

입금 처리 자동화

은행 입금 결제 주문 입금 처리 프로세스 자동화

입금 처리 프로세스

BEFORE

1. 은행 입금 결제 주문
2. 고객이 은행에 입금
3. 은행 입금 내역 db에 수집
4. 관리자가 수집한 입금 내역과 주문 금액을 확인
5. 관리자가 입금 내역을 기준으로 고객 주문 입금 처리(입금 상태 변경)

기존에는 관리자가 총주문 금액과 입금한 금액이 일치하는지 직접 계산

입금액이 부족할 경우 포인트를 사용할 수 있는지 계산 후 수동으로 포인트 차감

입금액이 많은 경우에도 계산해서 수동으로 포인트로 지급

AFTER

자동화

입금한 금액이 총주문 금액(선택한 주문의 주문 금액 합)과 일치하는지, 포인트로 처리 가능한지 자동으로 체크

차감 또는 지급할 포인트 자동으로 계산해서 처리

입금 처리 = 입금 내역 처리 완료 상태로 변경 & 주문 상태 입금 완료로 변경

하나의 입금 내역으로 여러 주문을 입금 처리할 수 있다.

CASE1. 총주문 금액 == 입금액

입금 처리

CASE2. 총주문 금액 < 입금액

입금 처리 후 남은 금액 자동으로 계산해서 고객에게 포인트로 지급

CASE3. 입금액 < 총주문 금액 < 입금액 + 고객이 보유한 포인트

부족한 금액만큼 고객 포인트 차감 후 입금 처리

CASE4. 입금액 < 입금액 + 고객이 보유한 포인트 < 총주문 금액

입금 처리 불가

자동화로 인한 이점

입금 처리를 위해 사람이 직접 금액 계산을 하지 않아도 된다.

입금 처리에 드는 시간이 줄어든다.

포인트를 잘못 지급하거나 금액 계산을 잘못하는 등 기존의 수동 작업으로 발생할 수 있는 문제를 예방할 수 있다.

주요 코드

ProcType.cs

```
// 입금 처리 CASE 구분
public enum ProcType
{
    GENERALPROC, // CASE1
    RETURNPOINTPROC, // CASE2
    DEDUCTIONPOINTPROC, // CASE3
    CANNOTPROC // CASE4
}
```

DepositProcInfoGetter.cs

```
/*
DepositMatchingInfoGetter로 필요한 데이터 조회 (총주문 금액, 고객 보유 포인트, 입금액)
DepositProcInfoGetter에서 DepositMatchingInfoGetter로 기반으로 지급 또는 차감할 포인트 계산, 입금 처리 CASE 구분(어떤 procType인지)
*/
```

```

public DepositProcInfo Get(int id, string orderIds)
{
    var matchingInfo = DepositMatchingInfoGetter.Get(id, orderIds); // 총주문 금액, 고객이 보유한 포인트, 입금액 가져오기
    return GetProcInfo(matchingInfo); // CASE분석 후, 지급할 포인트, 차감할 포인트 계산
}

private DepositProcInfo GetProcInfo(DepositMatchingInfo matchingInfo)
{
    var procType = GetProcType(matchingInfo); // CASE 분석
    var returnPoint = GetReturnPoint(procType, matchingInfo); // 반환할 포인트
    var deductionPoint = GetDeductionPoint(procType, matchingInfo); // 차감할 포인트
    return new DepositProcInfo() { DepositMatchingInfo = matchingInfo, ProcType = procType, ReturnPoint = returnPoint, DeductionPoint = deductionPoint };
}

```

DepositConfirmService.cs

```

/*
    입금 처리
    submitInfo : 적용 버튼 눌렀을 때 조회해 return 받은 DepositMatchingInfo
*/
[Transaction]
public void ConfirmDeposit(int id, string orderIds, string memo, DepositMatchingInfo submitInfo, string adminId)
{
    DepositConfirmOrders(id, orderIds, submitInfo, adminId);
    DepositDao.ConfirmDeposit(id, orderIds, memo, adminId); // 입금 내역 상태 변경
}

private void DepositConfirmOrders(int id, string orderIds, DepositMatchingInfo submitInfo, string adminId)
{
    var currentInfo = DepositProcInfoService.GetDepositProcInfo(id, orderNums);
    // 기존에 조회했던 데이터 중 바뀐 것이 있는지 체크
    if (InfoIsChanged(currentInfo, submitInfo)) throw new Exception("주문을 확인해주세요.");
    DoDepositConfirmProc(orderIds, adminId); // 주문 입금 상태 변경 (미입금 -> 입금완료)
    DoPointProc(currentInfo, id, orderIds.Split('/')[0]); // 포인트 처리 (지급 or 차감)
}

```

주요 코드 링크

🔴 결과

입금할 주문 선택 → 처리 버튼 클릭 → 입금 처리가 가능하면 '포인트 처리' 또는 '처리 완료' 버튼 활성화

→ 클릭하면 입금 처리

CASE1. 총주문 금액 == 입금액

[Layout]

* 입금액과 총주문금액이 일치할 때

入金確認処理

< 通帳内訳 >

銀行人	三井住友	入出金仕分け	入金
処理日付け	2019-11-26 00:00:00	金額	22,000 円
内容	ｼﾔﾆﾝｺﾞｶｶｺﾝﾔｸｶｲ ｸﾘﾐ		

選択	注文番号	銀行	塚主門禁額	価格(+)	価格(-)	価格(P)	入金内訳	注文金額	中継料	同時発送
<input type="checkbox"/>	191125 - 1984979		45,000	45,500	44,500	45,000	1984979 -	45,000	0	0
<input type="checkbox"/>	191125 - 1984990		49,700	50,200	49,200	49,700	1984990 -	49,700	0	0
<input checked="" type="checkbox"/>	191125 - 1985055		17,000	17,500	16,500	17,000	1985055 -	17,000	0	0
<input checked="" type="checkbox"/>	191125 - 1960487		5,000	5,500	4,500	5,000	1960487 -	5,000	0	0

注文番号

1985055/1960487

※多くの注文番号入力の時記号(/)で区別してください。 ex)124234/234325/234324

端書

入力

初期化

ポイント

保有ポイント	55,500円	差引きポイント	0円	還元ポイント	0円
--------	---------	---------	----	--------	----

1. 適用

2. 総注文額 22,000円

3. 処理可能

4.

5. ポイント処理

6. 処理完了

※処理完了は入金確認した事件だけ書いてください。

*처리가능 일때 선택 X, 버튼 비활성화

CASE2. 총주문 금액 < 입금액

[Layout]

* 입금액이 클 때

入金確認処理

< 通帳内訳 >

銀行人	三井住友	入出金仕分け	入金
処理日付け	2019-11-26 00:00:00	金額	22,000 円
内容	ｼｬｰﾈｸｼｮﾝｺﾝﾅｸﾃｨﾌﾞｲ ﾌﾘｰﾐ		

選択	注文番号	銀行	塚主門禁額	価格(+)	価格(-)	価格(P)	入金内訳	注文金額	中継料	同時発送
<input type="checkbox"/>	191125 - 1984979		45,000	45,500	44,500	45,000	1984979 -	45,000	0	0
<input type="checkbox"/>	191125 - 1984990		49,700	50,200	49,200	49,700	1984990 -	49,700	0	0
<input checked="" type="checkbox"/>	191125 - 1985055		17,000	17,500	16,500	17,000	1985055 -	17,000	0	0
<input type="checkbox"/>	191125 - 1960487		5,000	5,500	4,500	5,000	1960487 -	5,000	0	0

注文番号 1985055 適用 総注文額 17,000円 処理可能

※多くの注文番号入力の時記号(/)で区分してください。 ex)124234/234325/234324

端書

ポイント

保有ポイント	55,500円	差引きポイント	0円	還元ポイント	5,000円
--------	---------	---------	----	--------	--------

2. 3. 4. 5.

※処理完了は入金確認した事件だけ書いてください。

ポイント処理 処理完了

22000円 - 17000円 *처리가능 일 때 선택 X, 버튼 비활성화

CASE3. 입금액 < 총주문 금액 < 입금액 + 고객이 보유한 포인트

[Layout]

* 입금액이 부족할 때 (입금액 < 총주문금액 & 보유포인트 + 입금액 ≥ 총주문금액)

入金確認処理

< 通帳内訳 >

銀行人	三井住友	入出金仕分け	入金
処理日付け	2019-11-26 00:00:00	金額	22,000 円
内容	ｼｬｰﾈｸｼｮﾝｺﾝﾅｸﾃｨﾌﾞｲ ﾌﾘｰﾐ		

選択	注文番号	銀行	塚主門禁額	価格(+)	価格(-)	価格(P)	入金内訳	注文金額	中継料	同時発送
<input checked="" type="checkbox"/>	191125 - 1984979		45,000	45,500	44,500	45,000	1984979 -	45,000	0	0
<input type="checkbox"/>	191125 - 1984990		49,700	50,200	49,200	49,700	1984990 -	49,700	0	0
<input type="checkbox"/>	191125 - 1985055		17,000	17,500	16,500	17,000	1985055 -	17,000	0	0
<input type="checkbox"/>	191125 - 1960487		5,000	5,500	4,500	5,000	1960487 -	5,000	0	0

注文番号 1984979 適用 総注文額 45,000円 処理不可

※多くの注文番号入力の時記号(/)で区分してください。 ex)124234/234325/234324

端書

ポイント

保有ポイント	55,500円	差引きポイント	23,000円	還元ポイント	0円
--------	---------	---------	---------	--------	----

2. 3. 4. 5.

※処理完了は入金確認した事件だけ書いてください。

ポイント処理 処理完了

45000円 - 22000円

*처리불가 일 때 선택 X, 버튼 비활성화

CASE4. 입금액 < 입금액 + 고객이 보유한 포인트 < 총주문 금액

[Layout]

* 입금액 및 포인트도 부족할 때 (입금액 < 총 주문금액 & 보유포인트 + 입금액 < 총주문금액)

入金確認処理

< 通帳内訳 >

銀行人	三井住友	入出金仕分け	入金
処理日付け	2019-11-26 00:00:00	金額	22,000 円
内容	ｼﾝﾅｶｶｼｶﾝｺﾎﾞﾂｸﾅｲ ｱﾘｺﾐ		

選択	注文番号	銀行	塚主門禁額	価格(+)	価格(-)	価格(P)	入金内訳	注文金額	中継料	同時発送
<input checked="" type="checkbox"/>	191125 - 1984979		45,000	45,500	44,500	45,000	1984979 -	45,000	0	0
<input type="checkbox"/>	191125 - 1984990		49,700	50,200	49,200	49,700	1984990 -	49,700	0	0
<input type="checkbox"/>	191125 - 1985055		17,000	17,500	16,500	17,000	1985055 -	17,000	0	0
<input type="checkbox"/>	191125 - 1960487		5,000	5,500	4,500	5,000	1960487 -	5,000	0	0

注文番号

1985055

適用

総注文額

45,000円

1. 処理不可

※多くの注文番号入力の時記号(/)で区分してください。ex)124234/234325/234324

端書

2.

3.

入力

初期化

ポイント

保有ポイント	500円	差引きポイント	23,000円	還元ポイント	0円	ポイント不足で処理ができません。
--------	------	---------	---------	--------	----	------------------

※処理完了は入金確認した事件だけ書いてください。

4. ポイント処理

5. 処理完了

*처리불가 및 포인트 부족 시 선택 X, 버튼 비활성화