Montana Savings Bank

ATM System

Date: 2020

Document Version: 4

Table of Contents

Team Information……………………………………………………………………………....pg.3

introduction……...……………………………………………………………………………..pg.4

Use Case Diagram……………………………………………………………………………...pg.5

Card Verification Use Case Description.……………………………………………………….pg.6

Access Accounts​ Use Case Description…………………………………………………...…...pg.10

Transfer Funds Use Case Description.…………………………………………………………pg.19

Deposit Use Case Description……………………………………. …………………………...pg24

Withdraw Use Case Description.………………………………………………………………pg.28

System Interface Use Case Description.……………………………………………………….pg.32

Class Diagram………………………………………………………………………….............pg.36

Class Description.………………………………………………………………………..…….pg.38

Coding Implementation………………………………………………………………………..pg.xx

Glossary…………………………………………………………………………......................pg.xx

Index…………………………………………………………………………...........................pg.xx

Team Information:

# Michael Heerwig: Head of design and project workflow

I with planning, documentation, and design every task given to give the team a solid foundation and the other members as well as myself benefited greatly. The team communicated, worked efficiently, and finished tasks as intended.

# Melih Kartal: Head of documentation design & Software

The project was completed with extreme precision and efficiency thanks to the team assigned to the project. The team and I worked together to ensure that every aspect was completed in a reasonable timeframe. Each of the team members provided the group with the highest level of professionalism and teamwork to ensure the project would be completed on schedule and in the best possible state.

Arnell Pearson: Head of project workflow & code compliance

The team I was part of created an environment that allowed us to work quickly and efficiently in every aspect of the creation of this software. The communication between the members of the team were superb which allowed all of our tasks to be accomplished in a quick and reasonable time frame

Introduction

Client: Montana Savings Bank

Project Manager: Dr. Cyril S. Ku

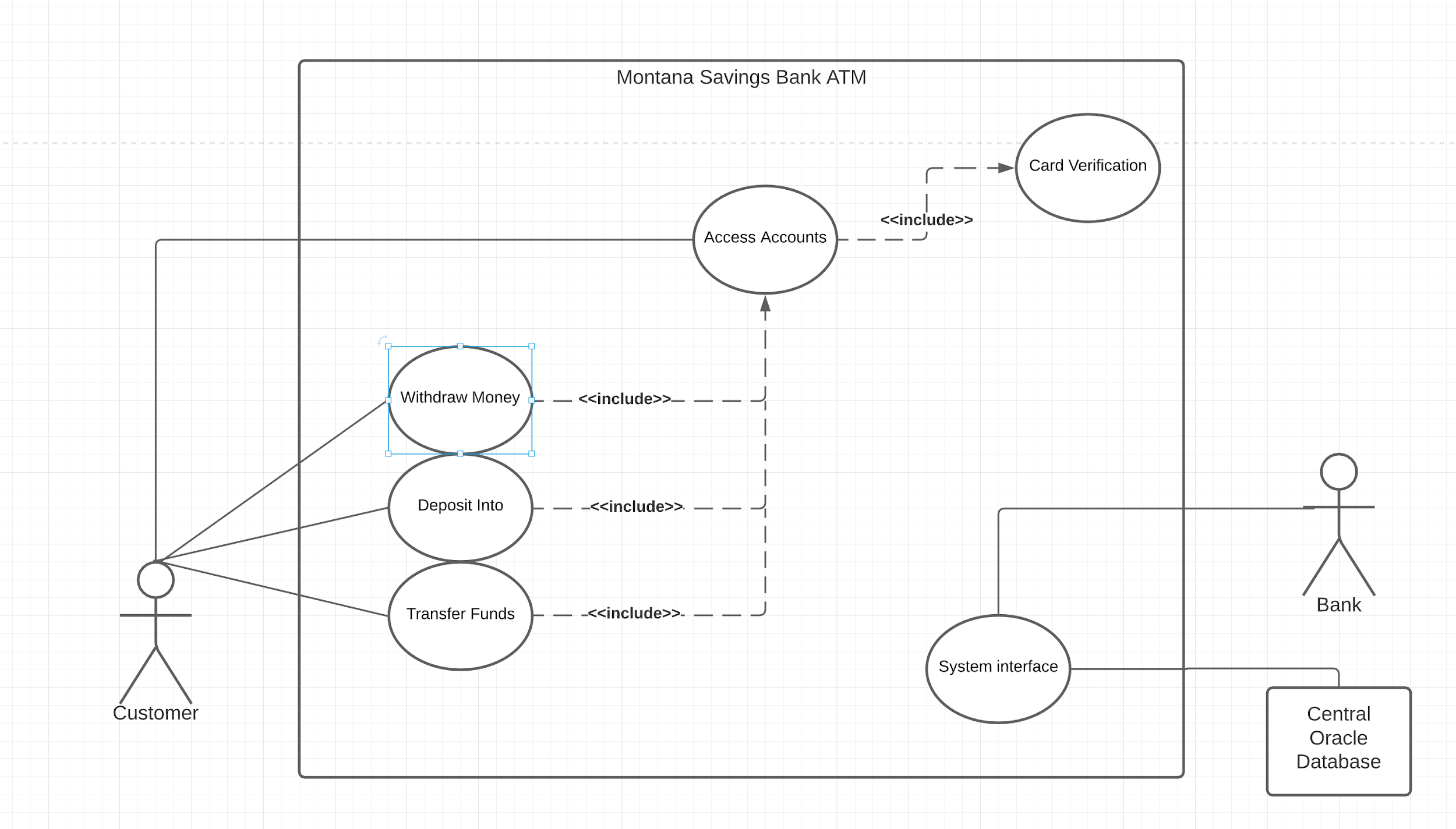
Agreement Date: 14 September, 2020

End Support Date: 24 September, 2020

This project breaks down the fundamentals of an Automated Teller Machine (ATM). It is broken up into key sections depending on the user/customer’s needs, and according to the rules the client desired implemented into their ATM. whether that be limiting the amount a user can withdraw or deposit all depending on that user’s current status with the bank which can be a preferred customer or a standard customer or even a non-customer.

This project outlines how a company would develop a system for the bank, and that system’s requirements. included in these requirements are specific charges for customers entirely dependent on their individual status as members, non-members, and preferred members of the bank. Each of these requirements is listed throughout the project as well as diagrams on how we implemented each to fit the client’s needs.

# Use Case Diagram



**Overview:**

***Card Verification***​: First use case; Security Verification

***Access Accounts***​: Second use case, choose what account to access the balance and make a transfer or a transaction on; the customer’s selections in this case, will determine what selections the customer has access to.

***Transfer Funds***: Third use case the customer will have access to the transfer use case which allows for funds to be moved between accounts (Directly tied to the Access Accounts Use Case for its functionality)

***Deposit***: Third use case customer will go through to deposit checks or cash to account customer selected on from Access Accounts Use Case

***Withdraw Money***​: Third use case the customer will go through to withdraw cash from account customer selected on from Access Accounts Use Case interface

***System Interface***​: Controls communication between the central database and Montana Savings Bank ATM use cases

***Database***​: Maintained by Oracle DBMS; All of the customers’ information is stored in the oracle database; bank to bank interaction is also handled here; Transactions are all gotten and sorted automatically.

# Use Case Descriptions: Card Verification

* Name:​ Card Verification
* Author:​ Melih Kartal Michael Heerwig Arnell Pearson
* Last Update:​ 10/20/2020
* Preconditions:​

The customer needs to present the their banking card to the ATM

get\_ATM\_card\_input()

* Dialog:​

1. Screen greets the Customer with “Welcome to the Montana Savings Bank”

**-display\_welcome\_message1(),. system\_interface()**

1. ATM gives customer instructions such as “Insert an ATM card now, to login to your account”.
2. The user may choose to insert their card or to click the bottom of the screen where it says “Non-msb customer” **non\_msb\_customer() , system\_interface()**
3. If user inserts their ATM card into the card reader and this provides access to the verification screen then:

* It will communicate with the database through system interface
* The user type will be set as **communicate\_With\_DB\_msb\_customer()**
* The user will be asked to verify its pin “Please enter your pin, now”
  1. Once the pin is entered, the pin will be verified through user interface with the database.
     + 1. if the pin is entered correctly **pin\_verification()** will be true
       2. else, the pin status is incorrect the **pin\_verification()** will be false : exception\_handling1()
       3. The database will give user three more attempts the enter the pin correctly, failure to do so will lock the account through system interface
* The customer can also exit the **pin\_verification()** at any time, this will exit the card verification: **exit()**
  + - Three incorrect entry of pin will display “You entered incorrect pin code for three consecutive times, please contact the bank 1-800-BANK” and will exit the screen : **exit()**

1. The status of **msb\_customer()** will then be checked with the database for **msb\_preferred()** or **msb\_nonpreferred() , communicate\_with\_DB**
   * + If the customer is **msb\_preferred()**
       - 1. msb\_preferred() status will return **true** to the database through user interface

Greeting message “Welcome, Dear… Preferred Customer” will appear on the screen to notify the user is a preferred customer

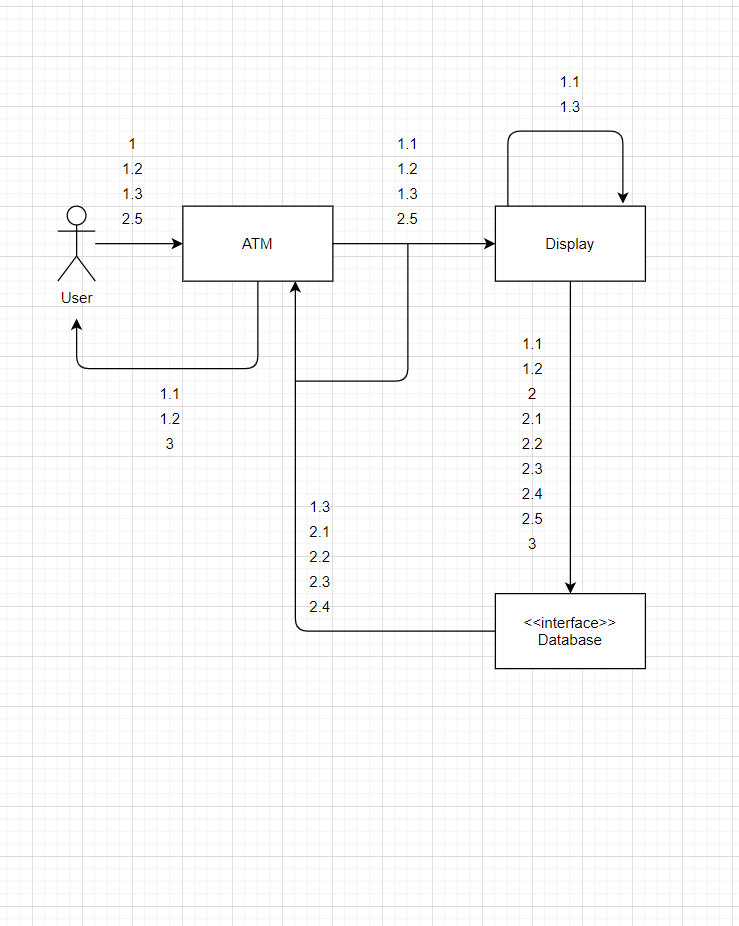
* + - else the customer is non preferred customer, **msb\_nonpreferred()** status will return true to the database through user interface

1. The customer changes their pin number , this will access the database through system interface
   * + Customer’s assigned\_pin will be changed
2. User terminates the session
   * + The screen prompts “Thank you for banking with Montana Savings Bank “
     + All data is erased upon exit

● Postconditions:​

1. User session terminates MSB ATM welcome screen appears for the next user -logout()
2. Upon successful verification, based on customer bank account and PIN, system displays appropriate menu screen from to form further transactions within​ account

**Card Verification Communication Diagram**



**Card verification Operation Sequence**

1.present\_card

1.1 system\_interface()

1.2 get\_pin():get

1.3 exception\_handling1()

2. pin\_verification():

2.1 non\_msb\_customer() bool

2.2 msb\_customer() bool

2.3 msb\_preferred() bool

2.4 msb\_nonpreferred() bool

2.5 communicate\_with\_DB\_msb\_customer()

3. exit()

zip\_code(): not needed

print\_receipt()

**Card verification Scenario Sequence**

1 , 1.1 , 1.2 , 1.3 , 1.1, 1.2, 1.3 , 1.1 , 1.2 , 1.3 : Invalid PIN - 3 TIME WRONG ENTRY

1 , 1.1 , 1.2 , 2 : ID password validated

1, 1.1 , 1.2 , 2, 2.2 , 2.3, 2.4, 2.5 , 3 : MSB CUSTOMER VALIDATION : PREFERRED OR NON-PREFERRED

1, 1.1 , 1.2, 2, 2.1 , 2.5 , 3 : NON MSB CUSTOMER VALIDATION

1, 1.1 , 1.2 , 3: User logon- logoff

# Use Case Description (Access Accounts​)​

* Name:​ Access Accounts
* Author:​ Melih Kartal, Michael Heerwig, Arnell Pearson
* Last Update:​ 10/20/2020
* Preconditions:​ The customer must successfully pass verification to gain access to the system/accounts(Under Card Verification Use Case)
* Dialog:​

1. The system will display the main menu page of accounts for MSB customers (preferred or non preferred customers):
   * Checking
     1. The customer can access this account through checkingAccount class
        1. The customer can view statements view\_statement() , communicate\_with\_DB
        2. The customer can filter transactions based on history: filter() , communicate\_with\_DB
        3. search transactions , search() , communicate\_with\_DB
        4. Transfer money from : Savings (TRANSFER FUNDS) use case , communicate\_with\_DB
        5. This operation calls the use case DEPOSIT INTO to have access to the accounts (checking or savings)
        6. This operation calls the use case WITHDRAW FROM to access to the accounts
     2. go back to menu, main\_menu()
     3. exit() , at anytime
   * Savings
     1. The customer can access this account through savingsAccount class
        1. The customer can view statements view\_statement() , communicate\_with\_DB
        2. The customer can filter transactions based on history filter() , communicate\_with\_DB
        3. search transactions , search() , communicate\_with\_DB
        4. Transfer money from : checking (TRANSFER FUNDS) use case, communicate\_with\_DB
     2. go back to menu, main\_menu()
     3. exit() , at anytime
   * Money Market
     1. The customer can access this account through monetMarket class
        1. The customer can view\_statement(), communicate\_with\_DB
        2. The customer can filter transactions based on history filter() , communicate\_with\_DB
        3. search transactions , search() , communicate\_with\_DB
        4. Transfer money from : Savings (TRANSFER FUNDS) use case , communicate\_with\_DB
     2. go back to menu, main\_menu()
     3. exit() , at anytime
   * Consumer loan
     1. The customer can access this account through consumerLoan class
        1. The customer can view\_statement(), communicate\_with\_DB
        2. The customer can filter transactions based on history filter() , communicate\_with\_DB
        3. search transactions , search() , communicate\_with\_DB
        4. Transfer money from : Savings (TRANSFER FUNDS) use case , communicate\_with\_DB
     2. go back to menu, main\_menu()
     3. exit() , at anytime
   * Mortgage
     1. The customer can access this account through Mortgage Account class
        1. The customer can view\_statement(), communicate\_with\_DB
        2. The customer can filter transactions based on history filter() , communicate\_with\_DB
        3. search transactions , search() , communicate\_with\_DB
        4. Transfer money from : Savings (TRANSFER FUNDS) use case , communicate\_with\_DB
     2. go back to menu, main\_menu()
     3. exit() , at anytime
   * Certificate of Deposit
     1. The customer can access this account through Cd Account class
        1. The customer can view\_statement(), communicate\_with\_DB
        2. The customer can filter transactions based on history filter() , communicate\_with\_DB
        3. search transactions , search() , communicate\_with\_DB
        4. Transfer money from : Savings (TRANSFER FUNDS) use case , communicate\_with\_DB
     2. go back to menu, main\_menu()
     3. exit() , at anytime
2. Else if the customer is not a bank member the system displays the main menu of Access Accounts ​for NON\_ACCOUNT\_HOLDER()
   * Non-Account-holder main menu allows Non-Account-Holder customer to access all available accounts
     1. Checking account:
        1. The customer can access this account through checkingAccount
           1. The customer can view statements view\_statement\_nonmsb() , communicate\_with\_DB
           2. The customer can filter transactions based on history filter\_nonmsb() , communicate\_with\_DB
           3. search transactions , search\_nonmsb() , communicate\_with\_DB
           4. This operation calls the use case DEPOSIT INTO to have access to the accounts (checking or savings)
           5. This operation calls the use case WITHDRAW\_FROM() to access to the accounts
        2. go back to menu, main\_menu()
        3. exit() , at anytime
     2. Savings account:
        1. The customer can access this account through SAVINGS\_ACCOUNT\_NONMSB class
           1. The customer can view statements view\_statement() , communicate\_with\_DB
           2. The customer can filter transactions based on history filter() , communicate\_with\_DB
           3. search transactions , search() , communicate\_with\_DB
           4. This operation calls the use case DEPOSIT\_INTO() to have access to the accounts (checking or savings)
           5. This operation calls the use case WITHDRAW\_FROM() to access to the accounts
     3. go back to menu, main\_menu()
     4. exit() , at anytime
3. Else , the customer can omit the previous menu items. This gives customer access to : **Profile**, and **Security** settings
   * The customer can access profile() through ACCOUNT class. This will grant access to the user to modify set\_address(), set\_phone\_number() , set\_personal\_email() , and set\_preferred\_language()
     1. The user chooses to update the **set\_address()** 
        1. The screen prompts “Please edit your address through the corresponding boxes and press ok when finished”
           1. The user enters the **set\_address()**

The data must be updated and sent back to the atm through system\_interface()

The user exits : exit() or logs out : logout()

* + - * 1. The user decides not to change the address and simply exits
    1. The user chooses to update **set\_phone\_number()**
       1. The screen prompts “ Please edit your phone number through the corresponding box and press ok when finished”
          1. The user enters the set\_phone\_number()

The data must be updated and sent back to the atm through system\_interface()

The user exits : exit() or logs out: logout()

* + - 1. The user decides not to change the phone number and simply exits
    1. The user chooses to update the set\_preferred\_language()
       1. The screen prompts “Please change the preferred language as you wish and press ok when finished”
          1. The user is prompted with couple different languages as he/she likes: german(), french(), italian(), russian(), chinese()
          2. The user chooses among the list

Upon selection the data is sent back to the database through system\_interface()

The user exits()

* + The customer can access Security() through ACCOUNT class. This will grant access to the user to modify set\_password() and set\_zip\_code()
    1. The user can choose to update the password: set\_password()
       1. The screen prompts “Please edit your password through the corresponding box and
          1. The user updates password
          2. The data is sent to the database through user\_interface()
          3. The user exits: exit() upon completion
       2. The user omits this step and simply exit()
* Postconditions​:

1. System displays checking account balance after the transaction

* display\_checking\_balance(), communicate\_with\_DB

1. System displays savings account balance after the transaction

* display\_savings\_balance(), communicate\_with\_DB

1. System displays money market balance after transaction

* display\_money\_market\_balance() ,communicate\_with\_DB

1. System displays CD balance ​after transaction

* display\_cd\_balance() ,communicate\_with\_DB

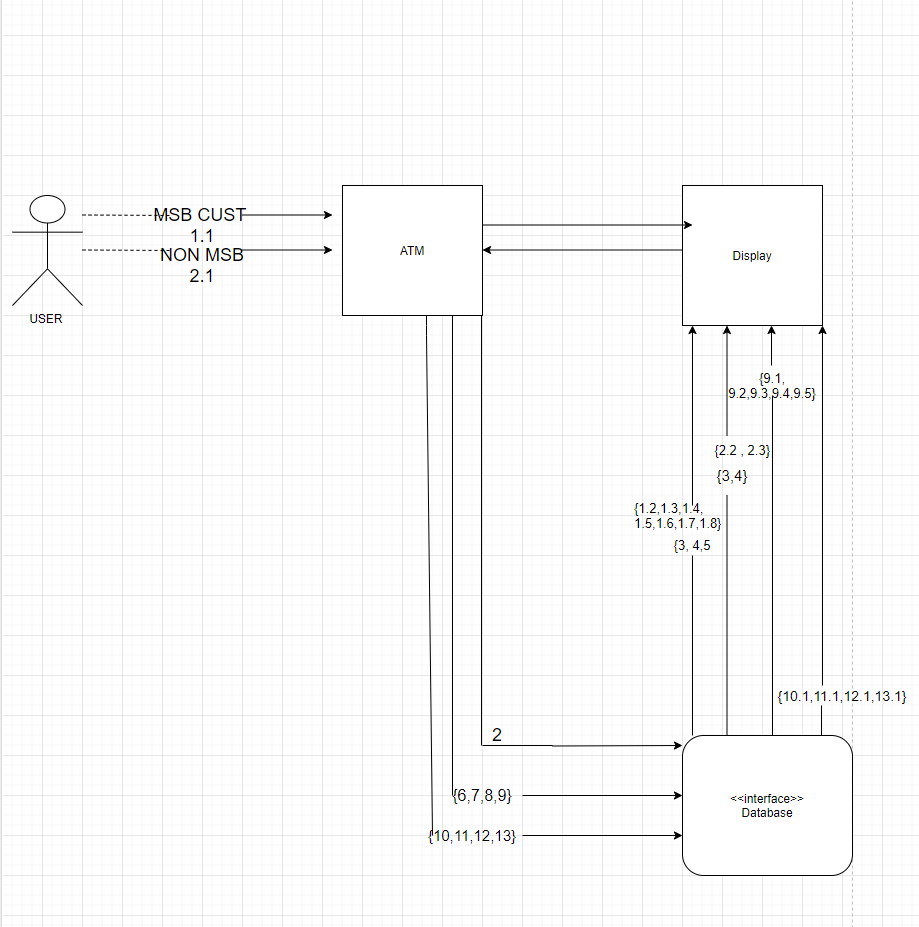
1. System displays mortgage account balance

* display\_mortage\_balance(), communicate\_With\_DB

1. User can choose to log-out anytime, without any operation

* exit()

**ACCESS ACCOUNTS COMMUNICATION DIAGRAM**



**Access Accounts OPERATION SEQUENCE**

1.1 view\_statement\_msb()

2 communicate\_with\_DB()

1.2 display\_checking\_balance()

1.3 display\_savings\_balance()

1.4 display\_money\_market\_balance()

1.5 display\_mortgage\_balance()

3 search()

4 filter()

4.1 filter\_msb()

2.1 view\_statement\_nonmsb()

2 communicate\_with\_DB()

2.2 display\_checking()

2.3 display savings()

3 search()

4 filter()

4.1 filter\_msb()

4.2 filter\_nonmsb()

5main\_menu()

6 set\_address()

7 set\_phone\_number() ,

8 set\_personal\_email() ,

9 set\_preferred\_language()

9.1 german(),

9.2 french(),

9.3 italian(),

9.4 russian(),

9.5 chinese()

10 set\_password()

10.1 password\_prompt()

11 set\_zipcode()

11.1 zipcode\_prompt()

12 logout()

12.1 logout\_display()

13 exit()

13.1 exit\_display()

**SCENARIO SEQUENCE:**

MSB ACCOUNT:

1.1 , 2 , 3 , 4 , 5 , 12, 13 : MAIN MENU ITEMS AND LOGOUT

1.1 , 2 , 1.2 , 3 , 4 , 4.1 , 5 , 12 , 12.1 , 13 , 13.1: DISPLAY CHECKING ACCT.

1.1 , 2 , 1.3 , 3 , 4 , 4.1 , 5 , 12 , 12.1 , 13 , 13.1 : DISPLAY SAVINGS ACCT

1.1 , 2 , 1.4 , 3 , 4 , 4.1 , 5 , 12 , 12.1 , 13 , 13.1 : DISPLAY MON.MARKET ACCT

1.1 , 2 , 1.5 , 3 , 4 , 4.1 , 5 , 12 , 12.1 , 13 , 13.1 : DISPLAY MORTGAGE ACCT

1.1, 2, 5 , 6 , 12 , 12.1 , 13 , 13.1 : SET A DIFFERENT ADDRESS

1.1, 2, 5 , 7 , 12 , 12.1 , 13 , 13.1 : SET A DIFFERENT PHONE NUMBER

1.1, 2, 5 , 8 , 12 , 12.1 , 13 , 13.1 : SET A DIFFERENT EMAIL

1.1, 2, 5 , 9 , 9.1 , 12 , 12.1 , 13 , 13.1 : SET LANGUAGE TO GERMAN

1.1, 2, 5 , 9 , 9.2 , 12 , 12.1 , 13 , 13.1 : SET LANGUAGE TO FRENCH

1.1, 2, 5 , 9 , 9.3 , 12 , 12.1 , 13 , 13.1 : SET LANGUAGE TO ITALIAN

1.1, 2, 5 , 9 , 9.4 , 12 , 12.1 , 13 , 13.1 : SET LANGUAGE TO RUSSIAN

1.1, 2, 5 , 9 , 9.5 , 12 , 12.1 , 13 , 13.1 : SET LANGUAGE TO CHINESE

1.1, 2, 5 , 10, 10.1 , 12 , 12.1 , 13 , 13.1 : SET A DIFFERENT PASSWORD

1.1, 2, 5 , 11, 11.1 , 12 , 12.1 , 13 , 13.1 : SET A DIFFERENT ZIP CODE

NON-MSB ACCOUNT:

2.1 , 2 , 3 , 4 , 5 , 12, 13 : MAIN MENU ITEMS AND LOGOUT

2.1 , 2 , 2.2 , 3, 4 , 4.2 , 5 , 12 , 12.1 , 13 , 13.1 : DISPLAY CHECKING ACCT

2.1 , 2 , 2.3 , 3, 4 , 4.2 , 5 , 12 , 12.1 , 13 , 13.1 : DISPLAY SAVINGS ACCOUNT

# Use Case Descriptions/Narratives (Transfer Funds​)​

* Name:​ Transfer Funds
* Author: Melih Kartal, Michael Heerwig, Arnell Pearson​
* Last Update:​ 10/20/2020
* Preconditions:​

- The customer needs to be verified from the CARD VERIFICATION

use case to further access

- The customer needs to be MSB customer (preferred or non-preferred)

- The access account must request access to TRANSFER FUNDS use case

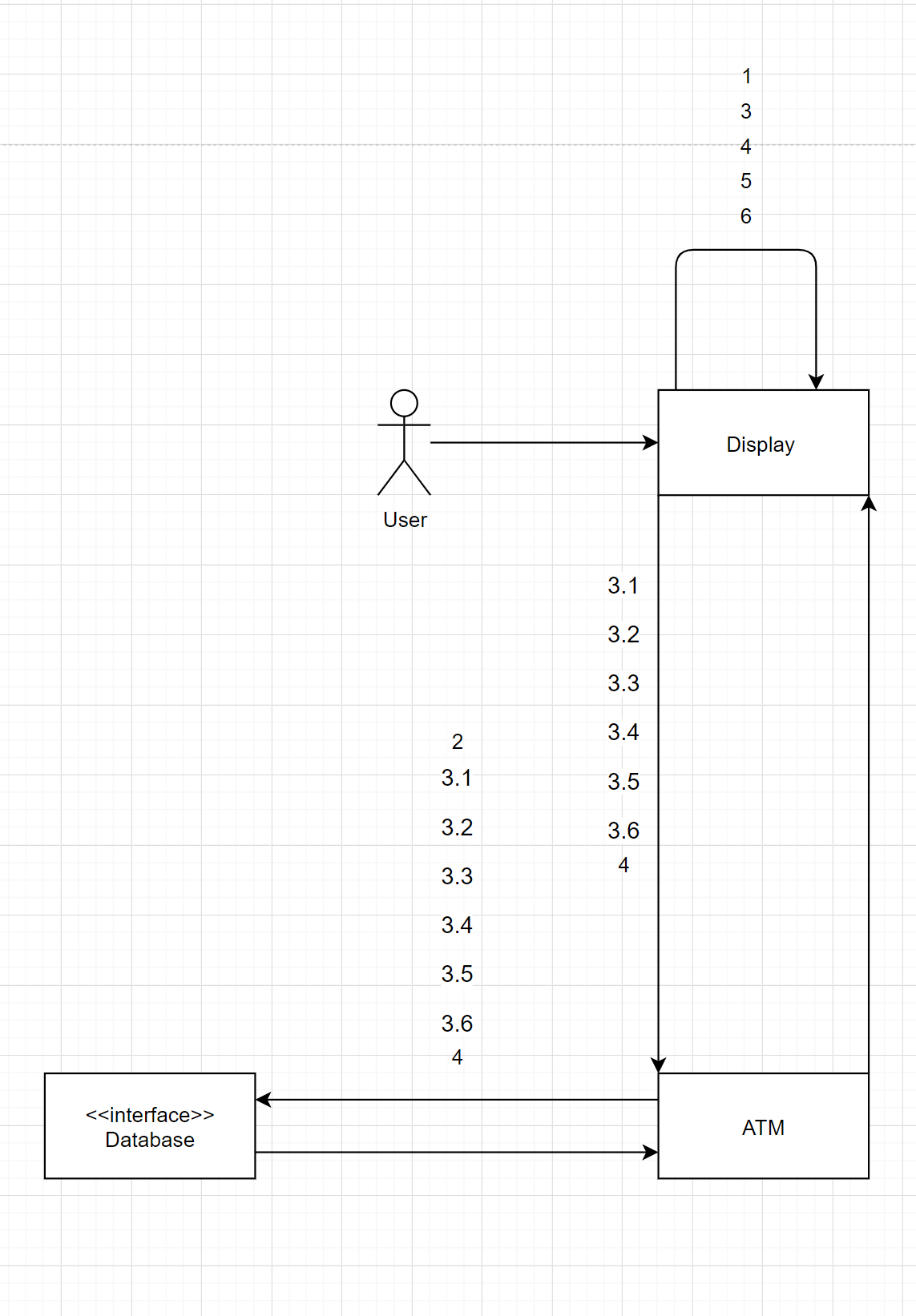
* Dialog:​

1. The system displays the main menu of the TRANSFER FUNDS display\_transfer\_funds\_screen()
2. The user needs to input “FROM:” to proceed, this calls ACCESS ACCOUNTS use case
   1. The user can choose from which account
      1. FROM:CHECKING from\_checking()
         1. To savings account : ACCESS ACCOUNTS use case is reached
            1. transfer to savings transfer\_money\_to\_savings()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: checking account
            4. The customer goes back to main menu exit()
         2. To money market account: ACCESS ACCOUNTS use case is reached
            1. transfer to money market transfer\_money\_to\_money\_market()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: checking account
            4. The customer goes back to main menu exit()
         3. To consumer loan account:ACCESS ACCOUNTS use case is reached
            1. transfer to consumer loan transfer\_money\_to\_consumer\_loan()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: checking account
            4. The customer goes back to main menu exit()
         4. To mortgage account: ACCESS ACCOUNTS use case is reached
            1. transfer to mortgage account transfer\_money\_to\_mortgage\_account()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: checking account
            4. The customer goes back to main menu exit()
      2. FROM: SAVINGS from\_savings()
         1. To checking account : ACCESS ACCOUNTS use case is reached
            1. transfer to checking transfer\_money\_to\_checking()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: savings account
            4. The customer goes back to main menu exit()
         2. To money market account: ACCESS ACCOUNTS use case is reached
            1. transfer to money market transfer\_money\_to\_money\_market()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: savings account
            4. The customer goes back to main menu exit()
         3. To consumer loan account:ACCESS ACCOUNTS use case is reached
            1. transfer to consumer loan transfer\_money\_to\_consumer\_loan()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: savings account
            4. The customer goes back to main menu exit()
         4. To mortgage account: ACCESS ACCOUNTS use case is reached
            1. transfer to mortgage account transfer\_money\_to\_mortgage\_account()
            2. The account is updated by withdrawing from one account to another : WITHDRAW MONEY
            3. The user interface is reached to update the final status: savings account
            4. The customer goes back to main menu exit()

Postconditions:​

1. ATM displays a confirmation screen as the account is updated communicate\_with\_DB
2. customer can choose to print a statement : ACCESS\_ACCOUNTS()
3. User can choose to log out at any time : exit()

**TRANSFER FUNDS COMMUNICATION DIAGRAM**



**TRANSFER FUNDS Operation Sequence**

1 system\_interface()

2 communicate\_with\_DB()

3 display\_transfer\_funds\_screen()

3.1 from\_checking()

3.2 from\_savings()

3.3 transfer\_money\_to\_savings()

3.4 transfer\_money\_to\_money\_market()

3.5 transfer\_money\_to\_consumer\_loan()

3.6 trasnfer\_money\_to\_mortage\_accounts()

4 enter\_amount()

5 exit()

6 logout()

**TRANSFER FUNDS SCENARIO SEQUENCE**

1, 3, 4, 5, 6 : UI for customer

2, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 4 : information sent to database

3.1, 3.3, 4 , 5 , 6 : transfer from checking to savings

3.1, 3.4, 4 , 5 , 6 : transfer from checking to money market

3.1, 3.5, 4 , 5 , 6 : transfer from checking to consumer loan

3.1, 3.6, 4 , 5 , 6 : transfer from checking to mortgage account

3.2,3.4, 4 , 5 , 6 : transfer from savings to money market

3.2,3.5, 4, 5, 6 transfer from savings to consumer loan

3.2, 3.6, 4, 5, 6 transfer from savings to mortgage accounts

1, 4, 5 user timeout

1, 6 user logout

# Use Case Descriptions/Narratives (​ Deposit Into)

* Name:​ Deposit Into
* Author:​ Melih Kartal, Michael Heerwig, Arnell Pearson​
* Last Update:​ 10/20/2020
* Preconditions:​

1. The user (customer) must have a successful verification to the system, refer to Card​ (through checking, savings, money market account)​

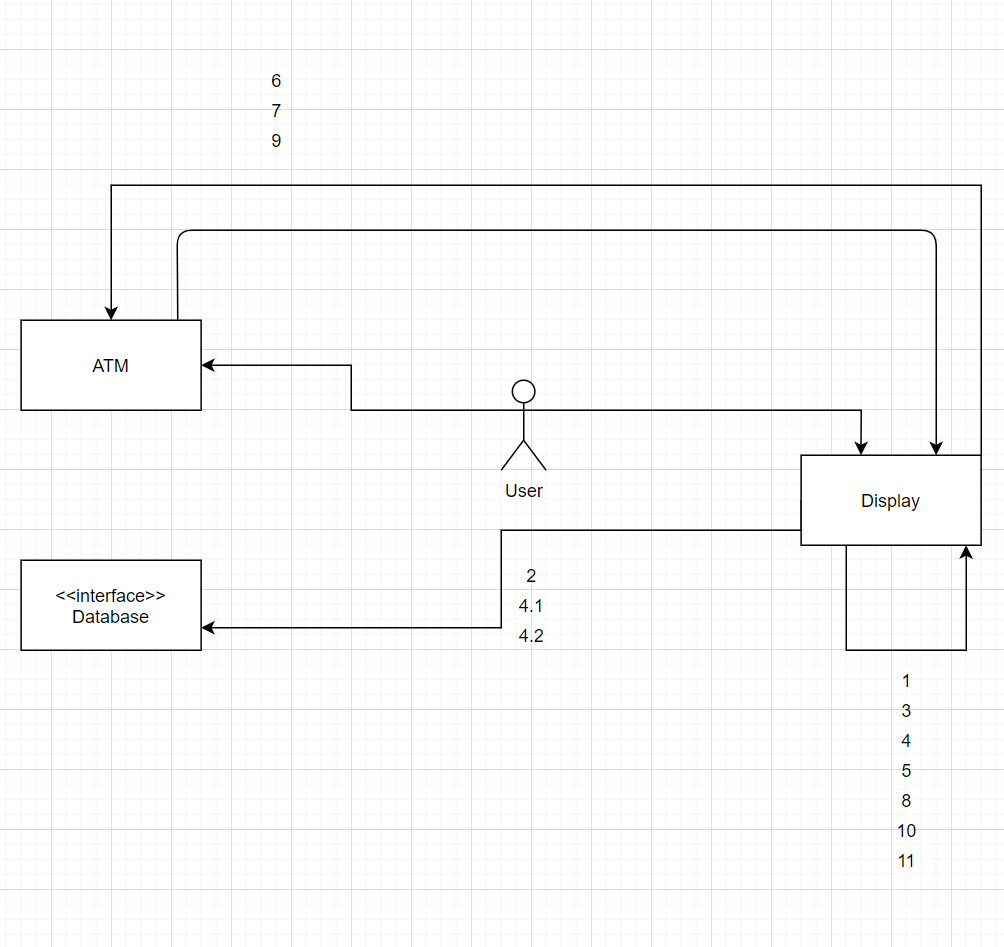
* Dialog:​

1. The ATM displays the Deposit Main Menu *– Deposit\_Main\_Menu()* ​ with the type of account the customer Selects *–*​ *Access\_Account()*
   1. checking
   2. saving
   3. money market
2. If the customer chooses cash deposit
3. Then the System prompts user to enter how much money they will deposit into account *–*​ *deposit\_prompt()*
4. The user will then enter amount through keypad on the ATM *– enter\_amount()*​
5. ATM will then open the cash slot *– open\_Dropoff\_slot()*​ ​the ATM will now tell the user to put the money into the cash slot *– close\_Dropoff\_slot()*​
6. The ATM will now count the money and confirm the amount -Confirm\_Cash\_value()
7. If the customer chooses the check deposit Use Case
8. Then the system will prompt the user into entering how much money they will deposit into the selected account*–*​ *deposit\_prompt()*
9. Then the user will enter the amount through the keypad on the ATM
10. Then the ATM opens the deposit slot *– open\_Dropoff\_slot()*​ ​and directs the Customer to input the check into the slot*– close\_Dropoff\_slot()*​
11. Then the System verifies the validity of the check. -Verify\_Check\_Integrity
12. Then the System checks if the amount entered is equal to the check’s amount. -Confirm\_Check\_Value
13. If the Check Verification and values are correct
    1. If Customer is Non-preferred, than no holds will be placed on check deposits*–*​ *communicate\_with\_DB*
    2. If the customer isPreferred, than a three-day hold will be placed on all check deposits*–*​ *communicate\_with\_DB*
    3. If the customer is a Non-Account-Holder, then charge a service fee of $3 per check deposit*– communicate\_with\_DB*​
14. If the System sees that the amounts are wrong or invalid, then the screen displays a pop-up message saying the amount the customer entered and deposited (either check or cash) does not match. The amount from deposited cash/check is: (print amount to ATM screen) -exception\_handling2()
15. Then Ask the customer to authenticate or override the Value they had entered*– confirmation\_screen()*​
16. If the user authenticates the amount to the correct value, then continue through the deposit process.
17. Otherwise If the user overrides the system
    1. Then the ATM reopens dropoff slot to return cash/checks to user*– open\_Dropoff\_slot()*​ ​
    2. Then after the customer takes cash/check *– close\_Dropoff\_slot()*​
    3. Then return to Deposit Money Main menu*– Deposit\_screen()*​
    4. System prompts user to print transaction receipt upon end of session
    5. Then If the Customer chooses to print out deposit receipt *– print\_receipt()*​
    6. Then the screen shows option to return to the main menu of Access Accounts ​*–*​ *Access\_Account\_screen()* ​or terminate the session *– logout()*​

* Postconditions:​

1. User successfully deposits cash or checks into account under (checking, saving, or money market, and transaction is automatically gathered into central DB ​- communicate\_with\_DB​
2. the Customer is returned to the main menu of the Access Accounts Use Case ​*–*​ *Access\_Account\_screen()*
3. Then the system prints out a receipt (or doesn’t) and the user's session terminates normally. Then the system returns to ATM homepage *-*​ *print\_receipt(), logout()*

**DEPOSIT INTO COMMUNICATION DIAGRAM**



**DEPOSIT INTO Operation Sequence**

1 deposit\_Main\_Menu()

2 communicate\_with\_DB()

3 deposit\_prompt()

4 enter\_amount()

4.1 confirm\_cash\_value()

4.2 confirm\_check\_value()

5 exception\_handling2()

6 open\_dropoff\_slot()

7 close\_dropoff\_slot()

8 confirmation\_screen()

9 print\_receipt()

10 exit()

11 logout()

**DEPOSIT INTO SCENARIO SEQUENCE**

1, 3, 4, 5, 6, 7, 8, 9, 10, 11 - UI for customer

2, 4, 4.1, 4.2, 5 10 database atm communication/interaction

1, 2, 4, 4.1, 4.2, 5 (return) - error handling

1, 11 - user logout

# Use Case Descriptions/Narratives (Withdraw Money​ )​

* Name​: Withdraw Money
* Author​: Melih Kartal, Michael Heerwig, Arnell Pearson​
* Last Update:​ 10/20/2020
* Preconditions​:

1. The user must have passed the Card Verification Use Case first.
2. The user next must have navigated to Checking, Savings, Money Market, or Customer loan through the ACCESS ACCOUNT use case.
3. Lastly the user must have pressed Withdraw in the system

₋ ​ ​

* Dialog​:

1. The screen will now display the Withdraw Money main menu ​*– display\_Withdraw\_Main\_Menu()* ​All available accounts will be shown here
   1. checking
      1. *–*​ *Access\_Account()*
   2. saving
      1. *–*​ *Access\_Account()*
   3. money market​
      1. *–*​ *Access\_Account()*
2. Now the system prompts user to enter how much money they want to withdraw​*– withdraw\_prompt()*
3. The customer will now enter the amount of money they want to withdraw in this transaction
4. However the customer’s withdrawal amount must be in multiples of $10​ if an invalid choice is made then -exception\_handling3()

* If the user is a Preferred MSB Customer, they are granted overdraft protection (MSB preferred customers have no limit on the amount of money they can withdraw from any individual account)
* If the user attempts to withdraw any more money than is permitted by the individual account they selected, the money must be withdrawn from their consumer loan account however it can not exceed the limit established by the loan account ​*– communicate\_with\_DB*

1. If this limit is exceeded then the system prompts the user to try again with the available loan amount displayed.

* If the customer is Non-preferred, then they can withdraw money from their consumer loan account

1. In this case the Non-preferred customer has no overdraft protection ​*– communicate\_with\_DB*
2. Also If any Non-Customers are charged a service fee of $3 per withdrawal

* The customer then confirms the amount to withdraw
* If the customer is Non-preferred they will have a $500 limit per day that they can draw from

1. If the limit is exceeded then the System will display an error and prompt the user to try again while displaying the current value available ​-exception\_handling4()
2. then the system returns the user to the Withdraw Money main screen​*– display\_Withdraw\_Main\_Menu()*

* Any and all preferred customers have no withdraw limit
* The ATM System draws from its cash system to provide the amount the customer requested and pushes it out of its cash slot -*open\_Cash\_slot()*.

1. The ATM ​system closes cash slot once money is taken by the user​*– close\_Cash\_slot()*

* The customer choice to print a receipt ​*–print\_receipt()*
* The session is terminated *– logout()*​

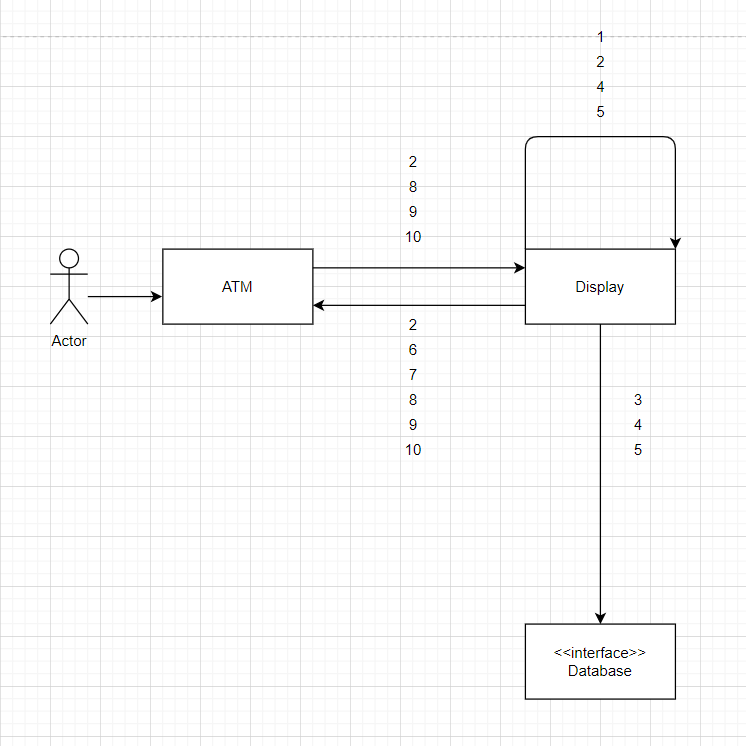
1. If the user wants another transaction they will have to go through the CARD VERIFICATION Use Case again.

* Postconditions​:
* The customer has taken money from their account and is received by the database -*communicate\_with\_DB*

1. checking
2. savings
3. money market

* The session has terminated *– logout()*​
* The Customer now either has a receipt or does not ​*- print\_receipt(), logout()*
* The system has returned to the Card Verification menu

**WITHDRAW MONEY COMMUNICATION DIAGRAM**

****

**Operation Sequence**

1 display\_Withdraw\_Main\_Menu()

2 withdraw\_prompt()

3 communicate\_with\_DB()

4 exception\_handling3()

5 exception\_handling4()

6 open\_Cash\_slot()

7 close\_Cash\_slot()

8 print\_receipt()

9 logout()

10 exit()

**WITHDRAW MONEY SCENARIO SEQUENCE**

1, 2, 4, 5, 6, 7, 8, 9, 10 - user ATM interaction

2, 3, 4, 5, 10 - ATM database interaction

1, 2, 4, 5 (return) - error handling

1, 9 - user log out

# Use Case Descriptions/Narratives (System Interface​ )​

* Name:​ System Interface
* Author:​ Melih Kartal, Michael Heerwig, Arnell Pearson​
* Last Update:​ 10/20/2020
* Preconditions:​ The system interface needs to be called from other use cases such as (ATM VERIFICATION, ACCESS ACCOUNTS , TRANSFER FUNDS, DEPOSIT INTO, WITHDRAW MONEY)
* Dialog:​

The system interface is the gateway between the ATM, the bank, and the Customer : It verifies whether the customer is a MSB Customer (preferred or non-preferred, also verifies pin and the zipcode) by reference variables.

* The use case is called initially by the CARD VERIFICATION use case
  + Allocate a memory in bank database for the customer

communicate\_with\_DB() [non\_msb\_customer , msb\_customer, msb\_preferred]

If the user type is invalid, the system interface will send exit message to the ATM machine

Then the system intakes pin: communicate\_with\_DB() to verify from the database.

If any information occurs to be wrong, the system will display “Wrong Input” exception\_handling1()

* The use case is later called by ACCESS ACCOUNTS use case
  + It will communicate with the database through account types((checking,savings, money market, CD, consumer loan, mortgage accounts) communicate\_with\_DB()
* The use case is called by DEPOSIT MONEY use case
  + This use case is responsible for the transactions of depositing check or cash, mainly for input and validating that input communicate\_with\_DB()
  + If the input is invalid, will exit()
* The use case is called by WITHDRAW MONEY use case
  + This use case checks the overall balance of accounts communicate\_with\_DB()
    - If the user tends to withdraw more money than the account balance, will return insufficient\_funds()
    - The system can deliver message “max amount is 500$ “ if customer tends to withdraw more than 500$
    - also will exit at command exit()
* The use case is called by TRANSFER FUNDS use case
  + This use case verifies the amount that needs to be transferred is present at the customer's account
    - Connects the central database with communicate\_with\_DB()
    - The system can deliver message “insufficient funds” when the account does not have enough balance communicate\_with\_DB()
    - Exit at the finishing transaction exit()
* Postconditions:​

₋After completion of the communicate\_with\_DB() variables will be initialized as the user input such as if money is withdrawn the amount is reduced or if deposited the amount will be added

If the user inputs the wrong pin number repeatedly, the CARD\_VERIFICATION will be declined, therefore *–*​exception\_handling1()will occur. This will exit() the application

₋If the user inputs wrong in or wrong zip code repeatedly, the CARD\_VERIFICATION will be declined, therefore -exception\_handling1() will occur. This will exit() the application

-

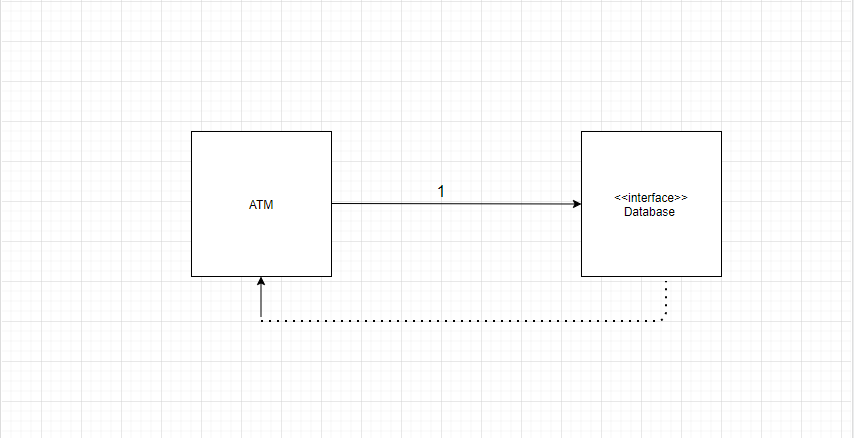
₋ Sends back account balances for all available accounts (checking, savings, money market, CD, consumer loan, mortgage accounts) to Access Accounts Use Case​

₋ Customer information stored and automatically updated from transactions made in

₋ User session terminates normally *-*​ *logout()*

₋ User returns to main menu of Access Accounts Use Case ​ *-*​ *ACCESS\_ACCOUNTS\_MENU*

**SYSTEM INTERFACE COMMUNICATION DIAGRAM**



**SYSTEM INTERFACE OPERATION SEQUENCE:**

1: communicate\_with\_DB()

2 verify\_preferred()

3 verify\_non\_preferred()

4 verify\_non\_msb()

5 pin\_verification()

6 exception\_handling1()

7: main\_menu()

8: logout()

**SYSTEM INTERFACE SCENARIO SEQUENCE**

1, 6 , 7 , 8 : CONNECTING TO DATABASE

1 , 2 , 5 , 7 , 8 : PREFERRED CUSTOMER RESPONSE

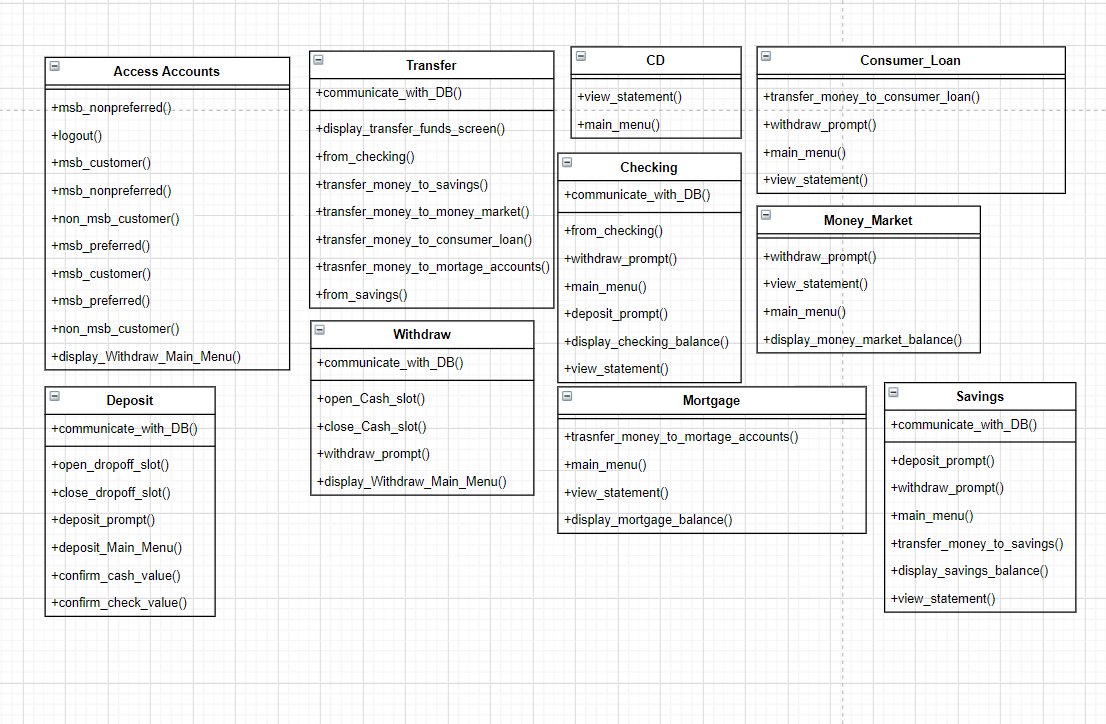
1 , 3 , 5 , 7 , 8 : VERIFY NON PREFERRED

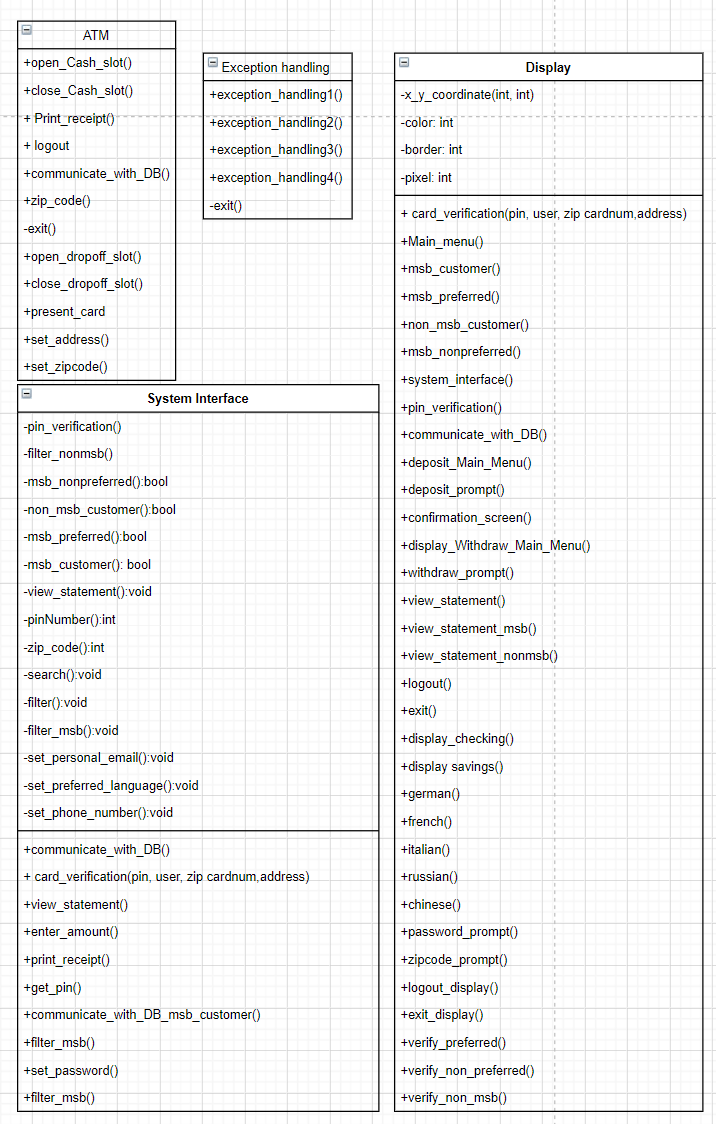
1 , 3 , 4 , 6 , 7 : NON-MSB CUSTOMER RESPONSE

1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 : ANY CUSTOMER LOGIN WRONG PIN 1 TIME EXIT

1 , 2 , 3 , 4 , 5 , 6 , 5 , 6 , 5 , 6 , 7 , 9 : ANY CUST LOGIN WRONG PIN 3 TIMES

**CLASS DIAGRAMS:**



****

# Class Diagram Descriptions

# System Interface: This class will handle all of the communications between any individual use case and the​ central database. This class is used to communicate accounts and all user information and card authentication

**Withdraw:** subclass of the Transaction Class. It is responsible for operations of the Deposit Money Use Case.

**Deposit:** subclass of Transactions Class. It is responsible for operations of the​ Deposit Money Use Case.

**Checking:** subclass of Accounts Class. This holds the total value of the checking account and the balance after transactions are made

**Transfer:**  subclass of Transactions Class. It is responsible for all operations of the Transfer​ Funds Use Case’.

**Display:** superclass of the Accounts class and Transactions Class. This is the first class the user will encounter​ on startup the screen presents the Card verification process. This class holds all screen settings relating to the other classes and displays their information for the user to follow.

**Accounts:** subclass of Display Class, superclass of Saving, Checking, Withdraw, Mortgage, Money\_Market,​ CD, and Consumer\_Loan. This class holds the display for accounts and all related information, after card verification the user is able to access various accounts, whichever one they select(so long as the user type allows).

**Transactions:** subclass of Display Class. Tells the system how the process of a transaction works takes the input and talks to the necessary classes.​

**Saving:** subclass of Accounts Class. This class holds the display and total value for the savings account and the balance after transactions are made

**Mortgage:** subclass of Accounts Class. This class holds the display and total value for the mortgage account and the balance after transactions are made

**Money\_Market:** subclass of Accounts Class. This class holds the display and total value for the money market account and the balance after transactions are made

**Consumer\_Loan:** subclass of Accounts Class. This class holds the display and total value for the consumer loan and the balance after transactions are made

**Certificate\_Deposit:** subclass of Accounts Class. This class holds the display and total value for the CD and the balance after transactions are made

**Exception Handling:** The class responsible for handling all errors. Any failures, validation errors or illegal entries that would end in failed attempts and sends messages to help the customer when they are encountering issues.

**ATM:** Superclass of everything all events take place here.

Glossary

***Access Accounts***​: Second use case, choose what account to access the balance and make a transfer or a transaction on; the customer’s selections in this case, will determine what selections the customer has access to.

**Accounts:** subclass of Display Class, superclass of Saving, Checking, Withdraw, Mortgage, Money\_Market,​ CD, and Consumer\_Loan. This class holds the display for accounts and all related information, after card verification the user is able to access various accounts, whichever one they select(so long as the user type allows).

**ATM:** Superclass of everything all events take place here.

***Card Verification***​: First use case; Security Verification

**Certificate\_Deposit:** subclass of Accounts Class. This class holds the display and total value for the CD and the balance after transactions are made

**Checking:** subclass of Accounts Class. This holds the total value of the checking account and the balance after transactions are made

**Consumer\_Loan:** subclass of Accounts Class. This class holds the display and total value for the consumer loan and the balance after transactions are made

***Database***​: Maintained by Oracle DBMS; All of the customers’ information is stored in the oracle database; bank to bank interaction is also handled here; Transactions are all gotten and sorted automatically.

***Deposit***: Third use case customer will go through to deposit checks or cash to account customer selected on from Access Accounts Use Case

**Display:** superclass of the Accounts class and Transactions Class. This is the first class the user will encounter​ on startup the screen presents the Card verification process. This class holds all screen settings relating to the other classes and displays their information for the user to follow.

**Exception Handling:** The class responsible for handling all errors. Any failures, validation errors or illegal entries that would end in failed attempts and sends messages to help the customer when they are encountering issues.

**Money\_Market:** subclass of Accounts Class. This class holds the display and total value for the money market account and the balance after transactions are made

**Mortgage:** subclass of Accounts Class. This class holds the display and total value for the mortgage account and the balance after transactions are made

**Saving:** subclass of Accounts Class. This class holds the display and total value for the savings account and the balance after transactions are made

***System Interface***​: Controls communication between the central database and Montana Savings Bank ATM use cases

# System Interface: This class will handle all of the communications between any individual use case and the​ central database. This class is used to communicate accounts and all user information and card authentication

**Transactions:** subclass of Display Class. Tells the system how the process of a transaction works takes the input and talks to the necessary classes.​

***Transfer Funds***: Third use case the customer will have access to the transfer use case which allows for funds to be moved between accounts (Directly tied to the Access Accounts Use Case for its functionality)

**Transfer:**  subclass of Transactions Class. It is responsible for all operations of the Transfer​ Funds Use Case’.

**Withdraw:** subclass of the Transaction Class. It is responsible for operations of the Deposit Money Use Case.

***Withdraw Money***​: Third use case the customer will go through to withdraw cash from account customer selected on from Access Accounts Use Case interface

# 

Index

***Access Accounts***​ page 2, 6, 7, 10, 13, 15, 16, 17, 20, 21, 22, 26, 33, 34, 42, 43, 44.

**Accounts** page 2, 6, 7, 10, 11, 13, 14, 15, 16, 20, 21, 22, 23, 24, 26, 29, 33, 34, 39, 42, 43, 44.

**ATM** page 1, 4, 7, 8, 14, 22, 25, 26, 28, 30, 32, 33, 39, 42, 44.

***Card Verification*** page 2, 6, 7, 8, 9, 10, 20, 29, 30, 33, 39, 42, 44.

**Certificate\_Deposit** page 39, 42, 44

**Checking** page 11, 13, 15, 17, 18, 19, 20, 21, 24, 25, 26, 29, 30, 33, 34, 39, 42, 45

**Consumer\_Loan** page 21, 22, 24, 37, 39, 42, 45

***Database*** *page 7, 8, 15, 24, 28, 30, 32, 33, 36, 38, 42, 45*

***Deposit*** *page 2, 4, 6, 11, 12, 13, 24, 25, 26, 28, 33, 39, 42, 43, 45*

**Display***page 8, 11, 13, 15, 17, 18, 19, 20, 22, 24, 25, 26, 29, 30, 32, 33, 39, 42, 43, 45*

**Exception Handling** page 38, 39, 42, 45

**Money\_Market** page 15, 17, 20, 22, 24, 38, 39, 42, 45

**Mortgage** page 12, 15, 17, 18, 21, 22, 24, 33, 34, 39, 42, 45

**Saving** page *7, 8, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 24, 25, 26, 29, 30, 33, 34, 39, 42, 45*

***System Interface***  *page 2, 7, 8, 32, 33, 34, 35, 38, 39, 42, 43, 45*

**Transactions** page *7, 8, 11, 12, 13, 33, 34, 39, 42, 43, 45*

***Transfer Funds*** *page 2, 6, 11, 12, 13, 19, 20, 23, 24, 33, 37, 43, 45*

**Transfer** page 2, 6, 11, 12, 13, 20, 21 , 22, 23, 24, 33, 34, 37, 39, 42, 43, 45

**Withdraw** page 2, 4, 7, 11, 13, 14, 20, 21, 22, 28, 29, 30, 32, 34, 37, 39, 40, 41, 42

***Withdraw Money*** *page 7, 20, 21, 28, 29, 30, 31, 32, 33, 41, 42*

// File Name: systemInterface.h // direct interface with the database and the customer accounts

#include “exceptionhandling.h” // uses the header files to detect any problems occur

#include “AccessAccounts.h”

Class systemInterface

{

public:

void pin\_verification ();

void filter\_nonmsb();

bool msb\_nonpreferred;

bool non\_msb\_customer;

bool msb\_preferred;

bool msb\_customer;

int pinNumber ;

int zipCode;

int cardNumber;

void view\_statement ();

void\_search ();

void filter();

void filter\_msb();

void set\_phone\_number();

void set\_personal\_email();

void set\_preferred\_language();

void set\_password();

private:

void communicate\_with\_DB ()

int card\_verification(pinNumber, zipCode, cardNumber) ;

void get\_pin() ;

void communicate\_with\_DB\_msb\_customer();

//File Name : ATM.cpp // the main CPP file ,, the main program of the atm

#include “systemInterface.h”

#include “Display.h”

#include “exceptionhandling.h ”

#include “Deposit.h”

#include “

class atmConfiguration //stands for ATM class

{

public:

protected:

int zip\_code;

void open\_Cash\_slot();

void Cash\_slot();

void Print\_receipt();

logout();

void communicate\_with\_DB();

void exit();

void open\_dropoff\_slot();

void close\_dropoff\_slot();

void present\_card();

void set\_adress();

void set\_zipcode();

//File Name : exceptionhandling.h // exception handling is error handling each exception is different from another

class exceptionHandling

{

public:

void exception\_handling1();

void exception\_handling2();

void exception\_handling3();

void exception\_handling4();

exit():

}

//File Name : Display.h // this is simply the display settings of the ATM screen

class atmDisplay

{

public:

int x\_ y\_coordinate;

int color;

int border;

int pixel;

private; // user settings of the ATM screen

void Main\_menu();

void communicate\_with\_DB();

void deposit\_Main\_menu();

void display\_checking();

void deposit\_prompt();

void confirmation\_screen();

void display\_withdraw\_main\_menu();

void view\_statement();

void view\_statement\_nonmsb();

void german();

void french();

void italian()

void russian ();

void chinese ();

void password\_prompt ();

void zipcode\_prompt ();

void logout\_display();

void exit\_play();

void verify\_preferred();

void verify\_nopreferred();

void verify\_nonmsb();

}

//Filenname : Deposit.h //this header file is responsible for the Deposit transaction

class Deposit

{

public:

communicate\_with\_DB();

private:

open\_dropoff\_slot()

open\_dropoff\_slot();

deposit\_prompt();

deposit\_Main\_Menu();

bool conirm\_cash\_value();

bool confirm\_check\_value();

}

//File Name: AccessAccounts.h // this file is responsible for accessing customer accounts

#include “withDraw.h”

#include “savings.h”

#include “Checking.h”

#include “CD.h”

#include “moneyMarket.h”

#include “mortgage.h”

#include “consumerLoan.h”

class accessAccounts

{

public:

bool msb\_nonpreferred;

bool non\_msb\_customer;

bool msb\_preferred;

bool msb\_customer;

bool non\_msb\_customer;

logout();

}

//File Name: Transfer.h //this header file is responsible for the transfer operation

class Transfer

{

public:

void communicate\_with\_DB();

private:

void display\_transfer\_funds\_screen();

void from\_checking();

void transfer\_money\_to\_saving();

void transfer\_money\_to\_money\_market();

void transfer\_money\_to\_consumer\_loan();

void transfer\_money\_to\_mortgage\_accounts();

void from\_savings();

}

//Filename : withDraw.h //this header file is responsible for the withdrawal operation

class Withdraw

{

public:

void communicate\_with\_DB();

void\_display\_withdraw\_main\_menu();

private:

void withdraw\_prompt();

void open\_Cash\_slot();

void close\_Cash\_slot();

}

//File Name: savings.h //this header file is responsible for savings account

class Savings

{

public:

void\_communicate\_with\_DB();

void\_display\_savings\_main\_menu();

void main\_menu();

void display\_savings\_balance();

void view\_statement();

private:

void deposit\_prompt();

void withdraw\_prompt();

void transfer\_money\_to\_savings();

}

//File Name: Checking.h //this header file is responsible for the checking account

class Checking

{

public:

void communicate\_with\_DB();

void main\_menu();

private:

void from\_checking;

void withdraw\_prompt();

void deposit\_prompt();

void display\_checking\_balance();

void view\_statement();

}

//File Name: CD.h //this header file is responsible for the Certificate of deposit account

class CD

{

public:

void view\_statement();

void main\_menu();

}

//File Name : moneyMarket.h //this header file is responsible for the money market acct

class moneyMarket

{

public:

void communicate\_with\_DB();

void main\_menu();

void view\_statement();

void withdraw\_prompt();

void display\_money\_market\_balance();

}

//File Name: mortgage.h //this header file is responsible for the mortgage account

class mortgage

{

public:

void communicate\_with\_DB();

void main\_menu();

void view\_statement();

void display\_mortgage\_balance();

}

File Name: consumerLoan.h //this header file is responsible for the consumer loan acct

class consumerLoan

{

public:

void communicate\_with\_DB();

void main\_menu();

private:

void transfer\_money\_to\_consumer\_loan();

void withdraw\_prompt();

void view\_statement();

}