# Bank Account System Sequence Diagram

Due date: 07 Oct 2016

# Assumptions

- Open Account Sequence
  - Checking if customer is registered
  - Customer gives the type of account too open (Chequing or Savings)
  - Security Manager interacts with customer
- Replace Card
  - Assume that the teller validates the card visually.
- Authenticate
  - We made a variable for use that will be returned letting you know if the customer is authenticated or not, based on the provided bank card and pin.
  - getBankCard gets the bank card in the system based off of the number on the bank card they provided.
  - validatePIN returns true or false on whether the PIN they entered matches the one associated with the bank card.
- Deposit
  - a transaction is created and added to the account that was found that belongs to the customer.
- Find Account
  - if no number is found, null is returned by default because the account does not exist.
- Register Customer
- Withdraw

**sd** Authenticate

: Customer

: SecurityManager

foundBankCard : BankCard

provideCard(bankCard)

cardNum = readCardNumber(bankCard)

authenticated = false

loop : min1, max 3  [authenticated == false]

promptForPin()

alt          [user chooses cancel]

break          [user chose cancel]
<<reply>>
cancel()

returnCard()

[user inputs password]

foundBankCard = getBankCard(cardNum)

PIN

break          [foundBankCard == null]

<<reply>>
error()

validate = validatePIN(PIN)

alt          [validate == true]

authenticated = true

[validate == false]

<<reply>>
error()

opt          [authenticated == false && foundBankCard != null]

deactiveCard()

<<reply>>
authenticated()

**sd** Deposit(amount)

: Customer         : AccountsManager         foundAccount : Account

The use case begins
when the customer
selects "Deposit"

Deposit(accountNumber, amount)

**ref**

FindAccount(accountNumber)

**alt**

<<reply>>
error()

[foundAccount == null]

[foundAccount != null]

<<create>>
tempTransaction = createTransaction(1, currentdate, amount)

newTransaction : Transaction

addTransaction(tempTransaction)

success

**sd** FindAccount(accountNumber)

: AccountsManager

Accounts

foundAccount : Account

findAccount(accountNumber)

loop  [foreach foundAccount in Accounts]

theNumber = getAccountNumber()

break

[accountNumber == theNumber]

<<reply>>
foundAccount()

null

**sd** OpenAccount

: Teller  
: SecurityManager  
theCustomer : Customer  
: AccountsManager  

Use case begins when teller selects "Open Account"

authenticateCust(theCustomer)

registered = isRegistered(theCustomer)

**alt**

[registered == false]

**ref**

RegisterCustomer(theCustomer)

[else]

**ref**

Authenticate

**break**    [authenticated == false]

<<reply>>
error()

openAccount(acctType)

<<create>>
Account(acctType)

new : Account

new

associateAccount(new)

**sd** RegisterCustomer

: Teller

: SecurityManager

1. The use case begins when "Register Customer" is
choosen
2. The teller collects customer information (firstname,
lastname, phone number, facsimile of photo id) and a
physical bank card with a card number on it

RegisterCustomer(firstname, lastname, phone number, facsimile, cardNumber)

<<create>>
newCustomer = CreateCustomer(firstname, lastname, phone number, facsimile)

newCustomer : Customer

<<create>>
newBankCard = NewBankCard(cardNumber)

newBankCard : BankCard

addCard(newBankCard)

success

**sd** ReplaceCardSequence

: Customer | : Teller | : AccountsManager | cust : Customer | foundAccount : Account | existingBankCard : BankCard

The use case begins when the teller
selects "Replace Card"

The teller gathers the account number
from the Customer

ReplaceCard(AccountNumber)

**ref**

FindAccount(accountNumber)

PresentPhotoId()

validatePhotoID()

**break**
[PhotoID is invalid]

<<reply>>
CancelInteraction()

provideCardNo(cardNumber)

<<create>>
newCard = CreateMessage(cardNumber)

newCard : BankCard

existingBankCard = getMyCard()

**opt**
[existingBankCard <> Null]

existingPIN = getPIN()

setPin(existingPIN)

<<destroy>>
DestroyCard()

cust = getCustomer()

setBankCard(newCard)

**sd** Withdraw

: Customer

: AccountsManager

foundAccount : Account

Customer selects
withdraw and provides
account number and
amount

Withdraw(accountNumber, amount)()

**ref**

FindAccount(accountNumber)

**break**

[foundAccount == null]

<<reply>>
error()

balance = getBalance()

**break**

[balance < amount]

<<reply>>
error()

<<create>>
tempTransaction = createTransaction(2,currentDate,amount)

newTransaction : Transaction

addTransaction(tempTransaction)

dispenseAmount