**ATM Machine for Montana Savings Bank**

Software Engineering

Joseph Virges

Dylan Flores

**ATM Machine System**

**Table of Contents:**

Introduction of ATM project: pg 2

Team Information: pg 3

Use Case Diagram: pg 4

Use Case Descriptions: pg 5

Class Diagrams: pg 13

Communication Diagrams: pg 19

Header Files: pg 27

**ATM Machine System**

**Introduction To the ATM Project**

Montana Savings Bank (MSB) only has three branches and they recently decided to streamline its operations for cost-cutting purposes. To save labor costs, eliminate manual errors, and provide 24-hour service to its customers, Montana Savings Bank wanted us to create an ATM Machine software to be used for their services.

The goal is to be able to create an ATM software that can be used by the MSB group to fulfill the customer’s needs. The customers will be able to access the ATM machine by having an ATM card issued by MSB. The ATM machine will be able to handle transactions of five customer accounts which are the checking, savings, money market, consumer loan, and mortgage accounts. For CDs, or Certificate of Deposits, customers are only allowed to check their balance on the ATM, any form of transaction is not allowed. This ATM machine software would allow the customers to check their balance, withdraw and deposit from their selected accounts, and transfer money between their accounts. Customers may deposit any amount but are limited to withdrawing, in multiples of $10, $500 per day from all accounts. They can transfer money among their checking, savings, and money market accounts, or they can move money from those accounts to consumer loan and mortgage. The ATM will allow non-MSB customers to use its services by having their banks’ ATM card to either withdraw or deposit, with checks or cash, money from their savings or checking accounts with an additional service fee. At the end of every ATM usage, the customer will be able to choose if they wish a receipt or not.

Additionally, all MSB ATMs, either within their three branches or elsewhere, must be connected to a central database, where every ATM transaction will be pooled into automatically. Customer information will be stored within the central database, which will be responsible for interacting with other banks. MSB has a group of preferred customers, which gives them special perks, such as the ability to withdraw more money than their balance from their accounts, which comes from their loan account, up to the limit. Regular customers only have the ability to take out money from a customer loan account, up to a limit negotiated per individual. It also comes with no hold on their deposits unlike the three-day hold placed on the check of the regular customers’ deposits.

**Team Information**

Joseph Virges:

Working on this project was pretty difficult because I am not used to the online environment. I tend to work and learn more hands on than being online at home. Mr. Flores and I also have different schedules and classes, so we had to set up a time and day when to talk and work on this together. We had to collaborate and communicate through google docs and an application called discord. I am a novice when it comes to software engineering. I didn’t know much about how software engineers work so I found working on this project a new experience. I didn’t know that there was a lot of documentation involved in creating software at first, but now I understand why there is a lot of documentation because it makes it easier to fix mistakes when a software is properly documented.

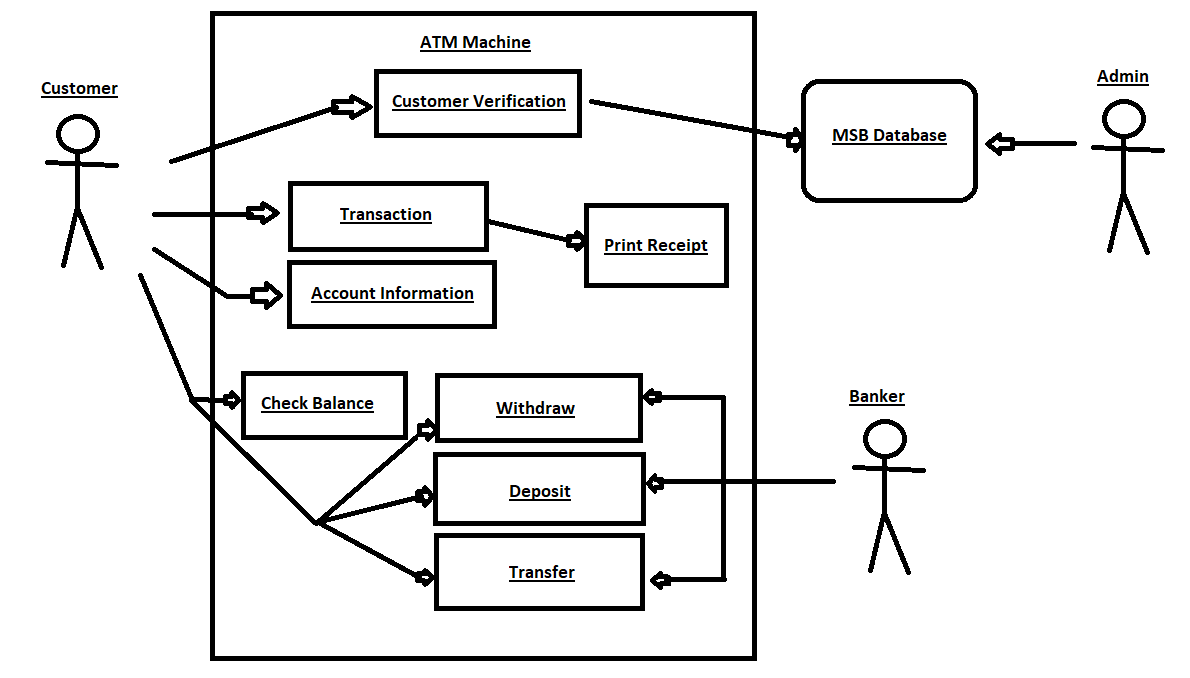
I think if we were able to meet in person, this project would’ve been easier. The main platform that we used for this final project is through the use of google docs to make it easier when we edit our document because our previous projects, we just sent word doc files and it did not work properly because we could not see changes in real time. It allowed us to see what changes were made and made it easier to follow up.

Dylan Flores:

This semester was a test run for continuing education during a quarantine, a period where both teachers and students had to get themselves accustomed to their new online environment. This semester has also taught me that I do far better in class, with less distractions, than I do at home. Mr. Virges and I had conflicting schedules, which made it difficult to coordinate ourselves in time for projects. Luckily, thanks to Google Docs and Discord, we managed to pull ourselves together. This project, while difficult on its own, also focused on something that I had very little experience with. This was the first project that focused exclusively on documentation, with no coding to be seen. In similar projects, I would usually code and let my partner handle the documentation with my input in how everything worked. However, as it was all documentation, I was forced to learn on the fly and then try to apply it to my projects.

Had we been able to have classes and meetings with my partner in person, this project, while still hard, would have been much more manageable. The platforms we were forced to resort to had their advantages but it failed to make up for what in-person meetings have. There was less discussion because of it and our conflicting schedules.

**Use Case Diagram**



Use Case Diagram Text Description:

In our use case diagram, there are four types of transactions that a customer can do at once using the MSB ATM: check balance, withdraw, deposit, and transfer. To start the transaction, a customer needs to verify whether they are a customer of MSB or a non-MSB customer by inserting their bank card. The machine will scan the card and communicate with the database. It will compare the card’s validity if it is an MSB or a non-MSB issued card. Then the ATM Machine will prompt the user to input their PIN to verify that it is the customer making the transaction. The PIN is then compared by the PIN stored in the database. Once it is verified, the customer is then prompted what kind of transaction to make.

During the transaction phase, the customer is to choose a type of transaction between check balance, withdraw, deposit, and transfer. Once they have chosen a transaction type, the machine prompts the customer to choose the account type: checking, savings, money market, consumer loan, and mortgage, that they wish to make the transaction from. After choosing the account type, the ATM machine communicates with the database to get the customer’s account type and balance information connected to that account. At the end of the transaction, the customer is then prompted whether they want to print a receipt or not and that would end the ATM operation.

The banker would be present near an ATM machine if the ATM machine is located near or inside the bank. If there are faults with the ATM machine transaction, the banker can handle the desired transaction of the customer. The administrator would handle and oversee the communication between the ATM machine and the bank’s database. If there are faults within the system, the administrator can look at the issue within the system and fix the issue. Given that, the administrator has full access to the operations of the machine so that they could fix the problem that occurred within the system.

**Use Case Descriptions**

1. Customer Verification

2. Access Account Type

3. Money Withdrawal

4. Money Deposit

5. Money Transfer

6. Check Balance

7. ATM Machine System

Customer Verification

* Author: Joseph Virges and Dylan Flores
* Last update: 11/17/20
* Preconditions:
  + The customer must have a valid card that is issued by MSB or other banks
* Dialog:
  + Screen displays welcome to MBS and please insert card - display\_logon\_screen( )
  + Customer inserts the ATM card and the ATM asks the customer for their PIN code to verify- capture\_card\_pin( )
    - The ATM communicates to the database and compare the PIN code associated with the bank card - communicate\_DB( ), verification (card\_num, PIN) : acc\_type
    - System verifies the account (if account = “verified”) and returns account type connected to the card
      * MSB prefered and non-preferred account types (if account\_type = “prefered”) or (if account\_type = “non-prefered”)
        + Check balance - check balance use case
        + Withdraw - money withdrawal use case
        + Deposit - money deposit use case
        + Transfer - money transfer use case
      * Non-MSB customers (if account\_type = ‘Non-MSB’)
        + Check balance - check balance use case
        + Withdraw from checking or savings account - money withdrawal use case
        + Deposit from checking or savings account - money deposit use case
    - System can not verify the account (if account = “unverified”)
      * The customer can enter the PIN code a maximum of three tries. After three tries, the user cannot access the account and screen displays cannot use the machine message - error\_handling\_1( ), lock\_customer( )
      * The system sends a message to the administrator about failed verification - error\_handling\_2( ), communicate\_DB(), send\_error( )
  + Customer session ends- exit( )
  + Postcondition:
    - Customer verification is successful, the screen will display appropriate information according to the customer’s bank account from the Access Account Type use case
    - Customer session ends normally, system will exit and display welcome to MBS screen - exit( ), display\_logon\_screen( )
    - Customer verification is unsuccessful, screen shows message then the session will end and send a en error to the administration then go back to welcome screen - error\_handling\_1( ), error\_handling\_2( ), send\_error( ), exit( ), display\_logon\_screen()

Access Account Type

* Author:  Joseph Virges and Dylan Flores
* Last Update: 11/17/20
* Preconditions:
  + The customer must have been verified by the system via Customer Verification use case
* Dialog:
  + For MSB customers, screen shows main menu of account types for MSB customers- display\_MSB\_Account()
    - Menu allows customer to access all available account to check balance from one of these accounts: checkings, savings, money market, consumer loan, mortgage, and Certificate of Deposit (CD) - check balance use case
  + For Non-MSB customers, screen shows main menu of account types for non-MSB customers- display\_nonMSB\_Account()
    - Menu allows customer to access their checkings or savings accounts and allow to check balance - check balance use case
  + If MSB or non-MSB customer selects checkings -checking\_account()
    - Screen shows checking account’s balance - display\_checking\_bal()
    - If user wants to deposit money - money deposit use case
    - If user wants to withdraw money - money withdrawal use case
    - If user, MSB customer only, wants to transfer money - money transfer use case
    - Additional icons show: go back to main menu of account types or end session - display\_MSB\_Account(), exit() or display\_nonMSB\_Account(), exit()
  + If MSB or non-MSB customer selects savings- saving\_account()
    - Screen shows savings account’s balance - display\_savings\_bal()
    - If user wants to deposit money  - money deposit use case
    - If user wants to withdraw money - money withdrawal use case
    - If user, MSB customer only, wants to transfer money - money transfer use case
    - Additional icons show: go back to main menu of account types or end session - display\_MSB\_Account(), exit() or display\_nonMSB\_Account(), exit()
  + If MSB customer selects money market - market\_account()
    - Screen shows money market account’s balance - display\_market\_bal()
    - If user wants to deposit money  - money deposit use case
    - If user wants to withdraw money - money withdrawal use case
    - If user wants to transfer money - money transfer use case
    - Additional icons show: go back to main menu of account types or end session - display\_MSB\_Account(), exit()
  + If MSB customer selects consumer loan - loan\_account()
    - Screen shows consumer loan account’s balance - display\_loan\_bal()
    - If user wants to withdraw money - money withdrawal use case
    - Additional icons show: go back to main menu of account types or end session - display\_MSB\_Account(), exit()
  + If MSB customer selects mortgage - mortgage\_account()
    - Screen shows mortgage account’s balance - display\_mort\_bal()
    - Additional icons show: go back to main menu of account types or end session - display\_MSB\_Account(), exit()
  + If MSB customer selects Certificate of Deposit - CD\_account()
    - Screen shows CD account’s balance - display\_cd\_bal()
    - Additional icons show: go back to main menu of account types or end session - display\_MSB\_Account(), exit()
* Postconditions:
  + If customer chose:
    - Checking account: screen shows checking account balance after transaction - display\_checking\_bal(), communicate\_DB()
    - Savings account: screen shows savings account balance after transaction - display\_saving\_bal(), communicate\_DB()
    - Money market account: screen shows money market account balance after transaction - display\_market\_bal(), communicate\_DB()
    - Consumer loan account: screen shows consumer loan account balance after transaction - display\_loan\_bal(), communicate\_DB()
    - Mortgage account: screen shows mortgage account balance after transaction - display\_mort\_bal(), communicate\_DB()
    - CD account: screen shows CD account balance after transaction - display\_cd\_bal(), communicate\_DB()
  + Screen will show transaction options after chosen account - display\_transaction(), communicate\_DB()
  + Depending on chosen transaction: Money withdrawal use case, money deposit use case, or money transfer use case will be done

Money Withdrawal

* Author:  Joseph Virges and Dylan Flores
* Last Update: 11/17/20
* Preconditions:
  + The customer must have been verified by the system via Customer Verification use case
  + Is called by Access Account Type use case(either checking, savings, money market, consumer loan accounts)
* Dialog:
  + Money withdrawal main screen is shown with the type of account chosen by the customer via Access Account Type use case(checking, saving, money market, or consumer loan accounts) - display\_withdraw\_screen()
  + Customer is prompted to enter withdrawal amount - getwithdrawAmount()
    - Withdrawal transaction must be in multiples of $10 - error\_handling\_with1()
    - If non-MSB customer: charge service fee of $3 is added per withdrawal - communicate\_DB()
    - If MSB customer is preferred , customer has overdraft protection which means they have no limit that they can withdraw
      * If withdrawal amount is more than in the balance of the account, money will go from their consumer loan account established by the loan - communicate\_DB()
    - If MSB customer is non-preferred, customer has a limit withdrawal of $500 per day, screen will show unvalid if amount exceeds $500 and returns back to money withdrawal main screen - error\_handling\_with2(), display\_withdraw\_screen()
    - If MSB customer is non-preferred with consumer loan account, they can withdraw money established by the loan - communicate\_DB()
  + Customer confirms the amount to be withdrawn from their chosen type of account- setwithdrawAmount(), communicate\_DB()
  + Withdraw slot opens with cash then closes after retrieval of cash - with\_open(), with\_close()
  + After withdrawal transaction, customer is prompted to print receipt or end session - print\_receipt(), exit()
* Postconditions:
  + The customer successfully withdraw money from their chosen account type (checking, saving, money market, or consumer loan accounts) and transaction is sent to central database via ATM Machine System use case
  + Customer gets a receipt or not after session - print\_receipt(), exit()
  + Screen shows welcome to MBS screen - display\_logon\_screen()

Money Deposit

* Author:  Joseph Virges and Dylan Flores
* Last Update: 11/17/20
* Preconditions:
  + The customer must have been verified by the system via Customer Verification use case
  + Is called by Access Account Type use case(either checking, savings, money market, consumer loan accounts)
* Dialog
  + Money deposit main screen is shown with the type of account chosen by the customer via Access Account Type use case(checking, saving, or money market accounts) - display\_deposit\_screen(), warning\_sign()
    - Screen is also showing check deposit or cash deposit and warning that Deposit can only be cash or check, no coins
  + Customer chooses check deposit: - checks()
    - If MSB preferred customer: no hold on check deposits - communicate\_DB()
    - If MSB non-preferred customer: a three day hold on check deposits- communicate\_DB()
    - If non-MSB customer: charge service fee of $3 is added per deposit - communicate\_DB()
    - Screen shows enter amount - getdeposit\_Amount()
      * Screen prompts user to insert check into deposit slot after entering amount then once inserted, deposit slot closes - open\_deposit(), close\_deposit()
      * System verifies if amount of check is equal to amount entered by customer - communicate\_DB()
        + If the same - setdeposit\_Amount(), communicate\_DB()
        + If not the same it shows error and it will open deposit slot and release check- error\_handling\_dep1(), open\_deposit()

Edit of amount screen will come up and prompts user to edit amount - error\_handling\_dep2(), close\_deposit()

* + - After system verifies amount and confirms transaction, customer is prompted to print transaction or terminate session - print\_receipt(), exit()
  + Customer chooses cash deposit: - cash()
    - If MSB preferred and non-preferred customers: no hold on their cash deposit - communicate\_DB()
    - If non-MSB customer: charge service fee of $3 is added per deposit - communicate\_DB()
    - Screen shows enter amount - getdeposit\_Amount()
      * Screen prompts user to insert cash into deposit slot after entering amount then once inserted, deposit slot closes - open\_deposit(), close\_deposit()
      * System verifies if amount of cash is equal to amount entered by customer- communicate\_DB()
        + If the same - setdeposit\_Amount(), communicate\_DB()
        + If not the same it shows error and it will open deposit slot and release cash- error\_handling\_dep1(), open\_deposit()

Edit of amount screen will come up and prompts user to edit amount - error\_handling\_dep2(), close\_deposit()

* + - After system verifies amount and confirms transaction, customer is prompted to print transaction or terminate session - print\_receipt(), exit()
* Postconditions:
  + The customer successfully deposit money from their chosen account type (checking, saving, or money market accounts) and transaction is sent to central database via ATM Machine System use case
  + Customer gets a receipt or not after session - print\_receipt(), exit()
  + Screen shows welcome to MBS screen - display\_logon\_screen()

Money Transfer

* Author:  Joseph Virges and Dylan Flores
* Last Update: 11/17/20
* Preconditions:
  + The customer must have been verified by the system via Customer Verification use case
  + Is called by Access Account Type use case(either checking, savings, money market, consumer loan accounts)
* Dialog:
  + Money transfer main screen is shown with the type of account chosen by the customer via Access Account Type use case(checking, saving, or money market accounts) - display\_transfer\_screen()
    - Customers can transfer money among checking, savings, and money market accounts
    - Customers can transfer money from checking, savings, and money market accounts to consumer loan
    - Customers can transfer money from checking, savings, and money market accounts to mortgage
    - Customers does not have the option to transfer from consumer loan and mortgage accounts
  + Screen asks customer to enter how much money to be transferred - transferAmount()
    - If amount is too much for account balance, error message come up and please edit amount again - error\_handling\_trans1(), change\_transferAmount()
    - If amount is in balance - transferAmount()
  + Screen asks customer which account to transfer to (checking, saving, money market; if applicable: consumer loan or mortgage accounts) - transferAcc()
  + Screen shows amount and transfer to account and ask confirmation from customer - display\_transfers(), transferAmount(), transferAcc(), communicate\_DB()
* Postcondition:
  + The customer successfully transfer money from their chosen account type (checking, saving, or money market accounts) to chosen account type (checking, saving, money market; if applicable: consumer loan or mortgage accounts) and transaction is sent to central database via ATM Machine System use case
  + Customer gets a receipt or not after session - print\_receipt(), exit()
  + Screen shows welcome to MBS screen - display\_logon\_screen()

Check Balance

* Author:  Joseph Virges and Dylan Flores
* Last Update: 11/17/20
* Preconditions:
  + The customer must have been verified by the system via Customer Verification use case
  + Is called by Access Account Type use case(either checking, savings, money market, consumer loan, mortgage accounts)
* Dialog:
  + After selection of account type via Access Account Type use case:
    - MSB customers can check the balance - MSB\_Account(), communicate\_DB()
    - Non-MSB customers can check the balance - nonMSB\_Account(), communicate\_DB()
* Postconditions:
  + Customers can check balance of chosen account type before choosing transaction -go\_backTrans()

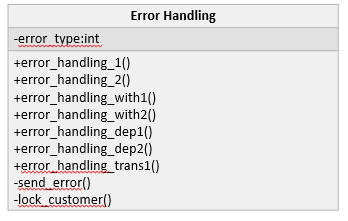
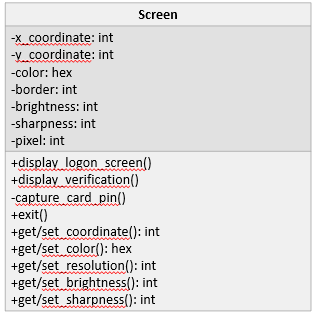
ATM Machine System

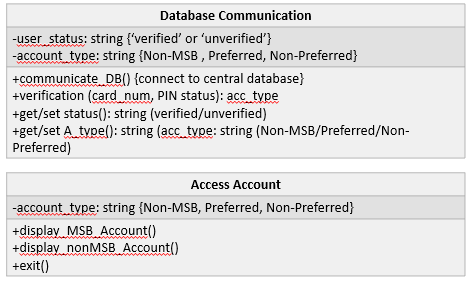
* Author:  Joseph Virges and Dylan Flores
* Last Update: 11/17/20
* Preconditions:
  + Verified transaction made by the customer
  + Connection with central database is working
* Dialog:
  + Called by Customer Verification use case:
    - Connects to central database, checks and compares customer card number and associated PIN - communicate\_DB()
      * Verified customer- (card\_num, PIN) : acc\_type = ‘verified’
      * Unverified customer- (card\_num, PIN) : acc\_type = ‘unverified’
    - After verification, checks customer type: MSB preferred, MSB non-preferred, non-MSB - account\_type, communicate\_DB()
  + Called by Access Account Type use case:
    - Connects to central database to check available accounts for the customer - communicate\_DB(), accounts()
  + Called by Money Withdraw use case:
    - Connects to central database to get available amount to be withdrawn and updates balance after withdrawal - communicate\_DB(), bals\_available(), updatebal()
  + Called by Money Deposit use case:
    - Connects to central database to check, verify, and update amount of balance after deposit verification process - communicate\_DB(), updatebal()
  + Called by Money Transfer use case:
    - Connects to central database to check, verify, and update amounts of balances of selected accounts after transfer transaction - communicate\_DB()
  + Called by Check Balance use case:
    - Connects to central database to get and set the balances of the customer’s accounts - communicate\_DB(), bals()
  + When there is no more cash: sends a message to central database (for administration) about no cash and shows on screen ATM is out of service - communicate\_DB(), sorry(), display\_nocash\_screen()
* Postconditions:
  + ATM machines of MSB are connected with the central database, allowing interaction with other banks
  + Allows further transaction of customer after verification
  + Updates customer information connected with their account and account balance
  + Allows interaction of customer, machine (screen), and database(DB)
  + Terminates sessions of customer and returns to welcome screen or no cash screen - exit(), display\_logon\_screen(), display\_nocash\_screen()

--------------------------------------------------------------------------------------------------------------------

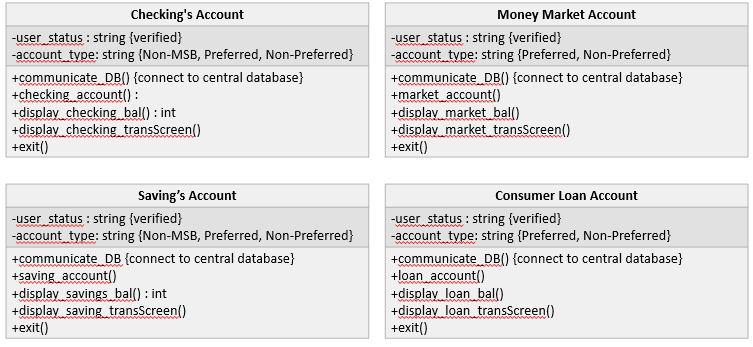
**Class Diagrams**

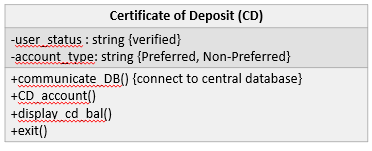
Atm Machine Screen –



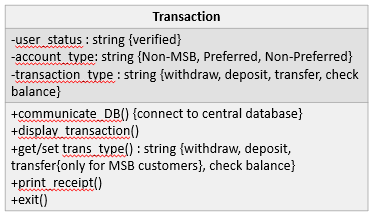


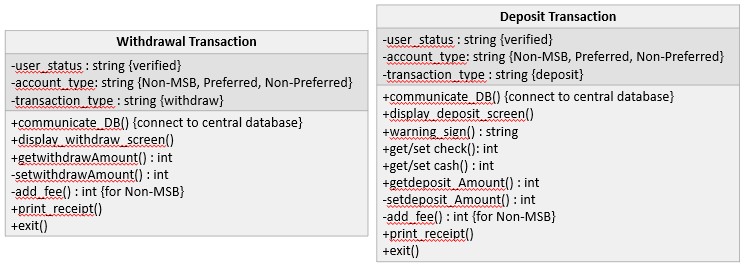
Customer Accounts-

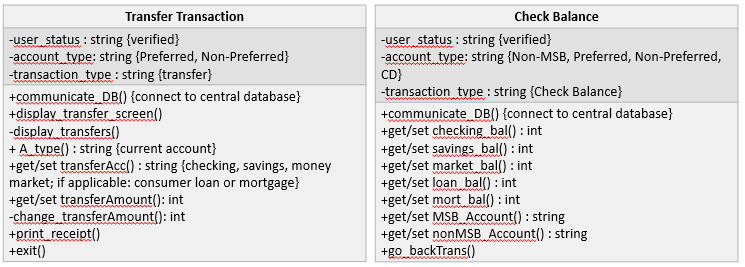




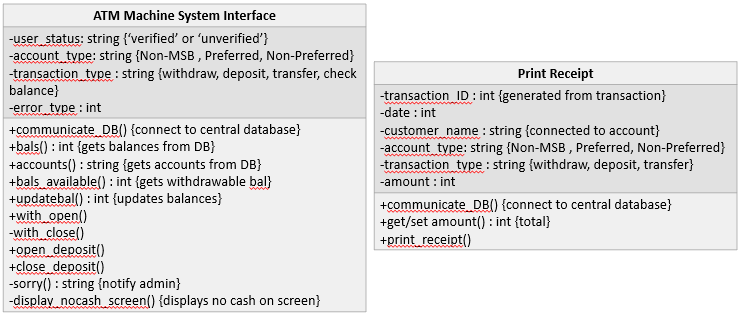
Transactions-



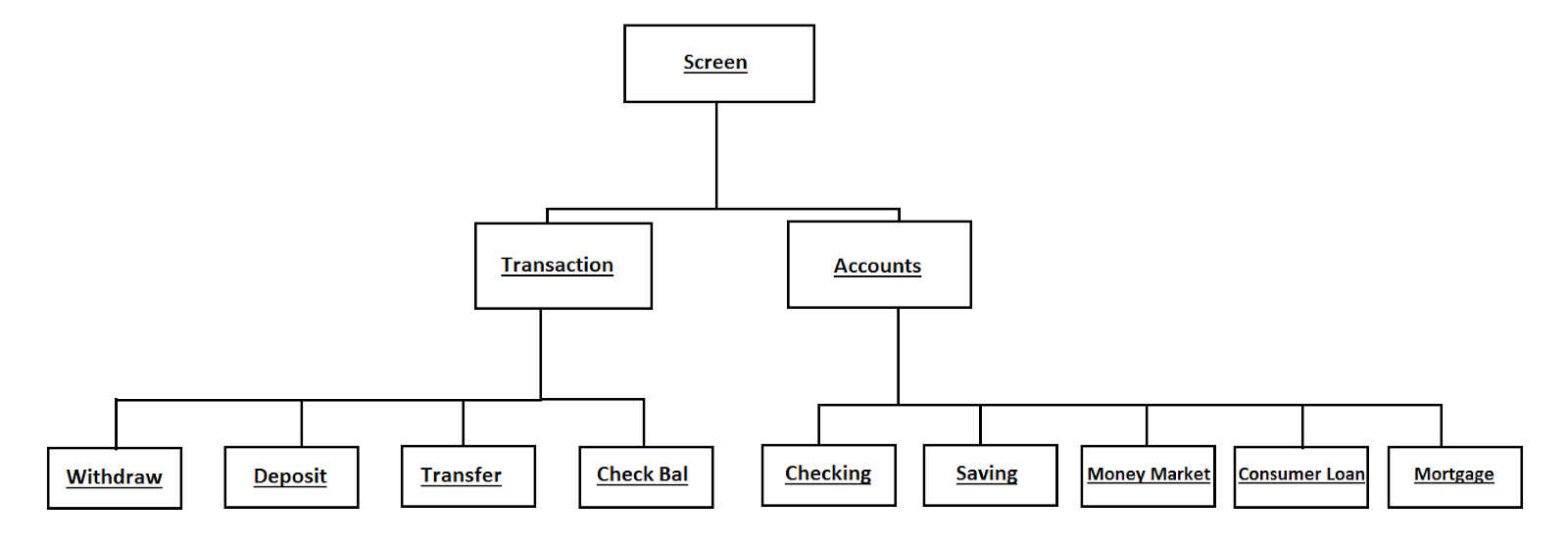




ATM Machine System -



**Hierarchical Diagrams**

****

**Class Descriptions**

Atm Machine Screen-

The Atm Machine Screen class interacts with the customer the most because it is where the customer will choose their accounts, form of transaction, options, verification, print receipt, etc. It displays the environment that allows the customer to communicate between physical interaction to the atm machine virtual system (central database). It has all the operations needed for this communication and displayed environment.

Error Handling-

The error handling class allows the system to handle any errors that will occur via communicating with the database (sending error messages to admin, etc.) and the error on the machine (error amount, transactions, etc.). It has all the operations to handle errors that will occur with the customer’s verification of accounts, transactions, etc.

Database Communication-

The database communication class is where the customer, machine, and central database are connected. It allows the atm machine to communicate (send and get information) from the central database of MSB and other atm machines. It has all the operations that will get information, send information, verification of customer and customer type, errors, etc.

Access Account-

The access account class is where the customer chooses their accounts. It allows the atm machine to get information (via database communication) from the central database of the chosen account of the customer. It has the operations to allow access over the accounts (checking, savings, money market, consumer loan[if applicable], mortgage[if applicable], CD[if applicable]) after verification of the customer depending on the customer type (MSB or Non-MSB).

Transactions-

The transaction class is where the customer does their transaction (withdraw, deposit, transfer[if applicable], check balance). It allows the machine to get information and send information (via database communication) to the central database about chosen transactions. It has the operations to set transactions and allow transactions.

Withdrawal Transaction-

The withdrawal transaction class is where the customer does their withdrawal. It allows the atm machine to get information (via database communication) from the central database about customer’s type (MSB[Preferred, Non-Preferred] or Non-MSB) and chosen account type (checking, savings, money market [if applicable], consumer loan [if applicable]). It has the operations for setting the amount to be withdrawn, sending the amount back to the central database, opening or closing withdrawal slot.

Deposit Transaction-

The deposit transaction class is where the customer does their deposit. It allows the atm machine to get information (via database communication) from the central database about customer’s type (MSB[Preferred, Non-Preferred] or Non-MSB) and chosen account type (checking, savings, money market [if applicable], consumer loan [if applicable], mortgage [if applicable]). It has the operations for setting the amount to be deposited, verifying deposit type and amount, and sending the amount back to the central database, opening or closing deposit slot.

Transfer Transaction-

The transfer transaction class is where the customer does their money transfer. It allows the atm machine to get information (via database communication) from the central database about customer’s type (MSB[Preferred, Non-Preferred] and chosen account type (checkings, savings, money market). It has the operations to set the amount being transferred, set the transferee account (consumer loan or mortgage), send the amount back to the central database to be added and deducted.

Check Balance Transaction-

The check balance transaction class is an optional function of the atm machine. It is where the customer can check their balance in their chosen account type. It has the operations to get and set the account type, get and set the balance of that account type, and display the balance.

ATM Machine System-

The ATM machine system class is where the system communicates internally and with the central database. It has the operations to verify the error, display errors, tell database communication to send a message to admin, exit session, display welcome screen.

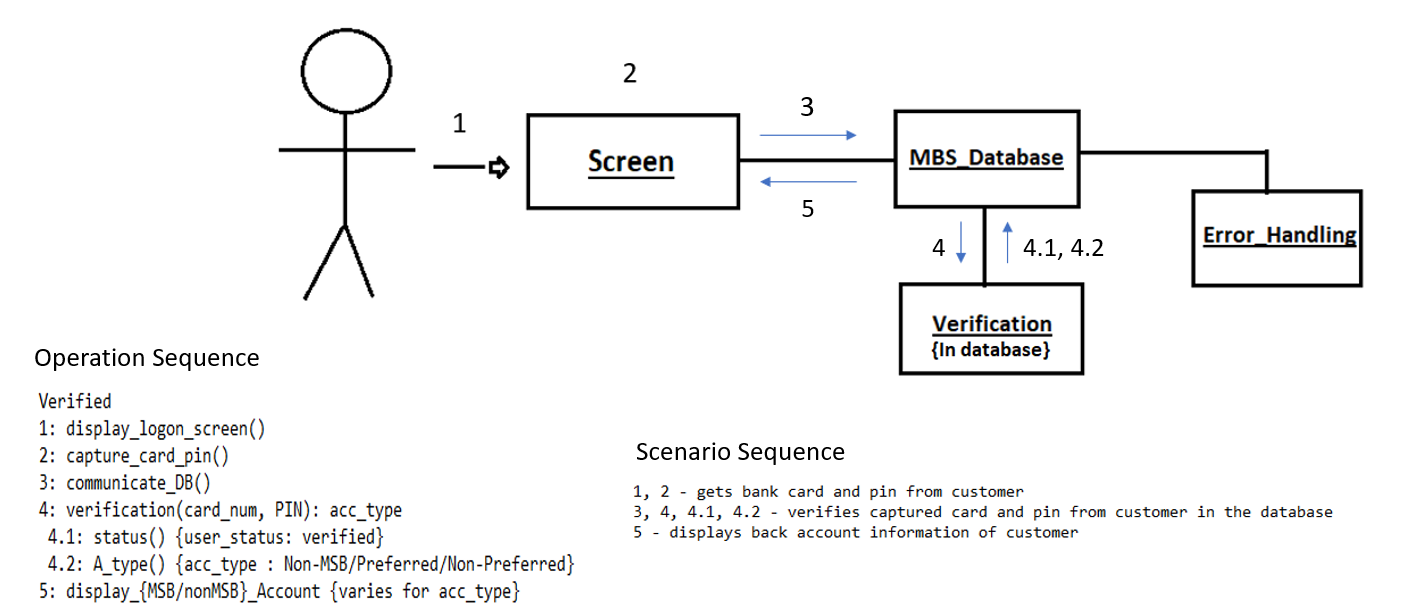
Print Receipt-

The print receipt class is an optional function of the atm machine. It allows the customer to choose whether they want to print a receipt or not. It has the operation of verifying printing of receipt, get and set information of customer/transaction/date/account/amount/balance, printing of receipt

---------------------------------------------------------------------------------------------------------------------

**Communication Diagrams**

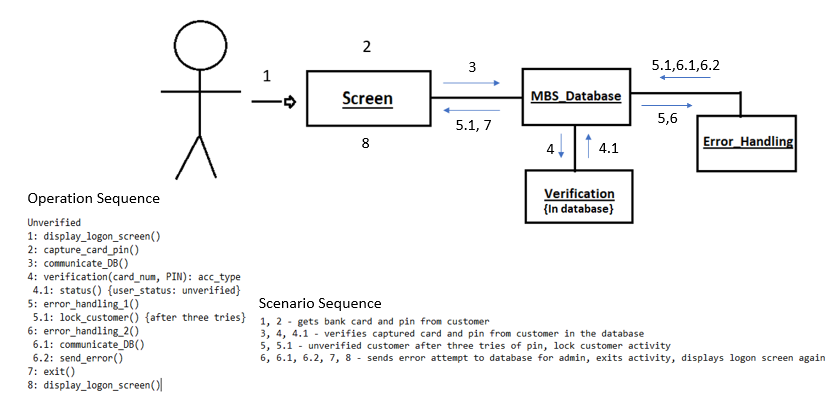
Customer Verification Use Case - verified customer

****

Textual Description:

The customer inserts their card into the machine and their unique pin. The machine then sends the captured bank card and pin to the database to verify the customer’s bank card and pin number connected to it. After verification is verified from the database, it sends back the account information of the customer if they are a Non-MSB or MSB customer via bank card. If they are an MSB customer, it will also send back the information if they are a preferred or non-preferred customer. Then after sending back information, the screen will display the customer’s account information with the options for further ATM activity such as choosing account type and transaction type.

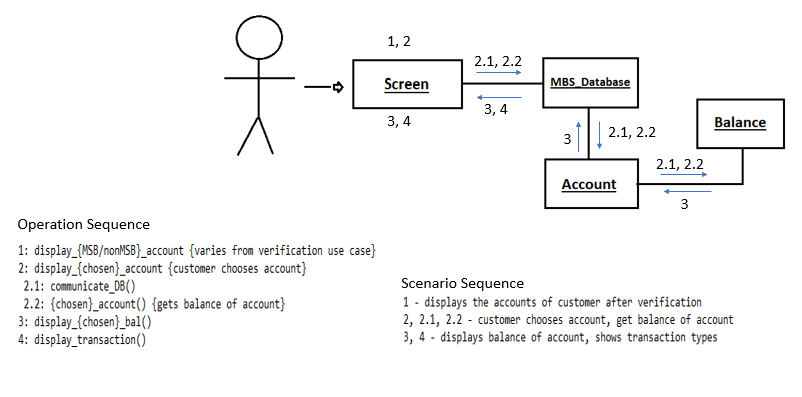
Customer Verification Use Case - unverified customer



Textual Description:

The customer inserts their card into the machine and their unique pin. The machine then sends the captured bank card and pin to the database to verify the customer’s bank card and pin number connected to it. After verification is unverified, it allows the customer to try inserting their pin code three times; however, after three tries and the pin code still does not match the one stored in the database,  the machine will lock the customer’s activity due to unauthorized access. The machine sends a message to the database for the admin to check about the attempt of access. Then the machine terminates the session and goes back to the welcome screen for other customers to use.

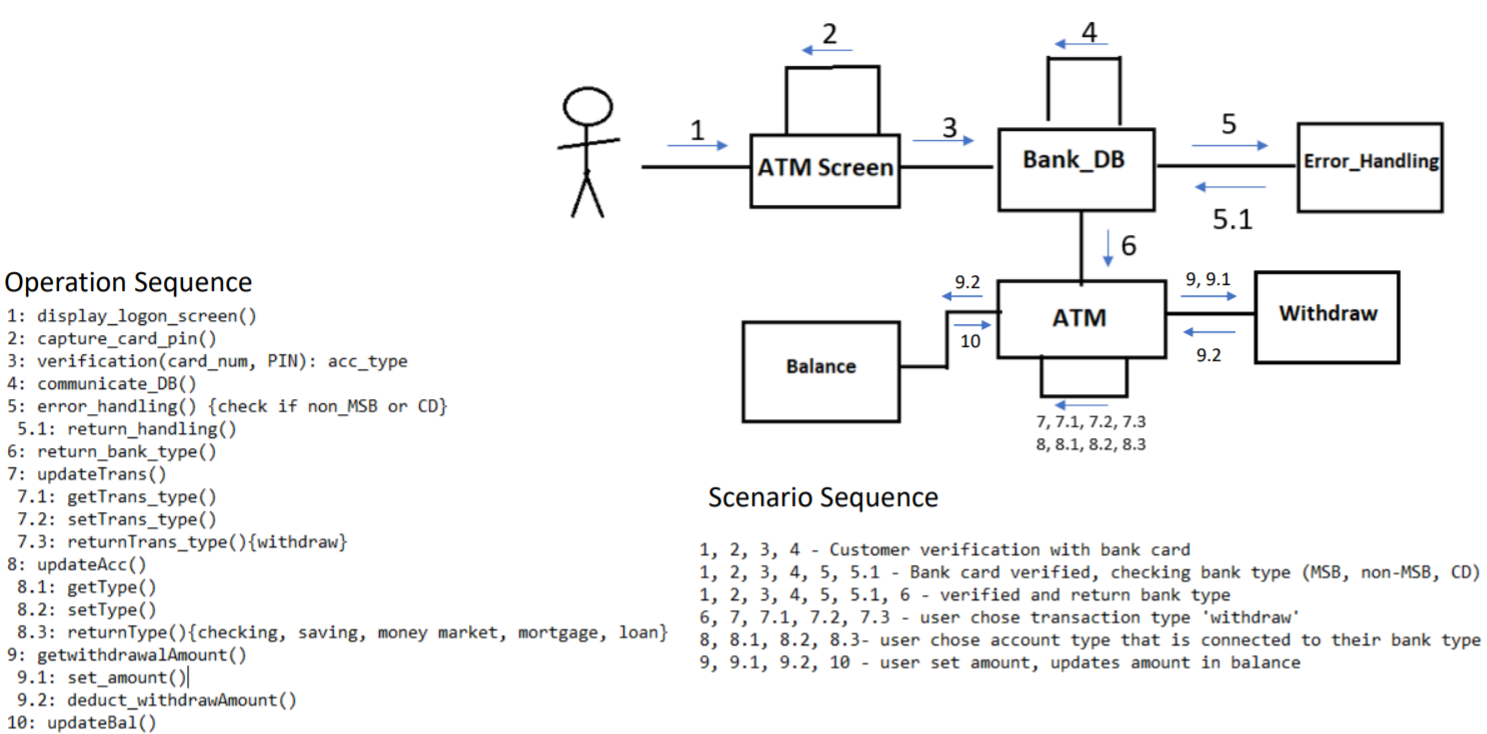
Access Account Types with Check Balance Use Case:



Textual Description:

The screen shows the customer’s accounts {checking, savings, money market; if applicable: consumer loan, mortgage}. The customer chooses an account type. The ATM machine sends the chosen account to the database via communication protocols and gets the information connected to the chosen account {account’s balance}. Then after checking the current balance of the chosen account, the screen will have the option to display the transaction screen for further ATM machine activity or exit which will terminate the current session.

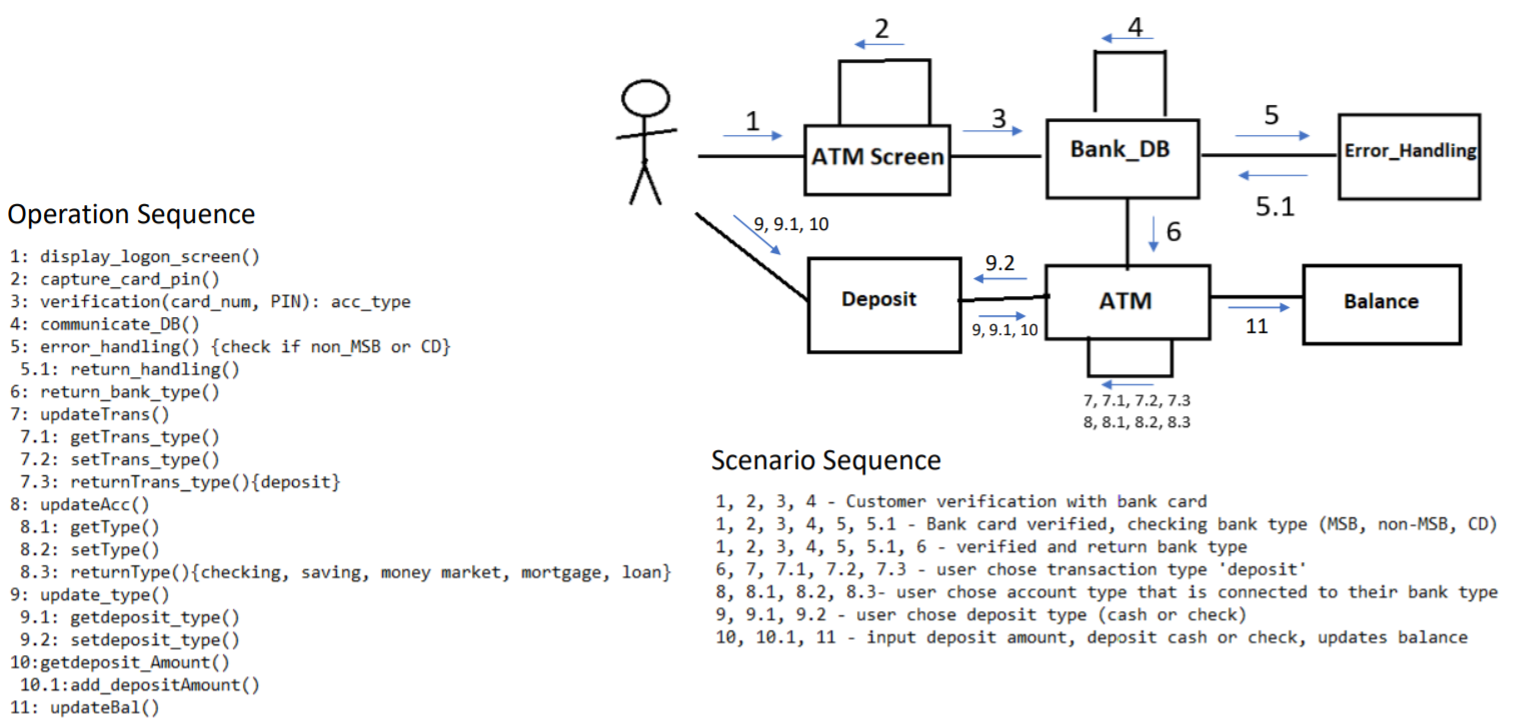
Money Withdrawal Use Case:



Textual Description:

The user chooses the {withdraw} transaction option and proceeds to choose an account from {checking, savings, money market; if applicable: consumer loan or mortgage}. User sets the desired amount, the machine sends it to the database via communication protocol, updates that account’s balance, and retrieves the money. If the customer is not an MSB customer, the machine also adds a fee for the database to process and remove from the account’s balance. If the amount to be withdrawn is over the amount currently in balance, the system will check if there is available balance in the consumer loan account; if there is available balance, it will deduct the amount from that account. But when there is no amount, the error handling system will ask the customer to change the amount into a sufficient amount to be withdrawn. The customer is then given the choice to print a receipt before exiting.

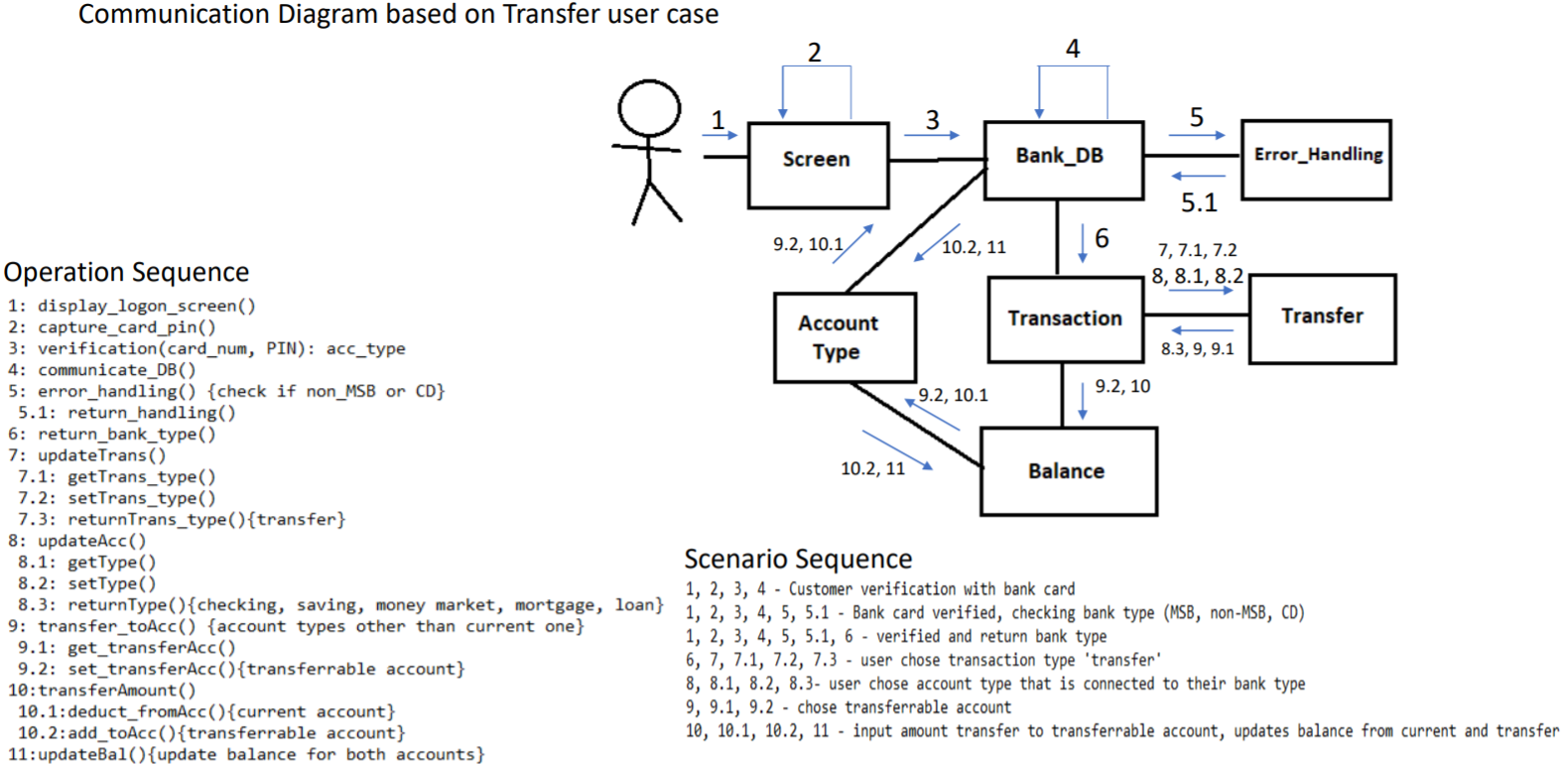
Money Deposit Use Case:



Textual Description:

The customer chooses the {deposit} transaction option and proceeds to choose an account from {checking, savings, money market; if applicable: consumer loan or mortgage}. The screen displays a warning sign, followed by the customer setting the deposit type{cash or check}. The deposit check hold will vary according to the verification use case. If the customer chooses check, the screen will ask for the amount then the customer sets how much they are depositing. The machine then receives the check, verifies the set amount with the check amount and sends an update to the database to change the account’s balance. If the customer is not an MSB customer, the machine also adds a fee for the database to process and remove from the account’s balance. If the customer chooses cash, the screen will ask for the amount then the customer sets how much they are depositing. The machine then receives the cash from the customer, verifies the set amount with the amount of cash, and sends an update to the database to change the account’s balance. The customer is then given the choice to print a receipt before exiting.

Money Transfer Use Case:

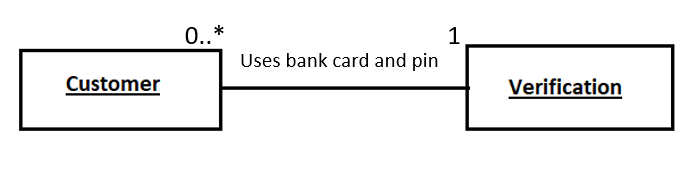


Textual Description:

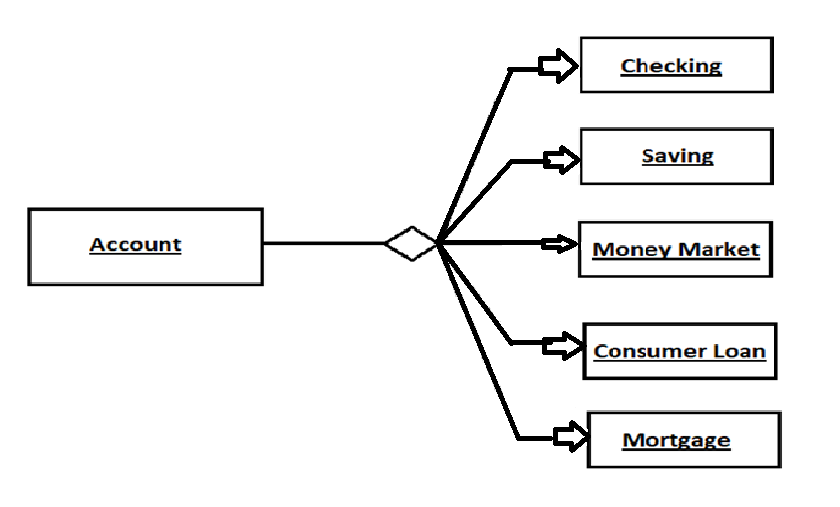
The customer chooses the {transfer} transaction option and proceeds to choose an account from {checking, savings, money market; if applicable: consumer loan or mortgage}. The machine shows accounts other than the current chosen account to transfer money to. The customer sets the desired amount to transfer which the machine then sends to the database to update both account balances via communication protocols. The error handling will check if the amount to be transferred is available in the current account. If it is available, it will continue with setting the amount and updating the balances of the chosen accounts. If it is not, the machine will ask the customer to change the amount to be transferred, then it continues with the setting the amount and updating balances of the chosen accounts. The customer is then given the choice to print a receipt before exiting.

**Relationship Diagrams**

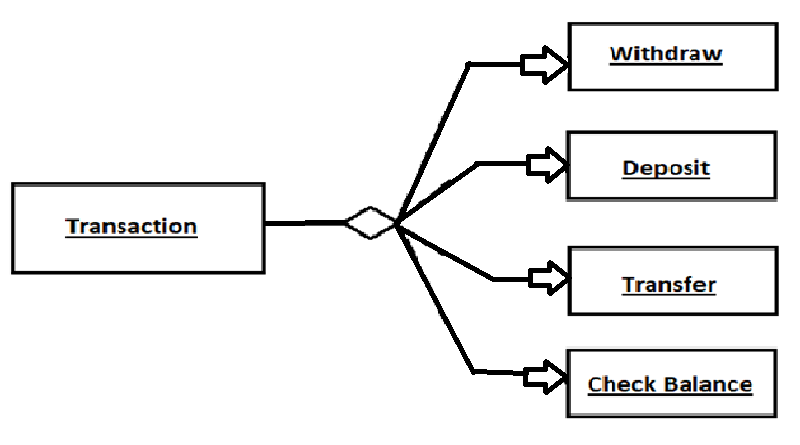
Verification Class Relationships

****

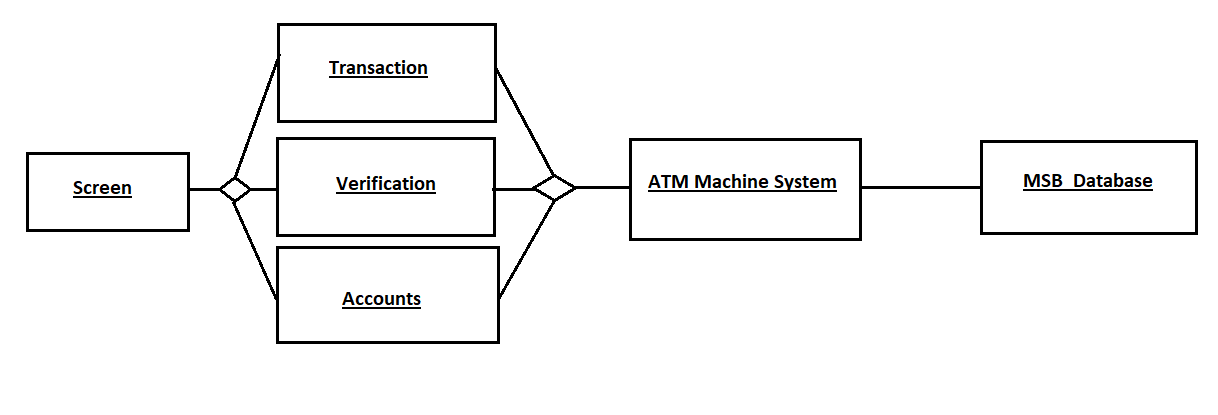
Access Account Class Relationships

****

Transaction Class Relationships



Communication Class Relationships



**Header Files**

#ifndef SCREEN\_H

#define SCREEN\_H

class Screen {

private: //Attributes

int x\_Coordinate;

int y\_Coordinate;

hex color;

int border;

int brightness;

int sharpness;

int pixel;

private: //Operation

void captureCardPin;

public: //Operations

void displayLogonScreen();

void displayVerification();

void exit();

void setCoordinate();

void setColor();

void setResolution();

void setBrightness();

void setSharpness();

int getCoordinate();

hex getColor();

int getResolution();

int getBrightness();

int getSharpness();

};

#endif

---------------------------------------------------------------------------------------------------------------------

#ifndef ERROR HANDLING\_H

#define ERROR HANDLING\_H

class Error\_Handling {

private; //Attribute

int errorType;

private; //Operations

void sendError();

void lockCustomer();

public; //Operations

void errorHandling1();

void errorHandling2();

void errorHandlingWith1();

void errorHandlingWith2();

void errorHandlingDep1();

void errorHandlingDep2();

void errorHandlingTrans1();

};

#endif

------------------------------------------------------------------------------------------------------------------

#ifndef DATABASE COMMUNICATION\_H

#define DATABASE COMMUNICATION\_H

class Database\_Communication {

private; //Attributes

string userStatus;

string accountType;

public; //Operations

void communicateDB();

void verification(int cardNum, bool PINStatus);

void setStatus();

void setAType();

string getStatus(string userStatus);

string getAType(string accountType);

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef ACCESS ACCOUNT\_H

#define ACCESS ACCOUNT\_H

class Access\_Account {

private; //Attribute

string accountType;

public; //Operations

void displayMSBAccount();

void displayNonMSBAccount();

void exit():

};

#endif

---------------------------------------------------------------------------------------------------------------------

#ifndef CHECKING’S ACCOUNT\_H

#define CHECKING’S ACCOUNT\_H

class Checking’s\_Account {

private; //Attributes

string userStatus;

string accountType;

public; //Operations

void communicateDB();

void checkingAccount();

void displayCheckingBal();

void displayCheckingTransScreen();

void exit();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef MONEY MARKET ACCOUNT\_H

#define MONEY MARKET ACCOUNT\_H

class Money\_Market\_Account {

private; //Attributes

string userStatus;

string accountType;

public; //Operations

void communicateDB();

void marketAccount();

void displayMarketBal();

void displayMarketTransScreen();

void exit();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef SAVING’S ACCOUNT\_H

#define SAVING’S ACCOUNT\_H

class Saving’s\_Account {

private; //Attributes

string userStatus;

string accountType;

public; //Operations

void communicateDB();

void savingAccount();

void displaySavingBal();

void displaySavingTransScreen();

void exit();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef CONSUMER LOAN ACCOUNT\_H

#define CONSUMER LOAN ACCOUNT\_H

class Consumer\_Loan\_Account {

private; //Attributes

string userStatus;

string accountType;

public; //Operations

void communicateDB();

void loanAccount();

void displayLoanBal();

void displayLoanTransScreen();

void exit();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef CERTIFICATE OF DEPOSIT (CD)\_H

#define CERTIFICATE OF DEPOSIT (CD)\_H

class Certificate\_Of\_Deposit\_(CD)\_Account {

private; //Attributes

string userStatus;

string accountType;

public; //Operations

void communicateDB();

void CDAccount();

void displayCDBal();

void exit();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef TRANSACTION\_H

#define TRANSACTION\_H

class Transaction {

private; //Attributes

string userStatus;

string accountType;

string transactionType

public; //Operations

void communicateDB();

void displayTransaction();

void setTransType();

void printReceipt();

void exit();

string transType();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef WITHDRAWAL TRANSACTION\_H

#define WITHDRAWAL TRANSACTION\_H

class Withdrawal\_Transaction {

private; //Attributes

string userStatus;

string accountType;

string transactionType;

private; //Operations

void setWithdrawalAmount();

int addFee();

public; //Operations

void communicateDB();

void displayWithdrawScreen();

void printReceipt();

void exit();

int getWithdrawAmount();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef DEPOSIT TRANSACTION\_H

#define DEPOSIT TRANSACTION\_H

class Deposit\_Transaction {

private; //Attributes

string userStatus;

string accountType;

string transactionType;

private; //Operations

void setDepositAmount():

void addFee();

public; //Operations

void communicateDB();

void displayDepositScreen();

void setCheck();

void setCash();

void printReceipt();

void exit();

int getCheck();

int getCash();

int getDepositAmount();

string warningSign();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef TRANSFER TRANSACTION\_H

#define TRANSFER TRANSACTION\_H

class Transfer\_Transaction {

private; //Attributes

string userStatus;

string accountType;

string transactionType;

private; //Operations

void displayTransfers();

int changeTransferAmount();

public; //Operations

void communicateDB();

void displayTransferScreen();

void setTransferAcc();

void setTransferAmount();

void printReceipt();

void exit();

int getTransferAmount();

string getTransferAcc();

string AType();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef CHECK BALANCE\_H

#define CHECK BALANCE\_H

class Check\_Balance {

private; //Attributes

string userStatus;

string accountType;

string transactionType;

public; //Operations

void communicateDB();

void setCheckingBal();

void setSavingsBal();

void setMarketBal();

void setLoanBal();

void setMortBal();

void setMSBAccount();

void setNonMSBAccount();

void goBackTrans();

int getCheckingBal();

int getSavingsBal();

int getMarketBal();

int getLoanBal();

int getMortBal();

string getMSBAccount();

string getNonMSBAccount();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef ATM MACHINE SYSTEM INTERFACE\_H

#define ATM MACHINE SYSTEM INTERFACE\_H

class ATM\_Machine\_System\_Interface {

private; //Attributes

string userStatus;

string accountType;

string transactionType;

int errorType;

private; //Operations

void displayNoCashScreen;

void withClose();

string sorry();

public; //Operations

void communicateDB();

void withOpen();

void openDeposit();

void closeDeposit();

int bals();

int balsAvailable();

int updateBal();

string accounts();

};

#endif

--------------------------------------------------------------------------------------------------------------------

#ifndef PRINT RECEIPT\_H

#define PRINT RECEIPT\_H

class Print\_Receipt\_H {

private; //Attributes

int transactionID;

int date;

int amount;

string accountType;

string customerName;

string transactionType;

public; //Operations

void communicateDB();

void setAmount();

void printReceipt();

int getAmount();

};

#endif