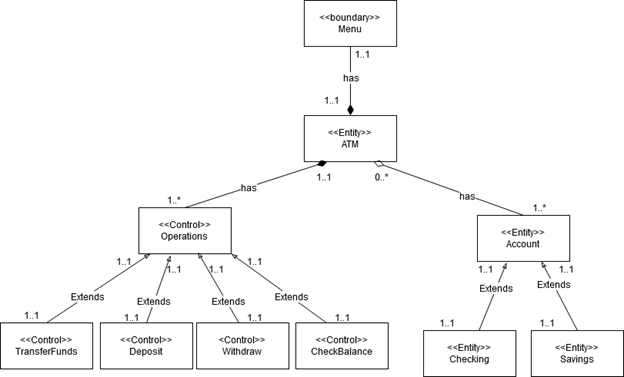
Osman Khan

Anthony Dawson

Kevin Vu

Erick Morales

## **Class Diagrams**

****

**Class Definitions**

●     **ATM:** The ATM entity object provides users with access to their accounts and the operations that manipulate the funds within those accounts. The ATM entity will also verify the PIN number with the account. An ATM can have 1 to many accounts associated with it.

●     **Account:** This is an entity class that represents a client's banking information every person has one and only one account. It will be the parent class to the checking class and savings class. Client’s pin is associated with this class.

●     **Checking:** This entity class extends the account class and stores the other way an account can hold a client’s money. As with the account class a user only has one checkings account.

●     **Savings:** This entity class extends the account class and stores one of the ways an account can hold a client’s money. As with the account class a user only has one savings account.

●     **Operation:** Control class which serves as a parent to the functions which can be performed in the ATM.

●     **Withdraw:** This control class allows the user to remove money from either their savings or checkings funds.

●     **Deposit:** This control class allows the user to insert money into either their savings or checking funds.

●     **CheckBalance:** This control class allows the user to see the amount of funds in either one of their account types. Comes from the use case diagram.

●     **TransferFunds:** This control class allows the user to transfer funds between the checking and saving account types.Comes from the use case diagram.

●     **Menu:** This boundary class displays the ATM menu. Input of the keypad, enter, clear, and cancel buttons are implemented in this class. Allows the user to enter choices and data to select operations they desire.

**Class Creation Narratives**

An ATM user will use the ATM to access and change the funds in their accounts through a pin.

●     The ATMUser is the actor in this situation.

●     The ATM entity object provides users with access to their accounts and the operations that manipulate the funds within those accounts. The ATM entity will also verify the PIN number with the account.

●     The Account entity will hold the pin number and general information about the ATM Customer, such as their name, address, and phone number.

 An ATM Customer will have a Checking and Savings account linked to their name and pin.

●     We will create two entities that inherit the Account entity, the Checking entity and Savings entity. These entities will hold the information about the funds within the accounts.

The ATM will present the user with multiple operations to choose from.

●     The Menu boundary will be created to handle the input and output between the user and ATM.

●     The Operations control object will represent each transaction operation in the ATM.

 The ATM user will be able to withdraw and deposit funds from their account.

●     A Withdraw control object will verify the account balance is sufficient for the chosen withdrawal amount and remove it from the balance.

●     A Deposit object will simply be used to add funds to the account balance

The ATM user will be able to view their different account balances and transfer funds between them.

●     A CheckBalance object will be used to present the balance amount from the user chosen account.

●     A TransferFunds object can take two accounts and move funds between them.

**Class Function Narratives**

**Account:**

The Account class will hold the users information such as name and phone. It will also hold an account number and the users PIN number.

The class will have methods to retrieve the PIN, Account Number, and an abstract method to get and set the balance.

**Checking and Savings:**

These classes will have attributes of the account balance and account names.

There is a method to retrieve the accounts balance.

**Operations:**

The Operations class creates a record of the ATMs transaction and gives the user a receipt.

**TransferFunds**:

This class will have a method that takes two accounts and moves funds between them.

**Deposit:**

This class has a method that takes the user account and adds funds dictated by the user.

**Withdraw:**

There will be a method that takes the account balance and verifies that the account has enough funds to withdraw from.

There will be a method that removes the funds after verification.

**CheckBalance:** This class will have a method that takes the account and returns its current balance.