ATM Project

By Steven Neri and Collin Brooks

Table of Contents

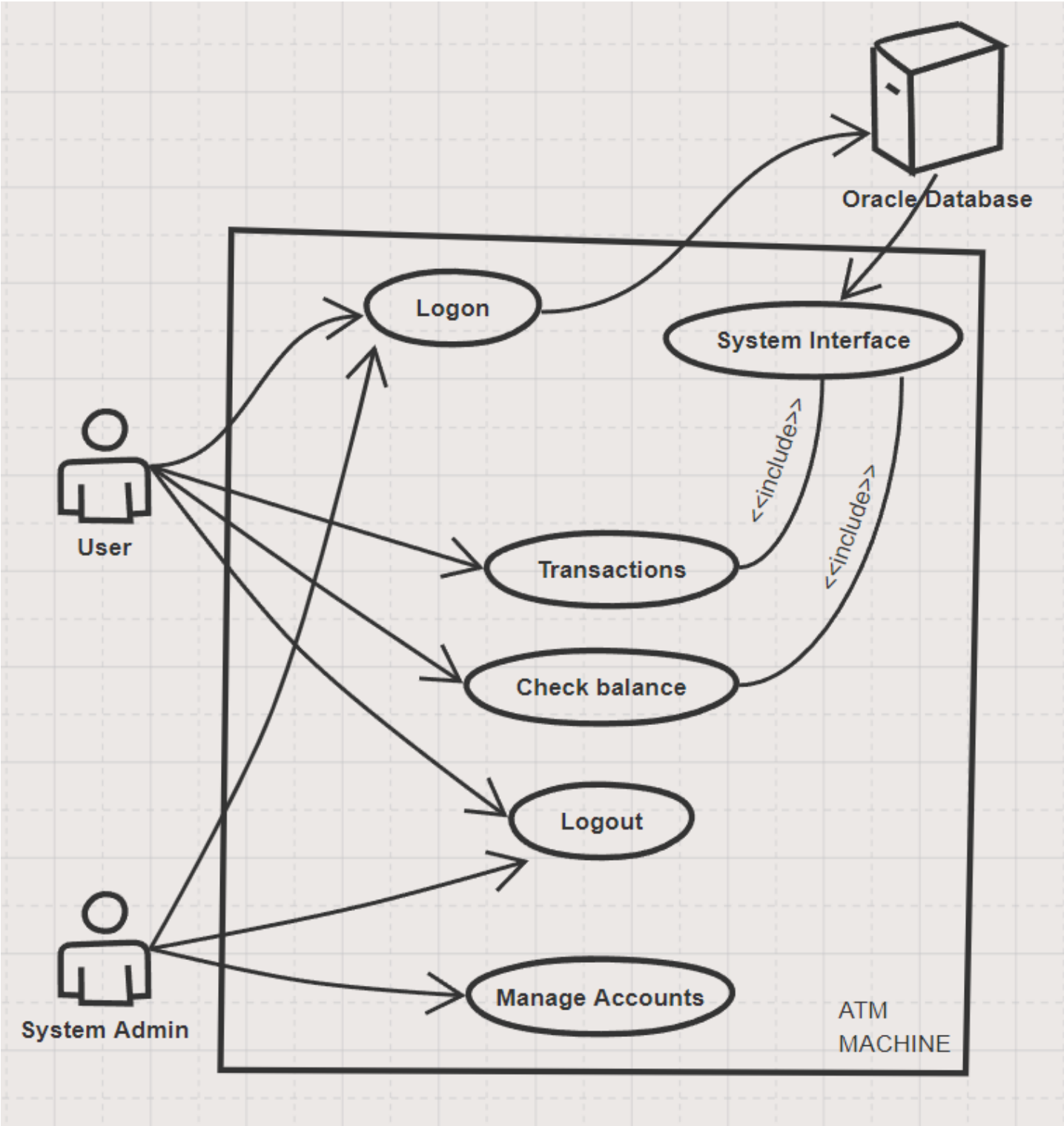
Introduction

A local bank named “Montana Savings Bank” (MSB) in a small town in Montana performs their operations with person to person contact. Recently, there have been some cost cutting measures and MSB has hired our consulting company to create an Automated Teller Machine (ATM). This will provide many benefits for the bank economically: save labor costs, eliminate manual errors, and provide 24-hour service to its customers. Our machine focuses on the software building of the ATM and concentrates on the functionalities. Our ATM describes interactions between the ATM, Customer, and the Central Database.

Team Information

The beginning of the project we decided to create a project to our understanding and did not follow recommended UML instructions that were presented to us. This was a big problem because our main goal was very unclear and hard to follow. We eventually collaborated and did research on the formatting of UML. We soon understood where we were wrong and redid the entire part of project #1. Going through the stress of reducing our entire project not only helped us with our grade portion of the project, but also brought us close together as a team and after project #1 the work felt flawless. Minimal errors, a full understanding of the procedures, and a healthy collaborative effort was displayed during this time.

Use Case Diagram



Use Case Descriptions

All attributes will be highlighted in yellow!

All operations will be highlighted in green!

Name: Logon

Author: User

Last Update: 4/6/22

Pre-Conditions: The user will access the ATM and will determine if they are a MSB Customer or not.

Dialog:

- The user will click the welcome button that is always present on the machine when using the machine for the first time. public welcomeScreen()
- There is an option that the user may be able to enter an ATM card that would automatically sign into their account.public readCard()
    - The ATM will ask the user to input the ATM card into the card slot with an animation of the card simulating the process (The animation will include the direction that the card needs to be placed into the machine) (Authentication of the user). If the card is not accepted or input wrong the user will be asked to try again or return to the welcome menu. account. public cardAnimation()
    - The user will then be prompted with a pin number to enter the ATM. public get/setPIN()
    - The ATM will then retrieve the customer's account userAccount : Object
- The ATM will then ask the user if they are a MSB Customer with a selection of: isMSBCustomer : Boolean

- Yes
- No

- If the customer is not a MSB Customer and wants to be one, then they will be prompted with the following to enter into the database (This will also be needed for first time users no matter the label):
  - Name public get/setName()
  - Date of birth public get/setDateOB()
  - Social security number public get/setSocialSN()
  - Account number public get/setAcctNumb()
  - Phone number public get/setPhoneNumb()
  - Username public get/setUsername()
  - Password public get/setPassword()
  - Email public get/setEmail()

- All information will be sent to the database and be held as record for the next use and the user is now shown what benefits they have being a part of the community and a Preferred Customer. userAccount : Object
  - Customers no longer have a $3 fee when doing any sort of transactions.
  - There is no hold placed on their deposits (either cash or check) for preferred customers.
  - Preferred customers can withdraw more money than the balance from their checking, savings, or money market accounts. In this case, the money will automatically come from their consumer loan account, up to the limit established by the loan (overdraft protection).

- Non-MSB Customers are allowed to use the atm as long as they are able to provide their pin number and authenticate themselves with a Username and Password.
    - Displays a screen that mentions to input a Username and Password.
    - If the system does not recognize the Username or Password after multiple attempts, the user will not be able to use the ATM and would be prompted with a phone number to get help. public numbOfDatabase() public uAndPCheck()
- The User may also select a “forgot password” option: public forgotPassword()
    - A message will display with the user’s email and to check for recovery. public emailRecovery()
    - They will be prompted with a temporary password and the original username listed in the email. They will then be able to change either one of these options in a fixed amount of time. If the user does not change it in this amount of time, the username and password will stay the same and they will be allowed to go through the process again.
- Non-MSB Customers do not have any of the qualities listed above and are only allowed to deposit or withdraw money from their savings or checking accounts.
- If the user is done, they will have the chance to logout and return to the main menu. public sessionTerminate() public welcomeScreen()

Post-Conditions:

- The database will be logged with the information provided by the user. userAccount : Object public logSession()

- The database will also make a note that the specific user logged into the atm at that date and time for record purposes.

  public logSession()

- Depending if the user forgot the password or not, the atm will record that an attempt was made and would be logged for security purposes. public logSession()

Name: System Interface

Author: Oracle Database

Last Update: 4/6/22

Pre-Conditions: The interface will communicate with the database in order to create a verify a new user's account

Dialog:

- A user will click the welcome button that is always present on the machine when using the machine for the first time public welcomeScreen()
- The ATM will then ask the user if they are a MSB Customer with a selection of:

  isMSBCustomer : Boolean public askMSBCustomer()

  - Yes
  - No

- There is also an option that the user may be able to enter an ATM card that would automatically sign in to their account. public readCard()

- The ATM will ask the user to input the ATM card into the card slot with an animation of the card simulating the process (The animation will include the direction that the card needs to be placed into the machine) (Authentication of the user). public cardAnimation()
- If the card is not accepted or input wrong the user will be asked to try again or return to the welcome menu. pinNumber : Integer

- A customer picks no because they do not have a bank account with MSB bank. serviceCharge : Double = 3.00
- The interface then prompts for the following information from the new customer: userAccount : Object public accountCreationScreen() public accountCreation()
  - Name
  - Date of birth
  - Social security number
  - Account number
  - Phone number
  - ID
  - Password
  - Email
- All information will be sent to the data base and be held as a record for the new MSB customer public setPIN()

- The user logs into their new MSB account using their username/password. public authenticateUser()
- The interface contacts the Database to ensure this information matches the information set, which it does, and the user is now logged into the atm.
- The user can now deposit money into their brand new MSB account. The database stores the deposit as a transaction, and the balance onto their account. depBalance : Integer public getAcctBalance() public setAcctBalance()
- If the user is done, they will have the chance to logout and return to the main menu. public sessionTerminate()

Post-Conditions:

- The user session will terminate normally
- The user is also prompted with a phone number of the main database customer service.
- After a fixed amount of time the user is brought back to the main welcome screen and be logged out. timeoutTime : Integer = 120

Name: Manage Accounts

Author: System Admin

Last Update: 4/6/22

Pre-Conditions: The system admin will have access to the accounts and be able to change important information

Dialog:

- The system admin will search for the account for the specific user on the Oracle database that has submitted a request to change information. userAccount : Object
- This system admin will change the address and primary phone number for the account phoneNumb : private Double address : private String public setInfo()
- The information now on the account will be verified with the written request for a change of account information to be sure it is correct.
- An error has occurred on a deposit slip for a customer and the balance on their account needs to be corrected.
- The system admin will once again search for the account for the user on the Oracle database that has submitted this error. userAccount : Object
- According to the amount on the error submission as well as the report filled by the investigation team the new amount will be set for the balance. public setAcctBalance()
- System admin will ensure that the balance is changed on the correct account (i.e. savings checking etc.)

Post-Conditions:

- The user will now have the correct information on the account.

- The other user will now have the correct balance on their account after proper investigation has been done.
- System admin keeps a record in the database of any changes made to account information or balance. userAccount : Object

Name: Transactions

Author: User

Last Update: 4/6/22

Pre-Conditions: The user is making several transactions using cash and checks for deposit.

Dialog:

- The user will click the welcome button that is always present on the machine when using the machine for the first time. public welcomeScreen()
- The ATM will then ask the user if they are a MSB Customer with a selection of: isMSBCustomer : Boolean
    - Yes
    - No
- The customer select yes and continues with his transaction

- There is also an option that the user may be able to enter an ATM card that would automatically sign in to their account. public readCard()
    - The ATM will ask the user to input the ATM card into the card slot with an animation of the card simulating the process (The animation will include the direction that the card needs to be placed into the machine) (Authentication of the user). public cardAnimation()
    - If the card is not accepted or input wrong the user will be asked to try again or return to the welcome menu. public welcomeScreen()
- The customer wants to make a deposit to his savings account using 3 $20 bills and two checks from his place of employment public setSavBalance()
    - Could also make deposit to checking, CD, Money Market, or Mortgage public getMortBalance() public setAcctBalance() public setCertBalance() public setMonMarkBalance() public setMortBalance()
- The ATM counts of the total amount of money that will be deposited into the account and the information is updated on the interface as well as the database. depBalance : Integer public countCash() public picOfChecks()
- The customer now wants to see the balance of his savings account so he chooses get balance and then chooses his savings account. public getSavBalance()
    - Could also view balance of checking, CD, Money Market, Loan, or Mortgage public getAcctBalance() public getCertBalance() public getMonMarkBalance() public getLoanBalance() public getMortBalance()

- The ATM contacts the interface and the database to get the balance of the account which is then printed as a receipt out of the ATM for the customer to view.
- The customer now wants to make a withdrawal from his checking account.
- he only has $10 in his checking account but would like to withdraw $20 so the remaining $10 that he is missing is taken from his loan balance. withBalance : Integer public getAcctBalance() public getLoanBalance()
    - Could also withdraw from savings, money market, or the loan only public getSavBalance() public getMonMarkBalance() public getLoanBalance()
- The interface then updates both balances on the database and the user is given 1 $20 bill public gatherCash() public setLoanBalance() public setAcctBalance()
- The customer now wants to transfer $100 from their checking account to the mortgage balance as a payment. public getAcctBalance() public getMortBalance()
    - Could also transfer between several other accounts transFromOptions : [String] transToOptions : [String]
- The balance is updated on the database for the checking account as -$100 and the New Balance on the mortgage becomes $100 less as well as the next payment on the mortgage which is all stored in the database. public setSavBalance() public setMortBalance()

Post-Conditions:

- The user now has updated balance of all their accounts.
- The user now has $20 which they have taken $10 from their loan balance and another $10 from their checking account.
- The user session will terminate normally public sessionTerminate()

After a fixed amount of time the user is brought back to the main welcome screen and be logged out. timeoutTime : Integer = 120

Name: Check Balance

Author: User

Last Update: 4/6/22

Pre-Conditions: The user wants to be informed about the amounts inside of his account.

Dialog:

- The user will click the welcome button that is always present on the machine when using the machine for the first time. public welcomeScreen()
- There is also an option that the user may be able to enter an ATM card that would automatically sign into their account. public readCard()
    - The ATM will ask the user to input the ATM card into the card slot with an animation of the card simulating the process (The animation will include the direction that the card needs to be placed into the machine) (Authentication of the user). If the card is not accepted or input wrong the user will be asked to try again or return to the welcome menu. public cardAnimation() public welcomeScreen()
    - The user will then be prompted with a pin number to enter the ATM. public getPIN()

public setPIN()

- The ATM will then retrieve the customer's account userAccount : Object

- The ATM will then ask the user if they are a MSB Customer with a selection of: isMSBCustomer : Boolean
    - Yes
    - No
- The user is now able to check balances within their account and is prompted with different options to choose from: userAccount : Object
    - Checking acctBalance : private Double
    - Savings savBalance : private Double
    - Money Market monMarkBalance : private Double
    - Certificate of Deposit certBalance : private Double
    - Consumer Loan loanBalance : private Double
    - Mortgage Account. mortBalance : private Double
- After the user chooses which account they want to check the balance of, a screen will show with the user's information and the value inside of the account.
- The atm will then ask the user if they want a receipt with the total and the user is prompted with 2 answers: -wantReceipt : Boolean
    - Yes
    - No
- Once the receipt is printed the user is asked if they want to check any other balances and given the same options as shown above. public printReceipt()

- If the user is done, they will have the chance to logout and return to the main menu.

Post-Conditions:

- The session will log that the balance was checked on this specific date and which balance was checked. public logSession()
- After each iteration of a receipt being printed, the machine will test to see if there is enough materials for another printed receipt. public checkReceipt()

Name: Logout

Author: User

Last Update: 3/8/22

Pre-Conditions: The user is done with all the required tasks and wants to end the session.

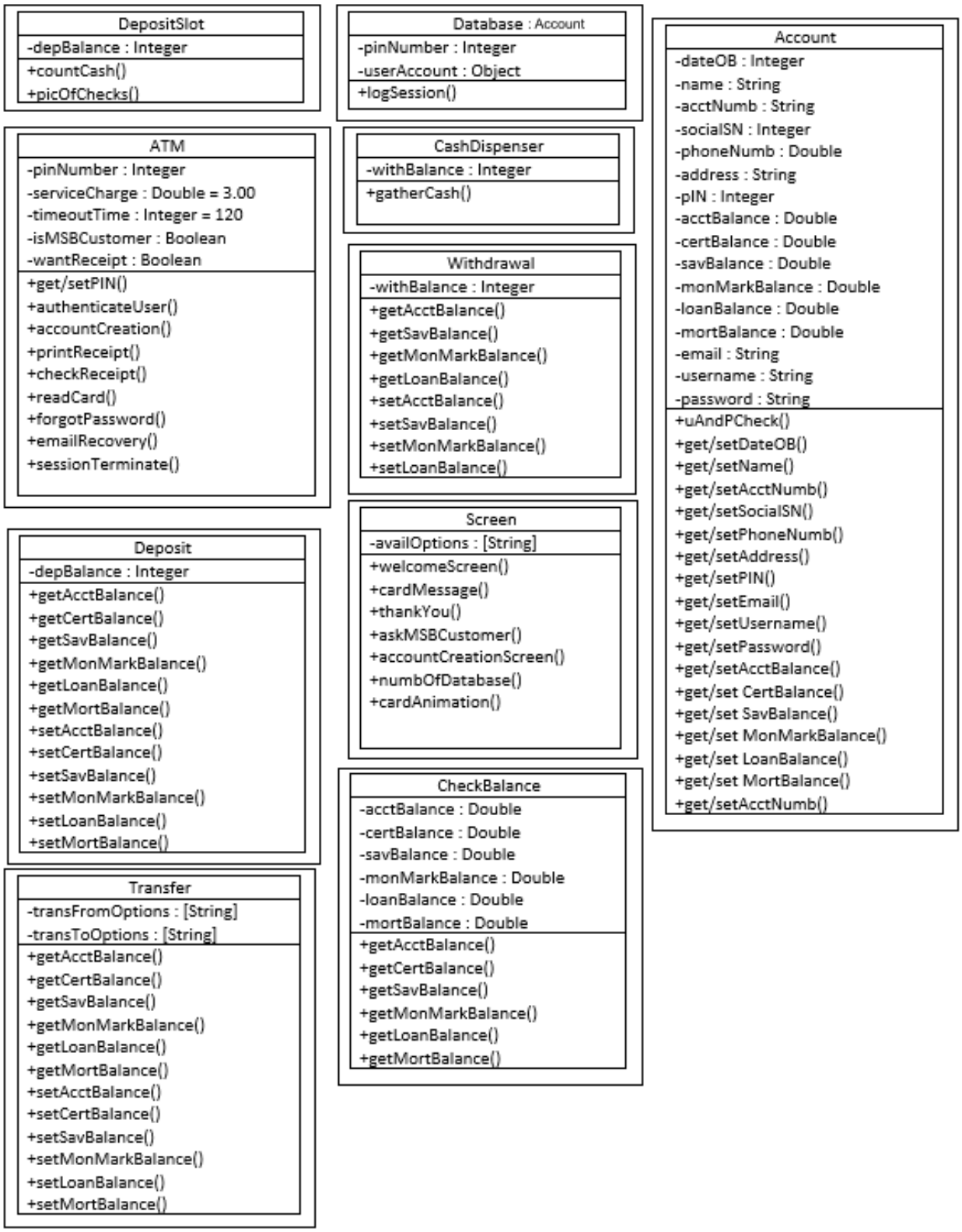
Dialog:

- The machine when in the termination stage will initiate task internally and output:
    - The ATM then outputs the card with a note that mentions “Please don’t forget to take your card!”. public cardMessage()
    - A new text box will show mentioning that the transfer was successful and thanking the user for using the ATM. public thankYou()
    - Logs the date and time that the user wanted to terminate the session
- The machine will then terminate and switch screens after a fixed amount of time.

Post-Conditions:

- The user session will terminate normally public sessionTerminate()
- The user is also prompted with a phone number of the main database customer service. public numbOfDatabase()
- After a fixed amount of time the user is brought back to the main welcome screen and logged out. public welcomeScreen() timeoutTime : Integer = 120

Class Diagram



Design

USE CASE - Logon

Communication diagram operation sequence-

1: welcomeScreen()

2: readCard()

- 2.1 cardAnimation()
- 2.2 get/setPIN()
- 2.3 return userAccount : Object

3: isMSBCustomer : Boolean

3: userAccount:Object

- 3.1 get/setName()
- 3.2 get/setDateOB()
- 3.3 get/setSocialSN()
- 3.4 get/setAcctNumb()
- 3.5 get/setPhoneNumb()
- 3.6 get/setUsername()
- 3.7 get/setPassword()
- 3.8 get/setEmail()

4: uAndPCheck()

- 4.1 numbOfDatabase()
- 4.2 forgotPassword()
- 4.3 emailRecovery()

5: sessionTerminate()

6: welcomeScreen()

7: userAccount : Object

8: logSession()

Communication diagram scenario sequence-

1, 2, 2.1, 2.2, 2.3, 5, 6, 7, 8 - User fetches account with card

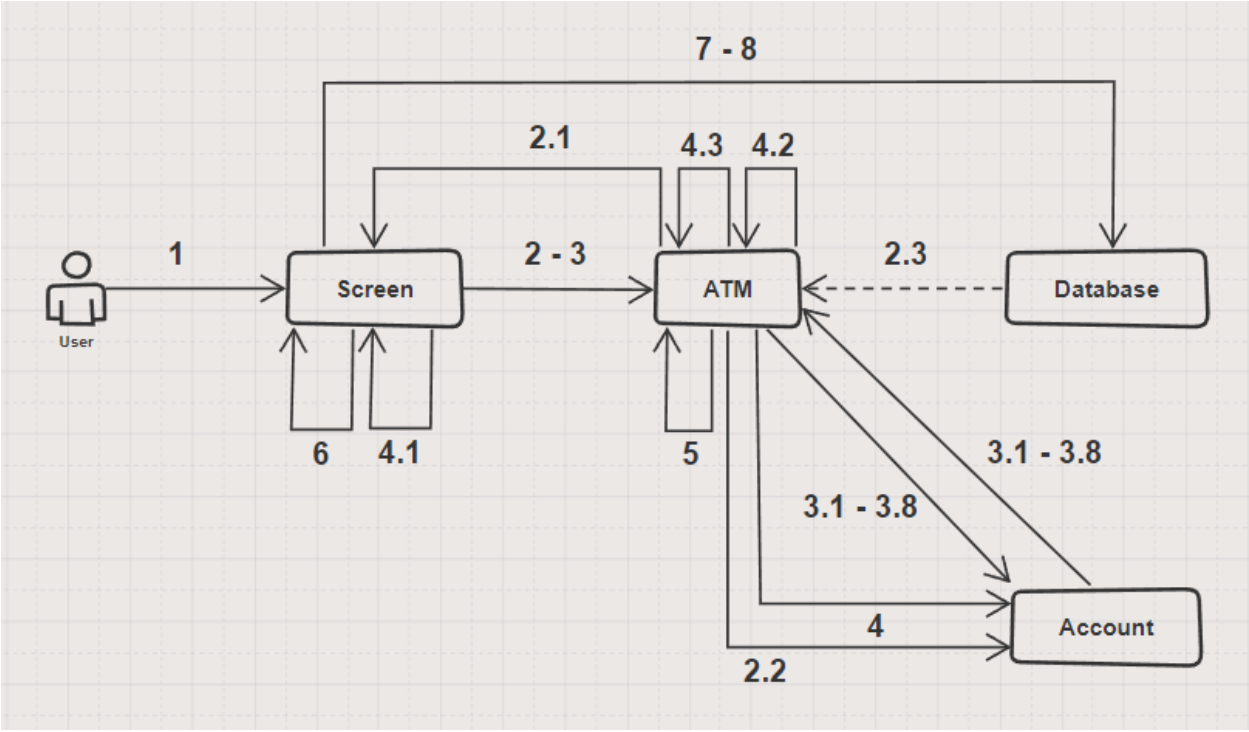
1, 3, 4, 4.1, 4.2, 4.3, 5, 6, 7, 8 - User is an MSB Customer and forgets password

1, 3, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 5, 6, 7, 8- User creates new account

1, 5, 6 - User taps the screen and does not want to use the ATM

Communication Diagram-

Communication diagram based on the LOGON Use Case Description



USE CASE - Check Balance

Communication diagram operation sequence-

1: welcomeScreen()

2: readCard()

    2.1 cardAnimation()

    2.2 get/setPIN()

    2.3 return userAccount : Object

3: isMSBCustomer : Boolean

4: userAccount : Object

    4.1: return acctBalance : Double

    4.2: return savBalance : Double

    4.3: return monMarkBalance : Double

    4.4: return certBalance : Double

    4.5: return loanBalance : Double

    4.6: return mortBalance : Double

5: wantReceipt : Boolean

    5.1: printReceipt()

6: logSession()

7: checkReceipt()

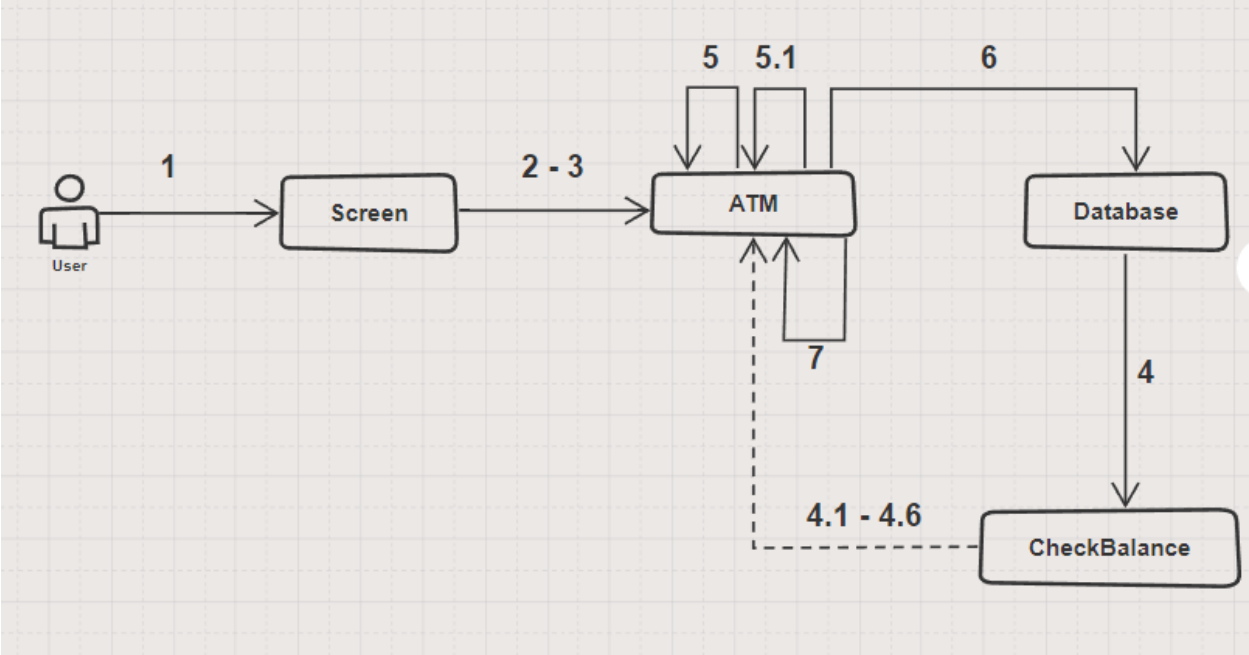
Communication diagram scenario sequence-

1, 2, 2.1, 2.2, 2.3, 4, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 5, 5.1, 6, 7 - The user chooses yes for all options besides number 3 since the user has an ATM card.

1, 3, 4, 5, 6, 7 - The user chooses yes for all options besides the printing of a receipt and logs out of the session.

Communication Diagram-

Communication diagram based on the CHECK BALANCE Use Case Description



<u>USE CASE - Logout</u>

Communication diagram operation sequence-

1. cardMessage()
2. thankYou()
3. sessionTerminate()
4. numbOfDatabase()
5. welcomeScreen()

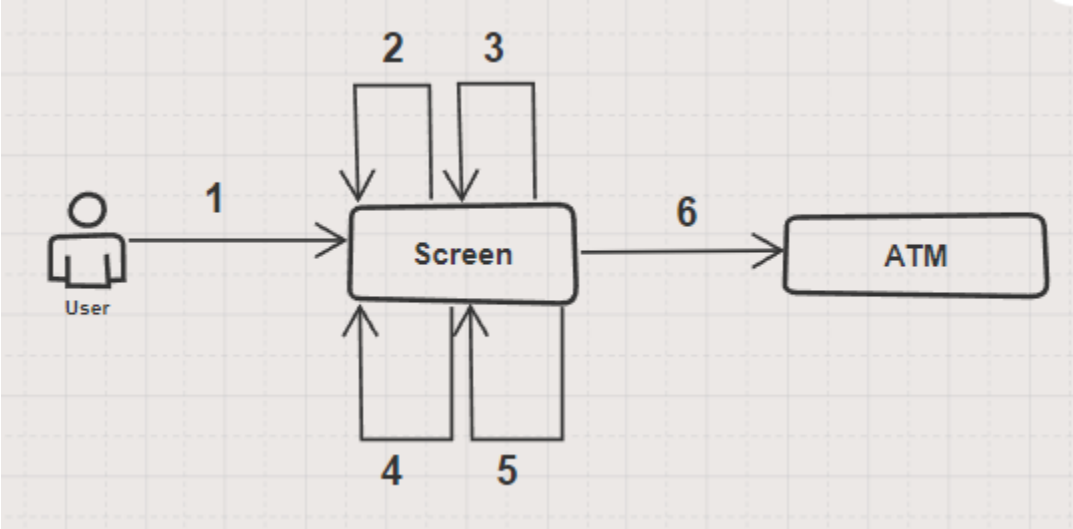
6. timeoutTime : Integer = 120

Communication diagram scenario sequence-

1, 2, 3, 4, 5, 6 - This will happen in every iteration when logging out of the system.

Communication Diagram-

Communication diagram based on the LOGOUT Use Case Description



<u>USE CASE - Manage Accounts</u>

Communication diagram operation sequence-

1. return userAccount : Object
2. public getPhoneNumb() public getAddress()
3. public setPhoneNumb() public setAddress()
4. public getAcctBalance()
5. return userAccount : Object
6. public setAcctBalance()

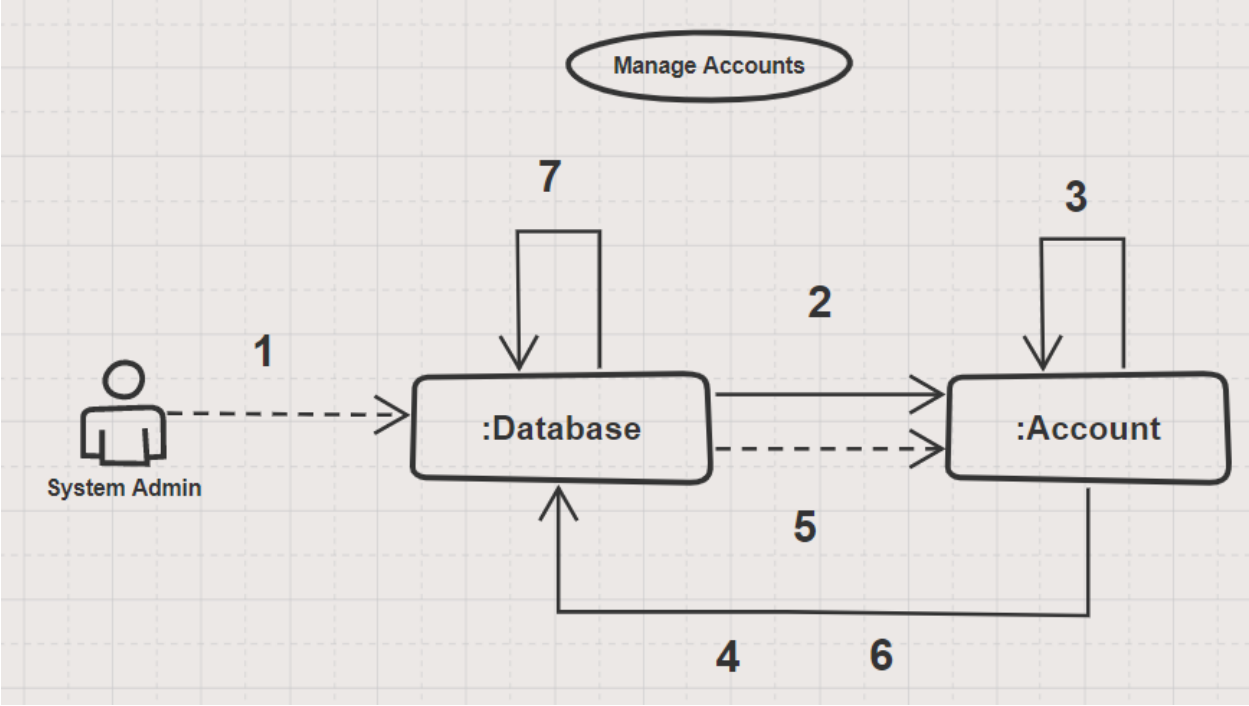
7. public get/setAcctBalance() public get/setSavBalance() public get/setCertBalance()

Communication diagram scenario sequence-

1, 2, 3 - A user has put in a request to change information on their account.
4, 5, 6, 7 - A user has requested that an error be corrected on their deposit receipt.

Communication Diagram-

Communication diagram based on the MANAGE ACCOUNTS Use Case Description



<u>USE CASE - System Interface</u>

Communication diagram operation sequence-

1. welcomeScreen()
2. readCard()

2.1 cardAnimation()

2.2 public get/setPIN()

2.3 userAccount : Object

3. isMSBCustomer : Boolean

3. userAccount:Object

3.1 get/setName()

3.2 get/setDateOB()

3.3 get/setSocialSN()

3.4 get/setAcctNumb()

3.5 get/setPhoneNumb()

3.6 get/setUsername()

3.7 get/setPassword()

3.8 get/setEmail()

4. public get/setPIN()
5. public uAndPCheck()

5.1 return userAccount : Object

6. return depBalance : Integer

6.1 public get/setAcctBalance()

7. public sessionTerminate()

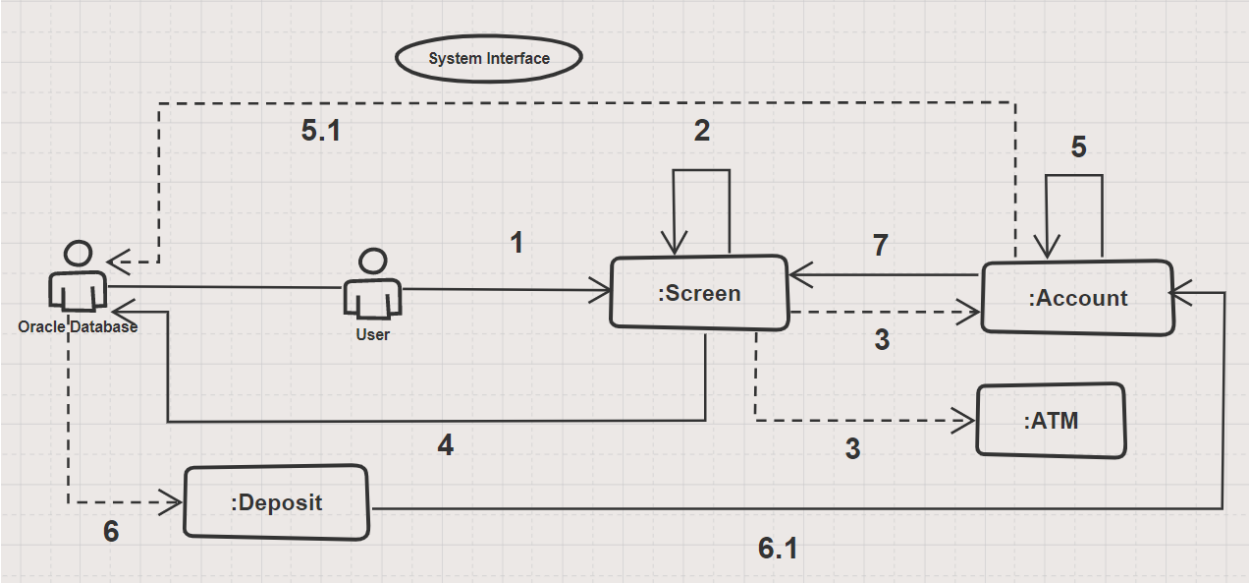
Communication diagram scenario sequence-

1, 2-2.2, 3, 5, 7 - The user uses their debit card from another bank, but creates an MSB account.

1, 3-3.8, 4, 5-5.1, 6-6.1, 7, - The user does not have a card, but creates an MSB account and deposits cash into it.

Communication Diagram-

Communication diagram based on the SYSTEM INTERFACE Use Case Description



USE CASE - Transactions

1. welcomeScreen()
2. readCard()
    2.1 cardAnimation()
    2.2 public get/setPIN()
    2.3 userAccount : Object
3. isMSBCustomer : Boolean

3. userAccount:Object
    3.1 get/setName()
    3.2 get/setDateOB()

3.3 get/setSocialSN()

3.4 get/setAcctNumb()

3.5 get/setPhoneNumb()

3.6 get/setUsername()

3.7 get/setPassword()

3.8 get/setEmail()

4. public get/setPIN()
5. public uAndPCheck()

    5.1 return userAccount : Object

6. public setSavBalance()

    6.1 public setMortBalance()

    6.2 public setAcctBalance()

    6.3 public setCertBalance()

    6.4 public setMonMarkBalance()

7. depBalance : Integer

    7.1 public countCash()

    7.2 public picOfChecks

8. public getSavBalance()

    8.1 public getAcctBalance()

    8.2 public getCertBalance()

    8.3 public getMonMarkBalance()

    8.4 public getLoanBalance()

    8.5 public getMortBalance()

9. withBalance : Integer
    9.1 public getAcctBalance()
    9.2 public getLoanBalance()
    9.3 public getSavBalance()
    9.4 public getMonMarkBalance()
10. public gatherCash()
    10.1 public setLoanBalance()
    10.2 public setAcctBalance()
11. public getAcctBalance()
    11.1 public getMortBalance()
    11.2 transFromOptions : [String]
    11.3 transToOptions : [String]
12. public setSavBalance()
    12.1 public setMortBalance()

Communication diagram scenario sequence-

1, 3-3.8, 4, 5-5.1, 6-6.4, 7-7.2 - The user creates a new MSB account with username and password and will deposit 2 checks and 3 bills into their savings account

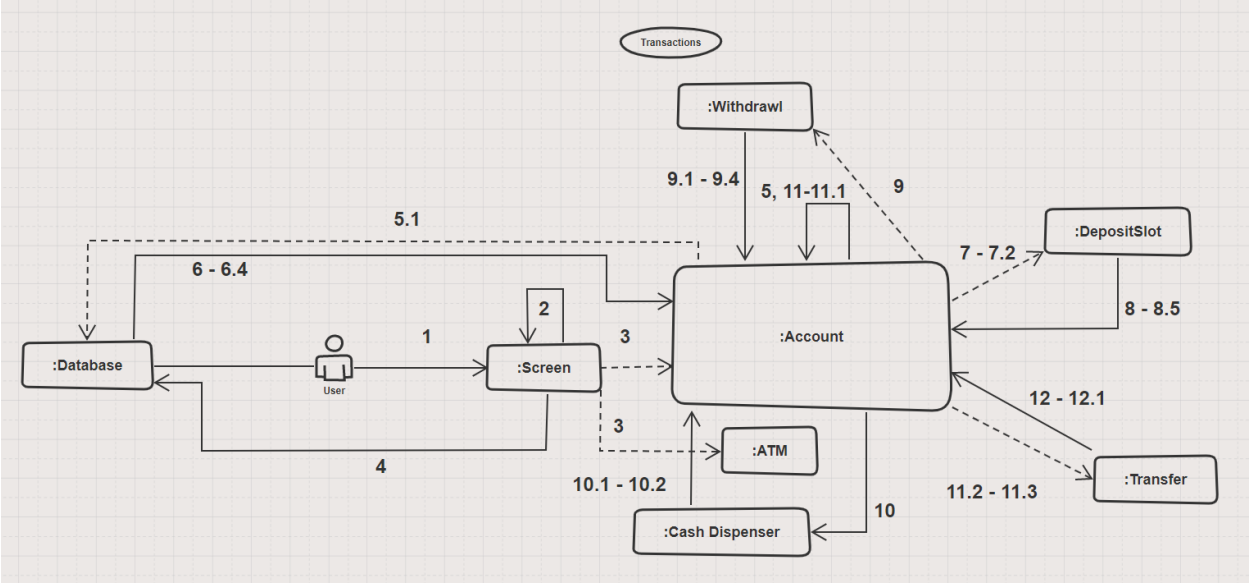
1, 2-2.3, 3, 4, 8.4, 9, 9.2, 10-10.1 - The user uses their debit card and PIN to sign into their MSB account to check the balance of their loan account and make a withdraw from that accounts

1, 2-2.3, 3, 4, 8.1, 9-9.1, 10-10.1 - The user uses their debit card and PIN to sign into their MSB account to check the balance of their checking account and make a withdraw from that account

1, 2-2.3, 3, 4, 11-11.3, 12 - The user uses their debit card and PIN to sign into their MSB account to transfer money from their checking account to savings account

Communication Diagram-

Communication diagram based on the TRANSACTIONS Use Case Description



Implementation/Code Generation

```
class DepositSlot {
        public:
                void countCash();
                void picOfChecks();
        private:
                int depBalance = 0;
};

class Database {
        public:
                void logSession(); //Logs information (time stamp and login inquiries whether successful or not)
        private:
                 int pinNumber = 0;
                Account userAccount;
};

class Account {
        public:
                 void uAndPCheck();
                 void getDateOB();
                void getName();
                void getAcctNumb();
                void getSocialSN();
                void getPhoneNumb();
                void getAddress();
                void getPIN();
```

```
void getEmail();

void getUsername();

void getPassword();

void getAcctBalance();

void getCertBalance();

void getSavBalance();

void getMonMarkBalance();

void getLoanBalance();

void getMortBalance();

void getAcctNumb();

void uAndPCheck();

void setDateOB();

void setName();

void setAcctNumb();

void setSocialSN();

void setPhoneNumb();

void setAddress();

void setPIN();

void setEmail();

void setUsername();

void setPassword();

void setAcctBalance();

void setCertBalance();

void setSavBalance();

void setMonMarkBalance();

void setLoanBalance();

void setMortBalance();

void setAcctNumb();
```

```
        private:

                int dateOB = 0, socialSN = 0, pIN = 0;

                String name = "", acctNumb = "", address = "", email = "", username = "", password = "";

                double phoneNumb = 0.0, acctBalance = 0.0, certBalance = 0.0, savBalance = 0.0,
                        monMarkBalance = 0.0, loanBalance = 0.0, mortBalance = 0.0;

};


class ATM {

        public:

                void setPIN();

                void getPIN();

                void authenticateUser();

                void accountCreation();

                void printReceipt();

                void checkReceipt();

                void readCard();

                void forgotPassword();

                void emailRecovery();

                void sessionTerminate();

        private:

                int pinNumber = 0; //checks with the same pinNumber in database

                double serviceCharge = 3.00;

                int timeoutTime = 120;

                bool isMSBCustomer = false;

                bool wantReceipt = false;

};
```

```
class CashDispenser {
        public:
                void gatherCash();//collects cash inside of ATM to give to user
        private:
                int withBalance = 0;
};

class Withdrawl {
        public:
                void getAcctBalance();
                void getSavBalance();
                void getMonMarkBalance();
                void getLoanBalance();
                void setAcctBalance();
                void setSavBalance();
                void setMonMarkBalance();
                void setLoanBalance();
        private:
                int withBalance = 0;
};

class Deposit {
        public:
                void getAcctBalance();
                void getCertBalance();
                void getSavBalance();
                void getMonMarkBalance();
                void getLoanBalance();
```

```
                void getMortBalance();
                void setAcctBalance();
                void setCertBalance();
                void setSavBalance();
                void setMonMarkBalance();
                void setLoanBalance();
                void setMortBalance();
        private:
                 int depBalance = 0;
};

class Screen {
        public:
                void welcomeScreen();
                void cardMessage();
                void thankYou();
                void askMSBCustomer();
                void accountCreationScreen();
                void numbOfDatabase();
                void cardAnimation();
        private:
                 String availOptions = "";//available options to tap and access for user
};

class Transfer {
        public:
                void getAcctBalance();
                 void getCertBalance();
```

```
                void getSavBalance();
                void getMonMarkBalance();
                void getLoanBalance();
                void getMortBalance();
                void setAcctBalance();
                void setCertBalance();
                void setSavBalance();
                void setMonMarkBalance();
                void setLoanBalance();
                void setMortBalance();
        private:
                String transFromOptions[];
                String transToOptions[];
};

class CheckBalance {
        public:
                void getAcctBalance();
                void getCertBalance();
                void getSavBalance();
                void getMonMarkBalance();
                void getLoanBalance();
                void getMortBalance();
        private:
                double acctBalance = 0.0;
                double certBalance = 0.0;
                double savBalance = 0.0;
                double monMarkBalance = 0.0;
```

```
                double loanBalance = 0.0;

                double mortBalance = 0.0;

};
```