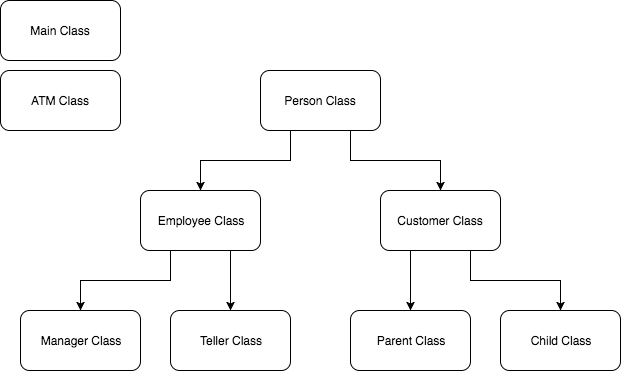
Members:

* Dylan Kane; [dqk5384@psu.edu](mailto:dqk5384@psu.edu); 983385531
* Eric Bray; ewb5319;

Our Code:

* Brief Description of Purposes and Abilities
* Five principles of OOP in our project
* Classes
  + Classes are a fundamental part of our code. Each main portion of our banking system was split up into a class, one for each of the following: ATM, Child, Customer, Employee, Main, Manager, Parent, Person, Teller
* Encapsulation
  + Encapsulation, as the process of hiding some details from the user, was used in many places throughout our project. An example would be how we used it to declare private variables in the customer classes to help hide sensitive information like account balances and pin numbers, making the only way to see them be through an account number protected \_\_str\_\_ method, or through getter methods that are only accessible by managers through login protection.
* Data Abstraction
  + Since data abstraction is achieved through polymorphism, many instances of this principle are the same as our use of encapsulation, including the private customer variable mentioned above. Another example would be our total customer count, which although is useful information, it is hidden from the user.
* Inheritance
  + Inheritance is used in most of the classes, many of which stem from the “Person” class. A hierarchical chart of this is shown later on in this paper.
* Polymorphism
  + We used polymorphism in a few places in the project. One is the Employee class, which acts as somewhat of an interface for its constituents being the Managers and Tellers. Methods are defined that the Manager and Teller class must implement with their own code or an exception will be raised. Polymorphism is also used in the “Person” class, with the “accessCustomerSupport” method. This method must be redefined in each child class if the permissions must change, being a customer can log on the get help, and a manager can log on the give help.



Documentation

* General Guide
  + Enter test cases into the “Main” function in the “Main” class. If you want to run the project from terminal make sure you are using python3 and running the Main.py file like so: python3 Main.py or python3 –i Main.py
* ATM
  + Usage Example:

>>>dylan = Customer("Dylan", 19, 7245498027, "MALE", 123456789, 10000, 2000)

>>>ATM(dylan)

Would you like to use the ATM, Dylan (y) (n):

* + Methods
    - def \_\_init\_\_(self, customer):
      * takes customer as parameter to access their account
      * automatically runs “interface” method on instantiation
    - def deposit(self, amount, pin):
      * automatically used by “interface” method when user wants to deposit money
    - def withdraw(self, amount, pin):
      * automatically used by “interface” method when user wants to withdraw money
    - def interface(self):
      * automatically used by “interface” method when ATM class is intantiated. Interface method enables user to interact with ATM through terminal to deposit and withdraw money
* Child
  + Methods
    - def \_\_init\_\_(self, name, age, phone, sex, accountnum, savbalance, checbalance):
    - def amountOfChildren(cls):
    - def deposit(self, amount):
    - def withdrawal(self, accountnum, amount):
    - def atmDeposit(self, amount, pin):
    - def atmWithdraw(self, amount, pin):
    - def checkSavBalance(self):
    - def checkChecBalance(self):
    - def \_\_str\_\_(self):
    - def \_\_del\_\_(self):
* Customer
  + Usage Example:

>>>dylan = Customer("Dylan", 19, 724, "MALE", 123, 10000, 2000)

Dylan’s assigned PIN Number: 8150

>>>Customer.amountOfCustomer

1

>>>dylan.accessCustomerSupport()

'Login: USER | Web address: [www.example.com](http://www.example.com)'

>>>dylan.deposit(1000)

\*\*\*\*\*ACTION COMPLETE\*\*\*\*\*

>>>dylan.withdrawal(123, 300)

\*\*\*\*\*ACTION COMPLETE\*\*\*\*\*

300

>>> dylanOther = Customer("Dylanother", 19, 724, "MALE", 123, 10000, 2000)

Dylanother’s assigned PIN Number: 4183

>>>dylan.transferFunds(100, dylanOther)

\*\*\*\*\*ACTION COMPLETE\*\*\*\*\*

\*\*\*\*\*ACTION COMPLETE\*\*\*\*\*

>>>dylan.checkSavBalance()

Enter the account number: 123

Dylan has $10000

>>>dylan.checkChecBalance()

Enter the account number: 123

Dylan has $2600

>>>print(dylan)

Enter the account number: 123

---Account Info---

Name: Dylan

Savings Account: 10000

Checking Account: 2600

-----------------------

>>>del dylan

* + Methods
    - def \_\_init\_\_(self, name, age, phone, sex, accountnum, savbalance, checbalance):
      * increases “customer count” variable by 1
      * established private variables
      * generates random pin number and displays it for user only once
    - def amountOfCustomer(cls):
      * classmethod
      * returns how many customer there are
    - def accessCustomerSupport(self):
      * returns a string including login information and website for customer to get customer support on
    - def accessCustomerInfoManager(self, manager):
      * This function will not be called by a customer, it is the function for a manager to view customer private variables, called by manager class method “seeCustomerDetails”, use of the function will be documented there.
    - def accessCustomerInfoTeller(self):
      * This function will not be called by a customer, it is the function for a teller to view customer variables, called by teller class method “seeCustomerDetails”, use of the function will be documented there. This will display less information compared to the “accessCustomerInfoManager” function
    - def deposit(self, amount):
      * This function will add the money given as the parameter “amount” to the customer checking balance
    - def withdrawal(self, accountnum, amount):
      * This function will subtract money given as the parameter “amount” form the customer checking balance, but will also need the account number as “accountnum” to verify the account.
    - def transferFunds(self, amount, account):
      * called through one customer with a second customer as an argument as to give that second customer money
    - def atmDeposit(self, amount, pin):
      * method used by the ATM class to deposit money into the account
    - def atmWithdraw(self, amount, pin):
      * method used by the ATM class to withdraw money from the account
    - def checkSavBalance(self):
      * displays just the savings balance, must enter account number
    - def checkChecBalance(self):
      * displays just the checkings balance, must enter account number
    - def \_\_str\_\_(self):
      * displays all customer private information except for pin number, must provide account number
    - def \_\_del\_\_(self):
      * deletes customer and subtracts 1 from customer count variable
* Employee
  + Methods
    - def \_\_init\_\_(self, name, age, phone, sex, experience):
    - def \_\_str\_\_(self):
    - def paygrade(self):
    - def getEmployeeNumber(self):
    - def employeeCount(cls):
    - def \_\_del\_\_(self):
    - def changePassword(self):
    - def accessCustomerSupport(self):
    - def setPaygrade(self, manager, paymentIncrease):
    - def delCustomerAccount(self, customer):
    - def seeCustomerDetails(self, customer):
* Main
  + Methods
    - def main():
* Manager
  + Methods
    - def \_\_init\_\_(self, name, age, phone, sex, experience, username, password):
    - def \_\_str\_\_(self):
    - def givePromotion(self, employee, paymentIncrease):
    - def isManager(self):
    - def setPaygrade(self, manager, paymentIncrease):
    - def delCustomerAccount(self, customer):
    - def hireEmployee(self):
    - def seeCustomerDetails(self, customer):
* Parent
  + Methods
    - def \_\_init\_\_(self, name, age, phone, sex, accountnum, savbalance, checbalance):
    - def amountOfParents(cls):
    - def deposit(self, amount):
    - def atmDeposit(self, amount, pin):
    - def atmWithdraw(self, amount, pin):
    - def withdrawal(self, accountnum, amount):
    - def childDeposit(self, amount, child):
    - def childCheckSavings(self, child):
    - def childCheckChec(self, child):
    - def checkSavBalance(self):
    - def checkChecBalance(self):
    - def \_\_str\_\_(self):
    - def \_\_del\_\_(self):
* Person
  + Methods
    - def \_\_init\_\_(self, name, age, phone, sex):
    - def accessCustomerSupport(self):
* Teller
  + Methods
    - def \_\_init\_\_(self, name, age, phone, sex, paygrade, position, experience, username, password, manager):
    - def \_\_str\_\_(self):
    - def setPaygrade(self, manager, paymentIncrease):
    - def delCustomerAccount(self, customer):
    - def seeCustomerDetails(self, customer):