MONTANA
STATE
BANK

CS 3500 – Software Engineering

ATM Machine

Kefin Sajan & Jesus Toxtle

Table of Contents

Introduction

Montana Savings Bank has recently been recommended the use of Automated Teller Machines to cut costs, eliminate errors, and provide a 24-hour service to its customers. The bank, and their consulting company have drafted a set of requirements they need to be programmed into their ATM. Our team was given was given the requirements and was told exactly what needed to be done but was also given some freedom to add features that would make the ATM better for the bank and for the customer.

The Use case diagram and descriptions were the first operations to be done for the project. After carefully reading the requirements given to us by the bank, we drafted the Use cases, the Use case diagram, and the Use cases' dialogs. This eased our way into step two of the process where we were to analyze and design how our project will work. A presentation was given to the class describing the details of our specific project, and even a prototype User Interface was designed. During the presentation we presented our Processing Use Case that was created during our Requirements phase. For part three of the projected we completed all of the communication diagrams for all of our Use cases, the class diagrams, and the class descriptions. Finally, after analyzing and designing our project, classes could easily be implemented using the C++ programming language. Our Project was completed and delivered on 12/18/19.

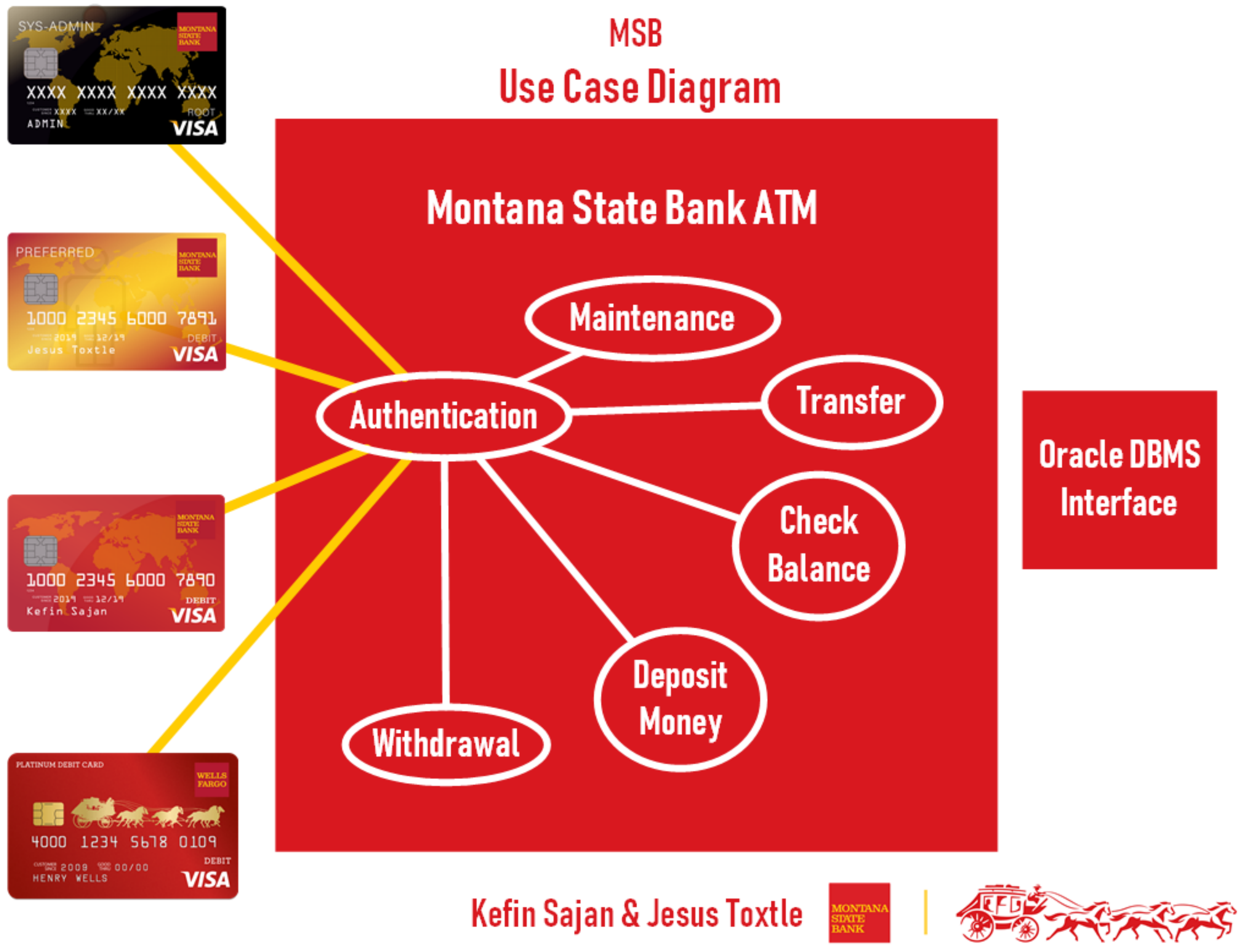
Who are we?

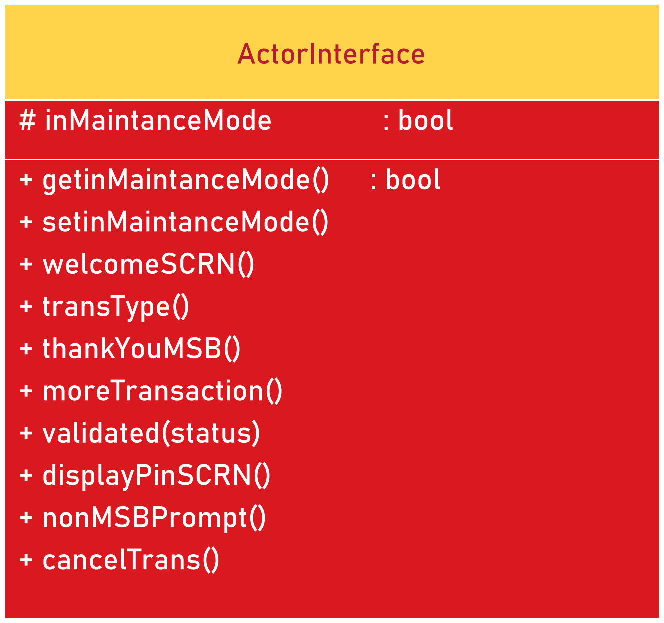
Our Team contained two developers working side by side to complete the project. Jesus Toxtle and Kefin Sajan chose to work together to build the given project over the course of the semester.

Every part of the project was carefully thought out and both members were on the same page before moving on. No conflicts did arise and working together went smoothly, and efficiently. The team developed most of the specifics of the design of the project together, and split the work that was done individually, equally.

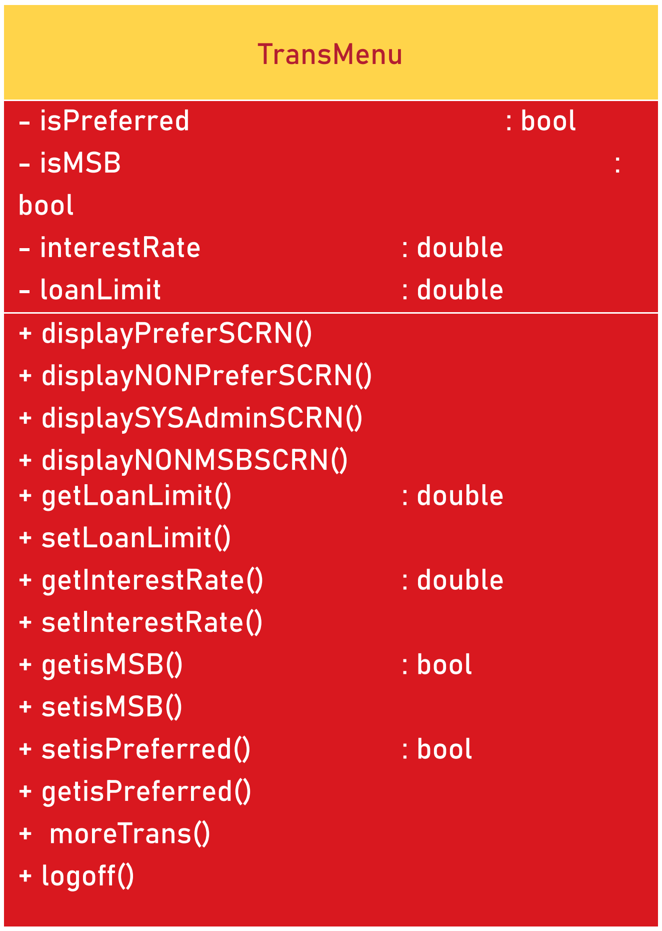
Overall no issues came up during the project. Some simple mistakes were easily fixed, and all projects were completed effectively and on time. Conflicting ideas were often settled easily as both members had similar ideas for the way things should work.

Use Case Diagram





The ActorInterface is the class which is focused on the user interactions. This class has the welcomeSCRN() which is shown to the user before any user interaction. If inMaintanceMode is true, then ATM machine is disabled for any non-System Administrator interaction. This can be triggered by other parts of the ATM system to prevent any causalities from happening. The transType() shows any type of transaction that a user would like to perform such as deposit, withdrawal, check balance, or transfer and records it by sending it to the Oracle Database. The thankYouMSB() is a message that appears whenever a user is done using the ATM and wants to confirm that their transaction was processed successfully. The moreTransaction() activates when a user is about to finish processing their transaction and is prompted if they would like to perform another one after their current one. It will then prompt for successful authentication by asking for their pin. The validated(status) is activated and displayed to the screen when a user has inputted their pin and based on their status, it will appear on the ATM if they're MSB preferred, non-preferred, system administrator, or a NON-MSB customer. The displayPINSCRN() is activated before validated(status) because the ATM will ask for a user's pin after inserting their card into the ATM to authenticate their identity. Overall, the ActorInterface will operations that correspond with every user to system interaction.



The TransMenu is the class which is focused on identifying the type of actor which is using the machine itself to enable the appropriate features correlating to a bank account. This class has the displayPreferSCRN() which is shown to the actor who is categorized as preferred according to the MSB database. This class has the displayNONPreferSCRN () which is shown to the actor who is categorized as non-preferred according to the MSB database. This class has the displaySYSAdminSCRN () which is shown to the actor who is categorized as the system administrator according to the MSB database. This class has the displayNONMSBSCRN () which is shown to the actor who is categorized as someone using an outside bank account according to the MSB database. The actor specified interest rate as well as loan limit is stored based on per actor basis. To summarize, the TransMenu class will display various type of screen based of validation status of any of the users and also hold important operations such as the interest rate and limits on loans.

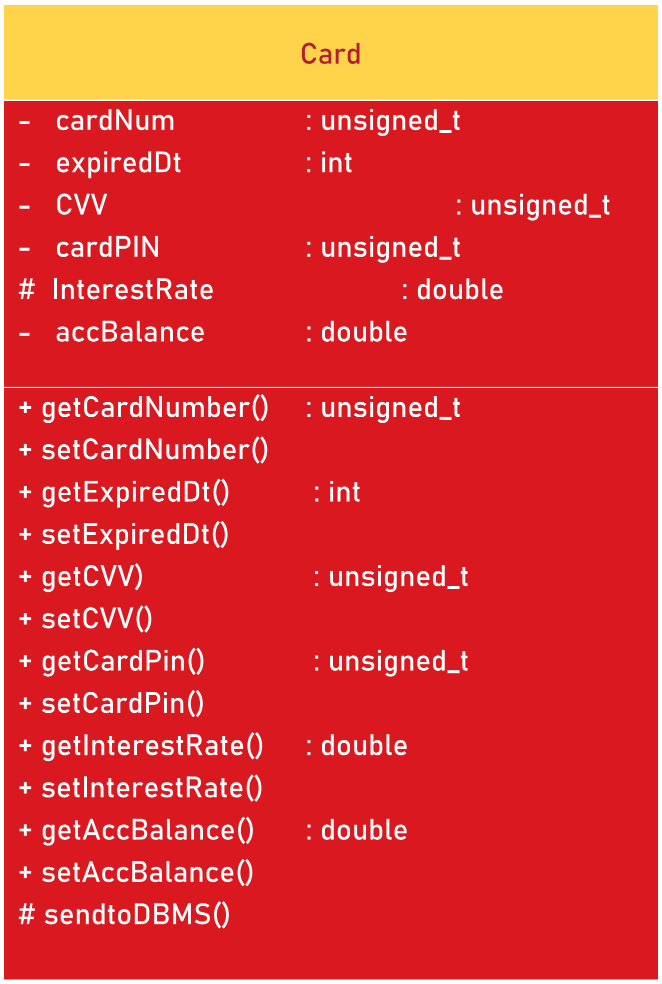
Transaction	
- isValidAcct	: bool
- transTime	: int
- transDate	: int
- transID	: int
- transAccNum	: int
- nearestBranch	: string
- isAuthenticated	: bool
- validedCurrency	: bool
- ATMLocation	: string
- accountBal	:double
- checkingBal	:double
- savingsBal	:double
- moneyMktBal	:double
- consLoanBal	:double
- mortgageBal	
- limitofWithdraw	
- confirmTrans	:bool
- destAccNum	
- fromAccNum	
+ sendtoDBMS()	
+ getValidAcct()	: bool
+ setValidAcct()	
+ getTransTime()	: int
+ setTransTime ()	
+ getTransDT()	: int
+ setTransDT()	
+ getTransAccNum()	: int
+ setTransAccNum()	
+ getTransID() : int	
+ setTransID()	
+ generateTransID()	
+ sendtoDBMS()	
+ captureNONMSBAgree()	
+ validAcct()	
+ captureTrans()	
+ validPartner()	
+ getAccountBal()	

```
+ setAccountBal()
+ getCheckingBal()
+ setCheckingBal()
+ getSavingsBal()
+ setSavingsBal()
+ setMoneyMktBal()
+ getMoneyMktBal()
+ setConsLoanBal()
+ getConsLoanBal()
+ setConsLoanBal()
+ getdestAccNum()
+ setdestAccNum()
+ getFromAccNum()
+ setFromAccNum()
+ addtoChecking()
+ addtoSavings()
+ addtoConsLoan()
+ addtoMortgage()
+ addtoMoneyMkt()
+ rmfromChecking()
+ rmfromSavings()
+ rmfromMoneyMkt()
+ rmfromConsLoan()
+ rmfromMortgage()
+ confirmSCRN()
+ cancelBtn()
+ insertBills()
+ depositSCRN()
# withdrawalSCRN()
+ ckbSCRN()
+ transferSCRN()
+ setConfirmTrans()
+ getConfirmTrans()
+ transError()
+ transferDestAcct()
+ transferFromAcct()
+ getlimitofWithdraw
```

The Transaction class is focused on cataloging transactions done at a particular ATM machine. This class stores if the customer account is valid, a timestamp, an identification number for organization, account number associated with the transaction, nearest branch located to the ATM machine, where the ATM is located, if the actor account was ever activated and if any currency is counterfeit. The sendtoDBMS is a function made to connect and store this class information with bank database. To summarize, the Transaction class will contain all of the types of transactions being performed at any MSB ATM and in the background record every single one of these transactions and send it to the Oracle database

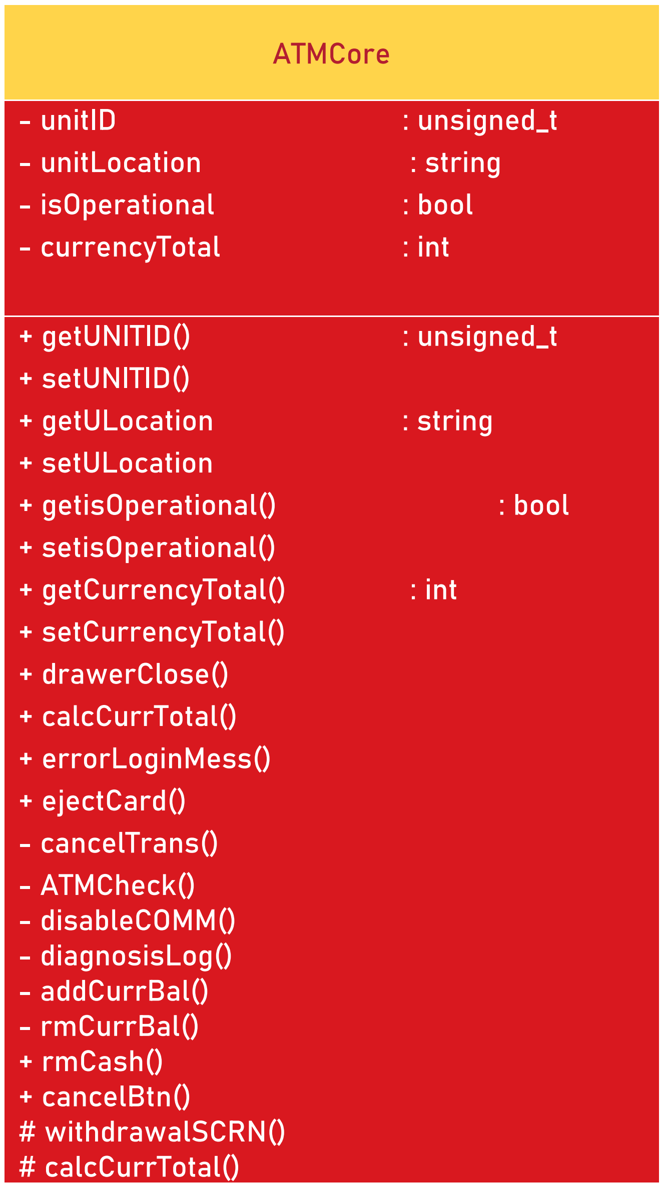
Actor	
- FName	: string
- MName	: string
- LName	: string
- Email	: string
# isPrefered	: bool
# isMSB	: bool
# restrictions	: int
+ nonMSBAgree	: bool
- isAdmin	: bool
+ getFName()	: string
+ setFName()	
+ getMName()	: string
+ setMName()	
+ getLName()	: string
+ setLName()	
+ getPrefered()	: bool
+ setPrefered()	
+ getisMSB()	: bool
+ setisMSB()	
+ getRestrictions()	: int
+ setRestrictions()	
+ getAdmin()	: bool
+ setAdmin()	
+ getEmail()	: string
+ setEmail()	
# validAcct()	
+ validInsertCard()	
# sendtoDBMS()	
+ nonMSBPrompt()	
+ moneyCheck()	
+ acctFault()	

The Actor class is focused on the particular ATM machine with the MSB bank. This class stores information regarding identifying the actor. This includes first name, middle name, last name, email, if the actor is preferred, whether the actor is using an account outside of MSB, if the actor is an Admin, if there are any restrictions placed on the account and whether if the user agrees regarding the three dollar fee if they are using an account outside of MSB. This class also checks if the account is properly validated and sends to the database all the information. Finally, the Actor class will store vital information regarding any user that's using the ATM and record their information along with their transaction and send it to the Oracle Database.

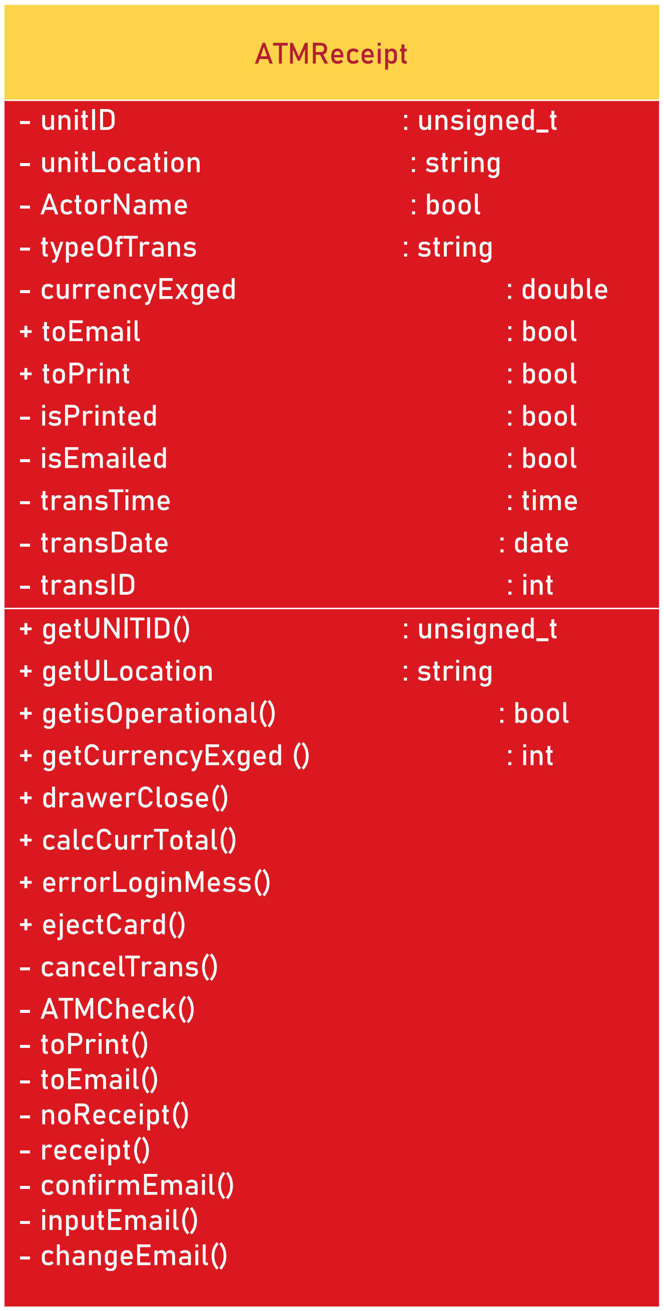


```
Card

-  cardNum          : unsigned_t
-  expiredDt        : int
-  CVV                              : unsigned_t
-  cardPIN          : unsigned_t
#  InterestRate             : double
-  accBalance       : double

+ getCardNumber()   : unsigned_t
+ setCardNumber()
+ getExpiredDt()     : int
+ setExpiredDt()
+ getCVV)            : unsigned_t
+ setCVV()
+ getCardPin()       : unsigned_t
+ setCardPin()
+ getInterestRate() : double
+ setInterestRate()
+ getAccBalance()   : double
+ setAccBalance()
# sendtoDBMS()
```

The CardAcc class is focused on the card inserted by the actor into the ATM machine. This class stores information regarding identifying the card. This includes card number, the expiration card, the card verification value, card pin number, interest rate associated with account and the account balance. To summarize, the CardAcc class is primarily focused on a user's ATM card that's associated with MSB and depending on their status, they will have different features and premiums over other accounts.

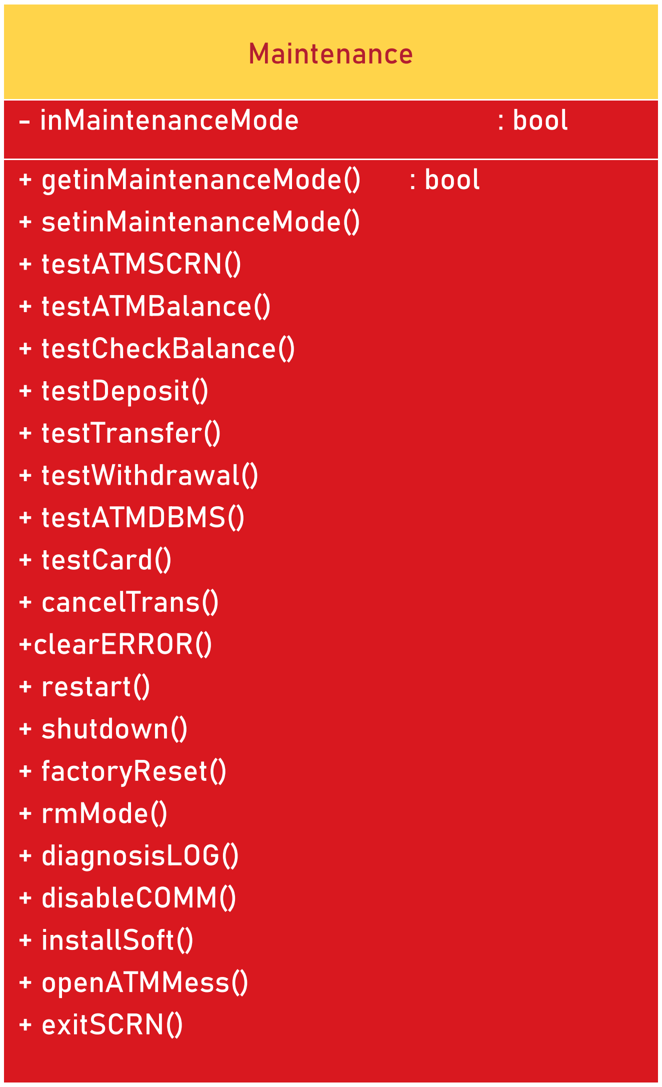


The ATMCore is a class that holds many important ATM operations that records any type of transaction being done and sends every recorded transaction to the connected Oracle database. There's many operations such as drawerClose() which activates when the ATM detects that the inputted cash/check from the user is inside and then closes immediately. Another operation that can activate and display to the screen is the errorLoginMess() which will activate if a user has unsuccessfully authenticate themselves after three times of inputting their pin incorrectly.



ATMReceipt	
- unitID	: unsigned_t
- unitLocation	: string
- ActorName	: bool
- typeOfTrans	: string
- currencyExged	: double
+ toEmail	: bool
+ toPrint	: bool
- isPrinted	: bool
- isEmailed	: bool
- transTime	: time
- transDate	: date
- transID	: int
+ getUNITID()	: unsigned_t
+ getULocation	: string
+ getisOperational()	: bool
+ getCurrencyExged ()	: int
+ drawerClose()	
+ calcCurrTotal()	
+ errorLoginMess()	
+ ejectCard()	
- cancelTrans()	
- ATMCheck()	
- toPrint()	
- toEmail()	
- noReceipt()	
- receipt()	
- confirmEmail()	
- inputEmail()	
- changeEmail()	

The ATMReceipt is the class that will hold all of the different types that any user that interacts with the ATM can display their receipt. The receipt can either be printed out, sent to a user's email, confirm whether it's the correct email, display the specific type of MSB ATM any transaction was performed, any currency exchange, location, and also some other vital ATM operations. It holds some operations that are also used in the ATMCore such as ejectCard() which will eject a user's card whenever they're done with their transaction and confirm they wouldn't like to perform any other type of transaction.

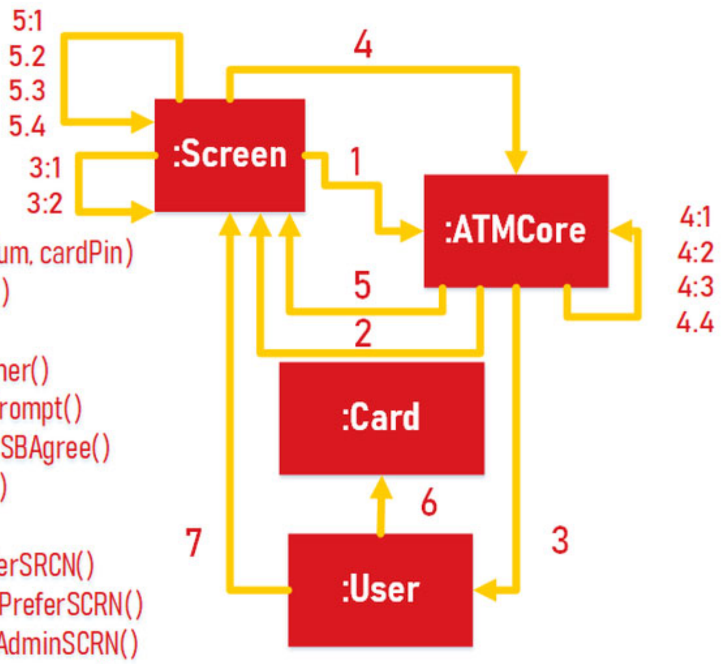


The Maintenance class is the type that can only be used by one of the actors which is none other than the System Administrator. The system administrator would already have to authenticate themselves with the ATM and once in it, they will have access various ATM procedures. They can perform many different types of tests ranging from deposits, withdrawals, checking balance, transferring money between accounts, testing connection with the database, restarting the ATM, shutting it down, and many more operations. It comes down to what exactly the system administrator would like to check based off of the diagnosisLOG() and from there perform the correct operation to resolve the issue.

Authentication

Communication Diagram

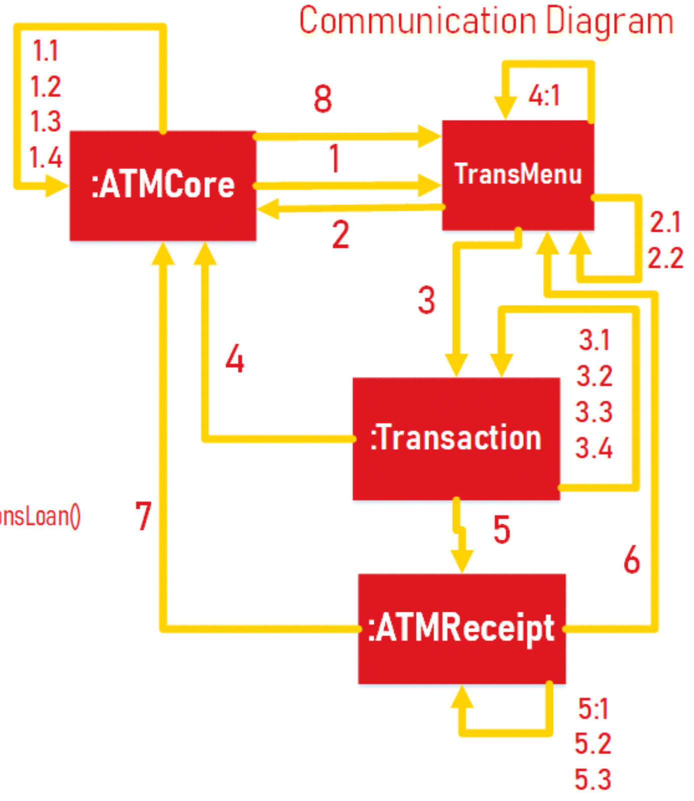
1: welcomeSCRN()
2. ATMCheck()
3: validInsertCard()
3: displayPinSCRN()
    3.1: [Based on the pin]:validAcct(cardNum, cardPin)
    3.2: [Based on the pin]:errorLoginMess()
4: sendtoDBMS(cardNum, cardPin)
    4.1:[Based on the validation]: validPartner()
    4.2:[Based on the validation]: nonMSBPrompt()
    4.3:[Based on the validation]: captureMSBAgree()
    4.4:[Based on the validation]:accFault()
5: validated(status)
    5.1: [Based on the status]: displayPreferSRCN()
    5.2: [Based on the status]: displayNONPreferSCRN()
    5.3: [Based on the status]: displaySYSAdminSCRN()
    5.4:[Based on the status]:displayNONMSBSCRN()
6: ejectCard()
7: Logoff()



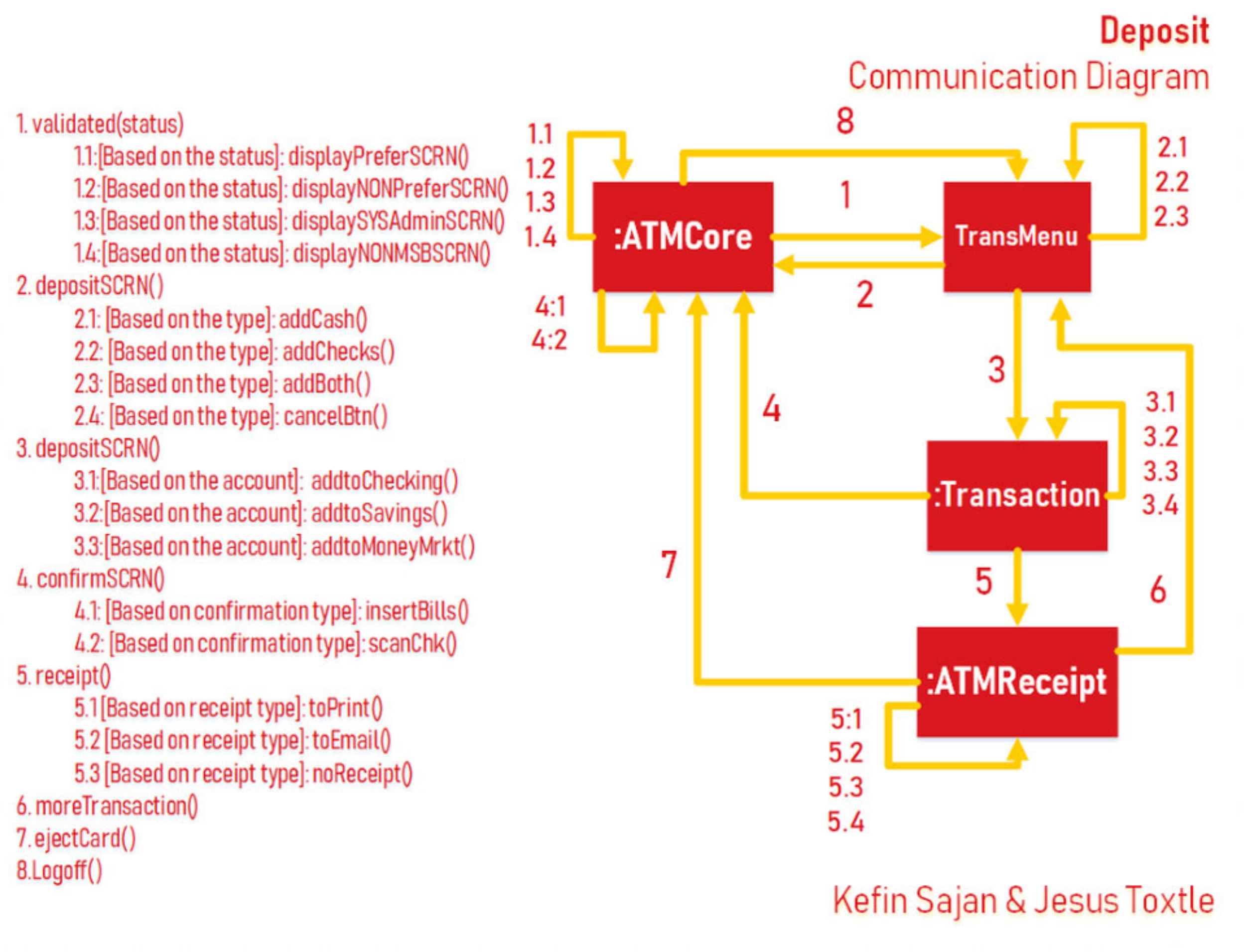
Kefin Sajan & Jesus Toxtle

Withdrawal
Communication Diagram

1. validated(status)
    - 1.1:[Based on the status]: displayPreferSCRN()
    - 1.2:[Based on the status]: displayNONPreferSCRN()
    - 1.3:[Based on the status]: displaySYSAdminSCRN()
    - 1.4:[Based on the status]: displayNONMSBSCRN()
2. calcCurrTotal()
2. withdrawalSCRN()
    - 2.1: [Based on the type]: rmCash()
    - 2.2: [Based on the type]: cancelBtn()
3. depositSCRN()
    - 3.1:[Based on the account]: rmfromChecking()
    - 3.2:[Based on the account]: rmfromSavings()
    - 3.3:[Based on the account]: rmfromMoneyMrkt()
    - 3.4:[Based on the account and isPreferred]: addtoConsLoan()
4. confirmSCRN()
    - 4.1: [Based on confirmation type]: rmCurrBal()
5. receipt()
    - 5.1 [Based on receipt type]: toPrint()
    - 5.2 [Based on receipt type]: toEmail()
    - 5.3 [Based on receipt type]: noReceipt()
6. moreTransaction()
7. ejectCard()
8. Logoff()



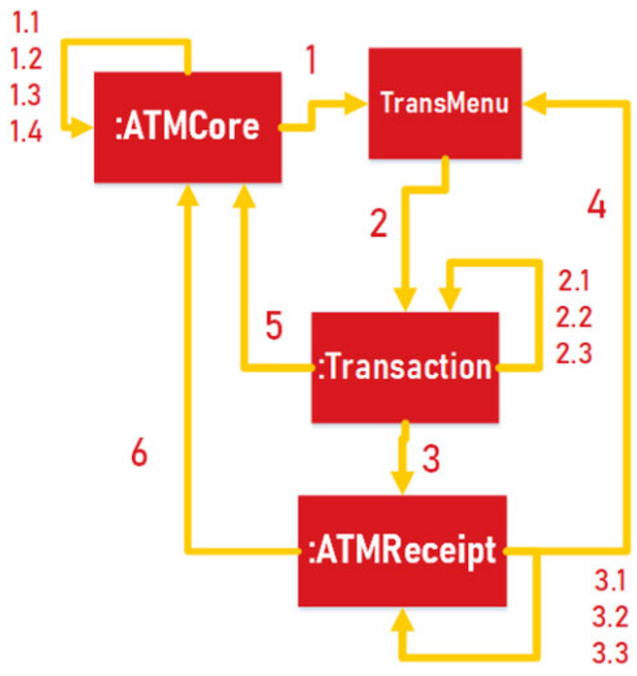
Kefin Sajan & Jesus Toxtle



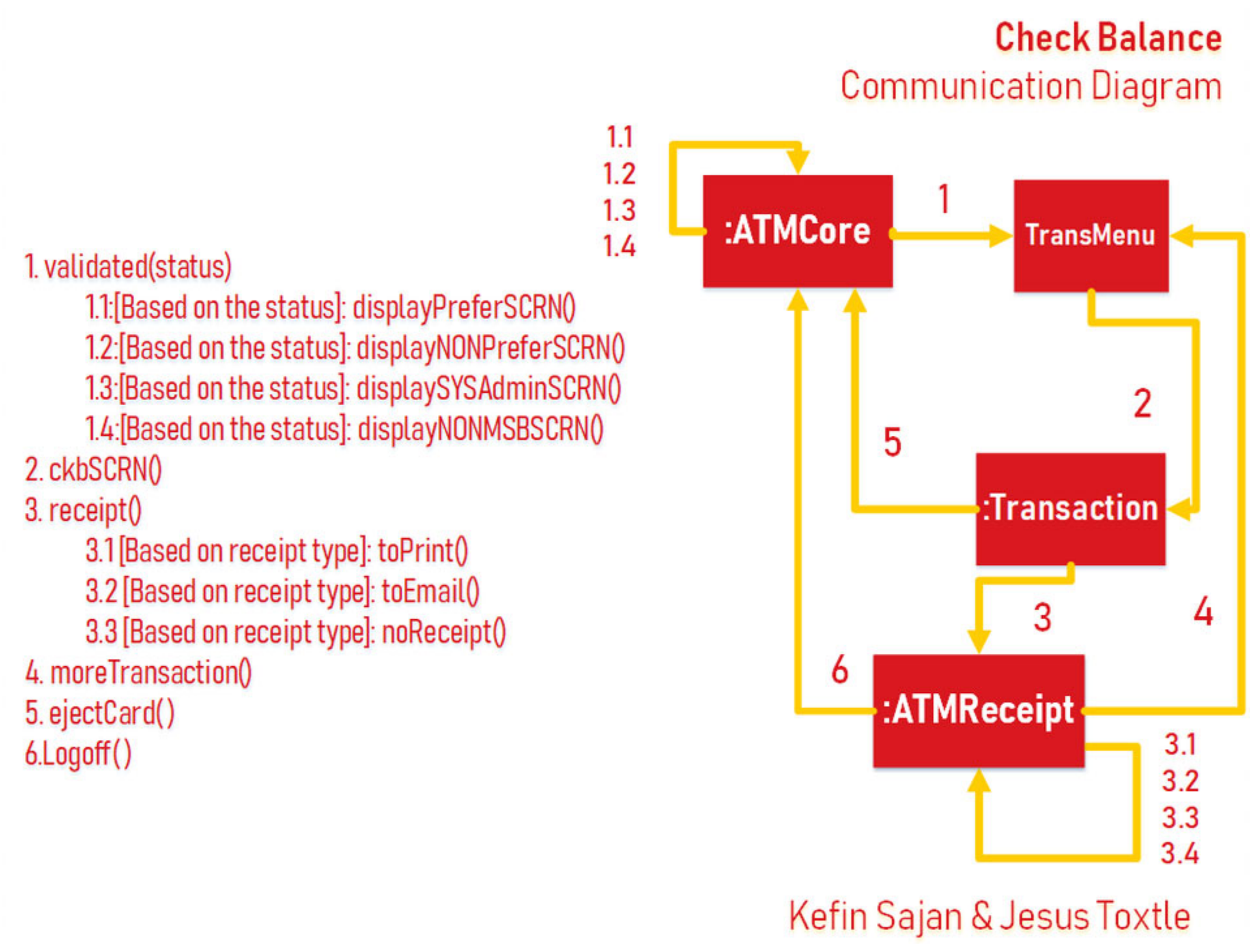
1. validated(status)
    1.1:[Based on the status]: displayPreferSCRN()
    1.2:[Based on the status]: displayNONPreferSCRN()
    1.3:[Based on the status]: displaySYSAdminSCRN()
    1.4:[Based on the status]: displayNONMSBSCRN()
2. depositSCRN()
    2.1: [Based on the type]: addCash()
    2.2: [Based on the type]: addChecks()
    2.3: [Based on the type]: addBoth()
    2.4: [Based on the type]: cancelBtn()
3. depositSCRN()
    3.1:[Based on the account]: addtoChecking()
    3.2:[Based on the account]: addtoSavings()
    3.3:[Based on the account]: addtoMoneyMrkt()
4. confirmSCRN()
    4.1: [Based on confirmation type]: insertBills()
    4.2: [Based on confirmation type]: scanChk()
5. receipt()
    5.1 [Based on receipt type]: toPrint()
    5.2 [Based on receipt type]: toEmail()
    5.3 [Based on receipt type]: noReceipt()
6. moreTransaction()
7. ejectCard()
8.Logoff()

Transfer
Communication Diagram

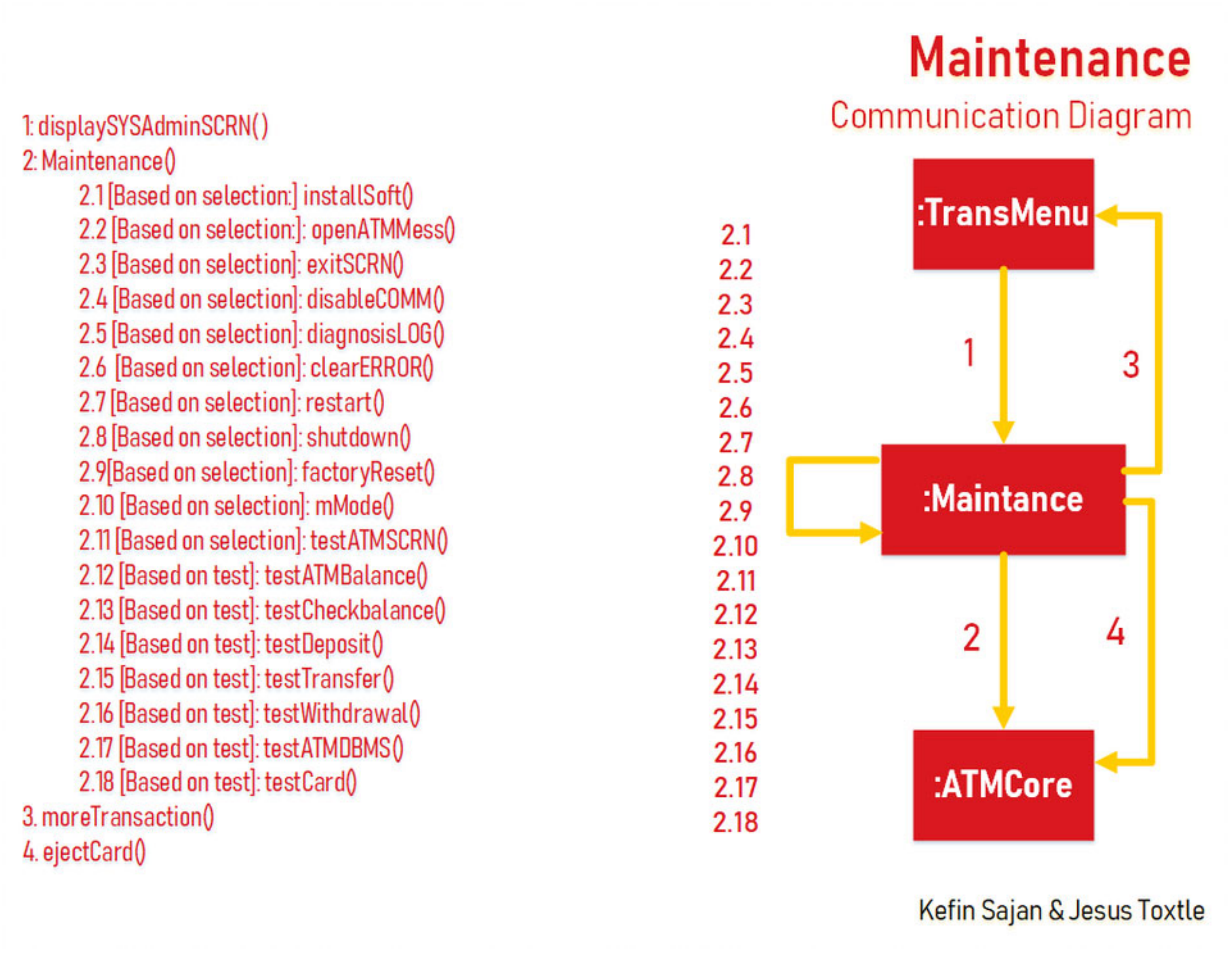
1. validated(status)
    - 1.1:[Based on the status]: displayPreferSCRN()
    - 1.2:[Based on the status]: displayNONPreferSCRN()
    - 1.3:[Based on the status]: displaySYSAdminSCRN()
    - 1.4:[Based on the status]: displayNONMSBSCRN()
2. transferSCRN()
2. confirmSCRN()
2. transferFromAcct()
    - 2.1:[Based on selected account]: rmfromChecking
    - 2.2:[Based on selected account]: rmfromsaving
    - 2.3:[Based on selected account]: rmfromMoneyMkt()
2. transDestAcct()
    - 2.1:[Based on selected account]: addtoChecking
    - 2.2:[Based on selected account]: addtosaving
    - 2.3:[Based on selected account]: addtoMoneyMkt()
    - 2.4:[Based on selected account]: addtoMortage
    - 2.5:[Based on selected account]: addtoConsLoan()
3. receipt()
    - 3.1 [Based on receipt type]: toPrint()
    - 3.2 [Based on receipt type]: toEmail()
    - 3.3 [Based on receipt type]: noReceipt()
4. moreTransaction()
5. ejectCard()
6.Logoff()



Kefin Sajan & Jesus Toxtle



1. validated(status)
    - 1.1:[Based on the status]: displayPreferSCRN()
    - 1.2:[Based on the status]: displayNONPreferSCRN()
    - 1.3:[Based on the status]: displaySYSAdminSCRN()
    - 1.4:[Based on the status]: displayNONMSBSCRN()
2. ckbSCRN()
3. receipt()
    - 3.1 [Based on receipt type]: toPrint()
    - 3.2 [Based on receipt type]: toEmail()
    - 3.3 [Based on receipt type]: noReceipt()
4. moreTransaction()
5. ejectCard( )
6. Logoff( )



Maintenance

Communication Diagram

1: displaySYSAdminSCRN( )
2: Maintenance()
- 2.1 [Based on selection:] installSoft()
- 2.2 [Based on selection:]: openATMMess()
- 2.3 [Based on selection]: exitSCRN()
- 2.4 [Based on selection]: disableCOMM()
- 2.5 [Based on selection]: diagnosisLOG()
- 2.6 [Based on selection]: clearERROR()
- 2.7 [Based on selection]: restart()
- 2.8 [Based on selection]: shutdown()
- 2.9[Based on selection]: factoryReset()
- 2.10 [Based on selection]: mMode()
- 2.11 [Based on selection]: testATMSCRN()
- 2.12 [Based on test]: testATMBalance()
- 2.13 [Based on test]: testCheckbalance()
- 2.14 [Based on test]: testDeposit()
- 2.15 [Based on test]: testTransfer()
- 2.16 [Based on test]: testWithdrawal()
- 2.17 [Based on test]: testATMDBMS()
- 2.18 [Based on test]: testCard()

3. moreTransaction()
4. ejectCard()

Kefin Sajan & Jesus Toxtle

C++ Implementation:

ActorInterface Class Implementation:

TransMenu Class Implementation:

Transaction Class Implementation:

Actor Class Implementation:

Card Class Implementation:

ATMCore Class Implementation:

ATMReceipt Class Implementation:

Maintenance Class Implementation: