A New Reliable ATM

OOPT Phase 2050 & 2060

Construct & Testing

Project Team T6

201411140 권성완
201511247 김선정
201510436 허윤아
201510285 조수빈

Date

2018-06-01

2050 Construct

Activity 2051. Implement Class & Methods Definitions

Type	Class
Name	MainSystem
Purpose	고객과 Offer, Account를 연결하는, ATM의 기능적인 부분의 대부분을 처리하는 클래스
Overview	N/A
Cross Reference	Use case: Deposit, Deposit Without Bankbook, Withdraw, Transfer, Exchange, Loan, Pay Utility Bill, Check Balance, Insert, Print Transaction Receipt, Print Error, Do Forced Termination, Take Charge, Functional Requirements : R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R2.1, R2.2, R2.3, R2.4, R3.1
Exceptional Courses of Events	N/A

Type	Method
Name	insert
Purpose	고객이 ATM에 계좌 혹은 신용카드를 입력할 수 있다.
Cross Reference	Use case : Insert Functional Requirements : R2.1
Input	card: String
Output	void
Abstract operation	고객으로부터 계좌 또는 신용카드 정보를 입력 받는다.
Exceptional Courses of Events	N/A

Type	Method
Name	deposit
Purpose	고객이 계좌 혹은 신용카드에 현금을 입금할 수 있도록 한다.
Cross Reference	Use case : Deposit Functional Requirements : R1.1
Input	money : String
Output	int
Abstract operation	고객이 입금하는 금액을 입력 받은 계좌 혹은 신용카드에 입금한다. 고객이 입력한 금액만큼 DB에 저장되어 있던 잔고가 늘어난다.
Exceptional Courses	N/A

of Events	

Type	Method
Name	depositWithoutBank
Purpose	고객이 입력하는 계좌에 현금을 입금한다.
Cross Reference	Use case : Deposit Without Bankbook Functional: R1.2
Input	money : String
Output	int
Abstract operation	고객이 입금하는 금액을 입력 받은 계좌에 입금한다. 입력 받은 금액만큼 DB에 저장되어 있던 잔고가 늘어난다.
Exceptional Courses of Events	ATM에서 관리하는 은행의 계좌에만 고객이 현금을 입금할 수 있다.

Type	Method
Name	withdraw
Purpose	고객이 입력하는 계좌에서 현금을 출금한다.
Cross Reference	Use case : Withdraw Functional Requirements : R1.3
Input	money : String
Output	int
Abstract operation	고객이 계좌를 입력한 이후 출금할 금액을 입력하면, 그에 맞게 출금(시뮬레이션이므로 거래 내역으로만 표시)하고, DB를 업데이트한다.
Exceptional Courses of Events	N/A

Type	Method
Name	transfer
Purpose	고객이 입력한 계좌로 돈을 송금할 수 있다.
Cross Reference	Use case : Transfer Functional Requirements : R1.4
Input	money : String, newAccount : Account
Output	int
Abstract operation	고객이 송금할 계좌를 입력한 이후 송금 받을 계좌를 입력하고, 송금할 금액을 입력하면, 그에 맞게 DB를 업데이트한다.
Exceptional Courses of Events	N/A

Type	Method
Name	exchange
Purpose	한화에서 다른 국가의 현금 단위로 환전한다.
Cross Reference	Use case : Exchange Functional Requirements : R1.5
Input	money : String, country : String
Output	int
Abstract operation	고객이 환전할 계좌를 입력하고, 환전할 금액을 입력하면, 그만큼 출금(시뮬레이션이므로 거래 내역으로만 표시)하고, DB를 업데이트한다.
Exceptional Courses of Events	N/A

Type	Method
Name	loan
Purpose	고객이 입력한 신용카드를 이용해 대출한다.
Cross Reference	Use case : Loan Functional Requirements : R1.6
Input	money : String
Output	int
Abstract operation	고객이 대출할 신용카드를 입력하고, 대출 금액을 입력하면, 그만큼 출금(시뮬레이션이므로 거래 내역으로만 표시)하고, DB를 업데이트한다.
Exceptional Courses of Events	N/A

Type	Method
Name	payUtilityBill
Purpose	지로고지서에 적힌 계좌로 공과금을 납부한다
Cross Reference	Use case : Pay Utility Bill Functional Requirements : R1.7
Input	newAccount : Account
Output	int
Abstract operation	고객이 계좌를 입력하고, 공과금을 납부할 국가 계좌를 입력하면, 자동으로 공과금을 납부하고, DB를 업데이트한다.
Exceptional Courses of Events	N/A

Type	Method
Name	checkBalance
Purpose	고객이 입력한 계좌 및 신용카드의 거래 내역을 확인한다.
Cross Reference	Use case : Check Balance Functional Requirements : R1.8
Input	void
Output	String
Abstract operation	고객이 입력한 계좌 또는 신용카드의 최근 거래 내역과 해당 거래에 따른 잔액을 출력한다.
Exceptional Courses of Events	N/A

Type	Method
Name	inputPassword
Purpose	계좌 혹은 신용카드의 Password를 입력 받는다.
Cross Reference	Use case : Withdraw, Transfer, Exchange, Loan, Pay Utility Bill, Check Balance, Check Password Functional Requirements : R1.3, R1.4, R1.5, R1.6, R1.6, R1.7, R1.8, R4.1
Input	password : int
Output	password : int
Abstract operation	고객이 입력한 비밀번호를 ATM에 저장한다.
Exceptional Courses of Events	-

Type	Method
Name	printTransactionRecepit
Purpose	거래가 끝난 이후 거래명세서를 출력한다.
Cross Reference	Use case : Print Transaction Receipt Functional Requirements : R2.2
Input	
Output	void
Abstract operation	거래가 끝난 이후, DB를 업데이트하고 거래 명세서를 출력한다.
Exceptional Courses of Events	N/A

Type	Method
Name	doForcedTermination

Purpose	강제로 거래를 종료하고 초기 화면으로 돌아간다.
Cross Reference	Use case : Do Forced Termination Functional Requirements : R2.4
Input	
Output	void
Abstract operation	에러가 3번 발생했을 경우 거래를 강제로 종료하고 초기 화면으로 돌아간다.
Exceptional Courses of Events	N/A

Type	Method
Name	printError
Purpose	에러가 발생했을 때 에러를 출력한다.
Cross Reference	Use case : Print Error Functional Requirements : R2.3
Input	errorType : int
Output	void
Abstract operation	에러가 발생한 경우(비밀번호 오류, 입력된 정보 오류 등) 에러를 출력한다.
Exceptional Courses of Events	N/A

Type	Method
Name	takeCharge
Purpose	고객의 등급에 따라 수수료를 부과한다
Cross Reference	Use case : Withdraw, Transfer, Exchange, Loan, Take Charge Functional Requirements : R1.3, R1.4, R1.5, R1.6, R3.1
Input	account : Account
Output	void
Abstract operation	수수료를 등급에 맞게 계산하여 부과하고, 그에 따른 변동사항을 DB에 저장한다.
Exceptional Courses of Events	등급이 높은 경우 수수료가 면제된다.

Type	Class
Name	Account
Purpose	MainSystem과 offer의 Database 사이의 정보를 연결하는 클래스.

	MainSystem은 이 클래스의 정보를 변경하고, 이 클래스의 정보를 Offer의 Database에 저장한다.
Overview	N/A
Cross Reference	Use case : Check Password Functional Requirements : R4.1
Exceptional Courses of Events	N/A

Type	Method
Name	getPassword
Purpose	Account의 password를 가져온다.
Cross Reference	Use case : Check Password Functional Requirements : R4.1
Input	void
Output	password : int
Abstract operation	account의 password를 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	getLimit
Purpose	Account의 limit를 가져온다.
Cross Reference	Use case : Loan Functional Requirements : R1.6
Input	void
Output	limit : String
Abstract operation	account의 대출 한도를 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	getBalance
Purpose	Account의 balance를 가져온다.
Cross Reference	Use case : Withdraw, Transfer, Exchange, Pay Utility Bill Functional Requirements : R1.3, R1.4, R1.5, R1.7
Input	void
Output	balance : String

Abstract operation	account의 남은 잔액을 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	getIsLocked
Purpose	Account의 isLocked를 가져온다.
Cross Reference	Use case : Transaction Lock Functional Requirements : R5.1
Input	void
Output	isLocked : boolean
Abstract operation	account가 잠긴 상태인지 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	getBank
Purpose	Account의 bank를 가져온다.
Cross Reference	Use case : Deposit, Withdraw, Transfer, Exchange, Pay Utility Bill, Check Balance Functional Requirements : R1.1, R1.3, R1.4, R1.5, R1.7, R1.8
Input	void
Output	bank : String
Abstract operation	account의 은행 정보를 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	getRate
Purpose	Account의 rate를 가져온다.
Cross Reference	Use case : Withdraw, Transfer, Exchange, Loan, Take Charge Functional Requirements : R1.3, R1.4, R1.5, R1.6, R3.1
Input	Void
Output	rate : int
Abstract operation	account의 등급을 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	getDept
Purpose	Account의 dept를 가져온다.
Cross Reference	Use case : Loan Functional Requirements : R1.6
Input	void
Output	dept : String
Abstract operation	account의 빚 금액을 리턴한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setLimit
Purpose	Account의 limit를 설정한다.
Cross Reference	Use case : Loan Functional Requirements : R1.6
Input	limit : String
Output	void
Abstract operation	account의 대출 한도를 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setBalance
Purpose	Account의 balance를 설정한다.
Cross Reference	Use case : Withdraw, Transfer, Exchange, Pay Utility Bill Functional Requirements : R1.3, R1.4, R1.5, R1.7
Input	balance : int
Output	void
Abstract operation	account의 남은 잔액을 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setDept

Purpose	Account의 dept를 설정한다.
Cross Reference	Use case : Loan Functional Requirements : R1.6
Input	dept : String
Output	void
Abstract operation	Account의 대출로 발생한 빚을 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setIsLocked
Purpose	Account의 isLocked를 설정한다.
Cross Reference	Use case : Transaction Lock Functional Requirements : R5.1
Input	isLocked : boolean
Output	void
Abstract operation	Account가 Transaction Lock 상태인지 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setRate
Purpose	Account의 rate를 설정한다.
Cross Reference	Use case : Withdraw, Transfer, Exchange, Loan, Take Charge Functional Requirements : R1.3, R1.4, R1.5, R1.6, R3.1
Input	rate : int
Output	void
Abstract operation	account의 신용등급을 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setBank
Purpose	Account의 bank를 설정한다.
Cross Reference	Use case : Deposit, Withdraw, Transfer, Exchange, Pay Utility Bill, Check Balance Functional Requirements : R1.1, R1.3, R1.4, R1.5, R1.7, R1.8

Input	bank : String
Output	void
Abstract operation	Int로 offer code같은 정보를 받아 string인 bank를 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	setPassword
Purpose	Account의 password를 설정한다.
Cross Reference	Use case : Check Password Functional Requirements : R4.1
Input	password : int
Output	void
Abstract operation	Account의 비밀번호를 설정한다.
Exceptional Courses of Events	N/A

Type	Method
Name	checkPasword
Purpose	Account의 password와 입력한 password가 일치하는지 확인한다.
Cross Reference	Use case : Check Password Functional Requirements : R4.1
Input	password : int
Output	boolean
Abstract operation	withdraw같은 과정 중에 유저가 입력한 password와 account에 설정된 password가 일치하는지 확인하고 true/false를 리턴한다.
Exceptional Courses of Events	N/A

Type	Class
Name	Offer
Purpose	Database를 관리하고 Main system에 정보를 넘겨주는 클래스
Overview	N/A
Cross Reference	Use case : Check Validation, Update Database Functional Requirements : R6.1, R6.2
Exceptional Courses of Events	N/A

Type	Constructor
Name	Offer
Purpose	company의 값에 따라 맞는 Offer를 생성한다.
Abstract operation	company값에 따라 각각 다른 offer를 생성하는 생성자
Exceptional Courses of Events	N/A

Type	Method
Name	updateDatabase
Purpose	Database를 업데이트한다
Cross Reference	Use case : Update Database Functional Requirements : R6.2
Input	newAccount : Account
Output	boolean
Abstract operation	Database를 account객체의 정보와 같게 저장한다.
Exceptional Courses of Events	N/A

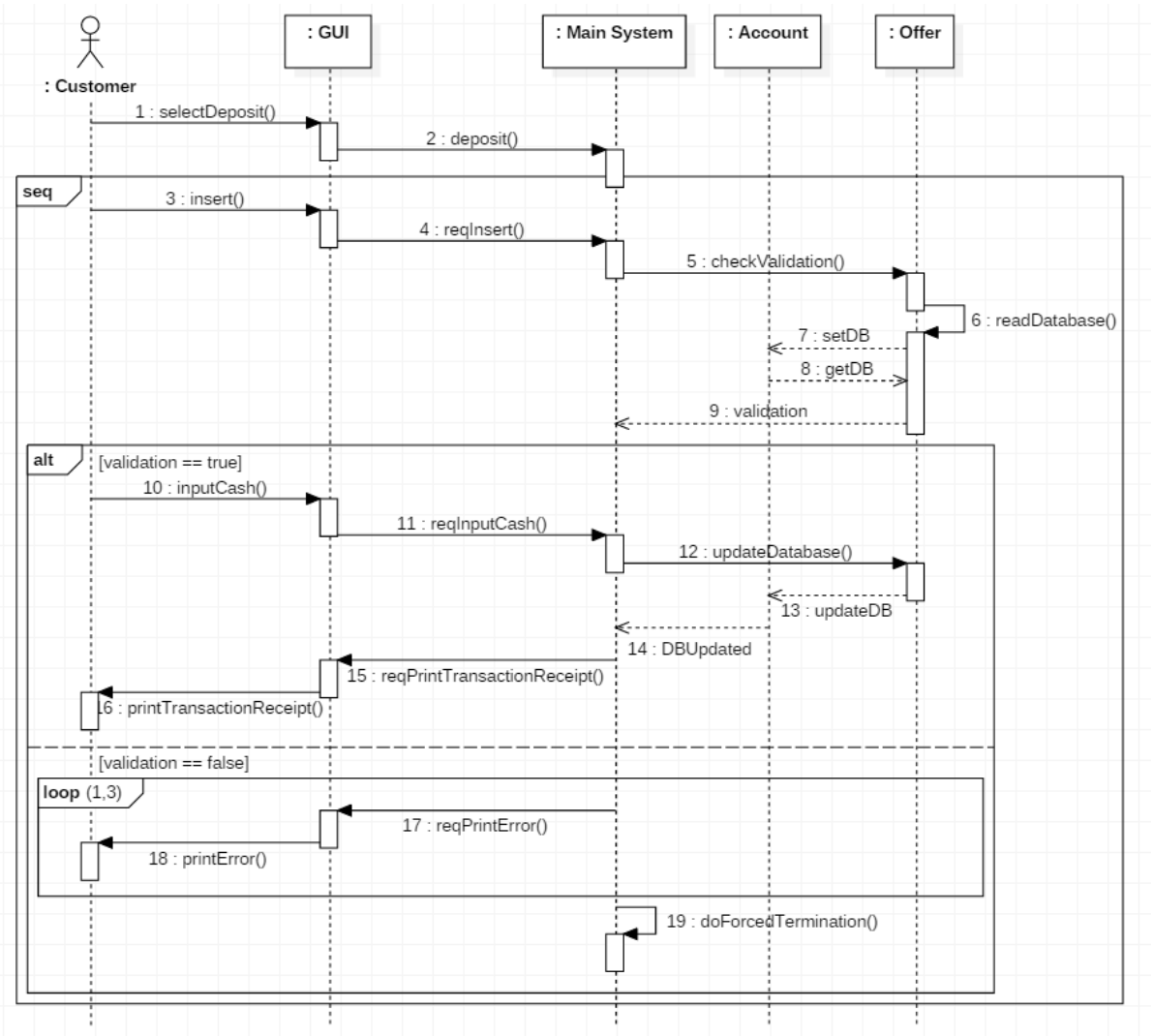
Type	Method
Name	checkValidation
Purpose	Account의 유효성을 체크하고 database의 내용을 Account에 저장한다.
Cross Reference	Use case : Check Validation Functional Requirements : R6.1
Input	account : Account
Output	boolean
Abstract operation	account객체를 database의 정보와 같게 저장한다.
Exceptional Courses of Events	N/A

Type	Method
Name	readDatabase
Purpose	Database에 있는 파일의 정보를 읽어온다.
Cross Reference	Use case: Check Validation Functional Requirements : R6.1
Input	currentAccount : Account

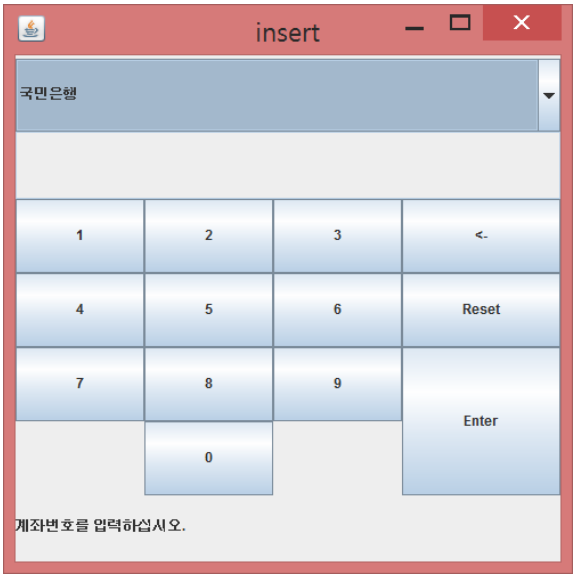
Output	Int
Abstract operation	Database(txt)파일의 정보를 읽어온다.
Exceptional Courses of Events	-

Activity 2052. Implements Windows

1. Deposit

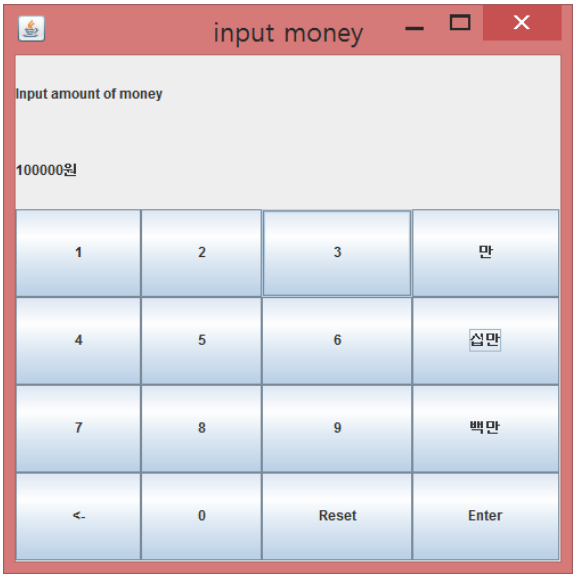


Name	selectDeposit
Responsibilities	기본 화면에서 입금 버튼을 누른다.
Type	GUI
Cross Reference	R1.1
Pre-Conditions	기본 화면
Post-Conditions	입금 기능을 시작한다.

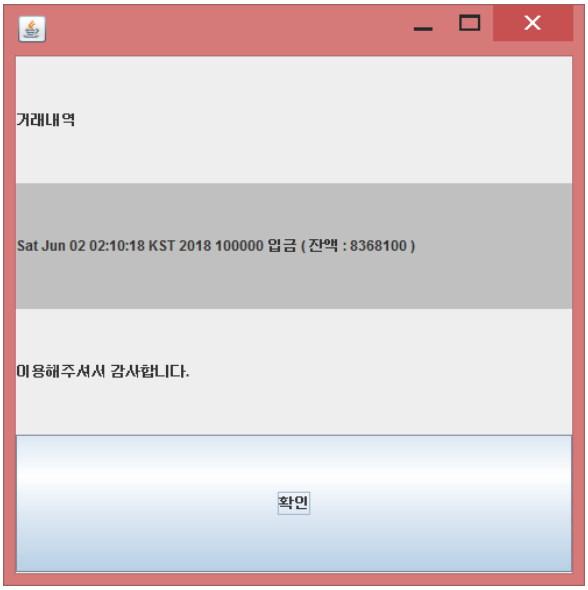


Name	insert
Responsibilities	계좌 혹은 신용카드를 입력한다.

Type	GUI
Cross Reference	R1.1, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8
Pre-Conditions	기본 화면에서 기능 선택 후
Post-Conditions	계좌 혹은 신용카드의 유효성을 확인한다.

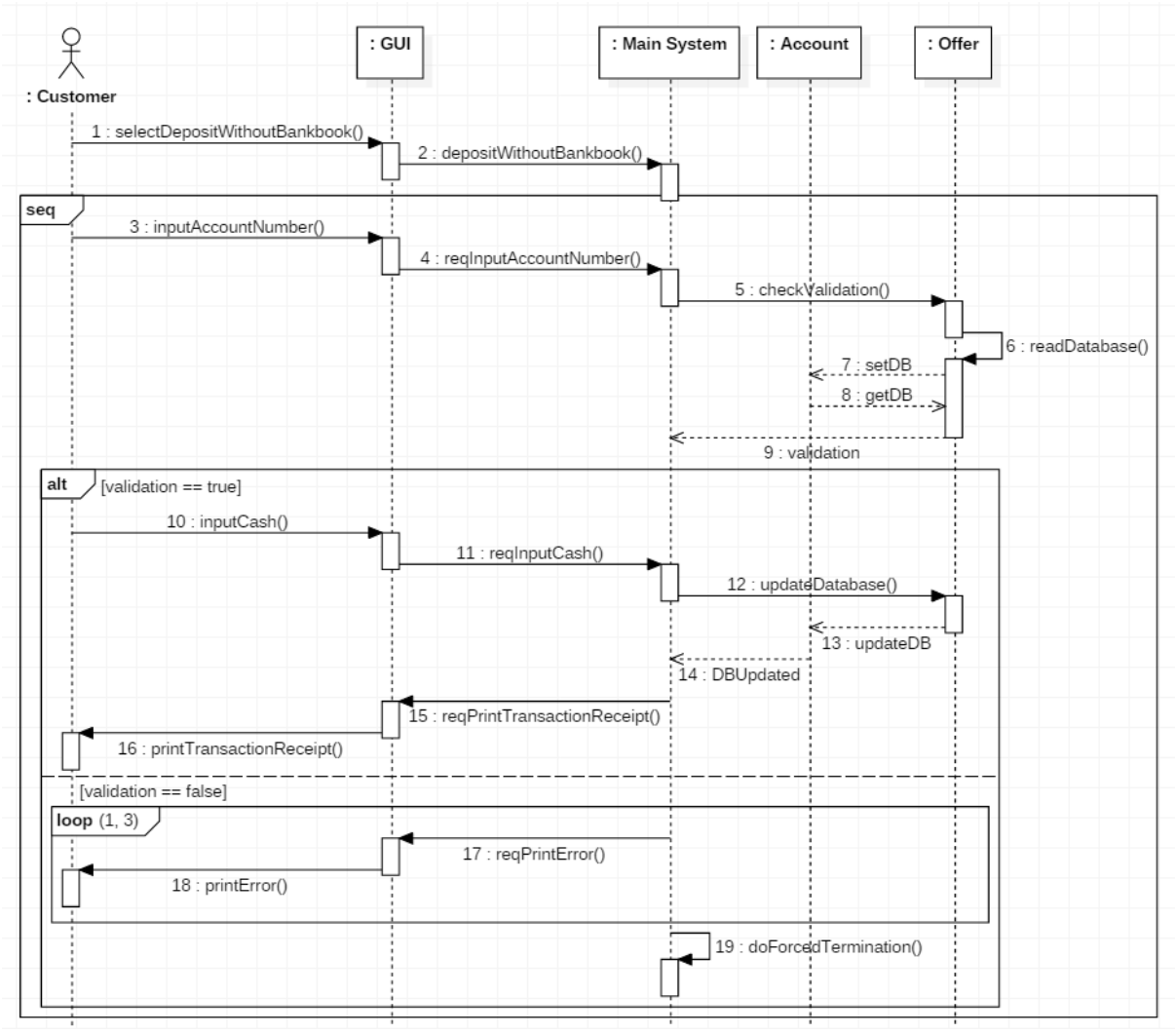


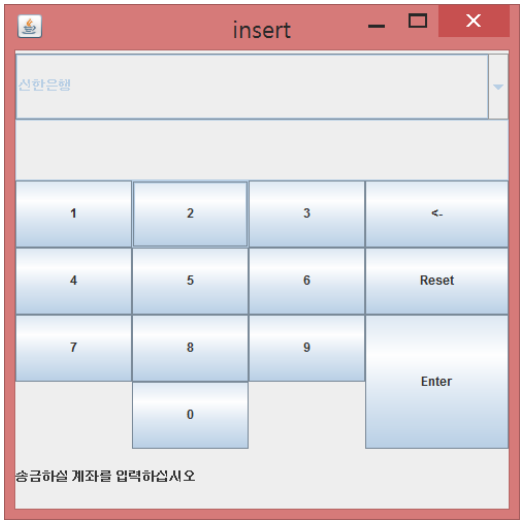
Name	inputCash
Responsibilities	현금을 입력한다
Type	GUI
Cross Reference	R1.1, R1.2
Pre-Conditions	기본 화면에서 기능 선택 후
Post-Conditions	현금을 세고, DB를 업데이트한다.



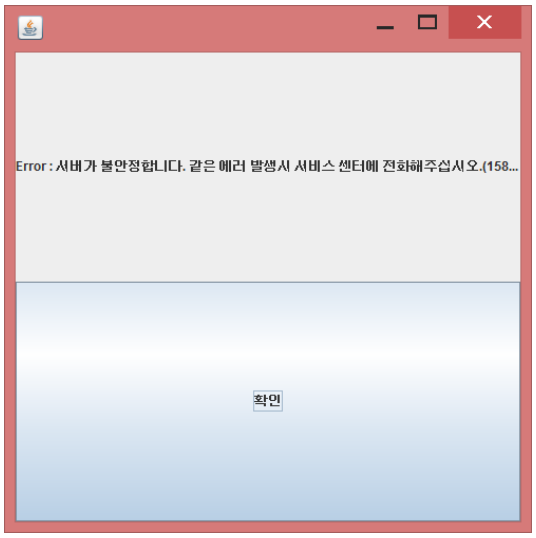
Name	printTransactionReceipt
Responsibilities	거래명세서를 출력한다. 수수료는 부과되지 않는다.
Type	GUI
Cross Reference	R1.1, R1.2, R2.2
Pre-Conditions	거래가 끝난 후 DB를 업데이트한다.
Post-Conditions	확인 버튼을 누르면 기본 화면으로 돌아간다.

2. Deposit Without Bankbook



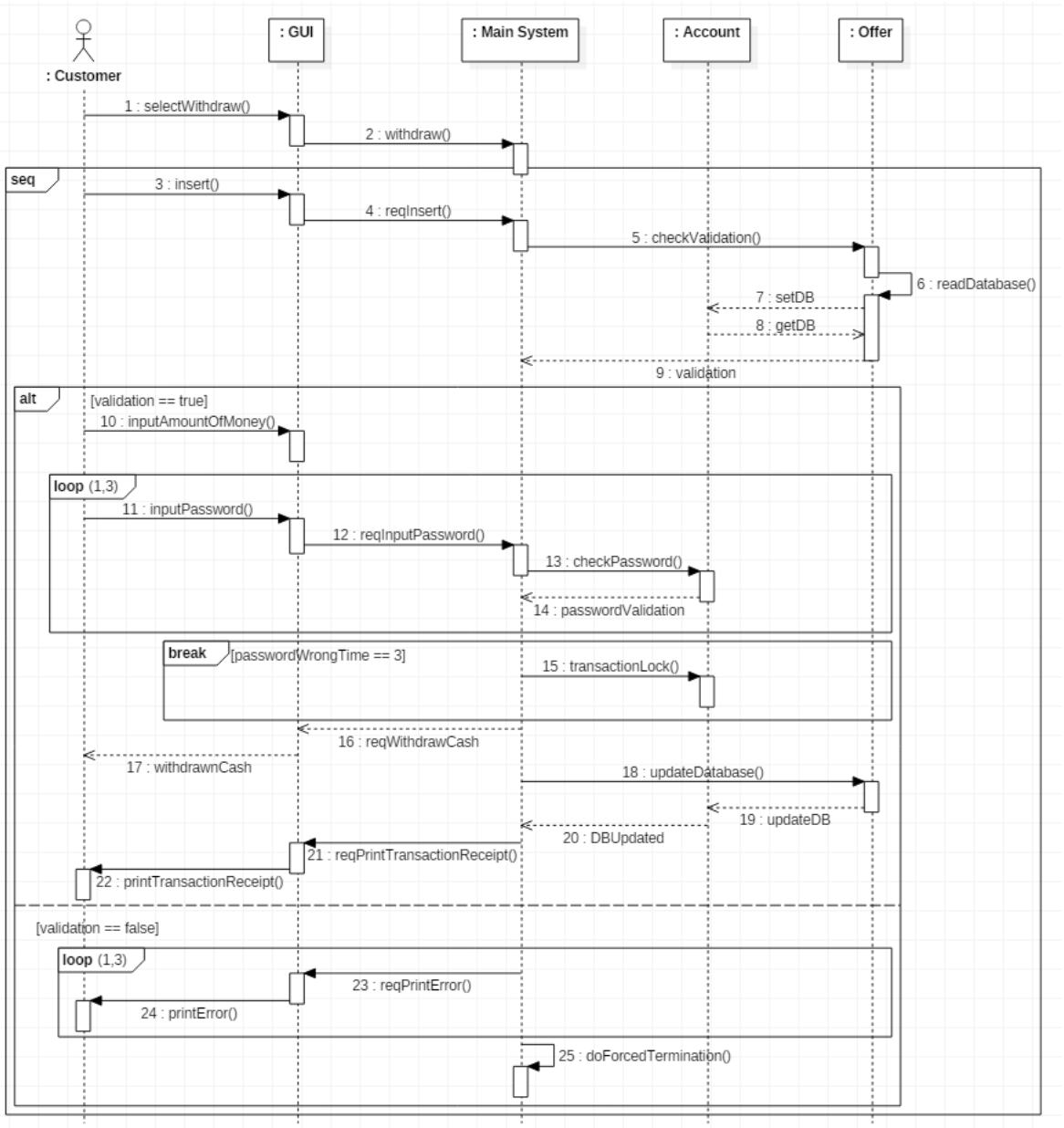


Name	inputAccountNumber
Responsibilities	계좌번호를 입력한다
Type	GUI
Cross Reference	R1.2
Pre-Conditions	무통장입금 기능 선택
Post-Conditions	계좌번호의 유효성을 확인한다.



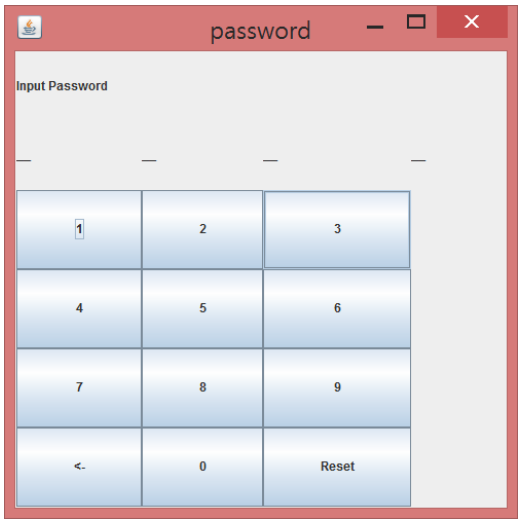
Name	checkValidation
Responsibilities	계좌번호의 유효성을 확인한다.
Type	GUI
Cross Reference	R1.2, R2.3, R6.1
Pre-Conditions	계좌번호 입력
Post-Conditions	초기 화면으로 돌아간다.

3. Withdraw

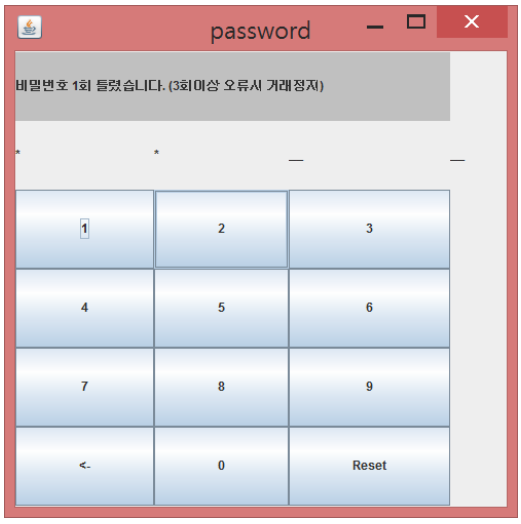

: Customer
: GUI
: Main System
: Account
: Offer
1 : selectWithdraw()
2 : withdraw()
seq
3 : insert()
4 : reqInsert()
5 : checkValidation()
6 : readDatabase()
7 : setDB
8 : getDB
9 : validation
alt
[validation == true]
10 : inputAmountOfMoney()
loop (1,3)
11 : inputPassword()
12 : reqInputPassword()
13 : checkPassword()
14 : passwordValidation
break
[passwordWrongTime == 3]
15 : transactionLock()
16 : reqWithdrawCash
17 : withdrawnCash
18 : updateDatabase()
19 : updateDB
20 : DBUpdated
21 : reqPrintTransactionReceipt()
22 : printTransactionReceipt()
[validation == false]
loop (1,3)
23 : reqPrintError()
24 : printError()
25 : doForcedTermination()


select
up
10000원권 :
3
down
up
50000원권 :
1
down
확인
금액 : 80000

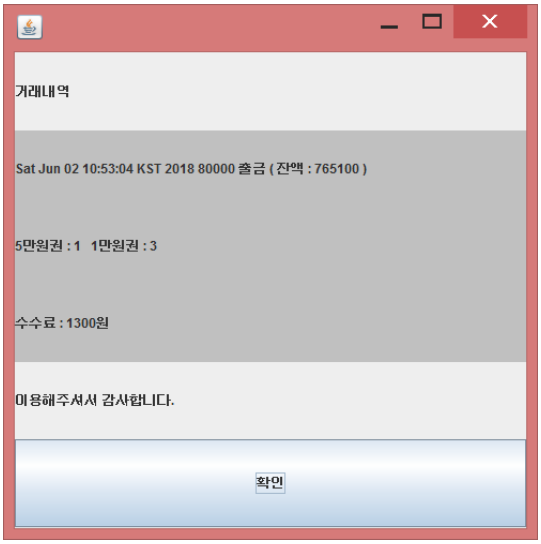
Name	inputAmountOfMoney
Responsibilities	거래할 금액을 입력한다. 5만원권으로 선택적으로 거래할 수 있다.
Type	GUI
Cross Reference	R1.3, R1.4, R1.5, R1.6
Pre-Conditions	거래할 금액 입력 이후
Post-Conditions	비밀번호를 입력한다.

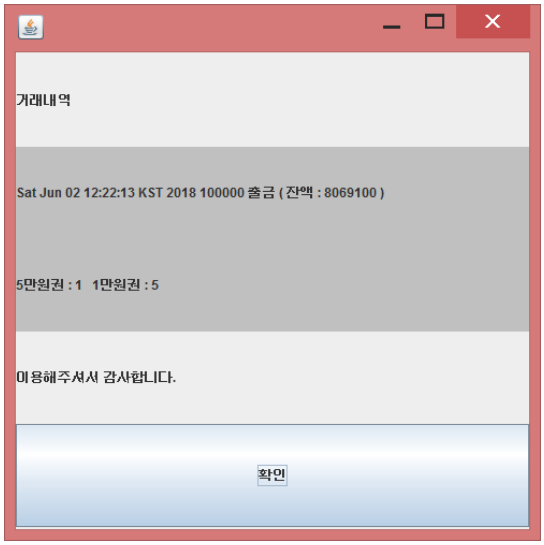


Name	inputPassword
Responsibilities	비밀번호를 입력한다
Type	GUI
Cross Reference	R1.3, R1.4, R1.5, R1.6, R1.7, R1.8
Pre-Conditions	거래 금액 입력
Post-Conditions	비밀번호가 맞는지 확인한다



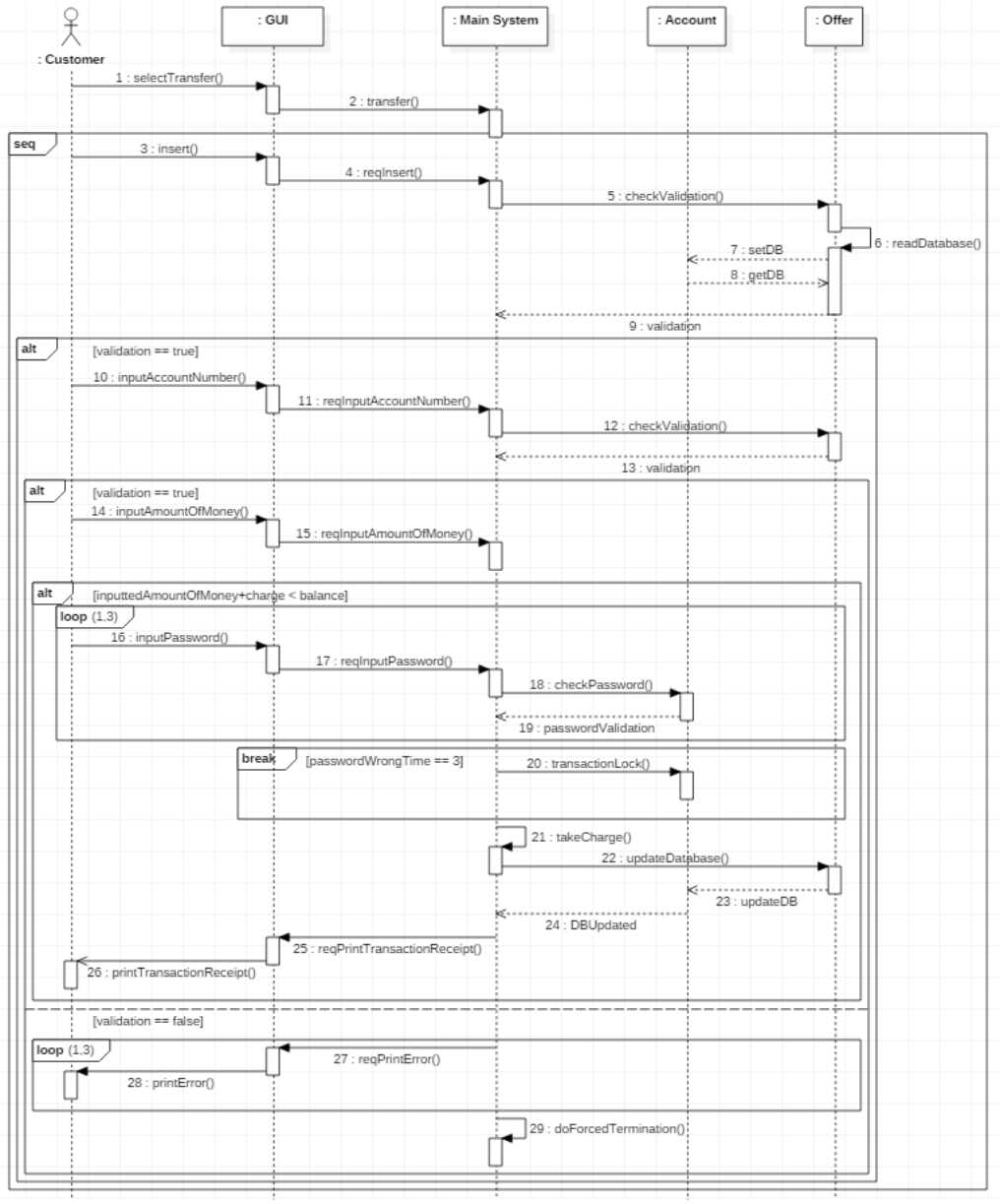
Name	checkPassword
Responsibilities	비밀번호가 맞는지 확인한다
Type	GUI
Cross Reference	R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R2.3, R4.1
Pre-Conditions	비밀번호 입력
Post-Conditions	3번 틀리면 계좌를 잠근다. 3번보다 적게 틀리고 올바른 비밀번호를 입력할 경우 거래를 계속한다.

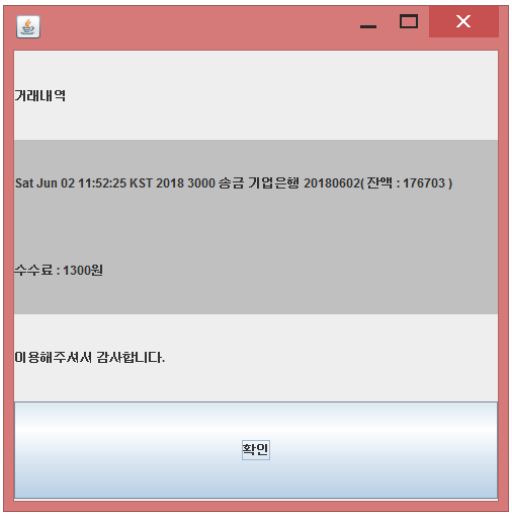




Name	printTransactionReceipt
Responsibilities	거래 명세서가 출력된다. 수수료가 등급에 따라 부여되고, 부여되지 않고를 확인할 수 있다.
Type	GUI
Cross Reference	R1.3, R2.3
Pre-Conditions	DB 업데이트
Post-Conditions	확인 버튼을 누르면 초기 화면으로 돌아간다.

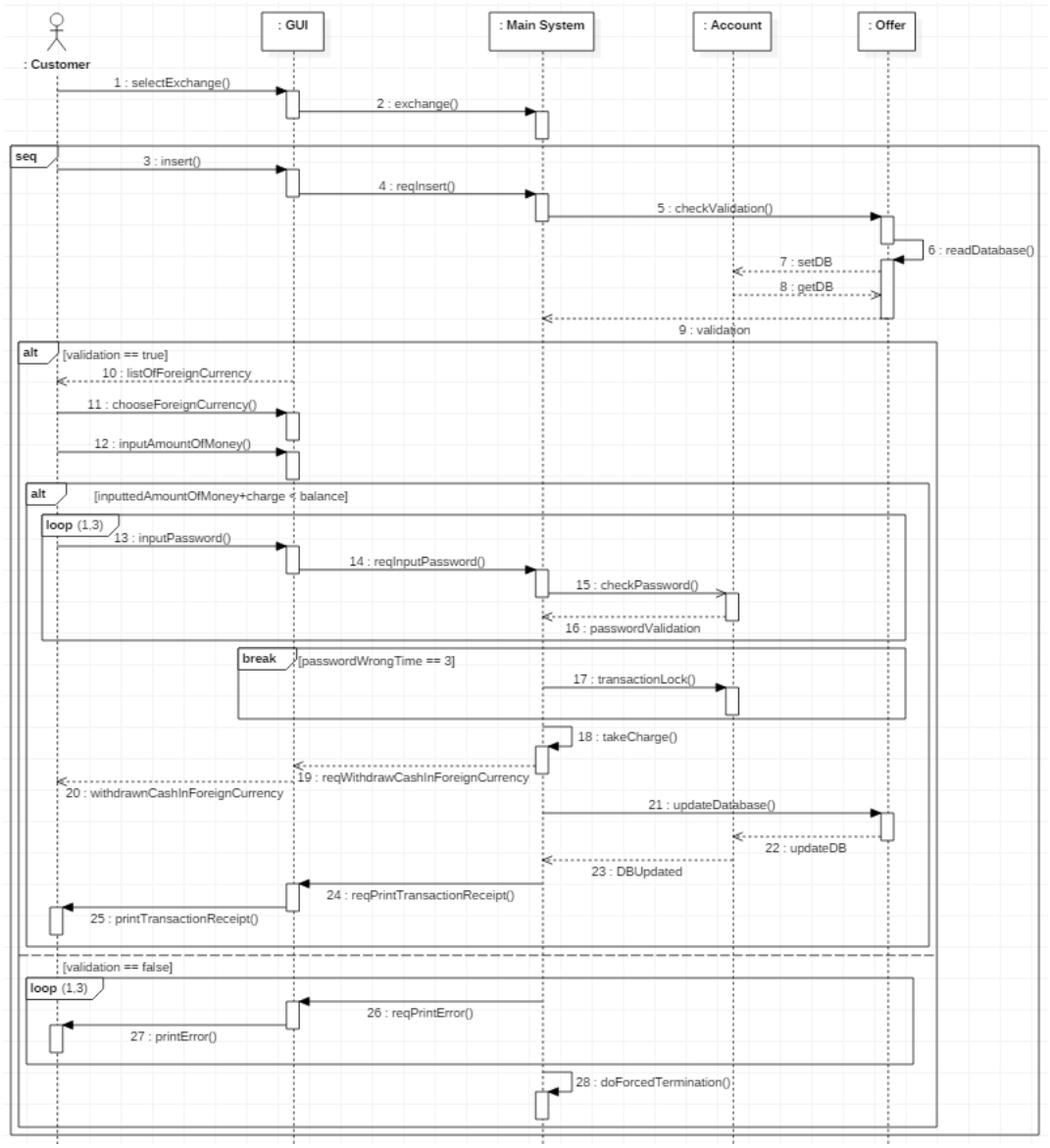
4. Transfer


: Customer
: GUI
: Main System
: Account
: Offer
1 : selectTransfer()
2 : transfer()
seq
3 : insert()
4 : reqInsert()
5 : checkValidation()
6 : readDatabase()
7 : setDB
8 : getDB
9 : validation
alt
[validation == true]
10 : inputAccountNumber()
11 : reqInputAccountNumber()
12 : checkValidation()
13 : validation
alt
[validation == true]
14 : inputAmountOfMoney()
15 : reqInputAmountOfMoney()
alt
[inputtedAmountOfMoney+charge < balance]
loop (1,3)
16 : inputPassword()
17 : reqInputPassword()
18 : checkPassword()
19 : passwordValidation
break
[passwordWrongTime == 3]
20 : transactionLock()
21 : takeCharge()
22 : updateDatabase()
23 : updateDB
24 : DBUpdated
25 : reqPrintTransactionReceipt()
26 : printTransactionReceipt()
[validation == false]
loop (1,3)
27 : reqPrintError()
28 : printError()
29 : doForcedTermination()

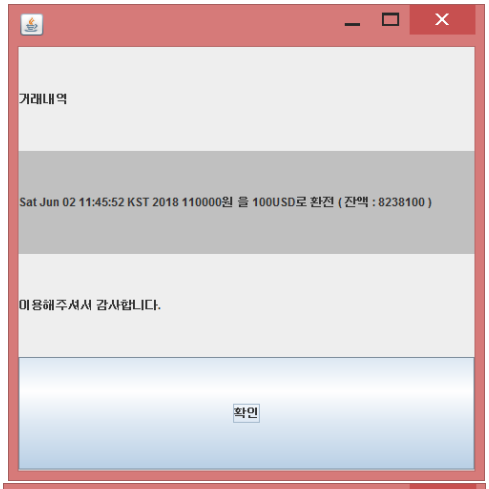

거래내역
Sat Jun 02 11:52:25 KST 2018 3000 송금 기업은행 20180602(잔액 : 176703)
수수료 : 1300원
이용해주셔서 감사합니다.
확인

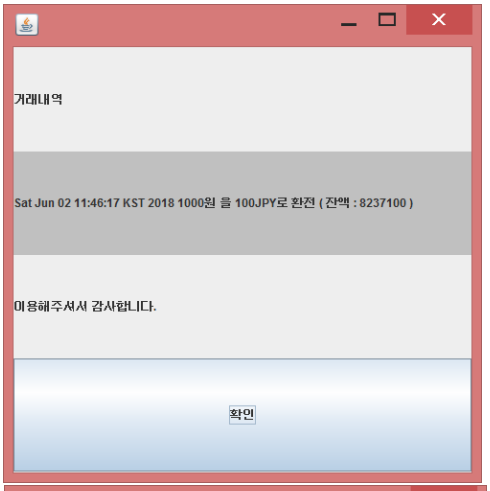
Name	printTransactionReceipt
Responsibilities	송금 거래 명세서를 출력한다. 수수료는 등급에 따라 부과된다.
Type	GUI
Cross Reference	R1.4, R2.3
Pre-Conditions	DB 업데이트
Post-Conditions	확인 버튼을 누르면 초기 화면으로 돌아간다.

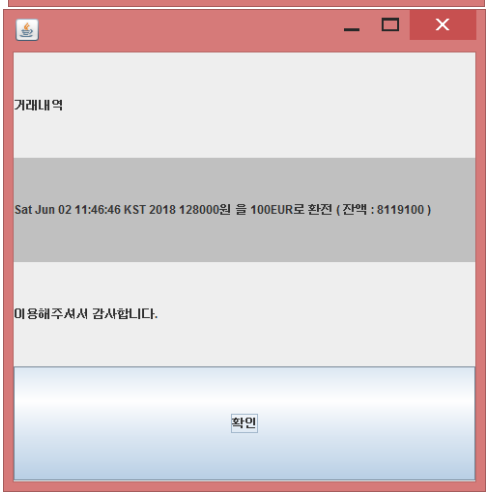
5. Exchange

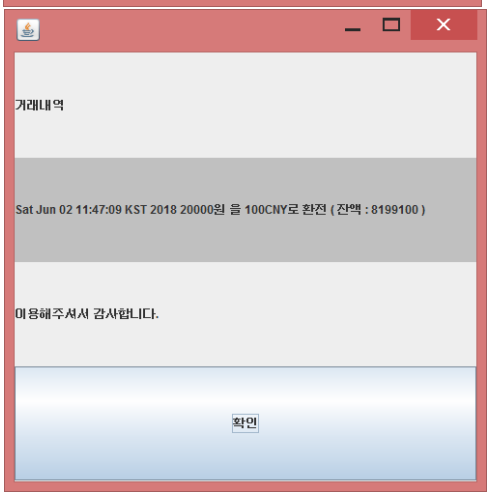


Name	chooseForeignCurrency
Responsibilities	환전하고자 하는 외화의 종류를 선택한다
Type	GUI
Cross Reference	R1.5
Pre-Conditions	고객이 입력한 계좌의 유효성 확인
Post-Conditions	환전할 금액을 입력한다



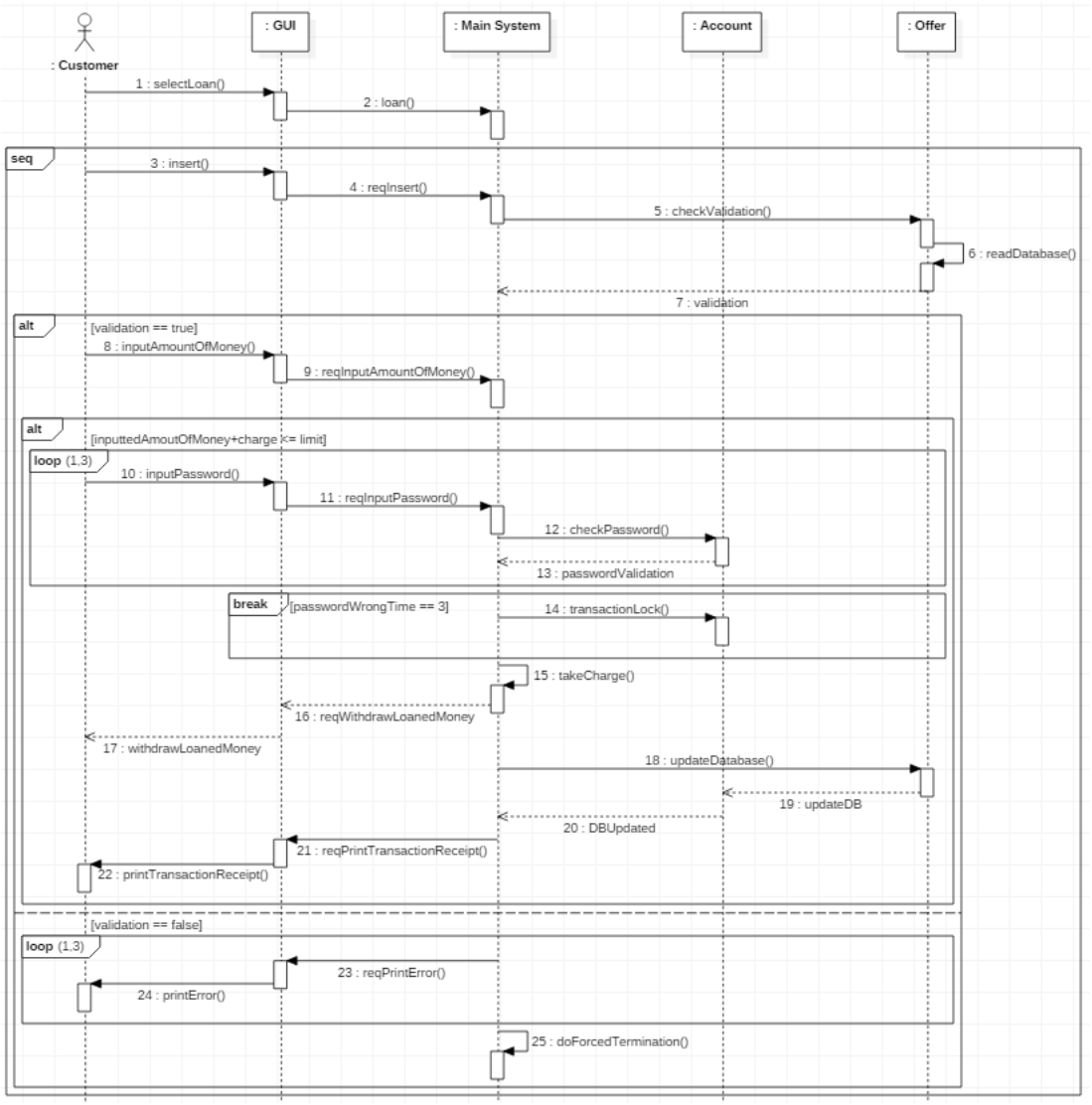


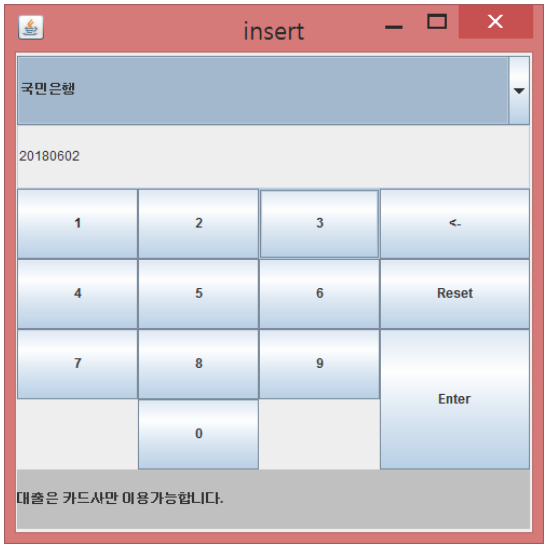




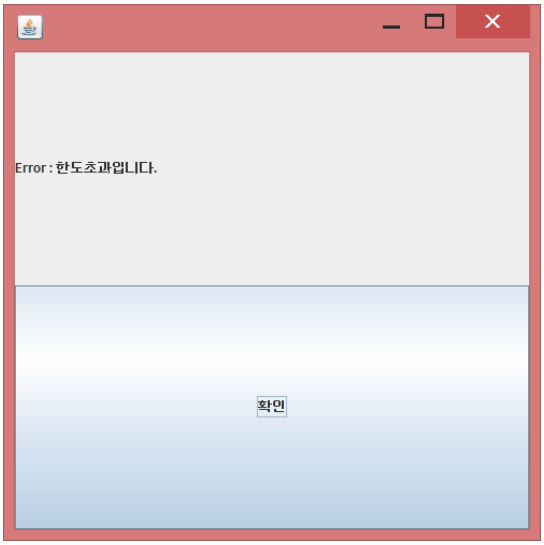
Name	printTransactionReceipt
Responsibilities	자동으로 수수료가 부과된 거래 명세서를 출력한다
Type	GUI
Cross Reference	R1.5, R2.3, R3.1
Pre-Conditions	DB 업데이트
Post-Conditions	확인 버튼을 누르면 초기 화면으로 돌아간다.

6. Loan



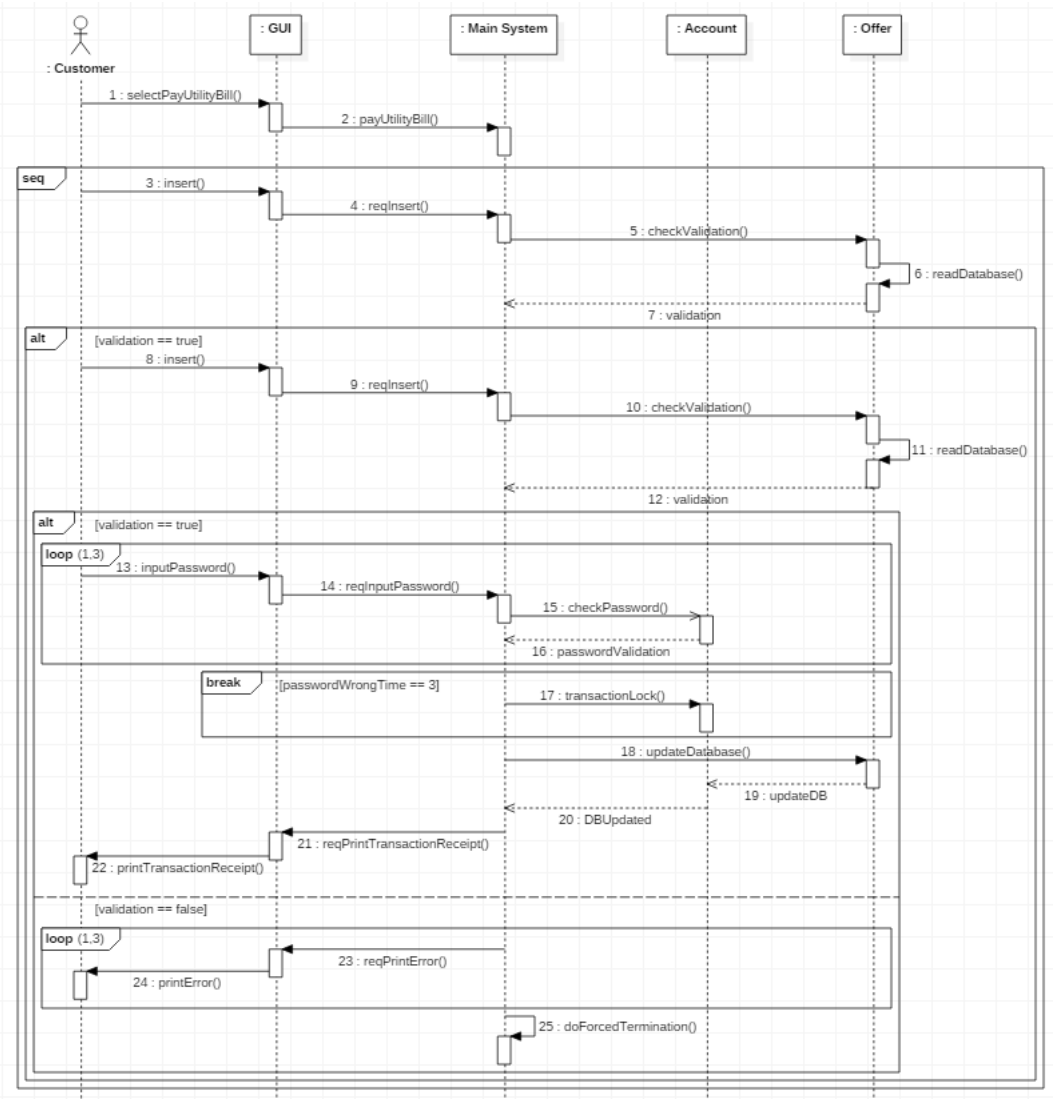


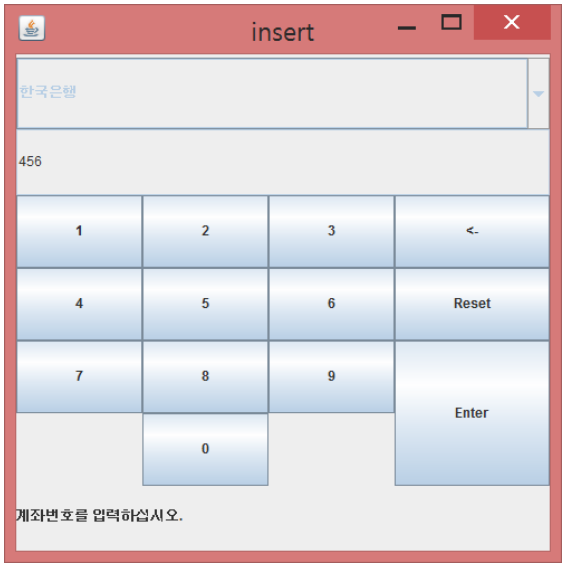
Name	checkValidation
Responsibilities	입력한 신용카드의 유효성을 확인한다
Type	GUI
Cross Reference	R1.6, R2.1, R6.1
Pre-Conditions	신용카드 입력
Post-Conditions	에러 메시지를 출력한다.



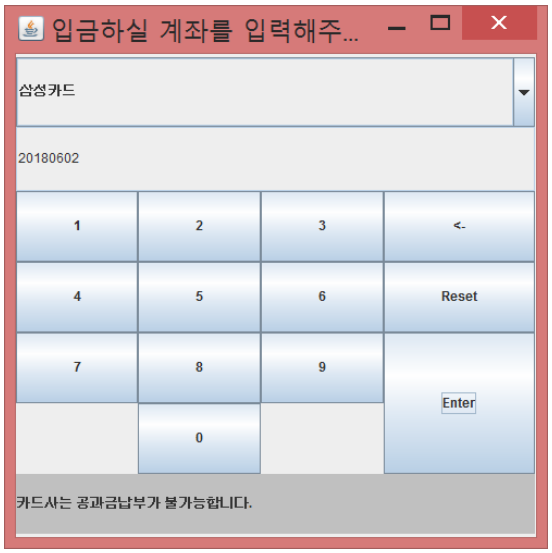
Name	printError
Responsibilities	신용카드 한도보다 더 많이 대출할 경우 한도 초과 메시지를 띄운다
Type	GUI
Cross Reference	R1.6, R2.3
Pre-Conditions	비밀번호 입력
Post-Conditions	확인 버튼을 누르면 초기 화면으로 돌아간다.

7. Pay Utility Bill

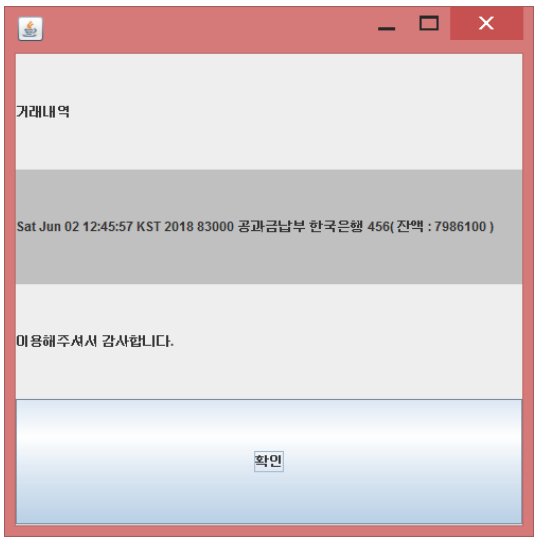




Name	insert
Responsibilities	본인 명의의 국가 계좌를 입력한다.
Type	GUI
Cross Reference	R1.7, R2.1
Pre-Conditions	공과금납부 버튼 선택
Post-Conditions	올바르게 입력했을 경우 본인 명의의 계좌를 입력하는 창으로 넘어간다.

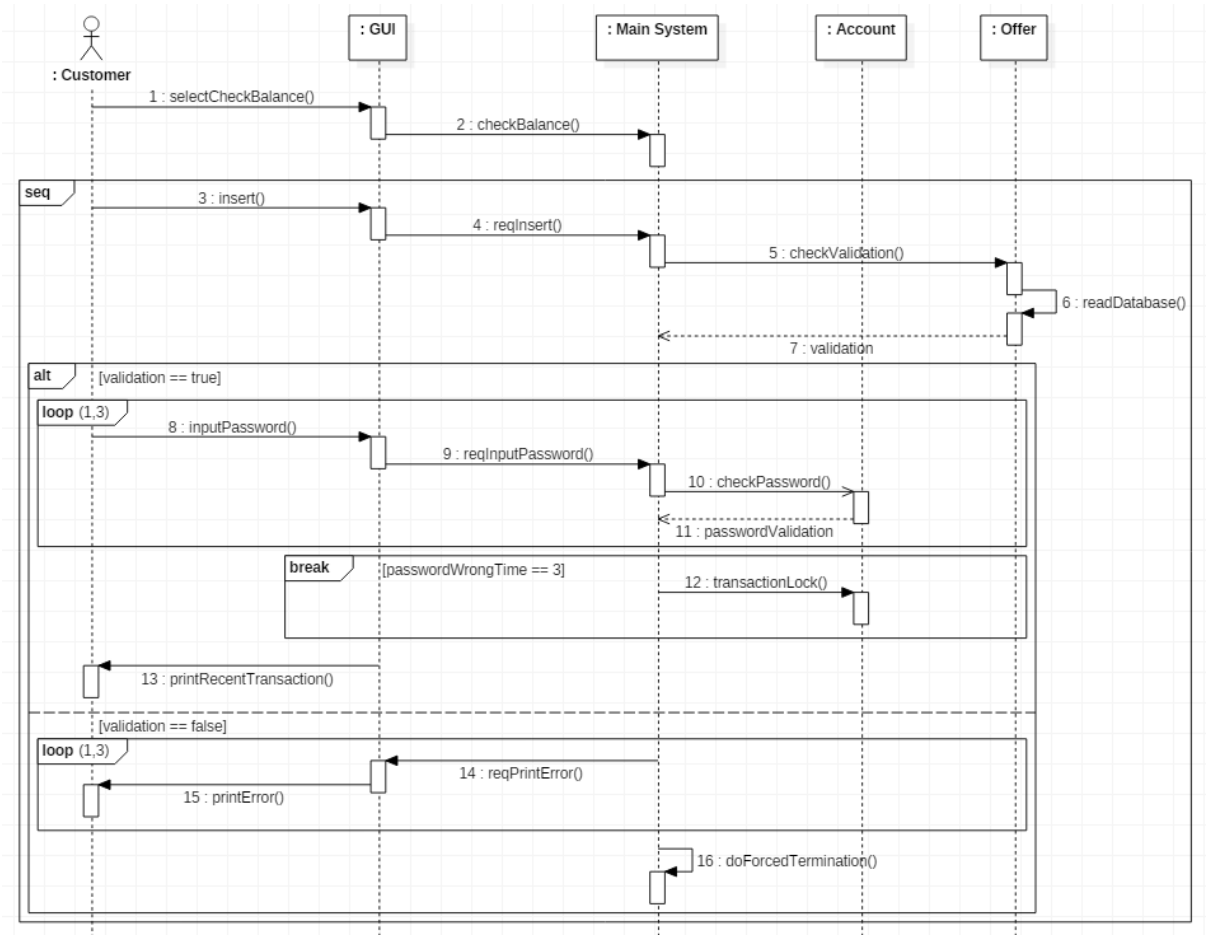


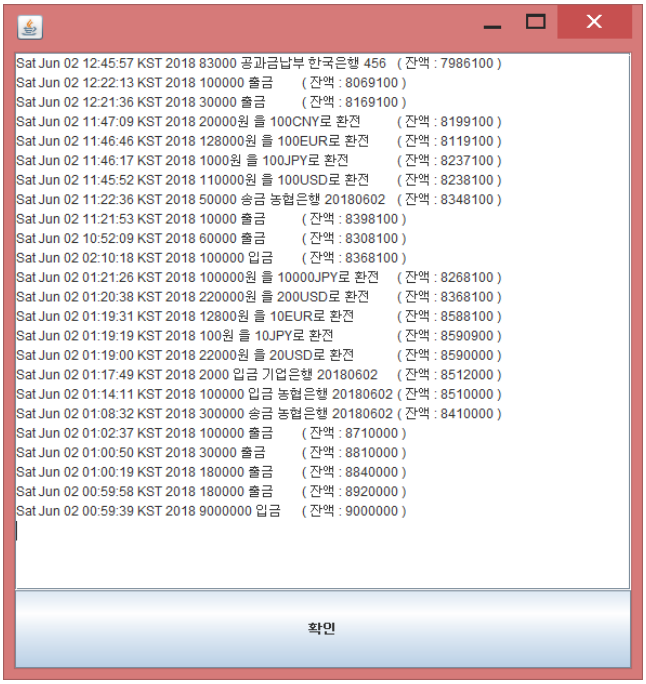
Name	printError
Responsibilities	계좌 대신 신용카드를 입력했을 경우 에러 메시지를 출력한다.
Type	GUI
Cross Reference	R1.7, R2.3
Pre-Conditions	본인 명의의 국가 계좌 입력
Post-Conditions	신용카드가 아닌 계좌를 입력할 경우 다음 창으로 넘어간다.



Name	printTransactionReceipt
Responsibilities	거래명세서를 출력한다.
Type	GUI
Cross Reference	R1.7, R2.2
Pre-Conditions	비밀번호 입력
Post-Conditions	확인 버튼을 누르면 초기 화면으로 돌아간다.

8. Check Balance





Name	printRecentTransaction
Responsibilities	최근 거래 내역을 출력한다.
Type	GUI
Cross Reference	R1.8
Pre-Conditions	비밀번호 입력
Post-Conditions	확인 버튼을 누르면 초기 화면으로 돌아간다.

Activity 2055. Write Unit Test Code

1. File Test Code

```
@Test
public void testCheckValid() {
    Offer test = new Offer(0);
    Account account = new Account();
    account.setBank("국민은행");
    account.setAccountNumber("23456789");
    assertEquals(true, test.checkValid(account));
}

@Test
public void testUpdateDatabase() {
    Offer test = new Offer(0);
    Account testaccount = new Account();
    testaccount.setBank("국민은행");
    testaccount.setAccountNumber("23456789");
    testaccount.setBalance("222080");
    testaccount.setIsLocked(false);
    testaccount.setName("권성환");
    testaccount.setPassword(4752);
    testaccount.setRate(3);
    testaccount.setLog("");
    assertEquals(true, test.updateDatabase(testaccount));
}
```

```
@Test
public void testReadDatabase() {
    Account testaccount = new Account();
    testaccount.setBank("국민은행");
    testaccount.setAccountNumber("23456789");
    testaccount.setBalance("222080");
    testaccount.setIsLocked(false);
    testaccount.setName("권성완");
    testaccount.setPassword(4752);
    testaccount.setRate(3);
    Account account = new Account();
    Offer test = new Offer(0);
    account.setBank("국민은행");
    account.setAccountNumber("23456789");
    test.readDatabase(account);
    assertEquals(testaccount.getBalance(), account.getBalance());
}
```

2. System Test Code

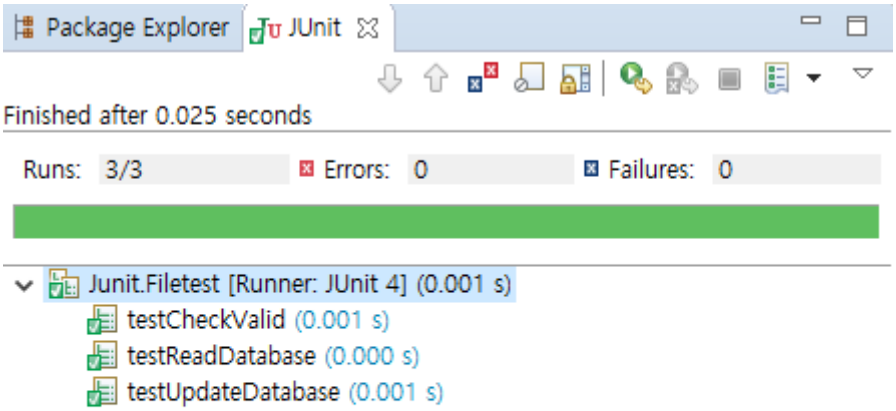
```
@Test
public void testGetAccount() {
    main.insert("023456789");
    main.offer[0].readDatabase(main.getAccount());
    assertEquals(main.getAccount().getAccountNumber(), "23456789");
}

@Test
public void testSetAccount() {
    Account account = new Account();
    account.setAccountNumber("23456789");
    account.setBank("국민은행");
    main.offer[0].readDatabase(account);
    main.setAccount(account);
    assertEquals(account.getBalance(), main.getAccount().getBalance());
}

@Test
public void testInsert() {
    main.insert("023456789");
    main.setAccount(main.getAccount());
    assertEquals(main.getAccount().getAccountNumber(), "23456789");
}

@Test
public void testLockAccount() {
    main.insert("023456789");
    assertEquals(main.lockAccount(), 2);
}

@Test
public void testDeposit() {
    main.insert("023456789");
    assertEquals(main.deposit("30000"), 2);
}

@Test
public void testWithdraw() {
    main.insert("023456789");
    assertEquals(main.withdraw("30000"), 2);
}
```

```java
@Test
public void testExchange() {
    main.insert("023456789");
    main.getAccount().setBalance("20000");
    assertEquals(main.exchange("30000", "USD"), 1);
}

@Test
public void testLoan() {
    main.insert("023456789");
    main.getAccount().setDept("30000");
    main.getAccount().setLimit("30000");
    assertEquals(main.loan("30000"), 7);
}

@Test
public void testTakeCharge() {
    main.insert("023456789");
    main.offer[0].readDatabase(main.getAccount());
    main.getAccount().setRate(1);
    main.takeCharge(main.getAccount());
    assertEquals(main.getAccount().getAccountNumber(), "23456789");
}

@Test
public void testDepositWithoutBank() {
    main.insert("023456789");
    assertEquals(main.depositWithoutBank("30000"), 2);
}

@Test
public void testCheckBalance() {
    main.insert("023456789");
    main.offer[0].readDatabase(main.getAccount());
    assertEquals(main.checkBalance(), main.getAccount().getLog());
}

@Test
public void testPayUtilityBill() {
    main.insert("023456789");
    main.offer[0].readDatabase(main.getAccount());
    Account account = new Account();
    main.offer[0].readDatabase(account);
    assertEquals(main.payUtilityBill(account),2);
}
```

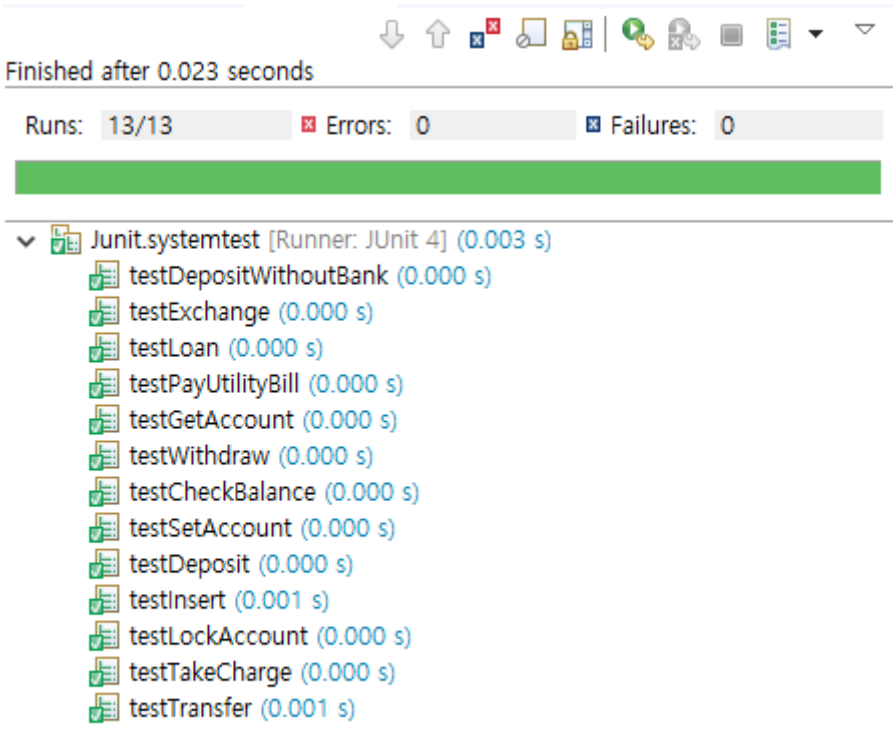
2060 Testing

Activity 2061. Unit Testing

1. File Test result



2. System Test Result



Activity 2063. System Testing

<OOPT Stage 2038, 2040 v3 참고>

Test #	Function	Use Case Name	Test	Test description	P/F
1	Deposit	Deposit	입금 TEST		
1.1				고객이 입력한 계좌에 입금한 금액만큼 입금이 되는지 TEST	p
1.2				계좌에 int범위보다 큰 값을 한번에 입력하면 입금이 안 되는지 test	p
1.3				계좌에 입금금액으로 음수 값/0을 입력하면 입금이 안 되는지 test	p
1.4				만원 단위로 입금되는지 TEST	p
1.5				고객이 올바른 계좌를 입력했을 때 입금이 가능한지 TEST	p
1.6				은행계좌가 아닌 신용카드정보를 입력하면 대출금이 있는 상태에서 대출금이 상환되는지 test	p
1.7				은행계좌가 아닌 신용카드정보를 입력하면 대출금이 없는 상태에서 입금이 되지 않는지 test	p
1.8				은행계좌가 아닌 신용카드정보를 입력하면 대출금보다 큰 금액이 상환되지 않는지 test	p
1.9				거래 불가능한 계좌에 입금이 안 되는지 test	p
1.10				만원 단위가 아니면 입금이 안 되는지 Test	p
1.11				만원 버튼이 잘 동작하는지 Test	P
1.12				10만원 버튼이 잘 동작하는지 test	P
1.13				100만원 버튼이 잘 동작하는지 test	p
1.14				올바르지 않은 계좌번호를 입력하면 입금이 불가능한지 test	p

1.15				입금 후 총 금액이 해당 데이터 타입의 범위를 넘어가게 되면 입금이 안 되는지 test	F
1.16				일정 시간이 지나면 초기화면으로 돌아가는지 test	F
1.17				공과금 납부계좌(한국은행)으로 입금할 수 없는지 Test	P
2	Deposit without Bank	Deposit without bank	무통장입금 Test	통장/ 체크카드 없이도 입금할 수 있는지 TEST	
2.1				ATM과 같은 은행에만 보낼 수 있는지 test	p
2.2				계좌 정보가 정확하지 않으면 안 보내지는지Test	P
2.3				만원 단위로 보내지는지 test	P
2.4				만원 단위가 아니면 안 보내지는지 Test	P
2.5				수신자의 계좌에 제대로 입금이 되었는지(DB가 업데이트 되었는지) test	P
2.6				거래 불가인 계좌에 무통장입금이 안 되는지 test	p
2.7				만원 버튼이 잘 동작하는지 Test	P
2.8				10만원 버튼이 잘 동작하는지 test	P
2.9				100만원 버튼이 잘 동작하는지 test	P
2.10				정확한 계좌 정보를 입력하면 입금이 잘 되는지 Test	P
2.11				음수 값을 넣으면 입금이 안 되는지 test	P
2.12				0을 넣으면 입금이 안 되는지 Test	P
2.13				은행계좌가 아닌 신용카드정보를 입력하면 무통장입금이 불가능한지 test	P
2.14				일정 시간이 지나면 초기화면으로 돌아가는지 test	F
2.15				공과금 납부계좌(한국은행)으로 입금할 수 없는지 Test	P
3	Withdraw	Withdraw	출금 TEST	출금이 제대로 동작하는지	

				Test	
3.1				고객이 올바른 계좌를 입력했을 때 인출이 가능한지 TEST	P
3.2				고객이 올바른 계좌를 입력하지 않았을 때 인출이 불가능한지 TEST	P
3.3				비밀번호가 맞아야 인출되는지 TEST	P
3.4				비밀번호가 맞지 않으면 인출이 되지 않는지 Test	P
3.5				만원 단위로 인출되는지 TEST	P
3.6				만원 단위가 아니면 인출이 안 되는지 Test	P
3.7				고객이 선택한 오만원권과 만원권의 장 수대로 출금되는지 test	P
3.8				오만원 이하의 출금금액을 입력했을 때, 오만원권으로 출금이 불가능한지 Test	P
3.9				계좌에 남은 잔고보다 큰 금액을 입력했을 때 출금이 불가능한지 test	P
3.10				거래불가계좌에서 출금이 불가능한지 test	P
3.11				음수 값을 넣으면 출금이 불가능한지 test	P
3.12				0을 입력하면 출금이 안 되는지 test	P
3.13				은행계좌가 아닌 신용카드정보를 입력하면 출금 메뉴 이용이 불가능한지 test	P
3.14				일정 시간이 지나면 초기화면으로 돌아가는지 test	F
3.15				비밀번호 오류를 낸 후 다시 제대로 비밀번호를 입력 한 후 출금이 제대로 되는지 test	P
3.16				ATM과 다른 은행의 계좌에서 출금 시 수수료가 포함되어 계좌 잔액이 줄어드는지 test	P
3.17				등급에 따라 수수료가 다르게 부과되는지 Test	P
3.18				일정 시간이 지나면 초기화면으로 돌아가는지 test	F

3.19				공과금 납부계좌(한국은행)으로 출금할 수 없는지 Test	P
4	Transfer	Transfer	송금 TEST	고객이 본인의 계좌에 있는 돈을 보낼 수 있는지 TEST	
4.1				고객의 계좌에서 송금하려는 금액만큼 인출되는지 test	P
4.2				수신자의 계좌에서 송금 받은 금액만큼 잔액이 추가되는지 test	P
4.3				계좌의 비밀번호가 맞으면 돈을 보내는지 TEST	P
4.4				계좌의 비밀번호가 틀리면 돈을 보내지 않는지 test	P
4.5				수신자의 계좌 정보가 틀리면 돈이 안 보내지는지 test	P
4.6				타행 이체 시 송금자의 계좌에서 송금하려는 금액 +수수료만큼 인출되는지 Test	F
4.7				송금자의 계좌 잔액보다 큰 금액은 송금이 안 되는지 test	P
4.8				타행이체의 경우 송금자의 계좌 잔액+수수료보다 큰 금액은 송금이 안 되는지 test	P
4.9				타행이체의 경우 수신자의 계좌에 송금시도한 금액만큼만(수수료 제외) 입금되는지 test	P
4.10				송신자의 계좌가 거래 불가 상태인 경우 송금이 안 되는지 test	P
4.11				수신자의 계좌가 거래 불가 상태인 경우 송금이 안 되는지 test	P
4.12				음수 값은 송금이 불가능한지 test	P
4.13				송금금액에 0을 넣으면 계좌 잔액에 변동이 없는지 Test	P
4.14				은행계좌가 아닌 신용카드정보를 입력하면 송금 진행이 불가능한지 test	P
4.15				자기 자신에게 송금이 안 되는지 Test	p
4.16				일정 시간이 지나면 초기화면으로 돌아가는지 test	F

4.17				등급에 따라 수수료가 다르게 부과되는지 Test	F
4.18				만원단위가 아니어도 송금이 되는지 test	P
4.19				공과금 납부계좌(한국은행)으로 송금할 수 없는지 Test	P
5	Exchange	Exchange	한화에서 외화로 환전 TEST	고객이 원하는 금액만큼 한화->외화로 환전이 가능한지 TEST	
5.1				고객이 올바른 계좌를 입력했을 때만 환전이 가능한지 TEST	P
5.2				비밀번호가 맞아야 환전이 되는지 TEST	P
5.3				고객의 계좌에서 환율을 적용한 금액만큼 인출되는지 TEST	P
5.4				계좌의 잔고보다 큰 금액만큼 환전이 불가능한지 test	P
5.5				비밀번호가 틀리면 환전이 되지 않는지 test	P
5.6				USD로 환전 시 수수료+금액만큼 제대로 인출 되는지 TEST	P
5.7				EUR로 환전 시 수수료+금액만큼 제대로 인출 되는지 TEST	P
5.8				JPY로 환전 시 수수료+금액만큼 제대로 인출 되는지 TEST	P
5.9				CNY로 환전 시 수수료+금액만큼 제대로 인출 되는지 TEST	P
5.10				계좌가 아닌 신용카드를 입력하면 환전이 진행되지 않는지 TEST	P
5.11				일정 시간이 지나면 초기화면으로 돌아가는지 test	F
5.12				만/10만/100만'원' 단위로 입력 할 수 없는지 test	P
5.13				등급에 따라 수수료가 다르게 부과되는지 Test	F
5.14				공과금 납부계좌(한국은행)으로 환전할 수 없는지 Test	P

6	Loan	Loan	대출 TEST	카드사를 통해 대출이 제대로 진행되는지 Test	
6.1				고객이 올바른 신용카드 번호를 입력했을 때 대출이 되는지 TEST	P
6.2				고객이 올바르지 않은 신용카드 번호를 입력했을 때 대출이 되지 않는지 test	P
6.3				신용카드 번호가 아닌 은행계좌를 입력하면 대출이 되지 않는지 test	p
6.4				만원 단위로만 인출되는지 TEST	P
6.5				한도를 넘지 않았을 경우에만 대출이 되는지 TEST	P
6.6				비밀번호가 맞을 경우에만 대출이 되는지 TEST	P
6.7				등급에 따라 수수료가 다르게 부과되는지 Test	F
6.8				카드사에서 대출 금액 + 수수료만큼 대출했다는 정보가 제대로 처리되는지 TEST	F
6.9				일정 시간이 지나면 초기화면으로 돌아가는지 test	F
6.10				공과금 납부계좌(한국은행)으로 대출할 수 없는지 Test	P
7	Pay Utility Bill	Pay Utility Bill	공과금 납부 TEST	공과금을 납부 가능한지 test	
7.1				고객이 올바른 계좌를 입력했을 때 공과금을 납부 할 수 있는지 TEST	P
7.2				공과금 계좌를 입력했을 때, 잔고가 충분한 경우 공과금 계좌에 입력되어 있는 금액만큼 납부가 되는지 TEST	P
7.3				고객의 계좌에 잔고가 충분하지 않으면 납부가 되지 않는지 Test	P
7.4				비밀번호가 맞아야 납부할 수 있는지 TEST	P
7.5				비밀번호가 맞지 않으면 납부가 안 되는지 Test	P
7.6				올바르지 않은 지로를 입력했	P

				을 때, 납부가 되지 않는지 test	
7.7				일정 시간이 지나면 초기화면으로 돌아가는지 test	F
7.8				공과금 계좌가 아닌 은행계좌를 입력하면 납부가 안 되는지 test	P
7.9				공과금 계좌가 아닌 신용카드 번호를 입력하면 납부가 안 되는지 test	P
7.10				신용카드로는 공과금 납부 진행이 안 되는지	P
8	Check Balance	Check Balance	계좌 조회 TEST	계좌의 최근 거래 내역(계좌의 경우 일시, 거래 종류, 거래 금액, 잔고, 신용카드의 경우 일시, 거래 종류, 대출 금액, 남은 한도)을 제대로 출력하는지 TEST	
8.1				고객이 올바른 계좌를 입력했을 때 계좌 조회를 할 수 있는지 TEST	P
8.2				고객이 올바르지 않은 계좌를 입력했을 때 계좌 조회를 할 수 없는지 TEST	P
8.3				비밀번호가 맞을 경우에 계좌 조회가 되는지 TEST	P
8.4				비밀번호가 맞지 않을 경우에 계좌 조회가 안 되는지 TEST	P
9	Check Validation	Check Validation	삽입된 매체의 유효성 TEST	- 삽입된 매체(계좌번호/신용카드/공과금 계좌)가 유효할 때만 프로세스가 진행되는지 TEST	
9.1				입금 시 올바른 계좌정보를 입력했을 때 입금이 진행되는지 test	P
9.2				입금 시 올바른 카드정보를 입력했을 때 상환이 진행되는지 test	P
9.3				입금 시 올바르지 않은 계좌정보를 입력했을 때 입금이 진행되지 않는지 test	P
9.4				입금 시 올바르지 않은 카드정보를 입력했을 때 입금이	P

				진행되지 않는지 test	
9.5				송금 시 올바른 계좌정보를 입력했을 때 송금이 진행되는지 test	P
9.6				송금 시 올바르지 않은 계좌정보를 입력했을 때 송금이 진행되지 않는지 test	P
9.7				출금 시 올바른 계좌정보를 입력했을 때 출금이 진행되는지 test	p
9.8				출금 시 올바르지 않은 계좌정보를 입력했을 때 출금이 진행되지 않는지 test	P
9.9				대출 시 올바른 신용카드정보를 입력했을 때 대출이 진행되는지 test	P
9.10				대출 시 올바르지 않은 카드정보를 입력했을 때 대출이 진행되지 않는지 test	P
9.11				환전 시 올바른 계좌정보를 입력했을 때 환전이 진행되는지 test	P
9.12				환전 시 올바르지 않은 계좌정보를 입력했을 때 환전이 진행되지 않는지 test	P
9.13				무통장입금 시 올바른 계좌정보를 입력했을 때 입금이 진행되는지 test	P
9.14				무통장입금 시 올바르지 않은 계좌정보를 입력했을 때 입금이 진행되지 않는지 test	P
9.15				거래내역 조회 시 올바른 계좌정보를 입력했을 때 거래내역조회가 진행되는지 test	P
9.16				거래내역 조회 시 올바르지 않은 계좌정보를 입력했을 때 거래내역 조회가 진행되지 않는지 test	P
9.17				공과금 납부 시 올바른 계좌정보를 입력했을 때 납부가 진행되는지 test	P
9.18				공과금 납부 시 올바르지 않은 계좌정보를 입력했을 때	P

				납부가 진행되지 않는지 test	
10	Print Transaction Receipt	Print Transaction Receipt	거래명세서 출력 TEST	- 각각의 프로세스가 끝난 후 거래명세서가 올바르게 출력되는지 TEST	
10.1				입금 완료 후 거래명세서가 올바르게 출력되는지 Test	P
10.2				송금 완료 후 거래명세서가 올바르게 출력되는지 Test	P
10.3				출금 완료 후 거래명세서가 올바르게 출력되는지 Test	P
10.4				무통장 입금 완료 후 거래명세서가 올바르게 출력되는지 Test	P
10.5				환전 완료 후 거래명세서가 올바르게 출력되는지 Test	P
10.6				대출 후 거래명세서가 올바르게 출력되는지 Test	P
10.7				공과금 납부 후 거래명세서가 올바르게 출력되는지 Test	P
11	Print Error	Print Error	에러 출력 TEST	- 각종 에러들이 발생했을 경우 그에 알맞은 에러 메세지를 출력하는지 TEST	
11.1				존재하지 않는 계좌번호를 입력한 경우 에러 메세지를 출력하는지 TEST	P
11.2				존재하지 않는 카드번호를 입력한 경우 에러 메세지를 출력하는지 TEST	P
11.3				비밀번호를 잘못 입력한 경우 에러 메시지를 출력하는지 TEST	p
11.4				잔고 이상의 금액을 거래한 경우 에러 메시지를 출력하는지 TEST	P
11.5				한도가 초과된 경우 에러 메시지를 출력하는지 TEST	P
11.6				거래가 정지된 계좌의 경우 에러 메시지를 출력하는지 TEST	P
11.7				출금 시 10000원단위가 아닌 금액을 인출 시도 시 제대로 된 에러 메시지를 출력하는지	p

				test	
12	Do Forced Termination	Do Forced Termination	강제 종료 TEST	- 오류가 3번 발생했을 경우 강제 종료가 되는지(처음 화면으로 돌아가는지) TEST	
12.1				오류가 3번 발생했을 경우 처음 화면으로 돌아가는지 TEST	F
13	Take Charge	Take Charge	수수료 부과 TEST	- 송금(타행의 경우)/출금/대출/환전 시 수수료가 붙는지 TEST	
13.1				송금(타행으로)할 때 수수료가 부과되는지 test	P
13.2				송금(타행으로)할 때 등급에 따라 수수료가 부과되는지 test	P
13.3				대출할 때 수수료가 부과되는지 test	F
13.4				대출할 때 등급에 따라 수수료가 부과되는지 test	F
13.5				환전할 때 수수료가 부과되는지 test	P
13.6				환전할 때 등급에 따라 수수료가 부과되는지 test	F
13.7				ATM과 다른 은행 계좌에서 출금 시 수수료가 부과되는지 test	P
13.8				ATM과 같은 은행 계좌에서 출금 시 수수료가 부과되지 않는지 test	p
14	Check Password	Check Password	비밀번호 확인 TEST	각 프로세스에서 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	
14.1				출금 시 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	P
14.2				송금 시 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	p
14.3				환전 시 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	P
14.4				대출 시 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	P

14.5				공과금납부 시 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	P
14.6				계좌조회 시 올바른 비밀번호를 입력했을 때만 프로세스가 진행되는지 TEST	p
15	Transaction Lock	Transaction Lock	거래 잠금 TEST	- 비밀번호를 3번 틀렸을 시 해당 계좌/카드로 거래를 할 수 없도록 계좌/카드를 잠그는지 TEST	
15.1				출금 시 비밀번호를 3번 틀렸을 시 계좌를 잠그는지 TEST	P
15.2				송금 시 비밀번호를 3번 틀렸을 시 계좌를 잠그는지 TEST	P
15.3				환전 시 비밀번호를 3번 틀렸을 시 계좌를 잠그는지 TEST	P
15.4				대출시 비밀번호를 3번 틀렸을 시 신용카드를 잠그는지 TEST	P
15.5				공과금납부 시 비밀번호를 3번 틀렸을 시 계좌를 잠그는지 TEST	P
15.6				계좌조회 시 비밀번호를 3번 틀렸을 시 계좌를 잠그는지 TEST	P
16	Update Database	Update Database	DB 업데이트 TEST	계좌 조회를 제외한 각 프로세스에서, 프로세스가 끝난 후 DB가 제대로 업데이트 되는지 TEST	
16.1				입금 : (계좌번호 이용 시) 거래 일시/입금 금액/잔고, 혹은 (신용카드 이용 시) 거래 일시/입금 금액/남은 대출 금액이 업데이트 되는지 TEST	P
16.2				공과금 납부: 거래 일시/납부 금액/잔고가 업데이트 되는지 TEST	P
16.3				출금 : 거래 일시/출금 금액/남은 잔고가 업데이트 되는지 TEST	P
16.4				대출 : 거래 일시/대출 금액/남은 한도가 업데이트 되는지 TEST	P

16.5				환전 : 거래 일시/환전 금액 + 수수료/잔고가 업데이트 되는지 TEST	P
16.6				송금 : (송금한 계좌의 경우)거래 일시/송금 금액 + (타행일 경우) 수수료/송금 계좌/잔고, (송금 받은 계좌의 경우)거래 일시/입금 금액/송금한 계좌/잔고가 업데이트 되는지 TEST	P
16.7				무통장입금: 거래 일시/입금 금액/잔고가 업데이트 되는지 TEST	P