INTRODUCTION

The goal for this project is to build an efficient, time saving ATM system. Sometimes waiting in incredibly long queues outside of a bank branch can result in a lot of time wasted. A solution to this problem can be found by implementing ATM systems.

An automated teller machine (ATM) commonly known as automated bank machines (ABM) or cash machines is an electronic banking outlet that allows customers to complete basic transactions without the aid of a branch representative or teller. Anyone with a credit card or [debit card](https://www.investopedia.com/best-debit-cards-for-teens-5079369) can access cash at most ATMs. That makes it effective in implementation and decreases the time it take to make simple transactions.

Another related system is the implementation of mobile banking applications which allow users to virtually make transactions. Mobile banking and ATM both have significantly moved the banking sector in a positive direction in the last decade.

We are building an ATM system using Java programming language and object oriented programming concepts. We hope our system proves efficient and time saving to use.

We are going to explain more about our project in the next few pages. We are going to discuss about the functionality, what our application will be doing, what is included and not. We are also going to build the UML diagram to explain more about our application.

SCOPE

The application will allow customers with accounts to make transaction using an ATM card. Therefore perform different operation we were forced to go to banks and wait in line for. Although this performs different tasks bank does there are some tasks this application will not be able to address these include opening an account or perform any kind of complaints about service. This is done by directly addressing the representative or teller of the bank.

1. The ATM will service one customer at a time.
2. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction.
3. The customer will be given a choice to choose between withdrawal and deposit.
4. Then the customer will be able to make any kind of transaction
5. The card will be retained in the machine until the customer indicates that he/she desires no further transactions or the maximum amount of transaction is reached at which point it will be returned - except as noted below.
6. The ATM must be able to provide the following services to the customer:

* A customer must be able to make a cash withdrawal from any suitable account linked to the card. Approval must be obtained from the bank before cash is dispensed.
* A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope.
* The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
* A customer must be able to make a balance inquiry of any account linked to the card.

USERS AND THEIR ROLES

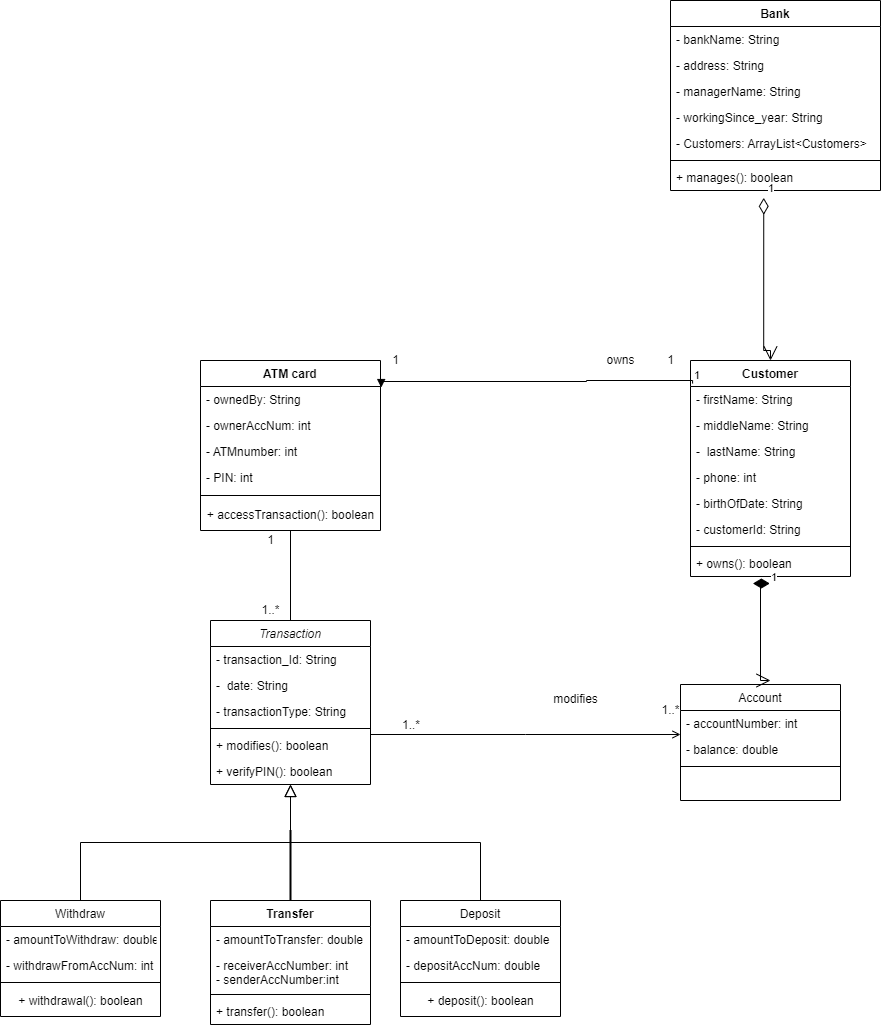
Our main user is the customer. The main goal of the customer is performing financial transactions by logging into the system. The customer will have to input his/her account number, ATM number and, also equally important, PIN (or Personal Identification Number). The system will check and verify the account and it’s PIN. And only then can the customer proceed to completing the various transactions available (withdrawing, depositing and transferring money).

FUNCTIONAL REQUIREMENT

|  |  |
| --- | --- |
| No | description |
| 1 | The ATM will consist of display, cash dispenser, cash **recycler,** card reader, buttons, a keypad and a printer. |
| 2 | The keypad will consists of 13 keys. 0 to 9 number, enter, back and delete |
| 3 | The enter key will let the user to enter if the pin is correct or ask again and reject the card when the customer is done. The back key will go to the previous page and the delete key will let the user to delete the inputs before entering to the next page. The buttons will let the user press their choices from the given alternative. |
| 4 | After the card has entered the ATM will display the place to enter the pin |
| 5 | The user will be able to enter if the pin is correct |
| 6 | The system will display options for withdraw, deposit and balance. |
| 7 | For withdraw and deposit the system will display 100, 200, 500, 1000, 2000 money amounts and other as an option. And for balance the system will display the amount of money in the users account. |
| 8 | The system will read the entered amount and compare it to the amount left in the account of the user, in the ATM machine and if the amount entered is not greater than the amount one user can ask at one time. |
| 9 | The system will add or subtract the amount entered plus the money that the bank cuts for withdraw |
| 10 | The system will then reject the card and withdraws the money entered. |
| 11 | The system will return to the home page, ready to accept another user. |

DESIGN

UML Diagram



Detailed Design Description

For the class Bank

|  |  |  |
| --- | --- | --- |
| Attribute | Visibility | Type |
| bankName | private | String |
| Address | Private | String |
| managerName | Private | String |
| workingSince\_year | Private | String |
| Customers | Private | ArrayList<Customers> |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| Manages | Public | boolean | customers: ArrayList <Customers> ,  customerId: int |

For the class Customer

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Visibility |
| firstName | String | Private |
| middleName | String | Private |
| lastName | String | Private |
| Phone | Int | Private |
| customerId | String | Private |
| birthOfDate | String | Private |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| Owns | Public | boolean | accountNumber: int |

For the class Account

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Visibility |
| accountNumber | int | Private |
| Balance | double | Private |

For the class ATM card

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Visibility |
| ownedBy | String | Private |
| ownerAccNum | int | Private |
| ATMnumber | int | Private |
| PIN | int | Private |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| accessTransaction | private | Boolean | ATMnumber : int,  ownerAccNum: int,  PIN :int |

For The Class Transaction

|  |  |  |
| --- | --- | --- |
| Attribute | type | Visbility |
| transaction\_Id | String | Private |
| Date | String | Private |
| transactionType | String | Private |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| modifies | private | boolean | transaction\_Id: String,  transactionType: String |
| verifyPIN | private | boolean | PIN: int |

For the class Withdraw

|  |  |  |
| --- | --- | --- |
| Attribute | type | Visbility |
| amountToWithdraw | double | Private |
| withdrawFromAccNum | Int | Private |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| withdrawal | private | boolean | ammountToWithdraw: double,  withdrawFromAccNum:  int |

For the class Transfer

|  |  |  |
| --- | --- | --- |
| Attribute | type | Visbility |
| amountToTransfer | double | Private |
| recieverAccNum | Int | Private |
| senderAccNum | int | Private |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| transfer | private | boolean | ammountToTransfer: double,  recieverAccNum:  int |

For the class Deposit

|  |  |  |
| --- | --- | --- |
| Attribute | type | Visbility |
| amountToDeposit | double | Private |
| depositAccNum | Int | Private |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument |
| deposit | private | boolean | ammountToDeposit: double,  depositAccNum:  int |

REFERENCES

- The Object Oriented Thought Process, book by Weistfield, Matt

- geeksforgeeks.org (article about association, composition, and aggregation)

- Mumbai University MCA programs and notes for MCA

- tutorialspoint.com/java

- guru99.com/functional-requirement-specification-example.html#3