Bank Emulation Program Documentation v1.0.0

Louden Maclay – lrm5407@psu.edu – 9 6056 3761

Isaiah Harshbarger – idh5029@psu.edu – 9 0078 6731

Lucas Overmoyer – email – ID#

Summary

The bank emulation program contains various methods, classes, and functions that in unison, imitate the functionality of a real-world bank. Users may begin by assigning person-objects, such as customer, manager, advisor, or teller, to a variable. After the assignment, an instantiation process will begin which prompts the user for different information and stores it, depending on the type of person-object being created. Each person-object has access to numerous methods; some of these take another person-object as an argument to simulate human to human interaction within a bank, and others simulate simpler actions such as introducing one’s self or viewing a list of customers. However, not every person-object has access to every method. For example, only the manager-object has direct access to methods capable of viewing sensitive information or making significant changes. In short, this program provides functionalities, from depositing money to investing in cryptocurrencies, which accurately mimics those observed in a bank.

Visual Representation of Classes

Person Class:

The Person class contains an initialization function which prompts the user for their first name, middle name, last name, birthdate, and address, with checks inserted to verify valid entries. It also produces and assigns a fake, randomly-generated social security number for the person and stores it, along with the

It contains two class attributes (customer\_list and past\_customer\_list) which are utilized by subclasses to keep updated lists of active and inactive customers.

**def talk(self):**

* Abstract class method, raises a NotImplementedError when talk hasn’t been explicitly defined in subclasses

**def make\_ssn():** (not a method of the person class but is called in its initialization)

* Uses the random python package to produce a fake social security number
* Returns a string in the form: ‘000-00-0000’

Customer Class:

The Customer class inherits from the Person class and uses the init function from Person in its own init function. In addition, it initializes an account number, routing number, balance, customer number, and investment account balances. It then increments the class variable customer\_number by one, sets up the account number for the next instance created, appends itself to the Person’s class variable customer\_list, sets \_\_deleted\_customer to false, prompts the user to create a 4-digit pin, and prompts the user for their initial deposit to their checking account.

**def \_\_mfAccountStatus(self):**

* Prints a message on the screen telling the user their mutual fund account balance, a reasonable, projected return rate, and their prospective balance after a year based on this return rate.

**def \_\_mgfAccountStatus(self):**

* Prints a message on the screen telling the user their precious metal and gem fund account balance, a reasonable, projected return rate, and their prospective balance after a year based on this return rate.

**def \_\_cryptoAccountStatus(self):**

* Prints a message on the screen telling the user their cryptocurrency account balance, a reasonable, projected return rate, and their prospective balance after a year based on this return rate.

**def \_\_str\_\_(self):**

* Overloads the string representation of a customer object to display their personal information in string format, displaying the customer’s name, date of birth, address, account and routing numbers, checking account balance, and net worth (checking account + investment accounts balances added together)

**def \_\_deposit(self):**

* Prompts user for an amount to deposit to their checking account
* Prints the updated balance once the deposit is complete

**def \_\_withdrawal(self):**

* Calls \_\_check\_balance to display balance
* Prompts user for amount to withdrawal
* Prints the updated balance once the withdrawal is complete

**def \_\_check\_balance(self):**

* Prints the customer’s checking account balance

**def \_\_CheckOutstandingBalance(self):**

* Prints the customer’s outstanding balance with the bank

**def atm(self):**

* Checks to make sure the customer accessing the ATM is a current customer, if not, prints that only active customers can use the ATM
* Prompts customer for their pin, if entered incorrectly 4 times prints ‘too many failed attempts’ and the customer must use ATM method again to access it
* If access is granted, prompts the customer to choose (1)balance inquiry, (2)deposit, (3)withdrawal, or (4)exit.
  1. Calls \_\_check\_balance() then prompts user if they would like another transaction. Returns to menu if yes, exits if no.
  2. Calls \_\_deposit() then prompts user if they would like another transaction. Returns to menu if yes, exits if no.
  3. Asks the user if he/she will accept the $3 surcharge for ATM withdrawals (skips next step if no), if yes subtracts 3 from balance and calls \_\_withdrawal() then prompts user if they would like another transaction. Returns to menu if yes, exits if no.
  4. Prints ‘Goodbye!’ and exits

**def talk(self):**

* Prints: “Hello! I’m {customer’s first name}”

Employee Class:

The employee class inherits from the person class and uses its initialization. This class provides some basic employee functions and is the parent class for each of the three types of employees.

**def ShowCustomerInfo(self, other):**

* If the customer is active (not a deleted customer), then this will print the customer’s name, date of birth, address, account and routing numbers, and their checking account balance.
* If the customer has been deleted by a manager, an inactive customer message is printed and no customer information is displayed.

**def DeleteCustomer(self, other):**

* When called by a non-manager, this function informs the employee who called it that they do not have the ability to perform this action.

**def SeeCustomers(self):**

* When called by a non-manager, this will just print just the number of active customers.

**def talk(self):**

* Employee introduces themselves, but recommends anybody looking for help to ask a specialized employee.

Advisor Class:

The advisor class inherits from the employee class and employs the same initialization process. A financial advisor can help a customer take out a loan or get involved with any of the three offered investment opportunities.

**def \_\_offer\_loan(self, other):**

* Prompts the user for how much the customer would like to borrow. After ensuring valid input, the user enters for how long they’d like the loan term to be. Then, based on the customer’s wealth relative to the loan amount and the loan term length, an interest rate is generated using the make\_interest\_rate function.
* The proposed loan and attached interest rate are presented to the customer, who can either choose to accept or decline the loan. If accepted, the total interest to be paid over the course of the loan term is added to the customer’s outstanding balance with the bank.

**def make\_interest\_rate(amount, net\_worth, period): (not an advisor class method but is used in choosing the interest rate on a proposed loan)**

* Returns the interest rate (Ex: 4.123) that will be offered to a customer based on the size of the proposed loan, their net worth, and the loan term length.
* The more risk the bank is perceived to incur (the greater the likelihood of the loan not being paid back), the greater the interest rate, for this bank. So, the greater the ratio of loan amount to customer’s net worth (checking account balance + all investment account balances), the greater the interest rate. Then, the greater the net worth, the less the interest rate. The longer the period, the less break a customer receives on their interest rate. This function combines these elements to yield an appropriate interest rate.

**def \_\_open\_investment\_account(self, other):**

* A customer must have $1,000 in checking to start investing, so if one does not, he/she is told to acquire more funds.
* If able, the customer can learn more about any of the investment options: mutual fund, precious metal & gem fund, or cryptocurrencies; or they can exit the investment menu.
* The customer is informed that from mutual fund, to precious metal & gem fund, to cryptocurrency, the safety of the investment decreases, but the potential reward increases.
* The customer can invest in any or all options and is asked to verify their investment after entering the amount but before finalizing. The amount invested is deducted from the customer’s checking account, provided the amount is there, and added to their respective investment account(s).

**def GiveAdvice(self, other):**

* This method is used for an advisor to give a customer the choice between discussing/taking out a loan and making/checking investments. The options are 1) Discuss a loan 2) Open/Add to investment account 3) Check performance of investment portfolio(s) 4) Exit the advising session.
* If option 3 is selected, the customer is either told they have not yet made any investments or shown the status of their investment(s). This will display the amount currently in the investment account, the estimated return rate of the investment, and the balance expected after one year based on the projected return rate.

**def talk(self):**

* The financial advisor introduces him/herself and says they are a financial advisor at I.L.L & Sons.

Manager Class:

The manager class inherits from the employee class and uses the same initialization process. A manager can perform more administrative tasks that have to do with bank management and oversight. Directly accessing a customer’s Social Security Number, deleting/reactivating customers, viewing a complete customer list, and seeing the total debts to the bank are things only managers can do.

**def ShowCustomerInfo(self, other):**

* If the referenced customer is inactive, the manager is informed of this.
* Otherwise, the customer’s name, birthdate, address, SSN, acct. #, routing #, checking acct. balance, and net worth are printed.

**def SeeCustomers(self):**

* This will print the name and account number of every active customer of the bank.

**def DeleteCustomer(self, other):**

* This “deletes” a customer by moving them from the current customer list to the past customer list. This disables a customer’s ability to use an ATM and get service from a teller or advice from an advisor, effectively deactivating the account.

**def ReactivateCustomer(self, other):**

* Restores a past customer’s full functionality as a customer with I.L.L & Sons by removing them from person class’s past customer list and adding them back to the active customer list.
* After reactivation, customers can again access the ATM, advisor, teller, etc.

**def SeeDebtsToBank(self):**

* This will print a statement that indicates the total of the outstanding balances owed to the bank (the total amount loaned out plus interest) and how many customers there are with outstanding balances.

**def talk(self):**

* The manager introduces him/herself by name and as a manager at I.L.L & Sons.

Teller Class:

The teller class inherits from the employee class and uses the same initialization process. A teller can help a customer perform actions he/she wouldn’t directly be able to perform themselves but will not give advice or be able to help with their financial planning. Tellers recognize customers by face, so they require no authentication.

**def \_\_TransferFromInvestmentAccount(self, other):**

* If the customer has made no investments, then a message will be printed telling them so.
* Otherwise, for each of their investments, they will be prompted to accept or decline a 5% fee for transfers from an investment account to their checking account. If they accept the fee, they can enter a valid amount to transfer from that investment account to their checking account. 95% of the entered amount will move to their checking account and the rest is paid to the bank.

**def \_\_PaymentOnOutstandingBalance(self, other):**

* If the customer has no outstanding balance, they are congratulated on their financial responsibility. If they have no funds to pay off their outstanding balance, they are told to return when they can make a payment.
* Otherwise, the customer is prompted to enter an amount to pay off their outstanding balance. This amount is deducted from the customer’s checking account and put towards paying off their outstanding balance. Once a valid payment is entered, the customer’s updated checking account and outstanding balances are printed.

**def talk(self):**

* The teller introduces him/herself by name and as a teller at I.L.L & Sons.