객체지향설계 4조

Team Project

4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

음식점 POS 시스템 문제기술서

4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

목 차

제 1 장　　업데이트 사항

업데이트 일자	업데이트 내역
2016. 10. 26	1차 수정 : 내용 수정 및 업데이트 사항 추가하이퍼링크 삭제

제 2 장　제품 개발의 필요성

제 1 절　제품 개발의 개요

1. POS(Point of Sale) 시스템이란 판매 시점을 관리하는 시스템으로 매장에 단말기를 설치하여 매출 관리는 물론, 재고, 고객 정보, 할인 정보 등을 판매 시점에서 실시간으로 기록, 분석, 수집을 하여 동향 파악 및 경영 분석의 자료를 제공해주는 시스템이다.
2. POS 시스템은 금전 등록기와 같은 레지스터 기능, 데이터를 일시 기록 하는 파일 기능, 고객 관리 기능, 전송하는 온라인 기능 등이 탑재되어 있으며, 신용카드 단말기와 통합되어 사용한다.
3. 인터넷을 기반으로 하는 시스템으로 사업장의 운영 상황을 인터넷이 가능한 곳이면 언제 어디서나 실시간으로 관리 할 수 있으며, 인터넷을 사용하여 시스템의 유지 관리 및 실시간 고객 현황, 매출 현황 데이터를 확인할 수 있다.

※ 본 문제 기술서는 Easy-pos 업체의 POS 시스템을 기준으로 만들어졌습니다.

제 2 절　제품 개발의 중요성

1. 기존의 제품에서는 사용법이 복잡하고 불편하였으며, 고가의 가격 때문에 POS 시스템을 교체하기가 쉽지 않아 경제적 손실과 낭비적 요인이 되어 왔다.
2. 그러나, 본 제품에서는 POS 시스템의 UI를 보다 쉽고 편리하게 사용자의 접근성을 높였으며, 제품의 가격을 대폭 낮추었다.
3. 따라서 현 시장에 판매되고 있는 고가의 POS 시스템을 대체하며 사용자의 작업 능률의 향상과 비용 부담 절감 효과가 기대된다.
4. 특히, 외국 제품의 경우에는 제품 가격이 매우 높아 대형 유통업체 중심으로 공급되고 있으며, 영세한 자영업 음식점에는 가격적 부담으로 인하여 공급이 부진하다.
5. 따라서 1 인 자영업 음식점을 대상으로, 이에 적합하고 사용이 편리한 저가 POS 시스템 개발이 절실한 실정이다.

제 3 절　관련 기술의 현황 및 전망

1. 국내•외 기술 개발 현황

- Tablet 기반 Mobile POS : 올해 신세계 백화점은 윈도우 기반 태블릿 Mobile POS를 총 8000 여대 도입, 고객을 응대할 때 고객 쇼핑 데이터와 연동하여 고객에게 먼저 할인 쿠폰이나 프로모션 혜택을 제공하고 모델 착용 사진을 보여주는 등 적극 활용 중이다.

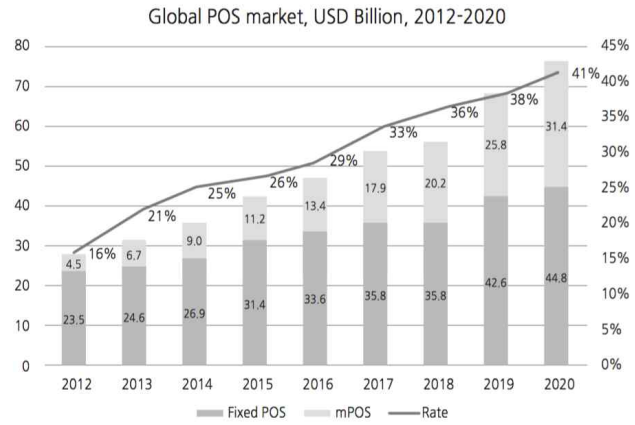
- **Big Data 활용 : 작년부터 대한상공회의소가 직접 빅데이터 분석 플랫폼을 운영하기 시작, 상품군별 인기 상품, 가격 동향, 지역 특성별 상품 등의 빅데이터 분석 결과를 중소 매장 POS 단말에 제공하고 있다.**
- **Web 기반 POS : POSBANK, EASYPOS 등 국내 업체들은 현재 POS정보를 Web에서 실시간 조회할 수 있도록 Web ASP 서비스를 개발, 운영 중이다.**

2. **문제점**

문제점	세부 내용
재고 관리를 할 수 없다.	메뉴에 들어가는 재료들의 재고를 관리하기 어렵다.
음식의 조리 가능한 양을 알 수 없다	재고를 토대로 음식의 조리 가능한 양을 알기 어렵다.

3. **앞으로의 전망**

<표 2> 세계 POS 단말기 시장현황 및 전망



<Mobile POS 관련 시장 및 동향, TTA Journal>

- **스마트폰의 보편화로 인해 Mobile POS의 보급이 가속화될 것이며 2020 년 이후 고정형(Fixed)POS와 Mobile POS의 보급 비율은 대등해질 전망이다.**
- **최근 POS 시스템의 단말기 해킹 및 신용카드 위•변조사건이 급증하고 있어, 미국에서는 보안을 강화한 신형 IC카드 단말기를 장착한 POS가 의무화했으나 막대한 교체 비용의 발생하는 상황이다. 그러므로 다양한 결제 방식에 호환 가능하며 비용이 저렴한 Mobile POS가 앞으로 각광받을 것이다.**
- **발전하고 있는 해킹 기술들로 인해 신형 IC 카드 단말기 또한 조만간 취약점을 드러낼 가능성이 있어 현 스마트폰이 지원하는 NFC와 지문**

인식 기능을 활용한 POS 시스템이 출현할 것으로 예상된다.

제 3 장　제품 개발 내용 및 방법

제 1 절　연구개발 목표

사용자	개발 목표
개발자	- 음식점 POS 시스템을 개발하는 과정에서의 문제 기술서를 작성해봄으로써 프로젝트의 주제를 명확하게 이해할 수 있는 능력을 갖추게 된다. - 실전 어플리케이션 개발 능력을 배양한다.
관리자	- 음식점 POS 시스템을 이용함으로써 음식점의 재무 관리를 보다 편리하고 정확하게 한다. - 음식점 POS 시스템으로 인한 주문 시간 단축을 통해 고객 만족도를 늘리고 결과적으로 음식점의 전체 매출 증가를 유도할 수 있도록 한다.
판매자	- 체계적인 시스템을 통해 업무를 더욱 효율적으로 진행할 수 있다.

제 2 절　연구 개발 내용 및 범위

POS 시스템을 사용하는 사용자 혹은 관리자는 메뉴의 등록을 할 수 있으며, 메뉴의 판매량을 파악하여 메뉴에 필요한 재료의 재고량 등을 파악하고 인기/비인기 메뉴를 확인하여 재료의 발주, 보관 업무를 처리한다. 매출의 동향을 POS시스템에서 확인할 수 있다.

음식 결제 시 결제 수단은 현금과 신용카드 결제로 구분한다. 손님이 결제할 총 음식값과 받은 값의 차액을 표시하고 관리자는 차액을 확인하여 고객에게 거스름돈을 단말기에 있는 금전등록기로부터 지불한다. 신용카드 결제는 결제대행업체(VAN사)와 온라인상으로 연결되어 있어야 하며, 카드 리더기에 입력한 신용카드의 정보를 VAN사로 넘기고, VAN사는 이 것을 카드사에 결제 요청을 하여 응답을 받은 후 다시 POS시스템으로 전송하여 카드 결제를 할 수 있도록 한다.

1. **주문 관련 기능**
   1) **사용자가 쉽고 간편하게 고객의 주문 내용을 POS 시스템에 입력할 수 있다.**
2. **계산 관련 기능**
   1) **받은 금액과 메뉴 가격으로의 차액을 계산하여 단말기 상에 거스름돈을 표시할 수 있다.**
3. **재고 관리 관련 기능**
   1) **어느 한 메뉴에 들어가는 일정량의 재료 재고를 POS 시스템에 등록하여 사용한다.**

2) 등록된 재료 재고를 기반, 해당 요리의 조리 가능한 양(인분)을 알려준다.

4. 매출 확인 관련 기능

1) 당일의 매출 내역을 보여준다.

2) 매출 내역에는 판매금액, 판매일시, 테이블번호가 포함되며 각 항목별로 정렬이 가능하다.

제 4 장　기대 성과 및 활용 방안

제 1 절　기술적 성과

본 시스템을 사용하여 기대할 수 있는 효과는 다음과 같다.

1. 바코드의 유용성 : 부착된 바코드를 사용함으로서 수작업으로 인한 오류를 방지할 수 있다.
2. 신용카드의 승인 : 신용카드의 승인과 현금 영수증의 발급이 한 기계안에서 해결이 가능하다. 또한 돈을 다른 곳에 관리하지 않아도 한 곳에서 관리하는 것이 가능하다.
3. 데이터의 전산화 : 상품과 고객 정보등 판매에 필요한 대부분의 정보들을 전산화 하는 것이 간단하다. 데이터 관리가 용이하다.
4. 시스템 메뉴의 다양화 : 프로그램 안에서의 수정으로 POS시스템을 사용하는 사용자의 편의에 따라 기능이 변경되는 것이 가능하다.
5. 데이터의 영구성 : 데이터가 전산화 되어있기 때문에 원한다면 오랫동안 저장하는 것이 가능하다.

제 2 절　경제적 성과

본 시스템을 사용하여 기대할 수 있는 효과는 다음과 같다.

1. 업무 효율의 향상 : 정확한 판매 계산과 정산 시간 단축으로 효율적인 업무가 가능하며, 실시간 재고 파악 및 발주 체계의 간소화 가능
2. 판매 실적 관리 및 통계 확인 : 메뉴 판매 실적에 따른 매장 구성으로 매출 이익 증가와 상품 관리가 가능함에 따라 소비 성향, 취급 메뉴, 재고 등을 용이하게 파악할 수 있다.
3. 고객 관리의 능력 향상 : 현금 혹은 신용카드 등을 통해 고객의 성향 파악을 할 수 있다.
4. 가격의 정확성 : 등록 가격을 정확히 기입하여 상품의 판매 정확도를 높일 수 있음
5. 상품의 순환 : 비인기 상품과 인기 상품의 구분을 정확히 하여 상품을 순환 시킬 수 있다.

제 3 절　활용 방안

본 시스템의 주된 활용법으로는 다음과 같다.

1. 재고 파악 활용 : 점포를 운영할 때, 메뉴에 쓰이는 재료를 실시간으로 확인 하여 재고를 파악할 수 있다.
2. 전산화로 인한 영업의 편리성 : 주문 관리의 전산화로 인하여 영업 및 고객 관리를 체계화할 수 있다.

제 5 장 기타

제 1 절 인원 편성표

제 2 절 연구 추진 일정

순번	주차													세부 추진 일정
	1	2	3	4	5	6	7	8	9	10	11	12	13	
1														문제 기술서 작성
2														요구사항 명세서 작성
3														Use-case diagram 작성
4														Class diagram 작성
5														Sequence diagram 작성 (+ Activity diagram)
6														화면설계서 작성
7														소스코드 구현
8														최종 보고서 및 결과
9														유지 보수

제 3 절 참고문헌

- 김재범(2016), 「Mobile POS 관련 시장 및 동향」,
『TTAJournal』,163,95-101
- 전형구, 지정근(1996), 「국내 유통업체의 POS시스템 활용사례」, 『물류학회지』, 6, 205-220
- 황유동(2000), 「안전한 POS SYSTEM의 설계에 관한 연구」, 순천향대학교 일반대학원 석사학위논문
- 이진철, 「신세계, 인텔 기반 윈도우 태블릿 8 천대 도입.. 스마트 쇼핑 환경 구축」, 『이데일리』, 2016.3.16.,
(http://m.news.naver.com/read.nhn?mode=LSD&mid=sec&sid1=101&oid=018&aid=0003500849), 2016.09.20

음식점 POS 시스템 요구사항 명세서

4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

목 차

제 1 장 요구사항 명세서 업데이트 내역

업데이트 일자	업데이트 내용
2016. 10. 3.	최초 생성
2016. 10. 10.	1차 수정
2016. 10. 12	2차 수정
2016. 10. 17	3차 수정
2016. 11. 21	4차 수정 : 기능 수정 및 레시피 관리 기능 추가 비기능 부분 수정

제 2 장 요구사항 분석

제 1 절 요구사항 분석표

4 조 음식점 POS 시스템		
수준 1	수준 2	수준 3
POS-System	테이블 주문 관리	음식메뉴 관리
		테이블 주문내역 확인
	테이블 결제	카드 결제
		현금 결제
	재고 관리	
	매출 조회	
	레시피 정보 관리	
	비기능	사용환경
		시스템 오류 처리
		다중 사용자 관리

제 3 장 기능 요구사항

제 1 절 주문

요구사항 번호	테이블 주문관리 1		
요구사항 명칭	음식메뉴 관리	요구사항 분류	테이블 주문 관리
요구사항 설명	정의	메뉴를 관리한다.	
	세부 내용	1. 테이블별로 음식점의 음식을 선택하여 주문할 수 있다. 2. 메뉴와 함께 가격을 표시할 수 있다. 3. 고객이 주문한 메뉴를 삭제할 수 있다. 4. 고객이 같은 메뉴를 중복 주문했을 시 메뉴의 수량을 설정할 수 있다. 5. 주문 후 취소했을 시 음식을 삭제할 수 있다.	

요구사항 번호	테이블 주문관리 2		
요구사항 명칭	테이블 관리	요구사항 분류	테이블 주문 내역 확인
요구사항 설명	정의	고객이 앉은 테이블(자리)를 관리할 수 있다.	
	세부 내용	1. 고객이 앉은 테이블을 시스템 상에서 선택할 수 있다. 2. 테이블 별 메뉴를 확인할 수 있다. 3. 테이블 별 총 주문가격을 확인할 수 있다.	

제 2 절 결제

요구사항 번호	결제 1		
요구사항 명칭	카드 결제	요구사항 분류	결제
요구사항 설명	정의	카드로 결제를 할 수 있다	
	세부 내용	1. 결제 시 체크카드, 신용카드를 이용하여 결제를 할 수 있다. 2. 결제 후 영수증을 출력할 수 있다.	

요구사항 번호	결제 2		
요구사항 명칭	현금 결제	요구사항 분류	결제
요구사항 설명	정의	현금으로 결제할 수 있다.	
	세부 내용	1. 결제 시 현금을 이용하여 결제를 할 수 있다. 2. 고객으로부터 받은 돈을 입력하면 거스름돈을 표시할 수 있다.	

제 3 절 재고

<table>
<tr><td>요구사항 번호</td><td colspan="3">재고 관리 1</td></tr>
<tr><td>요구사항 명칭</td><td>재고 관리</td><td>요구사항 분류</td><td>재고 관리</td></tr>
<tr><td rowspan="2">요구사항 설명</td><td>정의</td><td colspan="2">재고를 등록 및 관리할 수 있다.</td></tr>
<tr><td>세부 내용</td><td colspan="2">1. 재료를 등록할 수 있다.
2. 재료의 수량을 수정할 수 있다.
3. 재료를 삭제할 수 있다.
4. 재료의 재고를 조회할 수 있다.</td></tr>
</table>

제 4 절 레시피

요구사항 번호	레시피 1		
요구사항 명칭	레시피 정보 관리	요구사항 분류	레시피 정보 관리
요구사항 설명	정의	점포에서 판매하는 음식에 관한 레시피를 등록, 수정, 삭제할 수 있다.	
	세부 내용	1. 레시피를 등록할 수 있다. 2. 레시피를 수정할 수 있다. 3. 레시피를 삭제할 수 있다.	

제 5 절 매출

요구사항 번호	매출 확인 1		
요구사항 명칭	매출 조회	요구사항 분류	매출 확인
요구사항 설명	정의	매출 내역을 확인한다.	
	세부 내용	1. 매출 내역을 확인할 수 있다. 2. 월별 매출 내역을 확인할 수 있다. 3. 매출 내역에 대해 영수증을 조회할 수 있다.	

제 4 장 비 기능 요구사항

제 1 절 비 기능

요구사항 번호	비기능 1		
요구사항 명칭	실행환경	요구사항 분류	비기능
요구사항 설명	정의	시스템을 사용하기 위해 환경을 조성한다.	
	세부 내용	1. 시스템 부팅 시 첫 페이지에서 테이블 현황에 대한 정보를 볼 수 있다.	

요구사항 번호	비기능 2		
요구사항 명칭	시스템오류처리	요구사항 분류	비기능
요구사항 설명	정의	작동 중 생길 오류를 처리한다.	
	세부 내용	1. 시스템 오류가 생길 시 오류 처리에 대한 알람을 내보낸다. 2. 시스템 오류가 생길 시 현재 동작에 대한 정보를 저장 후 시스템을 종료 시킨다.	

요구사항 번호	비기능 3		
요구사항 명칭	보안설정	요구사항 분류	비기능
요구사항 설명	정의	비밀번호를 설정하여 권한을 가진 관리자(점장)만 접근할 수 있도록 한다.	
	세부 내용	1. 비밀번호를 등록, 선택하여 사용자별 관리가 가능하다.	

음식점 POS 시스템 Use-case

4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

목 차

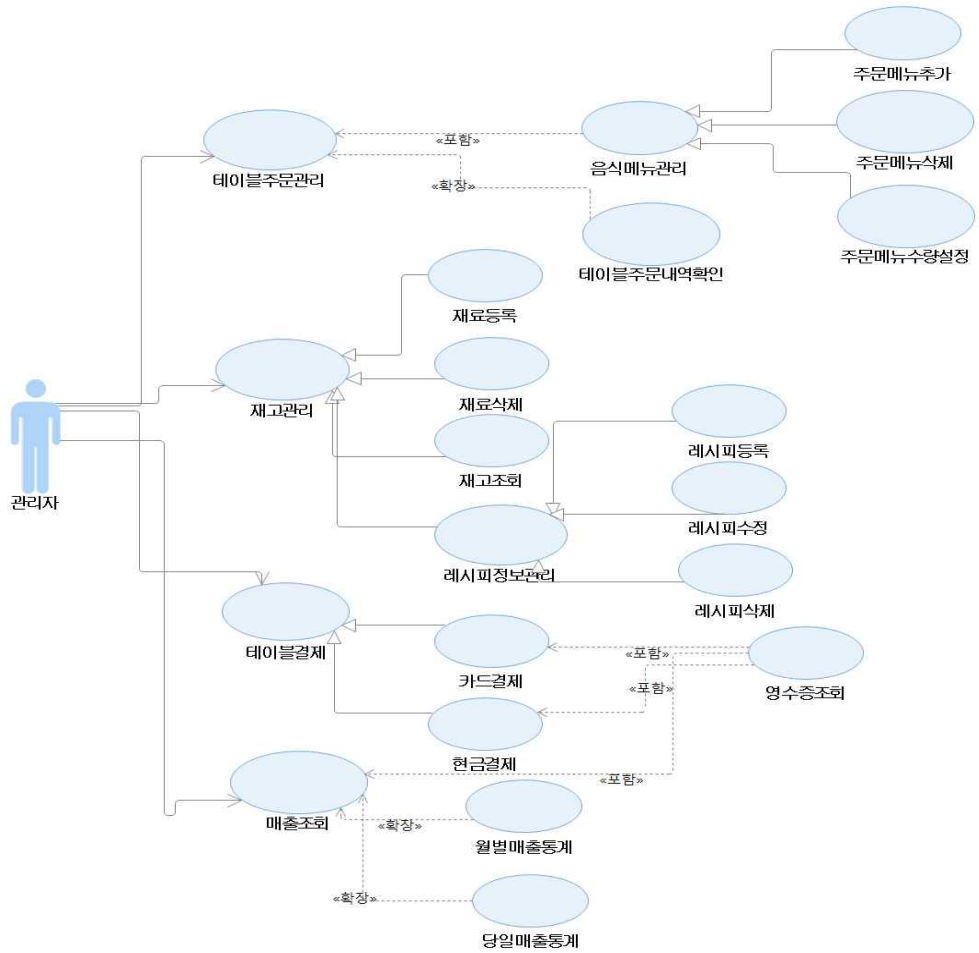
제 1 장 유스케이스 업데이트 사항

날 짜	업데이트 내용
2016. 11. 7	1차 수정 : 내용 추가(정보 속성 수정)
2016 11. 11	2차 수정 : 테이블 주문 관리 유스케이스 시나리오 (내용 수정)
2016. 11. 14	3차 수정 : 음식 메뉴 관리 유스케이스 시나리오 매출 조회 유스케이스 시나리오 (내용수정)
2016. 11. 19	4차 수정 : 유스케이스 다이어그램 교체 단어 수정
2016. 11, 21	5차 수정 : 유스케이스 다이어그램 수정 모든 유스케이스 시나리오 정보수정
2016 12. 16	6차 수정 : 모든 시나리오 Pre-condition 수정 재고관리 기본흐름 수정 매출조회, 영수증조회 예외흐름 삭제 영수증조회 대안흐름 삭제

제 2 장 유스케이스 다이어그램

제 1 절 유스케이스 다이어 그램 및 관계 설명

유스케이스 다이어그램



설 명

위 유스케이스 다이어그램은 음식점 POS시스템을 사용자관점에서 묘사한 다이어그램이다.

관리자는 테이블주문관리, 재고관리, 테이블결제, 매출조회, 재고관리, 레시피정보관리, 매출조회 기능을 사용할 수 있다.

테이블주문관리 기능은 음식메뉴관리 기능과 테이블주문내역확인 기능을 가지고 있다. 음식메뉴관리 기능은 주문메뉴 추가, 삭제, 수량설정을 할 수 있으며 테이블주문내역확인은 각 테이블의 주문한 음식내역을 확인할 수 있다.

재고관리 기능은 음식점의 재료를 등록, 수정, 삭제 및 재고 조회를 할 수 있는 기능을 가지고 있다.

레시피 정보관리기능은 레시피를 등록, 수정, 삭제할 수 있는 기능을 가지고 있다.

테이블 결제 기능은 각 테이블의 결제방법을 뜻하며 결제 방법에는 현금 결제와 신용카드 결제가 있다.

매출조회 기능은 해당 음식점의 매출을 조회해볼 수 있으며 추가적으로 월별 매출의 통계를 확인할 수 있다. 매출조회 기능은 영수증 조회를 할 수 있다.

제 2 장 유스케이스 시나리오

제 1 절 주문 메뉴 추가 기능 시나리오

1) 음식 메뉴 관리 시나리오

이 름	음식메뉴 관리 시나리오		
개 요	관리자는 고객이 주문한 메뉴를 추가하거나 삭제할 수 있으며 음식의 수량을 설정할 수 있다.		
Initiator	관리자	Supporters	
Pre-condition	메뉴에 대한 정보가 이미 입력되어있어야 한다.		
Post-condition			

기본 흐름

1) 테이블주문관리에서 주문할 테이블 선택한다.
2) 음식메뉴관리가 사용자에게 보여진다.
3) **음식명**과 현재 조리가능한 양(**인분**)이 함께 표시된다.
4) 음식메뉴관리에서 추가하고 싶은 **음식명**을 선택한다.
5) 기본 수량이 한 개로 설정되며 **주문번호**가 부여된다.
6) 테이블의 주문내역에 음식이 추가된 정보를 보여준다.

대안 흐름

1. 음식을 삭제하고자 하는 경우
   1) **테이블**을 선택한다.
   2) 테이블의 주문내역과 음식메뉴관리가 사용자에게 보여진다.
   3) 주문내역에서 삭제하고자 하는 음식을 선택한다.
   4) 주문한 메뉴 삭제 기능을 실행한다.
   5) 음식이 삭제된 테이블의 주문내역을 보여준다.

2. 음식의 수량을 새로 설정하고자 하는 경우
   1) **테이블**을 선택한다.
   2) 테이블의 주문내역과 음식메뉴관리가 사용자에게 보여진다.
   3) 주문내역에서 **수량**을 변경하고 싶은 음식을 선택한다.
   4) 고객이 원하는 수량만큼 수량증가 혹은 수량삭제 기능을 실행한다.
   5) 주문한 메뉴 수량 설정 기능을 실행한다.
   6) 수량이 변경된 테이블의 주문내역을 보여준다.

예외 흐름

1. 메뉴 주문 수량이 현재 조리 가능한 재료의 재고량을 초과하는 경우
   1) 재료의 재고량이 부족하여 주문할 수 없다는 메세지를 알린다.

2) 테이블 주문 내역 확인 시나리오

이 름	테이블 주문내역 확인 시나리오		
개 요	관리자는 테이블 별 정보를 볼 수 있다.		
Initiator	관리자	Supporters	
Pre-condition	테이블의 자리 및 정보가 고정되어 있어야 한다.		
Post-condition			
기본 흐름			
1) 원하는 테이블을 선택한다. 2) 테이블의 **테이블 번호**, **음식명**, **가격**, **총 주문금액**을 보여준다.			
대안 흐름			
예외 흐름			
1. 테이블이 현재 사용되고 있지 않은 경우 1) 원하는 테이블을 선택한다. 2) 타 정보들은 공백으로 표시된다.			

제 2 절 재고 관리 시나리오

1) 재고 관리 시나리오

이 름	재고 관리 시나리오		
개 요	POS시스템 상에서 음식점에서 판매하고 있는 음식들의 재료와 재료의 재고를 입력할 수 있으며, 필요 시 삭제를 할 수 있다. 재료의 재고를 조회할 수 있다.		
Initiator	관리자	Supporters	
Pre-condition			
Post-condition	입력 및 수정한 재료 및 재료의 재고에 관한 정보가 저장된다.		

기본 흐름

1) 재고 관리 기능을 선택한다.
2) 재료들의 정보(**재료명**, **재료량**)를 확인할 수 있다.
3) 수정하고자 하는 재료를 선택한다.
4) 재료의 정보(**재료명**, **재료량**)를 입력하고 재료 수정 기능을 실행한다.
5) 재고 관리 화면에 변경된 재료가 표시된다.

대안 흐름

1. 재료를 등록하고자 하는 경우
    1) 재고 관리 기능을 선택한다.
    2) 재료들의 정보(**재료명**, **재료량**)를 확인할 수 있다.
    3) 등록 하고자 하는 재료의 정보 해당(**재료명**, **재료량**)를 입력하고 재료 등록 기능을 실행한다.
    4) 재고 관리 화면에서 추가된 재료가 표시된다.

2. 재료의 정보를 삭제하고자 하는 경우
    1) 재고 관리 기능을 선택한다.
    2) 재료들의 정보(**재료명**, **재료량**)를 확인할 수 있다.
    3) 삭제하고자 하는 재료를 선택하고 재료 삭제 기능을 실행한다.

예외 흐름

1. 등록하고자 하는 재료가 이미 등록되어 있는 경우
    1) 동일 재료가 존재한다는 메세지를 보여준다.

2. 등록하고자 하는 재료의 재료명을 입력 안 했을 경우
    1) 재료의 재료명을 입력 안 했다는 메세지를 알리고 재료명 입력을 요청한다.
3. 등록하고자 하는 재료의 갯수를 입력 안 했을 경우
    1)재료의 갯수를 입력 안 했다는 메세지를 알리고 재료갯수 입력을 요청한다.

2) 레시피 정보 관리 시나리오

이 름	레시피 정보 관리 시나리오		
개 요	POS시스템 상에서 음식점에서 판매하고 있는 메뉴들의 레시피를 추가 및 수정할 수 있다. 입력된 레시피 정보는 현재 재료의 재고 기준으로 조리 가능한 양으로 환산되어 음식메뉴관리 화면에 표시한다.		
Initiator	관리자	Supporters	
Pre-condition			
Post-condition	레시피의 정보가 저장된다.		

기본 흐름

1) 레시피 정보 관리 기능을 선택한다.
2) 레시피를 등록하고자 하는 음식메뉴를 선택한다.
3) 해당 음식의 레시피의 정보(**재료명**, **필요양**)를 입력하고 레시피 등록 기능을 실행한다.

대안 흐름

1. 레시피 정보를 변경하고자 할 때
    1) 레시피 정보 관리 기능을 선택한다.
    2) 레시피의 정보를 변경하고자 하는 메뉴를 선택한다.
    3) 해당 음식의 변경이 필요한 정보(**재료명**)를 선택하고 수정한다.

2. 레시피 정보를 삭제하고자 할 때
    1) 레시피 정보 관리 기능을 선택한다.
    2) 레시피의 정보를 삭제하고자 하는 음식메뉴를 선택한다.
    3) 해당 음식의 삭제가 필요한 정보(**재료명**)를 선택하고 삭제한다.

예외 흐름

1. 음식의 등록된 레시피가 없음에도 불구하고 변경 혹은 삭제 기능을 실행하는 경우
    1) 레시피가 등록되어 있지 않다는 메시지를 알리고 재료를 추가해달라는 요청을 한다.
    2) 레시피 등록 기능을 실행한다.

제 3 절 테이블 결제 시나리오

이 름	테이블 결제 시나리오		
개 요	음식메뉴 주문 정보가 해당 테이블에 저장된 상태에서 점장 혹은 아르바이트생이 결제방식(신용카드결제, 현금결제)을 선택해 결제를 수행한다.		
Initiator	관리자	Supporters	고객
Pre-condition	주문 정보가 해당 고객의 테이블에 저장된 상태여야 한다. 그 후, 결제화면으로 이동한 상태이다.		
Post-condition	주문에 대한 결제정보가 생성되며 신용카드 결제를 선택한 경우 결제 승인이 이루어지면 결제상태가 '결제승인'으로 저장된다. 현금결제를 선택한 경우는 '결제완료'로 저장된다.		

기본 흐름

1) 테이블주문관리 기능에서 결제를 수행할 테이블을 선택한다.
2) 해당 테이블 음식메뉴관리 기능에 결제방식('신용카드결제'와 현금결제')을 선택할 수 있는 기능이 존재한다.
3) 또한, **총주문액**, **받은 금액**, **거스름돈**을 보여준다.
4) 결제방식으로 '신용카드결제'를 선택한다.
5) 신용카드 정보(**신용카드사**, **신용카드번호**, **유효기간**, **할부개월수**) 입력 기능을 실행한다.
6) 신용카드 정보를 입력하고 '결제' 기능을 실행한다.
7) 고객이 서명을 입력한다.
8) 신용카드 정보를 신용카드 인증회사에 보내 결제 승인을 요청한다.
9) 신용카드 인증회사는 신용카드 정보를 확인하고 주문총액만큼 결제 승인 처리를 수행한다.
10) 성공적으로 승인된 경우, 결제정보(**결제방식**, **결제상태**, **승인번호**, **신용카드사**, **신용카드번호**, **고객서명**, **테이블번호**, **영수증번호**, **결제일자**)를 저장한다. 결제상태는 '결제승인'으로 저장된다.
11) 결제가 완료되었음을 알려주는 메세지를 표시한다.

대안 흐름

1. 결제방식으로 현금결제를 선택한 경우
    1)테이블주문관리 화면에서 결제를 수행할 테이블을 선택한다.
    2) 해당 테이블 음식메뉴관리 화면에 결제방식('신용카드결제'와 현금결제')을 선택할 수 있는 기능이 존재한다.
    3) 또한 **총주문액**, **받은금액**, **거스름돈**을 보여준다
    4) 받은 금액에 고객으로부터 받은 현금을 입력한다.
    5) 결제방식으로 '현금결제'를 선택한다.
    6) **거스름돈**을 반영해서 보여준다
    7) 결제정보(**영수증번호**, **결제방식**, **결제상태**, **테이블번호**, **결제금액**, **결제일자**)를 저장한다. 결제상태는 '결제완료'로 저장된다.
    8) 영수증 출력 여부를 물어보는 메시지를 출력한다.

예외 흐름

1. 신용카드 번호 오류
    1) 입력한 카드번호에 오류가 있음을 알리고, 신용카드정보 재입력을 요청한다.
2. 신용카드 유효기간 오류
    1) 입력한 유효기간에 오류가 있음을 알리고, 신용카드정보 재입력을 요청한다.

3. 신용카드 이용한도 오류
    1) 신용카드의 이용한도가 초과되었음을 알리고, 신용카드정보 재입력을 요청한다.
4. 현금 부족 오류
    1) 받은 현금이 주문액에 비해 부족함을 알리고, 받은 금액 재입력을 요청한다.
5. 0 원 오류
    1) 해당 테이블에 주문한 메뉴가 없음을 알린다.

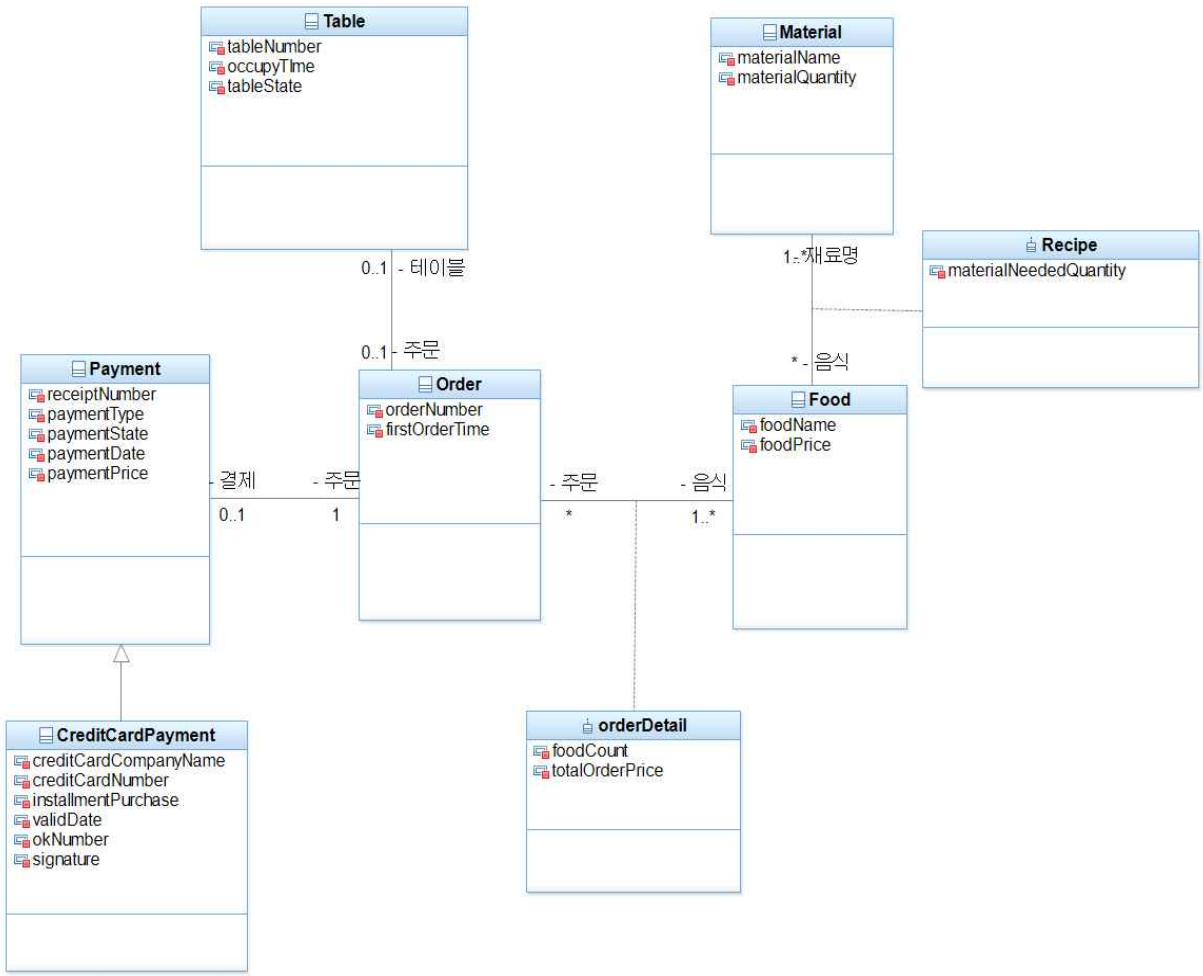
제 4 절 매출 조회 시나리오

이 름	매출 조회 시나리오		
개 요	매출 조회는 일정 기간의 매출 내역을 보여주며 당일 혹은 월별 매출통계를 보여줄 수 있다.		
Initiator	관리자	Supporters	
Pre-condition	조회해야 할 매출 정보가 저장된 상태여야 한다.		
Post-condition			
기본 흐름			
1) 관리자가 메인 화면에서 매출 통계를 선택한다. 2) 당일의 **결제방식**, **총매출액**을 보여준다. 2) 관리자가 조회하고자 하는 **조회일자**의 범위를 입력한다. 3) 조회 기능을 실행한다. 4) **결제일자**, **결제방식**, **영수증번호**, **결제금액**이 표시된 매출 목록을 최근 순으로 보여준다.			
대안 흐름			
1. 월별 매출조회를 선택한 경우 1) 관리자가 매출조회에서 월별 매출확인을 선택한다. 2) 당일의 **월**, **총매출액**, **카드결제건수**, **현금결제건수**을 보여준다. 3) 관리자가 조회하고 싶은 **연도**를 입력한다. 4) 조회 기능을 실행한다. 5) 해당 연도의 1 월부터 12 월까지 **월**, **총매출액**, **카드결제건수**, **현금결제건수**을 보여준다.			
예외 흐름			

제 5 절 영수증 조회 시나리오

이 름	영수증 조회 시나리오		
개 요	결제 정보가 저장된 상태에서 점장 혹은 아르바이트생이 결제일시를 선택해 영수증을 조회한다.		
Initiator	관리자	Supporters	
Pre-condition	조회해야 할 결제 정보가 저장된 상태여야 한다.		
Post-condition			
기본 흐름			
1) 매출 조회에서 영수증 조회를 선택한다. 2) 당일 결제된 영수증 목록을 기본적으로 보여준다. 3) 조회하고자 하는 **결제일자**의 범위를 입력한다. 4) 해당 결제일자의 영수증들의 목록을 보여준다. 5) 목록 중 한 영수증을 선택한다. 6) 선택한 영수증의 상세내역(**영수증 번호**, **결제일자**, **결제방식**, **결제상태**, **결제금액**)이 표시된다.			
대안 흐름			
예외 흐름			

클래스 다이어그램 (동적 미포함)



음식점 POS 시스템
시퀀스 다이어그램

4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

목 차

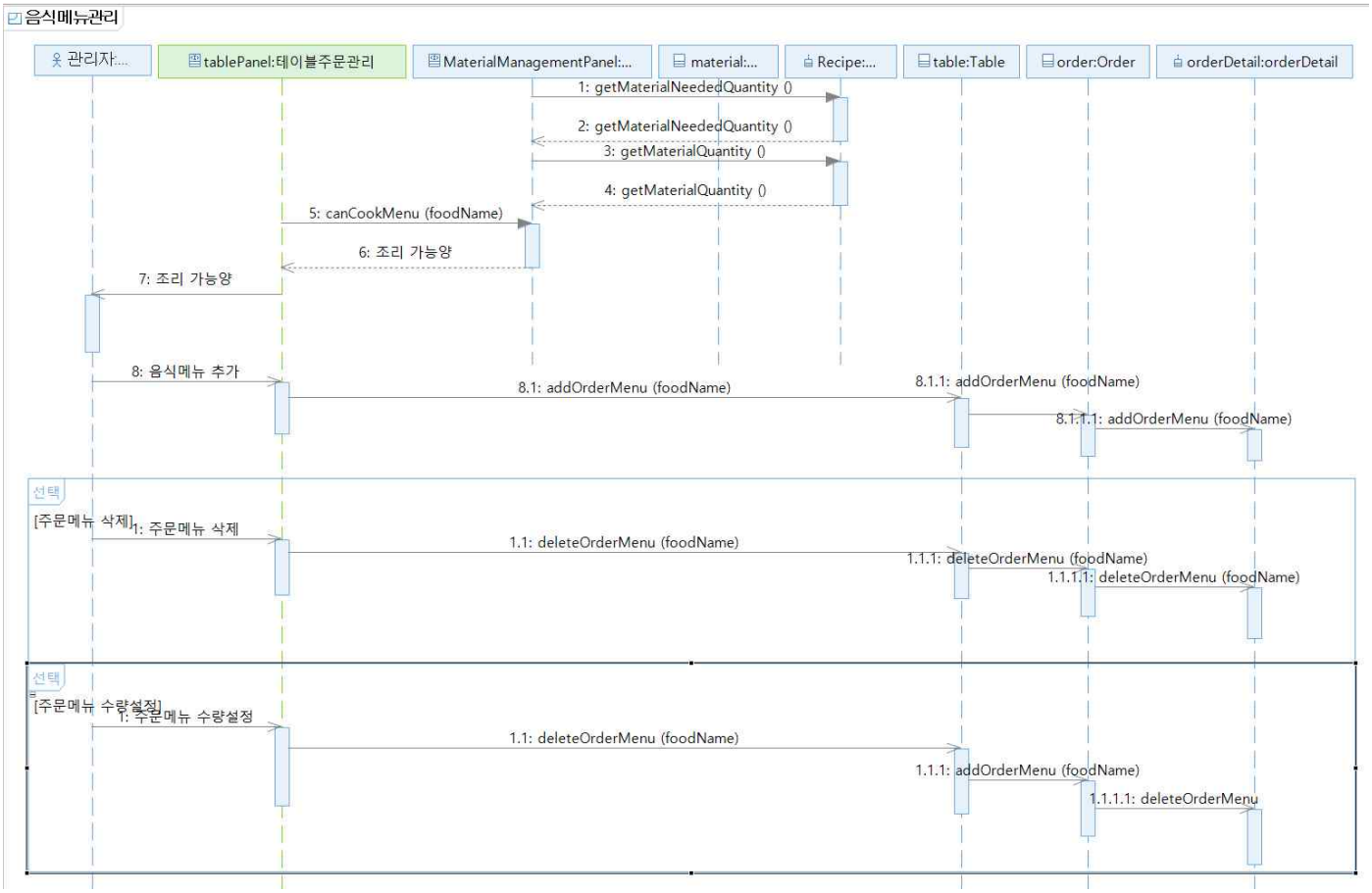
제 1 장 시퀀스 다이어그램 업데이트 사항

날 짜	업데이트 내용
2016. 11. 21	1차 수정 : 시퀀스 다이어그램 교체
2016 12. 16	2차 수정 : 시퀀스 다이어그램 교체

제 2 장 시퀀스 다이어그램

제 1 절 시퀀스 다이어그램 및 관계 설명

음식메뉴 관리 기능



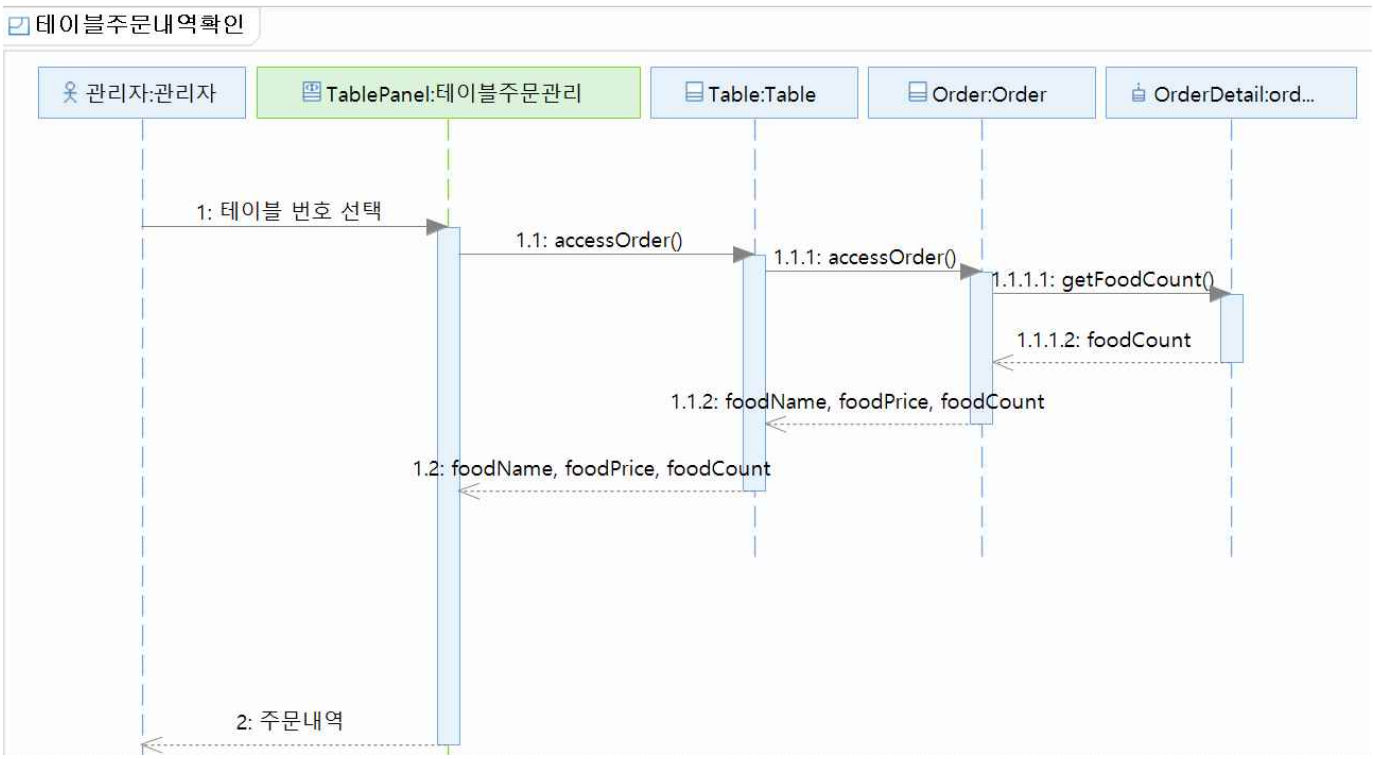
설 명

음식메뉴 관리는 기본적으로 현재 재료의 재고와 레시피를 반영해서 음식메뉴 별 조리가능양을 항상 보여준다. 관리자는 음식을 추가할 수 있다. 관리자는 tablePanel에서 음식메뉴추가 버튼을 클릭한다. 추가요청 받은 음식메뉴는 table, order를 거쳐 orderDetail클래스에 오퍼레이션을 요청하면 음식이 추가 되는데 이러한 오퍼레이션을 'addOrderMenu' 오퍼레이션으로 식별한다.

주문메뉴삭제는 주문된 음식을 삭제하고자 할 때 사용하는 기능이다. 관리자는 음식메뉴삭제버튼을 클릭하여 table, order를 거쳐 orderDetail클래스에 오퍼레이션을 요청하면 주문메뉴가 삭제 되는데 이러한 기능을 'deleteOrderMenu' 오퍼레이션으로 식별한다.

주문메뉴 수량 설정은 이미 테이블에 주문된 메뉴의 수량을 설정하고자 할 때 사용하는 기능이다. 관리자는 음식메뉴수량증가 버튼 혹은 음식메뉴수량감소 버튼을 클릭하여 주문내역에 음식의 수량설정을 요청하는데 이를 'deleteOrderMenu' 혹은 'addOrderMenu' 오퍼레이션으로 식별한다.

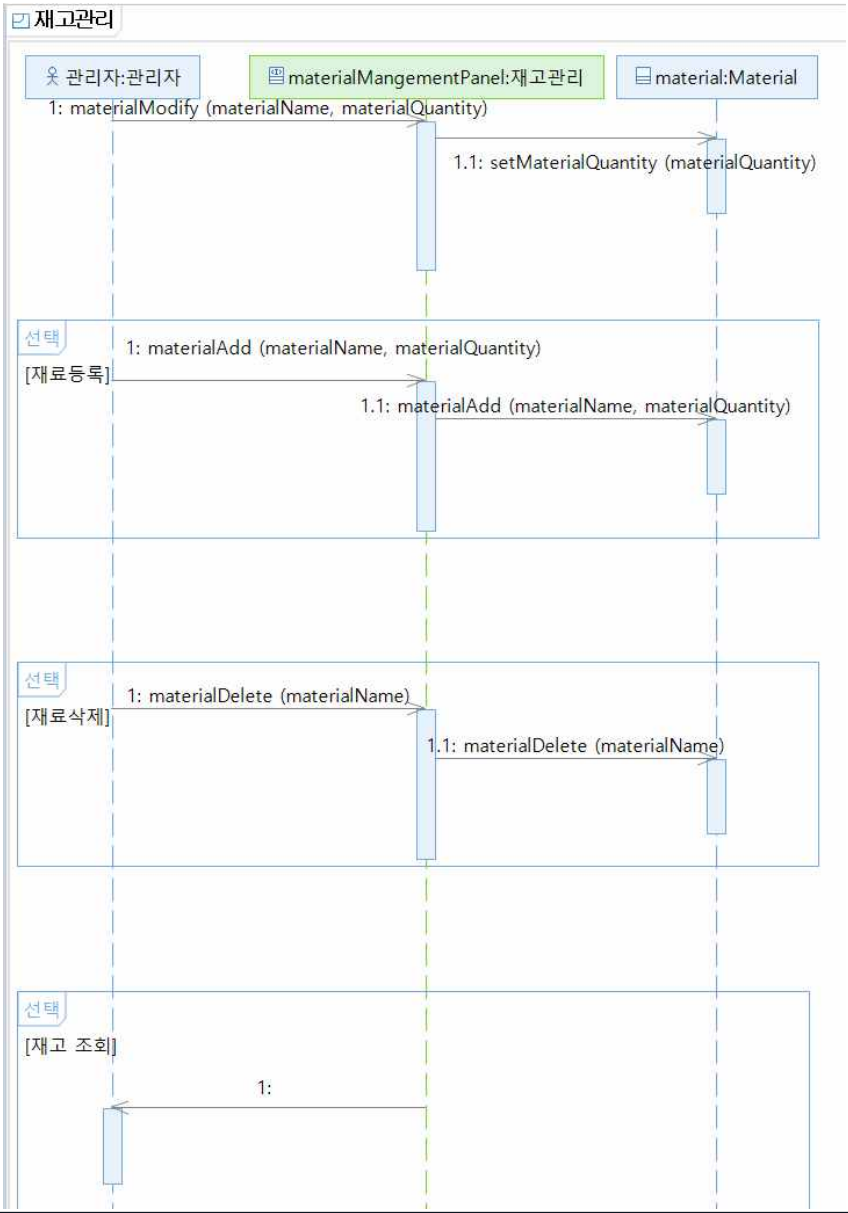
테이블 주문 내역 확인 기능



설 명

관리자는 각 테이블의 주문 내역을 확인 할 수 있다. 조회를 원하는 테이블을 선택하면 해당 테이블의 번호를 TablePanel 클래스에 넘겨주고, Table, Order, Orderdetail 클래스를 거쳐 음식명과 음식수량, 총주문금액을 반환한다. 이 기능을 'accessOrder' 오퍼레이션으로 식별한다.

재고 관리 기능



설 명

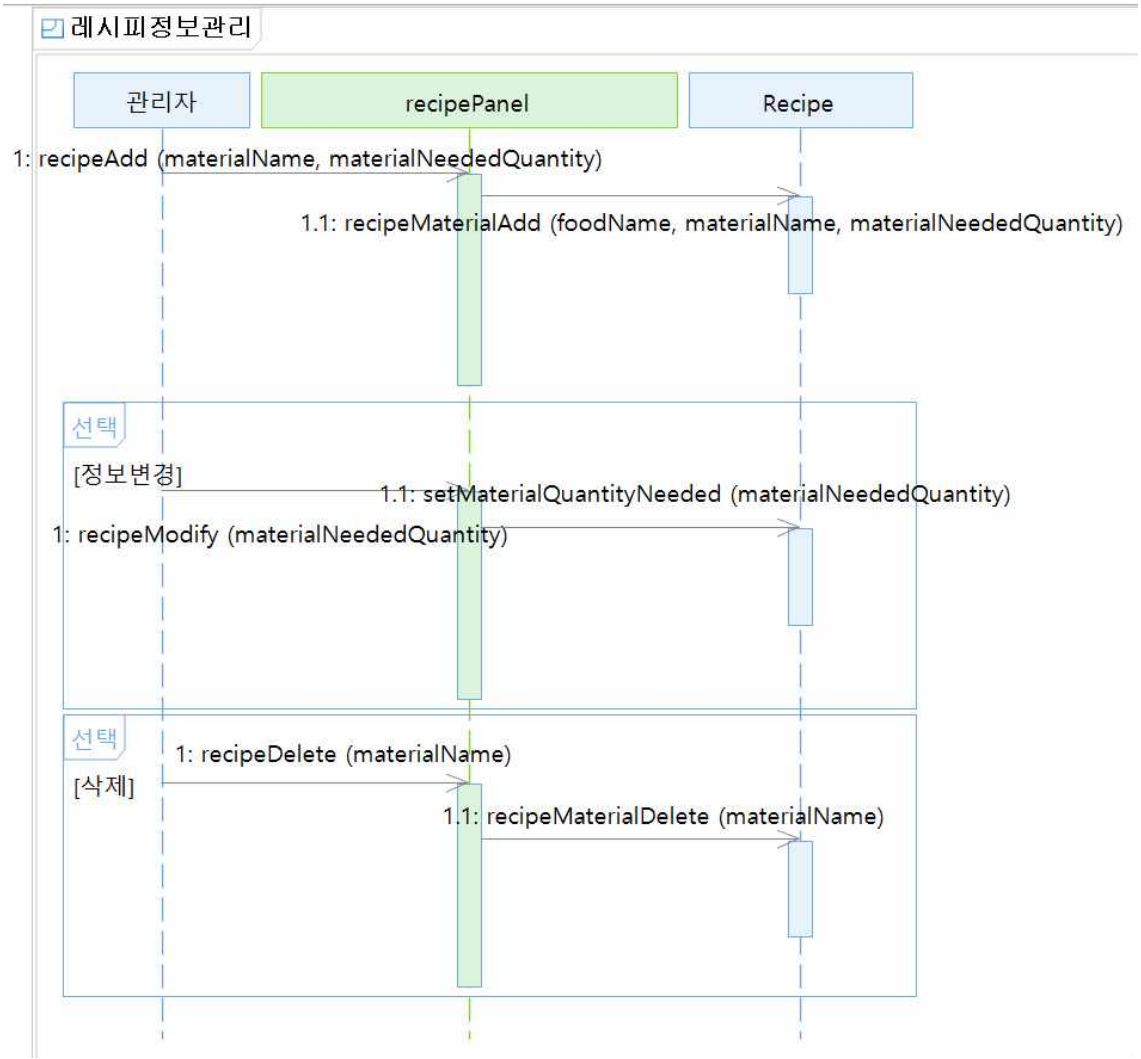
시스템은 재료들의 목록을 보여주고 수정을 할 수 있게 해준다. 재료목록을 보기 위해서는 재료명과 재료량이 필요한데, 이 정보는 'Material' 클래스에 저장되어 있다. 이들 정보를 관리자에게 수정하기 위해 관리자는 수정하고자 하는 재료의 재료명과 재료량을 입력값으로서 제공해야 한다. 가지고 있는 재료중 입력값으로 받은 재료명과 일치하는 재료가 있다면 해당 재료의 재료량을 수정한다. 이러한 역할을 하는 'Material' 클래스의 오퍼레이션을 'setMaterialQuantity'로 식별한다.

선택한 재료의 등록을 원하는 경우 관리자는 Material클래스에 재료명과 재료량을 입력한다. 이러한 역할을 하는 재료클래스의 오퍼레이션을 'materialAdd'로 식별한다.

선택한 재료를 삭제하고자 할 경우 관리자는 삭제하고자 하는 재료명을 입력한다. 이러한 역할을 하는 Material클래스의 오퍼레이션을 'materialDelete'로 식별한다.

재료의 재고을 조회하고자 하는 경우 MaterialManegementPanel 클래스에서 관리자에게 재고량을 보여준다.

레시피 정보 관리 기능



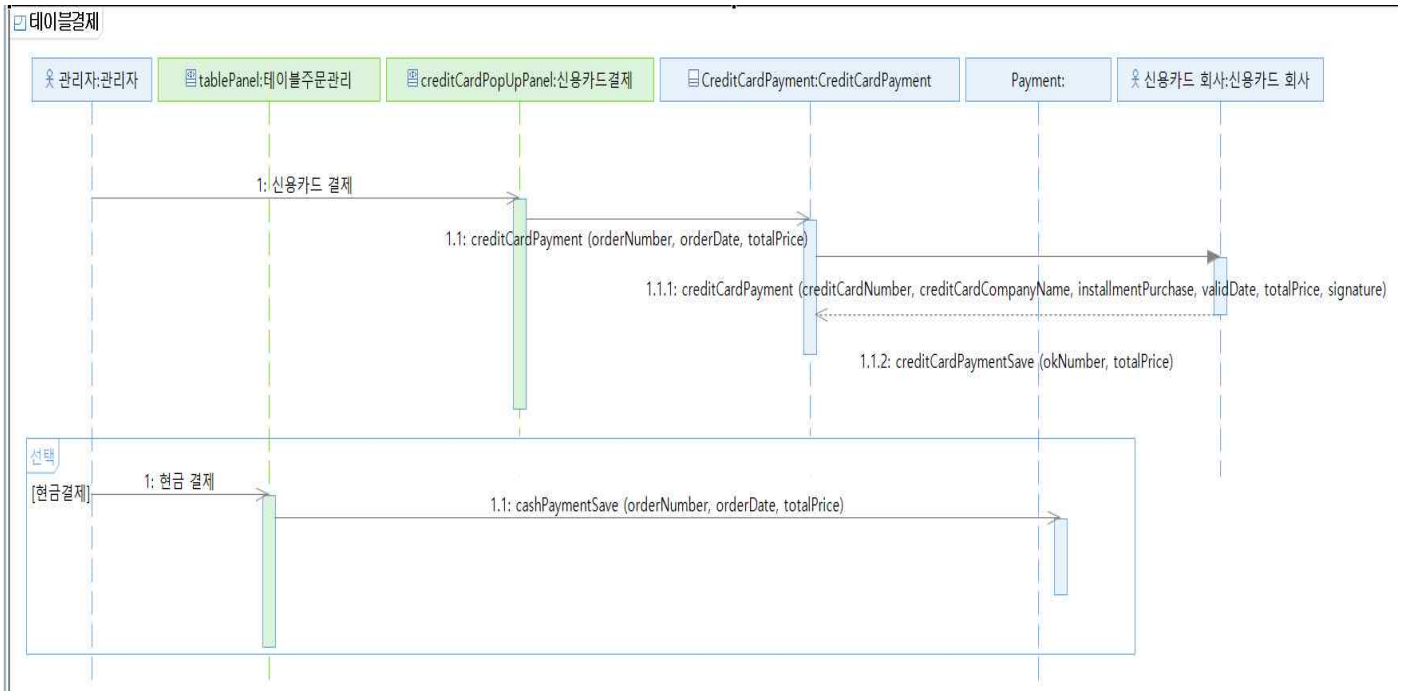
설 명

시스템은 각 음식의 레시피의 정보를 등록할 수 있게 해준다. 음식의 레시피 정보에는 음식명, 재료명과 필요양이 포함되며 Recipe 클래스에 저장되어 있다. 이 정보들을 등록하기 위해 관리자는 음식명, 재료명, 필요량을 입력한다. 그리고 Recipe클래스에 레시피 정보가 등록된다. 이러한 역할을 하는 레시피 클래스의 오퍼레이션을 'recipeMaterialAdd'로 식별한다.

또한, 시스템은 각 음식의 레시피 정보를 변경할 수 있게 해준다. 레시피 정보를 변경하기 위해 관리자는 필요량을 입력한다. 값이 제공되면 Recipe클래스는 변경된 정보를 저장한다. 이러한 역할을 하는 Recipe 클래스의 오퍼레이션을 'setMaterialQuantityNeeded'로 식별한다.

마지막으로, 시스템은 각 음식의 레시피 정보를 삭제할 수 있게 해준다. 레시피 정보를 삭제하기 위해서 관리자는 재료명을 입력한다. 그리고 Recipe 클래스에서 해당 레시피 정보가 삭제된다. 이러한 역할을 하는 Recipe클래스의 오퍼레이션을 'recipeMaterialDelete'로 식별한다.

테이블 결제 기능

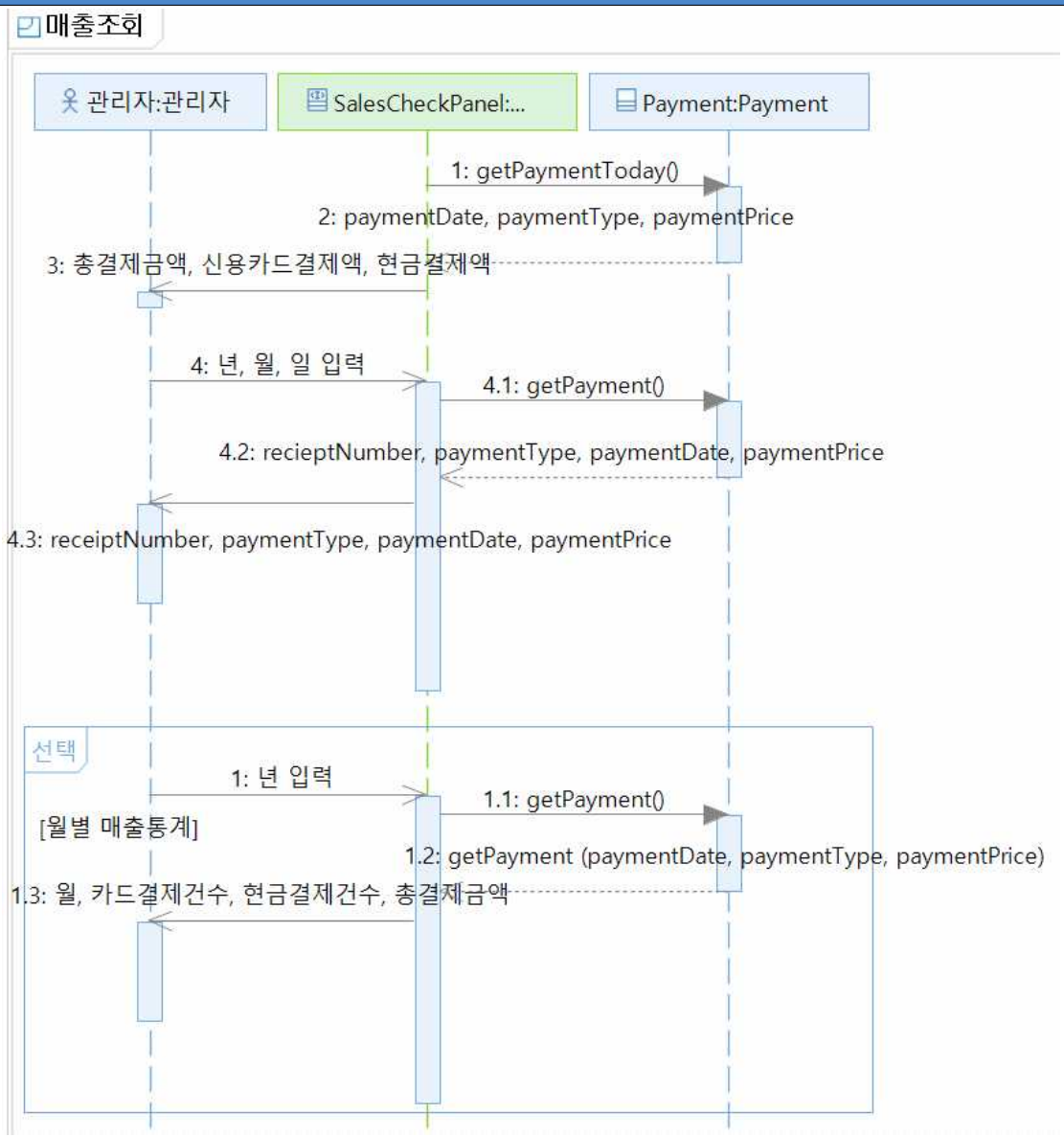


설 명

관리자는 결제를 위해서 결제를 처리할 건의 영수증 번호와 결제방식, 고객서명을 CreditCardPayment클래스로 전달한다. 이 오퍼레이션을 'creditCardPayment' 오퍼레이션으로 식별하였다. CreditCardPayment 클래스에서는 신용카드의 결제를 위해 신용카드 회사에 신용카드에 대한 기본정보(신용카드사, 카드번호, 할부개월, 유효개월)와 함께 결제해야할 결제 금액과 고객의 서명을 전송한다. 이것을 "creditCardPayment" 오퍼레이션으로 식별한다. 신용카드 회사는 본 시스템 외부에 있는 요소이므로 액터로 표현하였다. 신용카드회사에서 결제승인이 되면, 다시 creditCardPayment 클래스로 신용카드사이름과 카드번호, 그리고 승인번호와 결제금액이 반환되며 그 결과가 저장된다. 승인번호가 반환되었다는 것은 결제방식이 카드로 된 것을 의미하며 결제 상태가 승인된 상태이다. 또한 결제가 승인된 날짜가 곧 결제 일자가 된다.

현금 결제의 경우, 관리자가 결제를 시도하게 되면, 관리자가 영수증번호와 결제방식, 결제상태, 받은 금액, 거스름돈을 Payment 클래스로 전달한다. 신용카드회사처럼 별도의 승인을 받지 않아도 되므로 바로 결제 방식은 현금이며, 결제상태 및 결제일자 또한 바로 확인할 수 있다. 현금결제 클래스에서는 이러한 역할을 하는 오퍼레이션으로 "cashPaymentSave"를 식별한다.

매출 조회 기능

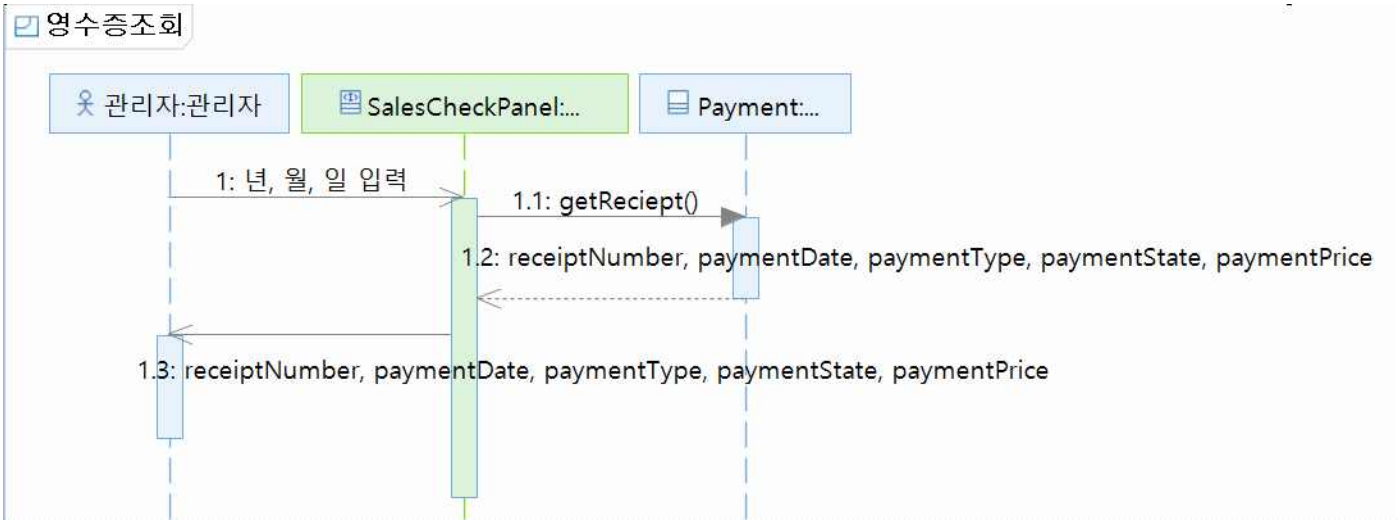


설 명

시스템은 점포의 매출을 조회할 수 있게 해준다. 매출조회 기능을 실행할 때, 점장은 원하는 조회일자를 입력한다. 그리고 'Payment' 클래스는 결제일자, 영수증번호, 결제방식, 결제금액을 결과값으로서 반환한다. 이 역할을 하는 오퍼레이션을 'getPayment'로 식별한다.

또한, 시스템은 점포의 월별 매출통계를 보여준다. 월별 매출통계 기능을 실행할 때, 관리자는 원하는 조회 년(year)를 입력한다. 그리고 'Payment' 클래스는 월(month), 결제방식, 결제금액을 결과값으로서 반환한다. 이 역할을 하는 오퍼레이션을 'getPayment'로 식별한다.

영수증 조회 기능



설 명

관리자는 원하는 날짜의 영수증을 조회를 할 수 있다. 관리자는 조회하고자 하는 결제일자를 입력하고 조회를 요청하는데 이 같은 오퍼레이션을 'getRecipt'로 식별한다.

영수증 조회를 실행하면 Payment클래스에서 결제정보를 받아와 관리자에게 영수증 번호와 함께 보여준다.

음식점 POS 시스템

클래스 명세서

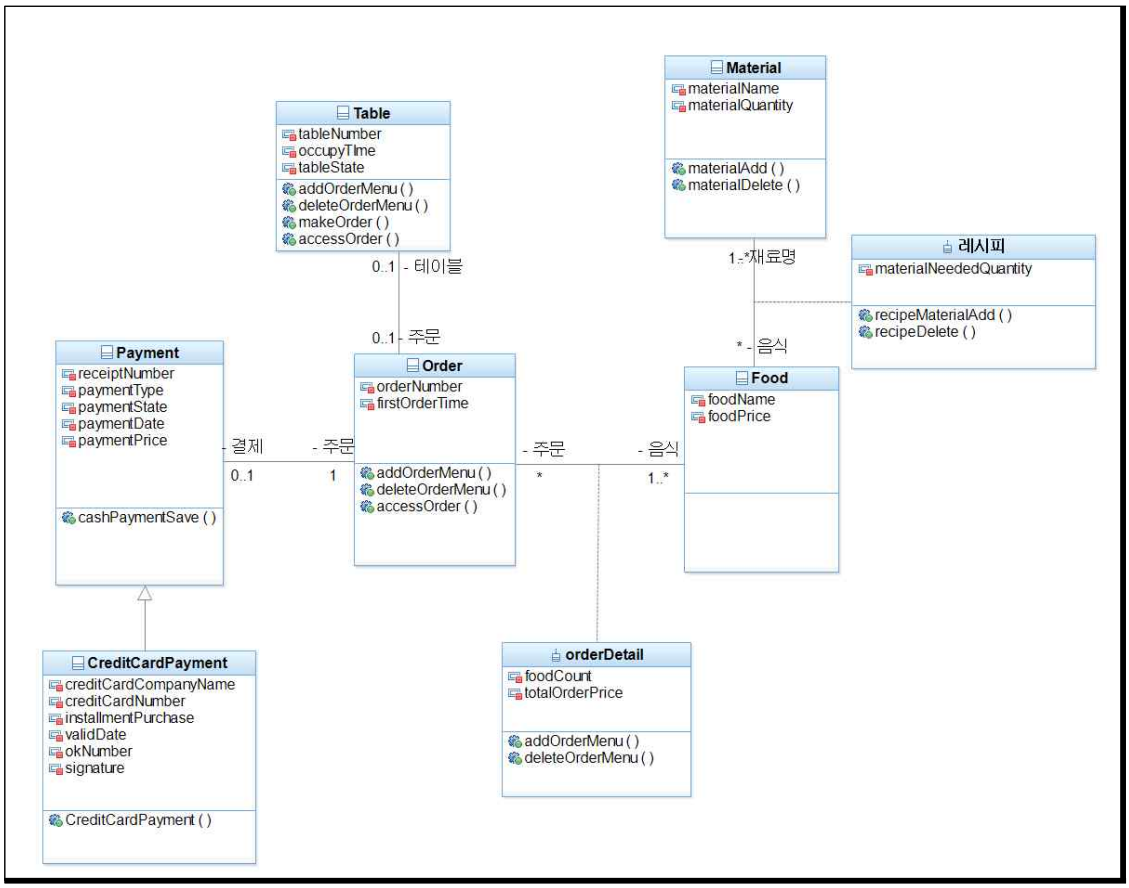
4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

목 차

제 1 장 클래스 다이어그램 업데이트 내역

업데이트 일자	업데이트 내역
2016. 12. 16	1차 수정 클래스 명세서 내역 갱신 클래스 다이어그램 교체

제 2 장 클래스 다이어그램



제 3 장 클래스 명세서
제 1 절 테이블

클래스명	Table
클래스 설명	테이블에 대한 정보를 가지고 있다.

1. 속성 정의

속성 명	자료형	Key	설명
tableNumber	int	O	테이블의 번호
tableState	int		테이블 상태
occupyTime	long		테이블 점유 시간

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
addOrderMenu	food		테이블에 주문한 메뉴를 추가. 매개변수는 음식
deleteOrderMenu	food		테이블에 주문한 메뉴를 삭제. 매개변수는 음식
makeOrder			주문번호를 가진 주문이 생성됨.
accessOrder		accessOrder	주문내역을 반환함

3. 관계 정보

관련 클래스	설명
Order	테이블에 주문번호를 부여하며 테이블에 주문된 음식들을 추가하거나 삭제한다.

제 2 절 재료

클래스명	재료
클래스 설명	점포에서 보유하고 있는 재료에 관한 정보를 가진 클래스

1. 속성 정의

속성 명	자료형	Key	설명
materialName	String	O	재료의 이름
materialQuantity	float		재료의 양
materialList	ArrayList<Material>		재료의 정보를 저장

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
materialAdd	materialName, materialNe		재료를 추가한다
materialDelete	materialName		재료를 삭제한다.
setMaterialQuantity	materialQuantity		재료량을 수정한다.

3. 관계 정보

관련 클래스	설명
Recipe	점포에서 판매하고 있는 음식의 레시피를 관리한다.

제 3 절 레시피

클래스명	레시피
클래스 설명	점포에서 판매하고 있는 음식의 레시피를 관리한다. 재료 클래스와 음식 클래스의 관계 클래스

1. 속성 정의

속성 명	자료형	Key	설명
foodName	String	O	음식의 이름
materialName	String	O	재료의 이름
materialNeededQuantity	float		음식이 만들어지기 위한 재료의 필요양
recipeList	ArrayList<Recipe>		레서피의 정보가 저장되는 변수

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
recipeMaterialAdd	foodName, materialName, materialNeededQuantity		레시피를 등록한다.
recipeMaterialDelete	materialName		레시피를 삭제한다
setMaterialNeededQuantity	materialNeededQuantity		레시피의 필요양을 수정한다.

3. 관계 정보

관련 클래스	설명
Material	점포에서 보유하고 있는 재료에 관한 정보를 가진 클래스
Food	점포에서 판매하고 있는 음식에 관한 정보를 가진 클래스

제 4 절 음식

클래스명	Food
클래스 설명	점포에서 판매하고 있는 음식의 정보를 가진 클래스

1. 속성 정의

속성 명	자료형	Key	설명
foodName	String	O	점포에서 판매하는 음식의 이름
price	int		점포에서 판매하는 음식의 가격

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
오퍼레이션 없음			

3. 관계 정보

관련 클래스	설명
Material	점포에서 보유하고 있는 재료에 관한 클래스
Recipe	점포에서 판매하고 있는 음식의 레시피의 정보에 관한 클래스
Order	테이블에 주문번호를 부여하는 클래스
OrderDetail	테이블의 주문한 내역을 확인하는 클래스

제 5 절 주문

클래스명	Order
클래스 설명	테이블에 주문번호를 부여하는 클래스

1. 속성 정의

속성 명	자료형	Key	설명
orderNumber	long	O	테이블에 주문할 때마다 부여하는 번호
firstOrderTime	long		주문번호가 부여되는 첫 시간

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
addOrderMenu	food		주문메뉴를 추가
deleteOrderMenu	food		주문한 메뉴를 삭제
accessOrder		orderTableModel	UI에 보여주기 위한 주문내역을 리턴

3. 관계 정보

관련 클래스	설명
Table	테이블에 관한 정보를 가진 클래스
Food	점포에서 판매하고 있는 음식의 정보를 가진 클래스
OrderDetail	테이블에서 주문한 음식의 내역의 정보를 가진 클래스
Payment	주문번호가 부여된 주문내역을 결제하는 클래스

제 6 절 주문내역

클래스명	OrderDetail
클래스 설명	테이블에서 주문한 음식의 내역의 정보를 가진 클래스 주문 클래스와 음식 클래스의 관계 클래스

1. 속성 정의

속성 명	자료형	Key	설명
totalOrderPrice	int	O	총 주문 금액
foodCount	int		음식량

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
addOrderMenu	Food food		주문 메뉴 추가
deleteOrderMenu	Food food		주문 메뉴 삭제

3. 관계 정보

관련 클래스	설명
Order	테이블에 주문번호를 부여하는 클래스
Food	점포에서 판매하고 있는 음식의 정보를 가진 클래스

제 7 절 결제

클래스명	Payment
클래스 설명	주문번호에 따른 결제내역을 결제하는 클래스

1. 속성 정의

속성 명	자료형	Key	설명
receiptNumber	int	O	주문내역에 따른 영수증의 번호
paymentType	int		현금결제인지 카드결제인지의 구분 현금 : 0, 카드 : 1
paymnetState	boolean		결제상태가 잘되었는지 보류되었는지의 상태
paymentDate	int		결제된 날짜
paymentPrice	int		결제 금액
paymentList	Payment		결제에 대한 정보를 저장

2. 오퍼레이션 정의

오퍼레이션 명	매개변수	반환값	설명
cashPayment Save	Order order		현금결제시 정보를 저장하는 함수

3. 관계 정보

관련 클래스	설명
Order	테이블에 주문번호를 부여하는 클래스
CreditCardPayment	신용카드 결제 시 사용되는 정보를 가진 클래스

제 8 절 신용카드 결제

클래스명	CreditCardPayment
클래스 설명	신용카드 결제시 사용되는 정보를 가진 클래스. 결제클래스로부터 상속받는 하위 클래스.

1. 속성 정의

속성 명	자료형	Key	설명
receiptNumber	int	O	주문 내역에 따른 영수증의 번호
creditCardCompanyName	String		신용카드 회사 이름
creditCardNumber	int		신용카드 번호
installmentPurchase	int		할부결제 시 나눠내는 개월수
vaildDate	int		카드의 유효기간
okNumber	int		카드사에서 승인함에 따라 부여되는 번호
signature	boolean		고객의 서명 성공인지 실패인지

2. 오퍼레이션 정의

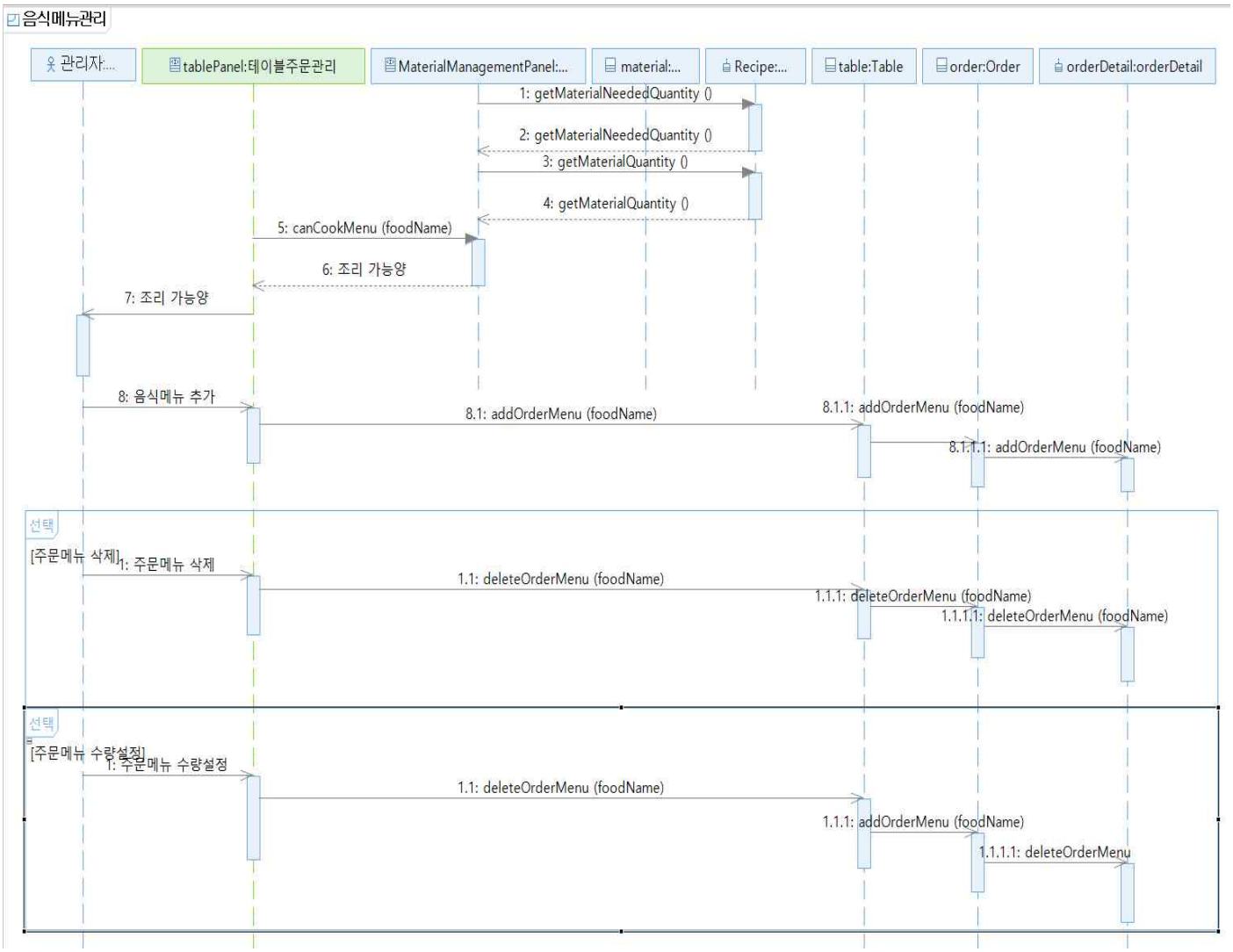
오퍼레이션 명	매개변수	반환값	설명
CreditCardPayment	Order		주문클래스에서 정보를 받아와 카드 결제시 결제정보를 저장한다.

3. 관계 정보

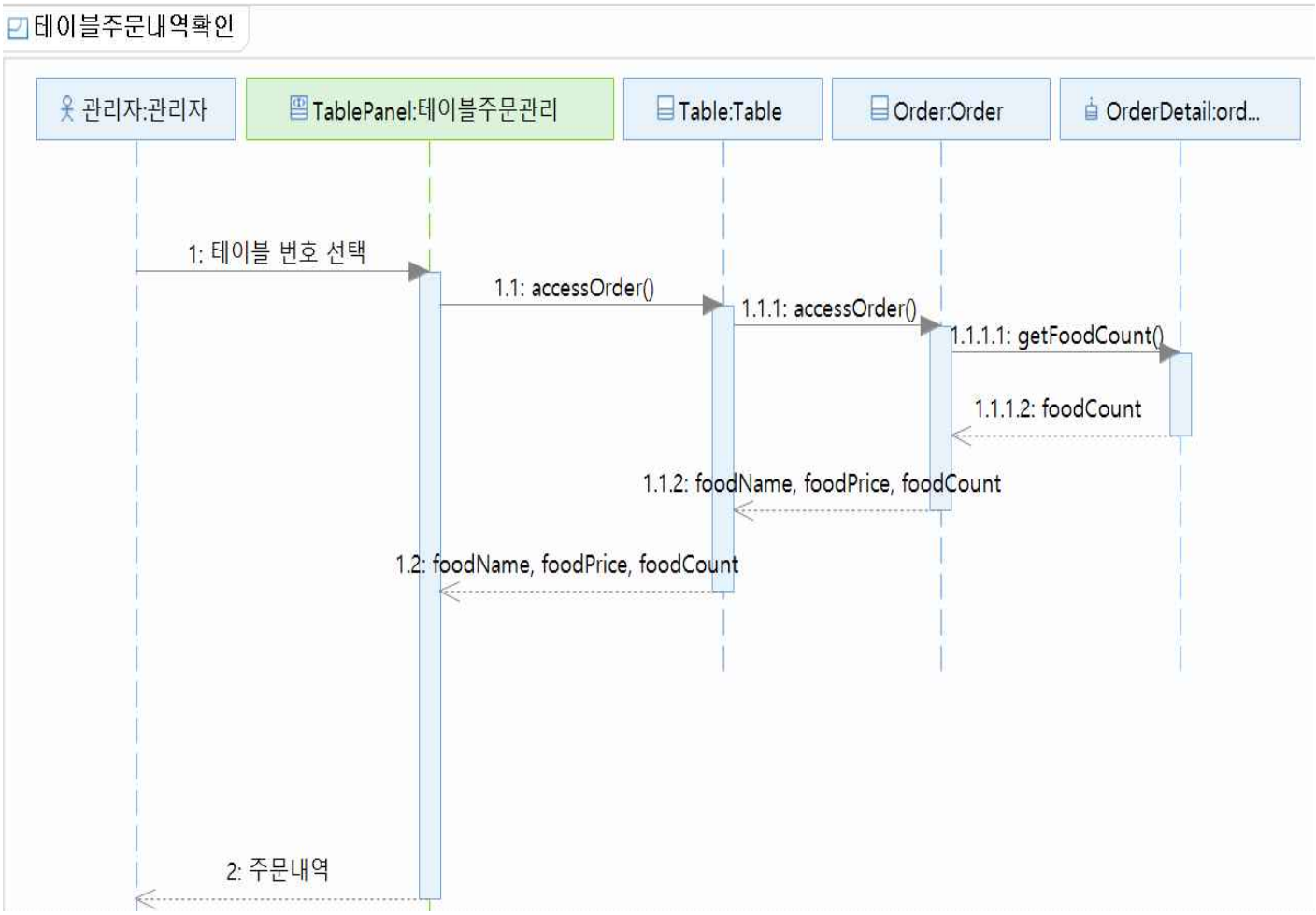
관련 클래스	설명
결제	주문 번호에 따른 결제내역을 결제하는 클래스

시퀀스 다이어그램 그림 (UI 클래스 포함)

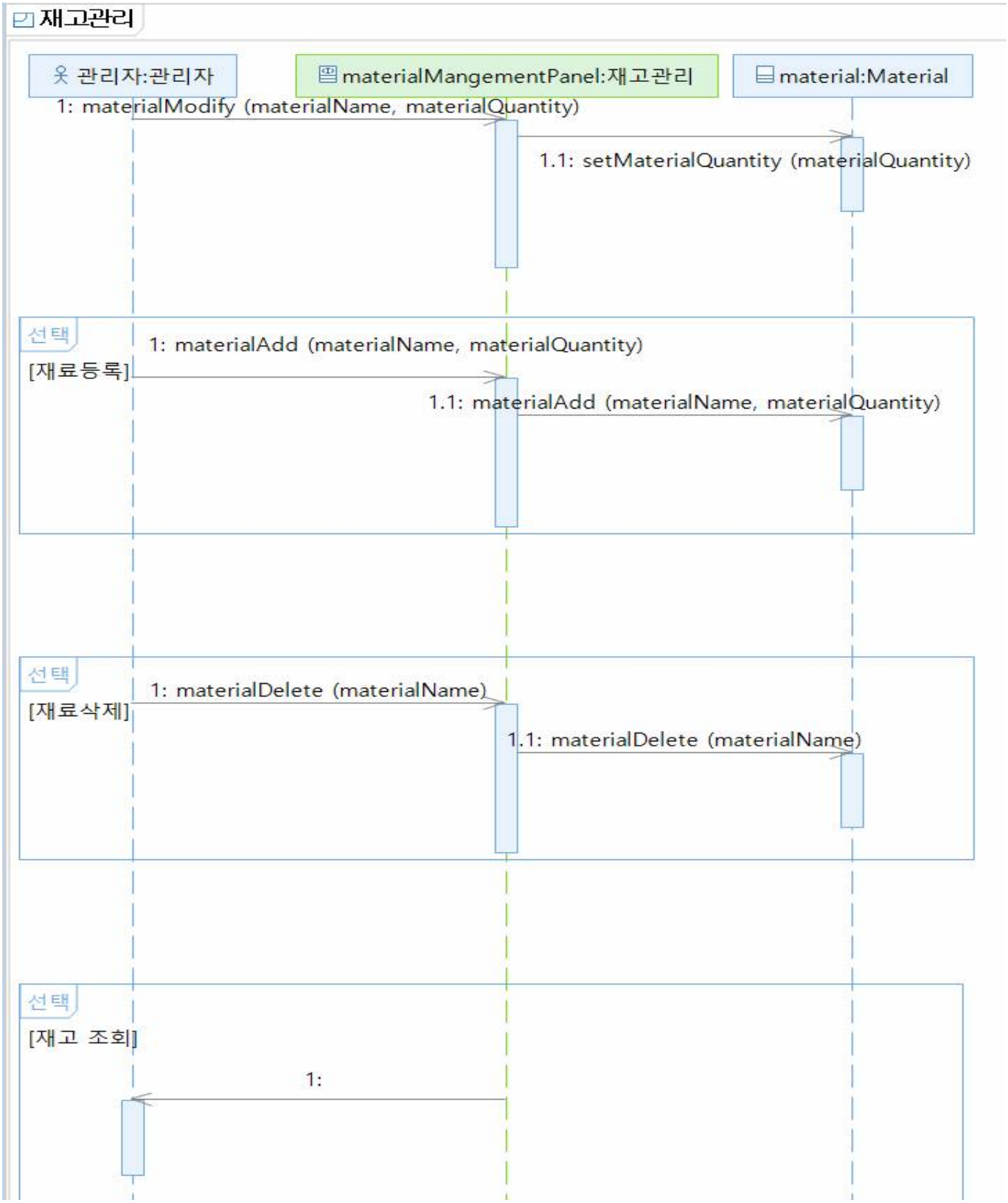
음식메뉴 관리기능



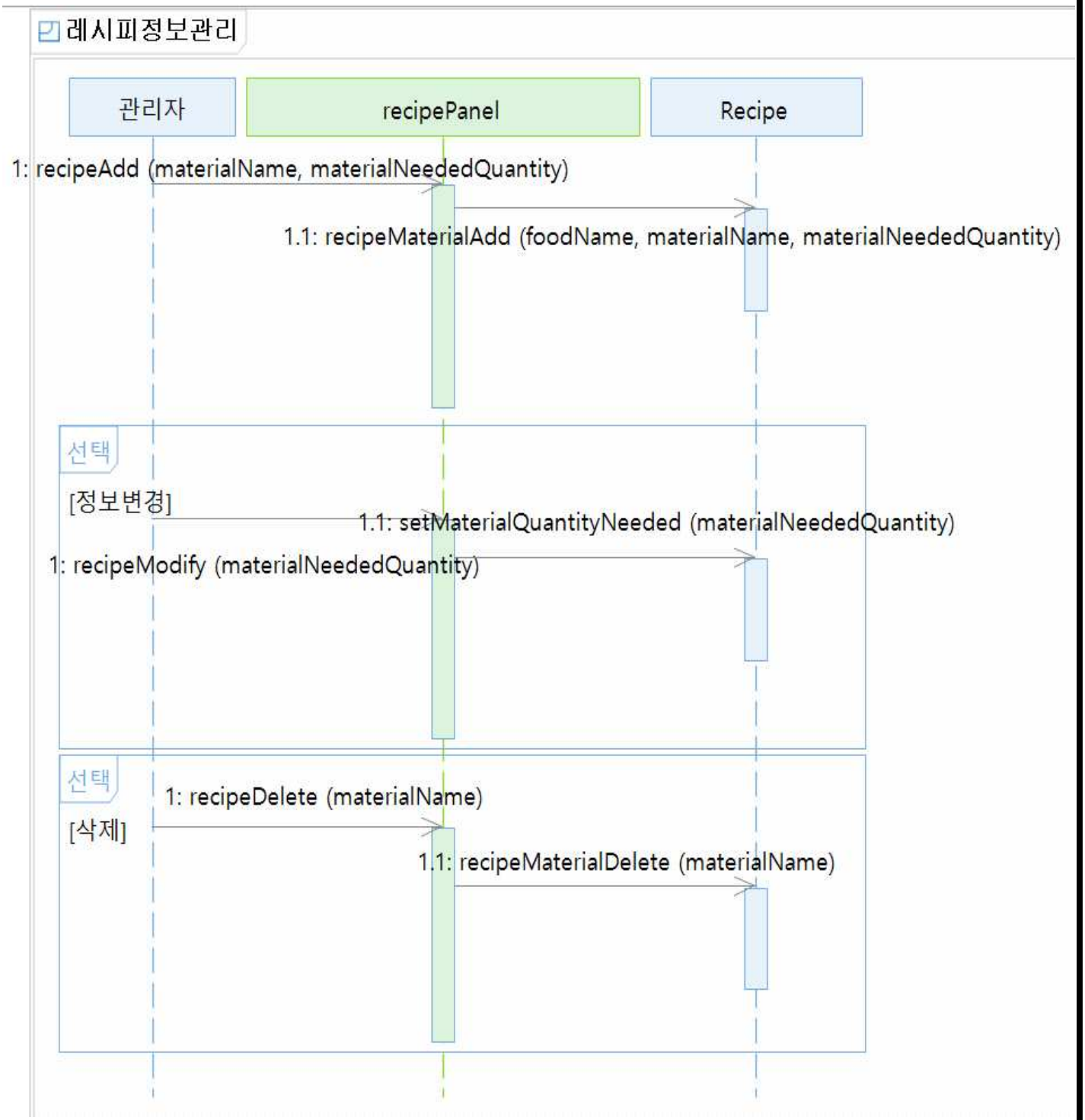
테이블 주문 내역 확인 기능



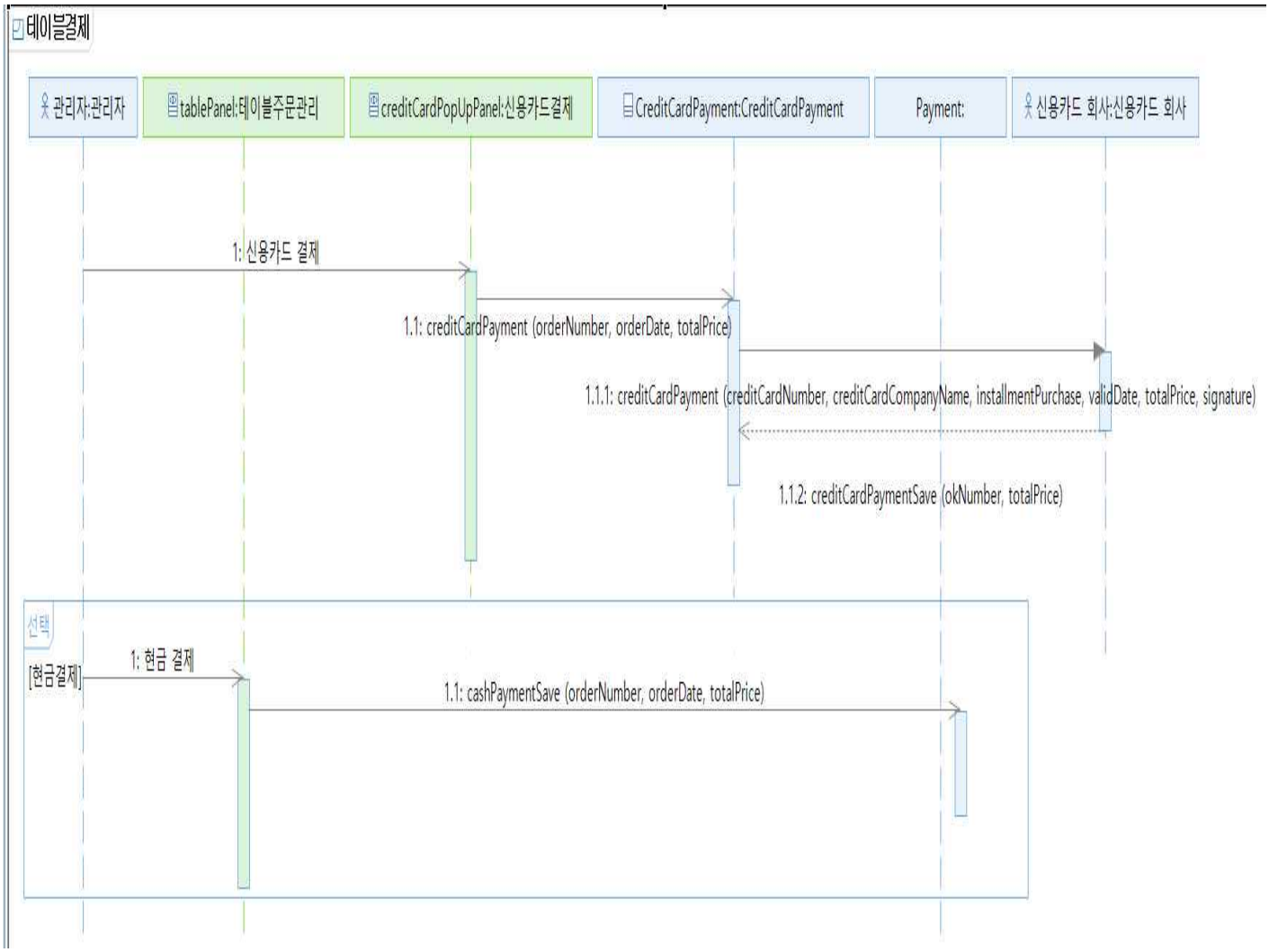
재고 관리 기능


재고관리
관리자:관리자
materialMangementPanel:재고관리
material:Material
1: materialModify (materialName, materialQuantity)
1.1: setMaterialQuantity (materialQuantity)
선택
[재료등록]
1: materialAdd (materialName, materialQuantity)
1.1: materialAdd (materialName, materialQuantity)
선택
[재료삭제]
1: materialDelete (materialName)
1.1: materialDelete (materialName)
선택
[재고 조회]
1:

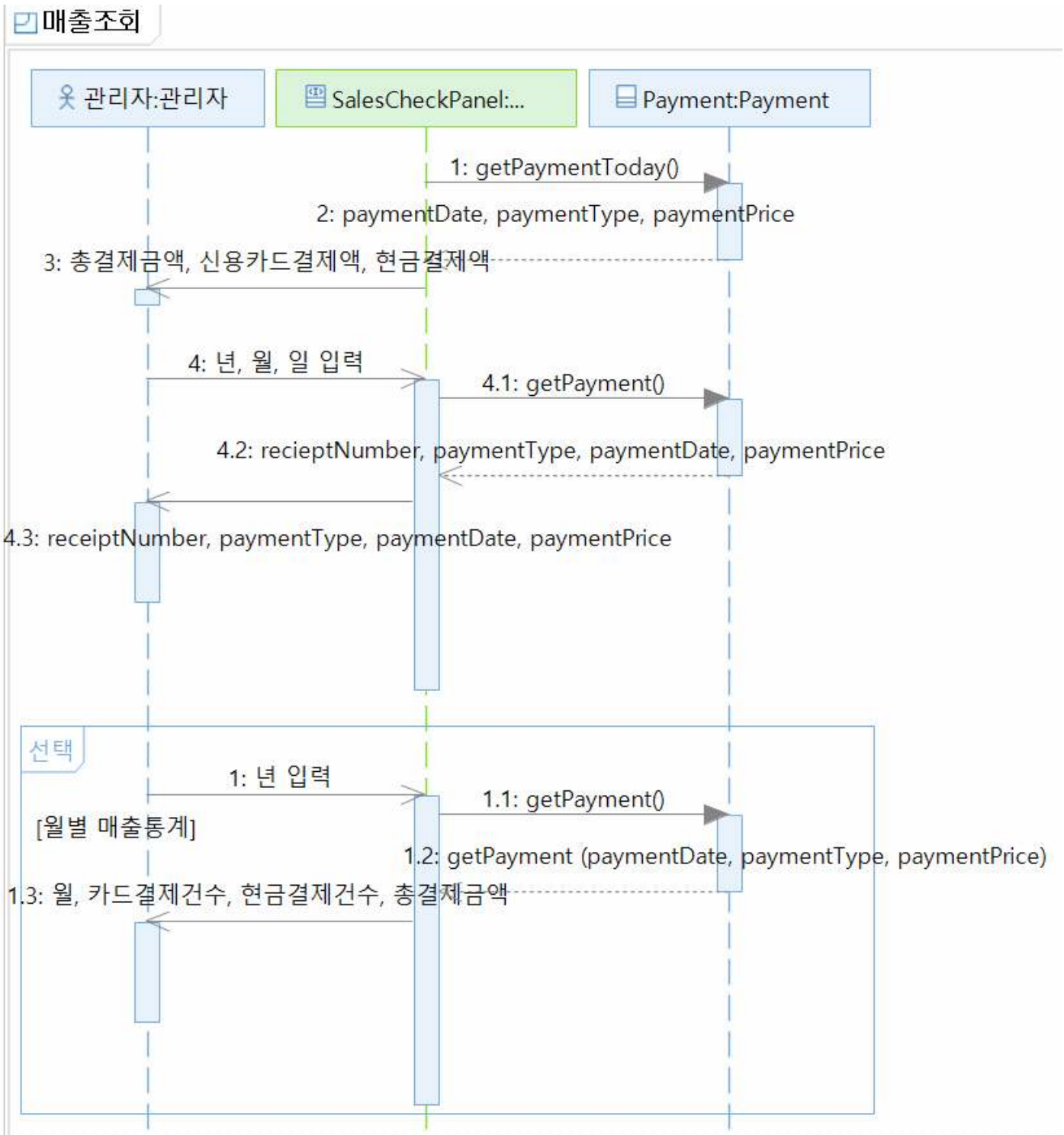
레시피 정보 관리 기능


레시피정보관리
관리자
recipePanel
Recipe
1: recipeAdd (materialName, materialNeededQuantity)
1.1: recipeMaterialAdd (foodName, materialName, materialNeededQuantity)
선택
[정보변경]
1.1: setMaterialQuantityNeeded (materialNeededQuantity)
1: recipeModify (materialNeededQuantity)
선택
1: recipeDelete (materialName)
[삭제]
1.1: recipeMaterialDelete (materialName)

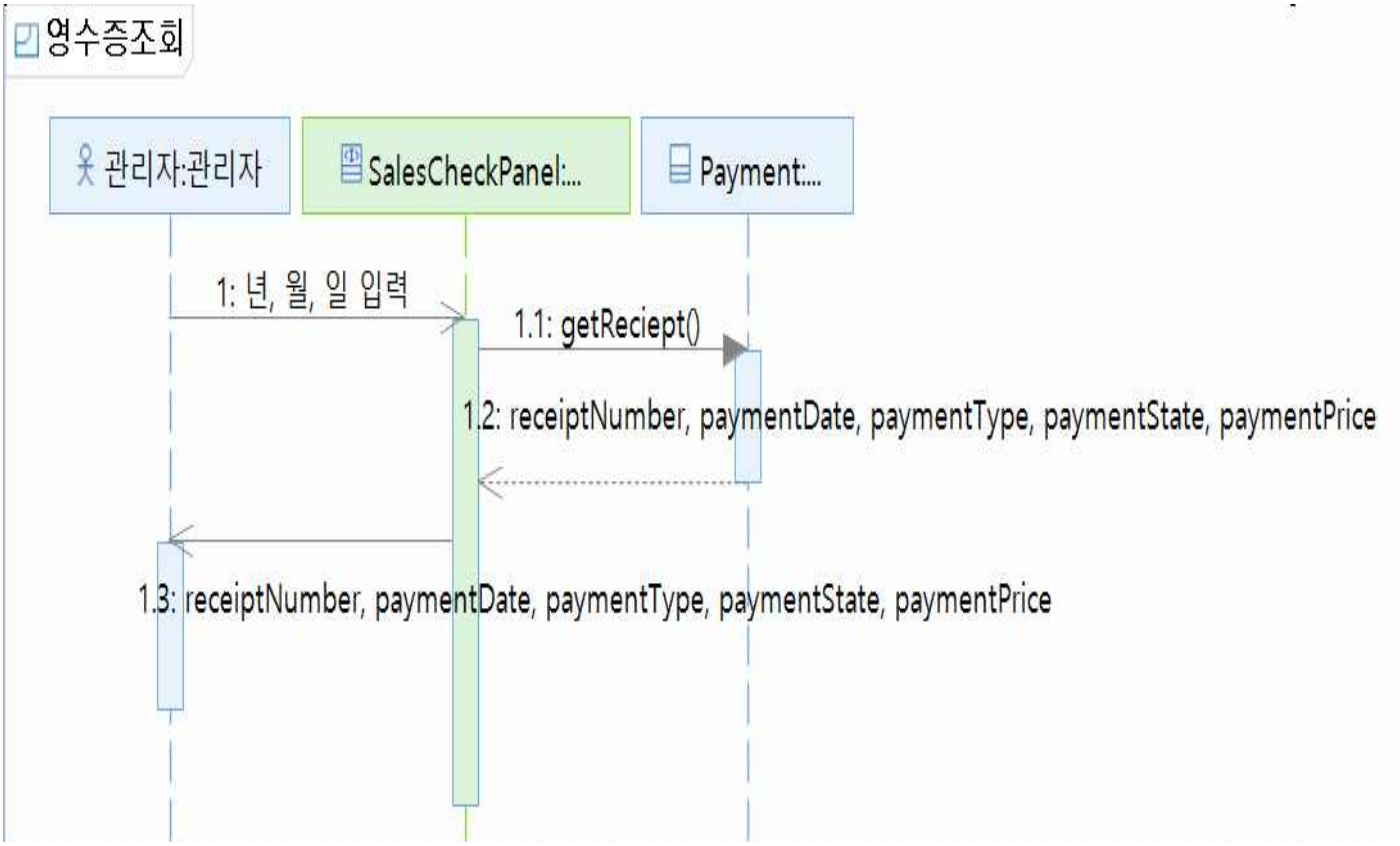
테이블 결제 기능


테이블결제
관리자:관리자
tablePanel:테이블주문관리
creditCardPopUpPanel:신용카드결제
CreditCardPayment:CreditCardPayment
Payment:
신용카드 회사:신용카드 회사
1: 신용카드 결제
1.1: creditCardPayment (orderNumber, orderDate, totalPrice)
1.1.1: creditCardPayment (creditCardNumber, creditCardCompanyName, installmentPurchase, validDate, totalPrice, signature)
1.1.2: creditCardPaymentSave (okNumber, totalPrice)
선택
[현금결제]
1: 현금 결제
1.1: cashPaymentSave (orderNumber, orderDate, totalPrice)

매출 조회 기능


매출조회
관리자:관리자
SalesCheckPanel:...
Payment:Payment
1: getPaymentToday()
2: paymentDate, paymentType, paymentPrice
3: 총결제금액, 신용카드결제액, 현금결제액
4: 년, 월, 일 입력
4.1: getPayment()
4.2: recieptNumber, paymentType, paymentDate, paymentPrice
4.3: receiptNumber, paymentType, paymentDate, paymentPrice
선택
1: 년 입력
1.1: getPayment()
[월별 매출통계]
1.2: getPayment (paymentDate, paymentType, paymentPrice)
1.3: 월, 카드결제건수, 현금결제건수, 총결제금액

영수증 조회 기능


영수증조회
관리자:관리자
SalesCheckPanel:...
Payment:...
1: 년, 월, 일 입력
1.1: getReciept()
1.2: receiptNumber, paymentDate, paymentType, paymentState, paymentPrice
1.3: receiptNumber, paymentDate, paymentType, paymentState, paymentPrice

음식점 POS 시스템

User Interface 명세서

4조			
조원	학번	학과	이름
	201000287	일어일문학과	유다훈
	201000506	문헌정보학과	배성진
	201401316	지질환경과학과	김한솔

목 차

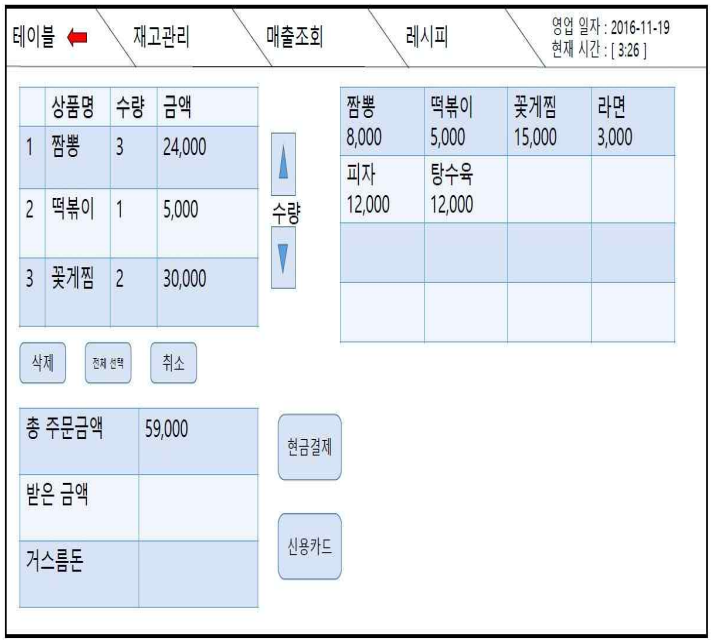
제 1 장 유저 인터페이스 명세서

화면명	메인 화면
화면 개요	테이블의 위치 및 번호와 주문시간을 알 수 있는 화면
관련 DAO 클래스	테이블 주문 주문내역
관련 유스케이스	테이블주문관리

입력	**출력**

오퍼레이션

화면명	테이블 주문 관리
화면 개요	테이블의 주문 내역 및 음식 추가, 수정, 삭제와 결제 기능을 제공하는 화면
관련 DAO 클래스	주문 결제 현금결제 주문내역 신용카드 결제
관련 유스케이스	테이블 주문 관리 테이블 결제



입력

출력

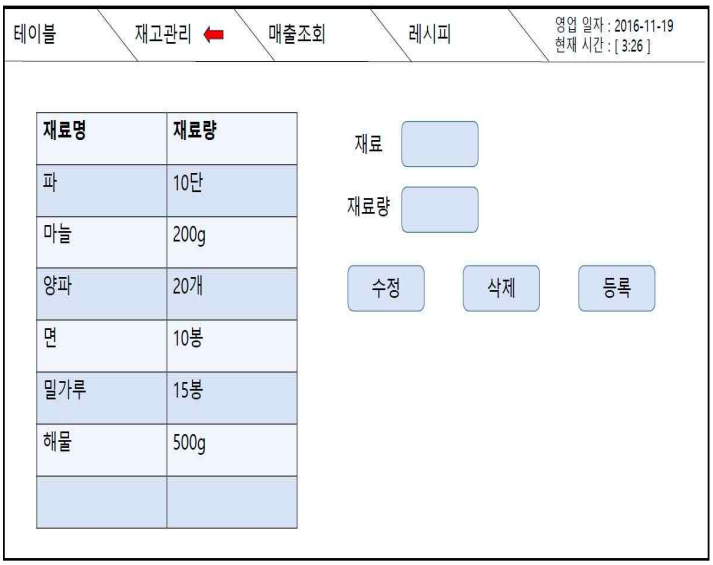
오퍼레이션

화면명	신용 카드 결제 화면
화면 개요	신용카드 결제시 팝업형태로 제공하며 각종 정보를 보여주는 화면
관련 DAO 클래스	신용카드 결제
관련 유스케이스	카드결제

입력	**출력**

오퍼레이션

화면명	재고 관리 화면
화면 개요	점포에서 가지고 있는 재료들을 관리할 수 있는 화면.
관련 DAO 클래스	재료
관련 유스케이스	재고 관리



입력	**출력**

오퍼레이션

화면명	구간별 매출 조회
화면 개요	점포의 매출을 조회해볼 수 있는 화면
관련 DAO 클래스	결제
관련 유스케이스	매출 조회

입력

출력

오퍼레이션

화면명	매출 조회 - 영수증 조회	
화면 개요	영수증을 조회할 수 있는 화면	
관련 DAO 클래스	결제	
관련 유스케이스	영수증 조회	

입력	출력

오퍼레이션

화면명	매출 조회 – 월별 매출 통계
화면 개요	월별 매출의 통계를 알 수 있는 화면
관련 DAO 클래스	결제
관련 유스케이스	월별매출통계

입력

출력

오퍼레이션

화면명	레시피 관리
화면 개요	점포에서 판매하는 음식의 레시피를 관리할 수 있는 화면
관련 DAO 클래스	레시피 재료 음식
관련 유스케이스	레시피 정보관리

입력	**출력**

오퍼레이션

객체지향설계 4조
클래스 다이어그램 소스코드

클래스 이름	Food	클래스 설명	음식

```
import java.util.ArrayList;

public class Food {
	private String foodName; //음식명
	private int price;		//가격'
	private ArrayList<Food> food = new ArrayList<Food>();

	public Food(){
		food.add(new Food("짜장면", 3000));
		food.add(new Food("짬뽕", 5000));
		food.add(new Food("볶음짬뽕", 7000));
		food.add(new Food("탕수육", 9000));
	}

	public Food(String foodName, int price) {
		this.foodName = foodName;
		this.price = price;
	}
	// getter 및 setter
	public String getFoodName() {
		return foodName;

	}

	public String getFoodName(int i) {
		return food.get(i).foodName;
	}

	public void setFoodName(String foodName) {
		this.foodName = foodName;
	}

	public int getPrice() {
		return price;
	}

	public void setPrice(int price) {
		this.price = price;
	}

}
```

클래스 이름	Material	클래스 설명	재료

```
import java.util.ArrayList;

public class Material {

	private String materialName;		//재료명
	private float materialQuantity; //재료량


	ArrayList<Material> materialList = new ArrayList<Material>();

	private MaterialManagementPanel materialPanel;


	public Material() {

	}

	public Material(String materialName, float materialQuantity) {

		this.materialName = materialName;
		this.materialQuantity = materialQuantity;

	}
	public Material(MaterialManagementPanel materialPanel, String materialName, String materialQuatity)
{ //생성자


		this.materialPanel = materialPanel;
		float x = Float.parseFloat(materialQuatity);
		materialList.add(new Material(materialName, x));

	}

	/* 게터 새터 */
	public String getMaterialName() {

		return materialName;
	}

	public void setMaterialName(String materialName) {
```

```
		this.materialName = materialName;

	}

	public float getMaterialQuantity() {
		return materialQuantity;

	}

	public void setMaterialQuantity(float materialQuantity) {
		this.materialQuantity = materialQuantity;

	}

	void materialAdd(String materialName, float materialQuantity) {


		materialList.add(new Material(materialName, materialQuantity ));

	}

	void materialDelete(String materialName) {
		if(materialList.contains(materialName)) {
			int i = materialList.indexOf(materialName);
			materialList.remove(i);

		}
	}

}
```

클래스 이름	Recipe	클래스 설명	레시피

```
import java.util.ArrayList;

public class Recipe {

	private String foodname; //음식이름
	private String materialName; //재료이름
	private float materialNeededQuantity; //재료량

	ArrayList<Recipe> recipeList = new ArrayList<Recipe>();


//	private ArrayList<String> foodNameList = new ArrayList<String>();


	public Recipe() {

	}

	public Recipe(String foodName, String materialName, float materialNeededQuantity ) { //생성자
		this.foodname = foodName;
		this.materialName = materialName;

		this.materialNeededQuantity = materialNeededQuantity;

	}

	/* 게터, 새터 */


	public void setMaterialName(String materialName) {
		this.materialName = materialName;

	}

	public void setMaterialNeededQuantity(float materialNeededQuantity) {
		this.materialNeededQuantity = materialNeededQuantity;



	}

	public String getMaterialName() {
```

```
                return materialName;
        }

        public float getMaterialNeededQuantity() {
                return materialNeededQuantity;
        }

        public void recipeMaterialAdd(String foodName, String materialName, float materialNeededQuantity) {

                recipeList.add(new Recipe(foodName, materialName, materialNeededQuantity));

        }

        public void recipeMaterialDelete(String materialName) {
                if(recipeList.contains(materialName)) {
                        int i = recipeList.indexOf(materialName);
                        recipeList.remove(i);

                }

        }

}
```

클래스 이름	Order	클래스 설명	주문

```
import java.util.ArrayList;
import java.util.List;
import javax.swing.table.DefaultTableModel;

public class Order {
	//변수
	private long orderNumber; //주문번호
	private long firstOrderTime;   //첫주문시간

	//Food 객체 생성
	private Food jja = new Food("짜장면", 3000);
	private Food jjam = new Food("짬뽕", 5000);
	private Food bok = new Food("볶음짬뽕", 7000);
	private Food tang = new Food("탕수육", 9000);

	private OrderDetail _orderDetail;

	//For table
	private String[] columnNames = {"번호", "상품명","금액", "수량"};
	private String[][] rowDatas = {};
	private DefaultTableModel orderTableModel = new DefaultTableModel(rowDatas,
columnNames);
	private ArrayList<OrderDetail> orderList = new ArrayList<OrderDetail>();

	// 생성자
	public Order() {
		java.text.SimpleDateFormat                                    formatter=new
java.text.SimpleDateFormat("yyyyMMddHHmmss");
		String strCurrentTime = formatter.format(new java.util.Date());
		orderNumber = Long.parseLong(strCurrentTime);
		firstOrderTime = System.currentTimeMillis ();
		_orderDetail = new OrderDetail(this.orderNumber);
	}
```

```
	//메서드
	public void addOrderMenu (Food food) { //주문메뉴 추가
			String lists[] = new String[4];
			int index = 0;
			boolean exist = false;
			for (int i = 0 ; i < orderTableModel.getRowCount() ; i++) {
				if (orderTableModel.getValueAt(i,
1).equals(_orderDetail.getFoodName(food))) {
					exist = true;
					index = i;
				}
			}
			if (exist == true) {
				_orderDetail.addOrderMenu(food);
				int foodCount = _orderDetail.getFoodCount(food);
				orderTableModel.setValueAt(foodCount, index, 3);
				}
			else {
				int number = orderTableModel.getRowCount();
				_orderDetail.addOrderMenu(food);
				lists[0] = Integer.toString(number);
				lists[1] = _orderDetail.getFoodName(food);
				lists[2] = Integer.toString(_orderDetail.getFoodPrice(food));
				lists[3] = Integer.toString(_orderDetail.getFoodCount(food));
				orderTableModel.addRow(lists);
				}
	}
	void deleteOrderMenu (Food food) { //주문메뉴 삭제
		int index = 0;
		boolean exist = false;
		for (int i = 0 ; i < orderTableModel.getRowCount() ; i++) {
			if (orderTableModel.getValueAt(i,
1).equals(_orderDetail.getFoodName(food))) {
				exist = true;
				index = i;
```

```
                        }
                }
                _orderDetail.deleteOrderMenu(food);
                int foodCount = _orderDetail.getFoodCount(food);
                orderTableModel.setValueAt(foodCount, index, 3);
        }
        public DefaultTableModel accessOrder() {            //UI 에 보여주기 위한 주문내역 return (상품명, 금액, 수량)
                return orderTableModel;
        }



        // getter & setter
        long getOrderNumber() {
                return orderNumber;
        }
        long getfirstOrderTime() {
                return firstOrderTime;
        }
        public int getOrderDate() { //orderNumber 는 yyyyMMddHHmmss return 160302 이런식으로 보내주기
                String orderNumber = String.valueOf(this.orderNumber);
                String orderDate = orderNumber.substring(2, 8);
                int date = Integer.parseInt(orderDate);
                return date;

        }
        int getTotalPrice() {
                return _orderDetail.calculatingTotalPrice();
        }

}
```

클래스 이름	OrderDetail	클래스 설명	주문내역

```
import java.util.ArrayList;

public class OrderDetail {
	//변수
	private long orderNumber;                              //주문번호
	private int totalOrderPrice;          //총주문금액
	private int[] foodCount = new int[4]; //음식량
	// 0: 짜장면 1: 짬뽕 2: 볶음짬뽕 3:탕수육

	//Food 객체
	private Food jja = new Food("짜장면", 3000);
	private Food jjam = new Food("짬뽕", 5000);
	private Food bok = new Food("볶음짬뽕", 7000);
	private Food tang = new Food("탕수육", 9000);

	// 생성자 //
	public OrderDetail(long orderNumber) {
		this.orderNumber = orderNumber;
	}

	//메서드
	public void addOrderMenu(Food food){ // 주문메뉴 추가
			if (food.getFoodName().equals("짜장면")) { //  짜장면
				foodCount[0] = foodCount[0] + 1;
			}
			else if (food.getFoodName().equals("짬뽕")) { //짬뽕
				foodCount[1] = foodCount[1] + 1;
			}
			else if ((food.getFoodName().equals("볶음짬뽕"))) { //볶음짬뽕
				foodCount[2] = foodCount[2] + 1;
			}
			else { //탕수육
				foodCount[3] = foodCount[3] + 1;
			}
	}
	void deleteOrderMenu(Food food){ // 주문메뉴 삭제
		if (food.getFoodName().equals("짜장면")) { //  짜장면
			foodCount[0] = foodCount[0] - 1;
		}
		else if (food.getFoodName().equals("짬뽕")) { //짬뽕
			foodCount[1] = foodCount[1] - 1;
```

```
                }
                else if ((food.getFoodName().equals("볶음짬뽕"))) {
                        foodCount[2] = foodCount[2] - 1;
                }
                else {
                        foodCount[3] = foodCount[3] - 1;
                }
        }

        // getter & setter
        public int calculatingTotalPrice() { // 총주문금액 반환
                int totalPrice = (jja.getPrice() * foodCount[0]) + (jjam.getPrice() * foodCount[1]) +
(bok.getPrice() * foodCount[2]) + (tang.getPrice() * foodCount[3]);
                totalOrderPrice = totalPrice;
                return totalOrderPrice;
        }
        public int getFoodCount(Food food) {
                if (food.getFoodName().equals("짜장면")) { //  짜장면
                        return foodCount[0];
                }
                else if (food.getFoodName().equals("짬뽕")) { //짬뽕
                        return foodCount[1];
                }
                else if ((food.getFoodName().equals("볶음짬뽕"))) {
                        return foodCount[2];
                }
                else {
                        return foodCount[3];
                }
        }
        public String getFoodName(Food food) {
                return food.getFoodName();
        }
        public int getFoodPrice(Food food) {
                return food.getPrice();
        }

}
```

클래스 이름	Table	클래스 설명	테이블

```
import javax.swing.table.DefaultTableModel;

public class Table {
	private int tableNumber;
	private int tableState;
	private long occupyTime;

	private Order _order;

	//생성자
	public Table(int tableNumber, int tableState) {
		this.tableNumber = tableNumber;
		this.tableState = 0;
	}

//메서드
public void addOrderMenu(Food food) { //주문메뉴 추가
		if (tableState == 0) {
			changeTableState();
			_order.addOrderMenu(food);
		}
		else {
			_order.addOrderMenu(food);
		}
	}
public void deleteOrderMenu(Food food) { //주문메뉴 삭제
		_order.deleteOrderMenu(food);
}
public DefaultTableModel accessOrder() { //주문내역을 UI 로 return 함
	return _order.accessOrder();
}
public void makeOrder () { //주문 생성 : 주문번호를 가진 주문이 생성됨
	_order = new Order();
}

//getter & setter
public int getTableNumber() {
	return this.tableNumber;
}

public void setTableNumber(int tableNumber) {
```

```
		this.tableNumber = tableNumber;
}

public String getTotalPrice() {
		if (tableState == 0) {
			return "";
		}
		return Integer.toString(_order.getTotalPrice());

}
public int getOccupyTime() {
		if (tableState == 0) {
			return 0;
		}
		else {
			long nowTime = System.currentTimeMillis ();
			long firstOrderTime = _order.getfirstOrderTime();
			long occupyTimeSecond = (nowTime-firstOrderTime)/1000;
			int intOccupyTime = (int)occupyTimeSecond;
			return intOccupyTime;
		}}
public void setOccupyTime(long OccupyTime) {
		this.occupyTime = OccupyTime;
}
public void changeTableState() {
		if(tableState == 0){
			this.tableState = 1;
		}
		else {
			this.tableState = 0;
		}
}
public int getTableState() {
		return tableState;
}
Public Order getOrder() {
		Return _order;
}
Public void setOrderNull() {
		 This._order = null;
}
```

클래스 이름	Payment	클래스 설명	결제

```
import java.util.ArrayList;
import javax.swing.table.DefaultTableModel;

public class Payment {
	protected static long receiptNumber;
	protected static boolean paymentState;
	protected static int paymentDate ;
	protected static int paymentPrice;
	protected static int paymentType;
	protected		ArrayList<Payment> paymentList = new ArrayList<Payment>();


	private String[] columnNames = {"receiptNumber", "paymentState", "paymentDate","paymentPrice",
"paymentType"};
	private String[][] rowDatas = {};
	protected  DefaultTableModel paymentTable = new DefaultTableModel(rowDatas, columnNames);

	/* receiptNumber long 영수증 번호를 저장하는 변수.
	 * paymentState  boolean 현재 결제 상태를 알려주는 변수, true : 결제완료, false : 결제안됨
	 * paymentDate    int 결제일자를 저장하는 변수  YYMMDD 의 형식으로 저장된다.
	 * paymentPrice  int 결제 금액을 알려주는 변수이다. 주문 총 금액과 같다.
	 * paymentType    int 결제의 방식을 알려주는 변수,  0 : 현금결제, 1 : 카드결제
	 * paymentList    ArrayList<Payment> 결제에 대한 정보를 저장하는 장소이다. */

	public Payment(Order order){
		this.receiptNumber = order.getOrderNumber();
		this.paymentState = false;
		this.paymentDate = order.getOrderDate();
		this.paymentPrice = order.getTotalPrice();
		this.paymentType = 0 ;


	}

	public Payment() {
```

```
}

//************* 함수추가 , 매개변수


public void cashPaymentSave(Order order){
	//현금 결제시 저장하는 함수
	// 영수증번호, 결제방식, 결제상태, 받은금액, 거스름돈
	//"receiptNumber", "paymentState", "paymentDate","paymentPrice", "paymentType"

	boolean trueState = true ;
	Payment paymentitem = new Payment(order);
	paymentitem.paymentState = true;
	paymentitem.paymentType = 0;

	paymentList.add(paymentitem);

	Object settingMaterial[] = new Object[5];
	settingMaterial[0] = order.getOrderNumber();
	settingMaterial[1] = true;
	settingMaterial[2] = order.getOrderDate();
	settingMaterial[3] = order.getTotalPrice();
	settingMaterial[4] = 0;
	paymentTable.addRow(settingMaterial);



/* parameter : Order (주문클래스)
 * return : 없음
 * 설명 : 주문결제시 결제정보를 저장한다. */


public int getPaymentType() {
	return paymentType;
}

public void setPaymentType(int paymentType) {
	this.paymentType = paymentType;
}

public long getReceiptNumber() {
	return receiptNumber;
```

```
	}

	public void setReceiptNumber(long receiptNumber) {
		this.receiptNumber = receiptNumber;
	}

	public boolean getPaymentState() {
		return paymentState;
	}

	public void setPaymentState(boolean paymentState) {
		this.paymentState = paymentState;
	}

	public int getPaymentDate() {
		return paymentDate;
	}

	public void setPaymentDate(int paymentDate) {
		this.paymentDate = paymentDate;
	}

	public int getPaymentPrice() {
		return paymentPrice;
	}

	public void setPaymentPrice(int paymentPrice) {
		this.paymentPrice = paymentPrice;
	}

	public DefaultTableModel paymentTableAccess(){

		return paymentTable;

	}

	public ArrayList getArrayList(){

		return paymentList;
	}
}
```

클래스 이름	CreditCardPayment	클래스 설명	신용카드 결제

```
import java.util.ArrayList;
import javax.swing.table.DefaultTableModel;

public class CreditCardPayment extends Payment  {
	private String creditCardCompanyName;
	private int creditCardNumber;
	private int installmentPurchase;//할부개월수
	private int validDate;//유효기간
	private int okNumber;//승인번호
	private boolean signature;

	/* creditCardCompanyName string  신용카드 회사의 이름을 저장한다.
	 * creditCardNumber        int       신용카드의 번호에 대한 정보를 저장한다.
	 * installmentPurchase    int       신용카드 결제시 할부개월수에 대한 정보를 저장한다.
	 * validDate                 int       신용카드의 유효기간에 대한 정보를 저장한다.
	 * okNumber                  int       신용카드 승인시 승인번호에 대한 정보를 저장한다.
	 * signature                 boolean 신용카드 결제시 고객의 서명을 받는데 그 서명에 대한 정보를
저장하는 것으로서,
	 *                                  true : 카드결제 성공,  false : 카드결제 실패  */

	public void CreditCardPayment (Order order){

		Payment paymentitem = new Payment(order);
		paymentitem.paymentState = true;
		paymentitem.paymentType = 1;

		paymentList.add(paymentitem);

		Object settingMaterial[] = new Object[5];
		settingMaterial[0] = order.getOrderNumber();
		settingMaterial[1] = true;
		settingMaterial[2] = order.getOrderDate();
		settingMaterial[3] = order.getTotalPrice();
		settingMaterial[4] = 1;
		paymentTable.addRow(settingMaterial);
	}
	// parameter : Order(주문 클래스 )
```

```
// return : 없음
// 설명 : 카드결제시 결제정보를 저장한다.
}
```