[제출]카카오뱅크 사전과제

1. API 설계서

1.1. API 목록

No.	API명	API설명
1	이체내역조회	카카오뱅크 특정 계좌의 이체내역을 조회한다.
2	이체내역상세조회	카카오뱅크 특정 계좌의 특정거래 상세내역을 조회한다.
3	계좌이체_송금	카카오뱅크계좌 혹은 카카오톡친구로 이체한다.
4	계좌이체_입금	카카오뱅크계좌 혹은 타행내계좌로 이체한다.
5	계좌이체_취소	입금이 완료되지 않은 거래에 대해 취소 처리한다.
6	이체취소(배치)	24시간이 지나도록 입금이 완료되지 않은 거래에 대해서 대량으로 취소 처리한다.
7	알림전송(배치)	알림메시지큐에 적재된 건들에 대해서 실제 전송하고 알림전송내역에 적재한다.

1.2. API In/Out

- **이체내역조회**

IN/OUT	Type	변수명		필수여부(기본값)	변수설명(도메인)
Input	String	계좌번호		O	수신계좌번호
	String	조회기간코드		O(3개월)	조회기간코드(직접설정,지난달,3개월,1개월)
	LocalDate	조회기준시작일자		O(Today-90days)	조회기준시작일자
	LocalDate	조회기준종료일자		O(Today)	조회기준종료일자
	String	입금출금구분코드		O(전체)	입금출금구분코드(전체,입금,출금)
	String	정렬구분코드		O(최신순)	정렬구분코드(최신순,과거순)
	String	거래종류코드		O(전체)	거래종류코드(전체,예금이자,자동이체,저금통)
Output	String	고객번호			(본인)고객번호 -> 추후 내역상세조회 Input으로 사용
	String	고객명			(본인)고객명
	String	List<OutputGrid>	계좌번호		(본인)계좌번호 -> 추후 내역상세조회 Input으로 사용
	String		거래일련번호		(본인)거래일련번호 -> 추후 내역상세조회 Input으로 사용
	LocalDate		거래일자		거래일자
	String		거래종류코드		거래종류코드(예금이자,자동이체,저금통)
	String		상대고객명		상대고객명
	String		입금출금구분코드		입금출금구분코드(입금,출금) -> 해당값에 따라 금액의 폰트 변경
	String		거래통화코드		거래통화코드(KRW,USD)
	BigDecimal		거래금액		거래금액
	BigDecimal		거래통화금액		거래통화금액
	BigDecimal		거래후잔액		거래후잔액

- **이체내역상세조회**

IN/OUT	Type	변수명	필수여부(기본값)	변수설명(도메인)
Input	String	계좌번호	O	계좌번호
	Long	거래일련번호	O	거래일련번호
	String	고객번호	O	고객번호
Output	String	고객번호		(본인)고객번호
	String	계좌번호		(본인)계좌번호
	String	거래비고		거래비고란 ex)밥값
	LocalDate	거래일자		거래일자
	String	거래시각		거래시각 ex) 18:30:40
	String	거래종류코드		거래종류코드(예금이자,자동이체,저금통)
	String	입금출금구분코드		입금출금구분코드(입금,출금) -> 해당값에 따라 금액의 폰트 변경
	String	거래통화코드		거래통화코드(KRW,USD)
	BigDecimal	거래금액		거래금액
	BigDecimal	거래통화금액		거래통화금액
	BigDecimal	거래후잔액		거래후잔액
	String	상대고객명		상대고객명
	String	상대거래은행		상대거래은행
	String	상대계좌번호		상대계좌번호

- **계좌이체_송금**

IN/OUT	Type	변수명	필수여부(기본값)	변수설명(도메인)
Input	String	원거래계좌번호	O	원거래계좌번호
	String	관련거래ID	O	관련거래ID(계좌번호,카카오아이디)
	String	관련거래업무구분코드	O	관련거래업무구분코드(송금,입금,카톡친구,타행 등)
	BigDecimal	거래금액		거래금액
	BigDecimal	수수료금액		수수료금액
	String	거래비고		거래비고 ex)밥값
	String	관련거래고객명		관련거래고객명
	String	관련거래비고		관련거래비고 ex)돈을 보냈습니다.
Output	String	리턴코드		리턴코드
	String	관련거래고객명		관련거래고객명
	BigDecimal	거래금액		거래금액
	String	거래비고		거래비고 ex)밥값

- **계좌이체_입금**

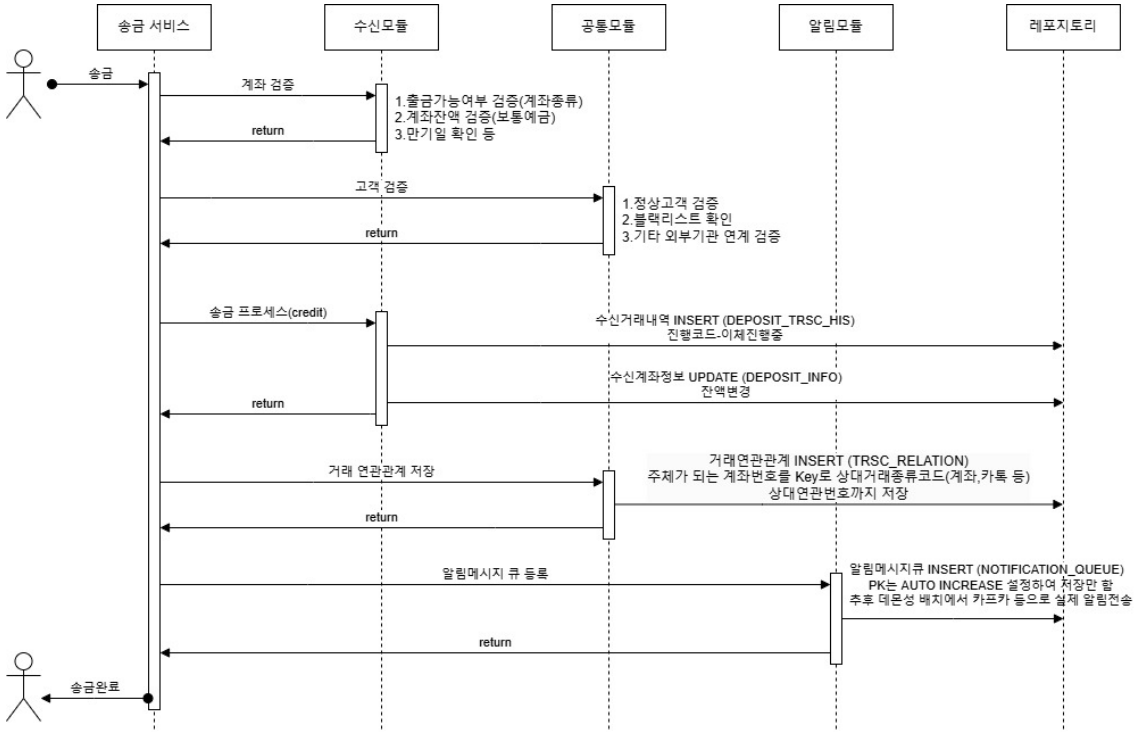
IN/OUT	Type	변수명	필수여부(기본값)	변수설명(도메인)
Input	String	원거래계좌번호	O	원거래계좌번호
	String	관련거래ID	O	관련거래ID(계좌번호,카카오아이디)
	String	관련거래업무구분코드	O	관련거래업무구분코드(송금,입금,카톡친구,타행 등)
	BigDecimal	거래금액		거래금액
	BigDecimal	수수료금액		수수료금액
	String	거래비고		거래비고 ex)누가 돈을 받았습니다.
	String	관련거래고객명		관련거래고객명
	String	관련거래비고		관련거래비고 ex)밥값
Output	String	리턴코드		리턴코드
	String	관련거래실명		관련거래고객명
	BigDecimal	거래금액		거래금액
	BigDecimal	거래후잔액		거래후잔액
	String	거래비고		거래비고 ex)누가 돈을 받았습니다.

- **계좌이체_취소**

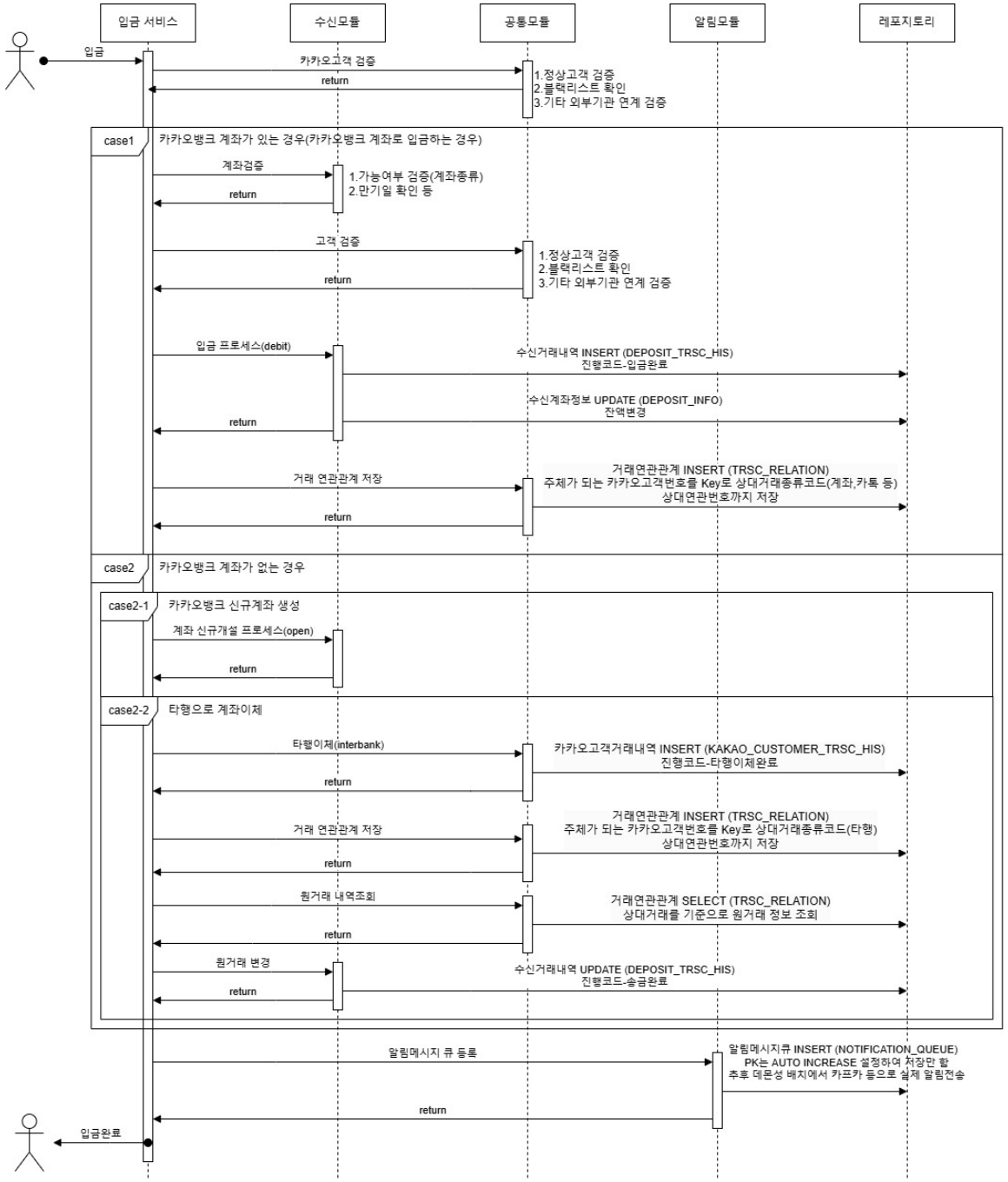
IN/OUT	Type	변수명	필수여부(기본값)	변수설명(도메인)
Input	String	원거래계좌번호	O	원거래계좌번호
	Long	원거래일련번호	O	원거래일련번호
	String	취소사유	O	취소사유 ex)24시간이 지날때까지 받지 않았습니다.
Output	String	리턴코드		리턴코드
	String	거래금액		거래금액
	BigDecimal	거래후잔액		거래후잔액
	String	취소사유		취소사유 ex)잘못 송금하였습니다.

2. API 별 시퀀스 다이어그램

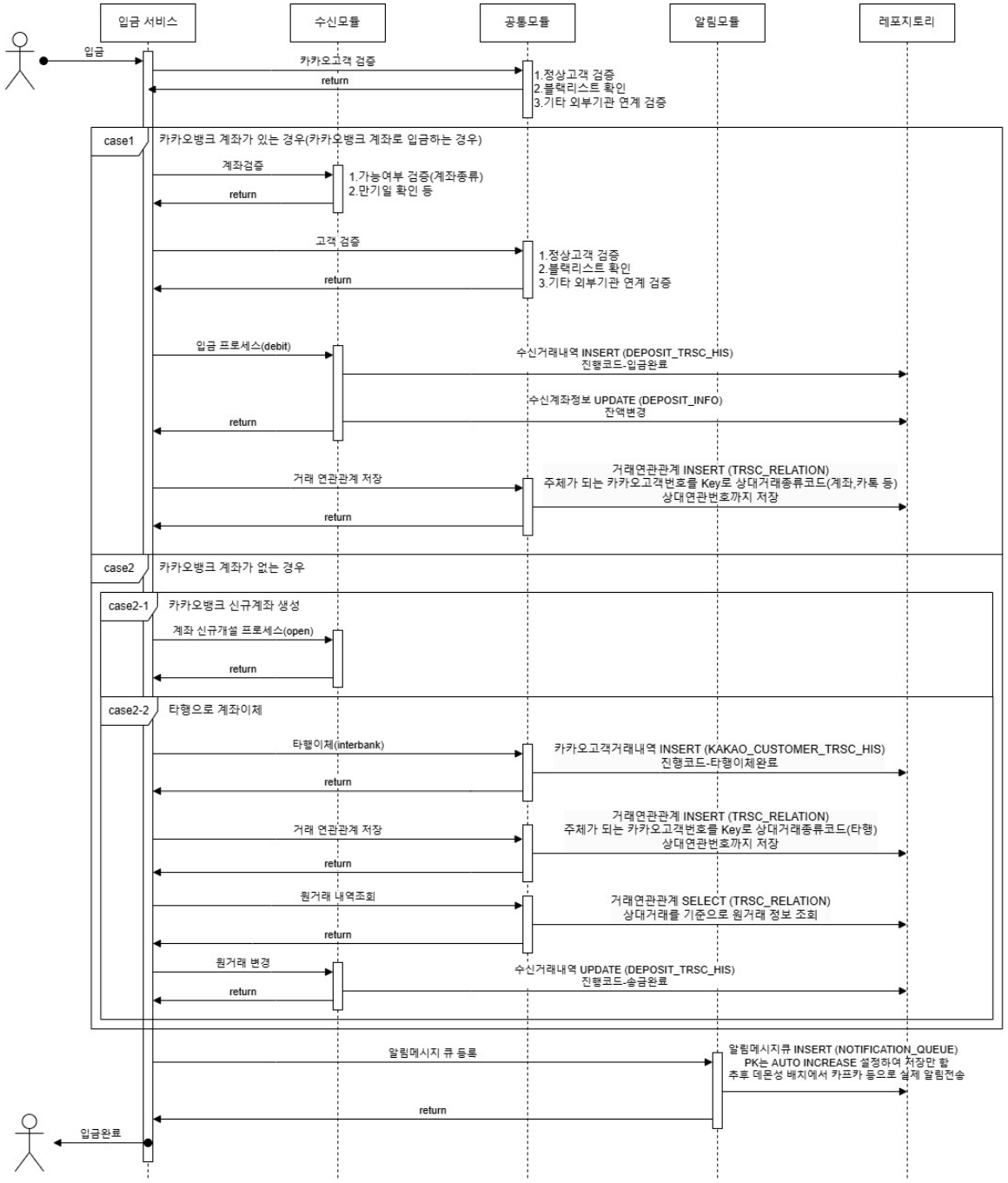
- **계좌이체_송금**


송금 서비스
수신모듈
공통모듈
알림모듈
레포지토리
송금
계좌 검증
1.출금가능여부 검증(계좌종류)
2.계좌잔액 검증(보통예금)
3.만기일 확인 등
return
고객 검증
1.정상고객 검증
2.블랙리스트 확인
3.기타 외부기관 연계 검증
return
송금 프로세스(credit)
수신거래내역 INSERT (DEPOSIT_TRSC_HIS)
진행코드-이체진행중
수신계좌정보 UPDATE (DEPOSIT_INFO)
잔액변경
return
거래 연관관계 저장
거래연관관계 INSERT (TRSC_RELATION)
주체가 되는 계좌번호를 Key로 상대거래종류코드(계좌,카톡 등)
상대연관번호까지 저장
return
알림메시지 큐 등록
알림메시지큐 INSERT (NOTIFICATION_QUEUE)
PK는 AUTO INCREASE 설정하여 저장만 함
추후 데몬성 배치에서 카프카 등으로 실제 알림전송
return
송금완료

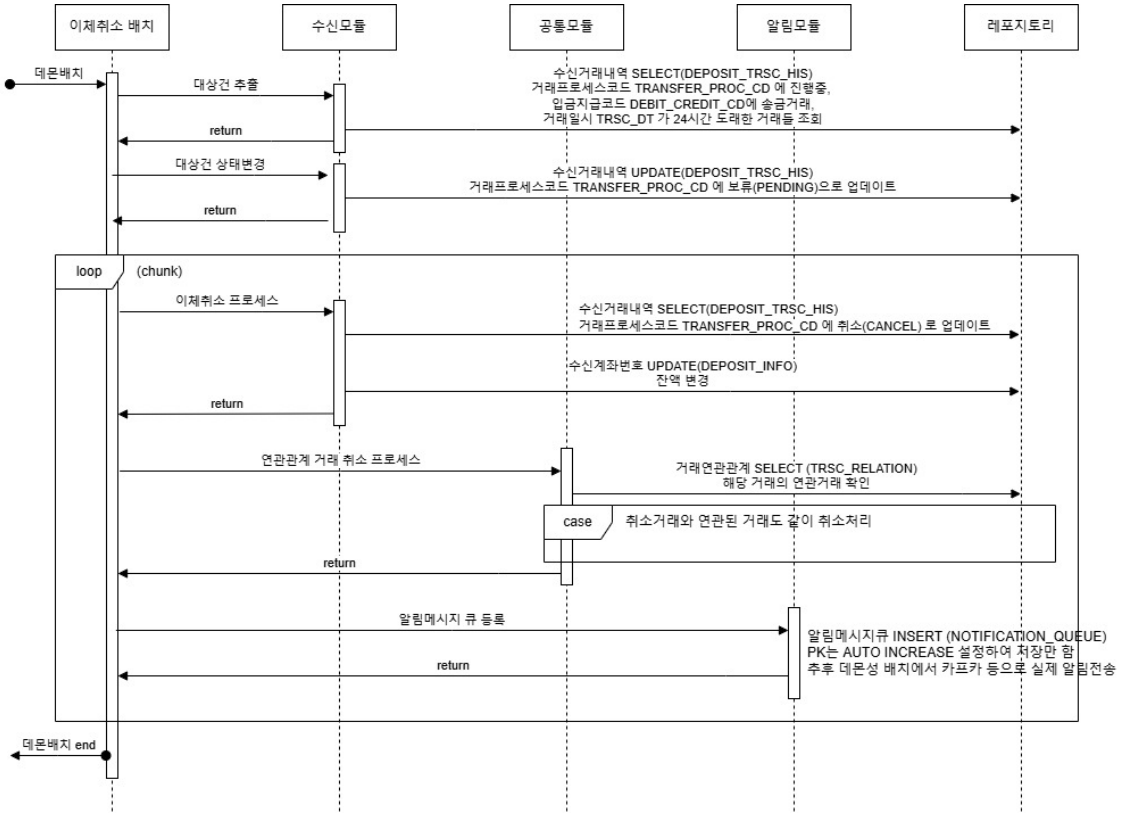
- **계좌이체_입금**


입금 서비스
수신모듈
공통모듈
알림모듈
레포지토리
입금
카카오고객 검증
return
1.정상고객 검증
2.블랙리스트 확인
3.기타 외부기관 연계 검증
case1
카카오뱅크 계좌가 있는 경우(카카오뱅크 계좌로 입금하는 경우)
계좌검증
return
1.가능여부 검증(계좌종류)
2.만기일 확인 등
고객 검증
return
1.정상고객 검증
2.블랙리스트 확인
3.기타 외부기관 연계 검증
입금 프로세스(debit)
수신거래내역 INSERT (DEPOSIT_TRSC_HIS)
진행코드-입금완료
수신계좌정보 UPDATE (DEPOSIT_INFO)
잔액변경
return
거래 연관관계 저장
거래연관관계 INSERT (TRSC_RELATION)
주체가 되는 카카오고객번호를 Key로 상대거래종류코드(계좌,카톡 등)
상대연관번호까지 저장
return
case2
카카오뱅크 계좌가 없는 경우
case2-1
카카오뱅크 신규계좌 생성
계좌 신규개설 프로세스(open)
return
case2-2
타행으로 계좌이체
타행이체(interbank)
카카오고객거래내역 INSERT (KAKAO_CUSTOMER_TRSC_HIS)
진행코드-타행이체완료
return
거래 연관관계 저장
거래연관관계 INSERT (TRSC_RELATION)
주체가 되는 카카오고객번호를 Key로 상대거래종류코드(타행)
상대연관번호까지 저장
return
원거래 내역조회
거래연관관계 SELECT (TRSC_RELATION)
상대거래를 기준으로 원거래 정보 조회
return
원거래 변경
수신거래내역 UPDATE (DEPOSIT_TRSC_HIS)
진행코드-송금완료
return
알림메시지 큐 등록
알림메시지큐 INSERT (NOTIFICATION_QUEUE)
PK는 AUTO INCREASE 설정하여 저장만 함
추후 데몬성 배치에서 카프카 등으로 실제 알림전송
return
입금완료

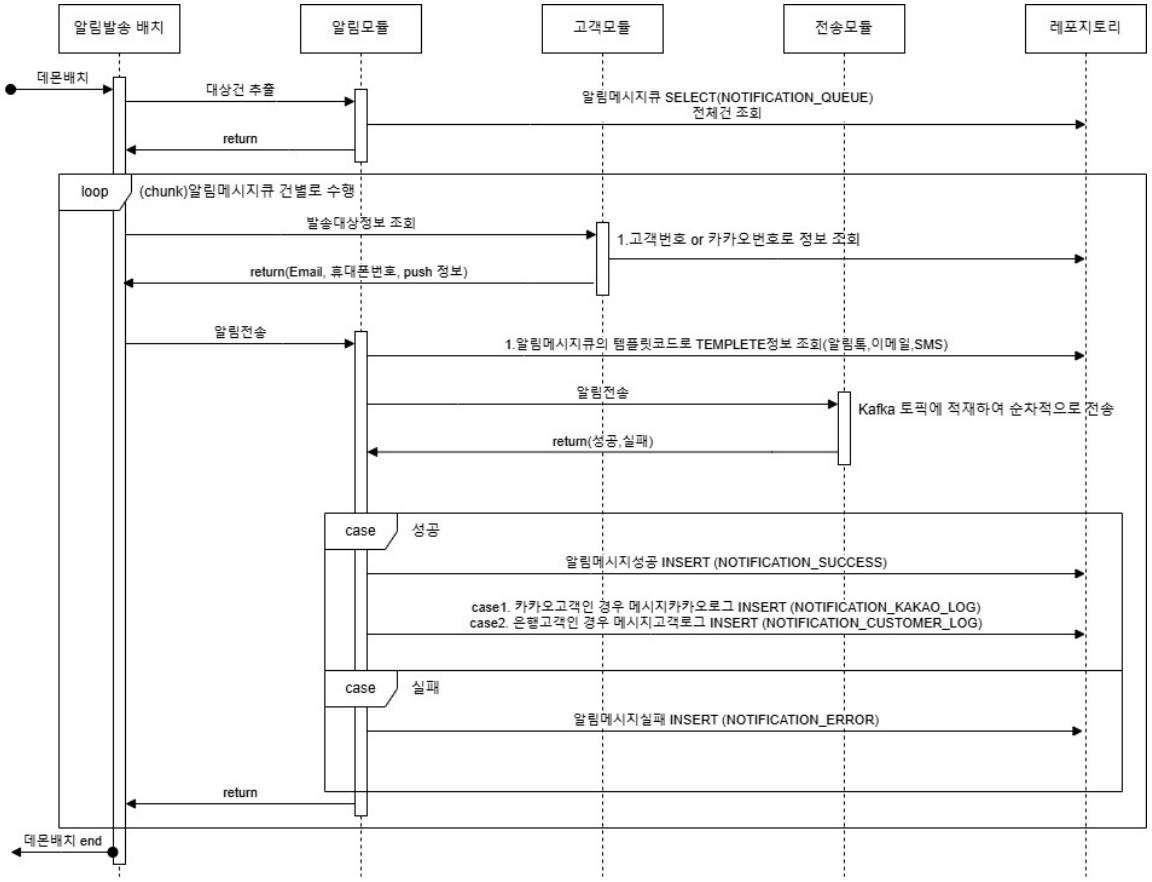
- **계좌이체_취소**


입금 서비스
수신모듈
공통모듈
알림모듈
레포지토리
입금
카카오고객 검증
return
1.정상고객 검증
2.블랙리스트 확인
3.기타 외부기관 연계 검증
case1
카카오뱅크 계좌가 있는 경우(카카오뱅크 계좌로 입금하는 경우)
계좌검증
return
1.가능여부 검증(계좌종류)
2.만기일 확인 등
고객 검증
return
1.정상고객 검증
2.블랙리스트 확인
3.기타 외부기관 연계 검증
입금 프로세스(debit)
수신거래내역 INSERT (DEPOSIT_TRSC_HIS)
진행코드-입금완료
수신계좌정보 UPDATE (DEPOSIT_INFO)
잔액변경
return
거래 연관관계 저장
거래연관관계 INSERT (TRSC_RELATION)
주체가 되는 카카오고객번호를 Key로 상대거래종류코드(계좌,카톡 등)
상대연관번호까지 저장
return
case2
카카오뱅크 계좌가 없는 경우
case2-1
카카오뱅크 신규계좌 생성
계좌 신규개설 프로세스(open)
return
case2-2
타행으로 계좌이체
타행이체(interbank)
카카오고객거래내역 INSERT (KAKAO_CUSTOMER_TRSC_HIS)
진행코드-타행이체완료
return
거래 연관관계 저장
거래연관관계 INSERT (TRSC_RELATION)
주체가 되는 카카오고객번호를 Key로 상대거래종류코드(타행)
상대연관번호까지 저장
return
원거래 내역조회
거래연관관계 SELECT (TRSC_RELATION)
상대거래를 기준으로 원거래 정보 조회
return
원거래 변경
수신거래내역 UPDATE (DEPOSIT_TRSC_HIS)
진행코드-송금완료
return
알림메시지 큐 등록
알림메시지큐 INSERT (NOTIFICATION_QUEUE)
PK는 AUTO INCREASE 설정하여 저장만 함
추후 데몬성 배치에서 카프카 등으로 실제 알림전송
return
입금완료

- **이체취소(배치)**

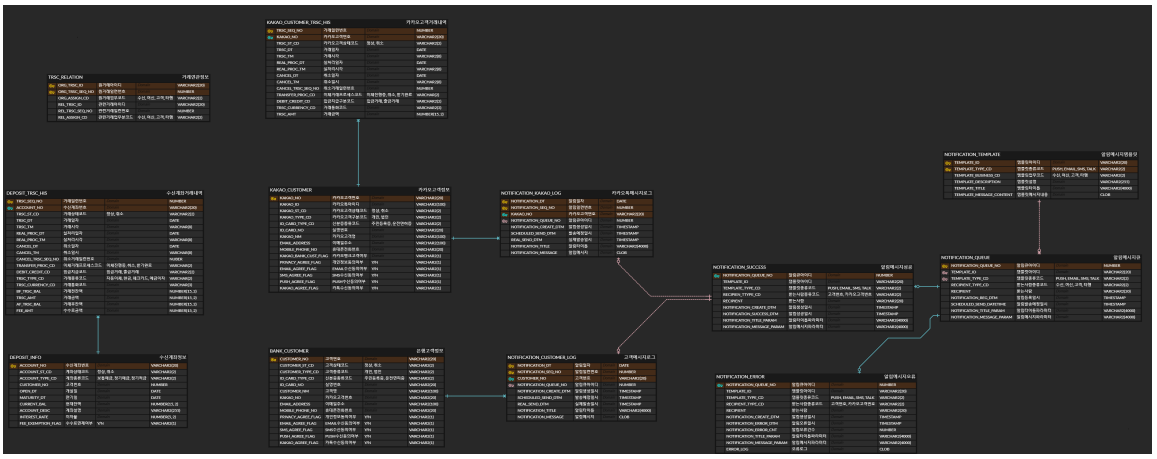

이체취소 배치
수신모듈
공통모듈
알림모듈
레포지토리
데몬배치
대상건 추출
수신거래내역 SELECT(DEPOSIT_TRSC_HIS)
거래프로세스코드 TRANSFER_PROC_CD 에 진행중,
입금지급코드 DEBIT_CREDIT_CD에 송금거래,
거래일시 TRSC_DT 가 24시간 도래한 거래들 조회
return
대상건 상태변경
수신거래내역 UPDATE(DEPOSIT_TRSC_HIS)
거래프로세스코드 TRANSFER_PROC_CD 에 보류(PENDING)으로 업데이트
return
loop
(chunk)
이체취소 프로세스
수신거래내역 SELECT(DEPOSIT_TRSC_HIS)
거래프로세스코드 TRANSFER_PROC_CD 에 취소(CANCEL) 로 업데이트
수신계좌번호 UPDATE(DEPOSIT_INFO)
잔액 변경
return
연관관계 거래 취소 프로세스
거래연관관계 SELECT (TRSC_RELATION)
해당 거래의 연관거래 확인
case
취소거래와 연관된 거래도 같이 취소처리
return
알림메시지 큐 등록
알림메시지큐 INSERT (NOTIFICATION_QUEUE)
PK는 AUTO INCREASE 설정하여 저장만 함
추후 데몬성 배치에서 카프카 등으로 실제 알림전송
return
데몬배치 end

- **알림전송(배치)**


알림발송 배치
알림모듈
고객모듈
전송모듈
레포지토리
데몬배치
대상건 추출
알림메시지큐 SELECT(NOTIFICATION_QUEUE)
전체건 조회
return
loop
(chunk)알림메시지큐 건별로 수행
발송대상정보 조회
1.고객번호 or 카카오번호로 정보 조회
return(Email, 휴대폰번호, push 정보)
알림전송
1.알림메시지큐의 템플릿코드로 TEMPLETE정보 조회(알림톡,이메일,SMS)
알림전송
Kafka 토픽에 적재하여 순차적으로 전송
return(성공,실패)
case
성공
알림메시지성공 INSERT (NOTIFICATION_SUCCESS)
case1. 카카오고객인 경우 메시지카카오로그 INSERT (NOTIFICATION_KAKAO_LOG)
case2. 은행고객인 경우 메시지고객로그 INSERT (NOTIFICATION_CUSTOMER_LOG)
case
실패
알림메시지실패 INSERT (NOTIFICATION_ERROR)
return
데몬배치 end

3. ERD&테이블 설명

- **ERD**



- **테이블 리스트**

수신계좌정보

DEPOSIT_INFO

Key	Column Name	Column Describe	Type	Nullable	Domain
PK	ACCOUNT_NO	수신계좌번호	VARCHAR2(20)	N	
	ACCOUNT_ST_CD	계좌상태코드	VARCHAR2(2)	N	정상, 취소
	ACCOUNT_TYPE_CD	계좌종류코드	VARCHAR2(2)	N	보통예금, 정기예금, 정기적금
	CUSTOMER_NO	고객번호	NUMBER	N	
	OPEN_DT	개설일	DATE		
	MATURITY_DT	만기일	DATE		
	CURRENT_BAL	현재잔액	NUMBER(15, 2)	N	
	ACCOUNT_DESC	계좌설명	VARCHAR2(255)		
	INTEREST_RATE	이자율	NUMBER(5, 2)		
	FEE_EXEMPTION_FLAG	수수료면제여부	VARCHAR2(1)		Y/N

수신거래내역

DEPOSIT_TRSC_HIS

Key	Column Name	Column Describe	Type	Nullable	Domain
PK, FK	ACCOUNT_NO	수신계좌번호	VARCHAR2(20)		
PK	TRSC_SEQ_NO	거래일련번호	NUMBER		
	TRSC_ST_CD	거래상태코드	VARCHAR2(2)		정상, 취소
	TRSC_DT	거래일자	DATE		
	TRSC_TM	거래시각	VARCHAR(8)		
	REAL_PROC_DT	실처리일자	DATE		
	REAL_PROC_TM	실처리시각	VARCHAR(8)		
	CANCEL_DT	취소일자	DATE		
	CANCEL_TM	취소일시	VARCHAR(8)		
	CANCEL_TRSC_SEQ_NO	취소거래일련번호	NUBER		
	TRANSFER_PROC_CD	이체거래프로세스코드	VARCHAR(2)		이체진행중, 취소, 받기완료
	DEBIT_CREDIT_CD	입금지급코드	VARCHAR2(2)		입금거래, 출금거래
	TRSC_TYPE_CD	거래종류코드	VARCHAR(2)		자동이체, 현금, 체크카드, 예금이자
	TRSC_CURRENCY_CD	거래통화코드	VARCHAR(3)		
	BF_TRSC_BAL	거래전잔액	NUMBER(15, 2)		
	TRSC_AMT	거래금액	NUMBER(15, 2)		
	AF_TRSC_BAL	거래후잔액	NUMBER(15, 2)		
	FEE_AMT	수수료금액	NUMBER(15, 2)		

거래관계

TRSC_RELATION

Key	Column Name	Column Describe	Type		Domain
PK	ORG_TRSC_ID	원거래아이디	VARCHAR2(20)		
PK	ORG_TRSC_SEQ_NO	원거래일련번호	NUMBER		
	ORG_ASSIGN_CD	원거래업무코드	VARCHAR2(2)		수신, 여신, 고객, 타행
	REL_TRSC_ID	관련거래아이디	VARCHAR2(20)		
	REL_TRSC_SEQ_NO	관련거래일련번호	NUMBER		
	REL_ASSIGN_CD	관련거래업무분코드	VARCHAR2(2)		수신, 여신, 고객, 타행

은행고객정보

BANK_CUSTOMER

Key	Column Name	Column Describe	Type		Domain
PK	CUSTOMER_NO	고객번호	VARCHAR2(20)		
	CUSTOMER_ST_CD	고객상태코드	VARCHAR2(2)		정상, 취소
	CUSTOMER_TYPE_CD	고객종류코드	VARCHAR2(2)		개인, 법인
	ID_CARD_TYPE_CD	신분증종류코드	VARCHAR2(2)		주민등록증, 운전면허증
	ID_CARD_NO	실명번호	VARCHAR2(20)		
	CUSTOMER_NM	고객명	VARCHAR2(100)		
	KAKAO_NO	카카오고객번호	VARCHAR2(20)		
	EMAIL_ADDRESS	이메일주소	VARCHAR2(100)		
	MOBILE_PHONE_NO	휴대폰전화번호	VARCHAR2(20)		
	PRIVACY_AGREE_FLAG	개인정보동의여부	VARCHAR2(1)		Y/N
	EMAIL_AGREE_FLAG	EMAIL수신동의여부	VARCHAR2(1)		Y/N
	SMS_AGREE_FLAG	SMS수신동의여부	VARCHAR2(1)		Y/N
	PUSH_AGREE_FLAG	PUSH수신동의여부	VARCHAR2(1)		Y/N
	KAKAO_AGREE_FLAG	카톡수신동의여부	VARCHAR2(1)		Y/N

카카오고객정보

KAKAO_CUSTOMER

Key	Column Name	Column Describe	Type		Domain
PK	KAKAO_NO	카카오고객번호	VARCHAR2(20)		
	KAKAO_ID	카카오톡아이디	VARCHAR2(100)		
	KAKAO_ST_CD	카카오고객상태코드	VARCHAR2(2)		정상, 취소
	KAKAO_TYPE_CD	카카오고객구분코드	VARCHAR2(2)		개인, 법인
	ID_CARD_TYPE_CD	신분증종류코드	VARCHAR2(2)		주민등록증, 운전면허증
	ID_CARD_NO	실명번호	VARCHAR2(20)		
	KAKAO_NM	카카오고객명	VARCHAR2(100)		
	EMAIL_ADDRESS	이메일주소	VARCHAR2(100)		
	MOBILE_PHONE_NO	휴대폰전화번호	VARCHAR2(20)		
	KAKAO_BANK_CUST_FLAG	카카오뱅크고객여부	VARCHAR2(1)		
	PRIVACY_AGREE_FLAG	개인정보동의여부	VARCHAR2(1)		Y/N
	EMAIL_AGREE_FLAG	EMAIL수신동의여부	VARCHAR2(1)		Y/N
	SMS_AGREE_FLAG	SMS수신동의여부	VARCHAR2(1)		Y/N
	PUSH_AGREE_FLAG	PUSH수신동의여부	VARCHAR2(1)		Y/N
	KAKAO_AGREE_FLAG	카톡수신동의여부	VARCHAR2(1)		Y/N

카카오고객거래내역

KAKAO_CUSTOMER_TRSC_HIS

Key	Column Name	Column Describe	Type		Domain
PK, FK	KAKAO_NO	카카오고객번호	VARCHAR2(20)		
PK	TRSC_SEQ_NO	거래일련번호	NUMBER		
	TRSC_ST_CD	카카오고객상태코드	VARCHAR2(2)		정상, 취소
	TRSC_DT	거래일자	DATE		
	TRSC_TM	거래시각	VARCHAR2(8)		
	REAL_PROC_DT	실처리일자	DATE		
	REAL_PROC_TM	실처리시각	VARCHAR2(8)		
	CANCEL_DT	취소일자	DATE		
	CANCEL_TM	취소일시	VARCHAR2(8)		
	CANCEL_TRSC_SEQ_NO	취소거래일련번호	NUMBER		
	TRANSFER_PROC_CD	이체거래프로세스코	VARCHAR(2)		이체진행중, 취소, 받기완료
	DEBIT_CREDIT_CD	입금지급구분코드	VARCHAR2(2)		입금거래, 출금거래
	TRSC_CURRENCY_CD	거래통화코드	VARCHAR2(3)		
	TRSC_AMT	거래금액	NUMBER(15, 2)		

알림메시지템플릿

NOTIFICATION_TEMPLATE

Key	Column Name	Column Describe	Type		Domain
PK	TEMPLATE_ID	템플릿아이디	VARCHAR2(20)		
PK	TEMPLATE_TYPE_CD	템플릿종류코드	VARCHAR2(2)		PUSH, EMAIL, SMS, TALK
	TEMPLATE_BUSINESS_CD	템플릿업무코드	VARCHAR2(2)		수신, 여신, 고객, 타행
	TEMPLATE_DESCRIPTION	템플릿설명	VARCHAR2(255)		
	TEMPLATE_TITLE	템플릿타이틀	VARCHAR2(4000)		
	TEMPLATE_MESSAGE_CONTENT	템플릿메시지내용	CLOB		

알림메시지큐

NOTIFICATION_QUEUE

Key	Column Name	Column Describe	Type		Domain
PK	NOTIFICATION_QUEUE_NO	알림큐아이디	NUMBER		
FK	TEMPLATE_ID	템플릿아이디	VARCHAR2(20)		
FK	TEMPLATE_TYPE_CD	템플릿종류코드	VARCHAR2(2)		PUSH, EMAIL, SMS, TALK
	RECIPIENT_TYPE_CD	받는사람종류코드	VARCHAR2(2)		수신, 여신, 고객, 타행
	RECIPIENT	받는사람	VARCHAR2(20)		
	NOTIFICATION_REG_DTM	알림등록일시	TIMESTAMP		
	SCHEDULED_SEND_DATETIME	알림발송예정일시	TIMESTAMP		
	NOTIFICATION_TITLE_PARAM	알림타이틀파라미터	VARCHAR2(4000)		
	NOTIFICATION_MESSAGE_PARAM	알림메시지파라미터	VARCHAR2(4000)		

알림메시지성공

NOTIFICATION_SUCCESS

Key	Column Name	Column Describe	Type		Domain
PK, FK	NOTIFICATION_QUEUE_NO	알림큐아이디	NUMBER		
	TEMPLATE_ID	템플릿아이디	VARCHAR2(20)		
	TEMPLATE_TYPE_CD	템플릿종류코드	VARCHAR2(2)		PUSH, EMAIL, SMS, TALK
	RECIPIEN_TTYPE_CD	받는사람종류코드	VARCHAR2(2)		고객번호, 카카오고객번호
	RECIPIENT	받는사람	VARCHAR2(20)		
	NOTIFICATION_CREATE_DTM	알림생성일시	TIMESTAMP		
	NOTIFICATION_SUCCESS_DTM	알림성공일시	TIMESTAMP		
	NOTIFICATION_TITLE_PARAM	알림타이틀파라미터	VARCHAR2(4000)		
	NOTIFICATION_MESSAGE_PARAM	알림메시지파라미터	VARCHAR2(4000)		

알림메시지오류

NOTIFICATION_ERROR

Key	Column Name	Column Describe	Type		Domain
PK, FK	NOTIFICATION_QUEUE_NO	알림큐아이디	NUMBER		
	TEMPLATE_ID	템플릿아이디	VARCHAR2(20)		
	TEMPLATE_TYPE_CD	템플릿종류코드	VARCHAR2(2)		PUSH, EMAIL, SMS, TALK
	RECIPIENT_TYPE_CD	받는사람종류코드	VARCHAR2(2)		고객번호, 카카오고객번호
	RECIPIENT	받는사람	VARCHAR2(20)		
	NOTIFICATION_CREATE_DTM	알림생성일시	TIMESTAMP		
	NOTIFICATION_ERROR_DTM	알림오류일시	TIMESTAMP		
	NOTIFICATION_ERROR_CNT	알림오류건수	NUMBER		
	NOTIFICATION_TITLE_PARAM	알림타이틀파라미터	VARCHAR2(4000)		
	NOTIFICATION_MESSAGE_PARAM	알림메시지파라미터	VARCHAR2(4000)		
	ERROR_LOG	오류로그	CLOB		

카카오톡메시지로그

NOTIFICATION_KAKAO_LOG

Key	Column Name	Column Describe	Type		Domain
PK, FK	KAKAO_NO	카카오고객번호	VARCHAR2(20)		
PK	NOTIFICATION_DT	알림일자	DATE		
PK	NOTIFICATION_SEQ_NO	알림일련번호	NUMBER		
FK	NOTIFICATION_QUEUE_NO	알림큐아이디	NUMBER		
	NOTIFICATION_CREATE_DTM	알림생성일시	TIMESTAMP		
	SCHEDULED_SEND_DTM	발송예정일시	TIMESTAMP		
	REAL_SEND_DTM	실제발송일시	TIMESTAMP		
	NOTIFICATION_TITLE	알림타이틀	VARCHAR2(4000)		
	NOTIFICATION_MESSAGE	알림메시지	CLOB		

고객메시지로그

NOTIFICATION_CUSTOMER_LOG

Key	Column Name	Column Describe	Type		Domain
PK, FK	CUSTOMER_NO	고객번호	VARCHAR2(20)		
PK	NOTIFICATION_DT	알림일자	DATE		
PK	NOTIFICATION_SEQ_NO	알림일련번호	NUMBER		
FK	NOTIFICATION_QUEUE_NO	알림큐아이디	NUMBER		
	NOTIFICATION_CREATE_DTM	알림생성일시	TIMESTAMP		
	SCHEDULED_SEND_DTM	발송예정일시	TIMESTAMP		
	REAL_SEND_DTM	실제발송일시	TIMESTAMP		
	NOTIFICATION_TITLE	알림타이틀	VARCHAR2(4000)		
	NOTIFICATION_MESSAGE	알림메시지	CLOB		

- **고려사항**

1. 알림메시지 전송의 경우 대량의 트래픽이 발생할 가능성이 크기 때문에 Queue 형식의 테이블 구조 설계, Queue_Id 는 Auto Increase 로 설정하여 온라인이 아닌 백그라운드 배치에서 데몬성으로 동작하며 실제 카프카 등으로 전송
2. 수신거래내역 테이블의 경우도 대량의 트래픽이 발생할 수 있어 연동거래정보에 대해서 테이블 분리하여 트래픽 분산
3. 거래연관정보에 연관관계를 찾을 경우 속도 향상을 위하여 관련거래ID, 관련거래번호 에 인덱스를 추가
4. 카프카 등으로 실제 전송을 했을 때에는 Queue 테이블은 삭제 후 Success 혹은 Error 테이블로 이관하여 알림전송 배치의 성능 저하를 최소로 함

4. 위의 ERD를 기반으로, 아래의 데이터를 추출할 수 있는 Query

	이체 완료건수	이체 오류건수	처리중 건수
2021.01			
2021.02			
2021.03			
합계			

- **SQL쿼리**

```
SELECT 비고,이체완료건수,이체오류건수,처리중건수
  FROM
    (
    SELECT TO_CHAR(TRSC_DT, 'YYYY.MM') AS 비고
         , SUM(CASE WHEN TRSC_ST_CD = '00' THEN 1 ELSE 0 END)
         , SUM(CASE WHEN TRSC_ST_CD = '09' THEN 1 ELSE 0 END)
         , SUM(CASE WHEN TRSC_ST_CD = '01' THEN 1 ELSE 0 END)
         , ROW_NUMBER() OVER (ORDER BY TO_CHAR(TRSC_DT, 'YYYY
      FROM DEPOSIT_TRSC_HIS
     GROUP BY TO_CHAR(TRSC_DT, 'YYYY.MM')
```

```
     UNION ALL
    SELECT '합계' AS 비고
         , SUM(CASE WHEN TRSC_ST_CD = '00' THEN 1 ELSE 0 END)
         , SUM(CASE WHEN TRSC_ST_CD = '09' THEN 1 ELSE 0 END)
         , SUM(CASE WHEN TRSC_ST_CD = '01' THEN 1 ELSE 0 END)
         , 999 AS RN
      FROM DEPOSIT_TRSC_HIS
    )
ORDER BY RN
```

- **결과**

	비고	이체완료건수	이체오류건수	처리중건수
1	2024.09	15	7	0
2	2024.10	10	4	1
3	2024.11	5	3	2
4	합계	30	14	3

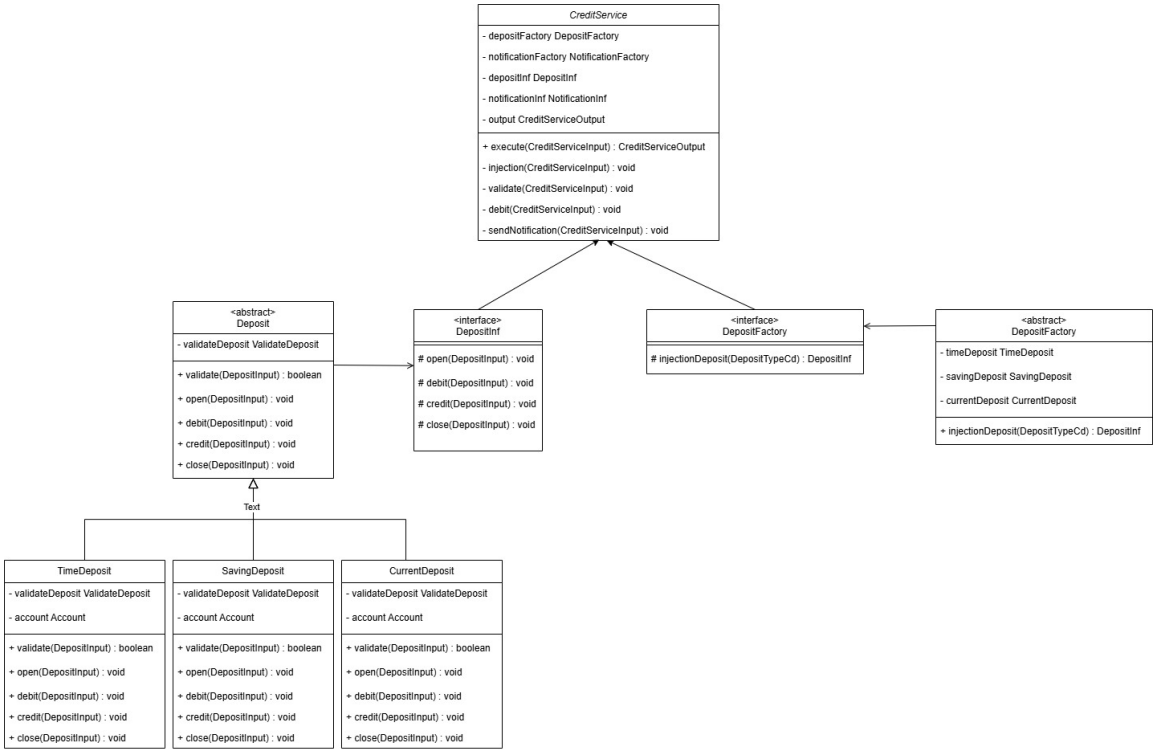
5. Java & RDBMS로 구현한 소스코드

- **프로젝트 구조**

```
kakao-online                    //root 프로젝트
├─api                           //api 업무성을 집합체
│  ├─common                     //공통업무
│  │  ├─com.kakao.common        //
│  │  │  ├─notification         //
│  │  │  │  ├─iomodel          //서비스 및 모듈 input, output 패키지
│  │  │  │  ├─service           //서비스 패키지
│  │  │  │  ├─model             //팩토리를 통해 주입받을 Provider들 집합체
│  │  │  │  └─module            //모듈 패키지
│  │  │  └─customer             //위와 동일
│  │  └─build.gradle            //domain을 의존성 주입한다.
│  ├─deposit                    //수신업무
│  │  ├─com.kakao.deposit       //
│  │  │  └─transfer             //위와 동일
│  │  └─build.gradle            //domain을 의존성 주입한다.
│  └─domain                     //각 업무별 인터페이스들 제공해줌
│     ├─com.kakao.domain        //
│     │  ├─inf                  //
│     │  │  ├─common            //업무별 interface 제공
│     │  │  ├─deposit           //
│     │  │  └─factory           //팩토리매서드를 이용하여 자식클래스 주입
│     │  └─jpa                  //
│     │     ├─repository        //커스터마이징한 IBaseRepository를 상속받은 Dao 집합체 (업무별로 나눈다)
│     │     └─entity            //테이블 Entity 집합체 (업무별로 나눈다)
│     └─build.gradle            //
├─onlinecore                    //config, filter, Application 등 프레임워크 파트
│  ├─com.kakao.core             //
│  │  ├─repository              //JpaRepository 커스터마이징을 통해 insert, update 함수 및 기타기능 추가
│  │  ├─exception               //오류메시지 통일화를 위해 구조 설정
│  │  ├─intercepter             //controller 제어(AOP 등으로)
│  │  ├─filter                  //jwt 인증, 인가, io모델 최초설정 등
│  │  ├─config                  //config 모음
│  │  └─Application             //SpringBoot 서버
│  └─resources                  //application.yml 설정파일들 관리 -local, -release 등
├─build.gradle                  //root, subproject 공통 dependency 추가
├─gradle.properties             //버전정보 등은 변수화 하여 관리하기 쉽게 셋팅
└─settings.gradle               //subModule 설정
```

업무별로 서브프로젝트로 나누어 모듈화 및 유지보수를 용이하게 함

- **클래스다이어그램**



- **고려사항**

  1. JpaRepository 커스터마이징하여 insert, update를 제공하여 명시적으로 사용할 수 있게 함

1. 엄무별 Dao의 상속을 BaseRepository 로 받도록 설정

```java
public interface DaoBankCustomer extends BaseRepository<BankCustomer, String> {

}
```

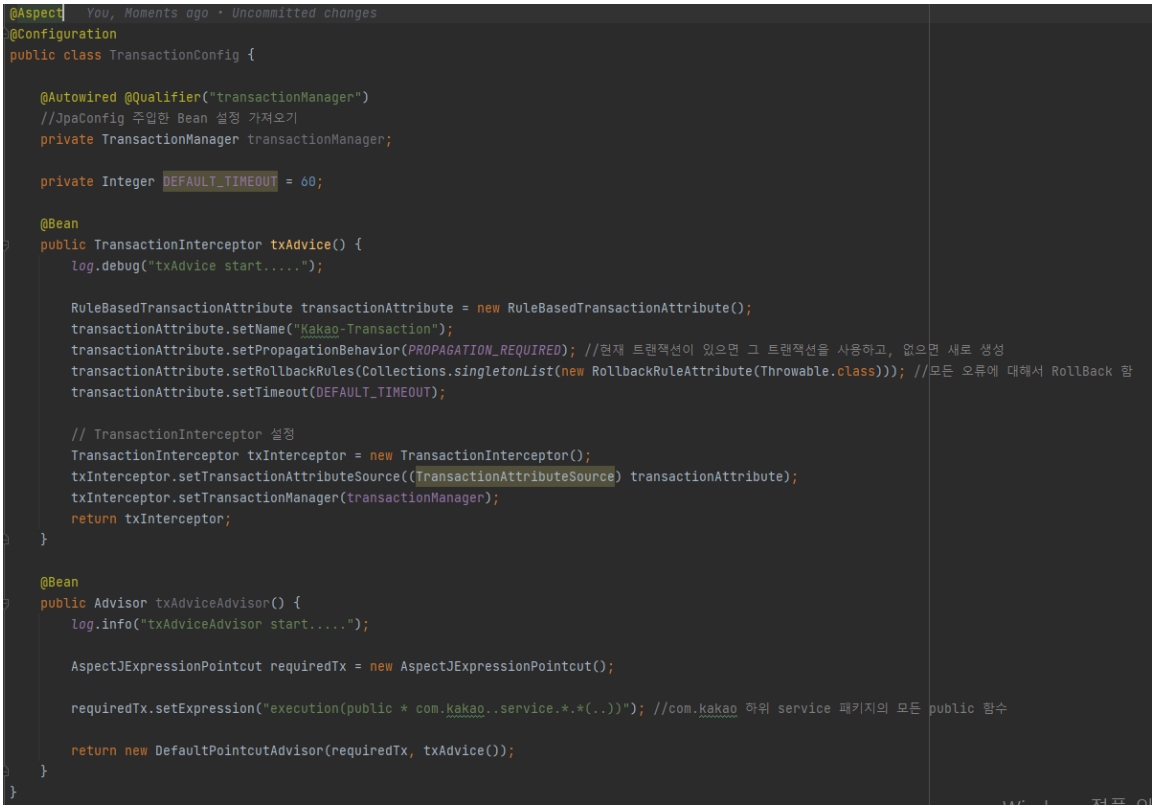
2. BaseRepository interface : JpaRepository를 상속받아 선언

```java
@NoRepositoryBean
public interface BaseRepository<T, Id> extends JpaRepository<T, Id> {

    <S extends T> S insert(S entity) throws Exception;

    <S extends T> S update(S entity) throws Exception;
}
```

3. BaseRepositoryImpl 실제 구현체 : BaseRepository를 implement 받아 실제 insert와 update를 구현함
SimpleJpaRepository를 상속받아 그 외의 모든 기능도 사용할 수 있게끔 구현
이유 : 엄무에서 insert, update를 명시함으로서 해당 기능을 손쉽게 알수 있게 함

```java
@Override
public <S extends T> S insert(S entity) throws Exception {
    try {
        //persist로 중복으로 insert하는 경우 오류처리
        entityManager.persist(entity);
    } catch (Exception e) {
        throw new Exception();
    }
    return entity;
}


@Override
public <S extends T> S update(S entity) throws Exception {

    //업데이트 전에 pk조회 후 값이 존재하지 않으면 오류
    Object pk = entityManager.getEntityManagerFactory().getPersistenceUnitUtil().getIdentifier(entity);
    T exists = entityManager.find(this.getDomainClass(), pk);

    if (exists == null) {
        throw new Exception();
    }

    entity = entityManager.merge(entity);
    return entity;
}
```

2. 트랜젝션 관리를 service의 public 함수를 호출할 경우 원트랜젝션으로 관리하도록 함



```java
@Aspect   You, Moments ago • Uncommitted changes
@Configuration
public class TransactionConfig {

    @Autowired @Qualifier("transactionManager")
    //JpaConfig 주입한 Bean 설정 가져오기
    private TransactionManager transactionManager;

    private Integer DEFAULT_TIMEOUT = 60;

    @Bean
    public TransactionInterceptor txAdvice() {
        log.debug("txAdvice start.....");

        RuleBasedTransactionAttribute transactionAttribute = new RuleBasedTransactionAttribute();
        transactionAttribute.setName("Kakao-Transaction");
        transactionAttribute.setPropagationBehavior(PROPAGATION_REQUIRED); //현재 트랜잭션이 있으면 그 트랜잭션을 사용하고, 없으면 새로 생성
        transactionAttribute.setRollbackRules(Collections.singletonList(new RollbackRuleAttribute(Throwable.class))); //모든 오류에 대해서 RollBack 함
        transactionAttribute.setTimeout(DEFAULT_TIMEOUT);

        // TransactionInterceptor 설정
        TransactionInterceptor txInterceptor = new TransactionInterceptor();
        txInterceptor.setTransactionAttributeSource((TransactionAttributeSource) transactionAttribute);
        txInterceptor.setTransactionManager(transactionManager);
        return txInterceptor;
    }

    @Bean
    public Advisor txAdviceAdvisor() {
        log.info("txAdviceAdvisor start.....");

        AspectJExpressionPointcut requiredTx = new AspectJExpressionPointcut();

        requiredTx.setExpression("execution(public * com.kakao..service.*.*(..))"); //com.kakao 하위 service 패키지의 모든 public 함수

        return new DefaultPointcutAdvisor(requiredTx, txAdvice());
    }
}
```

3. 팩토리메서드 패턴을 사용하여 의존성을 낮추며 핵심로직을 캡슐화 하고 유지보수에 용이하게 함 **(자세한 내용은 프로젝트 참고)**

1.Service 코드는 모두 동일하게 execute 만 public으로 생성

```java
@Service
@RequiredArgsConstructor
public class DebitService {

    //의존성주입 매서드
    private final CommonFactory commonFactory;
    private final DepositFactory depositFactory;

    //Dao Repository
    private final DaoBankCustomer daoBankCustomer;

    //해당 클래스에서 사용할 전역변수
    private DebitServiceInput input;
    private DebitServiceOutput output = new DebitServiceOutput();

    //구현체 호출 인터페이스
    private NotificationInf notificationInf;
    private DepositInf depositInf;
    private CustomerInf customerInf;

    public DebitServiceOutput execute(DebitServiceInput input) throws Exception{
        this.input = input;

        //1.injection(의존성주입)
        injection();

        //2.validate
        validation();

        //3.debit
        debit();

        //4.noti
        notification();

        //5.성공 반환
        outputSetting();

        return output;
    }
```

2. 최초에 의존성주입을 한 후 검증로직을 거친 후

```
private void injection() throws Exception {
    //수신은 보통예금 계좌
    this.depositInf = depositFactory.injectionDepositInf("saving");
    //알림은 카카오톡으로
    this.notificationInf = commonFactory.injectionNotificationInf("kakao");
    //고객은 카카오고객
    this.customerInf = commonFactory.injectionCustomerInf("kakao");
}

private void validation() throws Exception {
    /**
     * 계좌 종류에 따라서 검증함수 호출
     * if Saving(보통예금) 계좌인 경우
     * 1.정상계좌 여부 체크
     * 2.잔액, 만기 체크
     */
    DepositInput depositInput = new DepositInput();
    depositInput.setAcctNo(this.input.getAcctNo());
    this.depositInf.validate(depositInput);

    /**
     * 고객 종류에 따라서 검증함수 호출
     * 1.정상고객 여부 체크
     * 2.블랙리스트 체크(신용3사 제휴 검증 등)
     */
    CustomerInput customerInput = new CustomerInput();
    customerInput.setCustNo(this.input.getCustNo());
    this.customerInf.validate(customerInput);
}
```

3. main process 진행 후 리턴

```
private void debit() throws Exception {
    //출금거래 실행
    DepositInput depositInput = new DepositInput();
    depositInput.setAcctNo(this.input.getAcctNo());
    this.depositInf.debit(depositInput);
}

private void notification() throws Exception {
    //알림전송
    NotificationInput notificationInput = new NotificationInput();
    notificationInput.setCustNo(this.input.getCustNo());
    this.notificationInf.sendNotification(notificationInput);
}

private void outputSetting() throws Exception {
    //아웃풋 셋팅
    this.output.setReturnCode = "Success";
    //---   You, Moments ago • Uncommitted changes
}
```

- **추후 개선사항**
  1. Exception을 상속받아 공통적인 오류메시지 양식 생성, @ExceptionHandler를 사용하여 오류에 직접 핸들링 하도록 함
  2. ControllerInterceptor를 사용하여 서비스로 진입하기 전과 후에 양식을 통일하도록 함
  3. 배치 서버를 따로 두어 데몬, 스케쥴에 대해 Quartz를 사용하여 DB에서 핸들링 하도록 함