ITCS 1140 C0801

Professor Faisal Shakeel

Jacob Lee & Mohammed Shatit

ATM Project

Design Document

Here is our design plan and document for the semester project of the ATM by using classes:

1. Display a welcome message
2. Create a class called ATM
   1. Define the constructor function (init) that has the parameter userPin (a variable to get the user input for the pin of the ATM)
      1. Set userPin = userPin
      2. Declare self.pin (as a string) = “1234”
      3. Declare self.savings = 1000
      4. Declare self.checking = 1000
      5. Declare the list self.descriptions = []
      6. Declare the list self.amounts = []
   2. Define the method runATM with no parameters
      1. Use the method printMenu to display the menu
      2. Ask user for the choice (userChoice = int(input()))  
         While userChoice != 6
         1. If userChoice = 1

Call getBalance method

* + - 1. If userChoice = 2

Call deposit method

* + - 1. If userChoice = 3

Call withdraw method

* + - 1. If userChoice = 4

Call transfer method

* + - 1. If userChoice = 5

Call printHistory method

* + - 1. Else

Display a message showing that the user didn’t choose any of the six choices

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| userChoice | Determine the number the user entered | Call a function depending on the number entered |

* + - 1. Call printMenu method again

Ask for the user choice again

1. Define a method to check the pin the user entered called checkPin
   1. Set OkToRun = False
   2. Set numGuess = 0
   3. If self.userPin = self.pin
      1. Display (Correct pin, thank you!)  
          Set OkToRun = True
   4. Else
      1. Set numGuess = numGuess + 1
      2. While numGuess < 3
         1. Ask the user to enter the pin again into the variable self.userPin
            1. If self.userPin = self.pin

Display (Correct pin, thank you!)  
 Set OkToRun = True

Enter the keyword break to exit the loop once the user enter the correct pin

* + - * 1. Else

Set numGuess = numGuess + 1

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| userPin | Determine if the pin entered is correct or not | Return OkToRun to enter the next method of the program |

1. Define the printMenu method
   1. Display:

1: Balance Inquiry

2: Deposit Funds

3: Withdrawal Funds

4: Transfer Funds

5: History of Last 5 Transactions

6: Exit

1. Define getBalance method to show the balance with no parameters
   1. Display self.checking
   2. Display self.savings
   3. Add balance inquiry to the descriptions list
   4. Add “” to the amounts list

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| No Input | No Process | Display self.savings  Display self.checking |

1. Define deposit method to add money to the accounts
   1. Display:

1: Checking

2: Saving

3: Back

* 1. Ask the user for his or her choice into the variable userChoice
  2. If userChoice == 1
     1. Ask the user for the amount to deposit and store it in the variable userAmount
     2. Set self.checking += userAmount
     3. Add the string (Deposit to checking account) to the descriptions list
     4. Add userAmount to the amounts list
  3. If userChoice == 2
     1. Ask the user for the amount to deposit and store it in the variable userAmount
     2. Set self.savings += userAmount
     3. Add the string (Deposit to savings account) to the descriptions list
     4. Add userAmount to the amounts list
  4. If userChoice == 3
     1. Display an empty string and the program will be redirected to the main menu automatically
  5. Else
     1. Display a message to show that the user didn’t choose one of the three options above

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| userChoice  userAmount | self.checking += userAmount  self.savings += userAmount | No Output |

1. Define withdraw method no parameters
   1. Set flag = False
   2. Display:

1: Checking

2: Saving

3: Back

* 1. Ask the user for his or her choice into the variable userChoice
  2. If userChoice == 1
     1. Ask the user for the amount to withdraw and store it in the variable userAmount
     2. While flag = False
        1. If userAmount <= self.checking
           1. If (userAmount % 10) != 0

Ask the user to enter the amount in denominations of 10

* + - * 1. Else

Set self.checking -= userAmount

Add the string (withdrawal from checking account) to the descriptions list

Add userAmount to the amounts list

Set flag = True

Break to end the while loop

* + - 1. Else

Display a message that shows the user don’t have enough to withdraw in the account

* 1. If userChoice == 2
     1. Ask the user for the amount to withdraw and store it in the variable userAmount
     2. While flag = False
        1. If userAmount <= self.savings
           1. If (userAmount % 10) != 0

Ask the user to enter the amount in denominations of 10

* + - * 1. Else

Set self.savings -= userAmount

Add the string (withdrawal from savings account) to the descriptions list

Add userAmount to the amounts list

Set flag = True

Break to end the while loop

* + - 1. Else

Display a message that shows the user don’t have enough to withdraw in the account

* 1. If userChoice == 3

Display an empty string

* 1. Else

Display (Invalid Entry) to show that the user didn’t choose one of the three options

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| userChoice  userAmount | self.checking -= userAmount  self.savings -= userAmount  (userAmount % 10) != 0 | No Output |

1. Define transfer method with no parameters
   1. Set flag = False
   2. Display

1: Transfer to checking account

2: Transfer to savings account

3: Back

* 1. Ask the user for his or her choice into the variable userChoice
  2. If userChoice == 1
     1. Ask the user for the amount to transfer and store it in the variable userAmount
     2. While flag = False
        1. If userAmount <= self.savings
           1. Add the string (Transfer To Checking account) to the descriptions list
           2. Add userAmount to the amounts list
           3. Set self.checking += userAmount
           4. Set self.savings -= userAmount
           5. Set flag = True
           6. Break to end the while loop
        2. Else
           1. Ask the user to enter less amount and store it into the userAmount variable
  3. If userChoice == 2
     1. Ask the user for the amount to transfer and store it in the variable userAmount
     2. While flag = False
        1. If userAmount <= self.checking
           1. Add the string (Transfer To savings account) to the descriptions list
           2. Add userAmount to the amounts list
           3. Set self.savings += userAmount
           4. Set self.checking -= userAmount
           5. Set flag = True
           6. Break to end the while loop
        2. Else
           1. Ask the user to enter less amount and store it into the userAmount variable
  4. If userChoice == 3

Display an empty string

* 1. Else

Display (invalid entry)

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| userChoice  userAmount | self.checking -= userAmount  self.savings += userAmount | No Output |

1. Define printHistory method
   1. Set counter = 1
   2. If len(self.descriptions) > 4
      1. For i in range (len(self.descriptions)-1, len(self.descriptions)-6, -1)
         1. Display (counter, self.descriptions[i](index), self.amounts[i])
         2. Set counter = counter +1

|  |  |  |
| --- | --- | --- |
| IPO Chart | | |
| Input | Process | Output |
| No Input | len(self.descriptions) > 4 | Display the last five transactions |

1. Define main function
   1. Ask the user for the pin and store it in the variable userPin
   2. Create an instance of the class ATM (ATM1 = ATM(userPin))
   3. Using checkPin method isPinOk = ATM1.checkPin
   4. If isPinOK == True
      1. ATM1.runATM() (access the runATM method)
   5. Else

Display a friendly message to the user to try at a different time

1. Call the main function
2. main()