Software Modeling

A NEW RELIABLE ATM

OOPT STAGE 2050

201411140 권성완
201511247 김선정
201510436 허윤아
201510285 조수빈

CONTENTS

Class & method Definition

Type	Class
Name	MainSystem
Purpose	유저의 입력과 Offer,Account를 연결하는 클래스
Overview(Class)	Insert(), inputPassword(), printRecentTransaction(), printTransactionReceipt(), deposit(), withdraw(), transfer(), exchange(), loan(), doForcedTermination(), printError() takeCharge() depositWithoutBank() checkBalance(), payUtilityBill()
Cross Reference	Usecase: R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13,R15 Functional: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,R5.1
Input(Method)	-
Output(Method)	-
Abstract operation(Method)	
Exceptional Courses of Events	-

Class & method Definition

Type	Method
Name	Insert()
Purpose	유저의 매체 입력을 받는다.
Cross Reference	Usecase: 9 Functional :R2.1
Input(Method)	-
Output(Method)	void
Abstract operation(Method)	유저의 통장/카드번호와 지로번호를 입력받는다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	inputPassword()
Purpose	ATM의 기능을 이용하기 위한 password를 입력받아 withdraw등에 사용할 수 있게 한다.
Cross Reference	Usecase: 3, 4, 5, 6, 7, 8, 12, 14 Functional: R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R2.4, R4.1,
Input(Method)	
Output(Method)	int password
Abstract operation(Method)	유저가 비밀번호를 입력하고 이를 저장한다
Exceptional Courses of Events	-

Type	Method
Name	printTransactionRecepit()
Purpose	거래명세서를 출력한다..
Cross Reference	Usecase: 10 Functional: R2.2
Input(Method)	
Output(Method)	void
Abstract operation(Method)	유저가 비밀번호를 입력하고 이를 저장한다
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	deposit()
Purpose	예금한다.
Cross Reference	Usecase: 1 Functional: R1.1
Input(Method)	
Output(Method)	void
Abstract operation(Method)	유저가 입금할 금액을 입력하고 이를 account에 저장한다.
Exceptional Courses of Events	-

Type	Method
Name	depositWithoutBank()
Purpose	무통장 입금을 한다.
Cross Reference	Usecase: 2 Functional: R1.2
Input(Method)	-
Output(Method)	void
Abstract operation(Method)	통장이나 카드가 없어도 송금을 한다. 송금받은 계좌의 변동사항을 account에 저장한다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	withdraw()
Purpose	출금한다
Cross Reference	Usecase: 3 Functional: R1.3
Input(Method)	
Output(Method)	void
Abstract operation(Method)	유저가 출금할 금액을 입력하고 그 금액만큼의 변동사항을account에 저장한다.
Exceptional Courses of Events	-

Type	Method
Name	transfer()
Purpose	송금한다
Cross Reference	Usecase: 4 Functional: R1.4
Input(Method)	
Output(Method)	void
Abstract operation(Method)	유저가 송금할 금액을 입력하고 그 금액만큼의 변동사항을 유저의 account와 송금 받는 사람의 account에 저장한다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	exchange()
Purpose	환전한다
Cross Reference	Usecase: 5 Functional: R1.5
Input(Method)	
Output(Method)	void
Abstract operation(Method)	유저가 환전할 금액을 입력하고 그 금액만큼의 환전 후 변동사항을 account에 저장한다.
Exceptional Courses of Events	-

Type	Method
Name	loan()
Purpose	대출한다
Cross Reference	Usecase: 6 Functional: R1.6
Input(Method)	
Output(Method)	Void
Abstract operation(Method)	유저가 대출할 금액을 입력하고 그 금액만큼의 변동사항을account에 저장한다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	doForcedTermination()
Purpose	초기화면으로 돌아간다
Cross Reference	Usecase: 12 Functional: R 2.4
Input(Method)	
Output(Method)	void
Abstract operation(Method)	에러가 났을 경우나 선택한 기능이 끝난 경우 초기화면으로 돌아간다.
Exceptional Courses of Events	-

Type	Method
Name	printError(int errorType)
Purpose	에러를 출력한다
Cross Reference	Usecase: 11 Functional: R2.3
Input(Method)	Int errorType
Output(Method)	void
Abstract operation(Method)	errorType(error code)에 따라 맞는 에러를 출력한다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	takeCharge()
Purpose	수수료를 부과한다
Cross Reference	Usecase: 13 Functional: R3.1
Input(Method)	
Output(Method)	void
Abstract operation(Method)	수수료를 등급에 맞게 계산하여 부과하고 그에 따른 변동사항을 계좌에 저장한다.
Exceptional Courses of Events	-

Type	Method
Name	payUtilityBill()
Purpose	공과금을 수납한다
Cross Reference	Usecase: 7 Functional: R1.7
Input(Method)	
Output(Method)	void
Abstract operation(Method)	국가계좌로 공과금을 납부한다
Exceptional Courses of Events	-

Class & Method Definition

Type	Class
Name	Account
Purpose	MainSystem과 offer의 Database(test file)사이의 정보를 연결하는 클래스. MainSystem은 Account클래스의 정보를 변경하고, 이 Account클래스의 정보를 offer의 Database에 저장한다.
Overview(Class)	getPassword(), getLimit(), getBalance(), getIsLocked(), getBank(), getRate(), getDept(), setLimit(int limit), setBalance(int balance), setDept(int dept), setIsLocked(Boolean isLocked), setRate(int rate), setBank(int bank), setPassword(int password), checkPassword(int password)
Cross Reference	Usecase: 12, 14, 16, 17 Functional: R2.4, R4.1, R6.1, R6.2
Input(Method)	-
Output(Method)	-
Abstract operation(Method)	
Exceptional Courses of Events	-

Type	Method
Name	getPassword()
Purpose	Account의 password를 가져온다.
Cross Reference	Usecase: 14, 16,17 Functional: R4.1. R6.1, R6.2
Input(Method)	-
Output(Method)	int password
Abstract operation(Method)	account의 password를 읽고 리턴한다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	getLimit()
Purpose	Account의 limit를 가져온다.
Cross Reference	Usecase: 13, 16,17 Functional: R3.1 R6.1, R6.2
Input(Method)	-
Output(Method)	int limit
Abstract operation(Method)	account의 대출 한도를 리턴한다.
Exceptional Courses of Events	-

Type	Method
Name	getBalance()
Purpose	Account의 balance를 가져온다.
Cross Reference	Usecase: 8,16,17 Functional: R1.8, R6.1, R6.2
Input(Method)	-
Output(Method)	int balance
Abstract operation(Method)	account의 남은 잔액을 리턴한다.
Exceptional Courses of Events	-

Class & Method Definition

Type	Method
Name	getIsLocked()
Purpose	Account의 isLocked를 가져온다.
Cross Reference	Usecase: 15, 16,17 Functional: R5.1, R6.1, R6.2
Input(Method)	-
Output(Method)	Boolean isLocked
Abstract operation(Method)	account가 잠긴 상태인지 리턴한다.
Exceptional Courses of Events	-

Type	Method
Name	checkPasword(int password)
Purpose	Account의 password와 입력한 password가 일치하는지 확인한다.
Cross Reference	Usecase: 14, 16,17 Functional: R4.1. R6.1, R6.2
Input(Method)	int password
Output(Method)	boolean
Abstract operation(Method)	withdraw같은 과정 중에 유저가 입력한 password와 account에 설정된 password가 일치하는지 확인하고 true/false를 리턴한다.
Exceptional Courses of Events	-

Class & Method Definition

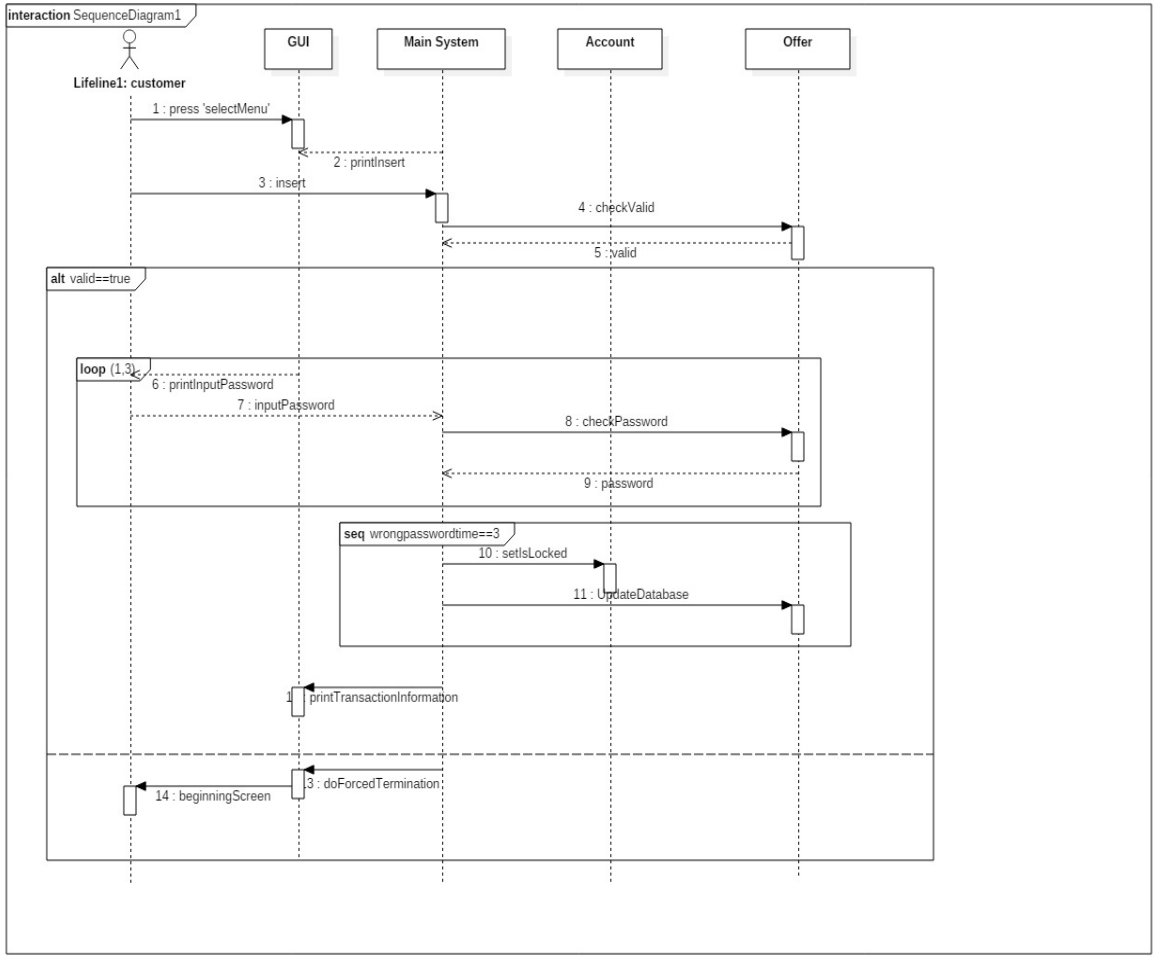
Type	Class
Name	Offer
Purpose	Database(text file)을 관리하고 atm(main system)에 정보를 넘겨주는 클래스
Overview(Class)	Offer(int company), updateDatabase(Account account), checkValid(Account account), readDatabase(Account account)
Cross Reference	Usecase: Functional:
Input(Method)	-
Output(Method)	-
Abstract operation(Method)	
Exceptional Courses of Events	-

Type	Method
Name	updateDatabase(Account account)
Purpose	Database를 업데이트한다
Cross Reference	Usecase: 17 Functional: R6.2
Input(Method)	Account account
Output(Method)	Int
Abstract operation(Method)	Database를 account객체의 정보와 같게 저장하고, error code를 int로 반환한다.
Exceptional Courses of Events	-

Class & Method Definition

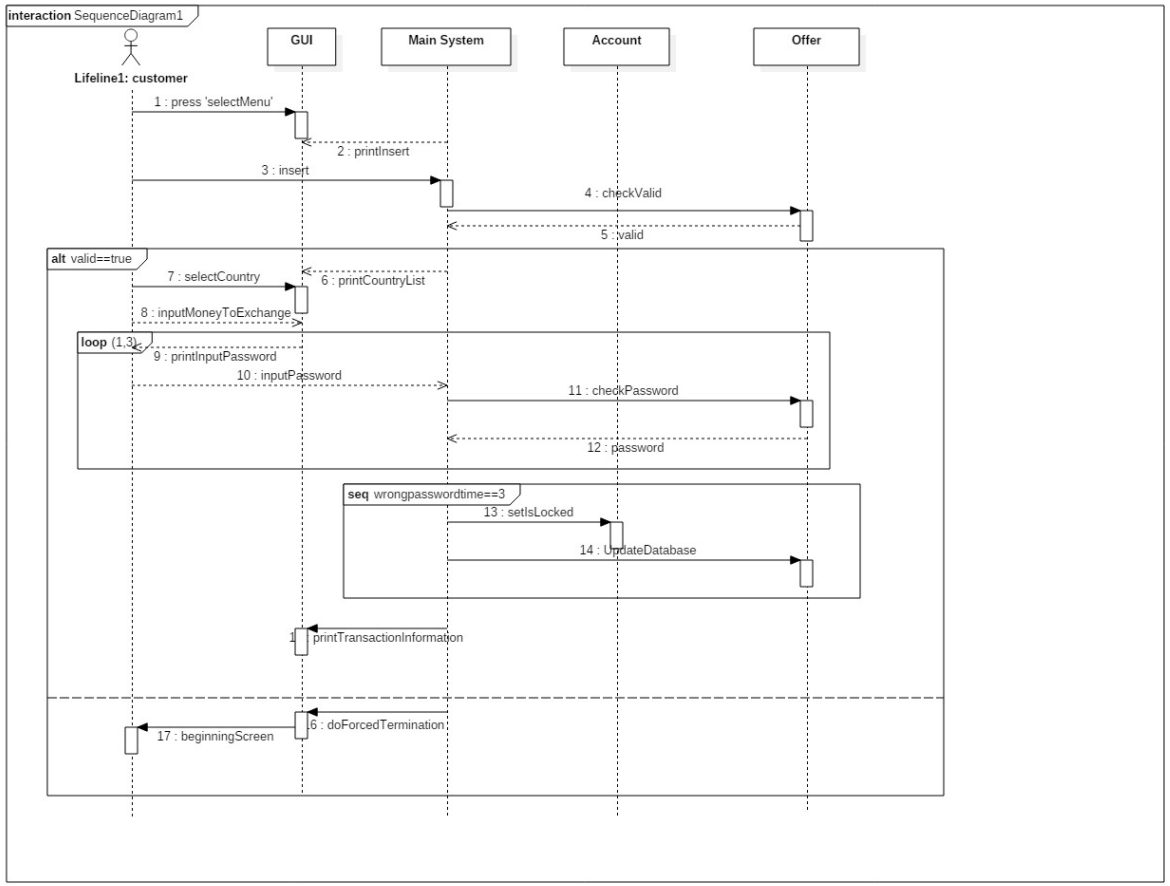
15	Method
Name	readDatabase(Account account)
Purpose	Database(txt)파일의 정보를 읽어온다.
Cross Reference	Usecase: 17 Functional: R6.2
Input(Method)	Account account
Output(Method)	Int
Abstract operation(Method)	Database(txt)파일의 정보를 읽어오고 error code를 int로 반환한다.
Exceptional Courses of Events	-

Implements Window



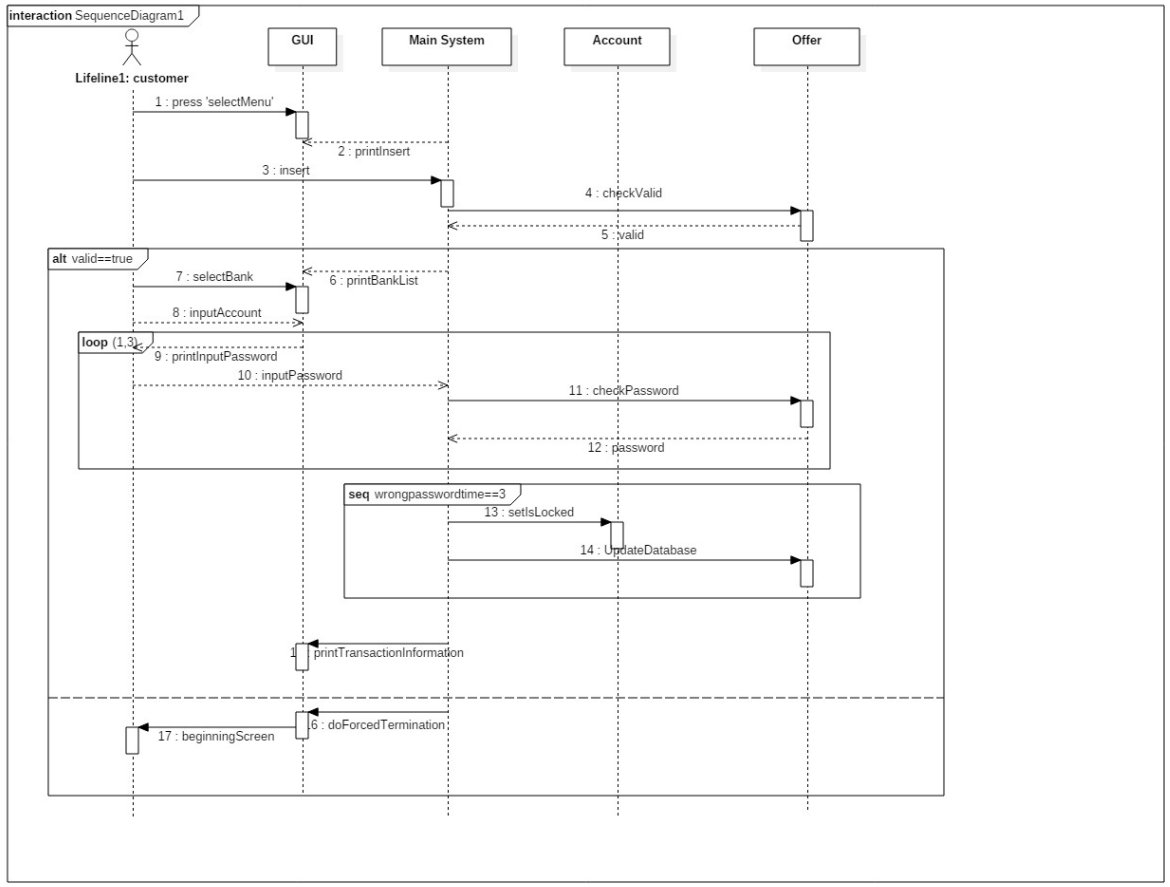
Name	selectMenu
Resposibilities	메뉴에서 원하는 목적의 버튼을 누른다
Type	GUI
Cross Reference	R1.1,R1.2,R1.3,R1.4,R1.5,R1.6,R1.7,R1.8
Notes	버튼을 누른다
Pre-Conditions	
Post-Conditions	다음 화면으로 이동한다.

Implements Window



Name	selectCountry
Resposibilities	환전을 원하는 국가를 선택한다.
Type	GUI
Cross Reference	Exchange
Notes	R1.5
Pre-Conditions	Insert한 카드나 통장이 valid해야한다.
Post-Conditions	다음 화면으로 이동한다.

Implements Window



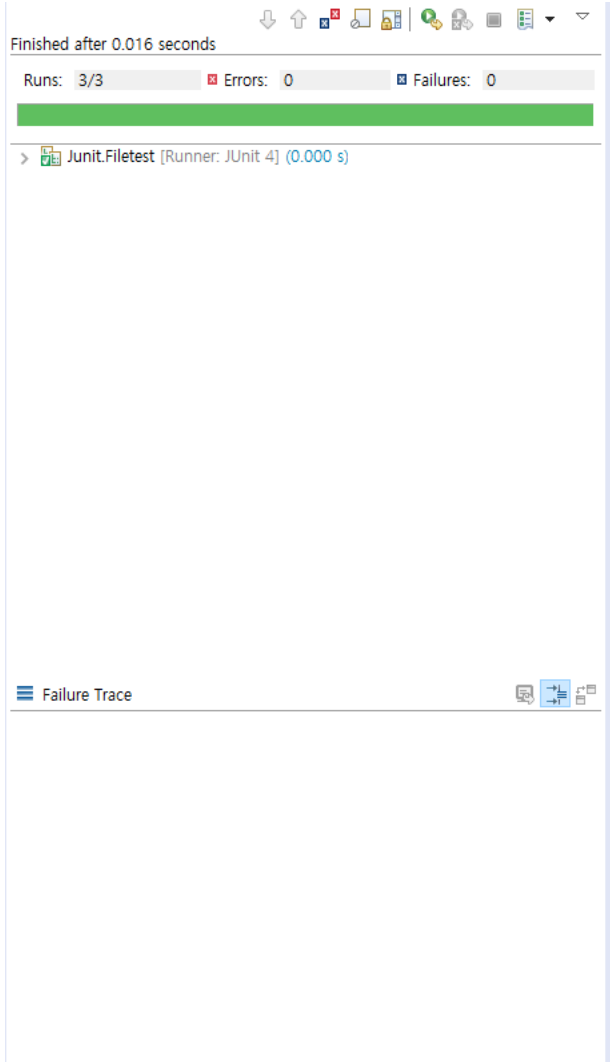
Name	selectBank
Resposibilities	메뉴에서 목적 은행 버튼을 누른다.
Type	GUI
Cross Reference	R1.4
Notes	버튼을 누른다
Pre-Conditions	Insert한 카드나 통장이 valid해야한다.
Post-Conditions	다음 화면으로 이동한다.

Unit System Code

```java
import static org.junit.Assert.*;

public class Filetest {

    @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("권성완");
        testaccount.setPassword(4752);
        testaccount.setRate(3);
        testaccount.setLog("");
        assertEquals(true, test.updateDatabase(testaccount));
    }
```

```java
    @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());
    }

}
```

Unit Testing




```java
package Junit;

import static org.junit.Assert.*;

public class Filetest {

    @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("권성완");
        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());
    }

}
```