 작성한 SSD 는 다음과 같다.

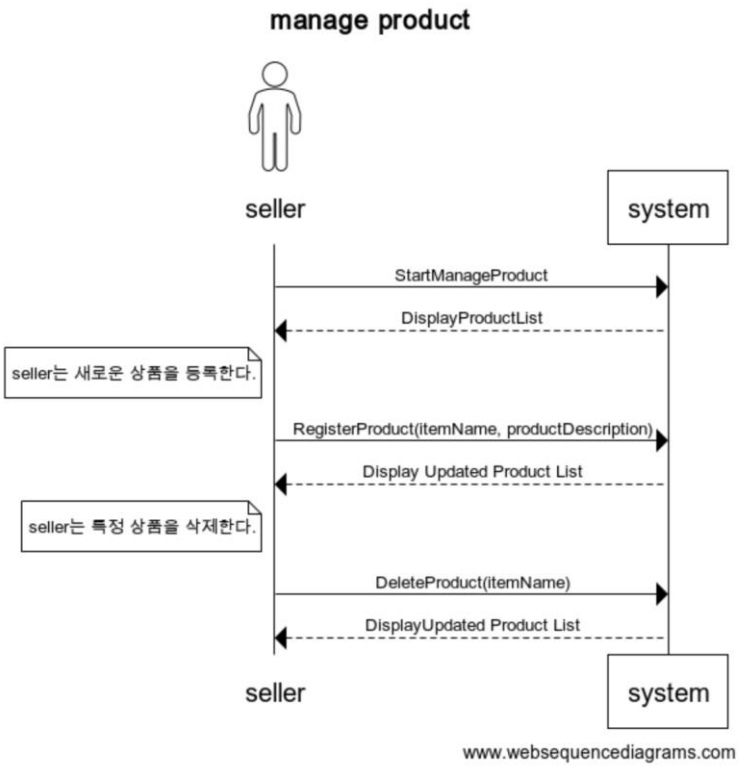


그림 15 manage product 시나리오 SSD

- manage inventory 시나리오의 SSD

manage inventory 시나리오는 판매자가 온라인 쇼핑몰에 등록되어 있는 상품들의 재고 양을 확인하고 수정하기 위한 것이다. manage inventory 유스케이스를 바탕으로 작성한 SSD 는 다음과 같다.

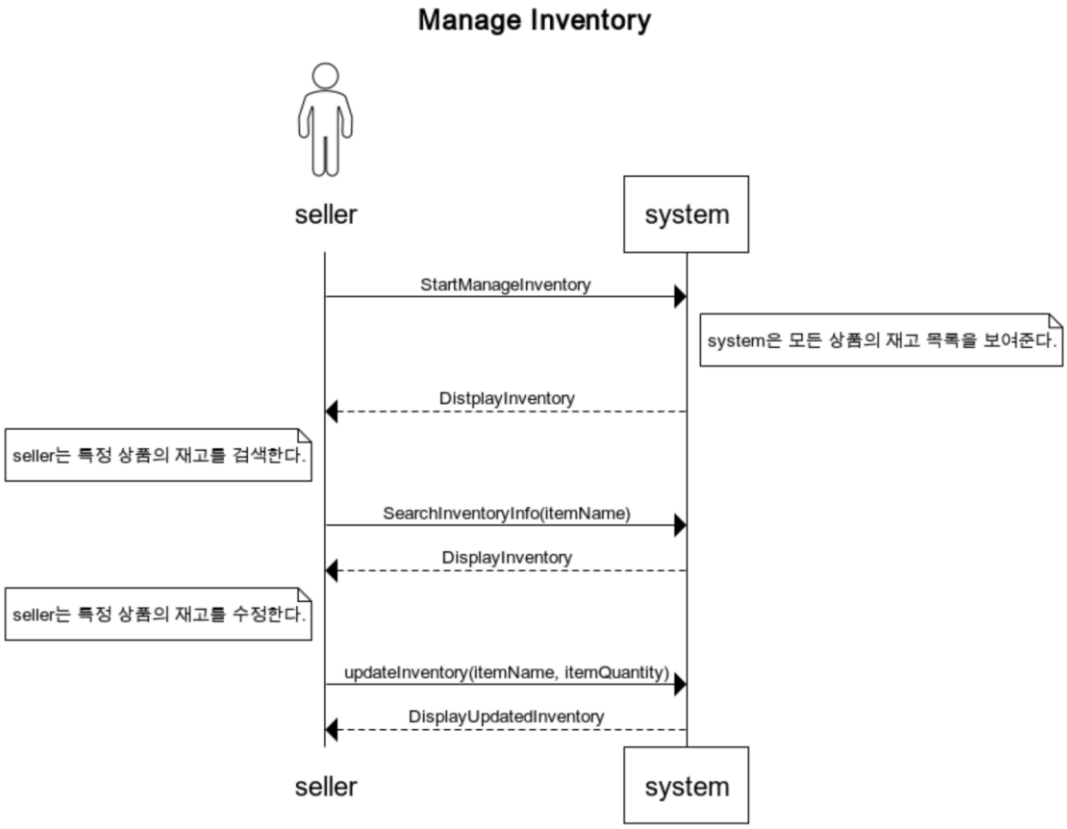


그림 16 manage inventory 시나리오 SSD

10.3 System Event 추출 근거

유스케이스는 우리가 만들고자 하는 소프트웨어 시스템과 시스템 외부의 액터가 어떻게 상호작용하는가를 설명한다. 이러한 상호작용의 과정에서 액터는 시스템 이벤트를 생성시킨다. 예를 들어, 판매자가 새롭게 등록할 상품 번호와 재고량을 상품 관리 시스템에 입력하는 것이 시스템 이벤트이다. 미리 작성한 유스케이스 시나리오에서, 시스템 외부의 액터가 시스템에 어떤 메시지를 전달하며 시스템에게 무언가를 요청하는 행위가 발생할 때, 그 것을 시스템 이벤트로써 추출하였다.

10.4 System Operation 선정 근거

소프트웨어 시스템과 시스템 외부의 액터가 상호작용하는 과정에서, 액터가 시스템에게 어떤 메시지를 전달하며 무언가를 요청하는 행위를 할때, 시스템 이벤트가 발생한다고 3.에서 설명했다. 액터가 시스템에게 요청하는 '무언가'가 바로 시스템 오퍼레이션이다. 예를 들어, 판매자가 새롭게 등록할 상품 번호와 재고양을 시스템에 입력하면, 상품 관리 시스템은 그 상품의 재고를 기록한다. 판매자가 발생시킨 시스템 이벤트에 의해 상품 관리 시스템의 시스템 오퍼레이션이 발생한 것이다. 앞서 유스케이스로부터 추출한 시스템 이벤트를 바탕으로, 시스템 이벤트를 처리하기위해 필요할 것이라고 생각되는 시스템 오퍼레이션을 추출하였다. 유스케이스를 분석한 결과, 위 세가지 SSD 모두 판매자가 상품 관리를 하기 위해, 찾기(검색), 등록, 삭제를 하기 때문에 Category, Name, Price, Code, Quantity 등을 매개변수로 받는 system operation 들을 만들 수 있었다.

10.5. Operation Contract 기술

실질적인 데이터의 변화가 일어나는 각 System Sequence Diagram 의 operation 에 대해 설명하자면 다음과 같다.

- manage category

오퍼레이션: addCategory(LargeCategoryCode , MediumCategoryCode, categoryName)

->새로운 소분류를 추가한다.

상호 참조: 유스케이스 manage category

사전조건: 추가할 카테고리의 상위 대분류, 중분류는 선택되어있다.

사후조건:

- SmallCategory 인스턴스 'S'가 생성되었다. (인스턴스 생성)
- S.CategoryCode 가 S.index 로 되었다. (속성 수정)
- S.CategoryName 이 입력된 소분류 이름으로 되었다. (속성 수정)
- Category 인스턴스 'C'가 생성되었다. (인스턴스 생성)
- Category 는 (Large,Medium,Small)Category 들과 연관되었다. (연관관계 생성)
- C.CatgorySetCode 는 LargeCategoryCode + MediumCategoryCode + S.CategoryCode 로 되었다.(속성 수정)

오퍼레이션: deleteCategory(categoryName) ->등록되어 있는 소분류를 삭제한다.

상호 참조: 유스케이스 manage category

사전조건: 입력 받은 categoryCode 의 소분류에 속한 product 가 없어야 한다.

사후조건:

- 입력 받은 categoryName 를 가진 Category 인스턴스가 삭제되었다. (인스턴스 삭제)
- (Large,Medium,Small)Category 와 Category 와 연관이 삭제되었다. (연관관계 삭제)
- SmallCategory 인스턴스가 삭제되었다. (인스턴스 삭제)

- manage product

오퍼레이션: RegisterProduct(itemName, itemQuantity, productPrice)

-> 새로운 상품을 등록한다.

상호 참조: 유스케이스 manage product

사전조건: 새로운 상품을 등록할 대분류, 중분류, 소분류가 선택되어 있다.

사후조건:

- Item 인스턴스 'i'가 생성되었다. (인스턴스 생성)
- i.ItemName 이 itemName 이 되었다. (속성 수정)
- i.itemQuantity 가 itemQuantity 가 되었다. (속성 수정)
- 현재 카테고리 인스턴스 C 와 현재 인스턴스 i 가 연관되었다. (연관관계 생성)
- Product Description 인스턴스 'pd'가 생성되었다. (인스턴스 생성)
- pd.productPrice 가 productPrice 가 되었다. (속성 수정)

- pd.productCode 가 C.categoryCode + pd.index 가 되었다. (속성 수정)
- i 가 pd 와 연관되었다. (연관관계 생성)
- i 가 현재 Category 인스턴스 c 와 연관되었다. (연관관계 형성)

오퍼레이션: deleteProduct(itemName) -> 기존 상품을 삭제한다.

상호 참조: 유스케이스 manage product

사전조건: 없음.

사후조건:

- 매개변수로 받은 itemName 과 일치하는 i.itemName 을 가지는 Item 인스턴스와 연관돼있는 인스턴스 i 와 연관되어있는 product description 인스턴스 pd 가 삭제된다. (인스턴스 삭제)
- 매개변수로 받은 itemName 과 일치하는 i.ItemName 을 가지는 Item 인스턴스 i 가 삭제된다. (인스턴스 삭제)

- manage inventory

오퍼레이션: updateInventory(itemName, itemQuantity) -> 상품의 재고를 수정한다.

상호 참조: 유스케이스 manage inventory

사전조건: 없음.

사후조건:

- 매개변수로 받은 itemName 과 일치하는 i.ItemName 을 가지는 Item 인스턴스 i 의 i.itemQuantity 를 매개변수로 받은 itemQuantity 로 업데이트 한다.. (속성 수정)

11. Logical Architecture

11.1 Control information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Logical Architecture
Document owner	소프트조
Document identification no	11
Document status	elaboration

11.2 Revision History

버전	*일자*	*설명*	*저자*
Logical Architecture 초안	2020.05.26	supplementary specification 분석	소프트
Logical Architecture v.1.0	2020.05.29	패키지 단위로 클래스 분류, 레이어 정의	안창희, 장유림
Logical Architecture v.1.1	2020.05.30	logical architecture 다이어그램 작성	안창희, 장유림

11.3 Introduction

논리적 아키텍처는 소프트웨어를 구성하는 요소들을 패키지, 서브시스템, 계층과 같은 큰 단위로 구성한 것이다. 이를 논리적 아키텍처라고 부르는 이유는 아키텍처 내의 모든 요소가 서로 다른 운영체제의 프로세스나 네트워크로 연결된 물리적인 컴퓨터로 어떻게 배치될 것인가를 결정할 필요가 없기 때문이다.

온라인 쇼핑몰의 상품 관리 시스템의 논리적 아키텍처를 작성하기 위해, 시스템의 비기능적 요소에 필요한 소프트웨어 클래스들을 패키지 단위로 구성한 후, 서로 연관된 책임을 갖는 패키지를 묶어 서로 구별되는 계층 단위로 분리하였다. 계층에는 UI, 도메인, 기술적 서비스 총 3 개가 있다. 논리적 아키텍처는 UML 패키지 다이어그램을 사용하여 시각적으로 표현하였다.

11.4 Architectural Factors

Supplementary Specification 문서의 비기능적 요구사항에 대한 솔루션이다.

요소	*측정시나리오*	*다양성*	*다른 요소와 이해관계자에 대한영향*	*우선 순위*	*관련 패키지*
다양한 운영체제에 대한 시스템 이식성과 호환성	Linux, Windows, Ios 세 가지 os 를 지닌 각각의 사용자 기기에서 시스템을 실행했을 때, 모두 같은 화면 구성을 지니게 되며, 사용자와 인터페이스의 상호작용 또한 모두 동일하게 진행된다.	현재: Web(웹)을 이용한 이라는 시스템 자체의 범용성으로 요구사항에 대한 충족이 가능하다. 진화: 시스템의 고객이 증가할 경우 더 다양한 사용자 요구사항을 수용하 고 기능향상을 위해서 Android 나 Ios 전용 어플리케이션 등의 추가적 플랫폼 구성을 꾀함으로써 유연성을 강화할 수 있다.	높은 영향도를 지닌다. 사용자들은 시스템을 사용함에 있어서 자신의 운영체제적 제약사항을 마주하게 된다면 큰 불편함을 느낄 것이다.	높음	web
논리적인 사용자 관점에서의 정보제공	PC, 태블릿, 스마트폰 등 다양한 기기에서 시스템을 작동 시킬 때, 한 페이지에 동일한 형식으로 10 개의 상품정보를 표현시켜진다.	현재: 웹페이지를 구성하는 프런트 엔드 차원에서의 요소를 반응형 웹을 지향하는 스크립트언어로 작성하여 하드웨어간 UI 의 통일성과 모색할 수 있다. 진화: 구성된 반응형	높은 영향도를 지닌다. 사용자들은 시스템을 사용함에 있어서 자신의 하드웨어	높음	반응형 웹

		웹이 수용하지 못하는 기능, 성능적 문제를 마찬가지로 어플리케이션 추가구성을 통해 해결할 수 있다. 또한 적응형 웹으로의 이중구축 또한 기존의 단점을 어느정도 해소해주는 방법이 된다. 하지만 반응형 웹구조는 발전가능성이 많은 기술이므로 기능, 성능적 보완이 계속해서 연구되고있으므로 계속해서 최신기술의 반응형 구조를 사용하도록 시스템의 버전 업데이트가 필요하다.	환경에 따라 인터페이스 출력문제가 발생하게 된다면 큰 불편함을 느낄 것이다.		
에러메세지	사용자가 시스템 사용 중 잘못된 입력값을 넣거나 프로그램간 충돌로 인해 시스템이 오동작 할 경우 사용자에게 1 분이내에 에러 메세지를 출력한다.	현재: 웹의 로깅 기능을 이용하여 실시간으로 에러상황에 대하여 대응하도록 시스템 자체기능을 구축한다. 진화: 에러에 대한 사전정의만으로 모든 상황을 다 예방할 순 없으므로 시스템을 운영하며 로그데이터를 누적 후 기능을 개선하며 보완 가능하다.	높은 영향도를 지닌다. 시스템이 오작동한 상황에 대하여 사용자에게 아무런 고지가 없을 경우	높음	web 로그기록

데이터 무결성 및 복구	사용자가 시스템을 통해 데이터에 접근한 경우 변경 이력, 접근 현황에 대한 로그가 데이터베이스에 기록되어 유지,관리된다. 관리중인 데이터베이스에 문제가 생길 경우 시스템관리자는 적합한 절차를 거쳐서 이전 상황의 데이터베이스로 복구시킨다.	현재: 웹의 로깅 기능과 데이터베이스가 DBMS 로 연동하여 접근이력을 유지 관리한다. 또한 일정기간의 텀을 두고 다양한 버전의 데이터베이스를 유지하여 문제시 복구절차에 사용할 수 있도록 한다. 진화: 시스템의 규모가 증가함에 따라 버전 별 데이터베이스를 효율적으로 관리하는 추가적인 시스템을 구축할 필요성이 증가한다.	보통의 영향도를 지닌다. 로그관리가 없는 경우 비정상적인 접근이나 이력을 관리하는 측면에서 시스템관리자 에게 어려움이 된다.	보통	로그기록 영구저장장치
시스템장애 상황에 대한 복구와 시스템의 동작유지	사용자가 시스템을 사용하는 도중 시스템 내부문제의 발생으로 서비스가 작동할 수 없다고 판단되는 경우 복구와 동시에 서비스운용을 최대한 유지한다.	현재: 웹서버의 구축 시 웹 호스팅에서의 분산구축 방식을 택하여 문제 발생시에도 예비서버가 작동하여 실시간 동작하도록 한다. 진화: 서버차원에서의 복구 자동화절차를 구축하는 것은 현재	보통의 영향도를 지닌다. 문제가 발생과 상관없이 사용자는 항상 이용가능한 시스템을 원한다.	낮음	웹호스팅

		시스템상으로는 높은 차원의 문제이다. 시스템 확장 시 추가적인 복구절차를 구축할 수 있다.			
요구 처리시간과 처리량 부합	사용자가 등록요청을 할 경우 시스템은 처리를 3 초 이내에 해야 한다. 또한 시스템의 전체 사용자 수는 최대 1000 명이며 최대 100 명의 동시 로그인 사용자를 수용해야 한다.	현재: 시스템은 기본적으로 웹호스팅 기술을 사용하여 보안, 성능적으로 검증된 웹서버를 대여한다. 진화: 사용자수 동향을 통계적으로 파악하여 적절한 시기에 서버를 증설하여 수용인원으로 인한 문제를 방지해야 한다.	보통의 영향도를 지닌다. 사용자들은 서버의 수용문제로 인한 시스템 동작중지를 용납하지 않을 것이다.	보통	웹호스팅
보안정책 지침 및 준수와 접근제어 및 인증	검증되지 않은 사용자가 시스템에 접근할 경우 시스템상의 인증절차에 따라 사용이 제한된다.	현재: 보안지침 준수를 통해 검증된 수준의 보안망을 지닌 시스템을 구축한다. 진화: 보안기능에 대한 별도 개발을 진행하여 발생가능한 문제에 대해 미리 유동적으로 대응 가능하도록 시스템을 업데이트한다.	보통의 영향도를 지닌다. 사용자는 시스템 사용 중 보안적인 결함으로 자신에게 생기는 문제에 대해 민감하게 반응할 것이다.	보통	보안

11.5 Layer and Package

ShoeRack 시스템의 아키텍처는 가장 일반적인 방법인 layerd architecture 방식으로 설계되었다. 또한 온라인 쇼핑몰의 서브시스템이라는 특성을 고려할 때 relaxed layer 가 적합했다. layer 를 구성하기 위해 필요한 요구사항은 다음 세 가지 였으며 각각을 1 개의 layer 로 선정했다.

-사용자와의 상호작용 => UI

-도메인 개념을 나타내는 인스턴스 =>Domain

-범용 인스턴스와 기술 서비스를 지원하기 위한 서브시스템 => Technical Service layer

- **UI 계층**

  시스템 사용자에게 보여지는 기능에 대한 책임을 갖는다. 사용자에게 보여주기 위한 데이터들이 합성 및 변환되며, 사용자의 요청을 직접적으로 입력 받는다.

패키지 이름	***분류 근거 및 역할***
Web	패키지 내부 요소는 HTML 언어를 사용하는 하이퍼텍스트 문서를 관리하는 공통의 목적을 갖는다. 이렇게 작성된 웹 상의 문서에 HTTP 라는 프로토콜을 사용하면 최종 사용자에게 직접 보여지는 웹페이지를 디자인할 수 있다.
반응형 웹구조	패키지 내부 요소들은 최종 사용자가 시스템에 접근하기 위해 직접적으로 이용하는 여러 장치의 다양한 특성에 대응한다. 브라우저의 크기(스크린의 크기)에 실시간으로 반응하여 최종 사용자에게 보여지는 웹 페이지의 레이아웃이 변하게 하려는 공통의 목적을 갖는다.

- **Domain 계층**

  UI 계층의 요청을 처리하는 책임을 갖는다. UI 계층에서 전달받은 요청을 처리하기 위해 소프트웨어가 수행할 작업을 정의하고, 표현력 있는 도메인 객체가 문제를 해결하게 한다.

패키지 이름	*분류 근거 및 역할*
Product	패키지 내부 요소들은 상품 관리, 즉 상품의 등록, 검색 및 삭제에 대한 사용자의 요청을 논리적으로 처리하는 공통의 목적을 갖는다.
category	패키지 내부 요소들은 상품 카테고리, 즉 상품의 분류에 대한 등록, 검색 및 삭제에 대한 사용자의 요청을 논리적으로 처리하는 공통의 목적을 갖는다.
inventory	패키지 내부 요소들은 상품 재고의 조회 및 수정에 대한 시스템 사용자의 요청을 논리적으로 처리하는 공통의 목적을 갖는다.

- **Technical Service 계층**

상위 계층에서 이루어지는 논리적인 작업을 보조하는 책임을 갖는다. 기술적서비스와 프레임워크가 포함된다.

패키지 이름	*분류 근거 및 역할*
로그 기록	패키지 내부 요소들은 최종 사용자가 언제, 어디서, 어떻게 시스템을 이용했는지에 대한 정보를 기록하는 역할을 한다. 이 정보를 기반으로 시스템 에러, 최종 사용자 시스템 사용 기록 등 시스템 예외 상황에 대한 대처와 시스템에 대한 다양한 분석과 통계가 가능해진다.
영구 저장소	패키지 내부 요소들은 상위 계층에서 이루어지는 데이터 처리 결과를 저장하는 역할을 한다.
보안	패키지 내부 요소들은 상위 계층에서 이루어지는 활동에 대한 보안을 담당한다. 외부의 악의적인 사용자에 의해 상위 계층에서 사용중인 데이터가 가로채어지지 않도록 한다.

웹호스팅	패키지 내부 요소들은 인터넷 연결을 제공할 뿐 아니라, 일반적으로 데이터 센터에서 클라이언트 이용에 대한 임대 또는 소유하는 서버의 공간을 제공한다.

11.6 Logical Architecture Diagram

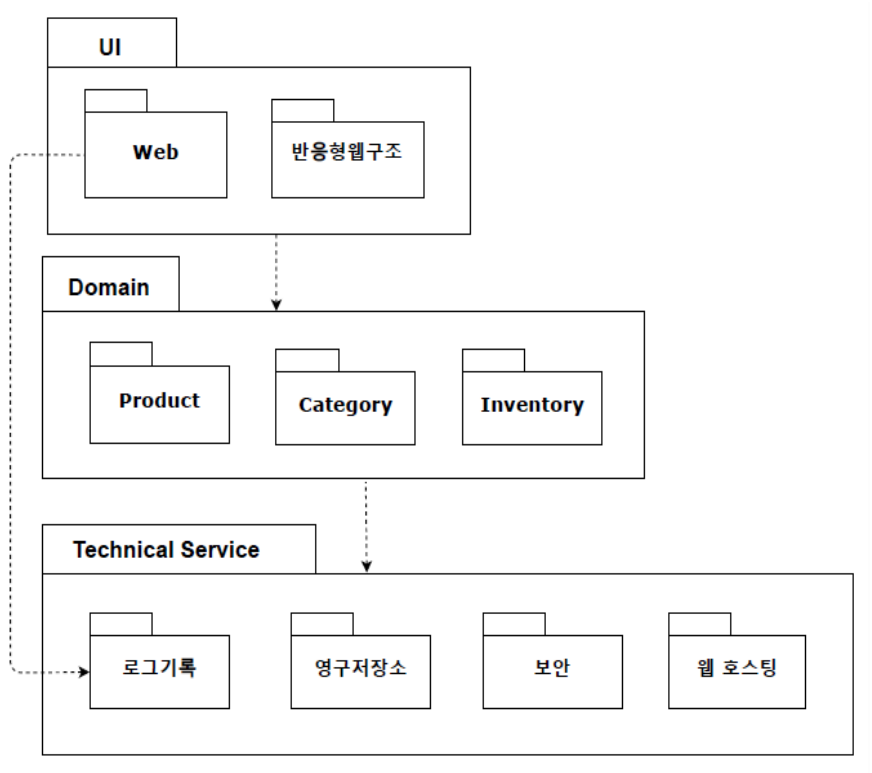


그림 17 논리적 아키텍처 다이어그램

12. Use Case Realization Design Model

12.1 Control information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Use Case Realization Design Model
Document owner	소프트조
Document identification no	12
Document status	elaboration

12.2 Revision History

버전	*일자*	*설명*	*저자*
Use Case Realization Design Model	2020.07.04	Use Case Realization 그림 작성	장유림, 정우택 박재영, 안창희
Use Case Realization Design Model v1	2020.07.09	Use Case Realization 그림 수정 및 설명 작성	장유림, 정우택 박재영
Use Case Realization Design Model v2	2020.07.12	Use Case Realization 디자인 클래스 다이어그램	소프트
Use Case Realization Design Model v3	2020.07.12	Use Case Realization 전체 디자인 클래스 다이어그램	소프트

12.3 Introduction

Use Case Design Model 에서는 앞서 만들어졌던 Use Case 에 기반하여 Use Case Realization 을 수행한다. 이 장에서는 Use Case 1 카테고리 관리의 카테고리 조회, 카테고리 등록, 카테고리 삭제에 대한 interaction diagram 과 design class diagram, Use Case 2 상품 관리의 상품 조회, 상품 등록, 상품 삭제에 대한 interaction diagram 과 design class diagram, Use Case 3 재고 관리의 재고 조회, 재고 수정에 대한 interaction diagram 과 design class diagram 을 다룬다. 각 Use Case 의 interaction diagram 과 design class diagram 을 기반으로 다음 장에 Source code 를 구성한다.

12.4 Communication Diagram

<UC1 카테고리 관리>

I. Interaction Diagram 1: 카테고리 조회(categoryName)

매개 변수로 받은 분류에 대하여 그 하위 분류를 조회한다.

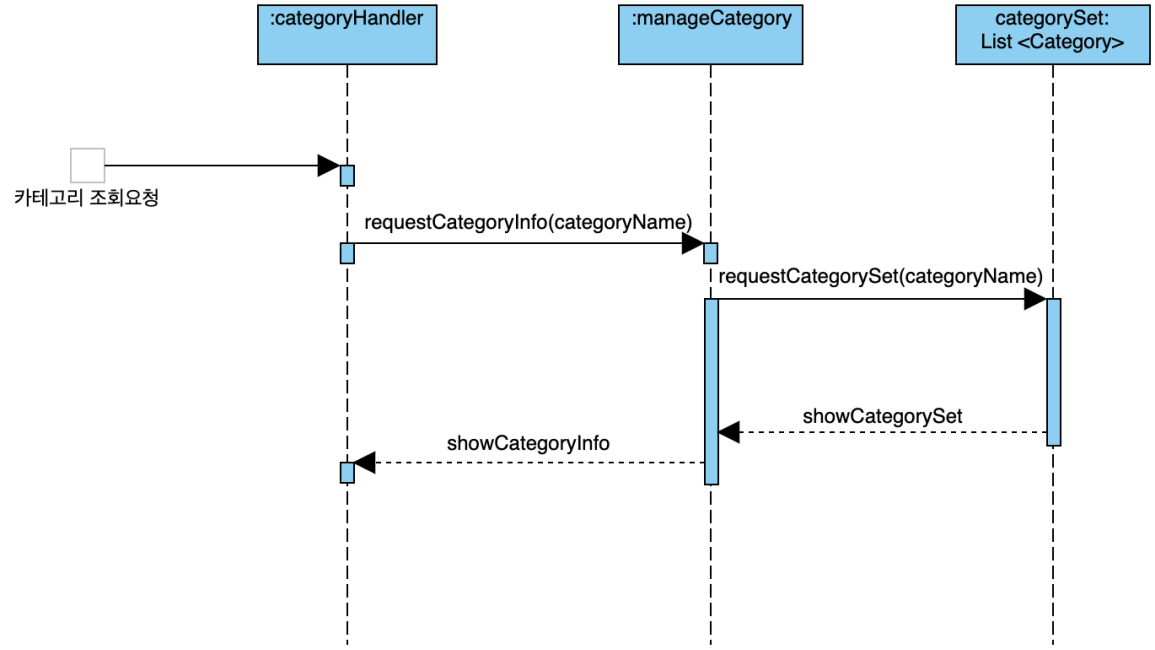


그림 18 manage category: 카테고리 조회

- Interaction Diagram 설명

1. [controller/high cohesion/pure fabrication pattern]

   시스템 오퍼레이션을 처리하는 책을 가지는 controller 인 categoryHandler 를 만들었다. Handler 가 많은 책임을 지지 않고 high cohesion 을 지키기 위해, category 만 handling 할 수 있는 categoryHandler 를 지정했으며, 이는 시스템 지원을 위한 인공적 클래스이다.

2. [indirection/low coupling, high cohesion/pure fabrication pattern]
   조회하고자 하는 정보를 CategorySet 이 갖고 있다. 그러나 controller 가 시스템의 수많은 CategorySet 과 각각 연관 관계를 맺고, 카테고리 정보를 조회하는 것은 low coupling, high cohesion 에 적합하지 않다. Item 정보를 가지는 manageCategory 를 두어, controller 와 categorySet 사이의 연결을 줄이고자 하였다.

3. [information expert pattern]
   requestCategorySet()은 사용자가 조회를 요청한 category 의 정보를 찾아 반환하는 것이다. category 의 정보는 manageCategory 가 가지고 있으므로, information expert pattern 에 의해, manageCategory 가 해당 책임을 갖는다.

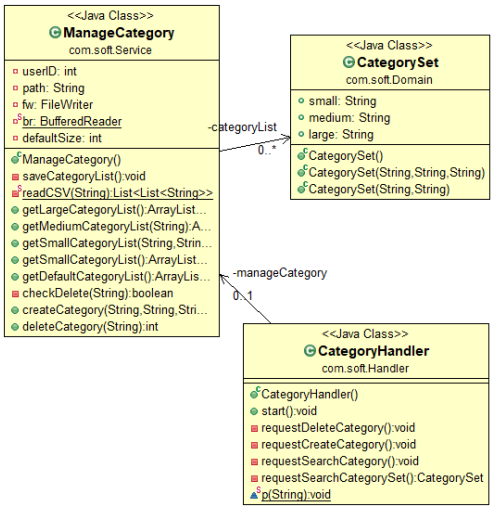


그림 19 manage Category: 카테고리조회

II. Interaction Diagram2

: 카테고리 등록(LargeCategoryName, MediumCategoryName, SmallCategoryName)

매개 변수로 받은 대분류, 중분류 하위에 새로운 소분류를 추가한다.

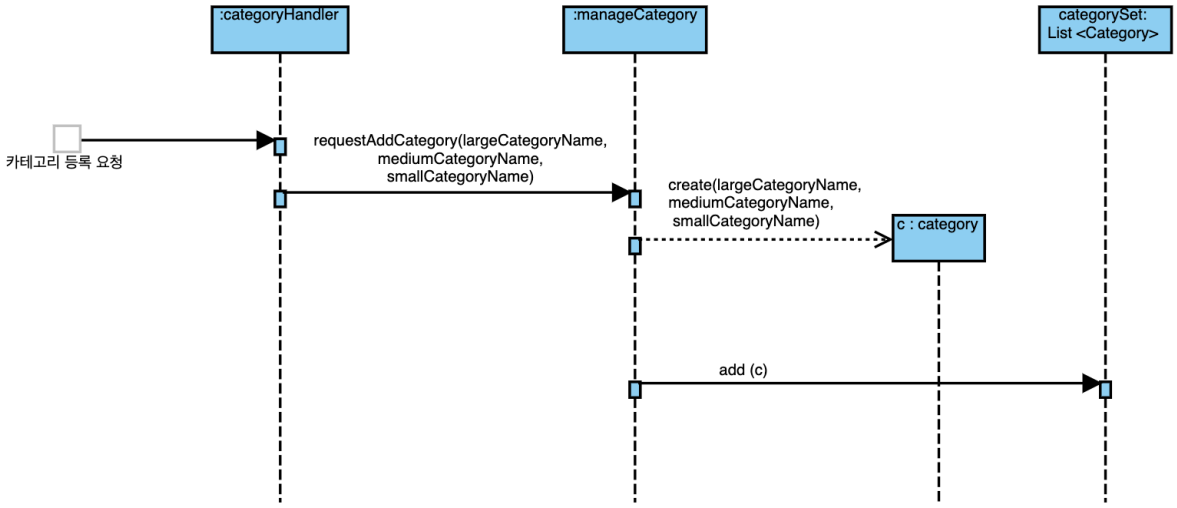


그림 20 manage category: 카테고리 등록

- Interaction Diagram 설명

1. [Information Expert pattern/low coupling/high cohesion]

   새로운 소분류를 추가하는 과정에서 manageCategory 인스턴스는 Controller 인 categoryHandler 에 의해 전달된 requestAddCategory 로 categorySet 을 전문적으로 관리하는 권한을 갖고, categoryHandler 의 low coupling 을 보장한다.

2. [Creator pattern]

   Information Expert pattern 을 가진 manageCategory 인스턴스는 소분류를 추가하기 위해 새로운 createSet 을 create 할 책임을 가진다.

- Design Class Diagram 2

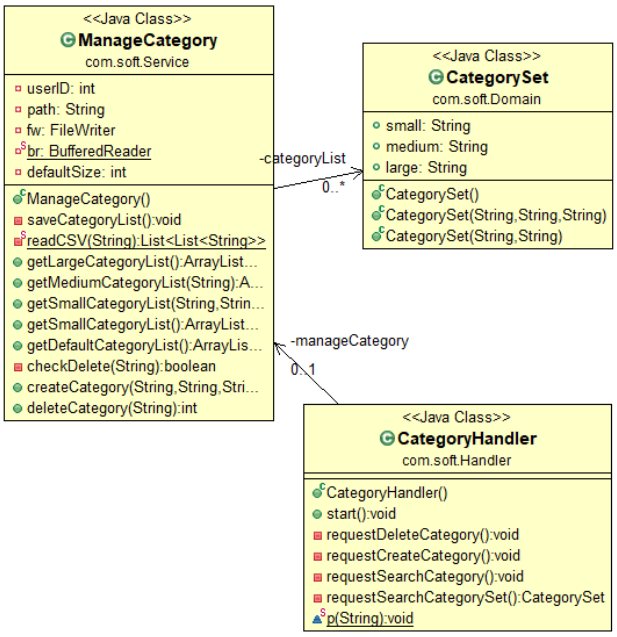


그림 21 manage category: 카테고리 등록

III. Interaction Diagram 3: 카테고리 삭제(SmallCategoryName)

매개변수로 받은 소분류를 삭제한다.

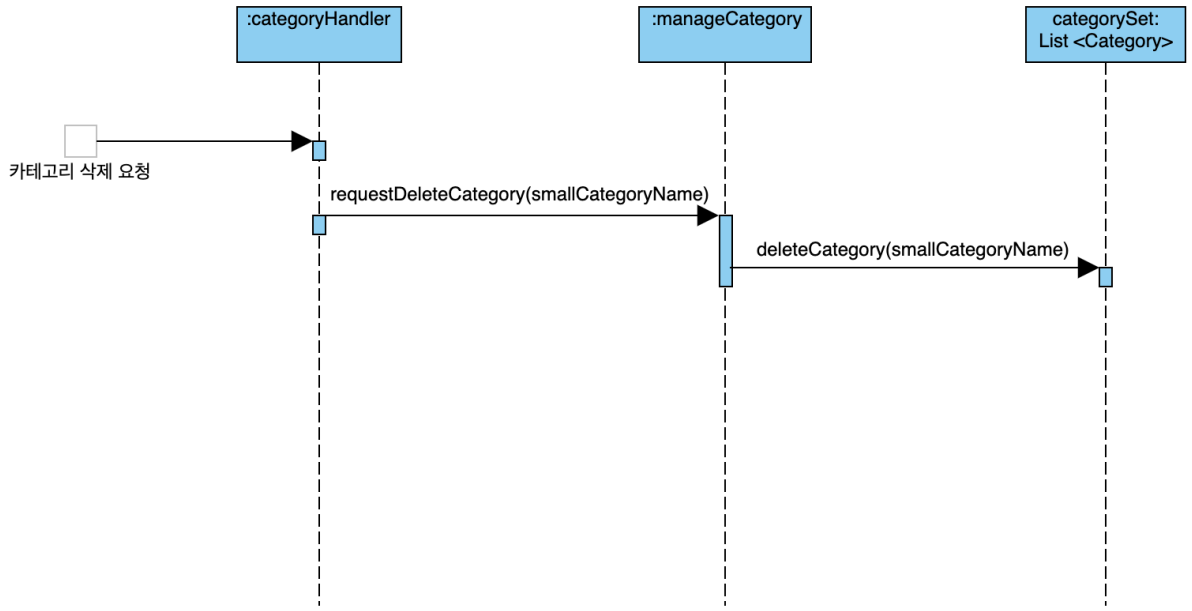


그림 22 manage category: 카테고리 삭제

- Interaction Diagram 설명

1. [information expert pattern]

모든 categoryset 의 정보는 manageCategory 가 가지고 있으므로, information expert pattern 에 의해, deleteCategory(categoryName)의 책임은 manageCategory 에게 부여된다.

- Design Class Diagram 3

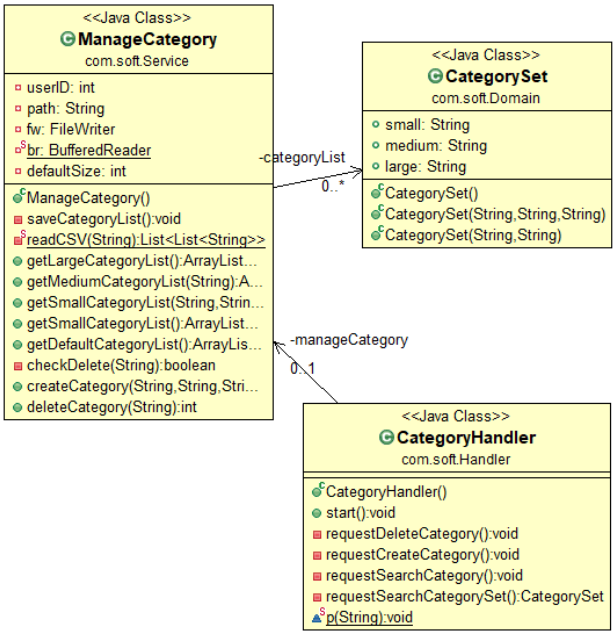


그림 23 manage category: 카테고리 삭제

<UC2 상품 관리>

I. Interaction Diagram1: 상품조회(ItemName)

매개변수로 받은 이름을 가지는 상품에 대한 상세 정보를 조회한다.

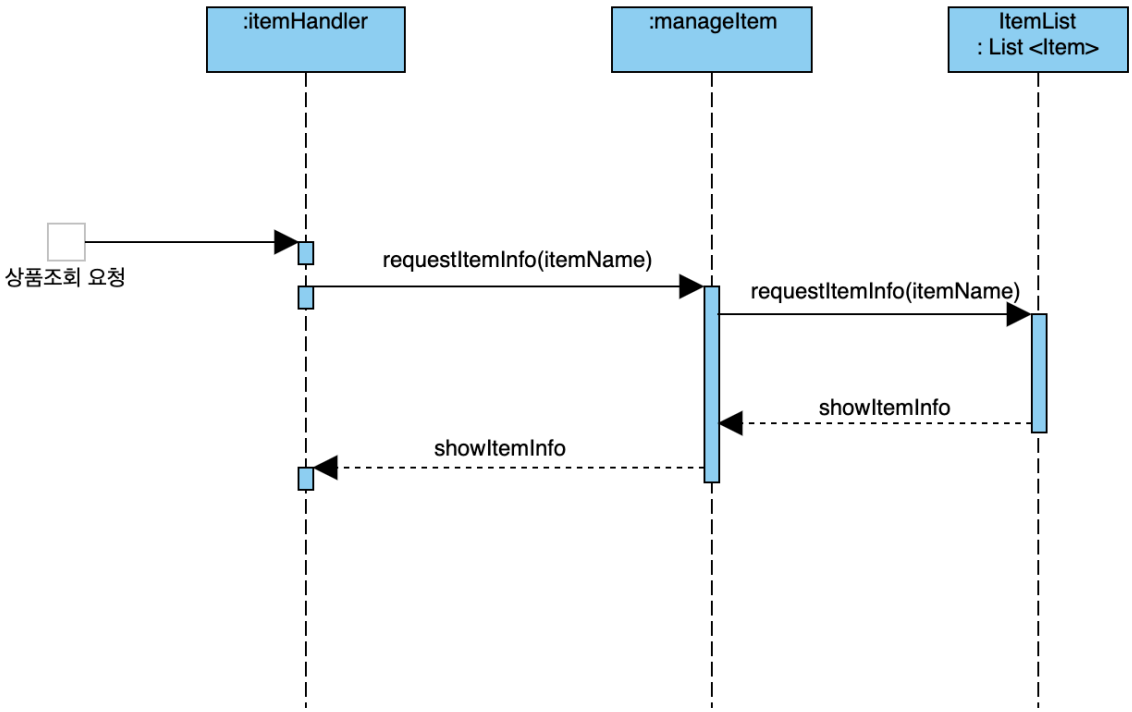


그림 24 manage product: 상품 조회

- Interaction Diagram 설명

1. [controller/high cohesion/pure fabrication pattern]

시스템 오퍼레이션 메시지를 받고 처리하는 책임을 맡을 controller 가 필요하다. 그러나 하나의 controller 에 너무 많은 책임을 부여하여, high cohesion 원칙을 위배하지 않도록, 각각의 유스케이스를 위해 서로 다른 controller 를 만드는 것을 선택하였다. 상품 관리 유스케이스를 위하여 Item Handler 를 만들었으며, 이는 시스템을 지원하기 위한 인공적인 클래스이다.

2. [indirection/low coupling, high cohesion/pure fabrication pattern]

조회하고자 하는 상품의 정보는 Item 이 갖고 있다. 그러나 controller 가 시스템의 수많은 Item 과 각각 연관 관계를 맺고, 상품의 정보를 조회하는 것은 low coupling, high

cohesion 에 적합하지 않다. Item 정보를 가지는 manage Item 를 두어, controller 와 Item 사이의 연결을 줄이고자 하였다.

3. [information expert pattern]

requestItemInfo()은 사용자가 조회를 요청한 Item 의 정보를 찾아 반환하는 것이다. Item 의 정보는 manage Item 이 가지고 있으므로, information expert pattern 에 의해, manage Item 이 해당 책임을 갖는다.

- Design Class Diagram 1

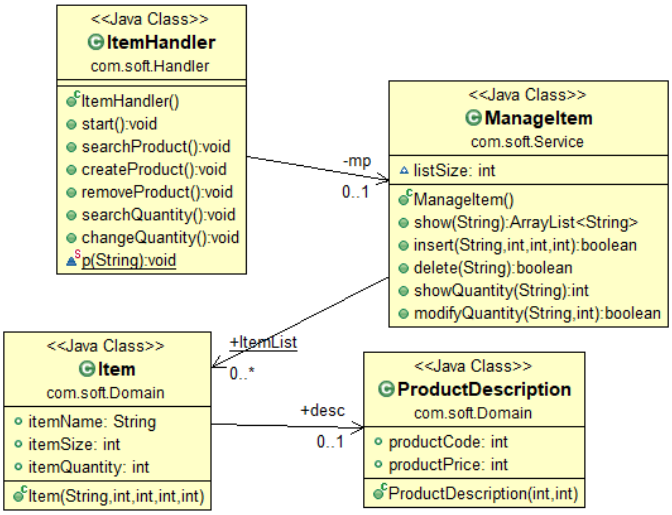


그림 25 manage product: 상품 조회

II. Interaction Diagram 2: 상품등록(ItemName, ItemSize, ItemQuantity, ProductPrice)

매개변수로 받은 정보를 바탕으로 새로운 상품을 등록한다.

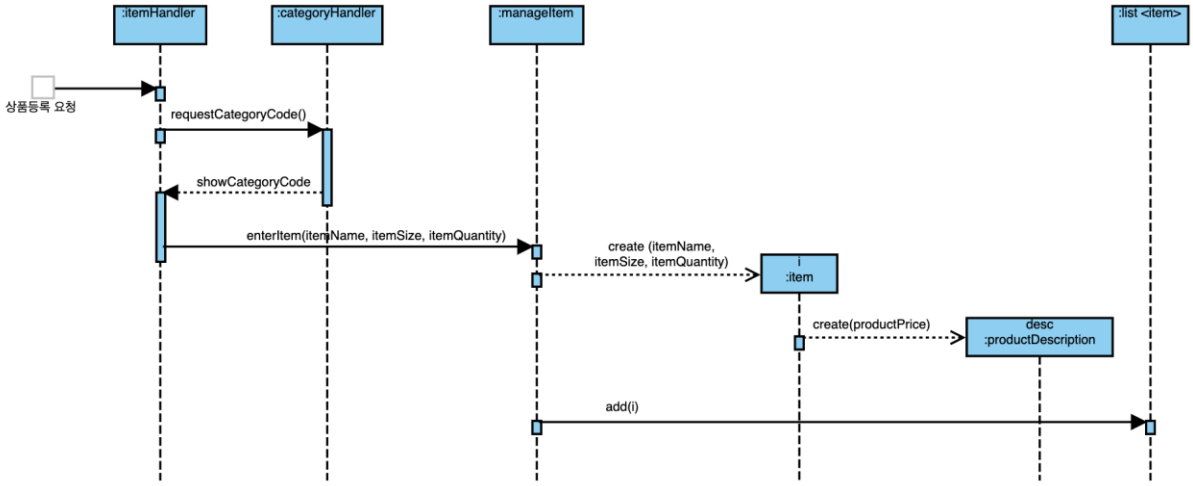


그림 26 manage product: 상품 등록

- Interaction Diagram 설명

1. [information expert pattern]

상품을 등록하기 위해서는, 상품이 등록될 분류 정보가 필요하다. 앞서 설명한 분류 조회 기능을 통해, 사용자가 상품을 등록할 카테고리를 이미 선택한 상황이라고 가정한다. 그렇다면, information expert 패턴에 의하여, 등록한 상품의 분류 정보는 category Handler 가 가지고 있을 것이므로, 상품을 등록하기 위한 분류 정보를 찾는 책임은 category Handler 에게 부여된다.

그리고 Item 의 정보는 manage Item 이 가지고 있으므로, information expert pattern 에 의해, enterItem()의 책임은 manage Item 에게 부여된다.

2. [creator pattern]

manage Item 은 Item 정보를 기록한다. 따라서 시스템에 새로 등록되는 상품인 Item 객체를 create 할 책임은 manage Item 이 가진다. 그리고 Product Description 은 Item 안에 포함되는 값이다. 이 역시 creator 패턴에 의하여. Product Description 을 create 할 책임은 Item 이 가진다.

- Design Class Diagram2

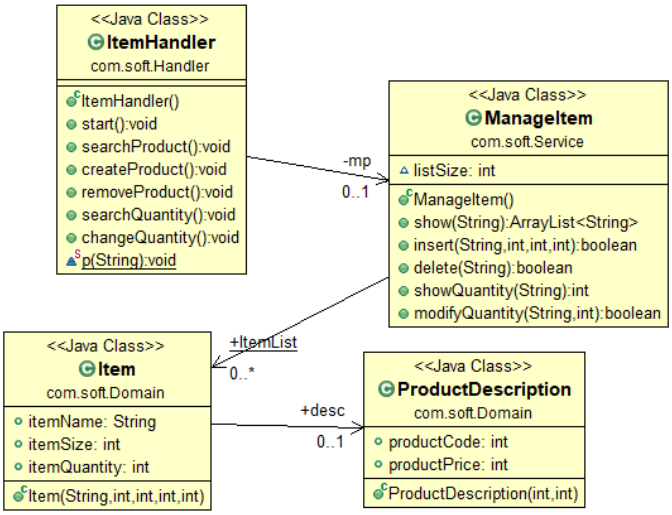


그림 27 manage product: 상품 등록

III. Interaction Diagram 3: 상품 삭제(ItemName)

매개 변수로 받은 이름을 가지는 상품을 삭제한다.

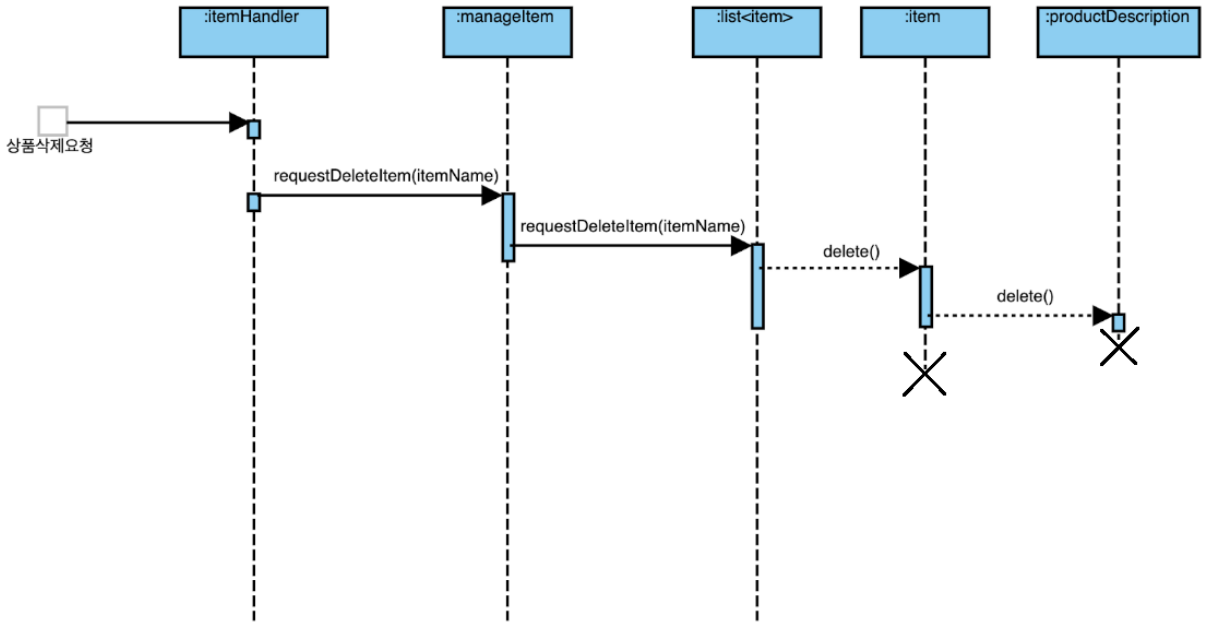


그림 28 manage Inventory: 상품 삭제

- Interaction Diagram 설명

1. [information expert pattern]

모든 Item 의 정보는 manage Item 이 가지고 있으므로, information expert pattern 에 의해, requestDeleteItem()의 책임은 manage Item 에게 부여된다.

- Design Class Diagram3

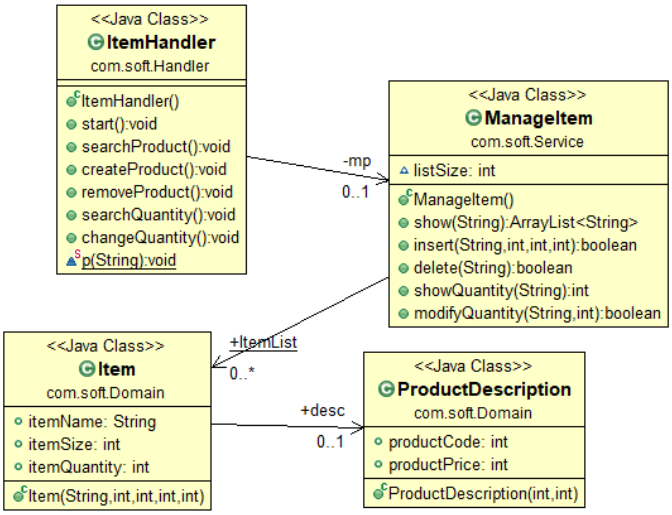


그림 29 manage Inventory: 상품 삭제

<UC3 재고관리>

I. Interaction Diagram1 : 재고조회(itemName)

매개변수로 받은 이름을 가지는 상품의 재고를 조회한다.

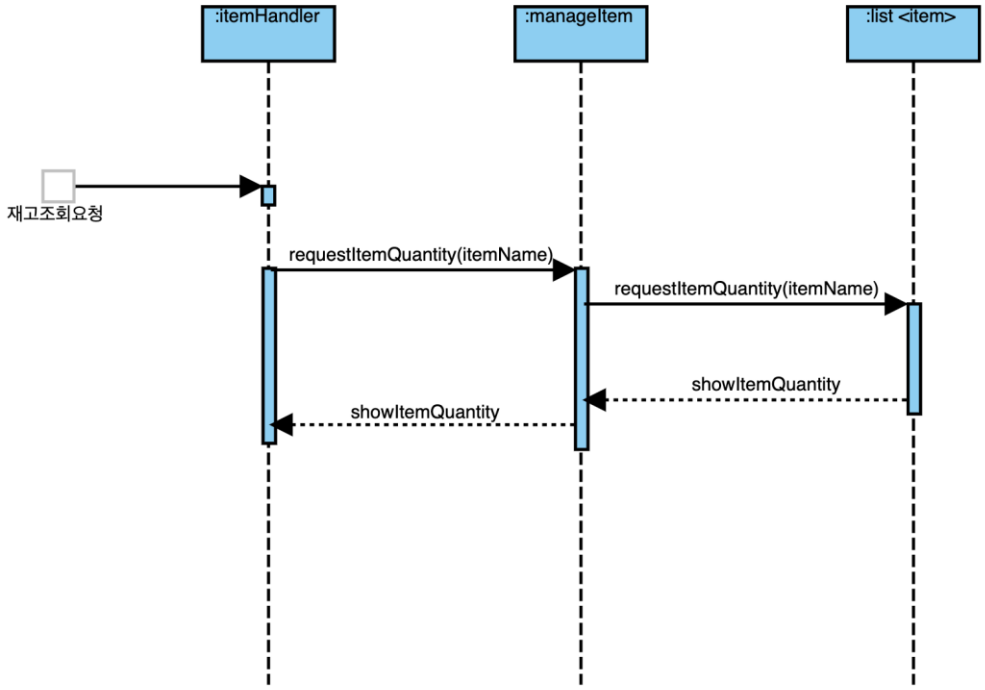


그림 30 manage Inventory: 재고 조회

- Interaction Diagram 설명

1. [controller/high cohesion/pure fabrication pattern]

시스템 오퍼레이션 메시지를 받고 처리하는 책임을 맡을 controller 가 필요하다. 그러나 하나의 controller 에 너무 많은 책임을 부여하여, high cohesion 원칙을 위배하지 않도록, 각각의 유스케이스를 위해 서로 다른 controller 를 만드는 것을 선택하였다. 그러나 재고를 관리하는 일은, 상품을 관리하는 것의 연장이라고 생각하여 재고 관리 유스케이스에서 상품 관리 유스케이스의 controller 인 itemHandler 를 재사용한다. itemHandler 는 시스템을 지원하기 위한 인공적인 클래스이다.

2. [indirection/low coupling, high cohesion/pure fabrication pattern]

조회하고자 하는 상품 재고 정보는 Item 이 갖고 있다. 그러나 controller 가 시스템의 수많은 Item 과 각각 연관 관계를 맺고, 상품 재고를 조회하는 것은 low coupling, high cohesion 에

적합하지 않다. Item 정보를 가지는 manageItem 를 두어, controller 와 Item 사이의 연결을 줄이고자 하였다.

3. [information expert pattern] requestItemQuantity(itemName)는 사용자가 재고 조회를 요청한 Item 의 재고 정보를 찾아 반환하는 것이다. Item 의 정보는 manage item 가 가지고 있으므로, information expert pattern 에 의해, manageItem 가 해당 책임을 갖는다.

- Design Class Diagram 1

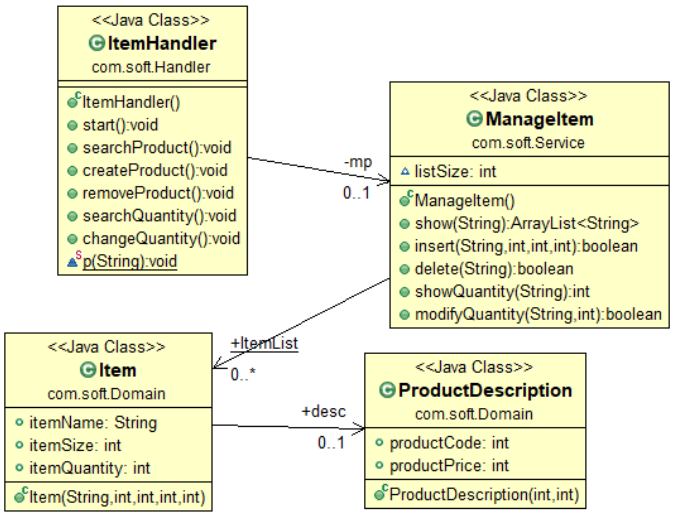


그림 31 manage Inventory: 재고 조회

II. Interaction Diagram2: 재고 수정(itemName,itemQuantity)

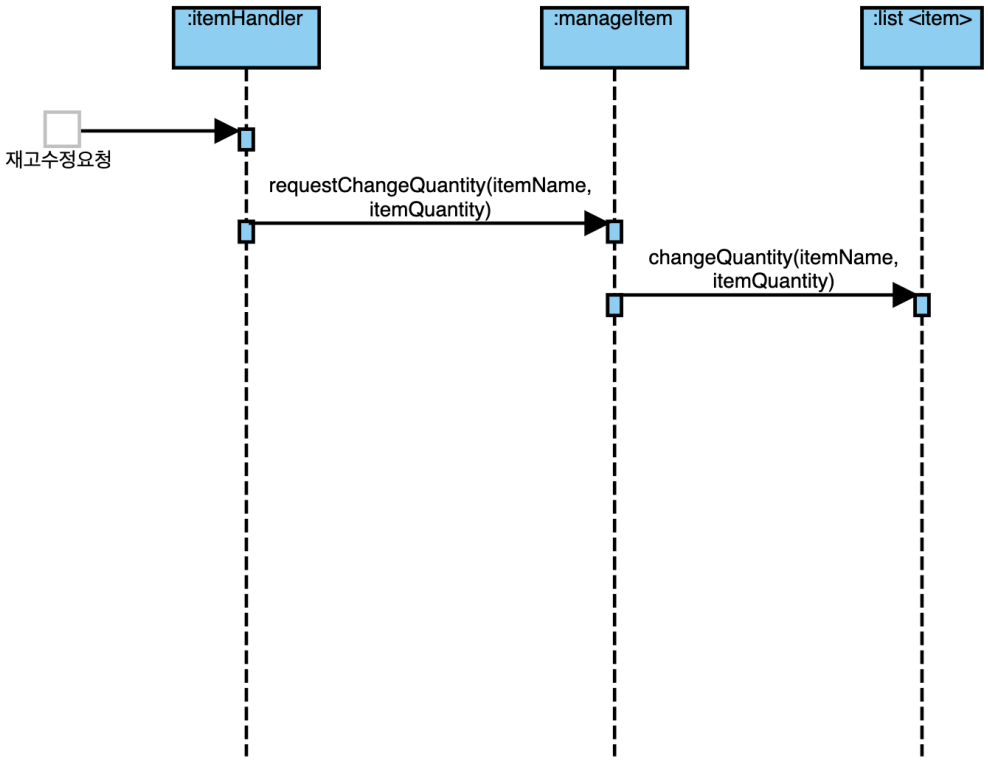


그림 32 manage Inventory: 재고 수정

- Interaction Diagram 설명

1. [information expert pattern]

   모든 Item 의 정보는 manage Item 이 가지고 있으므로, information expert pattern 에 의해, changeQuantity(S)의 책임은 manageItem 에게 부여된다.

- Design Class Diagram2

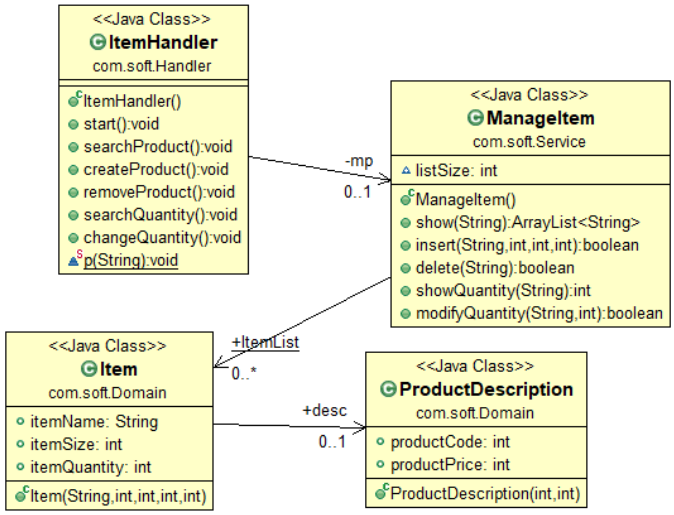


그림 33 manage Inventory: 재고 수정

[전체 Design Class Diagram]

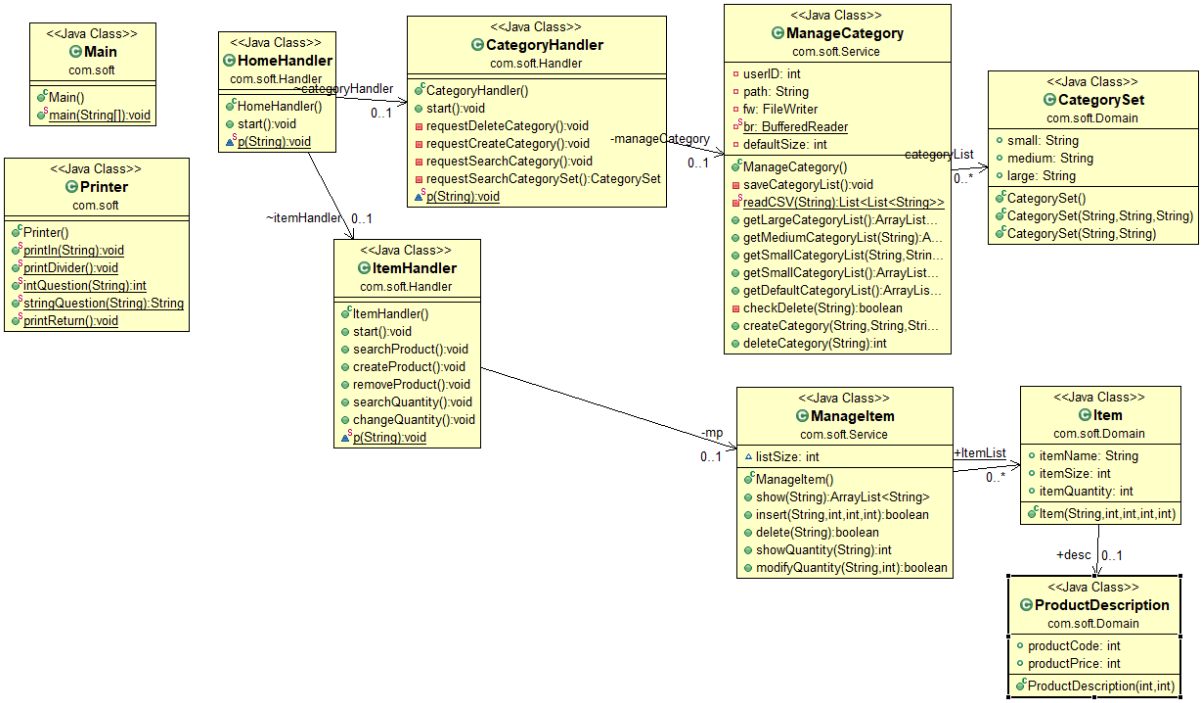


그림 34 전체 Design Class Diagram

13. Source code

13.1. Control Information

Document title	온라인 쇼핑몰 상품 관리 시스템의 Source Code
Document owner	소프트조
Document identification no	13
Document status	Elaboration

13.2. Revision History

버전	*일자*	*설명*	*저자*
Source Code	2020.07.10	카테고리 관리 소스코드 구현	안창희
Source Code v1	2020.07.11	카테고리 관리 소스코드 수정 및 상품관리 기능 구현	안창희, 이연주
Source Code v2	2020.07.12	재고관리 소스코드 구현 및 소스코드 작성 근거 작성	안창희, 이연주
Source Code v3	2020.07.12	카테고리 관리 및 상품 관리 소스코드 수정	소프트

13.3. Source Code 및 설명

1) Main Class

설명

ShoeRack 시스템 실행

```
package com.soft;

import com.soft.Handler.HomeHandler;

public class Main {
   public static void main(String[] args) {
       HomeHandler home = new HomeHandler();
       home.start();
   }
}
```

2) HomeHandler class

설명

ShoeRack 홈화면

```
package com.soft.Handler;
import com.soft.Printer;
public class HomeHandler {
    CategoryHandler categoryHandler =null;
    ItemHandler itemHandler = null;
    public void start(){
        int userInput = -1;
        Printer.printDivider();
        Printer.println("상품관리시스템 ShoeRack 의 홈화면입니다.");
        try{
            while(true) {
                Printer.printDivider();
                Printer.println("원하시는 메뉴를 골라 주세요.");
                p("1. 카테고리 관리");
                p("2. 상품 관리");
                p("3. 재고 관리");
                p("4. 프로그램 종료");
                userInput = Printer.intQuestion("입력");
                switch(userInput){
                    case 1:
                        categoryHandler = new CategoryHandler();
                        categoryHandler.start();
                        break;
                    case 2:
                        itemHandler = new ItemHandler();
                        itemHandler.start();
                        break;
                    case 3:
                        ItemHandler itemHandler = new ItemHandler();
                        itemHandler.start();
                        break;
                    case 4:
                    case 'q':
                    case 'Q':
```

```
                        Printer.printDivider();
                        p("시스템이 종료되었습니다. 감사합니다");
                        break;
                    default:
                        System.out.println("잘못 입력하셨습니다.");
                }
            }
        }catch (Exception e) {
        }
    }
    static void p(String x) {
        Printer.println(x);
    }
}
```

- **Printer Class**

설명

본 클래스에서는 시스템 구현시 간단하게 표준 출력 함수를 사용하기 위해 미리 정의해 두었다.

- Println 함수 : String x 출력
- PrintDiver 함수: 구분선 출력
- IntQuestion 함수: String x 를 출력하고 사용자에게 정수를 입력받음.
- stringQuestion 함수: String x 를 출력하고 사용자에게 String 을 입력받음.
- PrintReturn 함수: 개행 출력

```
package com.soft;

import java.util.Scanner;

public class Printer {
   public static void println(String x) {
       System.out.println(x);
   }
   public static void printDivider() {
       System.out.println("----------------------------------------");
   }
   public static int intQuestion(String x) {
       System.out.print(x + "  :  ");
       Scanner scan = new Scanner(System.in);
       int response  =  Integer.parseInt(scan.nextLine());
       return response;
   }
   public static String stringQuestion(String x) {
       System.out.print(x + "  :  ");
       Scanner scan = new Scanner(System.in);
       String response  =  scan.nextLine();
       return response;
   }
   public static void printReturn() {
       System.out.printf("\n");
   }
}
```

<Usecase 1. 카테고리 관리>

1) CategoryHandler class

Categoryhandler 는 카테고리 관리의 주요 기능인 카테고리 조회, 카테고리 생성, 카테고리 삭제에 대한 요청을 받고 관련 메세지를 전달한다. 주요 기능 3 가지에 대한 함수를 나누어 구현하여 관심사 분리하였다.

- **Start 함수**

설명

카테고리 관리를 시작하기 위해 필요한 함수이다. 변수 userInput 은 이용자가 카테고리 관련 주요 업무 3 가지 중 하나를 결정하여 입력한 정수를 저장한다.

- UserInput = 1 일 경우

카테고리 조회 요청을 처리하는 requestSearchCategory 함수를 호출한다.

- UserInput = 2 일 경우

카테고리 생성 요청을 처리하는 requestCreateCategory 함수를 호출한다.

- UserInput = 3 일 경우

카테고리 삭제 요청을 처리하는 requestDeleteCategory 함수를 호출한다.

- UserInput = 4 일 경우

상품관리시스템 홈화면으로 돌아가기를 원하는 요청이다. 상품관리시스템의 홈화면에 대한 요청을 처리하는 homeHandler 의 start 함수를 호출한다.

```
public void start() {
        int userInput = -1;
        try {
            while (true) {
                Printer.printDivider();
                Printer.println("원하시는 메뉴를 골라 주세요.");
                p("1. 카테고리 조회");
                p("2. 카테고리 생성");
                p("3. 카테고리 삭제");
                p("4. 돌아가기");
                userInput = Printer.intQuestion("입력");
                switch (userInput) {
                    case 1:
                        requestSearchCategory();
                        break;
                    case 2:
                        requestCreateCategory();
                        break;
                    case 3:
                        requestDeleteCategory();
                        break;
                    case 4:
```

```
                    case 'q':
                    case 'Q':
                        HomeHandler homeHandler = new HomeHandler();
                        homeHandler.start();
                        break;
                    default:
                        System.out.println("잘못 입력하셨습니다.");
                }
            }
        } catch (Exception e) {
        }
    }
```

- **RequestSearchCategory 함수**

설명

카테고리 조회 요청을 처리하기 위해 필요한 함수이다. 시스템은 RequestSearchCategorySet 함수를 통해 이용자에게 대분류, 중분류를 입력 받아 해당 CategorySet 의 소분류를 보여준다. 카테고리 조회 요청 처리 이후 시스템은 이용자에게 다른 카테고리를 볼 것인지, 카테고리 홈화면으로 갈 것인지 묻는다. 변수 userInput 은 이용자가 카테고리 관련 주요 업무 3 가지 중 하나를 결정하여 입력한 정수를 저장한다.

- UserInput = 1 일 경우:: 카테고리 조회 계속

카테고리 조회를 계속 하기 위해 requestSearchCategory 함수를 호출한다.

- UserInput = 2 일 경우

카테고리 홈화면으로 가기 위해 start 함수를 호출한다.

- default 일 경우

입력이 잘못되었을 때, 시스템은 '잘못 입력하셨습니다'라는 메세지를 보여준다.

```
private void requestSearchCategory() {
        int userInput = -1;
        Printer.printDivider();
        Printer.println("카테고리 조회를 진행합니다.");
        CategorySet userCategorySet = requestSearchCategorySet();
        Printer.printDivider();
        System.out.printf("다른 카테고리 목록을 보시겠습니까?");
        Printer.printReturn();
        p("1. 다른 카테고리 보기");
        p("2. 카테고리 홈화면으로 가기");
        userInput = Printer.intQuestion("입력");
        switch (userInput) {
            case 1:
                requestSearchCategory();
            case 2:
                start();
            default:
                System.out.println("잘못 입력하셨습니다.");
        }
    }
```

- **RequestSearchCategorySet 함수**

설명

본 함수는 카테고리 조회, 카테고리 생성시 호출된다.

CategorySet 은 {대분류, 중분류, 소분류} 형태로 구성되어 있어, 소분류에 접근하기 위해서 순차적으로 대분류, 중분류에 대한 접근이 우선되어야한다.

이에 따라 본 함수를 통해 먼저 시스템은 대분류, 중분류, 소분류 모두 비어있는 userCategorySet 을 만든다. 그 다음 getLargeCategoryList 함수를 통해 시스템에 등록된 대분류를 largeCategory 에 불러온다. 그리고 사용자에게 등록되어 있는 대분류를 보여준다. 시스템은 이용자에게 대분류를 입력 받고 변수 userInput 에 저장한다. UserInput 은 LargeCategory 의 순서번호(인덱스)으로 시스템은 largeCategory 의 해당 인덱스에 있는 값을 변수 large 에 저장한다. 중분류를 찾는 과정도 다음과 같다. 찾은 중분류는 medium 에 저장된다. 시스템은 large, medium 와 관련된 소분류를 보여준다.

본 함수는 카테고리 생성 요청과 관련되어 있기 때문에 이용자에게 입력 받은 대분류 large, 중분류 medium 로 생성된 CategorySet 을 반환한다. 즉, requestCreateCategory 함수는 해당 CategorySet 을 활용하여 사용자에게 입력받은 소분류를 처리하여 카테고리를 저장한다.

```
private CategorySet requestSearchCategorySet() {
        CategorySet userCategorySet = new CategorySet();
        Printer.printDivider();
        int userInput = -1;
        Printer.println("대분류를 골라 주세요.");
        manageCategory = new ManageCategory();
        ArrayList<String> largeCategory =
manageCategory.getLargeCategoryList();
        int index = 0;
        for (String data : largeCategory) {
            if(index == 0){
                index++;
                continue;
            }
            data = ". " + data;
            p(index++ + data);
        }
        userInput = Printer.intQuestion("입력");
        userCategorySet.large = largeCategory.get(userInput);
        Printer.printDivider();
        Printer.println("중분류를 골라 주세요.");
        ArrayList<String> mediumCategory =
manageCategory.getMediumCategoryList( userCategorySet.large);
        index = 0;
        for (String data : mediumCategory) {
```

```
            data = ". " + data;
            p(++index + data);
        }
        userInput = Printer.intQuestion("입력");
        userCategorySet.medium = mediumCategory.get(userInput
- 1);
        Printer.printDivider();
        Printer.println("소분류 현황입니다.");
        Printer.printDivider();
        ArrayList<String> smallCategory =
manageCategory.getSmallCategoryList(userCategorySet.large,
userCategorySet.medium);
        for (String data : smallCategory) {
            p(data);
        }
        return userCategorySet;
    }
```

2) ManageCategory class

본 클래스는 카테고리 관리의 주요 기능 카테고리 조회, 카테고리 생성, 카테고리 삭제 와 관련된 요청이 올 때 카테고리 데이터를 다루는 부분이다. 요청에 따라 데이터를 처리하기 위해 필요한 함수들을 분리하여 구현하였다.

- **ManageCategory Class 의 필요 변수**

설명

다음은 ManageCategory Class 에서 정의한 멤버 변수이다.

- **UserID**: 현재 시스템을 사용하는 이용자의 ID 이다. 여기서 userID 가 1 인 사용자가 시스템을 이용 중인 것으로 가정한다.
- **path:** 카테고리 데이터가 저장된 파일의 상대경로이다.
- **Fw:** csv 파일을 쓰기 위한 fileWriter 이다.
- **Br:** csv 파일을 한 줄씩 읽기 위한 BufferedReader 이다.
- **CategoryList:** 현재 시스템에 등록된 모든 CategorySet 이 저장된다.
- **DefaultSize:** 시스템에서 제공하는 기본 CategorySet 의 개수이다. 총 34 개의 CategorySet 을 제공한다.

```
private int userID = 1;
private String path = "./data/CategoryList.csv";
private FileWriter fw = null;
private static BufferedReader br = null;
private ArrayList<CategorySet> categoryList = new ArrayList<>(999);
private int defaultSize = 34;
```

- **ManageCategory 생성자**

설명

카테고리 관리 기능이 시작하면 ManageCategory 클래스는 saveCategoryList 함수로 시스템에 등록된 모든 CategorySet 을 멤버변수 CategoryList 에 저장한다.

```
public ManageCategory(){
   saveCategoryList();
}
```

- **SaveCategoryList 함수**

설명

본 함수의 역할은 시스템에 등록된 모든 CategorySet 을 멤버변수 CategoryList 에 저장하는 것이다.

```
 private void saveCategoryList(){

         List<List<String>> list = readCSV(path);

        for(List<String> line : list) {

            CategorySet categorySet = new CategorySet(line.get(2), line.get(3),
 line.get(4));

            categoryList.add(categorySet);

        }

    }
```

- **ReadCSV 함수**

설명

본 함수의 역할은 categoryList.csv 를 row 단위 리스트를 반환하는 것이다.

CSV 파일은 하나의 DATA Row(행)가 ,(콤마)단위로 구분한다. Csv 파일에서 Row(행) 단위로 읽어와서 ,(콤마)로 나눈 리스트인 linelist 는 list 는 저장된다. List 를 반환한다.

```
 Private static List<List<String>>readCSV(String path) {
```

```
        List<List<String>> list = new ArrayList<>();
        try {
            File csv = new File(path);
            br = new BufferedReader(new FileReader(csv));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        Charset.forName("UTF-8");
        String line = "";
        while (true) {
            try {
                if (!((line = br.readLine()) != null)) break;
            } catch (IOException e) {
                e.printStackTrace();
            }
            String[] token = line.split("\\s|,|\t", -1);
            List <String> lineList= new ArrayList<>(Arrays.asList(token));
            list.add(lineList);
        }
        return list;
    }
```

- **CreateCategory 함수**

설명

본 함수의 역할은 categorySet 을 생성하는 것이다.

인자 large, medium, small 으로 categorySet 인스턴스를 생성한다.
시스템에 등록된 모든 CategorySet 을 관리하는 CategorySetList 에 categorySet 이 포함되어있는지 확인한다.
`categoryList.contains(categorySet)`

만약 categorySet 인스턴스가 categoryList 에 포함되어있을 때, 생성이 불가하다는 의미로 false 를 반환한다.
fileWriter 인 Fw 로 CategoryList.csv 파일에 ,(콤마)를 구분자로 현재 CategorySet 의 개수+1

(categoryList.size()+1), userID, large, medium, smalll 를 추가한다.

```
   public boolean createCategory(String large, String medium, String small) throws
IOException {
       try {
           fw = new FileWriter(path, true);
       } catch (IOException e) {
           e.printStackTrace();
       }
       CategorySet categorySet = new CategorySet(large, medium, small);
       if ((categoryList.contains(categorySet))) {
           return false;
       }
       fw.append(String.valueOf(categoryList.size()+1));
       fw.append(',');
       fw.append(String.valueOf(userID));
       fw.append(',');
       fw.append(large);
       fw.append(',');
       fw.append(medium);
       fw.append(',');
       fw.append(small);
       fw.append('\n');
       fw.flush();
       fw.close();
       saveCategoryList();
       return true;
   }
```

- **DeleteCategory 함수**

설명

본 함수의 역할은 이용자가 입력한 소분류를 삭제하는 것이다.

소분류 small 가 삭제가능한지 확인

삭제할 소분류인 small 이 포함된 category 를 제외한 값들을 저장

```
   public int deleteCategory(String small) throws IOException {
       PrintWriter pw = null;
       File csv = new File(path);
       try {
           br = new BufferedReader(new FileReader(csv));
       } catch (IOException e) {
           e.printStackTrace();
       }

       //소분류 small 가 삭제가능한지 확인
       ArrayList <String> savedSmallCategoryList = getSmallCategoryList();
       if(!checkDelete(small)){
           return -1;
```

```
        }
        if(!savedSmallCategoryList.contains(small)){
            return -2;
        }

        // 삭제할 소분류인 small 이 포함된 category 를 제외한 값들을 저장
        String line = "";
        ArrayList <String> smallCategoryList = new ArrayList<>();
        while (true) {
            try {
                if (!((line = br.readLine()) != null)) break;
            } catch (IOException e) {
                e.printStackTrace();
            }
            if(line.contains(small)) {
                continue;
            }
           smallCategoryList.add(line);
        }
        br.close();

        // 삭제할 소분류가 제외된 카테고리 데이터를 CategoryList.csv 파일에 쓴다.
        pw = new PrintWriter(new BufferedWriter(new FileWriter(csv)));
        for (String smallCategory : smallCategoryList){
            pw.println(smallCategory);
        }
        Objects.requireNonNull(pw).close();
        return 0;
    }
}
```

- **DeleteCheck 함수**

설명

본 함수는 소분류 targetSmallCategory 가 삭제가 가능한지 확인한다.

GetDefaultCategoryList 함수를 통해 시스템에서 제공하는 기본 소분류 목록을 smallCategoryList 에 저장한다.

SmallCategoryList 에 targetSmallCategory 가 포함되는 경우는 **이용자가 삭제하려는 소분류인 targetSmallCategory 가 기본 제공 소분류라는 의미다. 삭제할 수 없다는 의미로 False 가 반환시킨다. 이 조건에 걸리지 않으면 targetSmallCategory 를 삭제할 수 있다는 의미로 True 가 반환된다.**

```
    private boolean checkDelete(String targetSmallCategory) {
        ArrayList<String> smallCategoryList = getDefaultCategoryList();
```

```
        for(String smallCategory: smallCategoryList){
            if(smallCategory.equals(targetSmallCategory)){
                return false;
            }
        }
        return true;
    }
```

- **GetDefaultCategoryList 함수**

설명

본 함수는 시스템에서 기본으로 제공하는 소분류 리스트를 구할 때 쓰인다.

시스템에서 기본으로 제공되는 소분류는 34 개이다.(defaultSize)

DefaultSize(34) 만큼 반복하여 등록된 모든 categorySet, 즉 **categorySetList 에 저장되어 있는 CategoryList 의 CategorySet 의 소분류 small** 를 smallCategoryList 에 저장한다.

모든 소분류를 중복 없이 defaultSmallCategoryList 에 저장하여 이를 반환한다. `smallCategoryList.contains(categorySet.small)` 을 통해서 중복되는 소분류를 제거한다.

```
    public ArrayList<String> getDefaultCategoryList() {
        ArrayList <String> smallCategoryList = new ArrayList<>();
        ArrayList <String> defaultSmallCategoryList = new ArrayList<>();
        for(int i = 1; i<defaultSize-1; i++){
            smallCategoryList.add(categoryList.get(i).small);
        }
        for(String small: smallCategoryList){
            if (defaultSmallCategoryList.contains(small)){
                continue;
            }
            defaultSmallCategoryList.add(small);
        }
        return defaultSmallCategoryList;
    }
```

- **GetSmallCategoryList 함수**

설명

시스템에 등록된 모든 categorySet, 즉 **categorySetList** 의 모든 소분류를 중복 없이

smallCategoryList 에 저장한다. `smallCategoryList.contains(categorySet.small)` 을 통해서 중복값을 제거한다.

```
    public ArrayList<String> getSmallCategoryList() {
       ArrayList <String> smallCategoryList = new ArrayList<>();
       for(CategorySet categorySet: categoryList){
    if  (smallCategoryList.contains(categorySet.small)||categoryList.indexOf(categorySet)
== 0){
                continue;
            }
            smallCategoryList.add(categorySet.small);
       }
       return smallCategoryList;
   }
```

- **GetMediumCategoryList 함수**

설명 시스템에 등록된 모든 categorySet, 즉 **categorySetList** 의 모든 중분류를 mediumCategoryList 에 저장하고 TreeSet 으로 largetCategoryList 의 중복값을 제거한다.

```
public ArrayList<String> getMediumCategoryList(String large) {
       ArrayList <String>mediumCategoryList = new ArrayList<>();
       for(CategorySet categorySet: categoryList){
           if(categorySet.large.equals(large)) {
               Printer.println(categorySet.medium);
               mediumCategoryList.add(categorySet.medium);
           }
       }
       TreeSet <String> mediumCategorySet =new TreeSet<>(mediumCategoryList);
       mediumCategoryList = new ArrayList<>(mediumCategorySet);
       return mediumCategoryList;
   }
```

- **GetLargeCategoryList 함수**

설명 시스템에 등록된 모든 categorySet, 즉 **categorySetList** 의 모든 대분류를 largeCategoryList 에 저장하고 TreeSet 으로 largetCategoryList 의 중복값을 제거한다.

```
public ArrayList<String> getLargeCategoryList() {
       ArrayList <String>largeCategoryList = new ArrayList<>();
       for(CategorySet categorySet: categoryList){
           largeCategoryList.add(categorySet.large);
       }
       TreeSet <String> largeCategorySet =new TreeSet<>(largeCategoryList);
       largeCategoryList = new ArrayList<>(largeCategorySet);
       return largeCategoryList;
   }
```

<Usecase 2. 상품 관리>

1) Item 클래스

설명

Item 은 itemName (상품명), itemSize (상품사이즈), itemQuantity(상품재고량), productPrice(상품가격), productCode(상품코드)로 구성된다. 여기서 productPrice 와 productCode 는 Product Description 의 멤버변수이다.

```
public class Item {
   public String itemName;
   public int itemSize;
   public int itemQuantity;
   public ProductDescription desc;

   public Item(String itemName, int itemSize, int itemQuantity, int productPrice, int
productCode) {
      this.itemName = itemName;
      this.itemSize = itemSize;
      this.itemQuantity = itemQuantity;
      this.desc = new ProductDescription(productPrice, productCode);
   }

}
```

2) Product Description 클래스

설명

Product **Description** 의 멤버 변수는 총 2 개로 구성된다.

- productCode: 상품코드
- productPrice : 가격

```
public class ProductDescription {
   public int productCode;
    public int productPrice;
```

```
    public ProductDescription(int productCode, int productPrice)
    {
        this.productCode = productCode;
        this.productPrice = productPrice;
    }
}
```

3) ItemHandler class

ItemHandler 는 상품조회, 상품등록, 상품삭제에 대한 요청을 받고 관련 메세지를 전달한다. 주요 기능 3 가지에 대한 함수를 나누어 구현하여 관심사 분리하였다.

- **Start 함수**

설명

상품 관리를 시작하기 위해 필요한 함수이다. 변수 userInput 은 이용자가 상품 관련 주요 업무 3 가지 중 하나를 결정하여 입력한 정수를 저장한다.

- UserInput = 1 일 경우

상품 조회 요청을 처리하는 requestSearchProduct 함수를 호출한다.

- UserInput = 2 일 경우

상품 등록 요청을 처리하는 requestCreateProduct 함수를 호출한다.

- UserInput = 3 일 경우

상품 삭제 요청을 처리하는 requestDeleteProduct 함수를 호출한다.

- UserInput = 4 일 경우

상품관리시스템 홈화면으로 돌아가기를 원하는 요청이다. 상품관리시스템의 홈화면에 대한 요청을 처리하는 homeHandler 의 start 함수를 호출한다.

```
public void start(CategoryHandler C) {
      int userInput = -1;
      try {
         while (true) {
            Printer.printDivider();
            Printer.println("원하시는 메뉴를 골라 주세요.");
            p("1. 상품 조회");
            p("2. 상품 등록");
            p("3. 상품 삭제");
            p("4. 돌아가기");
            userInput = Printer.intQuestion("입력");
            switch (userInput) {
            case 1:
               searchProduct();
               break;
            case 2:
               createProduct(C);
```

```
                break;
            case 3:
                removeProduct();
                break;
            case 4:
            case 'q':
            case 'Q':
                HomeHandler homeHandler = new HomeHandler();
                homeHandler.start();
                break;
            default:
                System.out.println("잘못 입력하셨습니다.");
            }
        }
    } catch (Exception e) {}
}
```

- **RequestSearchProduct 함수**

설명:

상품조회 요청을 받아 상품이름을 통해 검색한 상품상세정보를 출력하는 함수이다.

```
private void requestSearchProduct() {
        Scanner sc = new Scanner(System.in);
        String searchName = "";
        System.out.println("찾으실 상품명을 입력해주세요");
        searchName = sc.next();
        // 검색
        ArrayList<String> a = mp.show(searchName);
        // 검색값출력
        if (a.get(0) != null) {
                a.removeAll(Collections.singleton(null));
                System.out.println("1. 코드: " + a.get(0));
                System.out.println("2. 이름: " + a.get(1));
                System.out.println("3. 사이즈: " + a.get(2));
                System.out.println("4. 가격: " + a.get(3));
        } else {
                System.out.println("해당상품이 없습니다!");
        }
}
```

2) ManageItem class

본 클래스는 상품 관리의 주요 기능 상품 조회, 상품 생성, 상품 삭제 와 관련된 요청이 올 때 관련 상품 데이터를 다루는 부분이다. 요청에 따라 데이터를 처리하기 위해 필요한 함수들을 분리하여 구현하였다.

- **ManageItem Class 의 필요 변수**

설명

다음은 ManageItem Class 에서 정의한 멤버 변수이다.

- ItemtList: 모든 상품 정보를 저장하고 있는 리스트
- listSize: 리스트에 저장된 전체 상품 개수

```
public static ArrayList<Item> ItemList = new ArrayList<Item>();
int listSize = 1;
```

- **ManageItem 생성자**

설명

상품 관리 기능이 시작하면 ManageItem 클래스는 초기 상품 값을 생성한다.

```
public ManageItem() {
        this.ItemList.add(new Item("나이키프리런", 245, 39000, 30, listSize));
}
```

- **InsertItem 함수**

설명

이름, 사이즈, 가격, 재고수량을 입력 받아 상품을 등록한다. 상품 코드는 자동 생성된다.

```
public boolean insertItem(String name, int size, int price, int quantity) {
        this.ItemList.add(new Item(name, size, quantity, price, listSize));
        listSize += 1;
        System.out.println("완료");
        return true;
}
```

- **DeleteItem 함수**

설명

이름을 입력 받아 해당하는 상품정보를 삭제한다.

```
public boolean deleteItem(String name) {
        for (int i = 0; i < listSize; i++) {
                if (ItemList.get(i).itemName.equals(name)) {
                        ItemList.remove(i);
                }
        }
```

```
            listSize -= 1;
            System.out.println("완료");
            return true;
}
```

<UseCase 3. 재고관리>

- **showQuantity 함수**

설명

이름을 입력 받아 해당하는 상품의 재고량을 출력한다.

```
public int showQuantity(String name) {

            int result = 0;
            for (int i = 0; i < listSize; i++) {
                        if (ItemList.get(i).itemName.equals(name)) {
                                    result = ItemList.get(i).itemQuantity;
                        }
            }
            return result;
}
```

- **modifyQuantity 함수**

설명

이름을 입력 받아 상품을 찾고 해당하는 상품의 재고량을 입력 받아 수정한다.

```
public boolean modifyQuantity(String name, int quantity) {
            for (int i = 0; i < listSize; i++) {
                        if (ItemList.get(i).itemName.equals(name)) {
                                    ItemList.get(i).itemQuantity = quantity;
            }
            System.out.println("완료");
            return true;
}
```

13.4. Source Code 작성 근거

Use Case Realization 을 근거로 하여 Class Diagram 및 Source Code 를 작성하였다. 특히 Use Case Realization 의 Interaction Diagram 을 기반으로 Class 및 method 들을 추출하였다.

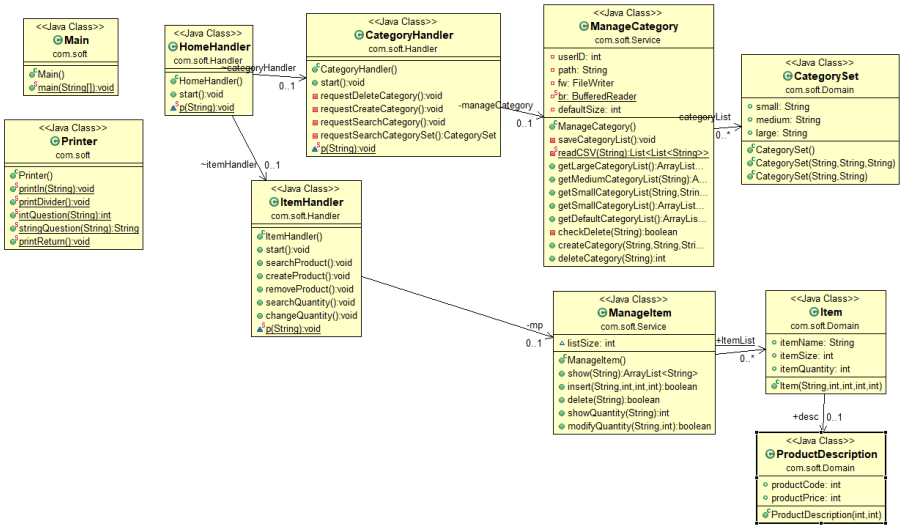


그림 35 앞서나온 Design Class Diagram 과 동일

14. Test 보고서

14.1 Control Information

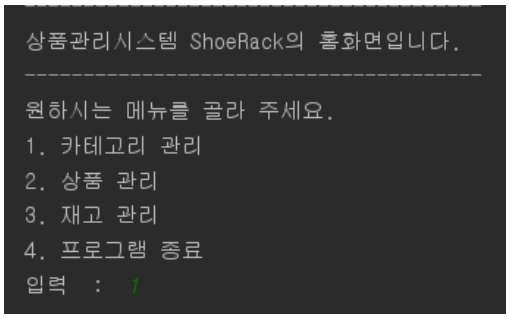
Document title	온라인 쇼핑몰 상품 관리 시스템의 Test 보고서
Document owner	소프트조
Document identification no	14
Document status	Elaboration

14.2. Revision History

버전	*일자*	*설명*	*저자*
Test Case v0	2020.07.11	카테고리 관리 및 상품 관리 테스트 케이스 작성	안창희, 이연주
Test Case v1	2020.07.12	재고관리 테스트 케이스 작성	안창희, 이연주
Test Case v2	2020.07.12	카테고리 관리, 상품 관리, 재고관리 테스트 케이스 점검	안창희, 이연주

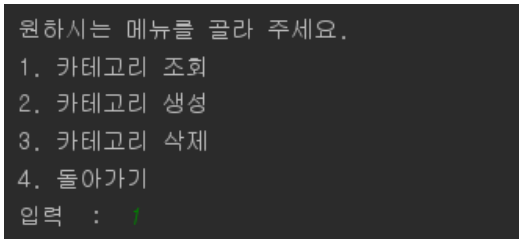
14.3. Test Case 및 Test Case 결과

시스템은 카테고리 관리를 시작하면 상품관리 홈화면을 보여주고 이용자는 원하는 작업을 선택한다.



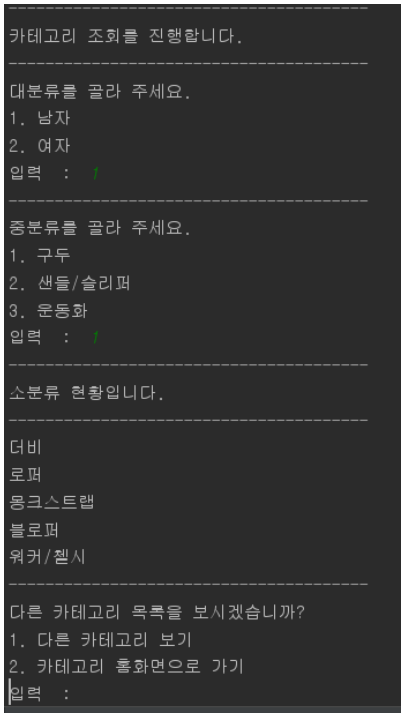
<UseCase1: 카테고리 관리>

시스템은 카테고리 관리를 시작하면 카테고리 관리 홈화면을 보여주고 이용자는 원하는 작업을 선택한다.



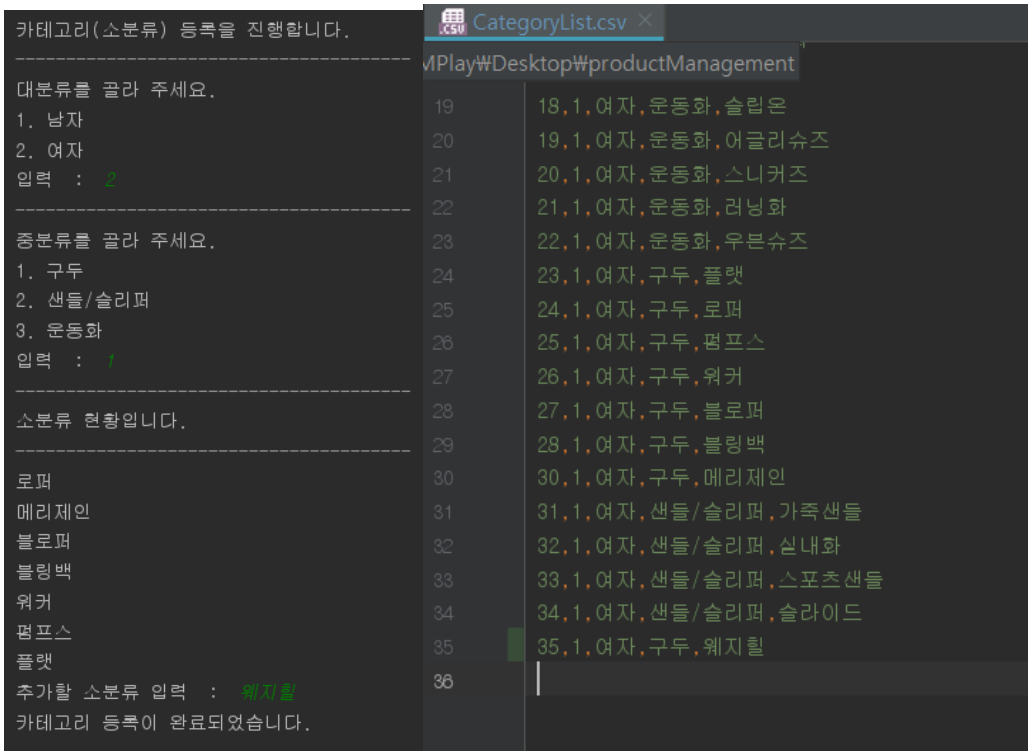
1) 카테고리 조회 Test

시스템은 이용자에게 대분류를 보여주고 대분류 입력을 받는다. 시스템은 대분류에 해당하는 중분류를 보여주고 중분류 입력 받는다. 중분류에 대한 소분류 목록을 보여준다. 시스템은 다른 카테고리를 볼 것인지 홈화면으로 나갈 것인지 묻는다. 이용자가 다른 카테고리를 보려고 할 때, 다시 대분류, 중분류를 입력하게 된다.



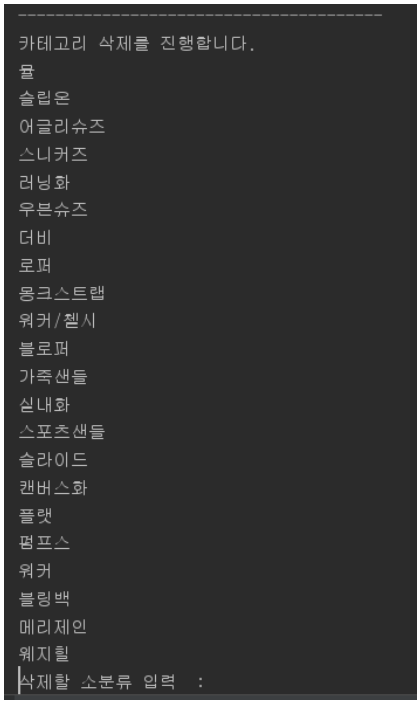
2) 카테고리 생성 Test

시스템은 이용자에게 대분류를 보여주고 대분류 입력을 받는다. 시스템은 대분류에 해당하는 중분류를 보여주고 중분류 입력 받는다. 중분류에 대한 소분류 목록을 보여준다. 이용자는 해당 대분류, 중분류 아래에 추가할 소분류를 입력한다. 시스템은 이용자가 입력한 소분류에 대한 CategorySet 을 CategoryList.CSV 파일에 추가한다.



3) 카테고리 삭제 Test

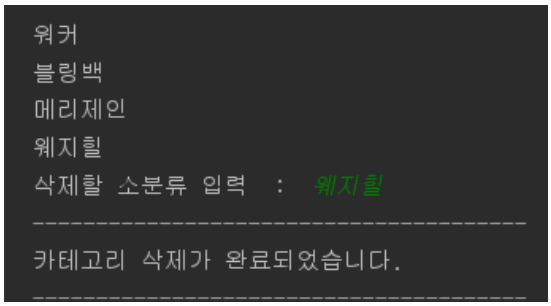
시스템은 해당 이용자의 CategorySet 에 대한 모든 소분류를 보여준다.

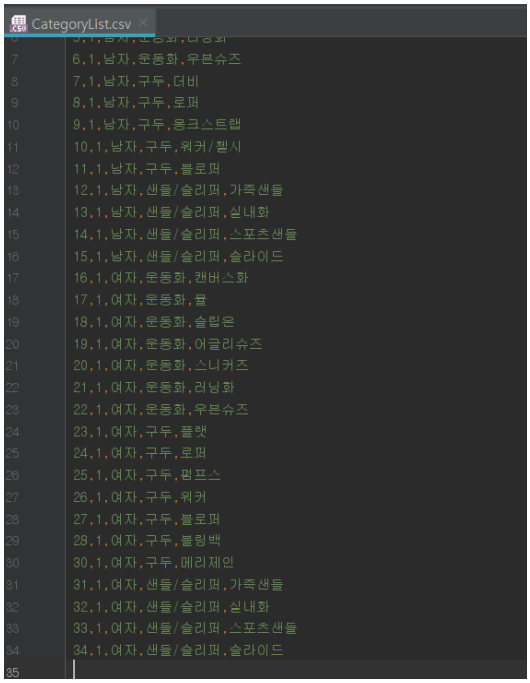


이용자는 삭제할 소분류를 입력한다.

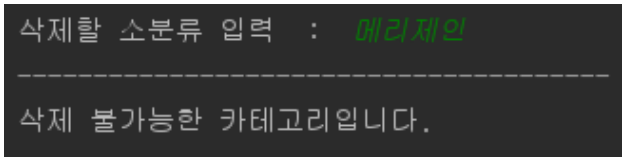
Case 1) 시스템에서 제공하는 기본 CategorySet 34 개에 대한 소분류가 아닌 경우

시스템은 입력한 소분류에 대한 Category 를 CategoryList.csv 에서 삭제하고 '카테고리 삭제가 완료되었습니다'라는 메시지를 보여준다.



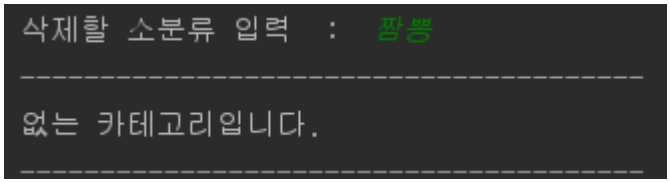


Case 2) 시스템에서 제공하는 기본 CategorySet 34 개에 대한 소분류인 경우시스템은 '삭제 불가능한 카테고리입니다' 메시지를 보여준다.

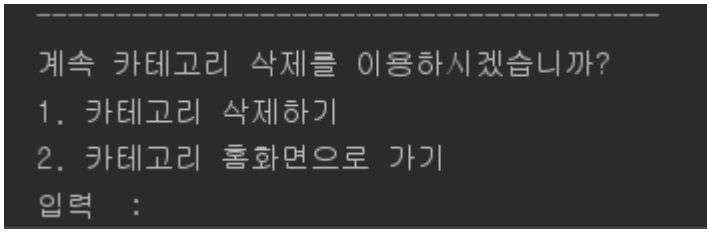


Case 3) 등록된 소분류에 대한 입력이 아닐 경우

시스템은 '없는 카테고리입니다' 메시지를 보여준다.



시스템은 이용자 입력 이후에 계속 카테고리를 삭제할 것인지 홈화면으로 갈 것인지 묻는다.
이용자는 카테고리 삭제하기를 계속 하기로 결정할 경우, 다시 지우고자하는 소분류 이름을 입력하게 된다.



<Usecase2: 상품관리>

1. 상품조회Test

시스템은 이용자에게 상품이름을 입력받고 해당하는 상품의 상세정보인 코드, 이름, 사이즈, 가격을 보여준다.

```
상품관리시스템 ShoeRack의 홈화면입니다.
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 카테고리 관리
2. 상품 관리
3. 재고 관리
4. 프로그램 종료
입력 :  2
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 상품 조회
2. 상품 등록
3. 상품 삭제
4. 돌아가기
입력 :  1
찾으실 상품명을 입력해주세요
나이키프리런
1. 코드: 1
2. 이름: 나이키프리런
3. 사이즈: 245
4. 가격: 39000
```

2. 상품생성Test

시스템은 이용자에게 상품이름을 입력받고 해당하는 상품의 상세정보인 코드, 이름, 사이즈, 가격을 보여준다.

```
----------------------------------------
상품관리시스템 ShoeRack의 홈화면입니다.
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 카테고리 관리
2. 상품 관리
3. 재고 관리
4. 프로그램 종료
입력 :  2
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 상품 조회
2. 상품 등록
3. 상품 삭제
4. 돌아가기
입력 :  2
등록할 상품이름을 입력하세요:
구두같지만구두는아닌
사이즈를 입력하세요:
260
가격을 입력하세요:
100000
재고를 입력하세요:
10
----------------------------------------
```

```
----------------------------------------
대분류를 골라 주세요.
1. 남자
2. 여자
입력 :  1
----------------------------------------
중분류를 골라 주세요.
1. 구두
2. 샌들/슬리퍼
3. 운동화
입력 :  1
----------------------------------------
소분류를 골라 주세요.
1. 더비
2. 로퍼
3. 몽크스트랩
4. 블로퍼
5. 워커/첼시
입력 :  1
완료
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 상품 조회
2. 상품 등록
3. 상품 삭제
4. 돌아가기
입력 :  1
찾으실 상품명을 입력해주세요
구두같지만구두는아닌
1. 코드: L0a8Md6cSd6c
2. 이름: 구두같지만구두는아닌
3. 사이즈: 260
4. 가격: 100000
----------------------------------------
```

3. 상품삭제Test

시스템은 이용자에게 상품이름을 입력받고 해당하는 상품의 상세정보인 코드, 이름, 사이즈, 가격을 보여준다.

```
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 상품 조회
2. 상품 등록
3. 상품 삭제
4. 돌아가기
입력 :  3
삭제할 상품이름을 입력하세요:
아디다스가젤
완료
----------------------------------------
원하시는 메뉴를 골라 주세요.
1. 상품 조회
2. 상품 등록
3. 상품 삭제
4. 돌아가기
입력 :  1
찾으실 상품명을 입력해주세요
아디다스가젤
해당상품이 없습니다!
----------------------------------------
```

<Usecase3: 재고관리>

1. 재고조회

```
조회할 상품이름을 입력하세요:
나이키프리런
나이키프리런의 현재재고는 30입니다
```

2. 재고수정

```
재고를 수정할 상품이름을 입력하세요:
나이키프리런
나이키프리런의 현재재고는 30입니다
수정할 재고량을 입력하세요:
13
완료
조회할 상품이름을 입력하세요:
나이키프리런
나이키프리런의 현재재고는 13입니다
```

14.4. Test case 추출 근거

카테고리 관리 유스케이스의 핵심 기능인 카테고리 조회, 카테고리 생성, 카테고리 삭제를 확인할 수 있는 사례를 기반으로 Test Case 를 추출 및 확인하였다. 또한 상품 관리 유스케이스의 핵심 기능인 상품 조회, 상품 생성, 상품 삭제를 확인할 수 있는 사례를 확인하기 위해 Test Case 를 추출 및 확인하였다. 카테고리 관리 유스케이스와 상품 관리 유스케이스 모두 조회, 생성, 삭제라는 공통된 기능을 제공하므로 실제 카테고리나 상품을 등록하는 방식으로 Test Case 를 추출하였다.

재고 관리 유스케이스의 경우 미리 등록이 되어있는 상품을 기반으로 재고조회 및 재고수정 기능이 수행되므로 등록된 상품을 기반으로 재고 조회 및 재고 수정 Test Case 를 추출하였다.

15. 참고문헌

[1] 국내 온라인 쇼핑 거래액 동향: 통계청, "2019 년 7 월 온라인 쇼핑 동향", 2019, 3 쪽.

[2] 국내 온라인 패션 시장 거래액 및 구성비: 통계청, "2019 년 국내 온라인쇼핑 거래액", 2019, 5 쪽.

[3] 지그재그 신규 의류 쇼핑몰 시장 진입의 어려움[웹사이트], (2020.04.20),

URL: https://www.mobiinside.co.kr/2018/03/21/creativemine-zigzag/

[4] 공정거래위원회, 전자상거래(인터넷사이버몰) 표준약관, 표준약관 제 10023 호, 2015.06.26.

[5] 국가법령정보센터, 국가종합전자조달시스템 종합쇼핑몰운영규정, 조달청고시 제 2016-9 호, 2016.02.16.

[6] 온라인 쇼핑몰 스니커굿샵 이용 약관 지적 재산권 침해 관련[웹사이트], 2020.04.20,

URL: https://ngoods.com/service/agreement.php

[7] 네이버 캐스트 용어로 보는 IT 깃허브란[웹사이트], (2020.05.02),

URL: https://terms.naver.com/entry.nhn?docId=3580149&cid=59088&categoryId=59096

[8] 공공 SW 사업 제안요청서 작성을 위한 요구사항 상세화 실무 가이드라인, SW 발주기술지원센터, 2018.10.15

[9] 제안요청서의 요구사항 작성 가이드, 한국정보화진흥원, 2011.09